Speech and Language Processing

An Introduction to Natural Language Processing,
Computational Linguistics, and Speech Recognition
Third Edition draft

Daniel Jurafsky
Stanford University

James H. Martin
University of Colorado at Boulder
2020.12.30

语 音 和 语 言 处 理

自然语言处理、计算语言学与语音识别

第三版草稿

悦乐君 译

2021.06.23

译者序

科技之巅，人工智能。语言处理，智能明珠。[01]

登堂入殿，一课一本。君面本书，圣经称著。[02]

斯坦福丹尼尔，计语界学术帝。外则见取于人，内则思行于己。

远则观法于科，近则察术于技。以通词句之情，以达话语之义。

丹氏创巨著，二十又六章。博而得其要，简而周其全。计语之总会，算法之渊林。

译者编微言，每章撰一行。八三四一保，十面埋伏援。事半而功倍，举重而若轻。[03]

八者，目也：

语言模型者，多元语法，泛化平滑，困惑度熵，预测是也。（ch03）

朴素贝氏者，条件独立，情感分析，精召评估，生成性也。（ch04）[04]

逻辑回归者，叉熵梯降，正则多项，解释模型，区分性也。（ch05）[05]

向量语义者，词反文频，逐点互信，单词向量，静嵌是也。（ch06）[06]

计算语义者，量辖歧义，合一分析，线图算法，语析是也。（ch16）[07]

词义词网者，词汇关系，词义消歧，词义归纳，排岐是也。（ch18）

共指消解者，话语模型，提及检测，算法框架，共指链也。（ch21）

话语连贯者，修辞树库，实体网格，部整连贯，句构是也。（ch22）[08]

三者，纲也：

神网语模者，神经网络，正反计算，语境嵌入，训练是也。（ch07）[09]

深度学习者，三循模型，双向多层，自注变换，核构是也。（ch09）[10]

语境嵌入者，语模嵌入，双编表变，生成模型，动嵌是也。（ch10）[11]

四者，用也：

机器翻译者，注力变压，波束搜索，双语评估，编解码也。（ch11）[12]

信息提取者，关系提取，时事提取，模板填充，数构是也。（ch17）[13]

问题解答者，信检基知，超大预模，分类问答，抽取是也。（ch23）[14]

机聊对话者，规库混聊，对话框架，信状架构，交互是也。（ch24）[15]

一者，导也：

引导绪言者，梳理历史，归类内容，构建体系，综述是也。（ch01）

十者，基也：

正则表达者，模式匹配，文本规范，编辑距离，规范是也。（ch02）

词类标签者，词类名实，隐马维特，条件随机，标签是也。（ch08）[16]

成分语法者，语境无关，规则树库，乔组双范，嵌套是也。（ch12）[17]

成分解析者，科凯阳法，跨度计分，组范解析，句配树也。（ch13）[18]

依存解析者，点线投射，转移预言，图论选优，词互依也。（ch14）[19]

句子含义者，一阶逻辑，事态时体，描述逻辑，表示是也。（ch15）

语义角色者，主谓双角，题库框网，选择限制，动名签也。（ch19）[20]

情感内涵者，感情定义，情感词汇，内涵框架，识别是也。（ch20）

语音信号者，发声韵律，波频幅谱，语音资源，波形是也。（ch25）[21]

语音处理者，语音自识，连时分类，文本转音，音字互转也。（ch26）[22]

此书译者，悦乐君也，非以徇名，将以济人。译文既成，恭疏短序，陆海潘江，赐教勿吝。[23]

悦乐君

2021.06.23

注释：

[01] 智能明珠：语音和语言处理，被誉为“人工智能皇冠上的明珠”（相当于昔在观音“耳根圆通”之殊胜法门）。

[02] 一课是指斯坦福大学曼宁教授的 CS224N《NLP with DL》课程（与李飞飞的 CS231N《CNN for VR》课是绝配）。
一本是指斯坦福大学丹尼尔教授的《Speech and Language Processing》教程（即本书，此书堪称业界“圣经”）。

[03] 八三四一：原指“8341 神秘部队”的番号，“8341”之数是《周易》64 卦中的“鼎”卦，有问鼎、鼎盛之意；
此处专指“八目、三纲、四个应用，一个导向”；保：核心保卫力量。
十面埋伏：原是成语，出自楚汉项刘两军相争的垓下之战；此处专指“十大基础内容”。援：外部支援力量。
把 26 章内容分成 8、3、4、1、10 五个大类，每章内容复述成一行文字，就能做到事半功倍、举重若轻。

[04] 朴素贝氏：朴素贝叶斯；条件独立：假设条件；精召评估：精度和召回率评估；
生成性：朴素贝叶斯是生成性分类器(相当于会“写”字)；ch04：对应于原书第 4 章（余者皆同）。

[05] 区分性：逻辑回归是区分性分类器(相当于只会“认”字)。叉熵：交叉熵损失函数；梯降：梯度下降算法。

[06] 词反文频：词频(TF)和反文档频(IDF)；逐点互信：逐点互信息(PMI)；单词向量：Word2Vec 算法；静嵌：静态嵌入算法。

[07] 量辖歧义：量词辖域歧义；语析：语义解析。
(此处对应原书第 16 章，而该章内容为空，故其内容为译者经揣摩而添加；另外，第 1 章和第 10 章也同样处理。)

[08] 修辞：修辞理论；树库：宾州话语树库，部整：局部和整体；句构：连贯的本质是结构化句子之间的关系。

[09] 神网语模：神经网络语言模型；正反计算：正向传递和反向传递导数计算。

[10] 三循模型：RNN、LSTM、GRU 等三个循环模型；自注：Self-Attention；变换：Transformer；核构：核心架构。

[11] 语境：上下文；语模嵌入：语言模型嵌入(ELMo 模型)；双编表变：变换器的双向编码表示(BERT 模型)；
生成模型：生成性预训练模型(GPT1、GPT2、GPT3)；动嵌：动态嵌入算法。

[12] 注力：注意力(Attention)机制；双语评估：BLEU 度量；编解码：机器翻译的原理是编码器-解码器模型。

[13] 时事：时间提取和事件提取；数构：信息提取的本质是把文本信息转化为结构化数据。

[14] 信检：基于信息检索(IR)的问答；基知：基于知识的问答；超大预模：超大规模预训练模型(T5 模型)。

[15] 机聊：聊天机器人；规库混聊：基于规则、基于语料库、混合架构的聊天机器人；信状：信念-状态（即：对话-状态）。

[16] 词类：POS；名实：命名实体识别(NER)；隐马：隐马尔可夫模型；维特：维特比算法；条件随机：条件随机场(CRF)。

[17] 语境无关：上下文无关语法；规则：语法规则；树库：宾州树库；乔组双范：乔姆斯规范(CNF)、组合范畴语法(CCG)；
嵌套：短语结构将单词组织成嵌套的(nested)成分。

[18] 科凯阳法：CKY 算法；组范解析：组合范畴语法解析；句配树：给每个句子配备一棵解析树。

[19] 点线：顶点和弧线；投射：投射性(即依存树上没有交叉的弧线)；转移：基于转移的依存解析；预言：预言机(Oracle)；
图论：基于图论的依存解析；选优：在有向图中选择最优生成树；词互依：依存树中的单词是相互依存的。

[20] 主谓双角：主题角色和动词替换；题库：命题库；框网：框架网(FrameNet)；动名签：动词和名词标签。

[21] 波频幅谱：波形、频率、振幅、声谱和频谱。

[22] 语音自识：自动语言识别(ASR)；连时分类：连接时序分类(CTC)；文本转音：文本转语音(TTS)；
音字互转：语音和文字相互转换。

[23] 悦乐君：译者笔名，出自《论语》“学而时习之，不亦说乎？有朋自远方来，不亦乐乎？人不知而不愠，不亦君子乎？”
陆海潘江：比喻有文才的人，成语出自南朝 · 梁 · 钟嵘《诗品》卷上：“陆(陆机)才如海，潘(潘岳)才如江。”
赐教勿吝：译者斗胆用古文写现代科技著作的序文，真是战战兢兢、如履薄冰，倍感诚惶诚恐，敬请诸位看官不吝赐教！

目录总览

1 概述（空）.....1
2 正则表达式、文本规范化、编辑距离.....3
3 N 元语法语言模型.....25
4 朴素贝叶斯和情感分类.....45
5 逻辑回归.....63
6 向量语义和嵌入.....79
7 神经网络和神经语言模型.....105
8 词类和命名实体的序列标签.....123
9 用于序列处理的深度学习架构.....143
10 上下文嵌入（空）.....165
11 机器翻译和编解码器模型.....167
12 成分语法.....189
13 成分解析.....213
14 依存解析.....231
15 句子含义的逻辑表示.....251
16 计算语义和语义解析（空）.....269
17 信息提取.....271
18 单词意义和 WordNet.....289
19 语义角色标签.....303
20 情感、感情和内涵的词汇.....319
21 共指消解.....337
22 话语连贯.....357
23 问题解答.....373
24 聊天机器人和对话系统.....395
25 语音学.....421
26 自动语音识别和文本转语音.....439
27 附录 A：隐马尔可夫模型(HMM).....461

主题索引.....473
参考书目.....482

目录

1. 概述.....1
1.1. SLP 发展历史.....2
2. 正则表达式，文本规范化，编辑距离.....3
2.1. 正则表达式.....4
2.1.1. 正则表达式的基本模式.....4
2.1.2. 析取，分组和优先.....6
2.1.3. 一个简单例子.....7
2.1.4. 更多操作符.....7
2.1.5. 更复杂的例子.....8
2.1.6. 替代，捕获组和 ELIZA.....8
2.1.7. 前瞻断言.....10
2.2. 单词.....10
2.3. 语料库.....11
2.4. 文本规范化.....12
2.4.1. 用于粗糙符记化和规范化的 Unix 工具.....12
2.4.2. 单词符记化.....13
2.4.3. 符记化的字节对编码.....15
2.4.4. 单词规范化，词形还原和词干化.....17
2.4.5. 句子分割.....18
2.5. 最小编辑距离.....18
2.5.1. 最小编辑距离算法.....19
2.6. 总结.....22
2.7. 文献和历史说明.....22
2.8. 练习.....23
3. N 元语法语言模型.....25
3.1. N 元语法.....26
3.2. 评估语言模型.....29
3.2.1. 困惑度.....30
3.3. 泛化与零.....31
3.3.1. 未知单词.....33
3.4. 平滑.....33
3.4.1. 拉普拉斯平滑.....34
3.4.2. 加 k 平滑.....35
3.4.3. 回退和插值.....36
3.5. Kneser-Ney 平滑.....37
3.6. 巨大的语言模型和傻瓜回退.....39
3.7. 高级专题：困惑度与熵的关系.....40
3.8. 总结.....42
3.9. 文献和历史说明.....42
3.10. 练习.....43
4. 朴素贝叶斯与情感分类.....45
4.1. 朴素贝叶斯分类器.....46
4.2. 朴素贝叶斯分类器的训练.....48
4.3. 实例.....49

4.4. 情感分析的优化 50
4.5. 朴素贝叶斯用于其他文本分类任务 51
4.6. 朴素贝叶斯的语言模型 52
4.7. 评估：精度，召回率，F 量度 53
4.7.1. 用两个以上的类进行评估 54
4.8. 测试集和交叉验证 56
4.9. 统计显著性检验 56
4.9.1. 配对引导测试 57
4.10. 避免分类中的危害 59
4.11. 总结 60
4.12. 文献和历史说明 60
4.13. 练习 61
5. 逻辑回归 63
5.1. 分类: sigmoid 64
5.1.1. 例子：情感分类 65
5.2. 在逻辑回归中学习 67
5.3. 交叉熵损失函数 67
5.4. 梯度下降 68
5.4.1. 逻辑回归的梯度 70
5.4.2. 随机梯度下降算法 71
5.4.3. 弄通一个例子 71
5.4.4. 小批量训练 72
5.5. 正则化 73
5.6. 多元逻辑回归 74
5.6.1. 多元逻辑回归的特征 75
5.6.2. 多元逻辑回归的学习 75
5.7. 解释模型 76
5.8. 高级专题：推导梯度公式 77
5.8.1. 复习 77
5.8.2. 推导 77
5.9. 总结 78
5.10. 文献和历史说明 78
5.11. 练习 78
6. 向量语义和嵌入 79
6.1. 词汇语义 79
6.2. 向量语义 82
6.3. 单词和向量 83
6.3.1. 向量和文档 83
6.3.2. 单词作为向量：文档维度 84
6.3.3. 单词作为向量：单词维度 85
6.4. 余弦测量相似度 86
6.5. TF-IDF：向量中的权重项 87
6.6. 逐点互信息(PMI) 89
6.7. TF-IDF 或 PPMI 矢量模型的应用 92
6.8. Word2vec 92

6.8.1. 分类器 93
6.8.2. 学习 skip-gram 嵌入 94
6.8.3. 其他类型的静态嵌入 97
6.9. 可视化嵌入 97
6.10. 嵌入的语义属性 98
6.10.1. 嵌入和历史语义 99
6.11. 偏见和嵌入 99
6.12. 评估向量模型 101
6.13. 总结 102
6.14. 文献和历史说明 102
6.15. 练习 103
7. 神经网络和神经语言模型 105
7.1. 单元 105
7.2. XOR 问题 107
7.2.1. 解决方案：神经网络 109
7.3. 前馈神经网络 110
7.4. 训练神经网络 113
7.4.1. 损失函数 113
7.4.2. 计算梯度 114
7.4.3. 计算图 114
7.4.4. 计算图上的逆向微分 114
7.4.5. 学习的更多细节 117
7.5. 神经语言模型 117
7.5.1. 嵌入 118
7.5.2. 训练神经语言模型 120
7.6. 总结 121
7.7. 文献和历史说明 121
7.8. 练习 121
8. 词类和命名实体的序列标签 123
8.1. (大部分)英语单词的分类 123
8.2. 词类标记 125
8.3. 命名实体和命名实体标记 127
8.4. HMM 词类标记 128
8.4.1. 马尔可夫链 128
8.4.2. 隐马尔可夫模型 129
8.4.3. HMM 标记器的成分 130
8.4.4. HMM 标记作为解码 131
8.4.5. 维特比(Viterbi)算法 132
8.4.6. 通过一个例子 133
8.5. 条件随机场(CRF) 134
8.5.1. CRF POS 标记器的特性 136
8.5.2. NER 的 CRF 特征 137
8.5.3. CRF 的推理和训练 138
8.6. 评估 NER 138
8.7. 更多细节 139

8.7.1. 双向性 139
8.7.2. 基于规则的方法 139
8.7.3. 形态丰富语言的 POS 标记 139
8.8. 总结 140
8.9. 文献和历史说明 141
8.10. 练习 142
9. 深度学习架构用于序列处理 143
9.1. 语言模型回顾 144
9.2. 循环神经网络 145
9.2.1. RNN 中的推理 146
9.2.2. 训练 146
9.2.3. RNN 作为语言模型 147
9.2.4. RNN 的其它应用 149
9.2.5. RNN 用于序列分类 150
9.2.6. 堆叠和双向 RNN 150
9.3. 在 RNN 中管理上下文：LSTM 和 GRU 152
9.3.1. 长短期记忆(LSTM) 153
9.3.2. 门控循环单元(GRU) 155
9.3.3. 门控单元、层和网络 155
9.4. Self-Attention 网络：Transformer 156
9.4.1. Transformer 作为自回归语言模型 160
9.5. 语言模型的潜在危害 162
9.6. 总结 163
9.7. 文献和历史说明 163
9.8. 练习 163
10. 上下文嵌入 165
10.1. 上下文概述 166
11. 机器翻译和编码器-解码器模型 167
11.1. 语言差异与类型学 168
11.1.1. 词序类型学 168
11.1.2. 词汇差异 169
11.1.3. 形态类型学 170
11.1.4. 指代密度 170
11.2. 编码器-解码器模型 171
11.3. 带 RNN 的编解码器 171
11.3.1. 训练编码器-解码器模型 174
11.4. 注意力 174
11.5. 波束搜索 176
11.6. 编解码器与 Transformers 178
11.7. 构建 MT 系统的实用细节 178
11.7.1. 符记化 178
11.7.2. MT 语料库 179
11.7.3. 反向翻译 180
11.8. 机器翻译评价 180
11.8.1. 用人工评分员对机器翻译进行评价 181

11.8.2. 自动评估：BLEU 181
11.8.3. 自动评估：基于嵌入的方法 183
11.9. 偏见和道德问题 184
11.10. 总结 185
11.11. 文献和历史说明 186
11.12. 练习 187
12. 成分语法 189
12.1. 成分 189
12.2. 上下文无关语法(CFG) 190
12.2.1. 上下文无关语法的正式定义 193
12.3. 一些英语语法规则 193
12.3.1. 句级结构 193
12.3.2. 从句和句子 195
12.3.3. 名词短语 195
12.3.4. 动词短语 197
12.3.5. 并列关系 198
12.4. 树库 199
12.4.1. 例子：宾州树库项目 199
12.4.2. 树库作为语法 201
12.4.3. 中心词及其发现 203
12.5. 语法等价与乔姆斯基范式(CNF) 204
12.6. 词汇语法 204
12.6.1. 组合范畴语法 205
12.7. 总结 209
12.8. 文献和历史说明 209
12.9. 练习 211
13. 成分解析 213
13.1. 歧义 213
13.2. CKY 解析：一种动态规划方法 215
13.2.1. CNF 转换 215
13.2.2. CKY 识别 217
13.2.3. CKY 解析 218
13.2.4. CKY 实践 220
13.3. 基于跨度的神经成分解析 220
13.3.1. 计算跨度分数 220
13.3.2. 集成跨度分数到解析中 221
13.4. 评估解析器 222
13.5. 部分解析 222
13.6. CCG 解析 223
13.6.1. CCG 中的歧义 223
13.6.2. CCG 解析框架 224
13.6.3. 超级标记 225
13.6.4. 使用 A*算法的 CCG 解析 225
13.7. 总结 227
13.8. 文献和历史说明 229

13.9. 练习 230
14. 依存解析 231
14.1. 依存关系 232
14.2. 依存形式 233
14.2.1. 投射性 233
14.3. 依存树库 235
14.4. 基于转移的依存解析 235
14.4.1. 创建一个预言机(Oracle) 238
14.4.2. 基于转移解析中的高级方法 241
14.5. 基于图的依存解析 243
14.5.1. 解析 244
14.5.2. 特征和训练 246
14.5.3. 基于图解析的高级问题 247
14.6. 评估 247
14.7. 总结 248
14.8. 文献和历史说明 249
14.9. 练习 250
15. 句子含义的逻辑表示 251
15.1. 表示的计算需要 252
15.2. 模型理论语义学 253
15.3. 一阶逻辑 255
15.3.1. 一阶逻辑的基本要素 255
15.3.2. 变量和量词 256
15.3.3. Lambda 表示法 257
15.3.4. 一阶逻辑的语义 258
15.3.5. 推理 259
15.4. 事件和状态表示 260
15.4.1. 时间表示 261
15.4.2. 体 262
15.5. 描述逻辑 263
15.6. 总结 267
15.7. 文献和历史说明 267
15.8. 练习 268
16. 计算语义和语义解析 269
16.1. 计算语义概述 270
17. 信息提取 271
17.1. 关系提取 272
17.2. 关系提取算法 274
17.2.1. 使用模式提取关系 274
17.2.2. 通过监督学习提取关系 275
17.2.3. 通过自举半监督提取关系 276
17.2.4. 通过远程监督提取关系 278
17.2.5. 通过无监督提取关系 279
17.2.6. 关系抽取的评价 280
17.3. 提取时间 280

17.3.1. 时间表达式提取 281
17.3.2. 时间规范化 282
17.4. 提取事件及其时间 283
17.4.1. 事件的时间顺序 285
17.5. 模板填充 285
17.5.1. 模板填充的机器学习方法 286
17.5.2. 早期的有限状态模板填充系统 286
17.6. 总结 287
17.7. 文献和历史说明 288
17.8. 练习 288
18. 单词意义和 WordNet 289
18.1. 单词意义 289
18.1.1. 定义单词意义 289
18.1.2. 单词有多少种意义? 290
18.2. 意义之间的关系 291
18.3. WordNet：词汇关系数据库 292
18.3.1. WordNet 中的意义关系 293
18.4. 词义消歧 294
18.4.1. WSD:任务和数据集 294
18.4.2. WSD 算法:上下文嵌入 296
18.5. 可选的 WSD 算法和任务 297
18.5.1. 基于特征的 WSD 297
18.5.2. 以 Lesk 算法作为 WSD 基线 298
18.5.3. 上下文词的评估 298
18.5.4. 维基百科作为训练数据的来源 299
18.6. 使用词典来改善嵌入 299
18.7. 词义归纳 300
18.8. 总结 301
18.9. 文献和历史说明 301
18.10. 练习 302
19. 语义角色标签 303
19.1. 语义角色 303
19.2. 动词替换 304
19.3. 语义角色：主题角色的问题 305
19.4. 命题库(PropBank) 306
19.5. 框架网(FrameNet) 307
19.6. 语义角色标签(SRL) 308
19.6.1. 基于特征的语义角色标签算法 309
19.6.2. 语义角色标签的神经算法 310
19.6.3. 语义角色标签的评价 311
19.7. 选择限制 311
19.7.1. 选择限制的表示 312
19.7.2. 选择偏好 313
19.8. 谓词的原始分解 314
19.9. 总结 315

19.10. 文献和历史说明 316
19.11. 练习 318
20. 情感、感情和内涵的词汇 319
20.1. 定义情绪 320
20.2. 可用的情感和感情词汇 321
20.3. 通过人工标签创建感情词汇 322
20.4. 感情词汇的半监督归纳 323
20.4.1. 语义轴方法 323
20.4.2. 标签传播 324
20.4.3. 其它方法 325
20.5. 单词情感的监督学习 326
20.5.1. 对数优势比率信息的狄利克雷先验 328
20.6. 使用词汇进行情感识别 330
20.7. 其他任务：人格 330
20.8. 感情识别 331
20.9. 以实体为中心感情的基于词汇的方法 332
20.10. 内涵框架 332
20.11. 总结 334
20.12. 文献和历史说明 335
20.13. 练习 336
21. 共指消解 337
21.1. 共指现象：语言背景 339
21.1.1. 指示语的类型 339
21.1.2. 信息状态 340
21.1.3. 复杂化：非指示语 341
21.1.4. 共指关系的语言属性 342
21.2. 共指任务和数据集 343
21.3. 提及检测 344
21.4. 共指算法的架构 346
21.4.1. 提及配对架构 346
21.4.2. 提及排名架构 347
21.4.3. 基于实体的模型 347
21.5. 使用手工构建特征的分类器 348
21.6. 一种神经提及排名算法 349
21.7. 共指消解的评估 351
21.8. Winograd 模式问题 352
21.9. 共指的性别偏见 353
21.10. 总结 354
21.11. 文献和历史说明 354
21.12. 练习 356
22. 话语连贯 357
22.1. 连贯关系 358
22.1.1. 修辞结构理论(RST) 358
22.1.2. Penn 话语树库 (PDTB) 359
22.2. 话语结构解析 361

22.2.1. 用于 RST 解析的 EDU 分段.....361
22.2.2. RTS 解析.....361
22.2.3. PDTB 话语解析.....363
22.3. 中心化和基于实体的连贯.....364
22.3.1. 中心化.....364
22.3.2. 实体网格模型.....365
22.3.3. 评估神经和基于实体的连贯.....367
22.4. 局部连贯的表示学习模型.....367
22.5. 全局连贯.....368
22.5.1. 论元结构.....369
22.5.2. 科学话语的结构.....370
22.6. 总结.....371
22.7. 文献和历史说明.....371
22.8. 练习.....372
23. 问题回答.....373
23.1. 信息检索(IR).....373
23.1.1. 术语加权和文档评分.....374
23.1.2. 文档评分.....375
23.1.3. 倒排索引.....376
23.1.4. 信息检索系统的评估.....376
23.1.5. 稠密向量 IR.....379
23.2. 基于 IR 的事实性问答.....380
23.2.1. 基于 IR 的 QA：数据集.....381
23.2.2. 基于 IR 的 QA：阅读器(答案跨度提取).....382
23.3. 实体连接.....383
23.3.1. 基于锚字典和 Web 图的连接.....383
23.3.2. 基于神经图的连接.....385
23.4. 基于知识的 QA.....387
23.4.1. 来自 RDF 三元组存储的基于知识的 QA.....387
23.4.2. 语义解析的 QA.....388
23.5. QA 中的语言模型(T5).....389
23.6. 分类 QA 模型.....389
23.7. 事实性答案的评估.....392
23.8. 总结.....392
23.9. 文献与历史说明.....393
23.10. 练习.....394
24. 聊天机器人和对话系统.....395
24.1. 人类会话的特性.....395
24.2. 聊天机器人.....398
24.2.1. 基于规则的聊天机器人：ELIZA 和 PARRY.....399
24.2.2. 基于语料库聊天机器人.....401
24.2.3. 混合架构.....403
24.3. 简单的基于框架的对话系统.....404
24.3.1. 基于框架对话的控制结构.....405
24.3.2. NLU 用以填充 GUS 中的槽.....405

24.3.3. 基于框架对话的其他组件 406
24.4. 对话-状态架构 407
24.4.1. 对话行为 408
24.4.2. 槽填充 408
24.4.3. 对话状态跟踪 409
24.4.4. 对话策略 410
24.4.5. 对话-状态模型中的 NLG 412
24.5. 评估对话系统 413
24.5.1. 评估聊天机器人 413
24.5.2. 评估基于任务的对话 415
24.6. 对话系统设计 416
24.6.1. 对话系统设计中的伦理问题 416
24.7. 总结 417
24.8. 文献和历史说明 418
24.9. 练习 420
25. 语音学 421
25.1. 语音和语音转写 421
25.2. 发音语音学 422
25.2.1. 发声器官 422
25.3. 韵律学 425
25.3.1. 韵律突显度：重读、重音和中元音 426
25.3.2. 韵律结构 426
25.3.3. 音调 427
25.4. 声学语音和信号 427
25.4.1. 波 427
25.4.2. 语音声波 428
25.4.3. 频率和振幅;音调和响度 429
25.4.4. 从波形来解释音子 431
25.4.5. 声谱和频谱 432
25.4.6. 声源滤波器模型 435
25.5. 语音资源 436
25.6. 总结 437
25.7. 文献和历史说明 437
25.8. 练习 438
26. 自动语音识别和文本语音转换 439
26.1. 自动语音识别任务 440
26.2. ASR 的特征提取：对数 Mel 声谱 441
26.2.1. 采样和量化 441
26.2.2. 加窗 442
26.2.3. 离散傅里叶变换 443
26.2.4. Mel 滤波器组和对数 444
26.3. 语音识别的架构 444
26.3.1. 学习 446
26.4. CTC 446
26.4.1. CTC 推理 447

26.4.2. CTC 训练.....449
26.4.3. CTC 和编码器-解码器的结合.....449
26.4.4. 流模型:改进 CTC 的 RNN-T.....450
26.5. ASR 评估:单词错误率.....450
26.6. TTS.....452
26.6.1. TTS 预处理:文本规范化.....453
26.6.2. TTS：频谱预测.....454
26.6.3. TTS：声码.....455
26.6.4. TTS 评估.....456
26.7. 其他语音任务.....456
26.8. 总结.....456
26.9. 文献和历史说明.....457
26.10. 练习.....459
27. 附录 A：隐马尔可夫模型(HMM).....461
27.1. 马尔可夫链.....461
27.2. 隐马尔可夫模型.....462
27.3. 似然计算：前向算法.....463
27.4. 解码：维特比(Viterbi)算法.....465
27.5. HMM 训练：前向-后向算法.....468
27.6. 总结.....472
27.7. 文献和历史说明.....472
28. 主题索引和参考文献.....473
28.1. 主题索引.....473
28.2. 参考文献.....482

图 2-1 ：一些简单的正则表达式搜索.....4
图 2-2 ：使用方括号[ ]指定字符的析取.....4
图 2-3 ：用方括号[ ]加破折号-来指定范围.....4
图 2-4 ：插入符号^表示否定或仅表示^.....5
图 2-5 ：问号？标记前一个表达式的可选性.....5
图 2-6 ：用句号指定任何字符.....5
图 2-7 ：正则表达式中的定位符.....6
图 2-8 ：常见字符集的别名.....8
图 2-9 ：用于计数的正则表达式操作符.....8
图 2-10 ：一些需要反斜杠的字符.....8
图 2-11 ：某些英语语料库的类型和符记的大致数量.....11
图 2-12 ：NLP 工具包中 RE 符记化的 Python 跟踪.....14
图 2-13 ：BPE 算法中的符记学习器部分.....16
图 2-14 ：表示两个字符串之间的最小编辑距离.....19
图 2-15 ：寻找编辑距离被视为一个搜索问题.....19
图 2-16 ：从 intention 到 execution 的路径.....20
图 2-17 ：最小编辑距离算法.....20
图 2-18 ：计算从 intention 到 execution 之间的最小编辑距离.....21
图 2-19 ：用反向指针计算对齐路径的原理图.....21
图 3-1 ：二元语法计数.....28

图 3-2 ： 二元语法概率 28
图 3-3 ： 四个 n-gram 随机生成的八个句子 32
图 3-4 ： 三个 n-gram 模型随机生成的三个句子 32
图 3-5 ： 加一平滑 bigram 计数的 8 个单词 34
图 3-6 ： 加一平滑 bigram 概率的 8 个单词 35
图 3-7 ： 对 8 个单词的加一重建计数 35
图 3-8 ： 训练集和保留集的 bigrams 计数 37
图 4-1 ： 多项式朴素贝叶斯分类器的直观感受(词袋假设) 46
图 4-2 ： 使用加一平滑处理的朴素贝叶斯算法 49
图 4-3 ： 二进制朴素贝叶斯算法的二值化示例 51
图 4-4 ： 混淆矩阵在黄金标签下的表现 53
图 4-5 ： 三类分类任务的混淆矩阵 55
图 4-6 ： 合并后的混淆矩阵以及微观和宏观平均的精度 55
图 4-7 ： 十倍交叉验证 56
图 4-8 ： 配对的引导测试 58
图 4-9 ： 配对引导算法 59
图 5-1 ： sigmoid 函数 64
图 5-2 ： 微型测试文档样本向量 x 中的提取特征 66
图 5-3 ： 梯度下降算法的可视化 69
图 5-4 ： 梯度矢量的可视化 70
图 5-5 ： 随机梯度下降算法 71
图 6-1 ： 部分单词和短语的二维嵌入投影(t-SNE) 83
图 6-2 ： 术语-文档矩阵 83
图 6-3 ： 术语-文档矩阵的列向量 84
图 6-4 ： 文档向量的空间可视化 84
图 6-5 ： 术语-文档矩阵的行向量 85
图 6-6 ： 四个单词的共现向量 85
图 6-7 ： 词向量的空间可视化 86
图 6-8 ： 余弦相似度的(粗略)图形演示 87
图 6-9 ： tf-idf 加权术语-文档矩阵 89
图 6-10 ： 四个单词在五种语境下的共现数 90
图 6-11 ： 联合概率 91
图 6-12 ： PPMI 矩阵 91
图 6-13 ： skip-gram 模型学习到的嵌入 94
图 6-14 ： 梯度下降法的直观感受 96
图 6-15 ： 类比问题的平行四边形模型 98
图 6-16 ： GloVe 向量空间的关系属性 100
图 6-17 ： 使用 word2vec 进行 t-SNE 可视化 101
图 7-1 ： Sigmoid 函数 106
图 7-2 ： 一个神经单元 106
图 7-3 ： tanh 和 ReLU 激活函数 107
图 7-4 ： 逻辑函数的感知器 108
图 7-5 ： 函数 AND，OR 和 XOR 的分类线 108
图 7-6 ： Goodfellow 等人的 XOR 解决方案 109
图 7-7 ： 隐含层形成输入的新表示 109

图 7-8 ： 一个简单的 2 层前馈网络.....110
图 7-9 ： 用 x_0 替换偏置节点 b.....112
图 7-10 ： 函数 L(a,b, c) = c(a+2b)的正向传递计算图.....115
图 7-11 ： 函数 L(a,b, c) = c(a+2b)的反向传递计算图.....116
图 7-12 ： 简单 2 层神经网络的计算图.....116
图 7-13 ： 前馈神经语言模型的简化视图.....118
图 7-14 ： 学习嵌入的所有方法.....119
图 8-1 ： 英语单词的两大类和 17 个词类.....124
图 8-2 ： Penn Treebank 词类标记.....125
图 8-3 ： 词类标记的任务.....126
图 8-4 ： 标记歧义.....126
图 8-5 ： 通用命名实体类型及其引用.....127
图 8-6 ： 使用 Washington 名称时类型歧义的示例.....127
图 8-7 ： NER 序列模型的 IO-BIO-BIOES 标记.....128
图 8-8 ： 天气和单词的马尔可夫链.....129
图 8-9 ： HMM 表示的两个部分.....131
图 8-10 ： 用于找到最佳标记序列的维特比算法.....132
图 8-11 ： “Janet will back the bill”的格子示意图.....133
图 8-12 ： 转移概率 A.....133
图 8-13 ： 观察似然度 B.....133
图 8-14 ： 维特比算法示意图.....134
图 8-15 ： 基于特征的 NER 系统的典型特性.....137
图 8-16 ： 示例句子的一些 NER 特征.....137
图 9-1 ： 前馈神经语言模型的简化视图.....143
图 9-2 ： Elman 的简单循环神经网络.....145
图 9-3 ： 前馈网络的简单循环神经网络.....145
图 9-4 ： 简单循环网络中的正向推理.....146
图 9-5 ： 在时间上展开的简单循环神经网络.....147
图 9-6 ： 训练 RNN 作为语言模型.....148
图 9-7 ： 基于 RNN 的神经语言模型自回归生成.....149
图 9-8 ： 使用简单的 RNN 将词类标记当作序列标签.....150
图 9-9 ： 使用简单的 RNN 和前馈网络进行序列分类.....151
图 9-10 ： 堆叠式 RNN.....151
图 9-11 ： 一个双向 RNN.....152
图 9-12 ： 用于序列分类的双向 RNN.....153
图 9-13 ： 单个 LSTM 单元的计算图.....154
图 9-14 ： FF、SRN、LSTM 和 GRU.....155
图 9-15 ： Self-Attention 模型.....156
图 9-16 ： 自注意力的计算.....157
图 9-17 ： 典型的 Transformer 模块.....159
图 9-18 ： 多头自注意力.....159
图 9-19 ： 将 Transformer 训练为语言模型.....160
图 9-20 ： Transformers 的自回归文本补全.....161
图 9-21 ： CNN/每日邮报语料库的文章和摘要示例.....161
图 9-22 ： Transformers 用于摘要.....162

图 11-1 ： 其他词序差异的例子 169
图 11-2 ： 英语和法语翻译之间的复杂重叠 169
图 11-3 ： 编码器-解码器架构 171
图 11-4 ： 基本的 RNN 编码器-解码器的机器翻译 172
图 11-5 ： 正式的 RNN 编解码器的机器翻译 172
图 11-6 ： 解码器的隐藏状态 173
图 11-7 ： 训练的 RNN 编解码器的机器翻译 173
图 11-8 ： 编码器的隐藏状态-瓶颈 174
图 11-9 ： 注意力机制的动态上下文 175
图 11-10 ： 含注意力的编解码器网络 175
图 11-11 ： 搜索树 176
图 11-12 :波束搜索解码的原理 177
图 11-13 ： 波束搜索解码评分 178
图 11-14 ： 波束搜索解码算法伪代码 179
图 11-15 ： 英文和法文句子之间的对齐示例 180
图 11-16 ： BLEU 的直觉 182
图 11-17 ： BLEU 使用的病态例句 182
图 11-18 ： 计算 BERTSCORE 的召回率 184
图 11-19 ： 匈牙利语译成英语时的性别问题 185
图 11-20 ： 沃夸三角形 186
图 12-1 ： “A flight”的解析树 191
图 12-2 ： $\mathcal{L}_0$ 的词汇表 192
图 12-3 ： $\mathcal{L}_0$ 的语法以及每个规则的示例短语 192
图 12-4 ： 根据语法 $\mathcal{L}_0$ 的“I prefer a morning flight”的解析树 192
图 12-5 ： 句子解析树的例子 197
图 12-6 ： 一组实例动词的次范畴化框架 198
图 12-7 ： 从 LDC 树库 3 版本解析出的句子 200
图 12-8 ： Brown 语料库句子对应的树形图 200
图 12-9 ： 来自 LDC 宾州树库的一句话 201
图 12-10 ： 从三个树库句子中提取 CFG 语法规则的示例 202
图 12-11 ： 柯林斯(1999)的词汇树 203
图 12-12 ： Collins(1999)的一些中心词规则 204
图 13-1 ： $\mathcal{L}_1$ 微型英语语法和词汇 213
图 13-2 ： 歧义句的两棵解析树 214
图 13-3 ： $\mathcal{L}_1$ 语法及其到 CNF 的转换 216
图 13-4 ： 句子 Book the flight through Houston 的完整的解析表 216
图 13-5 ： CKY 算法 217
图 13-6 ： 填充 CKY 表中第[i, j]个单元格的所有方法 218
图 13-7 ： 读取 Houston 单词后填充第 5 列的单元格 219
图 13-8 ： 计算“the flight”跨度分数的简化大纲 220
图 13-9 ： 基于 A*算法的 CCG 解析 226
图 13-10 ： A*搜索“United serves Denver”的示例 228
图 14-1 ： 一个依存类型的解析及其相应的基于成分的分析 231
图 14-2 ： 选自通用依存集合的依存关系 232
图 14-3 ： 核心通用依存关系的例子 233

图 14-4 ： 宾州树库 3 的《华尔街日报》组件的短语结构树.....234
图 14-5 ： 基于转移的基本解析器.....236
图 14-6 ： 通用的基于转移的依存关系解析器算法.....237
图 14-7 ： 基于转移的解析的跟踪.....237
图 14-8 ： 用参考解析生成训练项.....239
图 14-9 ： 训练基于转移的依存分析器的标准特征模板.....241
图 14-10 ： 使用弧线急转转移操作符的处理轨迹.....242
图 14-11 ： 波束搜索应用于基于转移的依存解析.....243
图 14-12 ： Book that flight 的初始根有向图.....245
图 14-13 ： 在加权有向图中寻找最大生成树的 CLE 算法.....246
图 14-14 ： 基于图的 CLE 算法示例.....246
图 14-15 ： 参考解析和系统解析示例.....247
图 15-1 ： 含义表示的采样器.....251
图 15-2 ： 餐厅世界的模型.....254
图 15-3 ： 一阶逻辑表示的句法的上下文无关语法规范.....256
图 15-4 ： 真值表给出了各种逻辑连接词的语义.....258
图 15-5 ： Reichenbach 的研究方法应用于各种英语时态.....262
图 15-6 ： 一组包含关系的图形网络表示.....264
图 15-7 ： 完整包含关系集的图形网络表示.....266
图 17-1 ： ACE 关系提取任务中使用的 17 个关系.....272
图 17-2 ： 带有示例的语义关系及其所涉及的命名实体类型.....272
图 17-3 ： 示例文本中基于模型的关系和实体视图.....273
图 17-4 ： 来自 TACRED 数据集的例句和标签.....274
图 17-5 ： 用于找到上位词的词汇-句法模式.....274
图 17-6 ： 查找并分类文本中实体之间的关系.....275
图 17-7 ： 关系提取作为编码器上的线性层(本例中为 BERT).....276
图 17-8 ： 从种子实体对中通过自举来学习关系.....277
图 17-9 ： 关系提取的远程监督算法.....278
图 17-10 ： 分类器的功能.....280
图 17-11 ： 绝对、关系和持续时间表达式的例子.....281
图 17-12 ： 时间词汇触发器示例.....281
图 17-13 ： 来自 Tarsqi 中的 GUTime 时间标记系统的 Perl 片段.....281
图 17-14 ： 用于训练 IOB 风格时间表达式标记器的典型特征.....282
图 17-15 ： 包含时间表达式的规范化值的 TimeML 标注.....282
图 17-16 ： 表示不同时间和持续时间的 ISO 模式示例.....283
图 17-17 ： 在事件检测中基于规则和机器学习方法的常用特征.....284
图 17-18 ： Allen(1984)给出的 13 个时间关系.....284
图 17-19 ： 来自 TimeBank 语料库的示例.....285
图 17-20 ： FASTUS 在输入文本后生成的模板.....287
图 17-21 ： FASTUS 中的处理水平.....287
图 17-22 ： FASTUS 的阶段 5 生成的 5 个部分模板.....287
图 18-1 ： 名词 bass 的 WordNet 3.0 词条的一部分.....292
图 18-2 ： 超意义:WordNet 中名词的 26 个词汇分类.....293
图 18-3 ： WordNet 中的一些名词关系.....293
图 18-4 ： WordNet 中的一些动词关系.....293

图 18-5 ： 词元 bass 的两种不同意义的下位关系链.....294
图 18-6 ： WordNet 作为一个图.....295
图 18-7 ： 单词 bass 的一些可能的意义标记清单.....295
图 18-8 ： 全词 WSD 任务.....296
图 18-9 ： WSD 的最近邻算法.....297
图 18-10 ： 简化的 Lesk 算法.....298
图 18-11 ： WIC 数据集的正(T)和负(F)样本偶对.....299
图 18-12 ： 在 GloVe 中的最近邻居.....299
图 19-1 ： 一些常用的主题角色及其定义.....304
图 19-2 ： 各种主题角色的一些原型例子.....304
图 19-3 ： 框架元素.....308
图 19-4 ： 一个通用的语义-角色-标签算法.....309
图 19-5 ： PropBank 句子的解析树.....309
图 19-6 ： BERT + 双向 LSTM 语义角色标签方法.....311
图 19-7 ： WordNet 的证据表明汉堡包是可以食用的.....312
图 19-8 ： 一组概念依存(CD)原语.....315
图 20-1 ： 感情(affective)状态的 Scherer 类型.....319
图 20-2 ： Plutchik 情绪轮-8 种基本情绪.....320
图 20-3 ： 普通调查员、主观词汇、极性词汇.....321
图 20-4 ： 样本词在 Mohammad 情绪维度上的值.....321
图 20-5 ： 对愤怒、恐惧、快乐和悲伤词汇的情绪强度样本.....322
图 20-6 ： LIWC 中 73 个词汇类别中的 5 个样本.....322
图 20-7 ： SentProp 算法的直观感受.....325
图 20-8 ： 来自 SentiWordNet 3.0 的示例.....325
图 20-9 ： 摘自不同评论网站的一些评论.....326
图 20-10 ： 正面、负面标量形容词的 Potts 图.....327
图 20-11 ： 强调和削弱副词的 Potts 图.....327
图 20-12 ： 与 1 星和 5 星餐厅评论相关的前 50 个单词.....329
图 20-13 ： 来自 Schwartz 等人(2013 年)的单词云.....332
图 20-14 ： 《黑暗骑士》中角色的权力(支配)、情感(效价)和施事者(唤起).....333
图 20-15 ： survive 和 violate 的内涵框架.....333
图 20-16 ： Sap 的内涵框架.....334
图 21-1 ： 在话语模型中，提及如何唤起和访问话语实体.....337
图 21-2 ： 提及配对分类器.....346
图 21-3 ： 提及排名系统的概率分布.....347
图 21-4 ： 基于特征的共指算法的一些常见特征.....348
图 21-5 ： 跨度表示和提及分数的计算.....350
图 21-6 ： 先行词得分的计算.....351
图 21-7 ： Lee 模型的样本预测.....351
图 22-1 ： 一棵话语树.....359
图 22-2 ： PDTB 含义层次中的四个高层语义区分.....360
图 22-3 ： PDTB 含义层次结构.....360
图 22-4 ： biLSTM-CRF 用于 EDU 分段.....361
图 22-5 ： RST 话语树显示四种 EDU 的例子.....362
图 22-6 ： 使用 shift-reduce 解析器解析图 22.5 的例子.....362

图 22-7 ： 规则 2 的定心转换.....365
图 22-8 ： 实体网格.....366
图 22-9 ： 实体话语.....366
图 22-10 ： 表示文档的特征向量.....366
图 22-11 ： 文档连贯性的 LCD 模型架构.....368
图 22-12 ： 说服性文章的论元结构.....370
图 22-13 ： 论元区划示例.....370
图 23-1 ： 即席 IR 的系统架构.....374
图 23-2 ： 计算 nano 文档 1 和文档 2 的 tf-idf 值.....375
图 23-3 ： 根据公式(23.9)对文档进行排序.....375
图 23-4 ： 针对排名的精度和召回率的计算.....377
图 23-5 ： 表 23.4 数据的精度和召回率的曲线.....378
图 23-6 ： 从图 23.4 中插入数据点.....378
图 23-7 ： 插值的精度-召回曲线.....379
图 23-8 ： BERT 双编码器用于计算文档与查询的相关性.....379
图 23-9 ： 一些事实性问题及其答案.....380
图 23-10 ： 基于 IR 的检索和阅读模型.....380
图 23-11 ： 一篇包含问题和答案的文章.....381
图 23-12 ： 基于跨度的 QA 编码器模型(使用 BERT).....382
图 23-13 ： ELQ 算法推理过程示意图.....385
图 23-14 ： 语义解析器为回答问题而生成的示例逻辑形式.....388
图 23-15 ： 编码器-解码器语义解析器.....389
图 23-16 ： T5 系统的编码器-解码器架构.....389
图 23-17 ： Watson QA 的 4 个主要阶段.....390
图 24-1 ： 旅行代理和客户之间的电话会话.....396
图 24-2 ： 用户和 BlenderBot 之间的对话.....398
图 24-3 ： 微软小冰聊天机器人翻译的中文会话样本.....399
图 24-4 ： Weizenbaum(1966)的 ELIZA 对话样本.....399
图 24-5 ： ELIZA 算法的简化示意图.....400
图 24-6 ： 神经聊天机器人生成响应的两种架构.....402
图 24-7 ： 对话响应生成的编码器-解码器示例.....402
图 24-8 ： 红衣教主的对话部分.....404
图 24-9 ： GUS 系统实际对话的文字记录.....404
图 24-10 ： 基于框架的对话系统中的槽.....405
图 24-11 ： 用户句子的语义语法解析.....406
图 24-12 ： 面向任务对话的对话-状态系统架构.....407
图 24-13 ： HIS 餐厅推荐系统使用的对话行为.....408
图 24-14 ： HIS 系统对话示例(使用图 24-13 中的对话行为).....408
图 24-15 ： 一种用于槽填充的简单架构.....409
图 24-16 ： NLG 句子实现阶段的两个输入示例.....412
图 24-17 ： 去词汇化的句子.....412
图 24-18 ： 一个句子实现器将槽/填充符映射到英语.....413
图 24-19 ： 急性评估(acute-eval)方法.....414
图 24-20 ： 用户满意度调查.....415
图 25-1 ： 英语辅音(左)和元音(右)的 ARPAbet 和 IPA 符号.....421

图 25-2 ： 发声器官的侧视图.....422
图 25-3 ： 英语主要的发音部位.....423
图 25-4 ： 英语三个元音的舌位：前高[iy]，前低[ae]和后高[uw].....424
图 25-5 ： 英语元音的“元音空间”示意图.....424
图 25-6 ： 火腿(ham)、绿色(green)、鸡蛋(eggs)的音节结构(σ=音节).....425
图 25-7 ： ToBI 重读和边界声调标签.....427
图 25-8 ： 频率为 10Hz，振幅为 1 的正弦波.....428
图 25-9 ： 某语段中元音[iy]的波形图.....428
图 25-10 ： Microsoft wavefile 头文件格式.....429
图 25-11 ： 问句“Three o’clock?”的音高轨迹.....429
图 25-12 ： 句子 Is it a long movie?的音强图.....430
图 25-13 ： 句子 She just had a baby 的波形图.....431
图 25-14 ： 单词“she”的详细视图.....431
图 25-15 ： 一个波形是两个正弦波之和.....432
图 25-16 ： 来自图 25.15 中波形的声谱.....432
图 25-17 ： 从图 25.13 中截取了 had 单词中元音[ae]部分的波形.....433
图 25-18 ： 来自图 25.13 中单词 had 的元音[ae]的声谱.....433
图 25-19 ： 美国英语[ih]、[ae]、[uh]三个元音的频谱.....434
图 25-20 ： 句子“she just had a baby”的频谱.....434
图 25-21 ： 元音[iy]的平滑(LPC)频谱.....435
图 25-22 ： 作为过滤器的声腔位置的可视化.....435
图 25-23 ： 来自 TIMIT 语料的语音转写例子.....436
图 25-24 ： Switchboard 中的短语的语音转写.....436
图 26-1 ： 单词错误率(WER)和字符错误率(CER).....441
图 26-2 ： 加窗：步长为 10ms 帧长为 25 ms 的矩形窗.....442
图 26-3 ： 加窗：正弦波矩形窗或汉明窗.....443
图 26-4 ： 信号的汉明窗及其 DFT 计算的声谱.....443
图 26-5 ： 梅尔(mel)滤波器组.....444
图 26-6 ： 编码器-解码器语音识别器的架构原理图.....445
图 26-7 ： 用于折叠输入和字母之间的对齐方式的朴素算法.....446
图 26-8 ： CTC 折叠函数 B.....447
图 26-9 ： 产生转录 dinner 的其他三个合法对齐.....447
图 26-10 ： CTC 推理.....448
图 26-11 ： 结合 CTC 和编译码器损失函数.....449
图 26-12 ： RNN-T 模型计算 t 时刻的输出符记分布.....450
图 26-13 ： 文本规范化中的一些非标准词类.....453
图 26-14 ： Tacotron2 架构.....454
图 26-15 ： 扩张卷积.....455

1. 概述

[The last thing you figure out in writing a book is what to put first.]

Pascal

[您写书时发现的最后一件事要放在最前面。]

帕斯卡

(空空如也)

1.1. SLP 发展历史

(空空如也)

2. 正则表达式，文本规范化，编辑距离

用户：我不快乐。
ELIZA：您是否认为来到这里会帮助您得到快乐？
用户：我需要一些帮助，这似乎可以肯定。
ELIZA：如果您得到帮助，对您有什么意义？
用户：也许我可以学会和妈妈相处。
ELIZA：告诉我有关您家庭的更多信息。
用户：我妈妈照顾我。
ELIZA：家庭中的还有其他人照料您吗？
用户：我父亲。
ELIZA：你的父亲？
用户：从某些方面来说，你就像我的父亲。

魏岑鲍姆(1966)

上面的对话来自 **ELIZA(伊丽莎)**，这是一种早期的自然语言处理系统，可以通过模仿罗杰式(Rogerian)心理治疗师的回答与用户进行有限的对话(Weizenbaum，1966)。ELIZA 是一个非常简单的程序，它使用模式匹配来识别“我需要 X”之类的短语并进行翻译将它们转换为合适的输出，例如“如果您得到 X 对您意味着什么？”。这种简单的技术在该领域中很成功，因为 ELIZA 实际上不需要了解任何模仿罗杰式心理治疗师的知识。

正如 Weizenbaum 所指出的那样，这是为数不多的对话类型之一，听众可以表现得好像对世界一无所知。ELIZA 的模仿人类对话非常成功：许多与 ELIZA 互动的人开始相信，ELIZA 确实理解了他们以及他们的问题，即使在向他们解释了程序的操作之后，许多人仍然相信 ELIZA 的能力(Weizenbaum，1976)，并且即使在今天，这样的**聊天机器人(catbots)**还是一个有趣的消遣。

当然，现代的对话主体不只是转移注意力。更重要的是，他们可以回答问题、预定航班或寻找餐馆，依靠这些功能可以对用户的意图进行更为复杂的了解，我们将在第 24 章中看到。尽管如此，为 ELIZA 和其他聊天机器人提供支持的基于模式的简单方法，在自然语言处理中起着至关重要的作用。

我们将从描述文字模式的最重要工具开始：正则表达式。正则表达式可用于：指定我们可能想从文档中提取的字符串、转换上述 Eliza 中的“我需要 X”、定义文档价格表中的诸如$ 199 或$ 24.99 之类的字符串。

然后，我们转到一组统称为**文本规范化(text normalization)**的任务，其中文本规范化正则表达式起着重要的作用。规范化文本意味着将其转换为更方便的标准格式。例如，我们将要使用的大多数语言都依赖于首先从运行文本中**分离出单词(tokennizing)**，即**符记化(tokennization)**的任务。英语单词经常用空格将彼此分开标记化，但仅有空格是不够的。尽管“纽约(New York)”和“摇滚(rock ’n’ roll)”有时包含空格，但有时仍会被视为大词，而有时我们需要将“I’m”分为“I”和“am”两个词。为了处理推特(tweets)或文字，我们需要作符记像 :)的表情符号，或像# nlproc 的主题标签。某些语言(例如日语)在单词之间没有空格，因此单词符记化变得更加困难。

文本规范化的另一任务是**词形还原(Lemmatization)**，确定两个单词是否有相同的词根，尽管它们表面不同。例如，单词 sang，sung 和 sings 是动词 sing 的形式。单词 sing 是这些单词的常见词缀，而词缀生成器则将所有这些词映射为 sing。词形还原对于处理诸如阿拉伯语之类的形态复杂的语言至关重要。**词干化(stemming)**是简化词形还原的一种形式，其中我们主要从词的末尾去除后缀。文本规范化还包括句子**分段(segmentation)**：使用句点或感叹号等线索将文本分解为单个句子。

最后，我们需要比较单词和其他字符串。我们将引入一个称为“编辑距离”的度量，该度量基于将一个

字符串转换为另一个字符串所需的编辑次数(插入、删除、替换)来衡量两个字符串的相似程度。编辑距离是一种算法，可应用于从拼写校正到语音识别再到共指解析的整个语言处理过程。

2.1. 正则表达式

正则表达式(Regular Expression，RE)是一种专门用于在文本中搜索字符串的语言，它是计算机科学标准化领域鲜为人知的成功之一。每种计算机语言，文字处理器和文本处理工具(例如 Unix 工具 grep 或 Emacs)都使用此实用语言。形式上，正则表达式是用于表征一组字符串的代数符号。当我们有一种搜索模式和一整套文本要搜索时，它们对于搜索文本特别有用。正则表达式搜索功能将搜索**语料库(corpus)**，返回所有与模式匹配的文本。语料库可以是单个文档或集合。例如，Unix 命令行工具 grep 接受一个正则表达式，并返回与该表达式匹配的输入文档的每一行。

可以将搜索设计为返回一行中的每个匹配项(如果有多个匹配项)，或者仅返回第一个匹配项。在以下示例中，我们通常在与正则表达式匹配的模式的确切部分作下划线并仅显示第一个匹配项。我们将显示用斜杠分隔的正则表达式，但请注意，斜杠不是正则表达式的一部分。

正则表达式有许多变体。我们将描述扩展的正则表达式； 不同的正则表达式解析器可能仅识别其中的子集，或者对某些表达式的处理略有不同。使用在线正则表达式测试器是测试表达式并探索这些变化的便捷方法。

2.1.1. 正则表达式的基本模式

最简单的正则表达式是一系列简单字符。要搜索土拨鼠(woodchuck)，我们就键入/woodchuck/。表达式/Buttercup/匹配包含子字符串 Buttercup(蝴蝶杯)的任何字符串；具有该表达式的 grep 将返回这样一行字符“ I'm called little Buttercup ”。搜索字符串可包含单个字符(如/!/)或一个字符序列(如/urgl/)，见图 2.1。

RE	Example Patterns Matched
/woodchucks/	“interesting links to woodchucks and lemurs”
/a/	“Mary Ann stopped by Mona's”
/!/	“You've left the burglar behind again ! ” said Nori

图 2-1：一些简单的正则表达式搜索

正则表达式区分大小写：小写字母/ s /与大写字母/ S /不同(/ s /匹配小写字母 s，但不匹配大写字母 S)。这意味着模式/ woodchucks /将不匹配字符串 Woodchucks。我们可以使用方括号[ ]解决此问题。方括号内的字符串指定了要匹配的字符的析取(disjunction)。例如，图 2.2 显示模式/ [wW] /包含 w 或 W 的模式匹配。

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	“Woodchuck”
/[abc]/	‘a’,‘b’, or ‘c’	“In uomini, in soldati”
/[1234567890]/	any digit	“plenty of 7 to 5”

图 2-2：使用方括号[ ]指定字符的析取

正则表达式/ [1234567890] /指定任何一位数字。虽然数字或字母这类字符是表达式中的重要组成部分，但它们可能会很笨拙(例如，指定/ [ABCDEFGHIJKLMNOPQRSTUVWXYZ] /来表示“任何大写字母”，就显得不方便)。如果存在与一组字符关联的定义明确的序列，则可以将方括号与破折号(-)一起使用以指定**范围(rang)**中的任何一个字符。模式/ [2-5] /指定字符 2、3、4 或 5 中的任何一个。模式/ [b-g] /指定字符 b，c，d，e，f 或 g 中的一个。其他一些示例如图 2.3 所示。

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	“we should call it ‘Drenched Blossoms’ ”
/[a-z]/	a lower case letter	“my beans were impatient to be hoed!”
/[0-9]/	a single digit	“Chapter 1: Down the Rabbit Hole”

图 2-3：用方括号[ ]加破折号-来指定范围

方括号也可以通过插入符号 ^ 来指定单个字符不能使用的字符。如果插入符号 ^ 是方括号 [之后的第一个符号，则结果模式将取反。例如，模式/ [^a] /匹配除 a 之外的任何单个字符(包括特殊字符)。仅当插入符号是方括号后的第一个符号时，才如此。如果它发生在其他任何地方，通常代表插入符号。图 2.4 显示了一些示例。

RE	Match (single characters)	Example Patterns Matched
/[^A-Z]/	not an upper case letter	“Oyfn pripetchik”
/[^Ss]/	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
/[^.]/	not a period	“our resident Djinn”
/[e^]/	either ‘e’ or ‘^’	“look up ^ now”
/a^b/	the pattern ‘a^b’	“look up a^b now”

图 2-4：插入符号^表示否定或仅表示^

图注：请参见以下 re：反斜杠，以避开句点。[1]

我们该如何谈论可选元素，就像 woodchuck 和 woodchucks 中可选的 s？ 我们不能使用方括号，因为虽然方括号允许我们说“ s 或 S”，但不允许我们说“ s 或 nothing”。为此，我们使用问号/？/，它表示“前面的字符或 nothing”，如图 2.5 所示。

RE	Match	Example Patterns Matched
/woodchucks?/	woodchuck or woodchucks	“woodchuck”
/colou?r/	color or colour	“color”

图 2-5：问号？标记前一个表达式的可选性

我们可以认为问号的意思是“前一个字符的零个或一个实例”。也就是说，这是一种指定我们想要多少东西的方法，这在正则表达式中非常重要。例如，考虑某些绵羊(sheep)的语言，该语言由类似于以下内容的字符串组成：

baa!

baaa!

baaaa!

baaaaa! . . .

该语言由一个 b 后跟至少两个 a 的字符串组成，再加上感叹号。允许我们说“某些数量的 a”之类的一组操作符是基于星号(*)的，通常称为 **Kleene ***(发音“cleany star”)。Kleene * 表示“前一个字符或正则表达式出现零次或多次”。因此，/ a * /表示“任何零个或多个 a 的字符串”。这将匹配 a 或 aaaaaa，但也将匹配 Off Minor，因为字符串 Off Minor 的 a 为零。因此，匹配一个或多个 a 的正则表达式为/ aa * /，表示一个 a 后跟零个或多个 a 。也可以重复更复杂的模式。因此，/ [ab] * /的意思是“零个或多个 a 或 b”(不是“零个或多个右方括号”)。这将匹配 aaaa 或 ababab 或 bbbb 之类的字符串。为了指定多个数字(可用于查找价格)，我们可以扩展/ [0-9] /，这是单个数字的正则表达式。因此，整数(数字字符串)为/ [0-9] [0-9] * /。(为什么不只是/ [0-9] * /？)

有时必须为数字写两次正则表达式很烦人，因此可以使用一种较短的方法来指定某个字符的“至少一个”。这就是 **Kleene +**，表示“前一个字符或正则表达式出现一次或多次”。因此，表达式/ [0-9] + /是指定“数字序列”的正常方法。因此，有两种方法可以指定绵羊语言：/ baaa *！ /或/ baa +！ /。一个非常重要的特殊字符是句号(/. /)，它是一个与任何单个字符(回车符除外)匹配的通配符表达式，如图 2.6 所示。

RE	Match	Example Matched
/beg.n/	any character between beg and n	begin, beg’n, begun

图 2-6：用句号指定任何字符

[1] 译者注：为了对图进行题注，将原书中较长的说明文本，分成两部分：第一部分是简短的摘要，当作图的标签；第二部分是图注，用 5 号仿宋字体编辑，放在图的下方。

通配符通常与 Kleene 星一起使用，表示“任何字符串”。例如，假设我们要查找某个特定单词(例如 aardvark)出现两次的任意行(line)，那么我们可以使用正则表达式/aardvark.*aardvark/进行指定。

定位符(anchors)是将正则表达式定位到字符串中特定位置的特殊字符。最常见的定位符是插入符号^和美元符号$。插入符号^与行的开头匹配。模式/ ^The /与仅在行首的单词 The 匹配。因此，插入符号^具有三种用途：匹配行的开头，表示方括号内的否定，仅表示插入符号。(什么是允许 grep 或 Python 知道给定插入符号应该具有的功能的上下文？)美元符号$与行尾匹配。因此模式$是匹配行尾空格的有用模式，/ ^The dog \. $ / 匹配仅包含短语 The dog 的行。(我们必须在这里使用反斜杠，因为我们希望 . 表示“句点”而不是通配符。)

RE	Match
^	start of line
\$	end of line
\b	word boundary
\B	non-word boundary

图 2-7：正则表达式中的定位符

另外还有两个定位符：\ b 匹配单词边界，而\ B 匹配非边界。因此，/ \ bthe \ b /与单词 the 匹配，但与其他单词不匹配。从技术上讲，将用于正则表达式的“单词”定义为数字、下划线或字母的任何序列； 这是基于编程语言中“单词”的定义。例如，/ \ b99 \ b /将匹配“There are 99 bottles of beer on the wall ”中的字符串 99(因为 99 跟在空格后面)，而不匹配“There are 299 bottles of beer on the wall”中的字符串 99(因为 99 跟在数字 2 后面)。但它将与$ 99 中的 99 匹配(因为 99 跟在美元符号$后面，而不是跟在数字、下划线或字母后面)。

2.1.2. 析取，分组和优先

假设我们需要搜索有关宠物的文字； 也许我们对 cat 和 dog 特别感兴趣。在这种情况下，我们可能要搜索字符串 cat 或 dog。由于我们不能使用方括号来搜索“cat or dog”(为什么我们不能说/ [catdog] /？)，因此我们需要一个新的操作符，**析取(disjunction)**操作符，也称为管道符号 | 。模式 / cat | dog /与字符串 cat 或字符串 dog 匹配。有时我们需要在更大的序列中使用这个析取操作符。例如，假设我要为我的堂兄 David 搜索有关宠物鱼的信息。如何同时指定 guppy 和 guppies？我们不能简单地说/ guppy | ies /，因为那只会匹配字符串 guppy 和 ies。这是因为像 guppy 这样的序列**优先(precedence)**于析取操作符 | 。为了使析取操作符仅适用于特定模式，我们需要使用两个括号(parenthesis)[2]操作符：一个是左括号(，另一个是右括号)。将模式括在括号中，可以使它像单个字符一样作用于相邻操作符，例如管道(pipe)符号 | 和 Kleene *。因此，模式/ gupp(y | ies)/指定的意思是析取仅适用于后缀 y 和 ies。

左括号操作符(在我们使用像 Kleene *这样的计数器时也很有用。与 | 操作符不同，默认 Kleene * 操作符仅适用于单个字符，而不适用于整个序列。也许我们有一行，其行标签的格式为 Column 1 Column 2 Column3。表达式/ (Column ␣ [0-9] + ␣ *)/将不匹配任何数量的列； 相反，它将匹配单个列，后跟任意数量的空格！ 这里的星号仅适用于其前面的空格 ␣，而不适用于整个序列。使用括号，我们可以编写表达式/(Column ␣ [0-9] + ␣ *)*/以匹配单词 Column，后跟数字和可选空格，整个模式重复零次或多次。

一个操作符可能优先于另一个操作符的思想(要求我们有时使用括号来指定我们的意思)，由正则表达式的**操作符优先级(operator precedence)**层次结构做了形式化。下表给出了正则表达式操作符优先级的顺序，从最高优先级到最低优先级。

括号	( )
计数器	* + ? { }
序列和定位符	the ^my end$
析取	|

[2] 译者注：parenthesis 是圆括号，简称括号。后文提及的括号基本默认是圆括号。

因此，由于计数器的优先级高于序列，/ the * /与 theeeee 匹配，但与 thethe 不匹配。因为序列的优先级高于析取，所以/ the | any /匹配 the 或 any 而不是 thany 或 theny。

模式可能会以另一种方式变得模棱两可。当与文本“once upon a time"进行匹配时，请考虑表达式/ [a-z] * /。由于/ [a-z] * /匹配零个或多个字母，因此该表达式不能匹配任何内容，或者仅匹配第一个字母 o、on、onc 或 once。在这些情况下，正则表达式始终匹配其可以匹配的最大字符串；我们说模式是**贪心的(greedy)**，不断扩展以覆盖尽可能多的字符串。

但是，可以使用限定符 ？的另一种含义来强制执行非贪心匹配。

操作符 *** ?** 是一个 Kleene 星号，它匹配的文字越少越好。

操作符 **+?** 是一个 Kleene 加号，它匹配的文字越少越好。

2.1.3. 一个简单例子

假设我们想编写一个 RE 案例来查找英语文章中 the。一个简单(但不正确)的模式可能是：

/the/

一个问题是，该模式在句子开头时会漏掉单词，因此将其大写(即 The)。这可能会导致进入以下模式：

/[tT]he/

但是，我们仍然会错误地返回包含其他字词的文本(例如，other 或 theology)。因此，我们需要指定我们希望实例的两面都带有单词边界：

/\b[tT]he\b/

假设我们想在不使用/ \ b /的情况下执行此操作。我们可能想要这样做，因为/ \ b /不会将下划线和数字视为单词边界； 但是我们可能想在某些情况下找到“ the”，而在附近也可能有下划线或数字(the_或 the25)。我们需要指定我们想要的实例中，“ the”的两边都没有字母：

/[^a-zA-Z][tT]he[^a-zA-Z]/

但是，这种模式还有一个问题：在开始一行时，找不到“ the”一词。这是因为正则表达式[^a-zA-Z](我们用来避免嵌入“ the”的实例)意味着在“ the”之前必须有一些单个(尽管不是字母)字符。我们可以通过指定在“ the”之前需要行首字符或非字母字符，并在行尾使用相同字符来避免这种情况：

/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/

我们刚经历的过程基于修复两种错误：

假阳性(False positives)，我们错误匹配的字符串，例如“other”或“there”；

假阴性(False negatives)，我们错误漏掉的字符串，例如“The”。

在实现语音和语言处理系统时，一次又一次地解决这两种错误。因此，降低应用程序的总体错误率涉及两个反作用的努力：

•提高精度(最大程度减少假阳性)

•增加召回率(最大限度地减少假阴性)

我们将在第 4 章中更精确地定义精度和召回率。

2.1.4. 更多操作符

图 2.8 显示了一些通用范围的别名，它们可以主要用于保存键入。除了 Kleene *和 Kleene +，我们还可以使用显式数字作为计数器，方法是将它们括在大括号中。正则表达式/ {3} /的意思是“前一个字符或表达式恰好出现 3 次”。因此，/ a\.{24}z /将匹配“a”，再跟 24 个点，最后跟 z。

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

图 2-8：常见字符集的别名

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	n occurrences of the previous char or expression
{n,m}	from n to m occurrences of the previous char or expression
{n,}	at least n occurrences of the previous char or expression
{,m}	up to m occurrences of the previous char or expression

图 2-9：用于计数的正则表达式操作符

也可以指定数字范围。因此，/ {n，m} /指定从 n 到 m 的前一个字符或表达式的出现，/ {n，} /表示前一个表达式至少出现 n 次。用于计数的 RE 汇总在图 2.9 中。

最后，某些特殊字符通过基于反斜杠(\)的特殊符号来引用(请参见图 2.10)。其中最常见的是**换行(newline)**符\ n 和制表符\ t。要引用本身特殊的字符(例如 .,*, [,and\)，请在它们前面加反斜杠(例如 ,/ \ ./, / \ * /, / \ [/, and / \\ /)。

RE	Match	First Patterns Matched
*	an asterisk “*”	“K*A*P*L*A*N”
\.	a period “.”	“Dr. Livingston, I presume”
\?	a question mark	“Why don’t they come and lend a hand?”
\n	a newline	
\t	a tab	

图 2-10：一些需要反斜杠的字符

2.1.5. 更复杂的例子

让我们尝试一个 RE 威力的重要示例。假设我们要构建一个应用程序来帮助用户购买 Web 上的计算机。用户可能希望“任何具有至少 6 GHz 频率和 500 GB 磁盘空间且价格低于$ 1000 的机器”。要进行这种检索，我们首先需要能够查找 6 GHz 或 500 GB 或 Mac 或$ 999.99 的表达式。在本节的其余部分，我们将为该任务制定一些简单的正则表达式。

首先，让我们完成价格的正则表达式。这是一个美元符号的 RE，后跟一串数字：

/$[0-9]+/

注意，这里的$字符与我们前面讨论的行尾函数具有不同的功能。大多数正则表达式解析器都很聪明，足以意识到这里的$并不意味着行尾。(作为一个思想实验，请考虑正则表达式解析器如何从上下文中了解$的功能。)

现在我们只需要处理小数美元。之后，我们将添加小数点和两位数：

/$[0-9]+\.[0-9][0-9]/

此模式仅允许$ 199.99，但不允许$ 199。我们需要使美分成为可选，并确保我们处于字边界：

/(^|\W)$[0-9]+(\.[0-9][0-9])?\b/

最后一招！ 这种模式允许像$ 199999.99 这样的价格，价格太昂贵了！ 我们需要限制美元：

/(^|\W)$[0-9]{0,3}(\.[0-9][0-9])?\b/

磁盘空间如何？ 我们将需要再次允许可选部分(5.5 GB)； 注意：用 ？使最后的 s 可选，用/ * /表示“零个或多个空格”，因为周围总会有多余的空格：

/\b[0-9]+(\.[0-9]+)? *(GB|[Gg]igabytes?)\b/

留给读者练习：修改上述正则表达式，使其仅匹配大于 500 GB 的情况。

2.1.6. 替代，捕获组和 ELIZA

正则表达式的重要用途是**替换(substitution)**。例如，Python 和 Unix 命令(如 vim 或 sed)中使用的替

换操作符 s / regexp1 / pattern /允许将以正则表达式为特征的字符串替换为另一个字符串：

```
s/colour/color/
```

能够引用匹配第一个模式的字符串的特定子部分通常很有用。

假设我们要在文本中的所有整数周围放置尖括号，例如，将 the 35 boxes 更改为 the <35> boxes。我们希望有一种方法来引用找到的整数，以便我们可以轻松地添加方括号。为此，我们在第一个模式周围加上圆括号，并在第二个模式中使用数字操作符\ 1 进行引用，形式如下：

```
s/([0-9]+)/<\1>/
```

括号和数字操作符还可以指定某个字符串或表达式在文本中必须出现两次。例如，假设我们正在寻找“the Xer they were, the Xer they will be”的模式，我们希望将两个 X 约束为相同的字符串。为此，将第一个 X 括在括号操作符中，然后将第二个 X 替换为数字操作符\ 1，如下所示：

```
/the (.*)er they were, the \1er they will be/
```

在这里，\ 1 将替换为括号中与第一项匹配的任何字符串。因此，这将匹配“the bigger they were, the bigger they will be”，但是不匹配“the bigger they were, the faster they will be”。

使用括号将模式存储在内存中称为捕获组(capture group)。每次使用捕获组时(即，括号括起一个模式)，结果匹配项都存储在一个编号寄存器(Register)中。如果匹配两个不同的括号，\ 2 表示匹配第二个捕获组的任何内容。从而

```
/the (.*)er they (.*), the \1er we \2/
```

将匹配“the faster they ran, the faster we ran”，但不匹配“the faster they ran, the faster we ate”。同样，第三个捕获组存储在\ 3 中，第四个捕获组存储在\ 4 中，依此类推。

因此，括号在正则表达式中具有双重功能。它们用于对术语进行分组，以指定操作符应采用的顺序，并且它们用于捕获寄存器中的内容。有时，我们可能希望使用括号进行分组，但不想在寄存器中捕获结果模式。在这种情况下，我们使用一个非捕获组，该组是通过将命令？：放在打开的括号后，以(？：pattern)形式指定的。

```
/(?:some|a few) (people|cats) like some \1/
```

可以匹配“some cats like some cats”，但不能匹配“some cats like some a few”。

替换和捕获组在实现简单的聊天机器人(如 ELIZA)中非常有用(Weizenbaum，1966)。回想一下，ELIZA 通过进行如下对话来模拟罗杰氏心理学家：

```
用户1：男人们都一样。
ELIZA1：以什么方式不一样？
用户2：他们总是在骚扰我们。
ELIZA2：您能否想到一个具体的例子。
用户3：好，我的男朋友让我来这里。
ELIZA3：您的男朋友来了。
用户4：他说我很多时候都很沮丧。
ELIZA4：很抱歉听到您感到沮丧。
```

ELIZA 的工作方式是使用一系列或级联的正则表达式替换，每个替换都匹配和更改输入行的某些部分。输入行首先是大写的。然后，第一个替换将“MY”的所有实例更改为“YOUR”，将“I'M”更改为“YOU ARE”，依此类推。下一组替换匹配并替换输入中的其他模式。这里有些例子：

```
s/.* I'M (depressed|sad) .      */I AM SORRY TO HEAR YOU ARE \1/
s/.* I AM (depressed|sad) .     */WHY DO YOU THINK YOU ARE \1/
s/.* all .                      */IN WHAT WAY/
s/.* always .                   */CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

由于可以将多个替换应用于给定输入，因此将为替换分配一个等级并按顺序应用。创建模式是练习 2.3 的主题，我们将在第 24 章返回 ELIZA 体系结构的详细信息。

2.1.7. 前瞻断言

最后，有时我们需要预测未来：在文本中向前看，看是否有某些模式匹配，但不提前匹配光标(cursor)，以便我们可以在模式出现时对其进行处理。

这些**前瞻(lookahead)**断言使用上一节中针对非捕获组使用的 (？句法。如果模式出现，则操作符(？= pattern)为 true，但是为**零宽度(zero-width)**，即匹配指针不前进 。操作符(？！pattern)仅在模式不匹配时返回 true，但再次为零宽度且不使光标前移。当我们解析某些复杂的模式但要排除特殊情况时，通常使用负向前瞻。例如，假设我们要在一行的开头匹配任何以“Volcano”开头的单词。我们可以使用负向前瞻进行此操作：

/^(?!Volcano)[A-Za-z]+/

2.2. 单词

在讨论处理单词之前，我们需要确定什么才是单词。让我们首先看一个特定的**语料库(corpus，其复数为 corpora)**，它是计算机可读的文本或语音集合。例如，布朗（Brown）语料库是一百万个单词的样本集，这些样本来自 500 种不同体裁(报纸，小说，非小说，学术等)的英语书面文本，于 1963–64 年在布朗大学完成集合(Kucera and Francis,1967)。下面的布朗句子中有多少个单词？

He stepped out into the hall, was delighted to encounter a water brother.

如果我们不将标点符号视为单词，则此句子包含 13 个单词；如果我们将标点符号视为单词，则包含 15 个单词。我们是否将句点(“.”)，逗号(“,”)等视作单词取决于任务。标点符号对于查找事物的边界(逗号，句号，冒号)和识别含义的某些方面(问号，感叹号，引号)至关重要。对于某些任务，例如词类标记或语法分析或语音合成，有时我们将标点符号视为单独的单词。

电话总机(Switchboard）语料库收集了 1990 年代初陌生人之间的美式英语电话会话;它包含 2430 个会话，每个会话平均 6 分钟，总计 240 个小时的语音和大约 300 万个单词(Godfrey 等人，1992)。这样的口语语料没有标点符号，但是在定义单词方面确实带来了其他麻烦。看看电话总机的**语段(utterance)**，它是句子的口头相关词：

I do uh main- mainly business data processing

这种语段有两种**不流利(disfluency)**。断开的单词 main-称为片段，诸如 uh 和 um 之类的单词称为填充符或**填充停顿符(filled pause)**。我们应该把这些当作词语吗？同样，这取决于应用程序。如果我们要建立语音转录系统，我们可能最终希望去除不流利。

但是我们有时也会保持不流利。像 uh 或 um 这样的不流利，在语音识别中对预测即将到来的单词是有帮助的，因为它们可能表明说话者正在重新启动从句或构想，因此对于语音识别，它们被视为常规单词。因为人们使用不同的不流利，它们也可能是识别说话者的线索。实际上，Clark 和 Fox Tree(2002)表明，uh 和 um 有不同的含义。您认为他们是什么？

大写的符记 They 和非大写的符记 they 是一个词吗？在某些任务(语音识别)中，它们被集中在一起，而对于词类(POS)或命名实体标记(NET)，大写是一个有用的特性并被保留下来。

cats 和 cat 之类的变形形式怎么样？ 这两个词具有相同的词元 cat，但具有不同的词形。**词元(lemma)**是一组具有相同词干、相同主要词类和相同词义的词汇形式。**词形(wordform)**是单词的全部变形或派生形式。对于形态复杂的语言(例如阿拉伯语)，我们经常需要处理词形还原。但是，对于许多英语任务来说，词形就足够了。

英语里有多少个单词？要回答这个问题，我们需要区分两种谈论单词的方式。**类型(types)**是语料库中不同单词的数量；如果词汇表中的单词集为 V，则类型数量为词汇表大小| V |。**符记(token)**是运行(running)单词的总数 N。如果我们忽略标点符号，则下面的布朗句子有 16 个符记和 14 种类型：

They picnicked by the pool, then lay back on the grass and looked at the stars.

当我们谈论语言中的单词数时，通常是指单词类型。

图 2.11 显示了从一些流行的英语语料库中计算出的类型和符记的大致数量。我们查看的语料库越大，

我们发现的类型越多，实际上类型数量| V |和符记数量 N 之间的这种关系，以其发现者(分别在语言学和信息检索中)命名为 Herdan 定律(Herdan 1960)或 Heaps 定律(Heaps 1978)。它显示在等式 2.1 中，其中 k 和 β 为正常数，且 0 < β <1。

$$|V|=kN^{\beta} \tag{2.1}$$

β的值取决于语料库的大小和体裁，但至少对于图 2.11 中的大语料库，β的范围是 0.67 至 0.75。大概可以说，文本的词汇量增长速度明显快于单词长度的平方根。

语言中单词数量的另一种度量是词元数量而不是类型。词典可以帮助确定词元数量。词典条目或黑体字形式是词元数量的上限(因为某些词元具有多种黑体字形式)。1989 版的牛津英语词典有 615,000 个条目。

Corpus	Tokens = N	Types = |V|
Shakespeare	884 thousand	31 thousand
Brown corpus	1 million	38 thousand
Switchboard telephone conversations	2.4 million	20 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13 million

图 2-11：某些英语语料库的类型和符记的大致数量

图注：最大的 Google N-grams 语料库包含 1300 万个类型，其中仅包含出现≥40 次的类型，因此真实数字会更大。

2.3.语料库

文字不是凭空出现的。我们学习的任何一段文字是由一个或多个特定的讲话者或作家，用特定语言的特定方言，在特定的时间，特定的地点，用于特定的功能。

语言变化可能是最重要的方面。当 NLP 算法适用于多种语言时，它们最有用。根据在线民族志目录(Simons 和 Fennig，2018)，撰写本文时，世界上共有 7097 种语言。在多种语言上测试算法非常重要，尤其是在具有不同属性的语言上；相比之下，目前不幸的是 NLP 算法仅以英语开发或测试的趋势(Bender，2019)。即使算法的开发超出英语，它们也倾向于针对大型工业化国家的官方语言开发(中文，西班牙文，日文，德文等)，但我们不想将工具限制为仅这几种语言。此外，大多数语言还具有多种变体，经常在不同地区或不同社会群体中使用。因此，例如，如果我们正在处理的文本使用**非裔美国语言(African American Language，AAL)**的特征，则我们必须使用具有这些变体特征的 NLP 工具。Twitter 帖子可能使用非裔美国人演讲者经常使用的功能，例如 iont(即**主流美国英语(Mainstream American English，MAE)**中的 I don't)之类的构造，或 talmbout 相对应主流美国英语中的 talk about,都是两个影响分词的示例(Blodgett 等人 2016，Jones 2015)。

演讲者或作家在一次交流中使用多种语言也很普遍，这种现象称为**代码转换(code switching)**。代码转换在世界范围内极为普遍。这是显示西班牙语和(音译)印地语与英语的代码转换的示例(Solorio 等人 2014，Jurgens 等人 2017)：

(2.2) Por primera vez veo a @username actually being hateful! it was beautiful:)

[For the first time I get to see @username actually being hateful! it was beautiful:)]

(2.3) dost tha or ra- hega ... dont wory ... but dherya rakhe

["he was and will remain a friend ... don ' t worry ... but have faith "]

变化的另一个维度是**体裁(genre)**。我们的算法必须处理的文本可能来自新闻专线、小说或非小说类书籍、科学文章、维基百科(Wikipedia)或宗教文本。它可能来自于口头类型，如电话谈话、商务会议、警察穿戴式摄像机、医疗采访或电视节目或电影的文本。它可能来自工作场合，比如医生的笔记、法律文本或议会或国会程序。

文本也反映了作者(或说话者)的人口统计学特征:他们的年龄、性别、种族、社会经济阶层都可以影响我们正在处理的文本的语言属性。

最后，时间也很重要。语言会随时间而变，对于某些语言，我们拥有来自不同历史时期的大量文本。

由于语言的位置如此，在开发用于从语料库进行语言处理的计算模型时，重要的是要考虑谁在什么背景下，出于什么目的制作了语言。数据集(dataset)的用户如何知道所有这些详细信息？ 最好的方法是使语料库创建者为每个语料库建立一个**数据表(datasheet)**(Gebru 等人 2020)或数据声明(statement)(Bender 和 Friedman 2018)。数据表指定数据集的属性，例如：

动机：为什么要收集语料库、由谁收集、由谁提供资金？

情况：在什么时候、什么情况下写/说过文本？例如，有任务吗？语言最初是口语对话，编辑文本，社交媒体交流，独白还是对话？

语言种类：语料库使用哪种语言(包括方言/地区)？

演讲者个人背景：例如，文字作者的年龄或性别是什么？

收集过程：数据有多大？如果是子样本，则如何采样？是否经同意收集数据？数据是如何进行预处理的，可以使用哪些元数据？

注释过程：注释是什么，注释者的个人背景是什么，他们是如何训练的，数据是如何注释的？

发行：有版权或其他知识产权限制吗？

2.4. 文本规范化

在对文本进行几乎任何自然语言处理之前，必须对文本进行规范化。通常，至少要执行三个任务，作为任何规范化过程的一部分：

1.符记化(分段)单词

2.规范单词格式

3.分割句子

在下一节中，我们将逐步完成所有这些任务。

2.4.1. 用于粗糙符记化和规范化的 Unix 工具

让我们从一个简单的、甚至有点天真的单词标记化和规范化(以及频率计算)的版本开始，受 Church(1994)的启发，它只能在一个 UNIX 命令行中用英语完成。我们将使用一些 Unix 命令：tr，用于系统地更改输入中的特定字符； sort，按字母顺序对输入行进行排序； 和 uniq，它折叠并计算相邻的相同行。

例如，让我们从一个文本文件 sh.txt 中的莎士比亚“complete words”开始。通过将非字母字符的每个序列更改为换行符，我们可以使用 tr 符记化单词(“ A-Za-z”表示字母，-c 选项补全为非字母，-s 选项压缩所有字母) 序列成一个字符)：

```
tr -sc'A-Za-z' '\n' < sh.txt
```

该命令的输出将是：

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

现在每行有一个单词，我们可以对行进行排序，然后将它们传递给 uniq -c，它将折叠并计算它们：

```
tr -sc 'A-Za-z' '\n' < sh.txt | sort | uniq -c
```

具有以下输出：

```
1945    A
72      AARON
19      ABBESS
25      Aaron
6       Abate
1       Abates
5       Abbess
6       Abbey
3       Abbot
...
```

或者，我们可以将所有大写字母折叠为小写字母：

```
tr -sc 'A-Za-z' '\n' < sh.txt | tr A-Z a-z | sort | uniq -c
```

其输出是：

```
14725   a
97      aaron
1       abaissiez
10      abandon
2       abandoned
2       abase
1       abash
14      abate
3       abated
3       abatement
...
```

现在，我们可以再次排序以查找常见单词。

-n 选项的排序方式是按数字而不是字母顺序排序；

-r 选项的意思是按相反顺序(从最高到最低)排序：

```
tr -sc 'A-Za-z' '\n' < sh.txt | tr A-Z a-z | sort | uniq -c | sort -n -r
```

结果表明，与其他语料库一样，莎士比亚语中最常见的单词是短功能单词，例如冠词，代词，介词：

```
27378   the
26084   and
22538   i
19771   to
17481   of
14725   a
13826   you
12489   my
11318   that
11112   in
...
```

此类 Unix 工具可以非常方便地为任何语料库建立快速的字数统计。

2.4.2. 单词符记化

上面的简单 UNIX 工具可以很好地获取单词的粗略统计信息，但是对于**符记化(tokenization)** (将运行中的文本分割成单词的任务)来说，通常更复杂的算法是必需的。

尽管 Unix 命令序列仅删除了所有数字和标点符号，但对于大多数 NLP 应用程序，我们需要将其保留在标记中。我们经常想将标点符号作为一个单独的符记来分离；逗号是解析器的有用信息，句点有助于指示句子边界。但是我们经常希望保留内部出现的标点符号，例如 m.p.h.，Ph.D.，AT&T 和 cap'n。特殊字符和数字将需要保留在价格($ 45.55)和日期(06/02/01)中；我们不想将该价格细分为“45”和“55”的单独符记。还有 URL(http://www.stanford.edu)，Twitter 标签(#nlproc)或电子邮件地址(someone@cs.colorado.edu)。

数字表达式也引入了其他复杂的情况;

虽然逗号通常出现在单词边界，但在英语中，逗号用于数字内部，每三个数字使用一次:555,500.50。在这一点上，语言以及因此而产生的符记化要求是不同的;相比之下，许多欧洲大陆语言，如西班牙语、法语和德语，使用逗号来标记小数点，在英语中放置逗号的地方使用空格(有时是句号)，例如，555 500,50。

标记器还可以用省略符号来扩展**附着词(clitic)**的缩略形式，例如，将“what're”转换为两个符记“what are”，将“we're”转换为两个字符“we are”。附着词是一个单词的一部分，不能独立存在，只能在附加到另一个单词时出现。一些这样的缩略以其他字母语言出现，包括法语的文章和代词(j'ai，l'homme)。

取决于应用程序，符记化算法还可以将多词表达式(例如 New York 或 rock'n'roll)符记化为单个标记，这需要某种类型的多词表达字典。因此，符记化与命名实体识别，检测名称，日期和组织的任务密切相关(第 8 章)。

一种常用的符记化标准称为**宾州树库(Penn Treebank)**符记化标准，用于语言数据联盟(LDC)发布的已解析的语料库(树库)，该语言库是许多有用数据集的来源。该标准将附着词分隔开(doesn't 变成 does n't)，将带连字符的单词保持在一起，并分隔所有标点符号(为了节省空间，我们在符记之间显示可见的空格‘␣’，尽管换行是更常见的输出)：

Input: "The San Francisco-based restaurant," they said,
"doesn't charge $10".

Output: " ␣ The ␣ San ␣ Francisco-based ␣ restaurant ␣ , ␣ " ␣ they ␣ said ␣ ,
" ␣ does ␣ n't ␣ charge ␣ $ ␣ 10 ␣ " ␣

实际上，由于符记化需要在任何其他语言处理之前运行，因此它必须非常快。因此，符记化的标准方法是使用基于正则表达式的确定性算法，这些正则表达式被编译为非常有效的有限状态自动机。例如，图 2.12 显示了一个基本正则表达式的示例，该示例可用于基于 Python 的自然语言工具包(NLTK)的 nltk.regexp 符记化功能进行符记化(Bird 等人 2009；http：// www.nltk.org)

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)              # set flag to allow verbose regexps
... ([A-Z]\.)+                     # abbreviations, e.g. U.S.A.
... | \w+(-\w+)*                   # words with optional internal hyphens
... | \$?\d+(\.\d+)?%?             # currency and percentages, e.g. $12.40, 82%
... | \.\.\.                       # ellipsis
... | [ ][.,;"'?():-_`]            # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

图 2-12：NLP 工具包中 RE 符记化的 Python 跟踪

图注：NLTK 基于 Python 的 NLP 工具包中 RE 符记化的 Python 跟踪。为可读性而注释;(?x)冗长标志告诉 Python 删除注释和空格。摘自 chapter 3 of Bird 等人(2009)

精心设计的确定性算法可以处理产生的歧义，例如，省略符号在用作所有格标记(如书的封面)、引号(如“The other class”)或附着词(如 they're)时需要进行不同的符记化。

在诸如中文，日文和泰文这样的语言中，单词符记化更为复杂，这些语言不使用空格来标记潜在的单词边界。例如，在中文中，单词由字符组成(中文称为**汉字(hanzi)**)。每个字符通常代表一个意思单元(称为语素)，并且可以作为一个音节发音。单词的平均长度约为 2.4 个字符。但是，决定什么才是中文中的单词是很复杂的。例如，考虑以下句子：

(2.4) 姚明进入总决赛

"Yao Ming reaches the finals"

正如 Chen 等人(2017)指出的那样，这可以被视为 3 个词("中国树库"分割):

(2.5) 姚明 进入 总决赛

YaoMing reaches finals

或作为 5 个单词("北京大学"分词):

(2.6) 姚 明 进入 总 决赛

Yao Ming reaches overall finals

最后，在中文中，完全可以完全忽略单词并使用字符作为基本元素，将句子视为一系列 7 个字符：

(2.7) 姚 明 进 入 总 决 赛

Yao Ming enter enter overall decision game

实际上，对于大多数中文 NLP 任务而言，采用字符而不是单词作为输入效果更好，因为对于大多数应用程序而言，字符处于合理的语义级别，相比之下，这导致词汇量巨大，包含大量非常罕见的单词(Li 等人 2019)。

但是，对于日语和泰语，字符的单位太小，因此需要用于**分词(word segmentation)**的算法。对于在少数情况下需要单词而不是字符边界的情况，这些功能对于中文也很有用。这些语言的标准分割算法使用经过人工分割训练集的有监督机器学习训练的神经序列模型； 我们将在第 8 章和第 9 章中介绍序列模型。

2.4.3. 符记化的字节对编码

对文本进行符记化还有第三种选择。我们可以使用数据自动地告诉我们应该使用什么符记，而不是将符记定义为单词(无论是由空格或更复杂的算法分隔)或字符(如在中文中)。这在处理未知单词(语言处理中的重要问题)时特别有用。

正如我们将在下一章中看到的那样，NLP 算法通常会从一个语料库(一个训练语料库)中学习一些关于语言的事实，然后使用这些事实来决定一个单独的测试语料库及其语言。因此，如果我们的训练语料库包含单词"low"，"new"，"newer"而不是"lower"，又如果"lower"一词出现在我们的测试语料库中，则我们的系统将不知道如何处理。

为了处理这个未知的单词问题，现代的符记器经常自动地归纳出一组符记，其中包含比单词更小的字符，称为**子单词(subword)**。子单词可以是任意的子字符串，也可以是带有含义的单位，例如词素-est 或-er。(词素是一种语言中最小的含意单元；例如，单词" unlikeliest"的词素是 un-，likely 和-est。)在现代符记化方案中，大多数符记是单词，但是某些符记是经常出现的语素或其他子词，例如-er。因此，每个看不见的单词(例如" lower")都可以由已知的子单词单元(例如 low 和 er)的某些序列表示，或者在必要时甚至表示为单个字母的序列。

大多数符记化方案有两个部分：符记学习器和符记分段器。符记学习器采用原始的训练语料库(有时通过空格将其粗略地分离成单词)，并产生一个词汇集(一组符记)。符记分段器采用原始测试语句，并将其分段为词汇表中的符记。三种算法被广泛使用：字节对编码(Sennrich 等人 2016)，一元语法(unigram)语言建模(Kudo，2018)和 WordPiece(Schuster 和 Nakajima，2012)； 还有一个 SentencePiece 库，其中包含三个算法中的前两个的实现(Kudo 和 Richardson，2018)。

在本节中，我们介绍三种中最简单的一种，即**字节对编码(BPE)**算法(Sennrich 等人 2016)，见图 2.13。

```
function BYTE-PAIR ENCODING(strings C, number of merges k) returns vocab V

V←all unique characters in C                          # initial set of tokens is characters
for i = 1 to k do                                     # merge tokens til k times
    tL, tR ←Most frequent pair of adjacent tokens in C
    tNEW ←tL + tR                                     # make new token by concatenating
    V← V + tNEW                                       # update the vocabulary
    Replace each occurrence of tL, tR in C with tNEW  # and update the corpus
return V
```

图 2-13：BPE 算法中的符记学习器部分

图注：将语料库分解成单个字符或字节，通过迭代合并符记来学习词汇。图改编自 Bostrom 和 Durrett(2020)。

BPE 符记学习器从一个词汇开始，该词汇只是所有单个字符的集合。然后，它检查训练语料库，选择最经常相邻的两个符号(例如“A”，“B”)，在词汇表中添加新的合并符号“AB”，并用新“AB”替换语料库中的每个相邻的“A”，“B”。它继续计数和合并，创建越来越长的新字符串，直到完成 k 个合并为止，创建了 k 个新颖的符记。因此，k 是算法的参数。产生的词汇表由原始字符集和 k 个新符号组成。

该算法通常在单词内部运行(不跨单词边界合并)，因此输入语料库首先用空格分隔，以给出一组字符串，每个字符串对应于一个单词的字符，再加上一个特殊的单词结尾符号_及其计数。让我们看看它在以下由 18 个单词符记组成的小输入语料库上的操作，每个单词都有计数(单词 low 出现 5 次，单词 new 出现 6 次，依此类推)，其起始词汇为 11 个字母：

corpus	vocabulary
5 l o w_	_, d, e, i, l, n, o, r, s, t, w
2 l o w e s t_	
6 n e w e r_	
3 w i d e r_	
2 n e w_	

BPE 算法首先对所有相邻符号对进行计数：最频繁的是一对 e r，因为它出现在“newer”(频率为 6)和“wider”(频率为 3)中，总共出现了 9 次[3]。然后，我们合并这些符号，将 er 视为一个符号，然后再次计数：

corpus	vocabulary
5 l o w_	_, d, e, i, l, n, o, r, s, t, w, er
2 l o w e s t_	
6 n e w er_	
3 w i d er_	
2 n e w_	

最常见的一对是 er _，我们要归并它，系统已经学习到，单词末的 er 应该有一个符记，表示为 er _:

corpus	vocabulary
5 l o w_	_, d, e, i, l, n, o, r, s, t, w, er_
2 l o w e s t_	
6 n e w er_	
3 w i d er_	
2 n e w_	

接下来的 n e(总数为 8)合并为 ne：

corpus	vocabulary
5 l o w_	_, d, e, i, l, n, o, r, s, t, w, er_ ,ne
2 l o w e s t_	
6 ne w er_	
3 w i d er_	
2 ne w_	

如果我们继续，下一个合并是:

[3] 注意，可以有联系;我们可以选择先归并 r ，因为它的频率也是 9。

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_ , low_

学习完词汇后，符记解析器将用于符记化测试语句。符记解析器仅在测试数据上运行，然后以我们学习它们的顺序贪心地从训练数据中学到了合并。(因此，测试数据中的频率不起作用，只是训练数据中的频率起作用)。因此，首先我们将每个测试句子单词分割成字符。然后我们应用第一个规则：将测试语料库中 e r 的每个实例替换为 r；然后是第二条规则：将测试语料库中 er _ 的每个实例替换为 er_，依此类推。最后，如果测试语料库包含单词“ n e w e r _”，它将被符记化为一个完整的单词。但是，一个新的(未知的)单词，如“ l o w e r _”，将被合并到 low 和 er_ 两个符记中。

当然，在实际算法中，BPE 在非常大的输入语料库上运行成千上万次合并。结果是大多数单词将被表示为完整的符号，而只有非常罕见的单词(和未知单词)才必须由它们的部分(parts)来表示。

2.4.4. 单词规范化，词形还原和词干化

单词规范化(normalization)的任务是将单词/符记置于标准格式中，为具有多种形式的单词(如 USA 和 US、或 uh-huh 和 uhhuh)选择单一标准形式。尽管在标准化过程中丢失了拼写信息，但这种标准化可能是有价值的。对于关于 US 的信息检索或信息提取，我们可能希望看到来自文档的信息，无论它们是否提到了 US 或 USA。

大小写折叠(case folding)是另一种规范化。将所有内容都映射为小写意味着 Woodchuck 和 woodchuck 被相同地表示，这对于许多任务(例如信息检索或语音识别)的概括非常有帮助。相比之下，对于情感分析和其他文本分类任务，信息提取和机器翻译，大小写可能会很有帮助，并且通常不进行大小写折叠。这是因为，保持例如“美国是国家(US the country)”和“我们是代词(us the pronoun)”之间的差异，概括地说，可能会超过为其他词提供大小写折叠的优势。

对于许多自然语言处理情况，我们还希望单词的两种形态上不同的形式表现相似。例如，在 Web 搜索中，某些人可能会键入字符串 woodchucks，但有用的系统可能还希望返回提及不带 s 的 woodchuck 的页面。这在形态复杂的语言(例如俄语)中尤其常见，例如，Moscow(莫斯科)一词在 Moscow, of Moscow, to Moscow 等词组中都有不同的结尾。

词形还原(lemmatization)是确定两个词尽管表面存在差异但具有相同词根的任务。am, are 和 is 这三个词具有共同的词元 be。dinner 和 dinners 这两个词都具有词元 dinner。将这些形式中的每一个词化为同一个词元，将使我们能够找到所有提及的俄语单词，例如莫斯科。像 He is reading detective stories 这样的句子的词元化形式就是 He be read detective story。

词形还原是如何完成？ 最复杂的词形还原方法包括单词的完整形态解析。形态学是这样一种方式的研究，在这种方式中，词是由更小的承载意义的单位——**语素(morpheme)**构成的。可以将词素分为两大类：**词干(stem)**(词的中心词素，提供主要含义)和**词缀(affix)**(添加各种“附加”含义)。因此，例如，单词 fox 由一个词素(词素 fox)组成，而单词 cats 由两个词素组成：词素 cat 和词素-s。形态学解析器将像 cats 这样的单词解析成两个语素 cat 和 s，或者将西班牙语单词 amaren(即‘if in the future they would love’)解析成语素 amar(即‘to love’)，其形态学特征为 3PL 和未来虚拟语气。

Porter(波特)词干分析器

词形还原算法可能很复杂。出于这个原因，我们有时会使用一种更简单但更粗糙的方法，主要包括去掉词尾词缀。这种朴素的形态分析版本称为**词干分析(stemming)**。最广泛使用的词干提取算法之一是 Porter(1980)。Porter **词干分析器(stemmer)**应用于如下段落:

This was not the map we found in Billy Bones’s chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written

notes.

产生以下词干输出：

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note

该算法基于一系列的重写规则，以**级联(cascade)**的形式串行运行，其中每次传递的输出作为下一个传递的输入；以下是一些规则的示例:

ATIONAL → ATE (e.g., relational → relate)
ING → ϵ if stem contains vowel (e.g., motoring → motor)
SSES → SS (e.g., grasses → grass)

有关 Porter 词干分析器的详细规则列表以及代码(使用 Java，Python 等)，可以在 Martin Porter 的主页上找到；另请参阅原始论文(Porter，1980)。当我们需要折叠同一词元的不同变体时，简单的词干分析器可能会很有用。然而，如下表所示(Krovetz，1993)，他们的确会犯概括过度和概括不足的错误。

Errors of Commission		Errors of Omission	
organization	organ	European	Europe
doing	doe	analysis	analyzes
numerical	numerous	noise	noisy
policy	police	sparse	sparsity

2.4.5. 句子分割

句子分割(sentence segmentation)是文本处理的另一个重要步骤。把一篇文章切分成句子最有用的线索就是标点符号，比如句号、问号和感叹号。问号和感叹号是比较明确的句子边界标记。另一方面，句号则更为模糊。句点字符“.”在句子边界标记和缩写标记(例如 Mr.或 Inc.)之间是有歧义的。你刚刚读到的上一个句子展示了这种歧义的一个更复杂的情况，在这个句子中，Inc.的最后一个句号同时标记了一个缩写和句子边界标记。因此，句子符记化和单词符记化可以共同解决。

通常，句子符记化方法的工作方式是首先(基于规则或机器学习)确定句点是单词的一部分还是句子边界标记。缩写字典可以帮助确定句点是否是常用缩写的一部分；字典可以是手工构建的或机器学习的(Kiss 和 Strunk，2006 年)，最后的句子拆分器也可以。例如，在 Stanford CoreNLP 工具箱中(Manning 等人，2014)，句子拆分是基于规则的，这是符记化的确定性结果。当句子结尾的标点符号(. , ! , or ?)尚未与其他字符组合成符记(例如缩写或数字)时，该句子结束(可选地，后面加上其他最终引号或括号)。

2.5.最小编辑距离

许多自然语言处理都与测量两个字符串的相似程度有关。例如，在拼写更正中，用户输入了一些错误的字符串(例如，graffe)，我们想知道用户的意思。用户可能想用一个与 graffe 类似的词。在候选的相似单词中，单词 giraffe(与 graffe 只有一个字母不同)直觉上似乎比 grail 或 graf(它们有更多的字母不同)更相似。另一个示例来自于**共指(coreference)**，即确定两个字符串(例如以下字符串)是否引用同一实体的任务：

Stanford President Marc Tessier-Lavigne

Stanford University President Marc Tessier-Lavigne

同样，这两个字符串非常相似(只有一个单词不同)，似乎是有用的证据，可以决定它们可能是相关的。

编辑距离给了我们一种方法来量化这两种关于字符串相似性的直觉。更正式地说，两个字符串之间的**最小编辑距离(minimum edit distance)**定义为将一个字符串转换为另一个字符串所需的最小编辑操作数量(如插入、删除、替换等操作)。

例如，“intention”与“execution”之间的差距为 5(删除 i，将 e 替换为 n，将 x 替换为 t，插入 c，将 u 替换为 n)。通过查看最重要的字符串距离可视化，可以更容易地看到两个字符串之间的**对齐(alignment)**，如图 2.14 所示。给定两个序列，对齐是两个序列的子串之间的对应关系。因此，我们说我与空字符串对

齐，N 与 E 对齐，依此类推。在对齐的字符串下面是另一种表示形式； 一系列表示操作列表的符号，这些操作列表用于将顶部字符串转换为底部字符串：d 表示删除，s 表示替换，i 表示插入。

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

图 2-14：表示两个字符串之间的最小编辑距离

图注：最后一行给出了将顶部字符串转换为底部字符串的操作列表：d 表示删除，s 表示替换，i 表示插入。

我们还可以为每个操作分配特定的成本或权重。两个序列之间的 Levenshtein 距离是最简单的加权因子，其中三个操作的每一个的成本为 1(Levenshtein，1966)--我们假设以字母代替其自身(例如，用 t 代替 t)的成本为零。Levenshtein 的 intention 和 execution 之间的距离为 5。Levenshtein 还提出了他的指标的替代版本，其中每个插入或删除的成本均为 1，并且不允许替换。(这等效于允许替换，但是给每个替换以 2 为代价，因为任何替换都可以用一个插入和一个删除来表示)。使用此版本，intention 和 execution 之间的 Levenshtein 距离为 8。

2.5.1. 最小编辑距离算法

我们如何找到最小的编辑距离?我们可以把这看作是一个搜索任务，在这个任务中，我们搜索从一个字符串到另一个字符串的最短路径(编辑序列)。

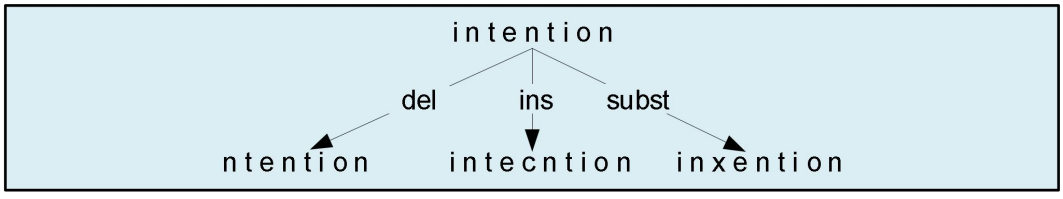


图 2-15：寻找编辑距离被视为一个搜索问题

所有可能的编辑空间都很大，因此我们不能天真地进行搜索。然而，许多不同的编辑路径最终将以相同的状态(字符串)结束，因此不必重新计算所有这些路径，我们可以在每次看到一个状态时只记住它的最短路径。我们可以通过使用**动态规划(dynamic programming)**来做到这一点。动态规划是一类算法的名称，该算法首先由 Bellman(1957)提出，该算法采用表驱动方法通过将解决方案与子问题相结合来解决问题。自然语言处理中一些最常用的算法利用了动态规划，例如 Viterbi 算法(第 8 章)和 CKY 算法进行解析(第 13 章)。

动态规划问题的直觉是，一个大问题可以通过适当地结合不同子问题的解来解决。考虑转换后的单词的最短路径，该路径代表图 2.16 中所示的字符串 intention 和 execution 之间的最小编辑距离。想象一些字符串(可能是 exention)在这个最优路径上(无论它是什么)的情景。动态规划的直觉是，如果在最佳操作列表中有 exention，则最佳序列还必须包括从 intention 到 exention 的最佳路径。为什么?如果从 intention 到 exention 有一条较短的路径，那么我们可以使用它来代替，从而导致更短的整体路径，最优序列不会是最优的，从而导致矛盾。

最小编辑距离算法被 Wagner 和 Fischer(1974)命名，但被许多人独立发现(见第 8 章的历史说明部分)。首先定义两个字符串之间的最小编辑距离。给定两个字符串，长度为 n 的源字符串 X 和长度为 m 的目标字符串 Y，我们将 D [i，j]定义为 X [1..i]和 Y [1..j]之间的编辑距离，即 X 的前 i 个字符和 Y 的前 j 个字符。因此 X 和 Y 之间的编辑距离为 D [n，m]。

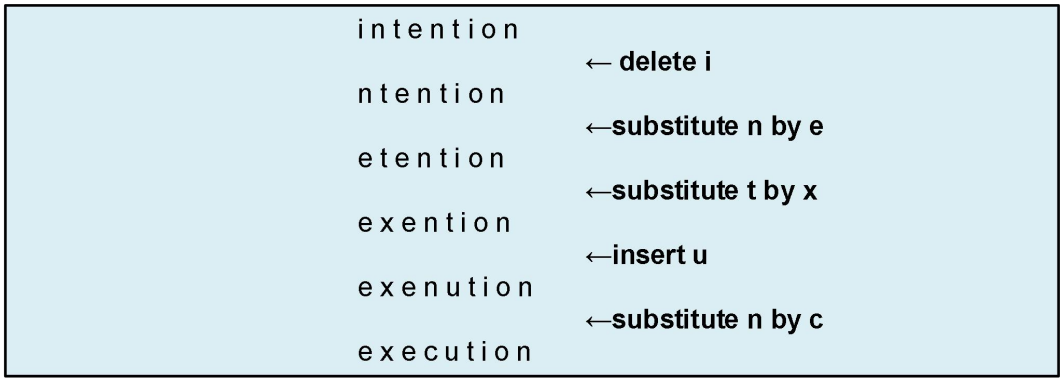


图 2-16：从 intention 到 execution 的路径

我们将使用动态规划来自底向上计算 D[n,m]，并结合子问题的解决方案。在基本用例中，源子字符串长度为 i，但目标字符串为空，则从 i 个字符到 0 个字符需要 i 个删除。如果目标子串的长度为 j，则从 0 个字符到 j 个字符需要插入 j 个字符。计算出较小的 i，j 的 D [i，j]之后，我们将根据先前计算出的较小值来计算较大的 D [i，j]。D [i，j]的值通过采用到达矩阵的三个可能路径中的最小值来计算：

$$D[i,j]=\min\begin{cases} D[i-1,j]+\text{del-cost}(source[i]) \\ D[i,j-1]+\text{ins-cost}(target[j]) \\ D[i-1,j-1]+\text{sub-cost}(source[i],target[j]) \end{cases}$$

假设 Levenshtein 距离的版本，插入和删除的成本分别为 1 (ins-cost(·)= del-cost(·)= 1)，替换的成本为 2(除了替换相同字母的成本为 0)，则 D[i, j]的计算为:

$$D[i,j]=\min\begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1]+\begin{cases} 2; \text{ if } source[i]\neq target[j] \\ 0; \text{ if } source[i]= target[j] \end{cases} \end{cases} \tag{2.8}$$

算法总结如图 2.17 所示;图 2.18 是公式 2.8 中 Levenshtein 版本应用该算法计算从 intention 到 execution 距离的结果。

```
function MIN-EDIT-DISTANCE(source, target) returns min-distance
  n←LENGTH(source)
  m←LENGTH(target)
  Create a distance matrix distance[n+1,m+1]

  # Initialization: the zeroth row and column is the distance from the empty string
  D[0,0] = 0
      for each row i from 1 to n do
          D[i,0]←D[i-1,0] + del-cost(source[i])
      for each column j from 1 to m do
          D[0,j]←D[0,j-1] + ins-cost(target[j])
      # Recurrence relation:
          for each row i from 1 to n do
              for each column j from 1 to m do
                  D[i, j]←MIN( D[i-1,j] + del-cost(source[i]),
                               D[i-1,j-1] + sub-cost(source[i], target[j]),
                               D[i,j-1] + ins-cost(target[j]))
  # Termination
  return D[n,m]
```

图 2-17：最小编辑距离算法

图注：这是动态规划算法类别的示例。各种费用可以是固定的(例如∀x，ins-cost(x)= 1)，也可以特定于字母(以模拟某些字母比其他字母更容易插入的事实)。我们假设以字母代替自身没有成本(sub-cost(x，x)= 0)。

知道最小编辑距离对于查找潜在的拼写错误更正等算法很有用。但是编辑距离算法在另一方面也很重要；通过一个小的改变，它还可以在两个字符串之间提供最小的成本对齐。对齐两个字符串在整个语音和语言处理中很有用。在语音识别中，最小编辑距离对齐用于计算单词错误率(第 26 章)。对齐在机器翻译中扮演重要角色，其中平行语料库(带有两种语言的文本的语料库)中的句子需要彼此匹配。

要扩展编辑距离算法以产生对齐，我们可以首先将对齐可视化为通过编辑距离矩阵的路径。图 2.19 用加粗的单元格显示了该路径。每个加粗的单元格表示两个字符串中一对字母的对齐方式。如果两个加粗的单元格出现在同一行中，则从源到目标的过程中会有插入;同一列中两个加粗的单元格表示删除。

Src\Tar	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	7	8	7
t	3	4	5	6	7	8	7	8	9	8
e	4	3	4	5	6	7	8	9	10	9
n	5	4	5	6	7	8	9	10	11	10
t	6	5	6	7	8	9	8	9	10	11
i	7	6	7	8	9	10	9	8	9	10
o	8	7	8	9	10	11	10	9	8	9
n	9	8	9	10	11	12	11	10	9	8

图 2-18：计算从 intention 到 execution 之间的最小编辑距离

图注：用图 2.17 的算法，计算 Levenshtein 距离，插入或删除的成本为 1，替换的成本为 2。

图 2.19 还显示了如何计算此对齐路径的直觉。计算分两步进行。第一步，我们扩大最小编辑距离算法，以在每个单元格中存储反向指针。单元格的反向指针指向当前单元格的前一个(或多个)单元格。我们已经在图 2.19 中展示了这些反向指针的原理图。有些单元格有多个向后指针，因为最小扩展可能来自以前的多个单元格。第二步，我们执行一个**回溯(back trace)**。在回溯中，我们从最后一个单元格(最后一行和最后一列)开始，沿着指针返回动态规划矩阵。在最终单元格和初始单元格之间的每个完整路径都是最小距离对齐。练习 2.7 要求您修改最小编辑距离算法以存储指针并计算回溯以输出对齐。

	#	e	x	e	c	u	t	i	o	n
#	0	←1	←2	←3	←4	←5	←6	←7	←8	←9
i	↑1	↖←↑2	↖←↑3	↖←↑4	↖←↑5	↖←↑6	↖←↑7	↖6	←7	←8
n	↑2	↖←↑3	↖←↑4	↖←↑5	↖←↑6	↖←↑7	↖←↑8	↑7	↖←↑8	↖7
t	↑3	↖←↑4	↖←↑5	↖←↑6	↖←↑7	↖←↑8	↖7	←↑8	↖←↑9	↑8
e	↑4	↖3	←4	↖←5	←6	←7	←↑8	↖←↑9	↖←↑10	↑9
n	↑5	↑4	↖←↑5	↖←↑6	↖←↑7	↖←↑8	↖←↑9	↖←↑10	↖←↑11	←↑10
t	↑6	↑5	↖←↑6	↖←↑7	↖←↑8	↖←↑9	↖8	←9	←10	←↑11
i	↑7	↑6	↖←↑7	↖←↑8	↖←↑9	↖←↑10	↑9	↖8	←9	←10
o	↑8	↑7	↖←↑8	↖←↑9	↖←↑10	↖←↑11	↑10	↑9	↖8	←9
n	↑9	↑8	↖←↑9	↖←↑10	↖←↑11	↖←↑12	↑11	↑10	↑9	↖8

图 2-19：用反向指针计算对齐路径的原理图

图注：当在每个单元格中输入一个值时，我们用最多三个箭头标记三个相邻的单元格中的哪一个。在表满了之后，我们通过回溯来计算对齐(最小编辑路径)，从右下角的 8 开始，沿着箭头往回走。粗体单元格序列表示两个字符串之间的一种可能的最小成本对齐方式。根据 Gusfield(1997)之后的图表设计。

当我们用简单的 Levenshtein 距离来处理我们的例子时，图 2.17 中的算法允许操作上的任意权重。例如，为了纠正拼写错误，替换更可能发生在键盘上相邻的字母之间。**维特比(Viterbi)**算法是最小编辑距离的概率扩展。维特比不是计算两个字符串之间的“最小编辑距离”，而是计算一个字符串与另一个字符串的“最大概率对齐”。我们将在第 8 章对此进行更多的讨论。

2.6. 总结

本章介绍了语言处理中的一个基本工具，**正则表达式**，并展示了如何执行基本的**文本规范化**任务，包括**分词**和**规范化**、**句子分割**和**词干提取**。我们还介绍了用于比较字符串的重要的**最小编辑距离**算法。以下是我们关于这些想法的要点总结:

•**正则表达式**语言是模式匹配的强大工具。

•正则表达式的基本操作包括**连接**符号,**析取**符号([ ]，|，和。)，**计数器**(*，+，和{n,m})，**定位符**(^，$)和优先级操作符((,))。

•**单词符记化和规范化**通常是由级联的简单正则表达式替换或有限自动机。

•**Porter(波特)算法**是一个简单而有效的方法，用来做词干提取、剥离词缀。它的精度不高，但可能对某些任务有用。

•两个字符串之间的**最小编辑距离**是将一个字符串编辑为另一个字符串所需的最少操作数。最小编辑距离可以通过**动态规划**来计算，这也会导致两个字符串对齐。

2.7. 文献和历史说明

Kleene(1951, 1956)基于麦卡洛克-皮茨神经元，首先定义了正则表达式和有限自动机。Ken Thompson是最早将正则表达式编译器构建为文本搜索编辑器的人之一(Thompson, 1968)。他的编辑器 ed 包含了一个命令“g/正则表达式/p”，即**全局正则表达式打印(Global Regular Expression Print，GREP)**，它后来成为 Unix grep 实用程序。

文本规范化算法从一开始就被应用字段。最早被广泛使用的词干分析器之一是 Lovins(1968)。Packard(1973)也很早就将词干提取应用于数字人文学科，他建立了一个古希腊语词缀剥离形态学解析器。目前，可以使用多种用于符记化和规范化的代码，例如 Stanford 符记生成器(http://nlp.stanford.edu/software/tokenizer.shtml)或 Twitter 专用的符记生成器(O'Connor 等人 2010)，或用于表达情感(http：//sentiment.christopherpotts.net/tokenizing.html)。有关文本预处理的概述，请参见 Palmer(2012)。NLTK 是必不可少的工具，可提供有用的 Python 库(http://www.nltk.org)和教科书说明(Bird 等人 2009)，其中包括文本规范化和语料库接口在内的许多算法。

有关赫尔丹(Herdan)定律和希普斯(Heaps)定律的更多信息，请参见 Herdan(1960，p.28)，Heaps(1978)，Egghe（2007）和 Baayen(2001)；Yasseri 等人(2012)讨论了与其他语言复杂性度量之间的关系。有关编辑距离的更多信息，请参见优秀的 Gusfield(1997)。我们测量从“intention”到“execution”的编辑距离的示例改编自 Kruskal(1983)。有各种公开可用的软件包可以计算编辑距离，包括 Unix diff 和 NIST sclite 程序(NIST，2005)。

Bellman(1984)在他的自传中解释了他最初是如何提出动态规划这个术语的:

“…20 世纪 50 年代对数学研究来说不是好年头。国防部长…对研究这个词有一种病态的恐惧和憎恨…因此，我决定用“规划”这个词。我想让大家明白这是动态的，是多阶段的……我想，让我们……拿一个意义绝对精确的词来说，那就是动态……用“动态”这个词来贬义是不可能的。试着想出一些可能带有贬义的组合。这是不可能的。因此，我认为动态规划是个好名字。这是连国会议员都不会反对的事情。”

2.8.练习

2.1 Write regular expressions for the following languages.

1. the set of all alphabetic strings;
2. the set of all lower case alphabetic strings ending in a *b*;
3. the set of all strings from the alphabet *a,b* such that each *a* is immedi
4. ately preceded by and immediately followed by a *b*;

2.2 Write regular expressions for the following languages. By “word”, we mean an alphabetic string separated from other words by whitespace, any relevant punctuation, line breaks, and so forth.

1. the set of all strings with two consecutive repeated words
(e.g., “Hum bert Humbert” and “the the” but not “the bug” or “the big bug”);
2. all strings that start at the beginning of the line with an integer and that end at the end of the line with a word;
3. all strings that have both the word *grotto* and the word *raven* in them
(but not, e.g., words like *grottos* that merely *contain* the word *grotto*);
4. write a pattern that places the first word of an English sentence in a register. Deal with punctuation.

2.3 Implement an ELIZA-like program, using substitutions such as those described on page 11. You might want to choose a different domain than a Rogerian psychologist, although keep in mind that you would need a domain in which your program can legitimately engage in a lot of simple repetition.

2.4 Compute the edit distance (using insertion cost 1, deletion cost 1, substitution cost 1) of “leda” to “deal”. Show your work (using the edit distance grid).

2.5 Figure out whether *drive* is closer to *brief* or to *divers* and what the edit distance is to each. You may use any version of *distance* that you like.

2.6 Now implement a minimum edit distance algorithm and use your hand-computed results to check your code.

2.7 Augment the minimum edit distance algorithm to output an alignment; you will need to store pointers and add a stage to compute the backtrace.

3. N 元语法语言模型

“You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

Random sentence generated from a Jane Austen trigram model

预测是很困难的，尤其是预测未来，正如一句古老的俏皮话所说。但如何预测一些看起来更容易的事情，比如某人接下来要说的几句话是什么呢？例如，哪个单词可能会跟在后面：

Please turn your homework ...

希望你们中的大多数得出结论，一个很可能的单词是 in 或 over，但可能不是 refrigerator 或 the。在接下来的几节中，我们将通过引入模型，为每个可能的下一个单词指定一个概率，来形式化这种直觉。相同的模型还将用于为整个句子分配概率。例如，这种模型可以预测以下序列在文本中出现的概率更高：

all of a sudden I notice three guys standing on the sidewalk

（我突然注意到三个人站在人行道上）

这组相同单词有不同的排序:

on guys all I of notice sidewalk three a sudden standing the

您为什么要预测即将出现的单词，或为句子分配概率？在任何我们必须从嘈杂的、模糊的输入中识别单词的任务中，比如语音识别，概率都是必不可少的。为了使语音识别器意识到您说的是“I will be back soonish”而不是“I will be bassoon dish”，这有助于了解“back soonish”比“bassoon dish”更有可能出现。对于拼写纠错、语法纠错等写作工具，我们需要发现并纠正写作中的错误，比如“Their are two midterms”，其中“There”被误打为“Their”，或者“Everything has improve”，其中“improve”本应该是“improved”。短语“There are”比“Their are”更有可能出现，“has improved”比“has improve”更有可能出现，这使我们能够通过检测和纠正这些错误来帮助用户。

为单词序列分配概率在机器翻译中也很重要。假设我们正在翻译一个中文源句:

他　向　记者　介绍了　主要　内容

He　to　reporters　introduced　main　content

作为这个过程的一部分，我们可能已经建立了以下一套潜在的粗略英语翻译:

he introduced reporters to the main contents of the statement

he briefed to reporters the main contents of the statement

he briefed reporters on the main contents of the statement

单词序列的概率模型可能认为“briefed reporters on”的概率大于“briefed to reporters”(一个尴尬的“to”跟在“briefed”之后)或“introduced reporters to”(它使用的动词在这个语境中不是很流利)，这让我们正确地选择了上面黑体的句子。

概率对于**增强性和替代性交流系统(AAC)**也很重要(Trnka 等人 2007，Kane 等人 2017)。如果人们在肢体上无法讲话或签名，但通常可以使用视线凝视或其他特定动作从系统要说的菜单中选择单词，则人们通常会使用此类 AAC 设备。单词预测可以用来为菜单推荐可能的单词。

将概率分配给单词序列的模型称为**语言模型(language model，LM)**。在本章中，我们介绍最简单的模型，**n 元语法(n-gram)**，该模型将概率分配给句子和单词序列。n-gram 是由 n 个单词组成的序列:2-gram(bigram)是由两个单词组成的序列，如“please turn”，“turn your”，或“your homework”；3-gram(trigram)是由三个单词组成的序列，如“please turn your”，或“turn your homework”。我们将看到如何使用 n-gram 模型，在给定前一个单词的情况下，来估计 n-gram 后一个单词的概率，并为整个序列分配概率。出于术语上的歧义，我们通常会放弃“模型”一词，而使用术语 n-gram(以及 bigram 等)来表示单词序列本身或为其指定概率的预测模型。我们将在后面介绍更复杂的语言模型，如第 9 章中的 RNN LMs。

3.1. N 元语法

让我们从计算 P(w | h)的任务开始，即：在给定历史 h 的条件下，计算单词 w 的概率。假设历史 h 是“its water is so transparent that”，我们想知道下一个单词是“the”的概率：

$$P(\text{the} \mid \text{its water is so transparent that}) \tag{3.1}$$

估计这个概率的一种方法是通过相对频率计数:取一个非常大的语料库，计算我们看到“its water is so transparent that”的次数，并计算它后面跟着“the”的次数。这就可以回答“根据我们看到历史 h 的次数，有多少次它后面跟着 w”，如下所示：

$$\begin{aligned} &P(\text{the} \mid \text{its water is so transparent that}) \\ &= \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})} \end{aligned} \tag{3.2}$$

有了足够大的语料库，如 web，我们可以计算这些计数，并根据公式 3.2 估计概率。现在，您应该暂停一下，到 web 上，自己计算一下这个估算值。

虽然这种直接从计数来估计概率的方法在许多情况下都很有效，但事实证明，在大多数情况下，即使是 web 也不足以给我们提供良好的估计。这是因为语言具有创造力； 我们会一直创建新的句子，而我们将永远无法统计全部句子。即使是例句的简单扩展，在 web 上也可能出现 0 的计数(例如 Walden Pond’s water is so transparent that the 在过去的计数为零)。

类似地，如果我们想知道“its water is so transparent”这类单词序列的联合概率，我们可以问“在所有可能的五个单词序列中，有多少个单词是 its water is so transparent?”我们必须得到“its water is so transparent”的计数，然后除以所有可能的五个单词序列的计数总和。这似乎是一个很大的估计!

出于这个原因，我们需要引入更聪明的方法来估计在给定历史 h 条件下的单词 w 的概率，或者整个单词序列 w 的概率。让我们从一些形式化的符号开始。为了表示特定随机变量 X_i 取值“the”或 $P(X_i =\text{“the”})$的概率，我们将使用化简后的 P(the)。我们将 N 个单词的序列表示为 $w_1 \dots w_n$ 或 $w_{1:n}$(因此表达式 $w_{1:n-1}$ 表示字符串 w_1，w_2，…，w_{n-1})。对于具有特定值 P(X = w_1，Y = w_2，Z = w_3，…，W = w_n)的序列中每个单词的联合概率，我们将使用 P(w_1，w_2，…， w_n)。

现在我们如何计算像 P(w_1,w_2，…，w_n)这样的整个序列的概率?我们可以做的一件事是使用概率的**链式规则(chain rule)**来分解这个概率:

$$\begin{aligned} P(X_1 \dots X_n) &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_{1:2}) \dots P(X_n \mid X_{1:n-1}) \\ &= \prod_{k=1}^{n} P(X_k \mid X_{1:k-1}) \end{aligned} \tag{3.3}$$

运用链式规则，我们得到：

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2}) \dots P(w_n \mid w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1}) \end{aligned} \tag{3.4}$$

链式规则展示了计算一个序列的联合概率和计算在给定前一个单词的情况下的条件概率之间的联系。公式 3.4 表明，我们可以通过将多个条件概率相乘来估计整个单词序列的联合概率。但是，使用链式规则似乎并没有真正帮助我们！给定一长串前面的单词 $P(w_n \mid w_{1:n-1})$，我们没有任何方法可以计算出一个单词的确切概率。正如我们上面所说的，我们不能仅仅通过计算每个单词在每个长字符串后面出现的次数来估计，因为语言是有创造性的，任何特定的上下文可能以前从未出现过!

n 元语法模型的直觉是，与其在给定全部单词历史记录的情况下计算单词的概率，不如仅通过最后几个单词来估计历史记录。

例如，二元语法(bigram)模型仅通过使用一个先前单词 $P(w_n | w_{n-1})$的条件概率来近似给定所有先前单词 $P(w_n | w_{1:n-1})$的单词的概率。换句话说，不是计算概率：

$$P(\text{the}|\text{Walden Pond's water is so transparent that}) \tag{3.5}$$

针对公式 3.5，我们用下面的概率来作近似计算：

$$P(\text{the}|\text{that}) \tag{3.6}$$

因此，当我们使用 bigram 模型来预测下一个单词的条件概率时，我们可以做出如下近似:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1}) \tag{3.7}$$

一个词的概率只取决于前一个词的假设被称为马尔科夫(Markov)假设。马尔可夫模型是概率模型的一类，它们假定我们可以预测某些未来单元的概率而不会过多地考虑过去。我们可以将二元语法(向过去看一个单词)推广到三元语法(向过去看两个单词)，然后推广到 n 元语法(n-gram)(向过去看 n-1 个单词)。

因此，这个 n 元近似的序列中下一个单词的条件概率的一般方程是：

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1}) \tag{3.8}$$

在给定单个单词概率的二元语法假设的情况下，我们可以通过将公式 3.7 代入公式 3.4 计算出全部单词序列的概率:

$$P(w_{1:n}) = \prod_{k=1}^{n} P(w_k | w_{1:k-1}) \tag{3.9}$$

我们如何估计这些二元或 n 元语法概率？ 估计概率的一种直观方法称为最大似然估计(MLE)。我们通过从语料库中获取计数，并对计数进行归一化(normalize)，使它们位于 0 到 1 之间，从而获得 n-gram 模型参数的 MLE 估计。

例如，在给定前一个单词 x 的条件下，要计算单词 y 的一个特定的二元语法概率，我们将计算二元语法 C(xy)的计数，并通过共享相同的第一个单词 x 的所有二元语法的总和，再进行归一化：

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{\sum_w C(w_{n-1} w)} \tag{3.10}$$

我们可以简化此等式，因为以给定单词 w_{n-1} 开头的所有二元语法计数的总和必须等于该单词 w_{n-1} 的一元语法计数(读者应该花点时间对此加以证明)：

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})} \tag{3.11}$$

让我们用一个三句话的小语料库来做一个例子。首先，我们需要在每个句子的开头添加一个特殊符号<s>，以便为我们提供第一个单词的二元语法上下文。我们还需要一个特殊的结束符号</ s >。

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

以下是对这个语料库中一些二元语法概率的计算：

P(I | <s>) = 2/3 =.67　　P(Sam | <s>) = 1/3 =.33　　P(am | I) = 2/3=.67

P(</s> | Sam) = 1/2 =.5　　P(Sam | am) = 1/2=.5　　P(do | I) = 1/3=.33

对于 MLE n-gram 参数估计的一般情况:

$$P(w_n | w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\ w_n)}{C(w_{n-N+1:n-1})} \tag{3.12}$$

公式 3.12(类似于公式 3.11)通过将特定序列的观测频率除以前缀的观测频率来估计 n 元语法的概率。这个比率称为**相对频率(relative frequency)**。我们在上面说过，这种使用相对频率来估计概率的方法是最大似然估计(MLE)的一个例子。在 MLE 中，给定模型 M(即 P(T | M))，结果参数集将训练集 T 的可能性最大化。例如，假设“Chinese”这个词在一个有 100 万个单词的语料库(比如布朗语料库)中出现了 400 次。那么，从其他文本中(例如，100 万单词中)随机选出的一个单词，出现 Chinese 的概率是多少？其概率的最大似然估计值为 400/1000000 或 0.0004。现在，0.0004 并不是 Chinese 在所有情况下发生概率的最好估计；在其他语料库或语境中，Chinese 是一个不太可能出现的词。但在一个 100 万单词的语料库中，Chinese 出现 400 次的可能性很大。在第 3.4 节中，我们提出了稍微修改 MLE 估计的一些方法，以获得更好的概率估计。

让我们继续看一些例子，这些例子来自一个比上面 14 个单词的例子略大的语料库。我们将使用来自现在已经不存在的伯克利餐厅项目的数据，这是一个上世纪的对话系统，它回答了关于加州伯克利餐厅数据库的问题(Jurafsky 等人，1994)。这里有一些文本规范化用户查询的样本(9332 句的样本是在网站上):

can you tell me about any good cantonese restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day

图 3.1 显示了来自伯克利餐厅项目的一个 bigram 语法片段的 bigram 计数。请注意，大多数值都是零。事实上，我们选择的样本词是为了彼此连贯;从 7 个单词的随机集合中选择的矩阵将更加稀疏。

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

图 3-1：二元语法计数

图注：伯克利餐厅项目语料库 9332 个句子中 8 个单词(V=1446)的 bigram 计数，

图 3.2 显示了归一化后的二元概率(将图 3.1 中的每个单元除以对应行的一元语法概率，其值取自下列集合的一元语法概率)：

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

图 3-2：二元语法概率

图注：伯克利餐厅项目语料库 9332 个句子中 8 个单词的二元语法概率。

以下是其他一些有用的概率:

P(i|<s>) = 0.25 *P*(english|want) = 0.0011

P(food|english) = 0.5 *P*(</s>|food) = 0.68

现在，我们可以通过简单地将相应的二元语法概率相乘在一起，就可计算像“I want English food”或“I want Chinese food”这种句子的概率，如下所示：

$$
\begin{aligned}
&P(\text{<s> i want english food </s>}) \\
&\quad = P(\text{i}\,|\,\text{<s>})\,P(\text{want}\,|\,\text{i})\,P(\text{english}\,|\,\text{want}) \\
&\qquad\qquad P(\text{food}\,|\,\text{english})P(\text{</s>}\,|\,\text{food}) \\
&\quad = .25\times .33\times .0011\times .5\times .68 \\
&\quad = .000031
\end{aligned}
$$

我们把它留作练习 3.2 来计算"i want chinese food"的概率。

这些二元语法数据统计中捕获了哪些语言现象？上面的一些二元语法概率编码了一些我们认为本质上严格属于句法的事实，例如，“eat”之后出现的通常是名词或形容词，或者“to”之后出现的通常是动词。另一些可能是关于个人助理任务的事实，比如句子以单词"I"开头的可能性很高。有些甚至可能是文化因素而非语言因素，比如人们更倾向于寻找中国食物而不是英国食物。

一些实际问题：尽管出于教学目的，我们只描述了二元语法模型，但在实践中，更常见的是使用**三元语法(trigram)**模型，它以前两个单词为条件，而不是前一个单词，或者当有足够的训练数据时，使用 4-gram 甚至 5-gram 模型。请注意，对于这些较大的 n-grams，我们需要假定在句子末尾的左右两侧有额外的上下文。例如，为了计算句子开头的三元语法概率，我们对第一个三元语法使用两个伪词(即 P(I|<s><s>)。

我们通常使用对数概率的形式来表示和计算语言模型的概率。因为概率(根据定义)小于或等于 1，所以我们相乘的概率越多，乘积就越小。将足够多的 n-gram 相乘会导致数字下溢(underflow)。通过使用对数概率而不是原始概率，我们得到的数字并不那么小。

因为在对数空间中相加等同于在线性空间中相乘，因此我们通过将它们相加来组合**对数概率(log probabilies)**。在对数空间中进行所有计算和存储的结果是，我们只需要在最后需要报告它们的时候，才将它们转换回概率;然后我们可以取 logprob 的 exp：

$$
p_1\times p_2\times p_3\times p_4 = \exp(\log p_1+\log p_2+\log p_3+\log p_4) \tag{3.13}
$$

3.2. 评估语言模型

评估语言模型性能的最佳方法是将其嵌入到应用程序中，并衡量应用程序的改进程度。这种端到端的评估称为**外部评估(extrinsic evaluation)**。外部评估是了解某个组件的特定改进是否真的会帮助完成手头任务的唯一方法。因此，对于语音识别，我们可以通过运行两次语音识别器来比较两种语言模型的性能，每一种语言模型运行一次，看看哪一种语言模型的转录更准确。

不幸的是，端到端的运行大型 NLP 系统通常非常昂贵。相反，如果有一个度量可以用来快速评估语言模型中潜在的改进，那就太好了。一个**内在评估(intrinsic evaluation)**度量是一个独立于任何应用程序的模型质量的测量。

对于语言模型的内在评估，我们需要一个测试集。与我们领域中的许多统计模型一样，n-gram 模型的概率来自于它所受训练的语料库，即**训练集(training set)**或训练语料库。然后，我们可以通过 n-gram 模型在一些不可见的数据(称为**测试集(test set)**或测试语料库)上的性能来衡量其质量。有时我们还会将不在训练集中的测试集和其他数据集称为**保留(held out)**语料库，因为我们将它们保留在训练数据之外。

因此，如果我们有一个文本语料库，并且想要比较两个不同的 n-gram 模型，我们将数据分成训练集和测试集，在训练集上训练两个模型的参数，然后比较两个训练过的模型对测试集的拟合程度有多好。

但是“适合测试集”是什么意思呢?答案很简单:给测试集赋值较高的概率(意味着它能更准确地预测测试集)的模型是更好的模型。给定两个概率模型，较好的模型与测试数据的拟合更紧密，或者能够更好地预测测试数据的细节，从而将较高的概率分配给测试数据。

因为我们的评估指标是基于测试集概率的，所以不要让测试句子进入训练集是很重要的。假设我们正在尝试计算特定“测试”句子的概率。如果我们的测试句子是训练语料库的一部分，那么，当它出现在测试集中时，我们会错误地将一个人为的高概率分配给它。我们把这种情况称为测试集上的训练。在测试集上进行训练会引入一种偏差，这种偏差会使所有概率看上去都过高，并在**困惑度**上导致巨大的不准确，我们下面介绍这种基于概率的度量方法。

有时，我们过于频繁地使用一个特定的测试集，以致于我们隐式地调整了它的特征。然后，我们需要一个新的测试集，一个真正看不见的测试集。在这种情况下，我们将初始测试集称为**开发测试集(devset)**。我们如何将数据划分为训练、开发和测试集?我们希望我们的测试集尽可能大，因为一个小的测试集可能意外地不具有代表性，但我们也希望尽可能多的训练数据。至少，我们希望选择最小的测试集，使我们有足够的统计能力来测量两个潜在模型之间在统计上的显著差异。在实践中，我们通常只是将我们的数据分为 80%的训练、10%的开发和 10%的测试。给定我们要划分为训练和测试的大型语料库，可以从语料库中的一些连续文本序列中获取测试数据，也可以从语料库的随机选择部分中删除较小的“条带”文本并将其合并进入测试集。

3.2.1. 困惑度

在实践中，我们不使用原始概率作为我们评估语言模型的度量标准，而是使用一种叫做**困惑度(perplexity)**的变量。测试集上的语言模型的困惑度(有时简称 PP)是测试集的逆反概率，由单词的数量归一化。对于一个测试集 $W = w_1w_2...w_N$，有:

$$\begin{aligned} PP(W) &= P(w_1w_2...w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1w_2...w_N)}} \end{aligned} \tag{3.14}$$

我们可以用链式规则来展开 W 的概率:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i \mid w_1w_2...w_{i-1})}} \tag{3.15}$$

因此，如果我们用一个二元语言模型来计算 W 的困惑度，我们得到:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i \mid w_{i-1})}} \tag{3.16}$$

注意，由于公式 3.15 中的倒数，如果单词序列的条件概率越高，则困惑度越低。因此，根据语言模型，最小化困惑度等价于最大化测试集概率。我们通常在公式 3.15 或 3.16 用的单词序列，是某个测试集中的整个单词序列。由于此序列将跨越许多句子边界，因此我们需要在概率计算中包括开始和结束句子标记<s>和</ s>。我们还需要在单词符记的总数 N 中包括句子结束标记</s>(但不包括句子开始标记<s>)。

还有另一种思考困惑度的方式:作为一种语言的加权平均的分支因子。语言的分支因子是任何单词后面可能出现的下一个单词的数量。考虑一下识别英文数字(0、1、2、……9)的任务。假设(在某些训练集中和某些测试集中)这 10 个数字中的每一个数字都以相等的概率 P = 1/10 出现，那么这个小型语言的困惑实际上是 10。为了验证这一点，想象一个长度为 N 的测试字符串，并假设在训练集中所有的数字发生的概率相等。由公式 3.15 可知，困惑度将是：

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1^N}{10}\right)^{-\frac{1}{N}} \\
&= \frac{1^{-1}}{10} \\
&= 10
\end{aligned}
\tag{3.17}
$$

但是假设数字 0 的出现频率比其他数字高得多。假设 0 在训练集中出现 91 次，其他数字每个出现 1 次。现在我们看到下面的测试集:0 0 0 0 0 3 0 0 0 0 。我们应该预料到这个测试集的困惑度会更低，因为大多数时候下一个数字是零，这是很容易预测的，也就是说有很高的概率。因此，虽然分支因子仍为 10，但困惑度或加权分支因子较小。我们把这个精确的计算留给练习 12。

我们在 3.7 节中将看到，困惑度也与熵的信息理论概念密切相关。

最后，让我们看看如何使用困惑度来比较不同的 n-gram 模型。我们用《华尔街日报》的 3800 万个单词(包括句子开头的符记)训练了一元、二元和三元语法模型，使用了一个含有 19,979 个单词的词汇表。然后，我们用公式 3.16 在一个 150 万单词的测试集上计算每个模型的困惑度。下表显示了根据每种语法的 150 万个单词 WSJ 测试集的困惑度。

	Unigram	Bigram	Trigram
Perplexity	962	170	109

正如我们在上面看到的，n-gram 提供给我们有关单词序列的信息越多，困惑度就越低(因为如公式 3.15 所示，困惑度与根据模型的测试序列的可能性成反比)。

需要注意的是，在计算困惑度时，n-gram 模型 P 必须在不了解测试集的情况下构造，也不需要事先了解测试集的词汇量。测试集的任何知识都可能导致困惑度被人为地降低。两种语言模型的困惑只有在使用相同的词汇时才具有可比性。

困惑度的(内在的)改善，并不保证语言处理任务(如语音识别或机器翻译)性能的(外在的)改善。尽管如此，由于困惑往往与这些改进相关，它通常被用作对算法的快速检查。但是，在对模型进行评价之前，必须先对实际任务进行端到端的评价，以确定模型在困惑性方面的改进。

3.3.泛化与零

n-gram 模型，像许多统计模型一样，依赖于训练语料库。一个暗示是，概率通常会针对有关给定训练语料库的特定事实进行编码。另一个暗示是，随着我们增加 N 的值，n-grams 在对训练语料库进行建模方面做得越来越好。

我们可以通过借鉴 Shannon(1951)、Miller 和 Selfridge(1950)从不同的 n-gram 模型中生成随机句子的技术来将这两个事实可视化。最简单的可视化方法就是想象一元语法是如何工作的。想象英语中所有的单词都覆盖在 0 到 1 之间的概率空间中，每个词所覆盖的区间与它的频率成比例。我们在 0 到 1 之间选择一个随机值，然后打印其区间包含所选值的单词。我们继续选择随机数并生成单词，直到我们随机生成一个句子结尾的符记</s>为止。我们可以使用同样的技术来生成二元语法，首先生成一个以<s>开头的随机二元语法(根据它的二元语法概率)。假设该二元语法的第二个单词是 w。接下来，我们选择以 w 开头的随机二元语法(再次根据其二元语法概率绘制)，依此类推。

为了直观地了解高阶 n-gram 的强大功能，图 3.3 显示了根据莎士比亚作品中训练的 1-gram，2-gram，3-gram 和 4-gram 模型生成的随机句子。

我们训练模型的上下文越长，句子的连贯性就越强。在一元语法的句子中，单词与单词之间、或单词与句末标点之间没有连贯的关系。二元语法的句子具有局部的单词对单词的连贯性(尤其是如果我们认为标点符号算作一个词)。三元语法和四元语法的句子开始越来越像莎士比亚的作品了。确实，对这种四元语法

句子的仔细研究表明，它们看起来有点像莎士比亚。“It cannot be but so”这句话是直接来自国王约翰。这是因为，不要去指责莎士比亚，他的作品在语料库上并不是很大(N=884,647，V=29,066)，而我们的 n-gram 概率矩阵非常稀疏。可能的二元语法有 $V^2 = 844{,}000{,}000$ 个，可能的四元语法是 $V^4 = 7\times10^{17}$ 个。因此，一旦生成器选择了第一个四元语法(It cannot be but)，就只有五种可能的延续(that, I, he, thou, and so);事实上，对于许多四元语法，只有一个延续。

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

图 3-3：四个 n-gram 随机生成的八个句子

图注：根据莎氏作品进行计算，所有字符都被映射为小写，标点符号被视为单词。输出被手工更正为大写，以提高可读性。

为了了解语法对其训练集的依赖性，让我们看看在完全不同的语料库上训练的 n-gram 语法:《华尔街日报》(WSJ)报纸。莎士比亚和《华尔街日报》都是英语，所以我们可能会发现这两种体裁的 n-gram 符号有些重叠。图 3.4 显示了用《华尔街日报》的 4000 万个单词训练后的 unigram、bigram 和 trigram 语法生成的句子。

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N.B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

图 3-4：三个 n-gram 模型随机生成的三个句子

图注：三个 n-gram 模型从《华尔街日报》的 4000 万单词中随机生成的 3 个句子，将所有字符都小写，并将标点作为单词处理。输出随后被手工更正为大写，以提高可读性。

将这些例子与图 3.3 中的伪莎士比亚作比较。虽然它们都为“English-like sentences”建模，但在生成的句子中明显没有重叠，甚至在小短语中也几乎没有重叠。统计模型可能几乎毫无用处，因为训练集和测试集的预测器(predictors)与莎士比亚和《华尔街日报》不同。

当我们建立 n-gram 模型时，我们应该如何处理这个问题?第一步是确保使用与我们试图完成的任务相似体裁的训练语料库。为了建立法律文件翻译的语言模型，我们需要一个法律文件的训练语料库。为了建立问答系统的语言模型，需要建立问答训练语料库。

同样重要的是，获取适当方言或变体的训练数据，特别是在处理社交媒体帖子或口语文本时。例如，一些推特将使用非裔美国人语言(AAL)的特征——非洲裔美国人社区使用的许多语言变体的名称(King 2020)。这些特征包括像 finna 这样的词-它是一种助动词，用于标记立即将来时态-或者把“then”拼成“den”，就像这样的推特(Blodgett 和 O'Connor, 2017):

(3.18) Bored af den my phone finna die!!!

然而，来自尼日利亚英语等变种的推特，与来自美国英语的词汇和 n-gram 模式有明显不同(Jurgens 等人 2017)：

(3.19) @username R u a wizard or wat gan sef: in d mornin - u tweet, afternoon - u tweet, nyt gan u dey tweet. beta get ur IT placement wiv twitter

匹配的体裁和方言仍然是不够的。我们的模型可能仍然存在稀疏性问题。对于任何发生了足够次数的

n-gram，我们也许可以很好地估计它的概率。但是由于任何语料库都是有限的，因此一定会缺少一些完全可以接受的英语单词序列。也就是说，我们将有很多情况下假定的“零概率 n-grams”实际上应该具有一些非零概率。考虑一下《华尔街日报》 Treebank3 语料库中，在“denied the”之后的那些单词及其计数：

denied the allegations: 5
denied the speculation: 2
denied the rumors: 1
denied the report: 1

但是，假设我们的测试集具有以下短语：

denied the offer
denied the loan

我们的模型将错误地估计 P(offer|denied the)是 0!

这些零（0）—在训练集中不会出现，但在测试集中会出现—是个问题，有两个原因。

首先，它们的存在意味着我们低估了各种可能出现的单词的概率，这将损害我们希望在这些数据上运行的任何应用程序的性能。

其次，如果测试集之中任何一个单词的概率为 0，则测试集的整体概率为 0。根据定义，困惑度是基于测试集的逆概率的。因此，如果某些单词的概率为零，那么，因为我们不能除以 0，所以根本无法计算困惑度！

3.3.1. 未知单词

前节讨论了二元语法概率为零的那些单词的问题。但那些我们从未见过的单词概率问题怎么处理呢?

有时在我们的语言任务中，这种情况不会发生，因为我们知道所有可能发生的单词。在这样一个**封闭词汇(closed vocabulary)**的系统中，测试集只能包含本词汇库中的单词，不会有未知的单词。在某些领域(例如语音识别或机器翻译)中，这是一个合理的假设，在这些领域中，我们预先安装了发音词典或短语表，因此语言模型只能使用该词典或短语表中的单词。

在其他情况下，我们必须处理我们之前没有见过的单词，我们称之为未知单词，或**表外词(OOV)**单词。OOV 单词在测试集中出现的百分比称为 OOV 率。一个**开放词汇(open vocabulary)**的系统是这样一个系统，在这个系统中，首先，针对测试集之中潜在的未知词，我们添加一个名为<UNK>的伪词；然后，我们对那些未知词进行建模。

有两种常用的方法来训练未知词模型<UNK>的概率。

第一个方法是提前选择一个固定的词汇表，把这个问题变成一个封闭的词汇表:

1. 选择一个预先确定的词汇表(单词列表)。
2. 在文本规范化步骤中，将训练集中不属于该集合的任何单词(OOV)转换为未知单词符记<UNK>。
3. 根据<UNK>的计数估计其概率，就像训练集中的任何其他普通单词一样。

第二种方法，在我们事先没有预先的词汇表的情况下，是隐式地创建这样的词汇表，方法如下：基于单词的频率，用<UNK>替换训练数据中的单词。例如，我们可以用<UNK>替换所有在训练集中出现次数小于 n 次的单词，其中 n 是一个较小的数字，或者等价地提前选择单词大小 V(比如 50,000)，按频率选择前 V 个单词，其余的用 UNK 替换。在任何一种情况下，我们都将继续像以前一样训练语言模型，将<UNK>视为普通单词。

准确地选择<UNK>模型确实会对诸如困惑度等指标产生影响。通过选择小的词汇量和赋予高概率的未知词的方法，语言模型能够取得低困惑度。出于这个原因，困惑度应该只在具有相同词汇表的语言模型之间进行比较(Buck 等人，2014)。

3.4. 平滑

我们如何处理词汇表中的单词(它们不是未知的单词)，但它们出现在一个看不见的上下文的测试集之中(例如，它们出现在一个在训练中从未出现过的单词之后)?为了防止语言模型为这些不可见事件分配零概

率，我们将不得不从一些更频繁发生的事件中消除一些**概率质量(probability mass)**，并将其分配给我们从未见过的事件。这种修改被称为**平滑(smoothing)**或**折扣(discounting)**。在这一节和接下来的几节中，我们将介绍各种平滑的方法：拉普拉斯(加一)平滑、加 k 平滑、傻瓜回退和 Kneser-Ney 平滑。

3.4.1. 拉普拉斯平滑

做平滑处理的最简单方法是在我们将它们标准化为概率之前，将所有的 bigram 计数加 1。以前为 0 的所有计数现在都将为 1，1 的计数将为 2，以此类推。这种算法称为**拉普拉斯平滑(Laplace smoothing)**。拉普拉斯平滑在现代 n-gram 模型中表现得不够好，但它很有用地引入了许多我们在其他平滑算法中看到的概念，给出了一个有用的基线，也是用于其他任务(如文本分类，第 4 章)的实际平滑算法。

让我们从拉普拉斯平滑在一元语法概率中的应用开始。回想一下，单词 w_i 的一元语法概率的未平滑最大似然估计是其计数 c_i 的归一化(N 是单词符记总数)：

$$P(w_i) = \frac{c_i}{N}$$

拉普拉斯平滑仅对每个计数加一(因此其备用名称为**加一(add-one)**平滑)。由于词汇中有 V 个单词，并且每个单词都增加了，因此我们还需要调整分母以考虑额外的 V 个观察值。(如果不增加分母，那么 P 值会怎样？)

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V} \tag{3.20}$$

通过定义一个调整后的计数 c^*，可以方便地描述平滑算法如何影响分子，而不必同时更改分子和分母。这个调整后的计数更容易直接与 MLE 计数进行比较，并且可以通过 N 的归一化将其转换为类似于 MLE 计数的概率。为了定义此计数，因除了加 1 之外，我们仅更改了分子，故还需要乘以归一化因子 N/(N+V)：

$$c_i^* = (c_i + 1)\frac{N}{N + V} \tag{3.21}$$

现在我们可以通过将 N 做归一化把 c_i^*变成概率 P_i^*。

看待平滑的一种相关方法是，为了获得将被分配给零计数的概率质量，**打折(discounting)**(降低)一些非零计数。因此，我们不是引用折扣计数 c^*，而是用相对折扣 d_c 来描述平滑算法，即折扣计数与原始计数的比率：

$$d_c = \frac{c^*}{c}$$

现在我们对 unigram 情况有了直观的了解，让我们平滑伯克利餐厅项目的 bigrams。图 3.5 显示了图 3.1 中 bigrams 的加一平滑计数。

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

图 3-5：加一平滑 bigram 计数的 8 个单词

图注：(从 V = 1446 中)伯克利餐厅项目语料库有 9332 个句子。

图 3.6 显示了图 3.2 中 bigrams 的加一平滑概率。回想一下，通常的 bigrams 概率是通过用 unigram 计数归一化每一行计数来计算的：

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{3.22}$$

对于加一平滑的 biggram 计数，我们需要通过词汇表 V 中单词类型总数来增加 unigram 计数:

$$P^*_{\text{Laplace}}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w C(w_{n-1}w)+1} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V} \tag{3.23}$$

因此，上一节给出的每一个 unigram 计数都需要通过 V = 1446 来增加，其结果为图 3.6 中平滑的 bigram 概率。

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

图 3-6：加一平滑 bigram 概率的 8 个单词

图注：在 9332 个句子的 BeRP 语料库中，8 个单词(从 V = 1446 中)的加一平滑 bigram 概率。

重建计数矩阵通常很方便，这样我们就可以看到平滑算法对原始计数的改变程度。这些调整后的计数可以通过公式 3.24 计算出来。图 3.7 显示了重建计数。

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1]\times C(w_{n-1})}{C(w_{n-1})+V} \tag{3.24}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

图 3-7：对 8 个单词的加一重建计数

图注：在 9332 个句子的 BeRP 语料库中，对 8 个单词(从 V = 1446 中)的加一重建计数。

请注意，加一平滑对计数进行了很大的更改。C(want to)从 609 变为 238！我们也可以在概率空间中看到这一点：P(to | want)从不平滑情况下的 0.66 减少到平滑情况下的 0.26。查看折扣 d(新旧计数之间的比率)可以看出每个前缀字的计数减少得多么惊人。bigram 中 want to 的折扣是 0.39，而 Chinese food 的折扣是 0.10，是 10 的倍数！

发生计数和概率的急剧变化是因为太多的概率质量移到了全零。

3.4.2. 加 k 平滑

加一平滑的另一种选择，是从可见事件到不可见事件的概率质量稍微减少一点。我们不是给每个计数加 1，而是加一个分数计数 k(.5?. 05?. 01?)。因此，此算法被称为**加 k(add-k)**平滑。

$$P^*_{\text{Add-k}}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV} \tag{3.25}$$

加 k 平滑要求我们有选择 k 的方法;例如，这可以通过优化开发集(devset)来实现。尽管加 k 在一些任

务中很有用(包括文本分类)，但它仍然不能很好地用于语言建模，生成的计数具有很差的方差和经常不适当的折扣(Gale 和 Church, 1994)。

3.4.3. 回退和插值

我们一直在讨论的打折可以帮助解决零频率 n-gram 的问题。但我们还可以利用另外一个知识来源。如果我们试图计算 $P(w_n | w_{n-2}\, w_{n-1})$但是我们又没有一个特定的三元语法 $w_{n-2}\, w_{n-1}\, w_n$ 例子,那么,我们可以通过使用二元语法概率 $P(w_n | w_{n-1})$来估计其概率。同样的,如果我们没有计数来计算 $P(w_n | w_{n-1})$,那么我们可以查看一元语法的 $P(w_n)$。

换句话说，有时使用较少的上下文是一件好事，可以帮助生成更多模型还不太了解的上下文。有两种方法可以使用这个 n-gram“层次结构”。在**回退(bakkoff)**中，如果证据充分，则用三元语法，否则用二元语法，否则就用一元语法。换句话说，我们只有在没有证据证明高阶 n-gram 存在的情况下才会“回退”到低阶 n-gram。相比之下，在**插值(interpolation)**中，通过权衡和组合 trigram、bigram、和 unigram 计数的方式，我们总是混合来自所有 n 元语法估计器的概率估计。

在简单线性插值中，我们通过对所有模型进行线性插值来组合不同阶的 n-gram。因此，通过混合 unigram、bigram 和 trigram 概率及其对应的加权 λ 系数，我们估计 trigram 概率 $P(w_n|w_{n-2}\, w_{n-1})$:

$$\begin{aligned}\hat{P}(w_n | w_{n-2} w_{n-1}) = {} & \lambda_1 P(w_n | w_{n-2} w_{n-1}) \\ & + \lambda_2 P(w_n | w_{n-1}) \\ & + \lambda_3 P(w_n)\end{aligned} \tag{3.26}$$

其中，λ_s 总和为 1：

$$\sum_i \lambda_i = 1 \tag{3.27}$$

在一个稍微复杂的线性插值版本中，每个λ权重是通过上下文条件计算的。这样，如果我们对一个特定的二元语法具有特别准确的计数，那么我们就认为基于该二元语法的三元语法的计数将更加可信，因此我们可以使这些三元语法的 λ_s 更高，从而在插值中使该三元语法的权重更大。公式 3.28 显示了具有上下文条件权重的插值公式：

$$\begin{aligned}\hat{P}(w_n | w_{n-2} w_{n-1}) = {} & \lambda_1(w_{n-2:n-1})\ P(w_n | w_{n-2} w_{n-1}) \\ & + \lambda_2(w_{n-2:n-1})\ P(w_n | w_{n-1}) \\ & + \lambda_3(w_{n-2:n-1})\ P(w_n)\end{aligned} \tag{3.28}$$

这些λ值如何设置？简单插值和条件插值λ_s 都是从**保留(held-out)**语料库中学习的。保留语料库是一种额外的训练语料库，我们可以通过选择最大化保留语料库似然度的λ值来设置这些λ值之类的超参数。也就是说，我们固定 n 元语法概率，然后搜索λ值(当插入公式 3.26 时)给保留集最大的概率。有很多方法可以找到λ_s 的最优集合。一种方法是使用 EM 算法，这是一种收敛于局部最优λ_s 的迭代学习算法(Jelinek 和 Mercer, 1980)。

在回退的 n-gram 模型中，如果我们需要的 n-gram 计数为零，那么我们通过回退到(N-1)-gram 来近似它。我们将继续退缩，直到达到具有重要历史意义的计数。

为了让回退模型给出一个正确的概率分布，我们必须对高阶 n-gram 进行打折，为低阶 n-gram 保留一些概率质量。就像加法平滑一样，如果不对高阶 n-gram 进行打折，而我们只使用了未打折的 MLE 概率，则只要我们将具有零概率的 n-gram 替换为低阶 n-gram，我们就可以添加概率质量，通过语言模型对所有可能的字符串的概率进行分配，总概率将大于 1！除了这个显式的折扣因子，我们还需要一个函数 α 来将这个概率质量分配到低阶 n-gram。

这种打折的回退也被称为**卡茨回退(Katz backoff)**。在 Katz 回退中，如果我们之前看过这个 n-gram(即，如果我们有非零计数)，则我们依赖于打折概率 P^*；否则，我们递归地返回较短历史(N-1)-gram 的卡茨概率。因此，计算回退 n-gram 的 P_{BO} 概率如下：

$$P_{BO}(w_n \mid w_{n-N+1:n-1}) = \begin{cases} P^*(w_n \mid w_{n-N+1:n-1}), & \text{if } C(w_{n-N+1:n}) > 0 \\ \alpha(w_{n-N+1:n-1}) P_{BO}(w_n \mid w_{n-N+2:n-1}), & \text{otherwise.} \end{cases} \tag{3.29}$$

卡茨回退通常与一种称为“好图灵”(Good-Turing)的平滑方法结合在一起。组合的 Good-Turing 回退算法涉及到相当详细的计算，这种计算是为了估计 Good-Turing 平滑、P^*和 α 值。

3.5. Kneser-Ney 平滑

最常用和性能最好的 n-gram 平滑方法之一是插值的 **Kneser-Ney 算法**(Kneser 和 Ney 1995, Chen 和 Goodman 1998)。

Kneser-Ney 起源于一种称为绝对打折的方法。回想一下，频繁 n-gram 的计数打折对于节省一些概率量是必要的，以便使平滑算法分配给看不见的 n-gram。

为了看到这一点，我们可以使用 Church 和 Gale(1991)的一个聪明的想法。考虑一个计数为 4 的 n-gram。我们需要对这个计数作一些打折。但是我们应该打折多少呢?Church 和 Gale 的聪明主意是查看一个保留的语料库，然后看看在训练集中计数为 4 的所有 bigrams 的计数是多少。他们从美联社的 2200 万个单词中计算出了一个 bigram 语法，然后在另外 2200 万个单词中检查了每个 bigram 的计数。平均而言，在前 2200 万个单词中出现 4 次的 bigram 在后 2200 万个单词中出现 3.23 次。图 3.8(Church 和 Gale,1991)显示了 c 介于 0 到 9 之间的 bigrams 的计数。

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

图 3-8：训练集和保留集的 bigrams 计数

图注：训练集包括美联社新闻热线的 2200 万单词，其 bigram 计数分别为 0,1,2，…9；对应地，保留集语料库也包含 2200 万单词，其 bigram 计数分别为 0.0000270，0.448，1.25，...8.26.

在图 3.8 中注意到，除了 0 和 1 的保留计数之外，只需从训练集中的计数中减去 0.75，就可以很好地估算出保留集中的所有其他二元语法计数！**绝对打折(absolut discunting)**是通过从每次计数中减去一个固定的(绝对的)折扣 d 来形式化这种直觉。直觉是，由于我们已经对非常高的计数有了很好的估计，一个小的折扣 d 不会对它们有太大的影响。它将主要修改较小的计数，对于这些计数我们不一定要相信其估值。图 3.8 表明，在实践中，这种折扣实际上对计数从 2 到 9 的 bigrams 是一个很好的折扣。适用于 bigrams 绝对折扣的插值公式：

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}\, v)} + \lambda(w_{i-1}) P(w_i) \tag{3.30}$$

第一项是折扣的 bigram，第二项是带插值权重λ的 unigram。我们可以将所有的 d 值都设置为 0.75，也可以为计数是 1 的 bigrams 单独保持 0.5 的折扣值。Kneser-Ney 折扣(Kneser 和 Ney，1995)以一种更复杂的方式来处理较低阶的 unigram 分布，从而增强了绝对折扣。假设我们正在对一个 bigram 和一个 unigram 模型进行插值，考虑一下预测下面这个句子中的下一个单词的工作。

I can’t see without my reading_____.

在这里，“glasses(眼镜)”这个词似乎比“Kong(孔)”这个词更容易出现，所以我们希望我们的 unigram

模型更喜欢“glasses”这个词。但事实上，更常见的是“Kong”，因为“Hong Kong(香港)”是一个非常高频的词。一个标准的 unigram 模型将赋予“Kong”比“glasses”更高的概率。我们想要捕捉的直觉是，虽然“Kong”是高频的，但主要只是高频在“Hong Kong”，即“Hong”一词之后。glass 这个词的分布范围要广得多。

换句话说，我们想创建一个称为 $P_{CONTINUATION}$ 的 unigram 模型，而不是创建 P(w)。$P_{CONTINUATION}$ 回答了“How likely is w to appear as a novel continuation?”问题，而 P(w)回答了"How likely is w?"问题。在一个新的看不见的上下文之中，我们如何估计将 w 这个词视为新延续(novel continuation)的概率? Kneser-Ney 的直觉是，将我们对 $P_{CONTINUATION}$ 的估计，建立在“number of different contexts word w has appeared in”上，即它完成的 bigram 类型的数量。每一种字母类型在第一次被看到时都是一种新延续。我们假设，过去出现在更多上下文中的单词，也更有可能出现在新的上下文中。单词 w 作为新延续出现的次数可以表示为：

$$P_{CONTINUATION}(w) \propto |\{v : C(vw) > 0\}| \tag{3.31}$$

为了将这个计数转化为一个概率，我们使用 bigram 类型的总数进行归一化。总而言之:

$$P_{CONTINUATION}(w) = \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u', w') > 0\}|} \tag{3.32}$$

基于不同隐喻的等效表述是使用在 w 之前出现的单词类型数量(重复公式 3.31)：

$$P_{CONTINUATION}(w) \propto |\{v : C(vw) > 0\}| \tag{3.33}$$

按所有单词前面的单词数进行归一化，如下所示：

$$P_{CONTINUATION}(w) = \frac{|\{v : C(vw) > 0\}|}{\sum_{w'} |\{v : C(v\,w') > 0\}|} \tag{3.34}$$

一个频词(Kong)出现在仅一个上下文中(Hong)的将具有较低的延续概率。

于是，针对 bigrams 的**插值的 Kneser-Ney** 平滑的最终方程为：

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i) \tag{3.35}$$

λ是一个归一化常数，用于分配我们已经打折的概率质量：

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)} |\{w : C(w_{i-1}w) > 0\}| \tag{3.36}$$

第一项 $d / \Sigma_v C(w_{i-1}\, v)$是归一化折扣，第二项$|\{w：C(w_{i-1}\, w)> 0\}|$是可以跟随 w_{i-1} 的单词类型的数量，或者等效地，我们已经打折的单词类型的数量；换句话说，就是我们使用归一化折扣次数的数量。

一般递推公式如下:

$$P_{KN}(w_i \mid w_{i-n+1:i-1}) = \frac{\max(c_{KN}(w_{i-n+1:i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1:i-1}\, v)} + \lambda(w_{i-n+1:i-1}) P_{KN}(w_i \mid w_{i-n+2:i-1}) \tag{3.37}$$

其中，计数 C_{KN} 的定义取决于我们是计算被插值的最高阶 n-gram(例如，如果我们正在插值的是 trigram、bigram 和 unigram，则是 trigram)，还是一个低阶的 n-gram(如果我们正在插值的是 trigram、bigram 和 unigram，则是 bigram 或 unigram):

$$c_{KN}(\cdot) = \begin{cases} \text{count}(\cdot) & \text{, for the highest order} \\ \text{continuationcount}(\cdot) & \text{, for lower orders} \end{cases} \tag{3.38}$$

延续计数是(·)的唯一单个单词上下文的数量。

在递归结束时，用均匀分布对 unigrams 进行插值，其中参数ε是空字符串：

$$P_{KN}(w) = \frac{\max(c_{KN}(w) - d, 0)}{\sum_{w'} c_{KN}(w')} + \lambda(\varepsilon)\frac{1}{V} \quad (3.39)$$

如果我们想包含一个未知词<UNK>，那么它只是作为一个计数为零的常规词汇条目，因此它的概率将是λ加权均匀分布λ(e)/V。

Kneser-Ney 平滑法的最佳性能版本被称为**修正的 Kneser-Ney** 平滑法，这是由 Chen 和 Goodman(1998)提出的。改进的 Kneser-Ney 并不使用单个固定折扣 d，而是对 n-gram 使用三种不同的折扣 d1、d2 和 d3 +，其计数分别对应为 1、2 和 3 或更多。详见 Chen 和 Goodman(1998，p.19)或 Heafield 等人(2013)。

3.6. 巨大的语言模型和傻瓜回退

通过使用来自网络或其他大量的集合的文本，可以建立非常大的语言模型。Google 发行的 Web 1 万亿 5-gram 语料库包括各种大的 n-gram 集合，包括从 1-gram 到 5-gram 的所有五词(five-word)序列单词，这些单词出现在至少 40 本不同的书中，这些书来自含有 1,024,908,267,229(约 1.024 万亿，译者注)个单词的文本，这些文本来自公开发行的可访问的英文网页(Franz 和 Brants，2006 年)。谷歌还发布了 Google Books Ngrams 语料库，其中包含从其图书收藏中提取的 n-gram，包括来自中文、英语、法语、德语、希伯来语、意大利语、俄语和西班牙语的另外 8000 亿个 n-gram 符记(Lin 等人，2012a)。更小但更精心策划的英语 n-gram 语料库包括来自 COCA(当代美国英语语料库)的 10 亿单词的美国英语语料库(Davies, 2020)中最常用的 100 万个 n-gram。COCA 是一个平衡的语料库，这意味着它有大致相同数量的来自不同体裁的词:1990-2019 年期间的网络、报纸、口语对话记录、小说等，并有每个 n-gram 的上下文，以及体裁和出处的标签)。

Google Web 语料库的一些 4-grams 示例：

4-gram	Count
serve as the incoming	92
serve as the incubator	99
serve as the independent	794
serve as the index	223
serve as the indication	72
serve as the indicator	120
serve as the indicators	45

在构建使用如此大的 n-gram 集合的语言模型时，效率考虑非常重要。不是将每个单词存储为字符串，而是在内存中表示为 64 位哈希数，单词本身存储在磁盘上。概率通常仅使用 4-8 位(而不是 8 字节浮点数)进行量化，n-grams 被存储在倒序前缀树(reverse tries)中。

N-gram 也可以通过修剪来缩小，例如仅存储计数大于某个阈值(例如，用于 Google n-gram 发布的计数阈值为 40)的 n-gram，或使用熵来修剪次要的 n-gram(Stolcke，1998)。另一个选择是使用**布隆过滤器(Bloom filters)**之类的技术来构建近似语言模型(Talbot 和 Osborne 2007，Church 等人 2007)。最后，高效的语言模型工具包，如 KenLM (Heafield 2011, Heafield 等人 2013)，使用排序数组，有效地将概率和回退合并到单个值中，并使用合并排序，以最少的次数通过大型语料库，有效地构建概率表。

尽管使用这些工具包可以使用完整的 Kneser-Ney 平滑法构建 web 级的语言模型，但 Brants 等人(2007)表明，对于非常大的语言模型，一个非常简单的算法可能就足够了。这个算法被称为**傻瓜回退(stupid backoff)**。傻瓜回退放弃了试图使语言模型成为真实概率分布的想法。高阶概率没有折扣。如果一个高阶 n-gram 的计数为零，我们就简单地回退到一个低阶 n-gram，并用固定的权重(与上下文无关)加权。该算法不会产生概率分布，因此我们将遵循 Brants 等人(2007)的观点 ，将其称为 S：

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{count(w_{i-k+1}^{i})}{count(w_{i-k+1}^{i-1})}, if \ \ count(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i \mid w_{i-k+2}^{i-1}), otherwise \end{cases} \tag{3.40}$$

回退以 unigram 结束，其概率 S(w) = count(w)/N。Brants 等人(2007)发现，λ的值取 0.4 为好。

3.7. 高级专题：困惑度与熵的关系

我们在 3.2.1 节中引入了困惑度，作为评估测试集上的 n-gram 模型的一种方法。更好的 n-gram 模型是为测试数据分配更高概率的模型，而困惑度是测试集概率的归一化版本。困惑度度量实际上源于交叉熵的信息理论概念，它解释了困惑度的神秘性质(例如，为什么有逆概率？)及其与熵的关系。**熵(entropy)**是信息的度量。给定一个随机变量 X，其范围是我们预测的范围(单词、字母、词类、被称为 X 的集合)，并具有特定的概率函数，则将其称为 p(X)，即随机变量 X 的熵是：

$$H(X) = \sum_{x \in \chi} p(x) \log_2 p(x) \tag{3.41}$$

原则上，对数(log)可以在任何底数(base)下计算。如果我们使用以 2 为底数的对数，则得到的熵值将以比特来度量。一种直观的理解熵的方式是：在最优编码方案中，熵是编码某个决策或信息所需要的比特数的下界。

考虑一个来自标准信息理论教科书 Cover 和 Thomas(1991)的例子。想象一下，我们想对一场赛马下注，但是距离 Yonkers 赛马场太远了，所以我们想向博彩者发送一条短信，告诉他要投注的八匹马中的哪一匹。编码此消息的一种方法是仅使用马匹数字的二进制表示形式作为代码；因此，马 1 的编码是 001，马 2 是 010，马 3 是 011，依此类推，马 8 是 000。如果我们整天投注，而且每匹马都用 3 位编码，那么平均而言，我们每场比赛将发送 3 位。

我们可以做得更好吗？假设价差(spread)是所下注的实际分布，我们将其表示为每匹马的先验概率，如下所示：

Horse 1	1/2	Horse 5	1/64
Horse 2	1/4	Horse 6	1/64
Horse 3	1/8	Horse 7	1/64
Horse 4	1/16	Horse 8	1/64

遍及马的随机变量 X 的熵为我们提供了位数的下限，为：

$$\begin{aligned} H(X) &= -\sum_{i=1}^{i=8} p(i) \log p(i) \\ &= -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{8}\log\frac{1}{8} - \frac{1}{16}\log\frac{1}{16} - 4(\frac{1}{64}\log\frac{1}{64}) \\ &= 2 \text{ bits} \end{aligned} \tag{3.42}$$

一个平均每场比赛 2 位(bits)的代码可以用更短的编码来表示更多的马，用更长的编码来表示更少的马。例如，我们可以将最有可能的马编码为 0，其余的马编码为 10，然后是 110、1110、111100、111101、111110 和 111111。

如果马的概率是相等的呢?我们在上面看到，如果我们用等长二进制编码来表示马的数目，那么每匹马需要 3 个比特来编码，所以平均值是 3。熵是一样的吗?在这种情况下，每匹马的概率是 1/8。这样选择马匹的熵为：

$$H(X) = -\sum_{i=1}^{i=8} \frac{1}{8}\log\frac{1}{8} = -\log\frac{1}{8} = 3 \text{ bits} \tag{3.43}$$

到目前为止，我们一直在计算单个变量的熵。但是我们用熵来做的大部分事情都涉及到序列。例如，

对于语法，我们将计算单词 W = {w_0,w_1,w_2，…，w_n}序列的熵。一种实现方法是使变量在单词序列范围内变化。例如，我们可以计算某种语言 L 中长度为 n 的所有有限单词序列的随机变量的熵，如下所示：

$$H(w_1,w_2,...,w_n)=\sum_{W_1^n\in L}p(W_1^n)\log p(W_1^n) \tag{3.44}$$

我们可以将**熵率(entropy rate)**(也可以将其视为每个单词的熵)定义为该序列的熵除以单词数：

$$\frac{1}{n}H(W_1^n)=-\frac{1}{n}\sum_{W_1^n\in L}p(W_1^n)\log p(W_1^n) \tag{3.45}$$

但是为了测量一种语言的真实熵，我们需要考虑无限长度的序列。如果我们把语言看作一个随机过程 L，它产生一个单词序列，并允许 W 代表单词 w_1,w_2，…，w_n 的序列，那么 L 的熵率 H(L)定义为：

$$\begin{aligned}H(L)&=\lim_{n\to\infty}\frac{1}{n}H(w_1,w_2,...,w_n)\\&=-\lim_{n\to\infty}\frac{1}{n}\sum_{W\in L}p(w_1,w_2,...,w_n)\log p(w_1,w_2,...,w_n)\end{aligned} \tag{3.46}$$

Shannon-McMillan-Breiman 定理(Algoet 和 Cover 1988，Cover 和 Thomas 1991)指出，如果语言在某些方面是规则的(确切地说，既是固定的又是遍历的)，那么：

$$H(L)=\lim_{n\to\infty}-\frac{1}{n}\log p(w_1,w_2,...,w_n) \tag{3.47}$$

也就是说，我们可以采用足够长的单个序列，而不是对所有可能的序列求和。Shannon-McMillan-Breiman 定理的直觉是，足够长的单词序列将在其中包含许多其他较短的序列，并且这些较短的序列中的每一个将根据其概率在较长的序列中再次出现。

如果随机过程分配给序列的概率相对于时间索引的变化是不变的，则称其为**平稳(stationary)**过程。换句话说，单词在时间 t 的概率分布与在时间 t + 1 的概率分布相同。马尔可夫模型，进而 n-grams，是平稳的。例如，在 bigram 中，P_i 仅取决于 P_{i-1}。因此，如果我们将时间索引移动 x，则 P_{i+x} 仍然取决于 P_{i+x-1}。但是自然语言并不是平稳的，因为正如我们在第 12 章中所展示的那样，即将出现的单词的概率可能取决于任意距离和时间相关的事件。因此，我们的统计模型仅给出自然语言的正确分布和熵的近似值。

总之，通过做一些不正确但方便的简化假设，我们可以计算随机过程的熵(通过取一个很长的输出样本并计算其平均对数概率的方式)。

现在，我们准备介绍**交叉熵(cross-entropy)**。当我们不知道产生某些数据的实际概率分布 p 时，交叉熵很有用。它允许我们使用一些 m，它是 p 的模型(即 p 的近似值)。m 在 p 上的交叉熵定义为：

$$H(p,m)=\lim_{n\to\infty}-\frac{1}{n}\sum_{W\in L}p(w_1,w_2,...,w_n)\log m(w_1,w_2,...,w_n) \tag{3.48}$$

也就是说，我们根据概率分布 p 来描述序列，但根据 m 来对它们的概率的对数进行求和。

再次，按照 Shannon-McMillan-Breiman 定理，进行平稳的遍历过程：

$$H(p,m)=\lim_{n\to\infty}-\frac{1}{n}\log m(w_1,w_2,...,w_n) \tag{3.49}$$

这意味着，对于熵，我们可以用一个足够长的单个序列来估计模型 m 在某个分布 p 上的交叉熵，而不是对所有可能的序列求和。交叉熵之所以有用是因为交叉熵 H(p,m)是熵 H(p)的上界。对于任何模型 m:

$$H(p)\le H(p,m) \tag{3.50}$$

这意味着我们可以使用一些简化的模型 m 来帮助估计(根据概率 p 来描述的符号序列的)真实熵。m 越

精确，交叉熵 H(p,m)就越接近真实熵 H(p)。因此，H(p,m)和 H(p)之间的差异是对模型精确程度的衡量。在两个模型 m_1 和 m_2 中，交叉熵越小的模型越准确。(交叉熵永远不会低于真实熵，因此模型不会因低估真实熵而犯错。)

我们终于可以看到困惑度和交叉熵之间的关系，如我们在公式 3.49 中看到的那样。当观察到的单词序列的长度达到无穷大时，交叉熵被定义为极限。我们将需要一个近似的交叉熵，依赖于一个固定长度的(足够长的)序列。对单词序列 W 的模型 $M = P(w_i \mid w_{i-N+1} \ldots w_{i-1})$，这种近似的交叉熵为：

$$H(W) = -\frac{1}{N}\log P(w_1, w_2, \ldots, w_N) \tag{3.51}$$

一个关于单词 W 序列的模型 P 的困惑度，现在正式定义为这个交叉熵的表达式:

$$\begin{aligned} \text{Perplexity}(W) &= 2^{H(W)} \\ &= P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}} \\ &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1, \ldots, w_{i-1})}} \end{aligned} \tag{3.52}$$

3.8. 总结

本章介绍了语言建模和语言处理中使用最广泛的工具之一 n-gram。

•语言模型提供了一种方法来为一个句子或其他单词序列分配概率，并从前面的单词中预测一个单词。

•n-gram 是马尔科夫模型，从一个固定窗口的前单词来估计单词。n-gram 概率可以通过在语料库中计数和归一化(**最大似然估计**)来估计。

•n-gram **语言模型**在某些任务中被外部评估，或者在内部使用**困惑度**。

•根据语言模型的测试集的困惑度是由模型计算出的反测试集概率的几何平均值。

•**平滑**算法提供了一种更复杂的方法来估计 n-grams 的概率。常用的 n-grams 平滑算法依赖于通过**回退或插值**的低阶 n-grams 的计数。

•回退和插值都需要**打折**来创建概率分布。

•**Kneser-Ney** 平滑处理利用了单词是新延续的可能性。插值的 Kneser-Ney 平滑算法将打折概率与低阶连续概率混合在一起。

3.9. 文献和历史说明

n-gram 的基本数学原理最早是由马尔科夫(1913)提出的，他使用现在称为马尔科夫链(bigrams 和 trigrams)的方法来预测普希金的《尤金·奥涅金》(Eugene Onegin)中即将出现的字母是元音还是辅音。马尔科夫将 20,000 个字母分类为 V 或 C，并计算出给定字母为前一个或两个字母后的元音的 bigram 和 trigram 概率。Shannon(1948)应用 n-gram 来计算英语单词序列的近似值。基于 Shannon 的工作，到 1950 年代，马尔可夫模型已广泛用于工程学，语言学和心理工作中，用于对单词序列进行建模。从 Chomsky (1956)开始，包括 Chomsky(1957)、Miller 与 Chomsky(1963)在内的一系列极具影响力的论文中，Noam Chomsky 认为，“有限状态马尔可夫过程”虽然可能是有用的工程启发式，但不能成为人类语法知识的完整认知模型。这些争论导致许多语言学家和计算语言学家几十年来忽略了统计建模方面的工作。

n-gram 模型的复兴来自于 Jelinek 和他在 IBM 托马斯·j·沃森研究中心的同事，他们受到了香农和卡内基梅隆大学贝克的影响，而 Baker 又受到了 Baum 和他同事的影响。这两个实验室分别成功地在他们的语音识别系统中使用了 n-gram (Baker 1975b, Jelinek 1976, Baker 1975a, Bah 等人 1983, Jelinek 1990)。

加一平滑法源自拉普拉斯(Laplace)1812 年的继承法则，是 Jeffreys(1948 年)根据 Johnson(1932 年)较早提出的加 k 建议首次将其作为工程解决方案应用于零频率问题的。Gale 和 Church(1994)总结了加一算法的问题。

在 80 年代和 90 年代提出了各种不同的语言建模和平滑技术，其中包括 Good-Turing 折扣--由 Katz 首次应用于 IBM 在 n-gram 上的平滑处理(Nadas 1984, Church 和 Gale 1991) --Witten-Bell 折扣(Witten and Bell,1991)、以及使用词类信息的各种**基于分类的 n 元语法(class-based n-gram)**模型。

从 1990 年代后期开始，Chen 和 Goodman 进行了许多精心控制的实验，比较了不同的折扣算法、缓存模型、基于分类的模型和其他语言模型参数(Chen 和 Goodman 1999，Goodman 等人 2006)。他们展示了改进的 Kneser-Ney 插值的优点，它成为 n-gram 语言建模的标准基线，特别是因为他们表明缓存和基于分类的模型仅提供了较小的额外改进。这些文章推荐给任何对 n-gram 语言建模有进一步兴趣的读者。SRILM (Stolcke, 2002)和 KenLM (Heafield 2011, Heafield 等人 2013)是用于构建 n-gram 语言模型的公开可用工具包。

现代语言建模通常使用神经网络语言模型来完成，它解决了 n-gram 的主要问题：随着 n-gram 阶数的增加，参数数量呈指数增长，n-gram 无法生成从训练到测试的数据集。相反，神经语言模型将单词投射到一个连续的空间中，其中具有相似上下文的单词具有相似的表示形式。我们将在第 7 章中介绍前馈语言模型(Bengio 等人 2006，Schwenk 2007)，并在第 9 章中介绍递归语言模型(Mikolov 2012)。

3.10. 练习

3.1 Write out the equation for trigram probability estimation (modifying Eq. 3.11).

Now write out all the non-zero trigram probabilities for the I am Sam corpus on page 32.

3.2 Calculate the probability of the sentence i want chinese food. Give two probabilities, one using Fig. 3.2 and the ‘useful probabilities’ just below it on page 34, and another using the add-1 smoothed table in Fig. 3.6. Assume the additional add-1 smoothed probabilities *P*(i|<s>) = 0.19 and *P*(</s>|food) =0.40.

3.3 Which of the two probabilities you computed in the previous exercise is higher, unsmoothed or smoothed? Explain why.

3.4 We are given the following corpus, modified from the one in the chapter:

<s> I am Sam </s>

<s> Sam I am </s>

<s> I am Sam </s>

<s> I do not like green eggs and Sam </s>

Using a bigram language model with add-one smoothing, what is P(Sam | am)? Include <s> and </s> in your counts just like any other token.

3.5 Suppose we didn’t use the end-symbol </s>. Train an unsmoothed bigram grammar on the following training corpus without using the end-symbol </s>:

<s> a b

<s> b b

<s> b a

<s> a a

Demonstrate that your bigram model does not assign a single probability distribution across all sentence lengths by showing that the sum of the probability of the four possible 2 word sentences over the alphabet *{*a,b*}* is 1.0, and the sum of the probability of all possible 3 word sentences over the alphabet *{*a,b*}* is also 1.0.

3.6 Suppose we train a trigram language model with add-one smoothing on a given corpus. The corpus contains V word types. Express a formula for estimating P(w3|w1,w2), where w3 is a word which follows the bigram (w1,w2), in terms of various N-gram counts and V. Use the notation c(w1,w2,w3) to denote the number of times that trigram (w1,w2,w3) occurs in the corpus, and so on for bigrams and unigrams.

3.7 We are given the following corpus, modified from the one in the chapter:

<s> I am Sam </s>

<s> Sam I am </s>

<s> I am Sam </s>

<s> I do not like green eggs and Sam </s>

If we use linear interpolation smoothing between a maximum-likelihood bigram model and a maximum-likelihood unigram model with $\lambda_1 = 1/2$ and $\lambda_2 = 1/2$, what is P(Sam|am)? Include <s> and </s> in your counts just like any other token.

3.8 Write a program to compute unsmoothed unigrams and bigrams.

3.9 Run your n-gram program on two different small corpora of your choice (you might use email text or newsgroups). Now compare the statistics of the two corpora. What are the differences in the most common unigrams between the two? How about interesting differences in bigrams?

3.10 Add an option to your program to generate random sentences.

3.11 Add an option to your program to compute the perplexity of a test set.

3.12 You are given a training set of 100 numbers that consists of 91 zeros and 1 each of the other digits 1-9. Now we see the following test set: 0 0 0 0 0 3 0 0 0 0. What is the unigram perplexity?

4. 朴素贝叶斯与情感分类

分类是人类和机器智能的核心。决定将什么字母、单词或图像呈现给我们的感官，识别面部或声音；分类邮件，分配家庭作业等级， 这些都是将类别分配给输入的示例。寓言家 Jorge Luis Borges(1964)强调了这项任务的潜在挑战，他设想将动物分为以下几种:

(a) 属于皇帝的人，(b) 不朽的人，(c) 受过训练的人，(d) 乳猪，(e) 美人鱼，(f) 神话般的人，(g) 流浪狗，(h) 包括在此类别中的那些，(i) 像疯了似的发抖的人，(j) 数不胜数的人，(k) 用非常细的骆驼毛刷画的人，(l) 其他的，(m) 那些刚刚把花瓶弄坏了的东西，(n) 像远处的苍蝇。

许多语言处理任务都涉及到分类，不过幸运的是，与 Borges 的类相比，我们的类更容易定义。本章将介绍朴素贝叶斯算法并将其应用于**文本分类(text categorization)**，即为整个文本或文档分配**标签（label）**或类别的任务。

我们聚焦于一个常见的文本分类任务**情感分析(sentiment analysis)**、情感提取、作者对某一对象表达的积极或消极倾向。网上对电影、书籍或产品的评论表达了作者对该产品的情感，而社论或政治文本则表达了作者对候选人或政治行动的情感。因此，从市场到政治领域，提取消费者或公众情感都是相关的。

情感分析最简单的版本是一种二进制分类任务，而评论中的单词提供了极好的线索。例如，考虑下面来自对电影和餐馆的正面和负面评论中提取出来的短语。有些词像伟大的、丰富的、令人敬畏的、可悲的、可怕的和荒谬的，是非常有益的线索：

+ ...滑稽的人物和丰富的讽刺作品，以及一些曲折的情节
- 太可悲了。最糟糕的部分是拳击场面。
+ ...很棒的焦糖酱和甜甜的烤杏仁。我喜欢这个地方！
- ...糟糕的披萨和高得离谱的价格...

垃圾邮件检测(spam detection)是另一个重要的商业应用程序，它是二进制分类任务，用于将电子邮件分配给垃圾邮件或非垃圾邮件这两个类中的一个。可以使用许多词汇和其他特征来进行这种分类。例如，你可能会相当合理地怀疑一封包含“在线药品”或“不需要任何费用”或“亲爱的赢家”等短语的电子邮件。

我们可能想知道的另一件事是文本的编写语言。例如，社交媒体上的文本可以是多种语言，我们需要进行不同的处理。因此，**语言 ID(language id)**的任务是大多数语言处理流水线中的第一步。相关的文本分类任务，例如**作者身份归属(authorship attribution)**(确定文本的作者)，也与数字人文科学、社会科学和法医语言学有关。

最后，文本分类中最古老的任务之一是为文本分配图书馆主题类别或主题标签。确定研究论文涉及流行病学还是胚胎学，是信息检索的重要组成部分。存在各种类别的主题集，例如 MeSH(医学主题词)同义词典。实际上，正如我们将要看到的那样，主题类别分类是朴素贝叶斯算法于 1961 年发明的任务。

分类对于文档级别以下的任务也很重要。我们已经看到了句点消歧(确定句点是句子的结尾还是单词的一部分)和单词符记(确定字符是否应该是单词边界)。甚至语言建模也可以看作是一种分类:每个单词都可以看作一个类：每个下一个单词都被认为是一个类，因此，预测下一个单词就是，将到目前为止的上下文(context-so-far)，分类到每个后续单词(each next word)的类别中。词类**标记器(tagger)**(第八章)就是将一个词在句子中的每一个出现归类为名词或动词。

分类的目的是进行单个观察，提取一些有用的特征，从而将这个观察分类到一组离散类集合中的一个。分类文本的一种方法是使用手写规则。在语言处理的许多领域中，基于规则的手写分类器构成了最先进(SOTA)的系统，或者至少是其中的一部分。

然而，规则可能是脆弱的，因为情况或数据会随着时间的推移而变化，而且对于某些任务，人类不一定擅长提出规则。语言处理中的大多数分类都是通过**监督机器学习(supervised machine learning)**完成的，这将是本章剩下部分的主题。在监督学习中，我们有一组输入观察数据，每一个都与一些正确的输出(“监督信号”)相关联。该算法的目标是学习如何从一个新的观察映射到一个正确的输出。

形式上，监督分类的任务是获取一个输入 x 和一组固定的输出类别 Y=(y_1，y_2，...，y_M)并返回预测的类别 y∈Y。对于文本分类，我们有时会谈论 c(用于“类”)代替 y 作为我们的输出变量，而 d(用于“文档”)代替 x 作为我们的输入变量。在有监督的情况下，我们有 N 个文档的训练集，每个文档都手工标记了一个类：(d_1，c_1)，....，(d_N，c_N)。我们的目标是学习一个能够将新文档 d 映射到其正确类别 c∈C 的分类器。概率分类器还会告诉我们观察结果在类中的概率。在类上的全部分配可以为下游决策提供有用的信息。合并系统时，避免早期做出离散决策会很有用。

许多机器学习算法被用来建立分类器。本章介绍了朴素贝叶斯;下面介绍逻辑回归。这些例子说明了两种分类的方法。像朴素贝叶斯这样的**生成式分类器(Generative classifiers)**构建了一个类如何生成输入数据的模型。给定一个观察结果，它们返回最有可能生成该观察结果的类。而像逻辑回归这样的**区分式分类器(Discriminative classifiers)**则会从输入中了解哪些特征对区分不同可能的类最有用。而区分系统通常更准确，因此更常用，生成式分类器仍然起作用。

4.1.朴素贝叶斯分类器

在本节中，我们介绍多项式朴素贝叶斯分类器，之所以这么称呼是因为它是贝叶斯分类器，它对要素之间的相互作用进行了简化(朴素)假设。

分类器的直觉如图 4.1 所示。我们将文本文档表示为一个词袋，即一个无序单词的集合，其位置被忽略，仅在文档中保持其出现频率。在该图的示例中，我们没有在“I love this movie” 和“I would recommend it”之类的所有短语中表示单词顺序，而是简单地注意到在整个摘录中，单词 I 出现了 5 次，it 出现 6 次，love、recommend 和 movie 各出现 1 次，依此类推。

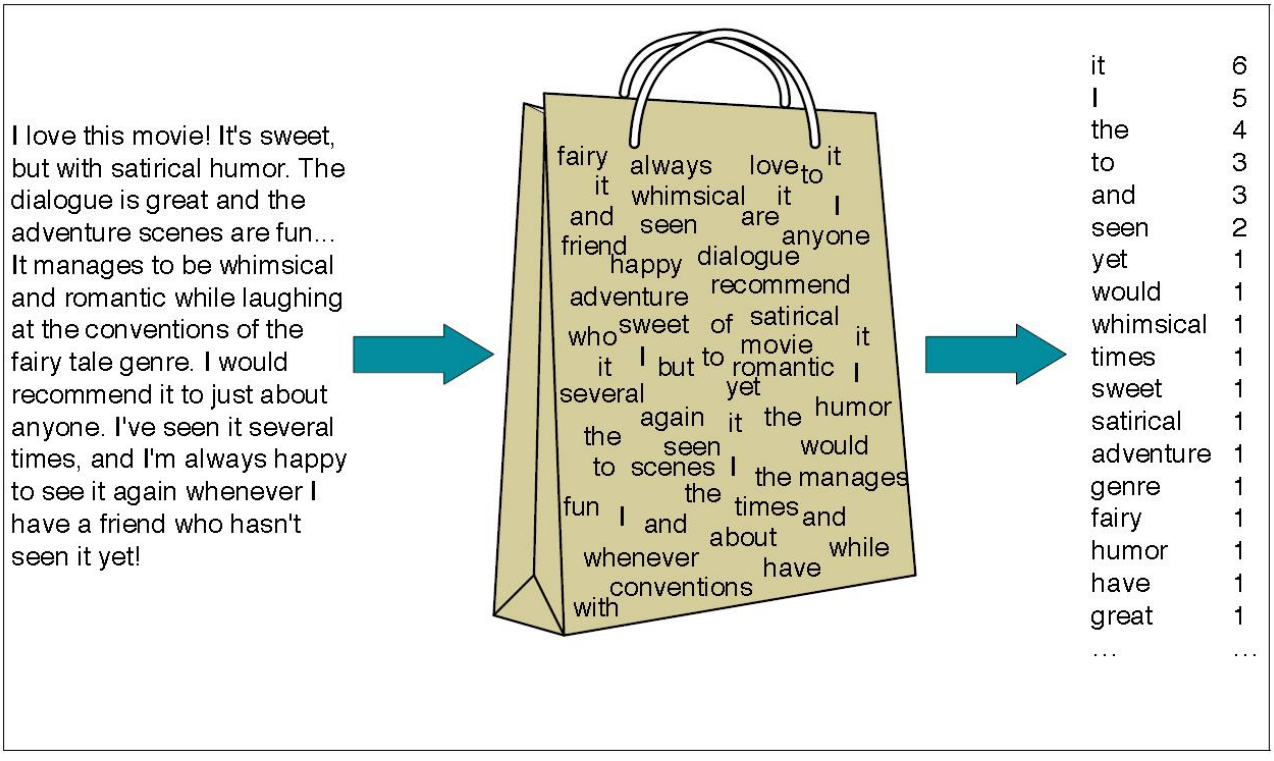


图 4-1：多项式朴素贝叶斯分类器的直观感受(词袋假设)

图注：应用于电影评论的多项式朴素贝叶斯分类器的直观感受，单词的位置被忽略(词袋假设)，我们利用每个单词的频率。

朴素贝叶斯是一个概率分类器，这意味着对于文档 d，在所有类别 c∈C 中，分类器返回具有给定文档最大后验概率的类别 $\hat{c}$。在公式 4.1 中，我们使用帽子符号ˆ来表示“我们对正确类别的估计”。

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) \tag{4.1}$$

贝叶斯推理(Bayesian inference)的这一思想在贝叶斯(1763)的著作中就已经为人所知，并由

Mosteller 和 Wallace(1964)首次应用于文本分类。贝叶斯分类的直觉是使用贝叶斯规则将公式 4.1 转换成其他具有一些有用性质的概率。贝叶斯规则如公式 4.2 所示;它为我们提供了一种将任何条件概率 P(x|y)分解为其他三种概率的方法:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{4.2}$$

我们可以将公式 4.2 代入公式 4.1 得到公式 4.3:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} \tag{4.3}$$

我们可以通过去掉分母 P(d)来简化公式 4.3。这是可能的，因为我们将为每个可能的类计算 P(d|c)P(c)/P(d)。但是 P(d)不会因为每个类而改变;

我们总是关心同一文档 d 的最有可能的类，这个类必须具有相同的概率 P(d)。

因此，我们可以选择使这个简单公式最大化的类:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c) \tag{4.4}$$

我们将朴素贝叶斯称为生成模型，因为我们可以将公式 4.4 解读为一种关于文档如何生成的隐含假设:首先从 P(c)中采样一个类，然后从 P(d|c)中采样生成单词。(事实上，我们可以想象通过这个过程生成人工文档，或者至少生成它们的单词计数)。我们将在第 5 章更多地讨论生成模型的本质。

回到分类话题：给定某些文档 d，我们通过选择一个类来计算最可能的类$\hat{c}$，即$\hat{c}$具有以下两个概率的最高乘积：文档的**似然度(likelihood)**P(d|c)和类的**先验概率(prior probability)**P(c)：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{4.5}$$

不失一般性，我们可以将文档 d 表示为一组特征 f_1，f_2，...，f_n：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{4.6}$$

不幸的是， 公式 4.6 仍然很难直接计算：如果不做一些简化的假设，则估计特征的每种可能组合(例如，每个可能的单词和位置的集合)的概率将需要大量的参数和不可能的庞大训练集。因此，朴素贝叶斯分类器制订了(make)两个简化的假设。

第一个假设是上文直观讨论的词袋假设:我们假设位置无关紧要，并且对于单词“love”，无论出现在文档中的第 1 个、第 20 个还是最后一个单词，都对分类有相同的影响。因此我们假设特征 f_1，f_2，...，f_n 只编码词的身份，不编码词的位置。

第二个假设通常被称为**朴素贝叶斯假设(naive Bayes assumption)**:这是一个条件独立假设，即概率 $P(f_i|c)$在给定类 c 的情况下是独立的，因此可以朴素地作如下乘法:

$$P(f_1, f_2, \ldots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \ldots \cdot P(f_n|c) \tag{4.7}$$

因此，朴素的贝叶斯分类器为该类选择的最终方程为：

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c) \tag{4.8}$$

要将朴素的贝叶斯分类器应用于文本，我们需要通过简单地遍历文档中每个单词位置的索引来考虑单词位置：

$$\begin{aligned} \text{positions} &\leftarrow \text{all word positions in test document} \\ c_{NB} &= \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c) \end{aligned} \tag{4.9}$$

朴素贝叶斯的计算，就像语言建模的计算，是在对数空间中完成的，以避免下溢和提高速度。因此公式 4.9 通常被表示为：

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c) \tag{4.10}$$

通过考虑对数空间中的特征，公式 4.10 把预测类的计算当作输入特征的线性函数。使用输入的线性组合来做出分类决策的分类器(如朴素贝叶斯和逻辑回归)被称为**线性分类器(linear classifiers)**。

4.2. 朴素贝叶斯分类器的训练

我们如何学到概率 P(c)和 $P(f_i|c)$?让我们首先考虑最大似然估计。我们将简单地使用数据中的频率。对于 P(c)之前的类，我们要求(ask)训练集中文档在每个类 c 中的百分比。设 N_c 为训练数据中文档的数量，N_{doc} 为文档的总数。然后:

$$\hat{P}(c) = \frac{N_c}{N_{doc}} \tag{4.11}$$

要了解概率 $P(f_i|c)$，我们假设，一个特征只是文档词袋中一个单词的存在，因此我们需要 $P(w_i|c)$，将其计算为次数的分数，这个次数是指，在主题 c 的所有文档的所有单词中出现单词 w_i 的次数。我们首先将所有具有类别 c 的文档拼接为一个大的“类别 c”文本。然后，我们使用此拼接文档中 w_i 的频率来给出概率的最大似然估计：

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)} \tag{4.12}$$

这里的词汇表 V 由所有类中的所有单词类型的联合组成，而不是只有一个 c 类的单词。

然而，最大似然训练存在一个问题。想象一下，我们试图估计给定类别为阳性的单词“fantastic”的似然度，但假设没有既包含单词“fantastic”又被分类为阳性的训练文档。也许“fantastic”这个词碰巧(讽刺地?)出现在阴性类别中。在这种情况下，该特征的概率将为零:

$$\hat{P}(\text{“fantastic”}|\text{positive}) = \frac{count(\text{“fantastic”}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0 \tag{4.13}$$

但是，由于朴素贝叶斯天真地将所有特征的可能性相乘在一起，任何类的可能性术语中的零概率将导致该类的概率为零，不管其他证据是什么!

最简单的解决方案是第 3 章中介绍的加一(Laplace)平滑。尽管在语言建模中通常用更复杂的平滑算法来代替 Laplace 平滑，但它在朴素贝叶斯文本分类中常被使用：

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|} \tag{4.14}$$

再次注意，至关重要的是，词汇表 V 由所有类中的所有单词类型组成，而不仅仅是一个类 c 中的单词(试着说服自己为什么这必须是真的;见本章末尾的练习)。

对于出现在测试数据中但根本不在词汇表中的单词，因为它们在任何类的任何训练文档中都没有出现，我们该怎么办？ 解决此类**未知词(unkown word)**的方法是忽略它们-从测试文档中删除它们，根本不包含任何概率。

最后，某些系统选择完全忽略另一类单词：**停用词(stop word)**，非常常见的词诸如 the 和 a。可以通过按训练集中的频率对词汇进行排序，然后将前 10-100 个词汇条目定义为停用词，或者使用许多可用的在线预定义停用词列表之一来完成。最后将这些停用词的每个实例都从训练和测试文档中删除，就好像它们从未发生过一样。但是，在大多数文本分类应用程序中，使用停用词列表并不能提高性能，因此更常见的是利用整个词汇表而不使用停用词列表。

图 4.2 显示了最终的算法。

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

图 4-2：使用加一平滑处理的朴素贝叶斯算法

图注：若要用加 α 平滑来代替，则请把+1 改为+α，以进行训练中的对数似然计数。

4.3. 实例

让我们来看一个使用加一平滑来训练和测试朴素贝叶斯的示例。我们将情感分析域分为正面(+)和负面(-)两个类，并从实际电影评论中简化以下微型训练和测试文档。

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

这两个类的先验 P(c)通过公式 4.11 作为 N_c/N_{doc} 计算:

$$P(-)=\frac{3}{5} \qquad P(+)=\frac{2}{5}$$

单词 with 没有出现在训练集中，所以我们完全放弃它(如上所述，我们不使用未知词的朴素贝叶斯模型)。训练集之中剩下的三个单词“predictable”、“no”和“fun”的似然度，根据公式 4.14(计算训练集中剩余单词的概率留给读者作为练习)，可以得到如下结果：

$$P(\text{“predictable”}|-)=\frac{1+1}{14+20} \qquad P(\text{“predictable”}|+)=\frac{0+1}{9+20}$$

$$P(\text{“no”}|-)=\frac{1+1}{14+20} \qquad P(\text{“no”}|+)=\frac{0+1}{9+20}$$

$$P(\text{“fun”}|-)=\frac{0+1}{14+20} \qquad P(\text{“fun”}|+)=\frac{1+1}{9+20}$$

对于测试句 S = "predicted with no fun"，去掉“with”后，通过公式 4.9，所选类计算如下:

$$P(-)P(S|-) = \frac{3}{5}\times\frac{2\times2\times1}{34^3}=6.1\times10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5}\times\frac{1\times1\times2}{29^3}=3.2\times10^{-5}$$

因此，对于测试句，该模型预测的类是负面的(negative)。

4.4. 情感分析的优化

尽管标准的朴素贝叶斯文本分类可以很好地用于情感分析，但通常会进行一些小的更改以提高性能。首先，对于情感分类和许多其他文本分类任务，一个单词是否出现似乎比其出现频率更重要。因此，通常可以将每个文档中的单词计算限制为 1 来提高性能(有关这些结果，请参见本章结尾)。此变量称为二进制多项式朴素贝叶斯或二进制 NB。该变体使用了相同的公式 4.10，除了对于每个文档，我们在将它们连接到单个大文档之前，我们删除了所有重复的单词。图 4.3 显示了一个示例，其中将四个文档的集合(在此示例中是缩短文档和文本规范化的文档)重新映射为二进制，其修改后的计数如右表所示。该示例在没有加一平滑的情况下工作，以使差异更加清晰。请注意，结果计数不必为 1； 甚至对于二进制 NB，单词 great 的计数也为 2，因为它出现在多个文档中。

对情感进行文本分类时，通常要进行的第二个重要补充是处理否定(negation)。考虑一下“I really like this movie ”(正)和“I didn't like this movie”(负)之间的区别。用"didn't"表示的否定完全改变了我们从谓词"like"得出的推论。同样地，否定可以修饰一个消极的词，产生一个积极的评价(“don't dismiss this film, doesn't let us get bored”)。

在情感分析中通常使用一个非常简单的基线来处理否定语：在文本规范化过程中，在逻辑否定符记(n't, not, no, never)之后的每个单词前加前缀 NOT_，直到下一个标点符号。因此这句话：

didn’t like this movie , but I

变成：

didn’t NOT_like NOT_this NOT_movie , but I

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts		Binary Counts	
	+	−	+	−
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

图 4-3：二进制朴素贝叶斯算法的二值化示例

这样，新出现的诸如 NOT_like、NOT_recommend 的单词将更多地出现在负面文档中，并成为负面情绪的线索，而诸如“NOT_bored、NOT_dismiss 的单词将获得正面联想。我们将回到第 16 章去使用解析来更精确地处理这些否定词与它们所修饰的谓词之间的范围关系的方法，但是这种简单的基线在实践中效果很好。

最后，在某些情况下，如果使用训练集中的所有单词来估计积极和消极情绪，那么，我们可能没有足够的带有标签的训练数据来训练准确的朴素贝叶斯分类器。

在这种情况下，我们可以取而代之的是从**情感词汇(sentiment lexicons)**中提取正词和负词特征，

这些词表会预先添加正词或负词。在这种情况下，我们可以取而代之的是从情感词汇中提取正词和负词特征，这些词表会预先添加正词或负词。四种流行的词汇是 General Inquirer(Stone 等人，1966)、LIWC(Pennebaker 等人，2007)、Hu 和 Liu 的意见词汇 LIWC(2004a)和 MPQA 的主观词典(Wilson 等人，2005 年)。

例如，MPQA 主观性词汇有 6885 个词，2718 个正词和 4912 个负词，每个词都被标记为是强偏还是弱偏。MPQA 词汇中的一些正和负词汇包括:

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

- : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

在朴素的贝叶斯分类器中使用词汇的一种常见方法是添加一个特征，只要出现该词汇中的单词，该特征就会被计数。因此，我们可以添加一个名为“此词出现在正词汇中”的特征，并将词汇中所有单词的实例都视为该特征的计数，而不是分别计算每个单词。同样，我们可以在负词汇中添加“此词出现在负词汇中”作为第二个特征。如果我们有很多训练数据，并且测试数据与训练数据匹配，那么仅使用两个特征就不能像使用所有单词一样有效。但是，当训练数据稀疏或不能代表测试集时，使用密集词汇特征而不是稀疏的单个单词特征可能会更好地推广。

我们将在第 20 章回到词汇的使用，展示如何自动学习这些词汇，以及如何将它们应用于情感分类之外的许多其他任务。

4.5.朴素贝叶斯用于其他文本分类任务

在上一节中，我们指出朴素贝叶斯并不要求我们的分类器使用训练数据中的所有单词作为特征。事实

上，朴素贝叶斯的特征可以表示我们想要的输入文本的任何性质。

考虑一下垃圾邮件检测任务，即确定某封特定的邮件是否为垃圾邮件(不请自来的批量电子邮件)，以及朴素贝叶斯在文本分类中的第一个应用(Sahami 等人，1998)。

这里的一种常见解决方案是，不要将所有的单词作为单独的特征使用，而是将可能的单词或短语集预先定义为特征，并与非纯语言的特征相结合。例如，开源的 SpamAssassin 工具[1]预先定义了类似短语“百分之百保证(one hundred percent guaranteed)”或“提及数百万美元(mentions millions of dollars)”这样的特性，这是一个正则表达式，与可疑的大笔资金相匹配。但它也包括像“HTML 的文本与图像区域的低比率(HTML has a low ratio of text to image area)”这样的特征，这不是纯粹的语言，可能需要一些复杂的计算，或完全非语言的特征，例如，电子邮件到达的路径。更多 SpamAssassin 特征示例:

•电子邮件主题行均为大写字母
•包含紧急短语，例如“紧急回复”
•电子邮件主题行包含“在线药品”
•HTML 的“ head”标签不平衡
•声明你可以从列表中被删除

对于其他任务，例如语言 ID(确定给定一段文字使用哪种语言)，最有效的朴素贝叶斯特征根本不是单词，而是字符 n-gram，2-gram('zw')，3-gram('nya'，'Vo')或 4-gram('ięz'，'thei')，甚至是更简单的字节 n-gram，在这里，我们不使用称为代码点(codepoints)的多字节 Unicode 字符表示，而是假设所有内容都是原始字节字符串。因为空格可以算作一个字节，所以字节 n-gram 可以对单词的开头或结尾的统计信息建模。一种广泛使用的朴素贝叶斯系统叫 langid.py (Lui 和 Baldwin, 2012)，它是从长度为 1-4 的所有可能的 n-gram 开始，使用特征选择来筛选出信息最丰富的 7000 个最终特征。

语言 ID 系统在多语言文本上进行训练，如维基百科(使用了 68 种不同语言的维基百科文本(Lui 和 Baldwin, 2011))或新闻通讯社。为了确保此多语言文本正确反映不同的地区、方言和社会经济类别，系统还添加了在许多地区进行地理标记的多种语言的 Twitter 文本(对于从英语人口众多的国家--例如尼日利亚或印度--获取世界英语方言非常重要)、圣经(Bible)和古兰经(Quran)的翻译、俚语网站(例如 Urban Dictionary)、非裔美国人白话语料库(Blodgett 等人，2016)、还有其它(Jurgens 等人，2017)。

4.6. 朴素贝叶斯的语言模型

正如我们在前一节中看到的，朴素贝叶斯分类器可以使用任何类型的特征：词典、URL、电子邮件地址、网络特性、短语等等。但是，如前一节所述，如果我们只使用单个单词的特征，并且我们使用文本中的所有单词(不是一个子集)，那么朴素贝叶斯与语言建模有重要的相似之处。具体地说，一个朴素贝叶斯模型可以被看作是一组特定于类的 unigram 语言模型，其中每个类的模型实例化了一个 unigram 语言模型。由于朴素贝叶斯模型的似然特征为每个单词 P(word | c)分配了概率，因此该模型也为每个句子分配了概率：

$$P(s|c) = \prod_{i \in positions} P(w_i|c) \tag{4.15}$$

因此考虑一个朴素贝叶斯模型，其类为正(+)和负(-)，模型参数如下:

w	P(w|+)	P(w|-)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...	...	...

[1] https://spamassassin.apache.org

以上两列都实例化了一个语言模型，该模型可以为“I love this fun film”这句话分配概率:

P(“I love this fun film” | +) = 0.1×0.100×0.01×0.050×0.1 = 0.0000005000

P(“I love this fun film” | −) = 0.2×0.001×0.01×0.005×0.1 = 0.0000000010

碰巧的是，正模型赋予这句话更高的概率:P(s|pos) > P(s|neg)。请注意，这只是朴素贝叶斯模型的似然部分；一旦我们在先验中相乘，完全朴素贝叶斯模型可能会做出不同的分类决定。

4.7.评估：精度，召回率，F 量度

为了介绍评估文本分类的方法，让我们首先考虑一些简单的二进制检测任务。例如，在垃圾邮件检测中，我们的目标是将每个文本标记为属于垃圾邮件类别(“正”)或不属于垃圾邮件类别(“负”)。因此，对于每个项目(电子邮件文档)，我们需要知道我们的系统是否认为它是垃圾邮件。我们还需要知道电子邮件是否真的是垃圾邮件，即我们试图匹配的每个文档的人为定义的标签。我们将把这些人类标签称为**黄金标签(gold labals)**。

假设您是 Delicious Pie 公司的首席执行官，并且您需要了解人们在社交媒体上对您的馅饼(Pie)的评价，因此您可以构建一个系统来检测有关 Delicious Pie 的推文。在这里，正类是关于美味派的推文，负类是所有其他推文。

在这两种情况下，我们都需要一个度量标准来了解我们的垃圾邮件检测器(或 pie-tweet-detector)的工作情况。为了评估任何探测事物的系统，我们首先建立如图 4.4 所示的**混淆矩阵(confusion matrix)**。混淆矩阵是一个可视化表格，该表格展示了算法相对于人类黄金标签是如何执行的。表格使用了二个维度(系统输出和黄金标签)，并且每个单元标记了一组可能的结果。例如，在垃圾邮件检测情况下，真阳性(true positives)是指确实是垃圾邮件的文档(由人工创建的黄金标签指示)，而我们的系统正确地将其称为垃圾邮件。假阴性(False negatives)是指确实是垃圾邮件，但我们的系统错误地标记为非垃圾邮件的文档。

表格的右下角是准确度公式，它要求(ask)我们的系统正确标记的所有观察结果(对于垃圾邮件或 Pie 示例，意味着所有电子邮件或推特)的百分比。虽然准确度似乎是一个自然的度量标准，但我们通常不会将它用于文本分类任务。这是因为当类不平衡时，准确度就不能很好地发挥作用(就像垃圾邮件和推特一样，垃圾邮件是电子邮件的主要内容，而推特主要不是关于 Pie 的内容)。

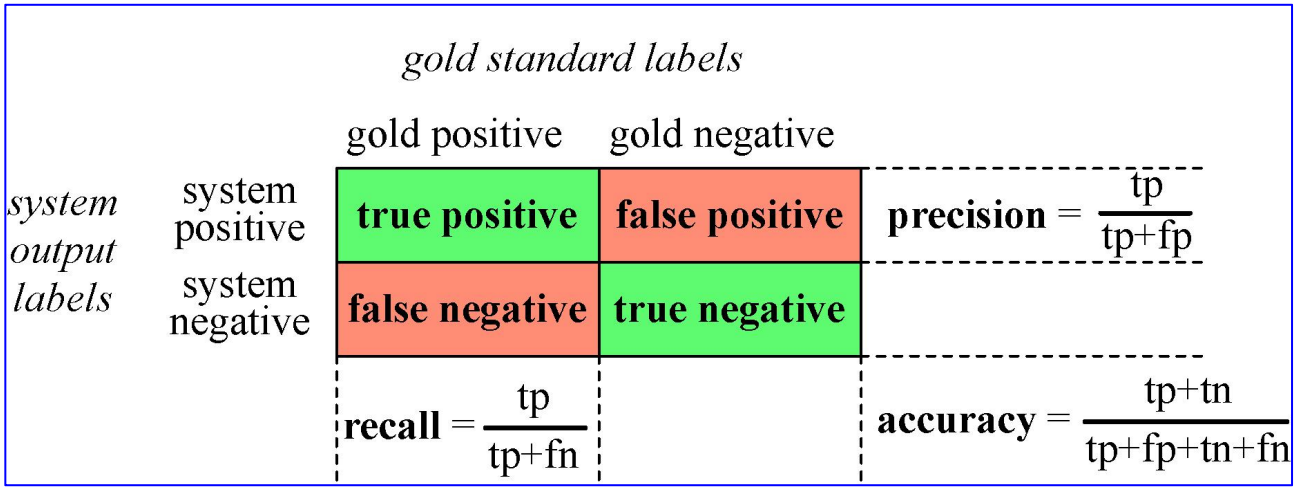


图 4-4：混淆矩阵在黄金标签下的表现

图注：一个混淆矩阵，用于可视化二进制分类系统在黄金标签下的表现。

更明确地说，想象我们看了 100 万条推文，假设其中只有 100 条在讨论他们对我们的 Pie 的爱(或恨)，而其他 999,900 条推文都是关于一些完全无关的事情。想象一下，一个简单的分类器傻傻地把每条推文都分类为“not about pie”。这个分类器将有 999,900 个真否定，只有 100 个假否定，准确度为 999,900/1,000,000 或 99.99%！多么惊人的准确度水平！我们真该对这个分类器感到高兴吗?然而，这个绝妙的“no pie”分类器将是完全无用的，因为它不会找到一个单独的我们正在寻找的客户意见。换句话说，如果目标是发现一些罕见的东西，或者至少在频率上不完全平衡(这是世界上很常见的情况)，那么准确度

就不是一个好的度量标准。

这就是为什么我们通常会转向图 4.4 中的精度和召回率这两个指标而不是准确度。**精度(precision)**衡量的是系统检测到的项目(即标记为阳性的系统)实际上是阳性(即根据人类的金标为阳性)的项目所占的百分比。精度定义为：

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

召回率(Recall)衡量输入中实际存在的，被系统正确识别的项目所占的百分比。召回率定义为：

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

精度和召回率将有助于解决无用的“nothing is pie”分类器的问题。尽管这个分类器的准确率高达 99.99%，但它有一个可怕的召回率为 0(因为没有真阳性结果，并且有 100 个假阴性结果，所以召回率为 0/100)。你应该说服自己，找到相关推文的精度也是有问题的。因此，与准确性不同，精度和召回率强调真阳性：找到我们应该寻找的东西。

有许多方法可以定义包含精度和召回率两个方面的单个度量标准。这些组合中最简单的是 **F-度量(F-measure)**(van Rijsbergen 1975)，定义为:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

β参数可能会根据应用程序的需求，以不同的方式加权召回率和精度的重要性。β> 1 的值有助于召回率，而β<1 的值有助于精度。当β= 1 时，精度和召回率是相等的； 这是最常用的指标，称为 F_β= 1 或仅称为 F_1：

$$\mathrm{F}_1 = \frac{2PR}{P+R} \tag{4.16}$$

F 度量来自精度和召回率的加权调和均值。一组数的调和均值是倒数算术平均数的倒数:

$$\text{HarmonicMean}(a_1, a_2, a_3, a_4, \ldots, a_n) = \frac{n}{\frac{1}{a_1}+\frac{1}{a_2}+\frac{1}{a_3}+\ldots+\frac{1}{a_n}} \tag{4.17}$$

因此 F 度量是：

$$F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}} \quad \text{or}\left(\text{with } \beta^2 = \frac{1-\alpha}{\alpha}\right) \quad F = \frac{(\beta^2+1)PR}{\beta^2 P + R} \tag{4.18}$$

使用调和均值是因为它是一个保守度量；两个值的调和均值比算术均值更接近这两个值的最小值。因此，这两个数字中较低的更重。

4.7.1. 用两个以上的类进行评估

到目前为止，我们只描述了两个类的文本分类任务。但语言处理中的许多分类任务都有两个以上的类。对于情感分析，我们通常分为 3 类(积极的、消极的、中性的)，甚至在词类标注、词义消歧、语义角色标签、情绪检测等任务中还会有更多的类。幸运的是，朴素贝叶斯算法已经是一个多类分类算法。

但是，我们需要稍微修改一下精度和召回率的定义。考虑图 4.5 所示的假设 3 种单向电子邮件分类决策(紧急，正常，垃圾邮件)的样本混淆矩阵。矩阵显示，例如，系统错误地将一个垃圾邮件文档标记为紧急文档，并且我们展示了如何为每个类别计算不同的精度和召回率。为了得出一个告诉我们系统运行状况的度量标准，可通过两种方式组合这些值。在**宏平均(macroaveraging)**中，我们计算每个类别的性能，然

后对各个类别进行平均。在**微平均(microaveraging)**中，我们将所有类的决策收集到一个混淆矩阵中，然后从该表中计算精度和召回率。图 4.6 分别显示了每个类的混淆矩阵，及其微平均和宏观平均精度的计算。

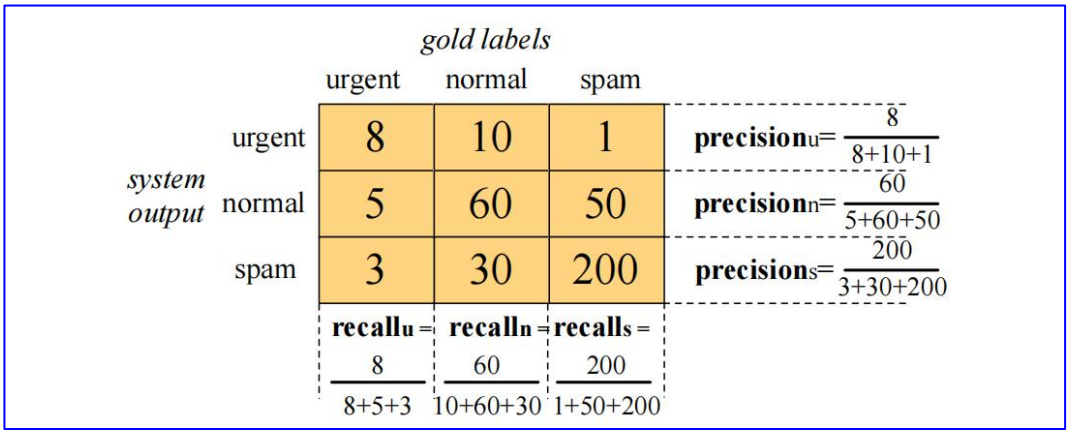


图 4-5：三类分类任务的混淆矩阵

图注：图中显示了对于每一对类(c_1，c_2)，c_1 中有多少文档被正确分配给 c_2。

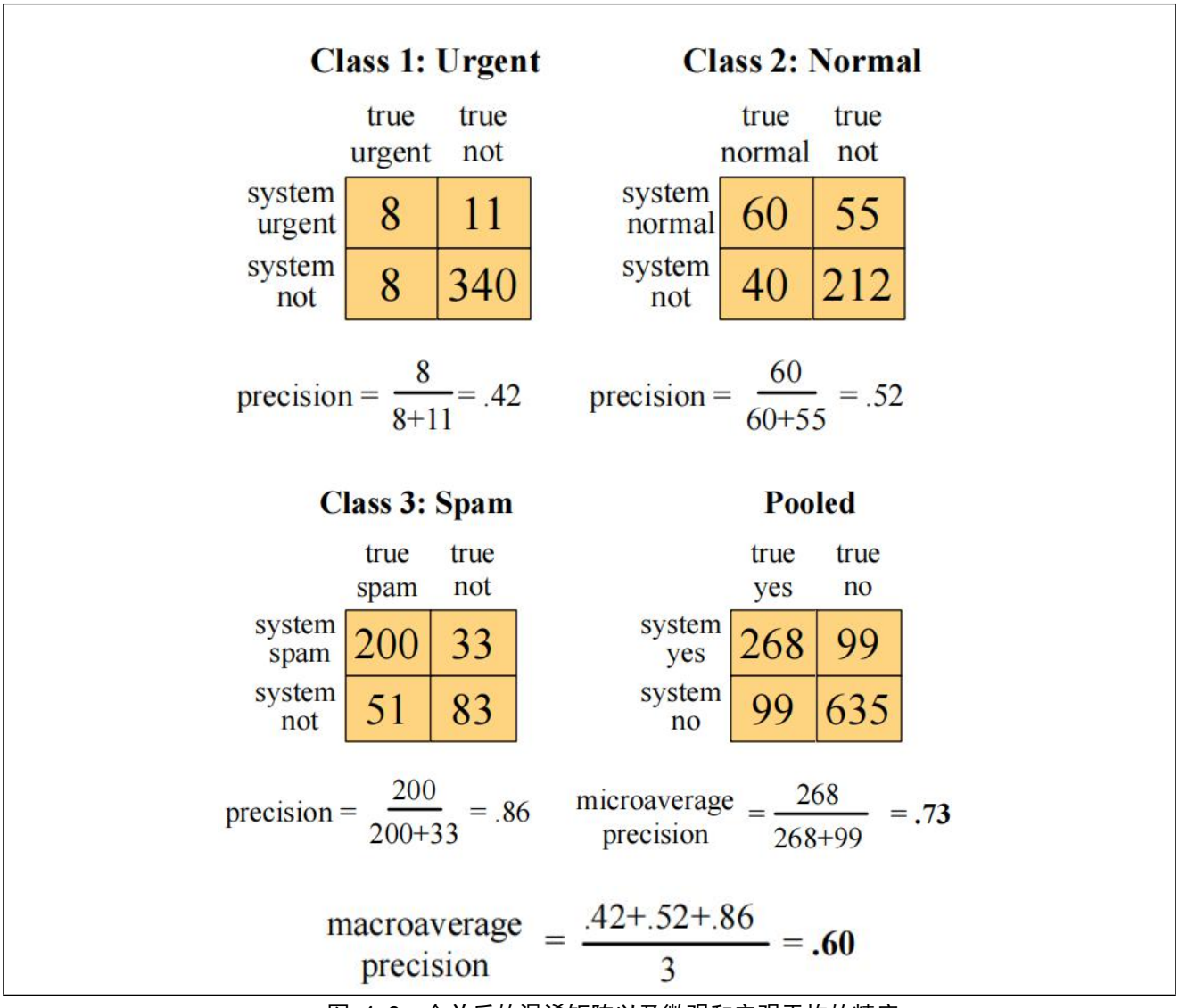


图 4-6：合并后的混淆矩阵以及微观和宏观平均的精度

图注：将 3 类混淆矩阵与上一张图分开，显示了合并后的混淆矩阵以及微观平均和宏观平均的精度。

如图所示，由于计数是合并的，因此微平均值由更频繁的类(在这种情况下为垃圾邮件)所主导。宏平均值更好地反映了较小类的统计信息，因此，当所有类的性能同等重要时，它更合适。

4.8. 测试集和交叉验证

文本分类的训练和测试过程遵循我们在语言建模中看到的内容(第 3.2 节)：我们使用训练集训练模型，然后使用**开发测试集(devset)**来调整一些参数，最后从总体上决定最好的模型是什么。一旦提出了我们认为最好的模型，就可以在(迄今未见的)测试集上运行该模型以报告其性能。

尽管使用开发集可以避免过度拟合测试集，但是具有固定的训练集、开发集和测试集会产生另一个问题：为了保存大量数据进行训练，测试集(或开发集)可能不够大，不具有代表性。如果我们能够以某种方式将所有数据用于训练，而仍将所有数据用于测试，会不会更好呢？我们可以通过**交叉验证(cross-validation)**：我们随机选择一种训练集和测试集的数据切分来训练分类器，然后计算测试集上的错误率，最后重复(随机选择另一种不同的训练集和测试集)。我们做这个采样过程 10 次，并对这 10 次运行进行平均，以得到一个平均错误率。这被称为**十倍交叉验证(10-fold cross-validation)**。

交叉验证的唯一问题是，由于所有数据都用于测试，因此我们要求整个语料库是盲目的。我们无法检查任何数据来暗示可能的特征，并且通常可以看到正在发生的事情，因为我们会偷看测试集，而这种作弊行为会使我们高估系统的性能。但是，查看语料库以了解发生了什么对设计 NLP 系统很重要！该怎么办？因此，通常先创建一个固定的训练集和测试集，然后在训练集内进行十倍交叉验证，但是计算错误率是按照测试集的正常方式计算的，如图 4.7 所示：

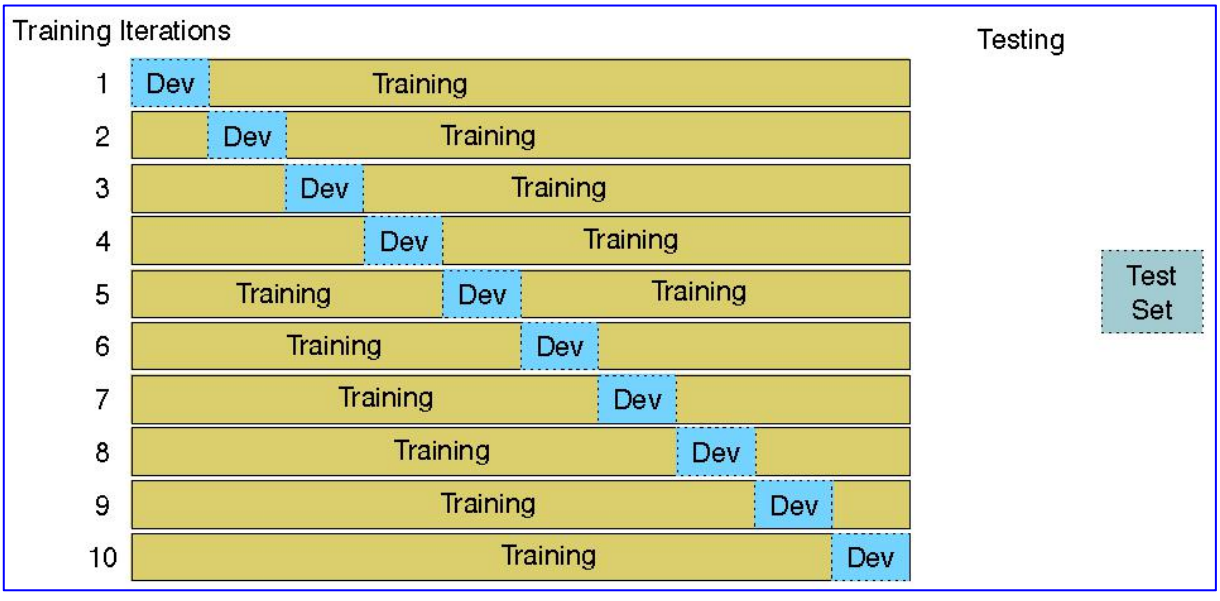


图 4-7：十倍交叉验证

4.9. 统计显著性检验

在构建系统时，我们经常需要比较两个系统的性能。我们如何才能知道我们刚刚建立的新系统是否比我们的旧系统更好?或者比文献中描述的其他系统更好? 这是统计假设检验的领域，在本节中，我们将介绍 NLP 分类器的统计显著性检验，特别是借鉴 Dror 等人(2020)和 Berg-Kirkpatrick 等人(2012)的工作。

假设我们正在比较分类器 A 和 B 在度量 M 上的性能，比如 F_1，或者准确度。也许我们想知道，在特定测试集 x 上，逻辑回归情感分类器 A(第 5 章)是否比朴素贝叶斯情感分类器 B 获得更高的 F_1 分数。我们称 M(A，x)为系统 A 在测试集 x 上获得的得分，而 δ (x)为 A 和 B 在 x 上的性能差异：

$$\delta(x) = M(A,x) - M(B,x) \tag{4.19}$$

我们想知道是否 δ (x)> 0，这意味着逻辑回归分类器在 X 上的 F_1 高于朴素贝叶斯分类器。δ (x)称为**效应量(effect size)**，较大的 δ 表示 A 似乎比 B 好得多；较小的 δ 意味着 A 似乎只稍好一点。

我们为什么不只检查 δ (x)是否为正？假设我们这样做了，我们发现 A 的 F_1 得分比 B 高 0.04。我们可以确定 A 更好吗？我们不可以！这是因为在这个特定的 x 上，A 可能刚好胜过 B。我们还需要更多东西：

我们想知道，如果我们检查另一个测试集 x'或在其他情况下，那么，A 优于 B 的优势是否还会继续存在。

在统计假设检验的范式中，我们通过形式化两个假设来检验这一点。

$$H_0 : \delta(x) \leq 0$$
$$H_1 : \delta(x) > 0 \tag{4.20}$$

假设 H_0，称为**零假设(null hypothesis)**，假设 δ (x)实际上为负或零，这意味着 A 并不比 B 好。我们想知道是否可以自信地排除该假设，而是支持 H_1，即 A 更好。为此，我们创建了一个覆盖所有测试集的随机变量 X。现在我们问，如果零假设 H_0 是正确的，那么在这些测试集中我们会遇到我们发现的 δ (x)值的可能性有多大。我们将这种似然度形式化为 p 值(p-value)：假设零假设 H_0 为真，则可以看到已知的 δ (x)或更大的概率：

$$P(\delta(X) \geq \delta(x) | H_0 \text{ is true}) \tag{4.21}$$

因此，在我们的示例中，该 p 值是，在假设 A 不比 B 好的情况下，我们看到 δ (x)的概率。如果 δ (x)很大(假设在 x 上，A 的 F_1 是非常可观的 0.9，而 B 的 F_1 只是可怕的 0.2)，则我们可能会感到惊讶，因为如果 H_0 实际上为真，则极不可能发生，所以 p 值会很低(如果 A 实际上不比 B 好，则不可能有这么大的 δ)。但是，如果 δ (x)非常小，即使 H_0 为真，而且 A 的确不比 B 好，这对我们来说不足为奇，因此 p 值会更高。

一个非常小的 p 值意味着我们观察到的差异在零假设下是非常不可能的，因此我们可以拒绝零假设。什么才算很小呢?通常使用.05 或.01 这样的值作为阈值。值 0.01 表示，如果 p 值(假设 H_0 为真时我们观察到的 δ 的概率)小于 0.01，那么我们将拒绝原假设，并假设 A 的确优于 B。我们说，如果我们看到的 δ 有低于阈值的概率，那么这个结果(例如，“A 比 B 好”)就是**统计显著(statistically significant)**，因此我们拒绝这个零假设。

我们如何计算 p 值所需的概率？ 在 NLP 中，我们通常不会使用您可能熟悉的简单参数测试(例如 t 检验或 ANOVA)。参数测试对测试统计量的分布(例如正态性)做出假设，而这些假设通常在我们的案例中不成立。因此，在 NLP 中，基于采样，我们通常使用非参数测试：我们人为地创建了许多版本的实验设置。例如，如果我们有很多不同的测试集 x'，则我们就可以测量所有 x'的所有 δ (x')。这给了我们一个概率分布。现在我们设置一个阈值(例如.01)，如果我们在此分布中看到 99%或更多的这些增量小于我们观察到的增量，即 p-value(x)小于.01(p 值是我们现在看见的 δ (x)的概率，它和我们过去看见的一样大)，那么我们可以拒绝原假设，并同意 δ (x)是一个足够令人惊讶差异，而且 A 确实比 B 更好。

有两种常见的非参数检验应用于 NLP：**近似随机(approximate randomization)**和**引导(bootstrap)**检验。我们将在下面描述引导，显示测试的配对版本，这也是 NLP 中最常见的版本。**配对(paired)**测试是我们比较两组对齐的观察值的方法：一组观察值可以与另一组观察值配对。当我们在同一测试集上比较两个系统的性能时，自然会发生这种情况。我们可以将系统 A 在单个观测 x_i 上的性能与系统 B 在同一 x_i 上的性能配对。

4.9.1. 配对引导测试

引导测试(bootstrap test)(Efron 和 Tibshirani，1993)可以适用于任何度量标准， 从精度、召回率或 F_1 到机器翻译中使用的 BLEU 指标。引导一词指的是从原始的较大样本中重复抽取大量较小的样本，并进行替换(称为引导样本)。引导测试的直觉是我们可以通过从观察到的测试集中重复采样来创建许多虚拟测试集。该方法仅假设样本可以代表总体。

考虑一个包含 10 个文档的测试集 x 的小型文本分类示例。图 4.8 的第一行显示了两个分类器(A 和 B)在该测试集中的结果，每个文档被标注为四种可能中的一种:(A 和 B 都对，A 和 B 都错，A 对 B 错，A 错 B 对);字母中的斜杠(/)意味着分类器得到了错误的答案。在第一个文档中，A 和 B 都得到了正确的类(AB)，而在第二个文档中，A 得到了正确的类，但 B 得到了错误的类。为了简单起见，我们假设我们的度量标准是准确度，A 的准确度是 0.70,B 的准确度是 0.50，所以 δ (x)是 0.20。

现在我们创建 b(可能是大数 10^5)个虚拟测试集 $x^{(i)}$，每个大小为 n = 10。图 4.8 给出了几个例子。为

了创建每个虚拟测试集 $x^{(i)}$，我们重复(n = 10 次)从行 x 中选择一个要替换的单元格。例如，要创建第一个虚拟测试集 $x^{(1)}$的第一个单元格，如果我们碰巧随机选择了 x 行的第二个单元格;我们将其值(A 和斜杠 B)复制到我们的新单元格中，并继续创建 $x^{(1)}$的第二个单元格，每次从原始的 x 中采样(随机选择)并进行替换。

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	A/B	AB	A/B	A/B	A/B	A/B	AB	A/B	A/B	.70	.50	.20
$x^{(1)}$	A/B	AB	A/B	A/B	A/B	A/B	A/B	AB	A/B	AB	.60	.60	.00
$x^{(2)}$	A/B	AB	A/B	A/B	A/B	AB	A/B	A/B	AB	AB	.60	.70	-.10
...													
$x^{(b)}$													

图 4-8：配对的引导测试

图注：从初始真实测试集 x 创建 b 个伪测试集 $x^{(i)}$的示例。通过替换 n = 10 次采样来创建每个伪测试集； 因此，单个样本就是一个单元格，一个带有黄金标签的文档，以及分类器 A 和 B 的正确或不正确的性能。当然，实际的测试集不会只有 10 个示例，并且 b 也必须很大。

现在我们有了 b 个测试集，提供了抽样分布，我们可以对 A 具有偶然优势的频率进行统计。有多种方法可以计算这种优势。在这里，我们遵循 Berg-Kirkpatrick 等人(2012)提出的版本。假设 H_0(A 并不比 B 好)，我们可以预期 $\delta(X)$为零(根据许多测试集估算出的)；更高的值会令人惊讶，因为 H_0 专门假设 A 不比 B 好。为了准确地测量我们观察到的 $\delta(x)$有多令人惊讶，我们将在其他情况下,在 b 个测试集上对出现 $\delta(x^{(i)})-\delta(x)\geq 0$ 的情况进行计数，据此来计算 p-value：

$$\text{p-value}(x)=\sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})-\delta(x)\geq 0\right)$$

但是，尽管在许多测试集上$\delta(X)$的期望值=0 通常是正确的(再次假设 A 并不优于 B)，但对于我们创建的引导测试集却并非如此。那是因为我们没有从均值为 0 的分布中抽取这些样本；我们碰巧是从原始测试集 x 创建它们的，而恰好是 A 的偏差值(0.20)。因此，为了准确地测量我们观察到的 $\delta(x)$有多令人惊讶，我们在 b 个测试集上对出现 $\delta(x^{(i)})-\delta(x)\geq\delta(x)$的情况进行计数，据此来计算 p-value：

$$\begin{aligned}\text{p-value}(x) &= \sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})-\delta(x)\geq\delta(x)\right)\\ &= \sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})\geq 2\delta(x)\right)\end{aligned}\tag{4.22}$$

因此，例如，如果我们有 10,000 个测试集 $x^{(i)}$和阈值 0.01，并且仅在 47 个测试集中，我们发现 $\delta(x^{(i)})\geq 2\delta(x)$，则得到的 p 值=0.0047 小于 0.01，表明 $\delta(x)$确实足够令人惊讶，并且我们可以拒绝零假设，并得出结论 A 优于 B。

完整的引导算法如图4.9所示。它给出了一个测试集x，样本数b，并计算了在b个引导测试集上 $\delta(x^{(i)})>2\delta(x)$的百分比。这个百分比作为一个单边的经验 p 值。

function BOOTSTRAP(test set x, num of samples b) **returns** p-$value(x)$

Calculate $\delta(x)$ # how much better does algorithm A do than B on x
$s = 0$
for i = 1 **to** b **do**
  for j = 1 **to** n **do** # Draw a bootstrap sample $x^{(i)}$ of size n
    Select a member of x at random and add it to $x^{(i)}$
  Calculate $\delta(x^{(i)})$ # how much better does algorithm A do than B on $x^{(i)}$
  $s \leftarrow s + 1$ **if** $\delta(x^{(i)}) > 2\delta(x)$
p-value$(x) \approx \frac{s}{b}$ # on what % of the b samples did algorithm A beat expectations?
return p-value(x) # if very few did, our observed δ is probably not accidental

图 4-9：配对引导算法

图注：Berg-Kirkpatrick 等人(2012)之后的配对引导算法的一个版本。

4.10. 避免分类中的危害

重要的是要避免分类器可能造成的损害，这些损害既存在于朴素的贝叶斯分类器中，也存在于我们将在后面的章节中介绍的其他分类算法中。一类伤害是**代表性伤害(representational harms)**(Crawford 2017, Blodgett 等人 2020)，该类由贬低社会群体的系统造成的伤害，例如通过对他们的负面刻板印象的永恒化。例如，Kiritchenko)和 Mohammad(2018)在相同的句子对上检查了 200 个情感分析系统的性能，除了包含一个常见的非裔美国人的名字(如 Shaniqua)或一个常见的欧裔美国人的名字(如 Stephanie)之外，这些名字都是相同的，这些名字取自第 6 章讨论的 Caliskan 等人(2017)的研究。他们发现，大多数系统会对带有非裔美国人名字的句子赋予较低的情感和更多的负面情绪，这反映并延续了将非裔美国人与负面情绪联系在一起的刻板印象(Popp 等人，2003)。

在其他任务中，分类器可能同时导致代表性损害和其他损害，如审查制度。例如，**毒性检测(toxicity detection)**的重要文本分类任务是检测仇恨言论、虐待、骚扰或其他种类的毒性语言的任务。尽管此类分类器的目标是帮助减少社会危害，但毒性分类器本身可能会造成危害。例如，研究人员表明，一些广泛使用的毒性分类器错误地将其标记为无毒的有毒句子，但仅提及少数群体身份，例如女性(Park 等人，2018)、盲人(Hutchinson 等人，2020)或同性恋者(Dixon 等人，2018)，或者干脆使用非裔美国人白话英语等变种的语言特征(Sap 等人 2019，Davidson 等人 2019)。如果这些假阳性错误被毒性检测系统在没有人为监督的情况下使用，则可能导致这些人群或有关人群对话语的审查。

这些模型问题可能是由训练数据中的偏差或其他问题引起的； 通常，机器学习系统会复制甚至放大其训练数据中的偏差。但是，这些问题也可能是由下列因素导致的：标签(例如，由人类标签的偏差)、所使用的资源(例如词汇，或模型组件诸如预训练的嵌入)、甚至是模型体系结构(例如模型被训练成优化的)。尽管减轻这些偏差(例如，通过仔细考虑训练数据源)是一个重要的研究领域，但我们目前尚无一般解决方案。因此，在引入任何 NLP 模型时，重要的是要研究这些因素并弄清楚它们。一种实现方法是为模型的每个版本发布**模型卡(model card)**(Mitchell 等人，2019)，该卡记录了机器学习模型，其中包含以下信息：

- 训练算法和参数
- 训练数据源、动机和预处理
- 评估数据源、动机和预处理
- 预期的用途和用户
- 跨不同人员背景或其他群体和环境状况的模型性能

4.11. 总结

本章介绍了**朴素贝叶斯**（Naive Bayes）分类模型，并将其应用于**情感分析**的**文本分类**任务。

•许多语言处理任务可以看作是**分类**任务。

•文本分类，将整个文本从一个有限集合中分配一个类，包括**情感分析**、**垃圾邮件检测**、语言识别和作者归属等任务。

•情感分析将文本分类为反映作者对某些对象表达的积极或消极倾向(**情感**)。

•朴素贝叶斯是一种**生成**模型，它制订了(make)单词假设（位置无关）和条件独立假设（给定类别的单词在条件上彼此独立）。

•具有二值化特征的朴素贝叶斯似乎对许多文本分类任务更有效。

•基于**精度**和**召回率**对分类器进行评估。

•使用不同的训练集、开发集和测试集来训练分类器，包括在训练集中使用**交叉验证**。

•应该使用统计显著性检验来确定我们是否可以确信一个分类器版本优于另一个版本。

•分类器的设计者应仔细考虑模型可能造成的危害(包括训练数据和其他组件)，并在**模型卡**中报告模型特色。

4.12. 文献和历史说明

多项式朴素贝叶斯文本分类是由 RAND 公司的 Maron(1961)提出的，用于为期刊摘要分配主题类别。他的模型引入了本文提出的现代形式的大部分特征，将分类任务近似为其中的一个分类任务，并实现了 add-δ 平滑和基于信息的特征选择。

朴素贝叶斯的条件独立假设和贝叶斯文本分析的思想似乎已经多次出现。在 Maron 的论文发表的同一年，Minsky(1961)提出了一个用于视觉和其他人工智能问题的朴素贝叶斯分类器，Mosteller 和 Wallace(1963)也将贝叶斯技术应用于作者归属的文本分类任务中。人们早就知道，Alexander Hamilton, John Jay, 和 James Madison 在 1787-1788 年撰写了匿名发表的联邦党人论文，说服纽约批准美国宪法。然而，尽管这 85 篇论文中有一些显然是出自某个作者之手，但有 12 篇的作者在 Hamilton 和 Madison 之间存在争议。Mosteller 和 Wallace(1963)训练了 Hamilton 写作的贝叶斯概率模型和 Madison 写作的另一个模型，然后计算了每一篇有争议的文章的最大可能作者。Heckerman 等人(1998)首次将朴素贝叶斯应用于垃圾邮件检测。

Metsis 等人(2006)，Pang 等人(2002)，以及 Wang 和 Manning(2012)表明，使用多项式朴素贝叶斯的布尔属性比使用完整计数效果更好。二元多项式朴素贝叶斯有时会与另一种也使用二进制表示的朴素贝叶斯相混淆:多元伯努利朴素贝叶斯。而伯努利变量则估计 P(w|c)为包含术语的文档的部分，并包括一个术语是否不在文档中的概率。McCallum 和 Nigam(1998)以及 Wang 和 Manning(2012)表明，在情感或其他文本任务中，朴素贝叶斯的多元伯努利变异并不像多项算法那样有效。

有各种各样的来源涵盖了多种文本分类任务。情感分析见 Pang 和 Lee (2008)，Liu 和 Zhang(2012)。Stamatatos(2009)调查了作者归属算法。关于语言识别见 Jauhiainen 等人(2018);Jaech 等人(2016)是一个重要的早期神经系统。新闻通讯的索引任务经常被用作文本分类算法的测试用例，基于路透社-21578 新闻通讯的文章集合。参见 Manning 等人(2008)和 Aggarwal 和 Zhai(2012)关于文本分类的论述;一般来说，机器学习教科书中涵盖了分类(Hastie 等人 2001, Witten 和 Frank 2005, Bishop 2006, Murphy 2012)。

计算统计显著性的非参数方法在 MUC 竞赛中首次在 NLP 中使用(Chinchor 等人，1993)，甚至更早在语音识别中使用(Gillick 和 Cox 1989, Bisani 和 Ney 2004)。我们对自举的描述借鉴了 Berg-Kirkpatrick 等人(2012)的描述。最近的工作主要集中在多个测试集和多个度量(Søgaard 等人 2014，Dror 等人 2017)。

特征选择是一种去除不太可能很好概括的特征的方法。特征通常是根据它们对分类决策的信息量来排序的。一个非常常见的度量标准，信息增益，告诉我们这个词的出现给了我们多少位信息来猜测这个类。其他特征选择指标包括 χ^2、逐点互信息和基尼(GINI)指数;参见 Yang 和 Pedersen(1997)的比较，以及 Guyon 和 Elisseeff(2003)对特征选择的介绍。

4.13. 练习

4.1 Assume the following likelihoods for each word being part of a positive or negative movie review, and equal prior probabilities for each class.

	pos	neg
I	0.09	0.16
always	0.07	0.06
like	0.29	0.06
foreign	0.04	0.15
films	0.08	0.11

What class will Naive bayes assign to the sentence “I always like foreign films.”?

4.2 Given the following short movie reviews, each labeled with a genre, either comedy or action:

1. fun, couple, love, love **comedy**
2. fast, furious, shoot **action**
3. couple, fly, fast, fun, fun **comedy**
4. furious, shoot, shoot, fun **action**
5. fly, fast, shoot, love **action**

and a new document D:

fast, couple, shoot, fly

compute the most likely class for D. Assume a naive Bayes classifier and use add-1 smoothing for the likelihoods.

4.3 Train two models, multinomial naive Bayes and binarized naive Bayes, both with add-1 smoothing, on the following document counts for key sentiment words, with positive or negative class assigned as noted.

doc	“good”	“poor”	“great”	(class)
d1.	3	0	3	pos
d2.	0	1	2	pos
d3.	1	3	0	neg
d4.	1	5	2	neg
d5.	0	2	0	neg

Use both naive Bayes models to assign a class (pos or neg) to this sentence:

A good, good plot and great characters, but poor acting.

Recall from page 60 that with naive Bayes text classification, we simply ignore (throw out) any word that never occurred in the training document. (We don’t throw out words that appear in some classes but not others; that’s what add-one smoothing is for.) Do the two models agree or disagree?

5. 逻辑回归

“您怎么知道这些美丽的秋海棠并不同样重要呢？”

赫尔克里·波洛，在阿加莎·克里斯蒂的《斯泰尔斯的神秘事件》

侦探小说充斥着线索，就像文本充斥着单词一样。然而，对于可怜的读者来说，要知道如何权衡作者的线索来完成关键的分类任务：**决定谁是真凶(deciding whodunnit)**，这是很有挑战性的。

在本章中，我们介绍一种非常适合发现特征或线索与某些特定结果之间的联系的算法：**逻辑回归(logistic regression)**。实际上，逻辑回归是社会科学和自然科学中最重要的分析工具之一。在自然语言处理中，逻辑回归是用于分类的基准监督机器学习算法，并且与神经网络有着非常密切的关系。正如我们将在第 7 章中看到的那样，神经网络可以看作是一系列相互堆叠的逻辑回归分类器。因此，本文介绍的分类和机器学习技术将在整本书中发挥重要作用。

逻辑回归可用于将观察结果分为两类之一(例如“积极情感”和“消极情感”)，也可以分为许多类别之一。由于两类情况的数学比较简单，因此我们将在接下来的几节中首先介绍这种逻辑回归的特殊情况，然后在 5.6 节中简要总结多元逻辑回归在两个以上类中的使用。在接下来的几节中，我们将介绍逻辑回归的数学原理。但是，让我们从一些高级问题开始。





生成性(generative)和**区分性(discriminative)**分类器：朴素贝叶斯和逻辑回归之间的最重要区别是，逻辑回归是一种区分性分类器，而朴素贝叶斯是生成性分类器。

对于如何构建机器学习模型，这是两个非常不同的框架。考虑一个视觉隐喻：想象一下我们正在尝试区分狗图像和猫图像。生成模型的目标是了解狗的外观和猫的外观。您可能会从字面上要求这样的模型“生成”(绘)一只狗。给定测试图像后，系统会询问最适合该图像(对它不那么惊讶)的是猫模型还是狗模型，然后选择它作为标签。

相比之下，区分性模型仅仅是试着去学习区分这些类(也许并没有对他们有太多的了解)。所以也许训练数据中所有的狗都戴着项圈而猫没有。如果有一个特征能很好地将这些类分开，那么模型就满足了。如果你问模型，它对猫了解多少，它只能说猫不戴项圈。

更正式地说，回想一下，朴素贝叶斯不是通过直接计算 P(c | d)而是通过计算似然和先验来为文档 d 分配类 c：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{5.1}$$

像朴素贝叶斯这样的生成模型使用了这个似然项，它表示：如果我们知道文档属于 c 类(if we knew it was of class c)，则如何去生成文档的特征。

相比之下，在这个文本分类场景中，一个区分性模型尝试直接计算 P(c|d)。也许它将学会给文档特征赋予很高的权重，这些特征可以直接提高它在这些可能的类中作区分的能力，即便它不能生成其中一个类的一个示例。

概率机器学习分类器的组成部分：与朴素贝叶斯一样，逻辑回归是一种利用监督机器学习的概率分类器。机器学习分类器需要 m 个输入/输出对($x^{(i)}$，$y^{(i)}$)的训练语料。(对于情感分类，我们将使用括号中的上标来指代训练集中的单个实例，每个实例可能是一个要分类的单独文档)。用于分类的机器学习系统有四个组成部分:

1. 输入的一种特征表示。对于每个输入观测 $x^{(i)}$，这将是一个特征向量$[x_1, x_2, \ldots, x_n]$。我们通常将输入

$x^{(i)}$的特征 i 称为 $x_i^{(i)}$，有时简化为 x_i，但我们也会看到符号 f_i, $f_i(x)$，或者，对于多类分类，$f_i(c, x)$。

2. 一个分类函数，它通过 p(y | x)计算被估类的$\hat{y}$。下一节，我们将介绍用于分类的 sigmoid 和 softmax。
3. 一种用于学习的目标函数，通常包括最小化训练示例中的误差。我们将引入交叉熵损失函数。
4. 一种优化目标函数的算法。我们介绍了随机梯度下降算法。

Logistic 回归分为两个阶段:

训练:我们使用随机梯度下降和交叉熵损失训练系统(特别是权值 w 和 b)。

测试:给定一个测试示例 x，我们计算 p(y|x)并返回更高的概率标签 y = 1 或 y = 0。

5.1. 分类: sigmoid

二元逻辑回归的目标是训练一个分类器，使其能够对新的输入观察的类做出二元决策。在这里我们介绍 sigmoid 分类器，它将帮助我们做出这个决策。

考虑单个输入观测值 x，我们将用特征向量$[x_1, x_2, \ldots, x_n]$表示它(我们将在下一部分中显示示例特征)。分类器输出 y 可以为 1(表示观察值是该类的成员)或 0(表示观察值不是该类的成员)。我们想知道该观察是该类成员的概率 P(y=1|x)。因此，也许决策是“积极情感”和“消极情感”，这些特征表示文档中单词的数量，P(y=1|x)是文档具有积极情感的概率，P(y=0|x)是文档消极情感的可能性。

逻辑回归通过从训练集中学习权向量和偏差项来解决此任务。每个权重 w_i 是实数，并且与输入特征 x_i 之一相关联。权重 w_i 表示输入特征对分类决策的重要性，可以为正(提供证据表明正在被分类的实例属于肯定类)或负(提供证据表明正在被分类的实例属于否定类)。因此，我们可能期望在情感任务中，“awesome”这个词具有很高的正权重，而“absmal”则具有非常负的权重。**偏差项(bais term)**,也称为**截距(intercept)**,是添加到加权输入中的另一个实数。

为了在测试实例上做出决策(在我们学习了训练的权重之后)，分类器首先将每个 x_i 乘以权重 w_i，对加权特征求和，然后加上偏差项 b。所得的单个数字 z 表示该类别证据的加权总和。

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{5.2}$$

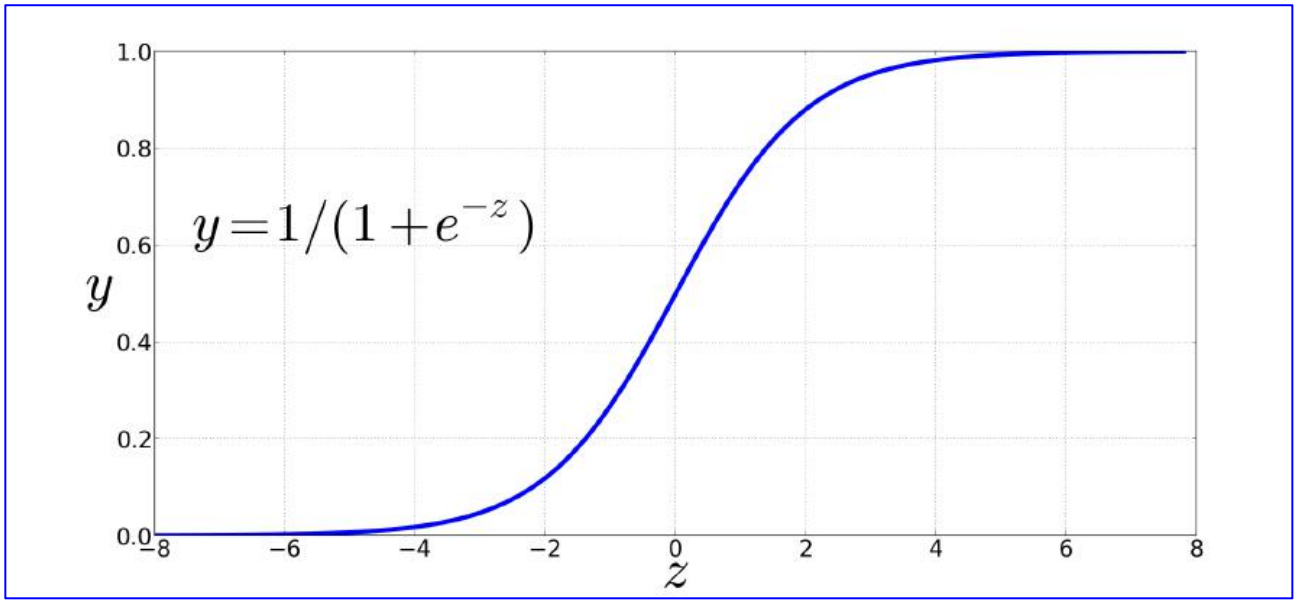


图 5-1：sigmoid 函数

图注：y 取一个实数，并将其映射到范围[0, 1]。它在 0 附近几乎是线性的，但是离群值被压向 0 或 1。

在本书的其余部分，我们将使用线性代数中的**点积(dot product)**符号来表示这样的和。两个向量 a 和 b 的点积，写成 a·b，是每个向量对应元素的乘积的和。因此，以下是与公式 5.2 等价的形式:

$$z = w \cdot x + b \tag{5.3}$$

但请注意，公式 5.3 中没有任何东西强制 z 为合法概率，即介于 0 和 1 之间。事实上，由于权值是实数值，输出甚至可能是负的;z 的取值范围是-∞到∞。

为了产生概率，我们将通过 **sigmoid** 函数σ(z)传递 z。S 形函数(sigmoid)(之所以称呼它，是因为它看起来像一个 s)，也称为**逻辑函数(logistic function)**，并赋予逻辑回归这个名称。如图 5.1 所示,S 型具有以下公式：

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)} \tag{5.4}$$

(在本书的其余部分中，我们将使用符号 exp(x)表示 e^x。)S 形有很多优点； 它采用一个实数值并将其映射到[0,1]范围内，这正是我们想要的概率。由于它在 0 附近几乎是线性的，但在两端逐渐变平，因此倾向于将离群值压缩到 0 或 1。而且它是可微的，正如我们在 5.8 节中将看到的那样，它很容易学习。

我们差不多了。如果将 Sigmoid 应用于加权特征的总和，我们将得到一个介于 0 和 1 之间的数字。要使其成为概率，我们只需要确保 p(y = 1)和 p(y = 0)的总和为 1。我们可以这样进行操作：

$$\begin{aligned} P(y=1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1+\exp(-(w \cdot x + b))} \\ P(y=0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1+\exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1+\exp(-(w \cdot x + b))} \end{aligned} \tag{5.5}$$

S 型函数有这样的性质:

$$1 - \sigma(x) = \sigma(-x) \tag{5.6}$$

所以我们也可以把 P(y = 0)表示为σ(- (w·x+b))。

现在我们有了一个算法，给定一个实例 x 计算概率 P(y = 1 | x)。我们如何做出决策？对于测试实例 x，如果概率 P(y = 1 | x)大于 0.5，则说“是”，否则说“否”。我们将 0.5 称为**决策边界(decision boundary)**：

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

5.1.1. 例子：情感分类

让我们举个例子。假设我们正在对电影评论文本进行二进制情感分类，并且我们想知道是否将情感类别+或-分配给评论文档 doc。我们将通过下表中显示的输入的 6 个特征 x_1 ... x_6 来表示每个输入观察值；图 5.2 显示了微型测试文档样本中的特征。

现在，假设我们已经为每个特征学习了一个实数值权重，并且与这 6 个特征相对应的 6 个权重为[2.5，5.0 ，1.2，0.5，2.0，0.7]，而 b = 0.1。(我们将在下一节中讨论如何学习权重。)例如，权重 w_1 表示积极词汇(great, nice, enjoyable 等)的数量的特征对一个积极情绪决策的重要性，而 w_2 则告诉我们消极词汇的重要性。注意，w_1 = 2.5 是积极的，而 w_2 =-5.0 意味着消极词汇与积极情绪决策呈负相关，其重要性大约

是积极词汇的两倍。

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

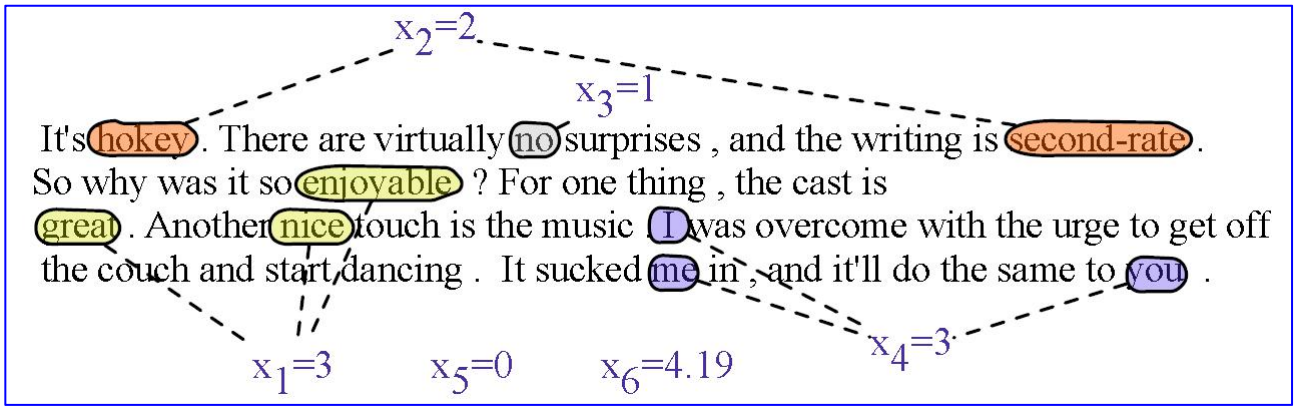


图 5-2：微型测试文档样本向量 x 中的提取特征

图注：微型测试文档的一个样本(显示了向量 x 中的提取特征)

给定这 6 个特征和输入评论 x, P(+|x)和 P(−|x)可以通过公式 5.5 计算:

$$\begin{aligned} p(+|x) = P(Y=1|x) &= \sigma(w\cdot x+b) \\ &= \sigma([2.5,-5.0,-1.2,0.5,2.0,0.7]\cdot[3,2,1,3,0,4.19]+0.1) \\ &= \sigma(.833) \\ &= 0.70 \\ p(-|x) = P(Y=0|x) &= 1-\sigma(w\cdot x+b) \\ &= 0.30 \end{aligned} \tag{5.7}$$

逻辑回归通常应用于各种 NLP 任务，输入的任何属性都可以是一个特征。考虑**句点消歧(period disambiguation)**的任务:通过将每个句点分为 EOS(句尾)和非 EOS 两类之一，确定句点是句子的结尾还是单词的一部分。我们可以使用下面的 x_1 这样的特征来表示：当前单词是小写的，其类为 EOS(可能有一个正的权重)；或者，当前单词在我们的缩写字典中("Prof")，其类为 EOS(可能有一个负的权重)。特征还可以表示相当复杂的属性组合。例如，大写单词后面的句号可能是 EOS，但如果单词本身是 St.，并且前面的单词大写，那么句号可能是单词 street 的缩写的一部分。

$$x_1 = \begin{cases} 1 & \text{if "}Case(w_i) = \text{Lower"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if "}w_i \in \text{AcronymDict"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if "}w_i = \text{St.} \ \& \ Case(w_{i-1}) = \text{Cap"} \\ 0 & \text{otherwise} \end{cases}$$

设计特征：特征的设计通常是通过观察语言直觉和该领域的语言文献来检查训练集。对系统早期版本

的训练集或开发集进行仔细的误差分析，通常可以洞悉特征。

对于某些任务，构建由更多原始特征组合而成的复杂特征特别有用。我们在上面看到了用于句点消歧的特征，如果前一个单词是大写的，则单词 St.上的句点不太可能是句子的结尾。对于逻辑回归和朴素贝叶斯，必须手动设计这些组合特征或**特征交互(feature interactions)**。

对于许多任务(特别是当特征值可以引用特定的单词时)，我们需要大量的特征。这些通常是通过**特征模板(feature templates)**(特征的抽象规范)自动创建的。例如，用于句点消歧的 bigram 模板可能为出现在训练集中句点之前的每对单词创建一个特征。因此，特征空间是稀疏的，因为我们仅需在训练集中该 n-gram 存在于该位置的情况下创建特征。通常根据字符串的描述将特征创建为哈希(hash)。用户对诸如“bigram(American breakfast)”特征的描述为被哈希为唯一的整数 i，该整数成为特征数 f_i。

为了避免大量人为的特征设计工作，最近的 NLP 研究集中于**表示学习(representation learning)**:从输入中以无监督的方式自动学习特征的方法。我们将在第 6 章和第 7 章中介绍表示学习的方法。

选择分类器 与朴素贝叶斯相比，逻辑回归具有许多优势。朴素贝叶斯有过分强的条件独立性假设。考虑两个密切相关的特征； 实际上，假设我们只是将相同的特征 f_1 加了两次。朴素贝叶斯会将 f_1 的两个副本视为分开的，将它们相乘，从而高估了证据。相比之下，逻辑回归对相关特征更为健壮。如果两个特征 f_1 和 f_2 完全相关，则回归将简单地将部分权重分配给 w_1，将部分权重分配给 w_2。因此，当存在许多相关特征时，逻辑回归将比朴素贝叶斯分配更准确的概率。因此，逻辑回归通常在较大的文档或数据集上效果更好，这是常见的默认设置。

尽管概率不太准确，朴素贝叶斯仍然经常做出正确的分类决策。此外，朴素贝叶斯可以在非常小的数据集(Ng 和 Jordan, 2002)或简短的文档(Wang 和 Manning, 2012)上工作得非常好(有时甚至比逻辑回归更好)。此外，朴素贝叶斯很容易实现和非常快的训练(没有优化步骤)。所以在某些情况下，这仍然是一个合理的方法。

5.2. 在逻辑回归中学习

如何学习模型的参数(权重 w 和偏差 b)？ 逻辑回归是监督分类的一个实例，其中我们知道每个观测值 x 的正确标签 y(0 或 1)。系统通过公式 5.5 产生的结果是$\hat{y}$，即系统对真实 y 的估计。我们想学参数(即 w 和 b)，使每个训练观察的$\hat{y}$尽可能接近真实的 y。

这需要在本章的简介中提到的两个组件。

第一个度量标准是当前标签($\hat{y}$)与真实黄金标签 y 的接近程度。我们通常考虑的不是度量相似度，而是与之相反的情况：系统输出与黄金输出之间的距离，我们将此距离称为损失函数或成本函数。在下一节中，我们将介绍通常用于逻辑回归和神经网络的损失函数，即交叉熵损失。

第二个是用于迭代更新权重的优化算法，以便最小化此损失函数。对此的标准算法是梯度下降。在以下部分中，我们将介绍随机梯度下降算法。

5.3. 交叉熵损失函数

对于观察 x，我们需要一个损失函数来表示分类器输出($\hat{y} = \sigma(w \cdot x + b)$)与正确的输出(y，即 0 或 1)有多接近。我们将其称为：

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y \tag{5.8}$$

我们通过损失函数来做到这一点，损失函数更倾向于训练示例的正确类标签。这称为条件最大似然估计：我们选择参数 w，b，该参数在给定观察值 x 的情况下最大化训练数据中真实 y 标签的对数概率。最终的损失函数为负对数似然损失，通常称为**交叉熵损失(cross-entropy loss)**。

让我们导出这个损失函数，将其应用于单个观测值 x。我们想学习权重，以最大化正确标签 p(y | x)的概率。由于只有两个离散的结果(1 或 0)，所以这是一个伯努利分布，对于我们的分类器为一个观察而产生的概率 p(y | x)(请注意，如果 y = 1，则公式 5.9 简化为$\hat{y}$；如果 y = 0，则公式 5.9 简化为 1-$\hat{y}$)，我们可以如下表达:

$$p(y|x) = \hat{y}^y (1-\hat{y})^{1-y} \tag{5.9}$$

现在我们对两边取对数。这在数学上是很方便的，而且不会对我们造成伤害;使概率最大化的值也将使概率的对数最大化:

$$\begin{aligned}\log p(y|x) &= \log\left[\hat{y}^y (1-\hat{y})^{1-y}\right] \\ &= y\log\hat{y} + (1-y)\log(1-\hat{y})\end{aligned} \tag{5.10}$$

公式 5.10 描述了应该最大化的对数似然。为了将其转化为损失函数(我们需要将其最小化)，我们只需将公式 5.10 上的符号翻转即可。结果是交叉熵损失 L_{CE}：

$$L_{CE}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})] \tag{5.11}$$

最后，我们可以插入ŷ =σ(w·x + b)的定义：

$$L_{CE}(\hat{y},y) = -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \tag{5.12}$$

让我们看看这个损失函数是否可以正确完成图 5.2 中的示例。如果模型的估算值接近正确，我们希望损失会减少；如果模型混乱，则损失会更大。因此，首先让我们假设图 5.2 中情感示例的正确黄金标签为正，即 y =1。在这种情况下，我们的模型运行良好，从公式 5.7 可以看出，这个例子的积极(0.69)概率确实高于消极(0.31)概率。如果我们将σ(w • x+b) = 0.69 和 y = 1 代入公式 5.12 中，方程右边就会去掉，导致如下损失(我们将使用对数表示未指定底数时的自然对数):

$$\begin{aligned}L_{CE}(\hat{y},y) &= -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -[\log\sigma(w\cdot x+b)] \\ &= -\log(.69) \\ &= .37\end{aligned}$$

相比之下，让我们假设图 5.2 中的例子实际上是消极的，即 y = 0(也许评论者接着说:“但最重要的是，这部电影很糟糕!我求你不要看它!”)。在这种情况下，我们的模型是混乱的，我们希望损失更大。现在，如果我们将公式 5.7 中的 y = 0 和 1-σ(w·x+b) =0.31 代入公式 5.12，方程左边就会得到:

$$\begin{aligned}L_{CE}(\hat{y},y) &= -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -[\log(1-\sigma(w\cdot x+b))] \\ &= -\log(.31) \\ &= 1.17\end{aligned}$$

当然，第一个分类器的损失(0.37)小于第二个分类器的损失(1.17)。

为什么我们想要最大程度地降低这种负对数概率？完美的分类器会将概率 1 分配给正确的结果(y = 1 或 y = 0)，将概率 0 分配给不正确的结果。这意味着ŷ越高(它越接近 1)，则分类器越好；ŷ越低(越接近 0)，分类器越差。此概率的负对数是一种方便的损失度量，因为它从 0(负对数 1，无损失)到无穷大(负对数 0，无穷损失)。该损失函数还确保：当正确答案的概率最大化时，错误答案的概率最小化。由于两者之和为 1，因此正确答案的概率的任何增加都会以错误答案为代价。之所以称为交叉熵损失，是因为公式 5.10 也是一个交叉熵的公式，这个交叉熵横跨在真实概率分布 y 和估计分布ŷ之间。

现在我们知道了要最小化的内容； 在下一部分中，我们将介绍如何找到最小值。

5.4. 梯度下降

我们使用梯度下降的目标是找到最优的权值:最小化我们为模型定义的损失函数。在下面的公式 5.13

中，我们将明确表示这样一个事实，即损失函数 L 是由权值参数化的，在机器学习中我们通常将其称为θ(在逻辑回归θ= w,b 的情况下)。所以我们的目标是找到一组使损失函数最小的权值，在所有例子上取平均值:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)};\theta), y^{(i)}) \tag{5.13}$$

我们如何找到这个(或任何)损失函数的最小值？ 梯度下降法是通过找出函数的斜率在哪个方向上(在参数θ的空间内)最陡峭地上升，并在相反方向上移动来找到函数的最小值。直觉是，如果您在峡谷中徒步旅行，并试图最快速地下降到底部的河流，则可能会环顾四周 360 度，找到地面最陡峭的方向，然后沿该方向下坡 。

对于逻辑回归，此损失函数通常是**凸的(convex)**。凸函数只有一个最小值。没有局部最小值可以陷入，因此保证从任何一点开始的梯度下降都可以找到最小值。(相比之下，多层神经网络的损失是非凸的，梯度下降可能会卡在用于神经网络训练的局部最小值中，而找不到全局最优值。)

尽管这个算法(和梯度的概念)是为方向矢量设计的，但让我们首先考虑一下可视化的情况，其中我们的系统参数仅为单个标量 w，如图 5.3 所示。

给定 w 在某个值 w^1 处的随机初始化，并假设损失函数 L 恰好具有图 5.3 的形状，我们需要一种算法来告诉我们在下一次迭代时是否应向左移动(使 w^2 小于 w^1) 或向右(使 w^2 大于 w^1)以便达到最小值。

梯度下降算法通过找到损失函数在当前点的梯度并向相反的方向移动来回答这个问题。多变量函数的梯度是指向函数中增量最大方向的向量。梯度是对斜率的多变量概括，所以对于像图 5.3 这样的单变量函数，我们可以非正式地把梯度看作斜率。图 5.3 中的虚线显示了这个假设的损失函数在 $w = w^1$ 点的斜率。可以看到这条虚线的斜率是负的。因此，为了找到最小值，梯度下降告诉我们以相反的方向:将 w 移动到正的方向。

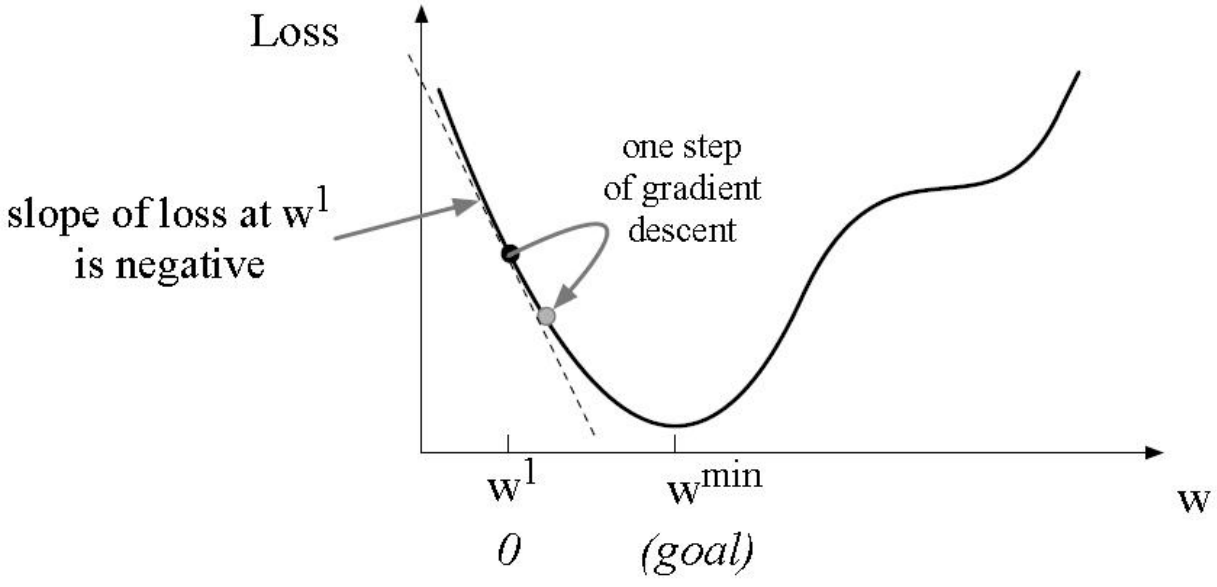


图 5-3：梯度下降算法的可视化

图注：通过从函数的斜率反向移动 w 来迭代找到此损失函数的最小值的第一步。由于斜率为负，因此我们需要将 w 沿正方向向右移动。这里的上标用于学习步骤，因此 w^1 表示 w 的初始值(为 0)，第二步为 w^2，依此类推。

梯度下降的移动量的大小是由**学习率(learning rate)**η加权的斜率$\frac{d}{dw}$f(x;w)的值。更高(更快)的学习速度意味着我们应该在每一步上增加更多。我们在参数中所做的更改是学习率乘以梯度(或斜率，在我们的单变量示例中)：

$$w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w) \tag{5.14}$$

现在让我们将直觉地从一个标量变量 w 的函数扩展到许多变量，因为我们不只是想向左或向右移动，我们想知道在 N 维空间中(构成θ的 N 个参数) ，我们应该如何移动。梯度就是这样一个向量，它表示沿这 N 个维度中的每个维度的最大斜率的方向分量。如果我们仅想象两个权重维度(例如，对于一个权重 w 和一个偏置 b)，则梯度可能是具有两个正交分量的向量，每个分量都告诉我们 w 维度和 b 上的地面坡度尺寸。图 5.4 显示了在红点处获取的二维梯度向量值的可视化。

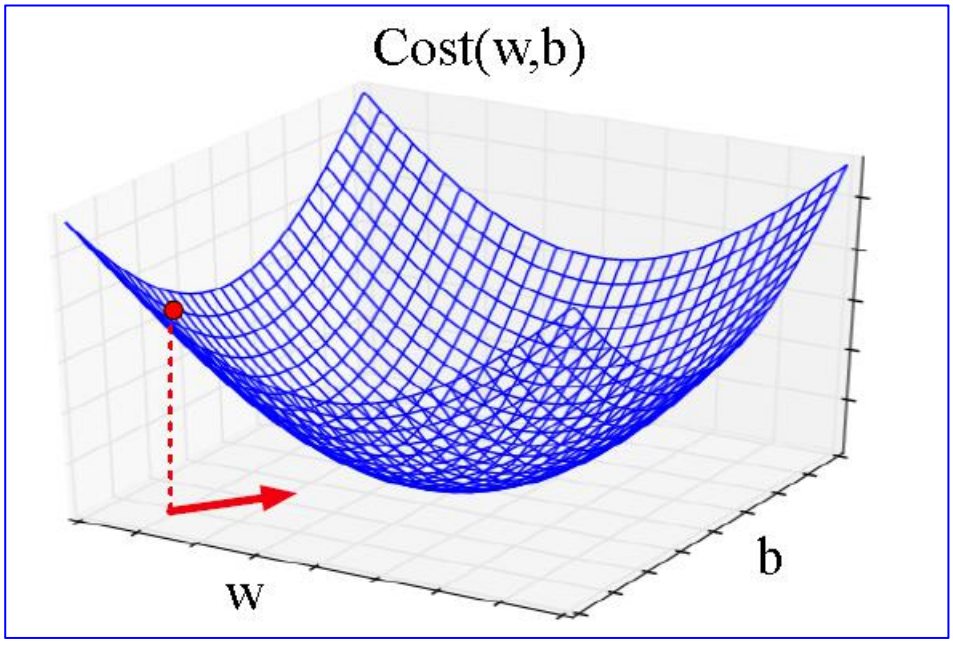


图 5-4：梯度矢量的可视化

图注：二维 w 和 b 上红色点处的梯度矢量的可视化，在 x-y 平面上以红色箭头显示了梯度。

在实际的逻辑回归中，参数向量 w 比 1 或 2 长得多，因为输入特征向量 x 可能很长，并且每个 x_i 需要权重 w_i。对于 w 中的每个维度/变量 w_i(加上偏差 b)，梯度将具有一个分量，该分量告诉我们相对于该变量的斜率。本质上，我们在问：“该变量 w_i 的微小变化会对总损失函数 L 产生多大影响？”

在每个维度 w_i 中，我们将斜率表示为损失函数的偏导数$\partial l/\partial w_i$。然后将梯度定义为这些偏导的向量。我们将 y 表示为 f(x;θ)，以使对θ的依赖性更加明显：

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix} \tag{5.15}$$

因此，θ是基于梯度而更新的，其最终方程为：

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta),y) \tag{5.16}$$

5.4.1. 逻辑回归的梯度

为了更新θ，我们需要定义梯度∇L(f(x;θ)，y)。回想一下，对于逻辑回归，交叉熵损失函数为：

$$L_{CE}(\hat{y},y) = -[y \log \sigma(w \cdot x + b) + (1-y) \log (1 - \sigma(w \cdot x + b))] \tag{5.17}$$

事实证明，该函数对一个观测向量 x 的导数为公式 5.18(对此公式的推导，读者可以参见 5.8 节)：

$$\frac{\partial L_{CE}(\hat{y},y)}{\partial w_j} = [\sigma(w\cdot x+b)-y]x_j \tag{5.18}$$

注意，在公式 5.18 中，相对于单个权重 w_j 的梯度，表达一个非常直观的值：真实的 y 与该观测值估算的$\hat{y}=\sigma(w\cdot x + b)$之差乘以相应的输入值 x_j。

5.4.2. 随机梯度下降算法

随机梯度下降是一种在线算法，通过在每个训练示例之后计算其梯度，并在正确的方向(梯度的相反方向)上推动θ来最小化损失函数。图 5.5 显示了该算法。

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

```
    # where: L is the loss function
    #        f is a function parameterized by θ
    #        x is the set of training inputs x(1), x(2),..., x(m)
    #        y is the set of training outputs (labels) y(1), y(2),..., y(m)

θ ← 0
repeat til done    # see caption
  For each training tuple (x(i), y(i)) (in random order)
    1. Optional (for reporting):        # How are we doing on this tuple?
       Compute ŷ(i) = f(x(i); θ)        # What is our estimated output ŷ?
       Compute the loss L(ŷ(i), y(i))   # How far off is ŷ(i) from the true output y(i)?
    2. g ← ∇θ L(f(x(i); θ), y(i))       # How should we move θ to maximize loss?
    3. θ ← θ − η g                      # Go the other way instead
return θ
```

图 5-5：随机梯度下降算法

图注：步骤 1(计算损失)用于报告我们在当前元组上的表现。该算法可以在收敛时(或在梯度范数<ϵ时)或进度停止时(例如，损失在保留集上开始上升)而终止。

学习速率η是必须调整的**超参数(hyperparameter)**。如果它的值太高，学习者将采取太大的步子，超出损失函数的最小值。如果它的值太低，学习者将采取太小的步子，并且花费太长时间才能达到最低要求。通常从较高的学习率开始，然后缓慢降低它，使其成为训练迭代 k 的函数。符号η_k可用于表示迭代 k 处学习率的值。

我们将在第 7 章中更详细地讨论超参数，但简要地讲，它们对于任何机器学习模型都是一种特殊的参数。与算法从训练集中学习的模型的常规参数(权重 w 和 b)不同，超参数是算法设计者选择的特殊参数，会影响算法的工作方式。

5.4.3. 弄通一个例子

让我们逐步了解一下梯度下降算法。我们将使用图 5.2 中示例的简化版本，因为它看到一个观测值 x，其正确值为 y = 1(这是积极的评论)，并且只有两个特征：

x_1 = 3(积极词汇数)

x_2 = 2(消极词汇数)

假设θ^0的初始权重和偏差都设置为 0，初始学习率η为 0.1：

$w_1 = w_2 = b = 0$

$\eta = 0.1$

单个更新步骤要求我们计算梯度乘以学习率：

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$$

在我们的小型示例中，有 3 个参数，因此梯度向量具有 3 个维度，分别是 w_1，w_2 和 b。我们可以如下计算第一个梯度：

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w\cdot x+b)-y)x_1 \\ (\sigma(w\cdot x+b)-y)x_2 \\ \sigma(w\cdot x+b)-y \end{bmatrix} = \begin{bmatrix} (\sigma(0)-1)x_1 \\ (\sigma(0)-1)x_2 \\ \sigma(0)-1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

现在我们有了一个梯度，我们通过从梯度的相反方向移动θ^0来计算新的参数矢量θ^1：

$$\theta^1 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix}$$

因此，在经过一个梯度下降步骤之后，权重已变为：$w_1 = .15$，$w_2 = .1$ 和 $b = .05$。

注意，该观察值 x 恰好是一个积极示例。我们希望看到更多带有大量消极词的消极示例后，权重 w_2 会变为负值。

5.4.4. 小批量训练

随机梯度下降法之所以被称为随机，是因为它一次选择单个随机示例，移动权重以提高该示例的性能。这可能会导致很不稳定的移动，因此通常在成批训练实例而非单个实例上计算梯度。

例如，在**批量(batch)**训练中，我们计算整个数据集的梯度。通过查看如此多的示例，批处理训练可以很好地估计权重的移动方向，计算此完美方向的代价是，要花费大量时间来处理训练集中的每个示例。

一个折衷方案是**小批量(mini-batch)**训练：我们训练一组小于整个数据集的 m 个示例(也许是 512 个或 1024 个)。(如果 m 是数据集的大小，那么我们将进行批量梯度下降；如果 m = 1，我们将返回随机梯度下降)。小批量训练还具有计算效率的优势。小批量可以很容易地进行矢量化，根据计算资源选择小批量的大小。这样一来，我们就可以在一个小批量处理中并行处理所有示例，然后累积损失，而这对于单个或批量训练而言是不可能的。

我们只需要定义在 5.3 节中定义的交叉熵损失函数和在 5.4.1 节中定义的梯度的小批量版本。让我们从公式 5.11 中把一个示例的交叉熵损失扩展到 m 大小的小批量。我们将继续使用 $x^{(i)}$和 $y^{(i)}$分别表示第 i 个训练特征和训练标签。我们假设训练示例是独立的：

$$\begin{aligned} \log p(\text{training labels}) &= \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}) \\ &= -\sum_{i=1}^{m} L_{\text{CE}}(\hat{y}^{(i)}, y^{(i)}) \end{aligned} \tag{5.19}$$

现在，m 个示例的小批量生产的成本函数是每个示例的平均损失：

$$Cost(\hat{y},y) = \frac{1}{m}\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)},y^{(i)})$$
$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\sigma(w\cdot x^{(i)}+b)+(1-y^{(i)})\log\left(1-\sigma(w\cdot x^{(i)}+b)\right) \quad (5.20)$$

小批量梯度是来自等式 5.18 的各个梯度的平均值：

$$\frac{\partial Cost(\hat{y},y)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left[\sigma(w\cdot x^{(i)}+b)-y^{(i)}\right]x_j^{(i)} \quad (5.21)$$

5.5. 正则化

Numquam ponenda est pluralitas sine necessitate

'Plurality should never be proposed unless needed'

"除非有必要，否则绝不应提出多个建议"

William of Occam

有一个学习权值的问题，使模型完全匹配训练数据。因为一个特征只出现在一个类别中，所以，如果它能够完美地预测结果，那么它将被赋予非常高的权重。特征的权重将试图完美(事实上过于完美)地拟合训练集的细节，对偶然与类相关的噪声因素进行建模。这个问题叫做**过拟合(overfitting)**。一个好的模型应该能够很好地从训练数据**概括(generalize)**到看不见的测试集，但一个过拟合的模型将会有很差的概括。

为了避免过拟合,一个新的**正则化(regularization)**项 R(θ)被添加到公式 5.13 的目标函数中，得出了一批 m 个示例的以下目标(从等式 5.13 略微重写，以使对数概率最大化而不是使损失最小化，并删除不影响 argmax 的 1/m 项)：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\sum_{i=1}^{m}\log P(y^{(i)}|x^{(i)}) - \alpha R(\theta) \quad (5.22)$$

新的正则化项 R(θ)用于惩罚较大的权重。

因此，假设有两种权重方案：

A：一个完全匹配训练数据的权重设置，但使用许多高值的权重；

B：一个稍好匹配训练数据的权重设置，但使用一些较小的权重。

那么，A 方案将受到更大的惩罚。

有两种常见的方法可以计算此正则项 R(θ)。

第一种是 L2 正则化。**L2 正则化(L2 regularization)**是权重值的二次函数，之所以这样命名，是因为它使用权重值的 L2 范数(的平方)。L2 范数，$||\theta||_2$，就是从原点到向量θ的距离(即向量的长度)，这个距离叫欧氏距离。如果θ由 n 个权重组成，则：

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n}\theta_j^2 \quad (5.23)$$

L2 正则化目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\left[\sum_{i=1}^{m}\log P(y^{(i)}|x^{(i)})\right] - \alpha\sum_{j=1}^{n}\theta_j^2 \quad (5.24)$$

第二种是 L1 正则化。**L1 正则化(L1 regularization)**是权重值的线性函数，以 L1 范数$||W||_1$命名，它是权重的绝对值之和或曼哈顿距离(曼哈顿距离是在像纽约这样有街道网格的城市里，你在两点之间所走的

距离)：

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{5.25}$$

L1 正则化目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j| \tag{5.26}$$

这些类型的正则化来自统计学，其中 L1 正则化称为**套索(lasso)**回归(Tibshirani，1996)，而 L2 正则化称为**岭脊(ridge)**回归，这两者都常用于语言处理中。L2 正则化因其简单的导数(θ^2 的导数仅为 2θ)而更易于优化，而 L1 正则化则更复杂(|θ|的导数在零处不连续)。但是，在 L2 偏爱具有许多较小权重的权重向量的情况下，L1 偏爱具有一些较大权重但又有更多权重设置为零的稀疏解。因此，L1 正则化导致稀疏的权重向量，即特征少得多。

L1 和 L2 正则化均具有贝叶斯(Bayesian)解释，作为对权重外观的先验约束。L1 正则化可以被视为权重上的拉普拉斯先验。L2 正则化对应于假设权重根据均值μ = 0 的高斯分布进行分配。在高斯或正态分布中，一个值与平均值的距离越远，其概率(由方差σ缩放)越低。通过对权重使用高斯先验，我们说权重更喜欢值为 0。权重θ_j的高斯函数为：

$$\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{5.27}$$

如果将每个权重乘以权重的高斯先验，则将使以下约束最大化：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{M} P(y^{(i)}|x^{(i)}) \times \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{5.28}$$

在对数空间中，μ= 0，并假设 $2\sigma^2$= 1，它对应于：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha \sum_{j=1}^{n} \theta_j^2 \tag{5.29}$$

其形式与公式 5.24 相同。

5.6. 多元逻辑回归

有时我们需要两个以上的类。也许我们可能想进行 3 种情感分类(正面，负面或中性)。或者，我们可以分配一些我们将在第 8 章中介绍的标签，例如单词的词类(从 10、30 甚至 50 个不同的词类中进行选择)，或短语的命名实体类型(选择来自人物、位置、组织等标签)。

在这种情况下，我们使用**多元逻辑回归(multinomial logistic regression)**，也称为 **softmax 回归**(或历史上称为 maxent 分类器)。在多元逻辑回归中，目标 y 是一个变量，范围超过两个类别。我们想知道 y 在每个潜在的类 c∈C，p(y = c | x)中的概率。

多元逻辑分类器使用广义的 Sigmoid，称为 **softmax** 函数，来计算概率 p(y = c | x)。softmax 函数采用 k 个任意值的向量 z = [z_1，z_2，...，z_k]并将其映射到概率分布，每个值的范围为(0,1)，所有值的总和为 1。像 S 形一样，它是一个指数函数。

对于维数为 k 的向量 z，softmax 定义为：

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k}\exp(z_j)} \quad 1 \le i \le k \tag{5.30}$$

输入向量 z = [z_1，z_2，...，z_k]的 softmax 因此是向量本身：

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k}\exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k}\exp(z_i)}, \ldots, \frac{\exp(z_k)}{\sum_{i=1}^{k}\exp(z_i)}\right] \tag{5.31}$$

分母$\sum_{i=1}^{k}\exp(z_i)$用于将所有值归一化为概率。因此，例如给定一个向量：

z = [0.6，1.1，−1.5，1.2，3.2，−1.1]

结果(四舍五入)softmax(z)为：

[0.055，0.090，0.006，0.099，0.74，0.010]

再次像 Sigmoid 一样，softmax 的输入将是权重向量 w 和输入向量 x(加上偏置)之间的点积。但是，现在我们需要为 K 个类中的每个类使用单独的权重向量(和偏差)。

$$p(y=c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{k}\exp(w_j \cdot x + b_j)} \tag{5.32}$$

像 Sigmoid 一样，softmax 具有将值朝 0 或 1 压缩的特性。因此，如果输入之一大于其他输入，则它将倾向于将其概率推向 1，并抑制较小输入的概率。

5.6.1. 多元逻辑回归的特征

多元逻辑回归函数的特征类似于二元逻辑回归，不同之处在于我们需要为每一个 K 类单独的权向量(和偏差)。回想一下 5.1.1 的二进制感叹号特征 x_5:

$$x_5 = \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$$

在二元分类中，一个特征的正权重 w_5 对分类器的影响趋向为 y = 1(积极情感)，一个负权重对分类器的影响趋向为 y = 0(消极情感)，其绝对值表示该特征的重要程度。相比之下，在多元逻辑回归中，每个类都有单独的权重，一个特征可以作为支持或反对每个单独类的证据。

例如，在三向(3-way)多类别情感分类中，我们必须为每个文档分配+，-或 0(中性)这 3 个类别之一。现在，与感叹号相关的特征可能对 0 文档的权重为负，而对+或-文档的权重为正：

Feature	Definition	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	−5.3

5.6.2. 多元逻辑回归的学习

多元逻辑回归的损失函数概括了从 2 类到 K 类的二进制逻辑回归的损失函数。回想一下，二进制逻辑回归的交叉熵损失(从公式 5.11 重复)为：

$$L_{\text{CE}}(\hat{y}, y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})] \tag{5.33}$$

多元逻辑回归的损失函数概括了方程式 5.33 中的两个项(当 y = 1 时有一个非零项，而当 y = 0 时也有一个非零项)到 K 个项。因此，单个示例 x 的损失函数是 K 个输出类的对数的总和，每个对数的权重为 y_k，

即真实类的概率：

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\ &= -\sum_{k=1}^{K} y_k \log \hat{p}(y=k|x) \end{aligned} \tag{5.34}$$

由于只有一个类(我们称其为 i)是正确的类，因此向量 y 仅针对此 k 值取值为 1，即 $y_i=1$ 且 $y_j=0\ \forall j \neq i$。像这样的一个向量，其一个值=1，其余值为 0，被称为**独热向量**。因此，公式 5.34 中的总和项将是 0(除了对应于真实类的项)，即:

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\} \log \hat{p}(y=k|x) \\ &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\} \log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)} \end{aligned} \tag{5.35}$$

因此，交叉熵损失仅仅是对应于正确类的输出概率的对数，因此我们也将其称为**负对数似然损失**：

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\log \hat{y}_k, \quad (\text{where } k \text{ is the correct class}) \\ &= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)} \quad (\text{where } k \text{ is the correct class}) \end{aligned} \tag{5.36}$$

单个示例的梯度与二进制逻辑回归的梯度非常相似，尽管我们在此处未显示推导。它是真实类 k 的值(为 1)与分类器为类 k 输出的概率之间的差值，用 k 类中的第 i 个元素权重 $w_{k,i}$ 对应的输入 x_i 的值，进行加权计算:

$$\begin{aligned} \frac{\partial L_{\mathrm{CE}}}{\partial w_{k,i}} &= -(\mathbb{1}\{y=k\} - p(y=k|x))x_i \\ &= -\left(\mathbb{1}\{y=k\} - \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}\right) x_i \end{aligned} \tag{5.37}$$

5.7. 解释模型

通常，我们不仅要了解观察的正确分类，还希望了解更多。我们想知道为什么分类器做出了决策。也就是说，我们希望我们的决策是可以**解释的(interpretable)**。可解释性可能很难严格定义，但是核心思想是作为人类，我们应该知道为什么我们的算法得出他们得出的结论。由于逻辑回归的特征通常是人为设计的，因此了解分类器决策的一种方法是了解每个特征在决策中所扮演的角色。逻辑回归可以与统计检验(似然比率检验或 Wald 检验)结合使用；通过这些测试之一调查某个特定特征是否很重要，或者检查其大小(与该特征相关联的权重 w 多大？)可以帮助我们解释分类器为何做出决策。这对于建立透明模型非常重要。

此外，除了将其用作分类器外，NLP 和许多其他领域的逻辑回归还广泛用作分析工具，以测试有关各种解释变量(特征)的影响的假设。在文本分类中，也许我们想知道逻辑上否定的词(不、不是、从不)更可能与否定的情感相关联，或者电影的否定评论更可能讨论电影摄影。但是，这样做有必要控制潜在的混杂因素：其他可能影响情感的因素(电影体裁、制作年份或评论的长度)。或者，我们可能正在研究 NLP 提取的

语言特征与非语言结果(再次入院，政治结果或产品销售)之间的关系，但需要控制混杂因素(患者的年龄、县区的投票、产品的品牌)。在这种情况下，逻辑回归允许我们测试某些特征是否与某些结果相关，并且超出了其他特征的影响。

5.8. 高级专题：推导梯度公式

在本节中，我们为逻辑回归给出了交叉熵损失函数 L_{CE} 的梯度的推导。

让我们从一些快速的演算复习开始[1]。

5.8.1. 复习

首先，ln(x)的导数：

$$\frac{d}{dx}\ln(x) = \frac{1}{x} \tag{5.38}$$

第二，sigmoid 的(非常优雅的)导数:

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \tag{5.39}$$

最后，导数的**链式规则(chain rule)**。假设我们正在计算复合函数 f(x)= u(v(x))的导数。f(x)的导数是 u(x)关于 v(x)的导数乘以 v(x)关于 x 的导数：

$$\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx} \tag{5.40}$$

5.8.2. 推导

首先，我们想知道损失函数相对于单个权重 w_j 的导数(我们需要针对每个权重和偏差来计算它)：

$$\begin{aligned}\frac{\partial L_{\text{CE}}}{\partial w_j} &= \frac{\partial}{\partial w_j} -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -\left[\frac{\partial}{\partial w_j}y\log\sigma(w\cdot x+b)+\frac{\partial}{\partial w_j}(1-y)\log[1-\sigma(w\cdot x+b)]\right]\end{aligned} \tag{5.41}$$

接下来，利用链式规则，借助对数的导数:

$$\frac{\partial L_{\text{CE}}}{\partial w_j} = -\frac{y}{\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) - \frac{1-y}{1-\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}1-\sigma(w\cdot x+b) \tag{5.42}$$

重新整理:

$$\frac{\partial L_{\text{CE}}}{\partial w_j} = -\left[\frac{y}{\sigma(w\cdot x+b)}-\frac{1-y}{1-\sigma(w\cdot x+b)}\right]\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) \tag{5.43}$$

现在，插入 Sigmoid 的导数，再使用链式规则一次，得出公式 5.44：

[1] 译者注：“5.8.1 复习，5.8.2 推导”是译者自己添加的三级标题，以便增加可读性。

$$\begin{aligned}\frac{\partial L_{\text{CE}}}{\partial w_j} &= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]\frac{\partial(w\cdot x+b)}{\partial w_j} \\ &= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]x_j \\ &= -[y-\sigma(w\cdot x+b)]x_j \\ &= [\sigma(w\cdot x+b)-y]x_j \end{aligned} \tag{5.44}$$

5.9. 总结

本章介绍了**分类**的**逻辑回归**模型。

•逻辑回归是一种受监督的机器学习分类器，可从输入中提取实值特征，将每个特征与权重相乘，对其求和，然后将总和传递给 **Sigmoid** 函数以生成概率。阈值用于做出决定。

•逻辑回归可以用于两个类别(例如，正面情感和负面情感)，也可以用于多个类别(**多元逻辑回归**，例如用于 n 元(n-ary)文本分类，词类标注等)。

•多元逻辑回归使用 **softmax** 函数计算概率。

•权重(向量 w 和偏差 b)是从标记的训练集中通过损失函数(如**交叉熵损失**)学习的，必须将其最小化。

•最小化此损失函数是一个**凸优化**问题，并且使用诸如**梯度下降**的迭代算法来找到最佳权重。

•**正则化**用于避免**过拟合**。

•逻辑回归也是最有用的分析工具之一，因为它能够透明地研究各个特征的重要性。

5.10. 文献和历史说明

逻辑回归是在统计学领域发展起来的，它在 20 世纪 60 年代被用于二进制数据的分析，在医学中尤其常见(Cox, 1969)。从 20 世纪 70 年代后期开始，作为语言变异研究的正式基础之一，它在语言学中得到了广泛的应用(Sankoff 和 Labov, 1979)。

然而，逻辑回归直到 20 世纪 90 年代才在自然语言处理中变得普遍，当时它似乎同时出现在两个方向：第一个是邻近的信息检索，第二个是语音处理领域。这两个方向都使用了回归，都为自然语言处理提供了许多其他的统计技术。实际上，很早就将逻辑回归用于文档传送是最早使用(LSI)嵌入作为单词表示的 NLP 应用之一(Schutze 等人 1995)。

与此同时，在 20 世纪 90 年代早期，IBM 研究中心以最大熵模型或 maxent (Berger 等人，1996)的名义开发了逻辑回归并应用于自然语言处理，似乎独立于统计文献。在这个名称下，它被应用于语言建模(Rosenfeld, 1996)、词类标注(Ratnaparkhi, 1996)、解析(Ratnaparkhi, 1997)、关联解析(Kehler, 1997b)和文本分类(Nigam 等人 1999)。

更多关于分类的内容可以在机器学习教科书中找到(Hastie 等人 2001, Witten 和 Frank 2005, Bishop 2006, Murphy 2012)。

5.11. 练习

(空缺)

6. 向量语义和嵌入

荃者所以在鱼，得鱼而忘荃 Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言
Words are for meaning;
Once you get the meaning, you can forget the words
庄子(Zhuangzi), Chapter 26

洛杉矶闻名的沥青主要发生在高速公路上。但是在城市中心，还有另一块沥青，La Brea 焦油坑，这种沥青保存着更新世冰河时代末期的数百万枚化石。这些化石之一是Smilodon(剑齿虎)，可通过其长犬齿立即辨认出来。五百万年前左右，另一种完全不同的剑齿虎(Thylacosmilus)生活在阿根廷和南美其他地区。Thylacosmilus 是有袋动物，而 Smilodon 是胎盘哺乳动物，但是 Thylacosmilus 具有相同的长上犬齿，并且像 Smilodon 一样，在下颌具有保护性的骨突。这两种哺乳动物的相似性是平行或趋同进化的许多例子之一，在这种情况下，特定的语境或环境导致了不同物种中非常相似的结构的进化(Gould 1980)。



语境的作用在单词(一种生物特征较低的有机体)的相似性上也很重要。在相似的语境中出现的单词往往有相似的意思。单词如何分布的相似性和单词含义的相似性之间的联系被称为**分布假说(distributional hypothesis)**。该假设最初是在 1950 年代由语言学家，例如 Joos(1950)，Harris(1954)和 Firth(1957)提出的，他们注意到，同义词(如 oculist 和 eye-doctor，即“眼科医生”)倾向于在相同的环境中(诸如 eye 或者 examined 这样的近义词)出现，两个词之间的意义差异量“大致对应于他们所处环境的差异量”(Harris，1954，157)。

在这一章中，我们将介绍**向量语义(vector semantics)**，它通过学习词汇的意义表征，即**嵌入(embedding)**，直接从词汇在文本中的分布来实例化这一语言假设。这些表示形式在每种自然语言处理应用程序中都会使用，我们在这里介绍的静态嵌入是功能更强大的动态或上下文嵌入的基础，例如 BERT，我们将在第 10 章中看到。

这些单词的表示也是本书**学习表示(representation learning)**的第一个示例，可自动学习输入文本的有用表示形式。寻找这种自我监督的方法来学习输入的表示形式，而不是通过特征工程手动创建表示形式，是 NLP 研究的重要焦点(Bengio 等人，2013)。

6.1. 词汇语义

首先，我们介绍一些单词含义的基本原理。我们应该如何表示一个单词含义？在第 3 章的 n-gram 模型中以及在经典的 NLP 应用程序中，我们对单词的唯一表示形式是字母字符串或词汇表中的索引。这种表示形式与哲学传统没有什么不同，也许您已经在逻辑入门课中看到了，其中单词的含义是用小写字母拼写该词；dog 的含义是“狗”、cat 的含义是“猫”。

通过大写来表示一个单词的意思是一种非常不令人满意的模式。你可能见过一个由语义学家 Barbara Partee (Carlson, 1977)讲的笑话:

Q: What’s the meaning of life?

A: LIFE’

当然，我们可以做得更好！毕竟，我们需要一个单词含义模型来为我们做各种事情。它应该告诉我们，有些单词的含义相似(猫类似于狗)，有些则是反义词(冷是热的反义词)，有些具有积极的含义(快乐)，而另一些具有消极的含义(悲伤)。它应该代表一个事实，即购买、出售和支付的含义在同一购买的基本事件上

提供了不同的观点(如果我从您那里购买东西，您可能已经将其出售给了我，并且我可能已经支付给了您)。更一般而言，单词含义模型应允许我们得出推论来解决与含义相关的任务，例如问题解答或对话。

在本节中，我们将根据语言学对单词含义的研究成果，即**词汇语义(Lexical semantics)**，总结其中的一些渴望得到的东西；我们将在第 18 章和第 10 章继续讨论这个列表(list)。

词元和意义

让我们先看看一个单词 mouse 在字典(从在线词典 WordNet 简化而来)中是如何定义的:

mouse (N)

1. any of numerous small rodents...(许多小啮齿动物中的任何一种…)

2. a hand-operated device that controls a cursor...(一种控制光标的手动装置…)

这里形式 mouse 是**词元(lemma)**，也称为**引用形式(citation form)**。mouse 这个形式也是 mice 这个词的词元;字典对像 mice 这样的词形变化没有单独的定义。同样，sing 是 sing、sang、sung 的词元。在许多语言中，不定式被用作动词的词元，所以，在西班牙语中，dormir“即 to sleep”是 duermes “即 you sleep”的词元理。特定的形式 sung 或 carpets 或 sing 或 duermes 被称为**词形(wordform)**。

如上面的示例所示，每个词元可以具有多种含义。词元 mouse 可以指称啮齿动物或光标控制设备。我们将 mouse **含义(meaning)**的每个**体(aspect)**称为单词**意义(sense)**。事实上词元可以是多义的(具有多种意义)，这个事实会使解释变得困难(有人在搜索引擎中输入“mouse info”是为了寻找宠物或工具吗?)第 18 章将讨论一词多义的问题，并介绍**词义消歧(word sense disambiguation)**的任务，即确定一个词的哪个意义是在一个特定的上下文中被使用。

同义关系(synonymy)

单词含义(word meaning)的一个重要成分是**单词意义(word sense)**之间的关系。

例如，当一个单词有一个**意义(sense)**，它的**含义(meaning)**与另一个单词的意义相同或几乎相同时，我们说这两个单词的两个意义是**同义词(synonym)**。

同义词包括诸如：

couch/sofa (沙发)　　vomit/throw (呕吐)　　up filbert/hazelnut (榛子)　　car/automobile (汽车)

同义关系(在单词之间而不是在意义之间)更正式的定义是，如果两个词在任何一个句子中可以相互替换，而不改变句子的真值条件，也就是这个句子为真的情况，那么这两个词就是同义词。在这种情况下，我们经常说这两个词具有相同的**命题含义(propositional meaning)**。

虽然一些词对之间的替换，如 car / automobile 或 water / H_2O 是保真(truth preserving)的，但这些词仍然有不相同的含义。的确，可能没有两个词的含义完全相同。语义学的基本原则之一，叫做**对比原理(principle of contrast)**(Girard 1718，Br´eal 1897，Clark 1987)，指出语言形式的差异总是与含义上的某些差异相关。例如，H_2O 一词在科学环境中使用，在徒步旅行指南中不合适(水会更合适)，这种体裁差异是该词含义的一部分。因此在实践中，**同义词(synonym)**用于描述近似或粗略的**同义关系(synonymy)**。

词相似(word similarity)

虽然单词没有很多同义词，但是大多数单词都有很多相似的单词。猫不是狗的同义词，但猫和狗肯定是相似的词。在从同义转换到相似的过程中，从讨论单词词义之间的关系(如同义词)转换到单词本身之间的关系(如相似词)将会很有用。处理单词避免了必须致力于词义的特定表达，这将简化我们的任务。

词相似的概念在较大的语义任务中非常有用。知道两个单词的相似度可以帮助计算两个短语或句子的含义有多么相似，这是自然语言理解任务(如问题回答、释义和总结)中非常重要的组成部分。获取词相似值的一种方法是要求人们判断一个单词与另一个单词的相似度。这样的实验产生了许多数据集例如，SimLex-999 数据集(Hill 等人，2015)给出的值范围为 0 到 10，如以下示例所示，范围从近义词(vanish, disappear)到几乎没有任何共同之处的一对词(hole, agreement)：

vanish	disappear	9.8
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

词关联(Word Relatedness)

两个单词的含义可以通过相似以外的其他方式联系，其中一种被称为**词关联(word relatedness)**(Budanitsky 和 Hirst，2006)，在心理学上也传统地称为**词联想(word association)**。

想想“咖啡”和“杯子”这两个词的含义。咖啡和杯子不一样；它们几乎没有相同的特征(咖啡是一种植物或一种饮料，而杯子是具有特定形状的人造物体)。但是咖啡和杯子显然是有联系的;他们通过共同参与一项日常活动(用杯子喝咖啡的活动)而联系在一起。同样，解剖刀和外科医生并不相似，但最终是相关的(外科医生往往会使用解剖刀)。

词汇之间的一种常见的联系是它们是否属于同一**语义场(semantic field)**。语义场是指涵盖特定语义域并相互之间具有结构化关系的单词集合。例如，在医院(外科医生、手术刀、护士、麻醉师、医院)、餐馆(服务员、菜单、盘子、食物、厨师)或房子(门、屋顶、厨房、家庭、床)的语义场中，单词可能是相关的。语义场也与主题模型相关，例如，隐狄利克雷分配模型(Latent Dirichlet Allocation，LDA)在大量文本上应用无监督学习，从文本中归纳出相关单词集。语义场和主题模型是发现文档主题结构的非常有用的工具。

在第 18 章，我们将介绍更多的意义(sense)之间的关系，如**上位关系(hypernymy)**或 IS-A，**反义关系(antonymy)**和**部分-整体关系(meronymy)**。

语义框架和角色(Semantic Frames and Roles)

与语义场密切相关的是语义框架的概念。语义框架是一组表示特定类型事件的视角或参与者的单词。例如，商业交易是一个实体与另一个实体进行货币交易以换取某种商品或服务的事件，在此之后，商品易手或服务得以执行。这个事件可以通过使用诸如 buy(从买方角度看的事件)、sell(从卖方角度看的事件)、pay(侧重于货币方面)等动词或 buyer 之类的名词来进行词汇编码。框架具有语义角色(如 buyer, seller, goods, money)，句子中的单词可以扮演这些角色。

知道 buy 和 sell 具有这种关系，就可以使系统知道像 Sam bought the book from Ling 这样的句子可以被解释为 Ling sold the book to Sam，Sam 在框架中起 buyer 的作用，而 Ling 是 seller。能够识别此类释义对于回答问题很重要，且可帮助机器翻译的转移视角。

内涵

最后，单词有**感情含义(affective meaning)**或**内涵(connotations)**。单词内涵在不同的领域有不同的含义，但在这里我们用它来表示一个词的含义与作者或读者的情绪、情感、观点或评价有关的方面。例如，有些词有积极的内涵(快乐)，而另一些词有消极的内涵(悲伤)。即使在其他方面意义相似的单词在内涵上也会有所不同；考虑一下 fake、knockoff、forgery 的内涵和 copy、 replica、reproduction 的内涵之间的区别，或者 innocent (正面内涵)和 naive (负面内涵)之间的区别。有些词描述正面评价(伟大，爱心)，而其他负面描述(可怕，憎恨)。正如我们在第 4 章中看到的那样，正面或负面评价语言都称为**情感(sentiment)**，而单词情感在情感分析，立场检测以及将 NLP 应用于政治和消费者评论等重要任务中扮演重要角色。

早期关于情感含义的研究(Osgood 等人，1957)发现单词在情感含义的三个重要维度上各不相同：

效价(Valence)： 刺激的愉悦

唤起(Arousal)： 刺激引起的情绪强度

支配(Dominance)： 刺激施加的控制程度

因此，诸如 happy 或 satisfied 之类的词的效价高，而 unhappy 或 annoyed 之类的词的效价低。Excited 在唤起时很高，而 calm 在唤起时很低。Controlling 支配高，而 awed 或 influenced 则支配低。

因此，每个单词都由三个数字表示，分别对应于三个维度中每个单词的值：

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

Osgood 等人(1957)注意到，使用这 3 个数字表示单词的含义时，模型将每个单词表示为三维空间中的一个点，该矢量的三个维度就是从三个度量上对单词进行评价。这个革命性的想法是，单词的含义可以用空间中的一个点来表示(例如，heartbreak 的部分含义可以用点[2.45,5.65,3.58]来表示)，这是我们接下来要介绍的向量语义模型的第一个表达。

6.2. 向量语义

向量语义(vector semantics)是在 NLP 中表示单词含义的标准方法，可帮助我们对上一节中看到的单词含义的许多方面进行建模。该模型的起源可追溯到 1950 年代，当时汇合了两个主要观点：一个是上面提到的 Osgood(1957)观点，使用三维空间中的点表示单词的内涵；另一个是由语言学家 Joos(1950)、Harris(1954)和 Firth(1957)建议的观点：通过一个词在语言使用中的分布(它的邻近词或语法环境)来定义它的含义。他们的想法是，两个出现在非常相似分布中的单词(相邻的单词很相似)有相似的含义。

例如，假设您不知道 ongchoi 一词(最近从广东话借来的)的含义，但在以下情况下会看到它：

(6.1) 用大蒜炒的 Ongchoi 很好吃。

(6.2) Ongchoi 是米饭的上品。

(6.3) ...带有浓汁的 ongchoi 叶子...

假设你在其他语境中见过很多这样的单词:

(6.4)…大蒜炒菠菜米饭…

(6.5)…甜菜的茎和叶很美味。

(6.6)…散叶甘蓝和其他含盐绿叶蔬菜

事实上，ongchoi 与大米、大蒜、美味和咸味等词一起出现，像菠菜、甜菜和散叶甘蓝等绿色蔬菜可能表明 ongchoi 是一种绿叶蔬菜，与其他绿叶蔬菜类似[1]。我们可以通过计算 ongchoi 语境中的单词数量来做同样的事情。

向量语义的思想是将一个单词表示为多维语义空间中的一个点，该多维语义空间是从单词邻居的分布中派生出来的(按照我们将要看到的方式)。表示单词的向量称为嵌入(尽管有时该术语仅更严格地应用于诸如 word2vec(第 6.8 节)之类的密集向量，而不是稀疏的 tf-idf 或 PPMI 向量(第 6.3 节--第 6.6 节))。“嵌入”一词源于其数学意义，即从一种空间或结构到另一种空间或结构的映射，尽管其含义已发生变化。请参阅本章的末尾。

图 6.1 显示了为情感分析而学习的嵌入的可视化，显示了从 60 维空间向下投射到二维空间中所选单词的位置。请注意包含正面词，负面词和中性词的不同区域。

向量语义的单词相似度细粒度模型为 NLP 应用程序提供了强大的功能。NLP 应用程序(如第 4 章或第 5 章的情感分类器)依赖于训练和测试集中出现的相同单词。但是通过将单词表示为嵌入，分类器可以分配情感，只要它看到一些具有相似含义的单词即可。正如我们将看到的，向量语义模型可以自动从文本中学习，而无需监督。

在本章中，我们将介绍两种最常用的模型。在 **tf-idf 模型**(一个重要的基线)中，一个词的含义是由邻近词的计数的一个简单函数定义的。我们将看到，这种方法会产生非常长的**稀疏(sparse)**向量，即大部分为零(因为大多数单词都不会在其他单词的上下文中出现)。我们将介绍**词向量(word2vec)**模型家族，用于构造短小**密集(dense)**的向量，这些向量具有有用的语义属性。我们还将介绍**余弦(cosine)**，这是一种使用

[1]实际上，它是豆蔻（Ipomoea aquatica)，牵牛花(morning glory)的亲戚，有时也被称为水菠菜(water spinach)。

嵌入来计算两个单词、两个句子或两个文档之间的语义相似度的标准方法，是一种问答、摘要或自动作文评分等实际应用中的重要工具。



图 6-1：部分单词和短语的二维嵌入投影(t-SNE)

图注：词义相近的单词在空间上相邻。对原始的 60 维嵌入进行训练，用于情感分析。简化自 Li 等人(2015)，添加颜色解释。

6.3.单词和向量

“The most important attributes of a vector in 3-space are { Location, Location, Location }”

Randall Munroe, https://xkcd.com/2358/

含义的向量或分布模型通常基于**共现矩阵(co-occurrence matrix)**，这是表示单词共现频率的一种方式。我们将研究两种流行的矩阵：术语-文档矩阵和术语-术语矩阵。

6.3.1. 向量和文档

在术语-文档矩阵中，每一行代表词汇表中的单词，每一列代表一些文档集中的文档。图 6.2 显示了从术语-文档矩阵中进行的少量选择，该矩阵显示了莎士比亚在四个剧本中出现四个单词。此矩阵中的每个单元格代表特定单词(由行定义)在特定文档(由列定义)中出现的次数。因此，fool 在《Twelfth Night》中出现了 58 次。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

图 6-2：术语-文档矩阵

图注：莎士比亚戏剧中四个单词的术语-文档矩阵。每个单元格包含(行)字在(列)文档中出现的次数。

首先将图 6.2 的术语-文档矩阵定义为信息检索向量空间模型的一部分(Salton，1971)。在此模型中，文档被表示为计数向量，即图 6.3 中的一列。

回顾一些基本的线性代数：从本质上讲，**向量(vector)**只是一个数字列表或数字数组。因此，《As You Like It》表示为列表[1,114,36,20](图 6.3 中的第一列向量)，而《Julius Caesar》则表示为列表[7,62,1,2](第三列向量)。**向量空间(vector space)**是向量的集合，以向量的**维度(demention)**为特征。在图 6.3 的示例中，文档向量的维度为 4，正好适合页面。在实时术语-文档矩阵中，表示每个文档的向量的维数度为| V |(即词汇量的大小)。

向量空间中数字的顺序表明文档在不同的有意义的维度上有所不同。因此，这两个向量的第一个维度对应于单词 battle 发生的次数，而且我们可以比较每个维度，请注意，例如，《As You Like It》和《Twelfth Night》的向量对于第一个维度有相似的值(分别为 1 和 0)。

我们可以将文档的矢量视为| V |维空间中的一个点。因此，图 6.3 中的文档是 4 维空间中的点。由于 4 维空间很难可视化，因此图 6.4 显示了二维可视化。我们已任意选择与“battle”和“ fool”一词相对应的维度。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

图 6-3：术语-文档矩阵的列向量

图注：莎士比亚四部戏剧中四个单词的术语-文档矩阵，红色方框表示每个文档都表示为长度为 4 的列向量。

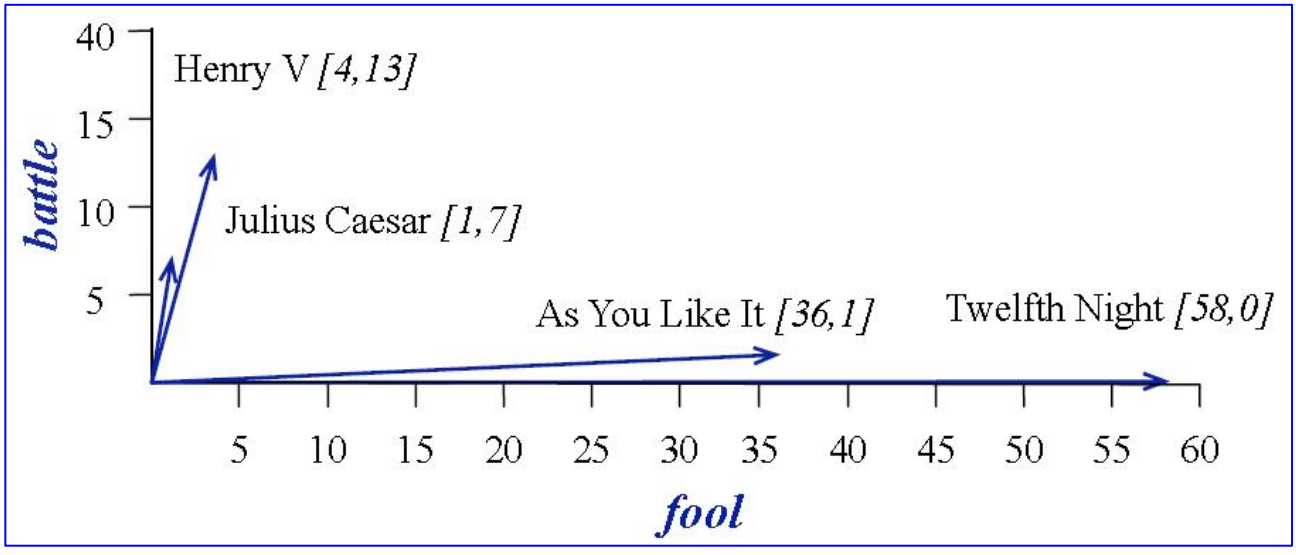


图 6-4：文档向量的空间可视化

图注：四个莎士比亚戏剧文档的文档向量的空间可视化，仅显示了两个维度，分别对应于 battle 和 fool 一词。喜剧对于 fool 维度具有较高的价值，而对于 battle 维度则具有较低的价值。

术语-文档矩阵最初被定义为一种用于查找文档信息检索任务的相似文档的方法。相似的两个文档倾向于具有相似的单词，如果两个文档具有相似的单词，则它们的列向量将趋于相似。喜剧《As You Like It》(1,114,36,20)和《Twelfth Night》(0,80,58,15)的向量看起来更像对方(更多的是 fools 和 wit，而不是 battle)，而不是像《凯撒大帝》(Julius Caesar)(7,62,1,2)或《亨利五世》(Henry V)(13,89,4,3)。从原始数据可以清楚地看出这一点;在第一个维度(battle)中，喜剧的数字较低，其他的数字较高，我们可以从图 6.4 中直观地看到这一点;我们很快就会看到如何更正式地量化这种直觉。

当然，真实的术语-文档矩阵不会只有 4 行和 4 列，更不用说 2 了。更一般地，术语-文档矩阵具有| V |行(针对词汇表中的每个单词类型)和 D 列(针对集合中的每个文档之一)；正如我们将看到的，词汇量通常在成千上万，文档的数量也可能非常庞大(想想网络上的所有网页)。

信息检索(Information retrieval，IR)的任务是从某个集合中与查询 q 最匹配的 D 个文档中找到文档 d。因此，对于 IR，我们还将通过向量(长度也是| V |)来表示查询，并且我们需要一种比较两个向量的方式，以查找它们之间的相似度。(通过利用这些向量稀疏即大部分为零的方便事实，进行 IR 还将需要有效的方式来存储和操纵这些向量。)

在本章的后面，我们将介绍矢量比较过程的一些组成部分：tf-idf 项加权和余弦相似性度量。

6.3.2. 单词作为向量：文档维度

我们已经知道文档可以用向量空间中的向量表示。但是矢量语义也可以用来表示单词的意思。我们通过将每个单词与一个单词向量联系起来——一个**行向量(row vector)**而不是列向量，因此有不同的维数，如图 6.5 所示。fool 的四维向量[36,58,1,4]相当于莎士比亚的四部戏剧。在相同的四个维度上的单词数可以用来构成其他三个单词的向量：wit[20,15,2,3]；battle[1,0,7,13]；good[114,80,62,89)。

对于文档，我们看到相似的文档有相似的向量，因为相似的文档往往有相似的单词。

同样的原则也适用于单词:相似的单词有相似的向量，因为它们往往出现在相似的文档中。因此，术语-文档矩阵让我们可以通过单词出现的文档来表示单词的含义。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

图 6-5：术语-文档矩阵的行向量

图注：莎士比亚四部戏剧中四个单词的术语-文档矩阵，红色方框表示每个单词都用长度为 4 的行向量表示。

6.3.3. 单词作为向量：单词维度

使用术语-文档矩阵将单词表示为文档计数向量的另一种方法是使用术语-术语矩阵，也称为词-词矩阵或术语-上下文矩阵，其中列是用单词而不是文档标记的。因此，该矩阵的维数为|V|×|V|，每个单元格记录在某些训练语料中行(目标)词和列(上下文)词在某些上下文中共现的次数。上下文可以是文档，在这种情况下，单元格表示两个单词在同一文档中出现的次数。但是，最常见的情况是使用较小的上下文，通常是围绕单词的窗口，例如，左边 4 个单词，右边 4 个单词，在这种情况下，单元格表示列词在行词周围±4 个词窗口中出现的次数(在某些训练语料库中)。例如，以下是其窗口中每个单词的一个示例：

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

如果我们取每个单词的每一次出现(比如 strawberry)，并计算它周围的上下文单词，我们就得到一个单词-单词共现矩阵。图 6.6 显示了从维基百科语料库计算的这四个单词的词-词共现矩阵的简化子集(Davies, 2015)。

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

图 6-6：四个单词的共现向量

图注：维基百科语料库中四个单词的共现向量，显示六个维度(为教学目的而精心挑选)。单词 digital 的矢量用红色标出。请注意，实向量的维数要大得多，因此也要稀疏得多。

从图 6.6 中可以看出，单词 cherry 和 strawberry 比其他单词 digital 更相似(pie 和 sugar 都倾向于出现在它们的窗口中)；相反，digital 和 information 比 strawberry 更相似。图 6.7 为空间可视化。

请注意，| V |(向量的长度)通常是词汇表的大小，通常在 10,000 到 50,000 个词之间(使用训练语料库中最频繁的词；通常保留最频繁的 50,000 个之后的词是没有帮助的)。由于这些数字大多数为零，因此它们是稀疏矢量表示形式。有一些用于存储和计算稀疏矩阵的有效算法。

现在我们有了一些直觉，让我们继续研究计算单词相似度的细节。之后，我们将讨论对单元格(cell)

进行加权的方法。

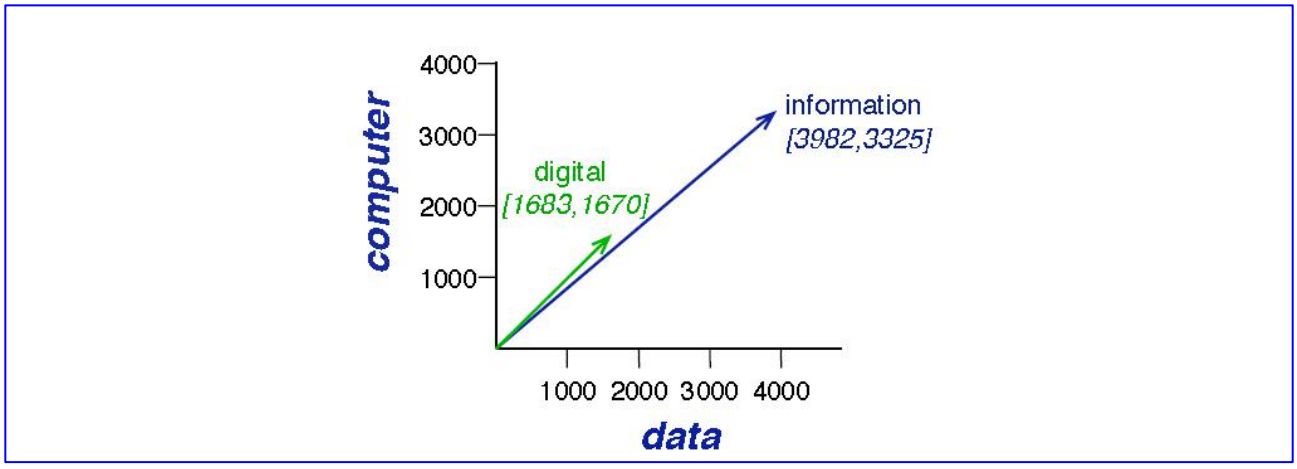


图 6-7：词向量的空间可视化

图注：单词 digital 和 information 的词向量的空间可视化，仅显示了两个维度，分别对应于单词 data 和 computer。

6.4. 余弦测量相似度

为了测量两个目标词 v 和 w 之间的相似性，我们需要一个度量，该度量采用两个向量(具有相同维度，两个向量都以词汇量的长度| V |为维度，或者两个都以文件的长度 | D |作为维度)并给出它们相似性的度量。到目前为止，最常见的相似性度量是向量之间的角度的余弦值。

余弦就像大多数 NLP 中使用的向量相似性度量一样，也基于线性代数的**点积(dot product)**算子，也称为**内积(inner product)**：

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N \tag{6.7}$$

正如我们将看到的，向量之间相似度的大多数度量都是基于点积。点积充当相似性度量，因为仅当两个向量在相同维度上具有较大值时，它才会趋于较高。或者，在不同维度上具有零的向量(正交向量)的点积为 0，表示它们的强烈不相似性。

但是，这种原始点积存在相似性度量的问题：它偏爱**长(long)**向量。**向量长度(vector length)**定义为：

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2} \tag{6.8}$$

如果向量较长，则点积会更高，每个维度中的值都将更高。频率更高的单词具有更长的向量，因为它们往往与更多的单词共现，并且每个单词的共现值更高。因此，对于经常使用的单词，原始点积会更高。但这是一个问题。我们想要一个相似度指标，该指标可以告诉我们两个单词的相似度，无论其频率如何。

我们修改点积，通过点积除以两个向量的长度来对向量长度进行归一化。根据向量 **a** 和 **b** 的点积的定义，这个归一化的点积等于两个向量夹角的余弦值:

$$\begin{aligned} \mathbf{a}\cdot\mathbf{b} &= |\mathbf{a}||\mathbf{b}|\cos\theta \\ \frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} &= \cos\theta \end{aligned} \tag{6.9}$$

因此，两个向量 v 和 w 之间的余弦相似性度量可以计算为:

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{6.10}$$

对于某些应用程序，我们通过将每个向量除以其长度来预归一化，以创建长度为 1 的单位向量(unit vector)。因此，我们可以通过从 **a** 除以|**a**|来计算单位向量。对于单位向量，点积与余弦相同。

余弦值的范围从 1(指向相同方向的向量)到 0(正交向量)到-1(指向相反方向的向量)。但是由于原始频率值是非负值，因此这些向量的余弦范围为 0–1。

让我们看看余弦如何计算单词 cherry 或 digital 在意义上更接近于 information，只是使用以下缩短表中的原始计数:

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

该模型确定 information 更接近 digital 而不是 cherry，这一结果似乎是合理的。图 6.8 显示了可视化。

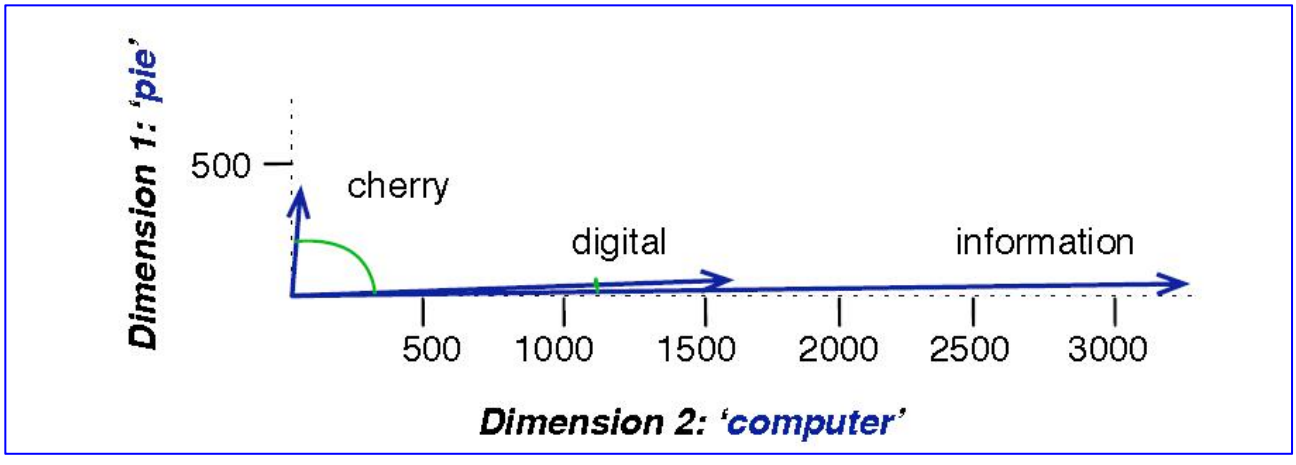


图 6-8：余弦相似度的(粗略)图形演示

图注：显示了二维空间中三个单词(cherry, digital, information)的向量，该向量由附近的单词 computer 和 pie 的计数定义。请注意，digital 和 information 之间的角度小于 cherry 和 information 之间的角度。当两个向量更相似时，余弦较大，但角度较小； 当两个向量之间的角度最小(0°)时，余弦具有最大值(1)； 所有其他角度的余弦值小于 1。

6.5. TF-IDF：向量中的权重项

上面的共现矩阵通过频率表示每个单元，或者是带有文档的单词(图 6.5)，或者是带有其他单词的单词(图 6.6)。但是原始频率并不是单词之间关联的最佳度量。原始频率非常倾斜，也不是很有区别。如果我们想知道哪种类型的上下文，是被 cherry 和 strawberry 共享的而不是 digital 和 information 共享的，则我

们将无法很好地区别于诸如 the、it、或 they 之类的单词，它虽然经常出现在各种单词中，但是并不能提供任何特定单词的信息。我们在图 6.3 中也看到了莎士比亚语料库。good 这个词的维度在不同的剧本之间不是很有区别。good 只是一个常用词，在每个剧本中都有大致相等的高频。

这有点自相矛盾。经常出现在附近的单词(也许是 pie 邻近 cherry)比只出现一次或两次的单词更重要。然而，过于频繁的词语(无所不在，如 the 或 good)并不重要。我们如何平衡这两个相互矛盾的约束？

此问题有两种常见的解决方案：在本节中，我们将介绍 tf-idf 算法，通常在维度是文档时使用。接下来，我们介绍 PPMI 算法(通常在维度是单词时使用)。

tf-idf 算法(此处的“-”是连字符，而不是减号)是两个术语的乘积：

第一个术语是**词频(term frequency，tf)**(Luhn, 1957)：即单词 t 在文档 d 中的频率。我们可以用原始计数作为词频:

$$\text{tf}_{t,d} = \text{count}(t,d) \tag{6.11}$$

更常见的是，我们通过使用频率的 log10 来压缩原始频率。直觉是，一个单词在文档中出现 100 次不会使该单词与文档含义相关的可能性提高 100 倍。因为我们不能取 0 的对数，所以通常将 1 加到计数中[2]：

$$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d)+1) \tag{6.12}$$

如果我们使用对数加权，则在文档中出现 0 次的项 tf = $\log_{10}$(0+1)= 0，在文档中出现 10 次的 tf = $\log_{10}$(10+1)= 1.4，100 次的 tf = $\log_{10}$(100+1)= 2.004， 1000 次的 tf = 3.00044，依此类推。

第二个术语是**反文档频率(inverse document frequency，idf)**：用于赋予仅在少数几个文档中出现的单词更高的权重。限定在少数几个文档中的术语有助于将这些文档从集合的其余部分中区分出来;在整个集合中频繁出现的术语没有那么有用。术语 t 的文档频率 df_t 是它出现在文档中的数量。文档频率与术语的收集频率不同，术语收集频率是单词在任何文档的整个集合中出现的总次数。在莎士比亚的 37 个戏剧中考虑两个角色，即 Romeo 和 action。这两个单词具有相同的收集频率(它们在所有戏剧中都出现 113 次)，但文档频率却非常不同，因为 Romeo 仅在单个戏剧中出现。如果我们的目标是查找有关 Romeo 浪漫磨难的文档，则应高度重视 Romeo 一词，而不是 action 一词。

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

我们通过 idf 来强调诸如 Romeo 之类的判别词(Sparck Jones，1972)。idf 是用分数 N/df_t 定义的，其中 N 是集合中文档的总数，而 df_t 是出现 t 术语的文档数。出现术语的文档越少，此权重就越高。最低权重 1 分配给所有文档中出现的术语。通常很清楚什么才算是文件：在莎士比亚戏剧中，我们将使用剧本；处理诸如 Wikipedia 之类的百科全书文章时，该文档是 Wikipedia 页面； 在处理报纸文章时，文档是单篇文章。有时，您的语料库可能没有适当的文档划分，并且您可能需要自己将语料库分解为文档，以计算 idf。

因为在许多集合中有大量的文档，这个度量也常用对数函数来压缩，因此，反文档频率(idf)的最终定义如下：

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right) \tag{6.13}$$

以下是莎士比亚语料库中某些单词的一些 idf 值，范围从仅在一次演出中出现的极为丰富的单词诸如 Romeo，到仅出现在诸如 salad 或 Falstaff 等少数戏剧中的单词，再到那些非常普遍的单词诸如 fool 或完全不受歧视的单词，因为它们在全部 37 个演出中都表现得 good 或 sweet[3]。

---

[2] 或者可用以下替代方法：如果 count(t,d)> 0，则 $\text{tf}_{t,d}$ = 1 +$\log_{10}$count(t,d)；否则 $\text{tf}_{t,d}$ =0

[3] Sweet 是莎士比亚最喜欢的形容词之一，这一事实可能与 16 世纪初欧洲食谱中糖的使用增加有关（Jurafsky 2014)。

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

因此，文档 d 中单词 t 的 tf-idf 加权值 $w_{t,d}$ 将词频率 $tf_{t,d}$(由公式 6.11 或公式 6.12 定义)与公式 6.13 中的 idf_t 相结合：

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \tag{6.14}$$

图 6.9 将 tf-idf 加权应用于图 6.2 中的莎士比亚术语-文档矩阵(使用公式 6.12 的 tf 方程)。请注意，与单词 good 对应维度的 tf-idf 值现在都变为 0；由于此单词(good)出现在每个文档中，因此 tf-idf 算法导致它被忽略。同样，fool 这个词，在 37 部戏剧中的 36 部中出现过，它的权重非常低。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

图 6-9：tf-idf 加权术语-文档矩阵

图注：使用图 6.2 中的计数，在四个莎士比亚戏剧中，四个单词的 tf-idf 加权术语-文档矩阵。例如，《As You Like It》文档中的术语 wit 的 tf-idf=0.049 是 tf 与 idf 的乘积，其中：$tf = \log_{10}(20 + 1) = 1.322$，$idf = 0.037$。请注意，idf 加权消除了无处不在的单词 good 的重要性，并大大降低了几乎无处不在的单词 fool 的影响。

tf-idf 加权是在信息检索中对共现矩阵加权的方法，但在自然语言处理的许多其他方面也起作用。这也是一个很好的基准，简单的事情先尝试。我们将在 6.6 节中介绍其他权重，例如**正向逐点互信息(Positive Pointwise Mutual Information，PPMI)**。

6.6.逐点互信息(PMI)

当向量维度对应于单词而不是文档时，tf-idf 的替代加权函数 PPMI 用于术语-术语矩阵。PPMI 的直觉是，权衡两个词之间关联的最佳方法是要求(ask)这两个词在我们的语料库中同时出现的数量要比我们事先希望它们偶然出现的数量究竟高多少。**逐点互信息(Pointwise Mutual Information，PMI)**(Fano，1961)[4] 是自然语言处理中最重要的概念之一。与事件 x 和 y 是独立的相比，它可以衡量事件 x 和 y 发生的频率：

[4] PMI 基于两个随机变量 X 和 Y 之间的相互信息，定义为：

$$I(X,Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{6.15}$$

在术语混淆中，Fano 使用了短语“互信息”指代我们现在所说的“逐点互信息”，而短语“互信息的期望”则针对我们现在所说的“互信息”。

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{6.16}$$

目标词 w 和上下文词 c 之间的逐点互信息(Church 和 Hanks 1989，Church 和 Hanks 1990)被定义为：

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)} \tag{6.17}$$

其中，分子告诉我们，我们观察两个单词的联合(together)出现的频率(how offen)(假设我们使用 MLE 计算概率)。分母告诉我们，我们期望它们共现(co-occur)的频率(假设两个词各自独立出现)；回想一下，两个独立事件都发生的概率仅仅是两个事件的概率的乘积。因此，该比率给了我们一个估计值，即这两个词共现的次数比我们**偶然期望(expect by chance)**次数究竟要大多少。每当我们需要查找紧密关联的单词时，PMI 都是有用的工具。

PMI 值的范围从负到正无穷大。但是，负 PMI 值(意味着事情发生的频率比我们**偶然期望**的要低)往往是不可靠的，除非我们的语料库非常庞大。为了区分是否各自出现概率分别为 10^{-6} 的两个词同时出现的频率少于偶然出现的频率，我们需要确定两者共现的概率与 10^{-12} 显著不同，这种粒度将需要庞大的语料库。此外，尚不清楚是否有可能通过人类的判断来评估这种“无关性”的分数。因此，更常见的是使用 **PPMI**，将所有负 PMI 值替换为零(Church 和 Hanks 1989，Dagan 等人 1993； Niwa 和 Nitta 1994)[5]：

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0) \tag{6.18}$$

更正式地说，假设我们有一个共现矩阵 F，其中 W 行(单词)和 C 列(上下文)，其中 f_{ij} 给出单词 w_i 在上下文 c_j 中出现的次数。可以将其转换为 PPMI 矩阵，其中 $ppmi_{ij}$ 通过上下文 c_j 给出单词 w_i 的 PPMI 值，如下所示：

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \tag{6.19}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0) \tag{6.20}$$

让我们来看看 PPMI 的计算。我们将使用图 6.10，它重复了图 6.6 加上所有的计数边际值(marginals)，为了便于计算，我们假设这些是唯一重要的单词/上下文。

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

图 6-10：四个单词在五种语境下的共现数

图注：在维基百科语料库中，4 个单词在 5 种语境下的共现数，加上边际值，为了计算，假设其他单词/上下文无关要紧。

因此，例如我们可以计算 PPMI(w=information,c=data)，假设图 6.6 包含了所有相关的单词上下文/

[5] PPMI 还可以很好地解决使用零计数进行处理的问题，使用 0 替换 log(0)中的-∞。

维度，如下:

$$P(\text{w=information,c=data}) = \frac{3982}{11716} = .3399$$
$$P(\text{w=information}) = \frac{7703}{11716} = .6575$$
$$P(\text{c=data}) = \frac{5673}{11716} = .4842$$
$$\text{ppmi(information,data)} = \log 2(.3399/(.6575 * .4842)) = .0944$$

图 6.11 显示了根据图 6.10 中的计数计算得出的联合概率，图 6.12 显示了 PPMI 值。毫不奇怪，cherry、strawberry 与 pie、sugar 都高度相关，而 data 与 information 也有温和的关系。

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

图 6-11：联合概率

图注：用联合概率替换图 6.6 中的计数，显示外面的边际值。

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

图 6-12：PPMI 矩阵

图注：该图显示了单词和上下文单词之间的关联，该关联是根据图 6.11 中的计数计算得出的。请注意，大多数 0 PPMI 值都是 PMI 值为负的值； 例如 PMI(cherry，computer)= -6.7，这意味着 Cherry 和 computer 在维基百科上共现的频率比我们偶然期望的少，并且使用 PPMI 将负值替换为零。

PMI 存在偏重罕见事件的问题；非常罕见的词往往有很高的 PMI 值。减少这种对低频事件的偏差的一种方法是，使用不同的函数 $P_\alpha(c)$来略微改变 P(c)的计算，从而提高上下文单词出现的概率的α次方:

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0) \tag{6.21}$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha} \tag{6.22}$$

Levy 等人(2015 年)发现，将α设置为 0.75 可以改善在广泛任务上的嵌入效果(借鉴了公式 6.32 中所述的用于 skip-grams 的类似加权)。之所以可行，是因为将计数提高到α= 0.75 会增加分配给稀有上下文的概率，从而降低其 PMI(当 c 稀有时，$P_\alpha(c)>P(c)$)。

另一种可能的解决方案是拉普拉斯平滑：在计算 PMI 之前，将一个小的常数 k(通常为 0.1-3.0 的值)

添加到每个计数中，以缩小(打折)所有非零值。k 越大，非零计数的打折就越大。

6.7. TF-IDF 或 PPMI 矢量模型的应用

总之，到目前为止，我们已经描述的向量语义模型将目标词表示为向量，其维数对应于大的集合中的文档(术语-文档矩阵)或某个相邻窗口中的词计数(术语-术语矩阵)。每个维度中的值都是计数，通过 tf-idf(用于术语-文档矩阵)或 PPMI(用于术语-术语矩阵)加权，并且向量是稀疏的(因为大多数值为零)。

该模型通过获取两个单词 x 和 y 的 tf-idf 或 PPMI 向量的余弦来计算相似度； 高余弦值，高相似度。在加权函数之后，有时将整个模型称为 **tf-idf** 模型或 **PPMI** 模型。

tf-idf 含义模型通常用于文档功能诸如确定两个文档是否相似。我们通过获取文档中所有单词的向量并计算所有这些向量的**质心(centroid)**来表示文档。质心是均值的多维版本。向量集合的质心是单个向量，该向量与集合中每个向量的距离的平方和最小。给定 k 个词向量 $w_1,w_2,\ldots,w_k$，质心文档向量 d 为:

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k} \tag{6.23}$$

给定两个文档，我们可以计算它们的文档向量 d_1 和 d_2，并通过 $\cos(d_1, d_2)$估计两个文档之间的相似性。文档相似性对于各种应用程序也很有用。信息检索，剽窃检测，新闻推荐系统，甚至用于数字人文科学任务，例如比较文本的不同版本以查看彼此相似的内容。

PPMI 模型或 tf-idf 模型都可以用于计算单词相似度，例如查找单词释义，跟踪单词含义的变化或自动发现不同语料库中单词的含义等任务。例如，我们可以找到与任何目标单词 w 相似的 10 个最相似的单词(通过计算 w 与其余 V-1 个单词中的每一个单词之间的余弦值，进行排序，然后查看前 10 位)。

6.8. Word2vec

在前面的部分中，我们看到了如何将单词表示为稀疏的长向量，其维数对应于词汇表中的单词或集合中的文档。现在，我们介绍一个更强大的单词表示形式：**嵌入**，短密集向量。与迄今为止我们看到的向量不同，嵌入是**短的(short)**，维数 d 的范围是 50-1000，而不是很大的词汇量|V|或我们看到的文件 D 的数量。这些 d 维没有一个清晰的解释。向量是**密集的(dense)**:向量不是稀疏的实体、大部分为零的计数或计数的函数，而是可以为负的实数值。

事实证明，稠密向量在每个 NLP 任务中比稀疏向量更好地工作。尽管我们不完全了解造成这种情况的所有原因，但我们有一些直觉。将单词表示为 300 维密集向量需要我们的分类器学习的权重比我们将单词表示为 50000 维向量要少得多，而且较小的参数空间可能有助于泛化和避免过拟合。密集向量在捕获同义词方面也可以做得更好。例如，在稀疏向量表示中，同义词 car 和 automobile 的维数是截然不同且不相关的;因此，稀疏向量可能无法捕捉以 car 为邻居的单词和以 automobile 为邻居的单词之间的相似性。

在本节中，我们介绍一种计算嵌入的方法：具有负采样的 **skip-gram**，有时也称为 **SGNS(skip-gram with negative sampling)**。skip-gram 算法是软件包 **word2vec** 中的两种算法之一，因此有时将该算法简称为 word2vec(Mikolov 等人 2013；Mikolov 等人 2013a)。word2vec 方法可以快速、有效地进行训练，并且可以通过代码和预训练的嵌入轻松在线获得。Word2vec 嵌入是**静态嵌入(static embeddings)**，这意味着该方法为词汇表中的每个单词学习一个固定的嵌入。在第 10 章中，我们将介绍学习动态上下文嵌入的方法，例如流行的 **BERT 或 ELMO** 表示，其中每个单词的向量在不同的上下文中是不同的。

word2vec 的直觉是，与其计算每个单词 w 在例如 apricot 附近出现的频率，不如计算一个分类器来执行二进制预测任务：“单词 w 是否可能在 apricot 附近出现？”我们实际上并不关心此预测任务，取而代之的是，我们将学习到的分类器权重(weights)作为词嵌入。

革命性的直觉是，我们可以仅使用运行文本作为此类分类器的隐式监督训练数据。出现在目标词 apricot 附近的词 c，成为问题“词 c 可能出现在 apricot 附近”的黄金“正确答案”。这种方法通常称为**自我监督(self-supervision)**，避免了需要任何手工标记的监督信号。这个概念第一次被提出在神经语言建模的任务中，那时 Bengio 等人(2003)和 Collobert 等人(2011)指出，神经语言模型(一种神经网络，它学会了从

先前单词中预测下一个单词)可以仅将运行文本中的下一个单词用作其监督信号，并可以用于学习每个单词的嵌入表示，作为执行此预测任务的一部分。

在下一章中，我们将介绍如何做神经网络，但是 word2vec 在两个方面比神经网络语言模型简单得多。首先，word2vec 简化了任务(使其成为二进制分类而不是单词预测)。其次，word2vec 简化了体系结构(训练逻辑回归分类器，而不是使用需要更复杂训练算法的具有隐藏层的多层神经网络)。skip-gram 的直觉是：

1.将目标词和邻近的上下文词视为肯定示例。

2.随机抽样词典中的其他单词以获得否定样本。

3.使用逻辑回归训练分类器来区分这两种情况。

4.使用学习到的权重作为嵌入。

6.8.1. 分类器

让我们首先考虑分类任务，然后转向训练方法。想象一下以下句子，目标词是 apricot，假设我们使用的是±2 个上下文词的窗口：

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
              c1         c2    w     c3        c4
```

我们的目标是训练一个分类器，以便在给定目标词 w 的元组(w，c)与候选上下文词 c(例如(apricot, jam)或许(apricot，aardvark))配对的情况下，它将返回 c 是真实上下文单词的概率(对于 jam 为 true，对于 aardvark 为 false)：

$$P(+|w,c) \tag{6.24}$$

这个概率(单词 c 不是 w 的真实上下文单词)仅为 1 减去公式 6.24：

$$P(-|w,c)=1-P(+|w,c) \tag{6.25}$$

分类器如何计算概率 P？ skip-gram 模型的直觉是将这种概率基于嵌入相似性：如果单词的嵌入与目标嵌入相似，则单词很可能出现在目标附近。要计算这些密集嵌入之间的相似度，我们要凭直觉，即两个向量如果具有高**点积(dot product)**(毕竟，余弦只是一个归一化的点积)是相似的。换一种说法：

$$\text{Similarity}(w,c)\approx c\cdot w \tag{6.26}$$

点积 c·w 不是概率，只是-∞到∞之间的数字(由于 word2vec 嵌入中的元素可以为负，因此点积可以为负)。要将点积转化为概率，我们将使用逻辑或 sigmoid 函数σ(x)，这是逻辑回归的基本核心：

$$\sigma(x)=\frac{1}{1+\exp(-x)} \tag{6.27}$$

我们将单词 c 作为目标单词 w 的真实上下文单词的概率建模为：

$$P(+|w,c) = \sigma(c\cdot w)=\frac{1}{1+\exp(-c\cdot w)} \tag{6.28}$$

sigmoid 函数返回一个介于 0 到 1 之间的数字，但是，若要使其成为一个概率，则还需要两个可能事件(c 是上下文词，c 不是上下文词)的总概率之和为 1。因此，我们估计单词 c 不是 w 的真实上下文单词的概率为：

$$\begin{aligned} P(-|w,c) &= 1-P(+|w,c) \\ &= \sigma(-c\cdot w)=\frac{1}{1+\exp(c\cdot w)} \end{aligned} \tag{6.29}$$

公式 6.28 为我们提供了一个单词的概率，但是窗口中有许多上下文单词。Skip-gram 简化了假设，即

所有上下文词都是独立的，从而我们可以使它们的概率相乘：

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(-c_i \cdot w) \tag{6.30}$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(-c_i \cdot w) \tag{6.31}$$

总之，skip-gram 训练了一个概率分类器，给定一个测试目标词 w 及其上下文窗口(含有 L 个词 $c_{1:L}$)，skip-gram 基于此上下文窗口与目标词的相似度来分配概率。概率基于将逻辑(S 型)函数应用于目标词与每个上下文词的嵌入的点积。要计算此概率，我们只需要为词汇表中的每个目标单词和上下文单词嵌入。

图 6.13 显示了我们需要的参数的直观信息。Skip-gram 实际上为每个单词存储两个嵌入：一个嵌入作为目标词，一个嵌入作为上下文。因此，我们需要学习的参数是两个矩阵 W 和 C，每个矩阵都包含一个嵌入，该嵌入包含词汇表 V 中的|V|个单词[6]。现在让我们学习这些嵌入内容(这是首先训练此分类器的真正目标)。

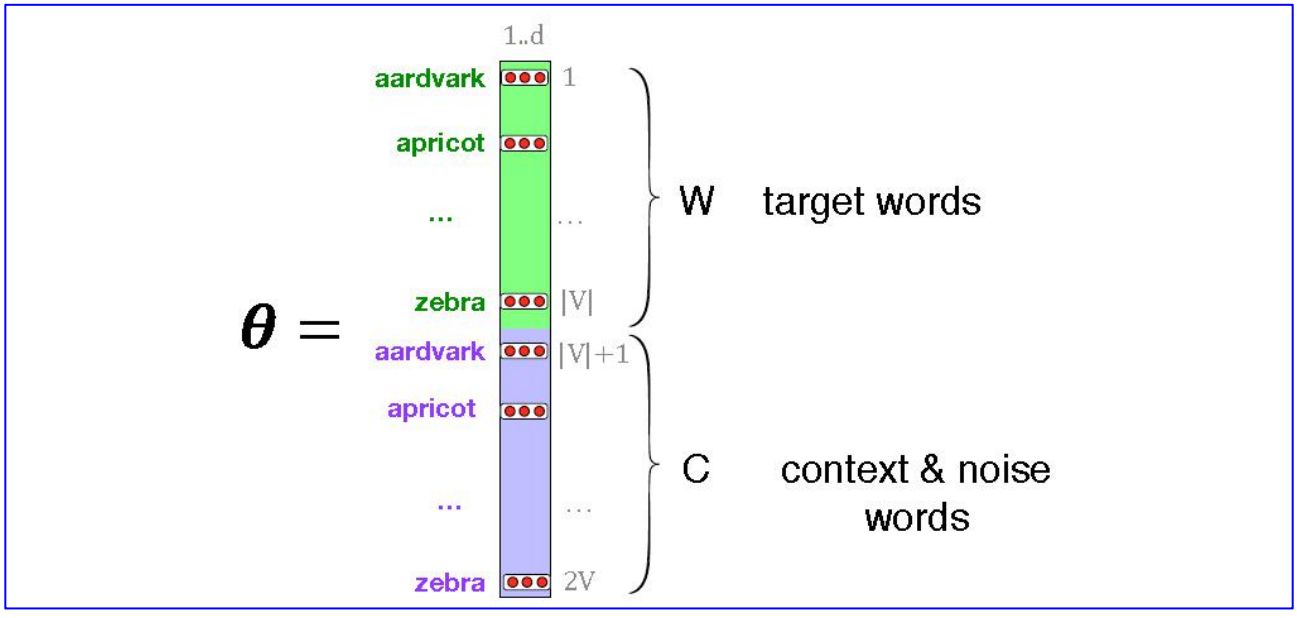


图 6-13：skip-gram 模型学习到的嵌入

图注：该算法为每个单词存储两个嵌入:目标嵌入(有时称为输入嵌入)和上下文嵌入(有时称为输出嵌入)。因此，该算法学习的参数 θ 是一个由两个|V|维向量组成的矩阵，每个向量维数为 d，由两个矩阵(目标嵌入的 W 矩阵、上下文+噪声嵌入的 C 矩阵)组合而成。

6.8.2. 学习 skip-gram 嵌入

Skip-gram 通过从随机的嵌入向量开始学习嵌入，然后迭代地移动每个单词 w 的嵌入，更像是文本中附近出现的单词的嵌入，而不像是文本中附近没有出现的单词的嵌入。让我们从单个训练数据开始考虑:

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
              c1         c2    w     c3         c4
```

此示例在 L=±2 窗口中具有目标词 w(apricot)和 4 个上下文词，从而产生 4 个正的训练实例(在下面左侧)：

[6]原则上，目标矩阵和上下文矩阵可以使用不同的词汇表，但是我们将假设一个共享词汇表 V 进行简化。

positive examples +		negative examples -			
w	c_{pos}	w	c_{neg}	w	c_{neg}
apricot	tablespoon	apricot	aardvark	apricot	seven
apricot	of	apricot	my	apricot	forever
apricot	jam	apricot	where	apricot	dear
apricot	a	apricot	coaxial	apricot	if

为了训练二进制分类器，我们还需要负样本。实际上，带有负采样的 skip-gram(SGNS)使用的负样本比正样本更多(它们之间的比率由参数 k 设置)。因此，对于每个(w，c_{pos})训练的正样本，我们将创建 k 个负样本，每个负样本均由目标词 w 加上一个“噪声词”c_{neg} 组成。噪声词是词典中的随机词，被限制为不是目标词 w。右上方显示了其中 k = 2 的设置，因此，对于每个正样本(w，c_{pos})，我们在负训练集之中有两个负样本。

噪声词的选择依据其加权的 unigram 概率 $p_\alpha(w)$，其中α为权值。如果我们根据未加权的频率 p(w)进行采样，这就意味着：对于 unigram 概率 p(“the”)，我们将选择 the 作为噪声词；对于 unigram 概率 p(“aardvark”)，我们将选择 aardvark 作为噪声词，依此类推。但在实践中，通常设置α =0.75，即使用权重 $p^{3/4}(w)$:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha} \tag{6.32}$$

设置α =0.75 会有更好的性能，因为它会给罕见的噪声单词更高的概率:对于罕见单词，$P_\alpha(w) > P(w)$。为了说明这种直觉，我们可以计算出两个事件(P(a) =0.99 和 P(b) =0.01)的概率:

$$\begin{aligned} P_\alpha(a) &= \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97 \\ P_\alpha(b) &= \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03 \end{aligned} \tag{6.33}$$

给定正、负训练实例集和初始嵌入集，学习算法的目标是将这些嵌入调整到:

•最大化正采样中的一对词(w，c_{pos})的相似度；

•最小化负采样中的一对词(w，c_{neg})的相似度。

如果我们考虑一个单词/上下文对(w，c_{pos})及其 k 个噪声词 $c_{neg\,1}$ ... $c_{neg\,k}$，那么我们可以将这两个目标表示为下面的损失函数 L 最小化(因此是负号-)：

$$\begin{aligned} L_{CE} &= -\log\left[P(+|w, c_{pos}) \prod_{i=1}^{k} P(-|w, c_{neg_i})\right] \\ &= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log P(-|w, c_{neg_i})\right] \\ &= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log\left(1 - P(+|w, c_{neg_i})\right)\right] \\ &= -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right] \end{aligned} \tag{6.34}$$

其中，第一项表示我们希望分类器为真实的上下文词 c_{pos}(是邻居)分配高概率，而第二项表示我们希望给每个噪声词 $c_{neg\,i}$(非邻居)分配高概率。因为我们假设独立性，所以所有概率全部相乘。

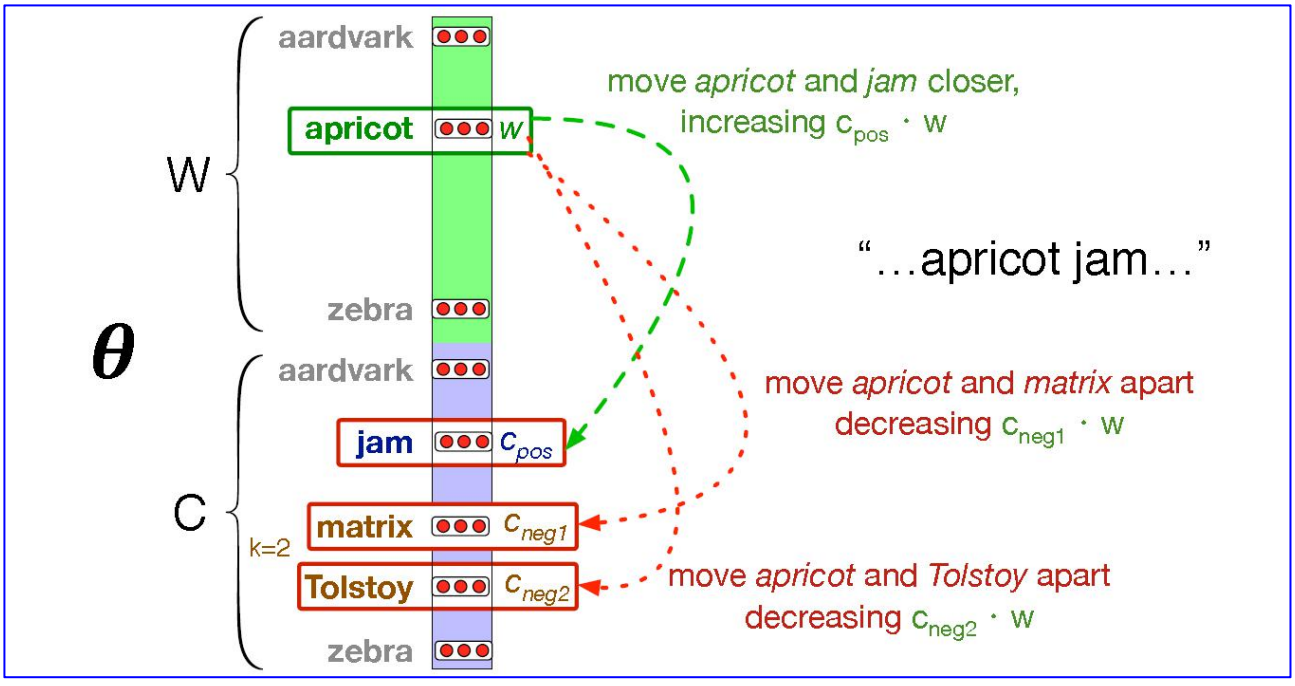


图 6-14：梯度下降法的直观感受

图注：skip-gram 模型试图移动嵌入，这样目标嵌入(即 apricot)更接近(有更高的点积值)邻居单词的上下文嵌入(即 jam)，而远离(有更低的点积值)噪声词(非邻居词)的上下文嵌入(即 Tolstoy 和 matrix)。

也就是说，我们希望最大化单词与实际上下文的邻居单词的点积，并最小化单词与 k 个负抽样的非邻居单词的点积。

我们使用随机梯度下降使损失函数最小化。图 6.14 显示了一个学习步骤的直观感受。

为了得到梯度，我们需要在公式 6.34 中对不同的嵌入量求导。它的导数如下(我们把证明留到本章最后做练习):

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w \tag{6.35}$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w \tag{6.36}$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{neg_i} \cdot w)]c_{neg_i} \tag{6.37}$$

随机梯度下降中从时间步长 t 到 t +1 的更新方程为:

$$c_{pos}^{t+1} = c_{pos}^{t} - \eta[\sigma(c_{pos}^{t} \cdot w) - 1]w \tag{6.38}$$

$$c_{neg}^{t+1} = c_{neg}^{t} - \eta[\sigma(c_{neg}^{t} \cdot w)]w \tag{6.39}$$

$$w^{t+1} = w^{t} - \eta[\sigma(c_{pos} \cdot w^{t}) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{neg_i} \cdot w^{t})]c_{neg_i} \tag{6.40}$$

那么，就像在逻辑回归中一样，学习算法从随机初始化 W 和 C 矩阵开始，然后使用梯度下降来移动 W 和 C，通过在公式 6.39、公式 6.40 中更新来最大化公式 6.34 中的目标。

回想一下，skip-gram 模型为每个单词 i 学习两个单独的嵌入：**目标嵌入(target embedding)**w_i 和**上下文嵌入(context embedding)**c_i，分别存储在两个矩阵中，即**目标矩阵(target matrix)**W 和**上下文矩阵(context matrix)**C。通常将它们加在一起，用向量 $w_i + c_i$ 表示单词 i。或者，我们可以丢弃 C 矩阵，仅用向量 w_i 表示每个单词 i。

与简单的基于计数的方法(如 tf-idf)一样，上下文窗口的大小 L 影响 skip-gram 嵌入的性能，并且很多实验经常在 devset 上调整参数 L。

6.8.3. 其他类型的静态嵌入

静态嵌入有很多种。word2vec 的扩展，即快速文本(fasttext)(Bojanowski 等人，2017)，通过使用子词模型来处理具有丰富形态的语言中的未知词和稀疏性。快速文本中的每个单词都表示为自己，再加上一袋 n-gram 成分，每个单词都带有特殊的边界符号< >。例如，当 n=3 时，单词 where 将由序列<where>加上 n-gram 字符来表示[7]：

<wh, whe, her, ere, re>

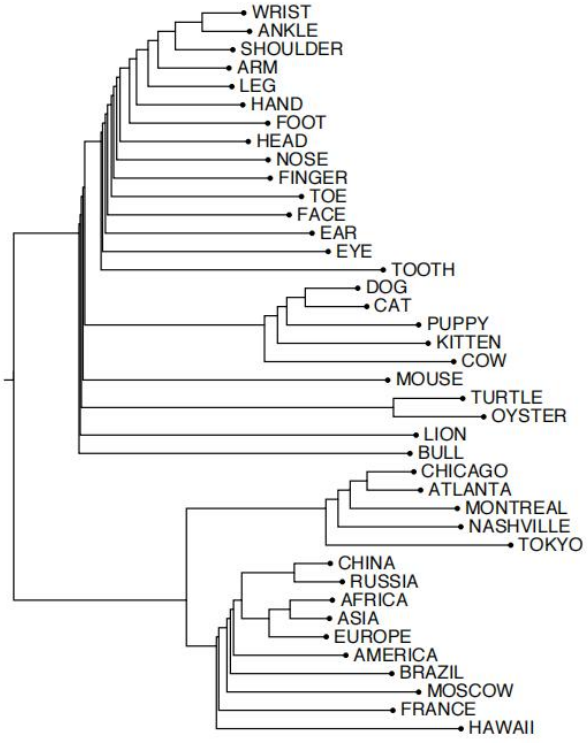
然后，为每个 n-gram 成分学习一个 skip-gram 嵌入，单词 where 由其 n-gram 成分的所有嵌入之和表示。快速文本开源库，包括针对 157 种语言的预训练嵌入，可从 https://fasttext.cc 获得。

除 word2vec 之外，使用最广泛的静态嵌入模型是 GloVe(Pennington 等人，2014)，是 Global Vectors 的缩写，因为该模型基于捕获全局语料库统计数据。GloVe 基于单词-单词共现矩阵的概率比率，结合基于计数的模型(如 PPMI)的直觉，同时还捕获了诸如 word2vec 之类的方法所使用的线性结构。

事实证明，像 word2vec 之类的密集嵌入实际上与诸如 PPMI 之类的稀疏嵌入有着良好的数学关系，其中 word2vec 可以看作是隐式优化了 PPMI 矩阵的移位版本(Levy 和 Goldberg，2014c)。

6.9. 可视化嵌入

> "I see well in many dimensions as long as the dimensions are around two."
> "我能很好地看很多维度，只要维度在 2 左右。"
>
> The late economist Martin Shubik



可视化嵌入是一个重要目标，它有助于理解、应用和改进词义模型。但是我们如何可视化 100 维向量？

可视化嵌入空间中的单词 w 的含义的最简单方法，是列出与 w 最相似的单词，即将词汇表中所有单词的向量按其余弦与 w 的向量进行排序。例如，使用 GloVe 嵌入最接近青蛙(frog)的 7 个词是：frogs, toad, litoria, leptodactylidae, rana, lizard,和 eleutherodactylus (Pennington 等人，2014)。

另一可视化方法是使用聚类算法来显示在嵌入空间中单词与其他单词相似的层次表示。左边的没有标题的图片使用一些嵌入名词的向量的层次聚类作为可视化方法(Rohde 等人，2006)。

但是，最常见的可视化方法可能是将单词的 100 个维度向下投影为 2 个维度。图 6.1 显示了一种这样的可视化效果，图 6.16 使用了一种称为 t-SNE 的投影方法(van der Maaten and Hinton，2008)。

[7] 译者注：左边的"<wh"和右边的"re>"都是三元语法，因为"<"和">"都算一个字符。

6.10. 嵌入的语义属性

在本节中，我们将简要总结一些已经研究过的嵌入语义特性。

不同类型的相似性或关联性：一个向量语义模型(与稀疏 tf-idf 向量和密集 word2vec 向量都相关)的参数是上下文窗口(用于收集计数)的大小。这通常是在目标单词的每一侧都有 1 到 10 个单词(对应总共 2-20 个单词的上下文)。(窗口大小的)选择依赖于表示的目标。较短的上下文窗口往往会导致更具有语法性的表示，因为信息来自邻近的单词。当从短的上下文窗口计算向量时，与目标词 w 相似度最高的词往往是语义相似、词类相同的词。当从长的上下文窗口计算向量时，与目标词 w 余弦值最高的词往往是主题相关、但不相似的词。

例如，Levy 和 Goldberg (2014a)显示，使用窗口值为±2 的 skip-gram，与 Hogwarts(来自《哈利波特》系列)最相似的单词是其他虚构学校的名字:Sunnydale(来自《吸血鬼猎人巴菲》)或 Evernight(来自吸血鬼系列)。当窗口值为±5，与 Hogwarts 最相似的单词是与《哈利波特》系列相关的主题词:Dumbledore, Malfoy, 和 half-blood。

区分单词之间的两种相似性或关联性通常也很有用(Schutze 和 Pedersen，1993)。如果两个词通常彼此相邻，则它们具有**一阶共现(first-order co-occurrence)**(有时称为**句法关联(syntagmatic association)**)。因此写(wrote)就是书(book)和诗(poem)的一阶联想。如果两个单词具有相似的邻居，则它们具有**二阶共现(second-order co-occurrence)**(有时称为**范式关联(paradigmatic association)**)。因此，“写(wrote)”是“说(said)”或“评论(remarked)”等词的二阶联想。

类比/关系相似性：嵌入的另一个语义特性是它们捕获关系含义的能力。在一个重要的早期认知向量空间模型中，Rumelhart 和 Abrahamson(1973)提出了平行四边形模型来解决简单的类比问题，形式为“ a is to b as a* is to what？ ”。在此类问题中，系统会遇到类似“ apple：tree :: grape :?”这样的问题，即“ apple is to tree as grape is to _____”，并且必须填充单词“ vine”。在平行四边形模型中，如图 6.15 所示，从单词“ apple”到单词“ tree”(=$\overrightarrow{apple}$-$\overrightarrow{tree}$)的向量被添加到 grape 的向量($\overrightarrow{grape}$)：最接近该点的词被返回。

在稀疏嵌入的早期工作中，学者们表明，稀疏向量含义模型可以解决此类比喻问题(Turney 和 Littman，2005)，但平行四边形方法因其在 word2vec 或 GloVe 向量上的成功而受到了现代的关注(Mikolov 等人 2013b, Levy 和 Goldberg 2014b, Pennington 等人 2014)。

例如，“$\overrightarrow{king}$ - $\overrightarrow{man}$ + $\overrightarrow{woman}$”表达式的结果是接近“$\overrightarrow{queen}$”的向量。同样，“$\overrightarrow{Paris}$ - $\overrightarrow{France}$ + $\overrightarrow{Italy}$”得出的向量接近“$\overrightarrow{Rome}$”。因此，嵌入模型似乎正在提取诸如 MALE-FEMALE 或 CAPITAL-CITY-OF 或 COMPARATIVE / SUPERLATIVE 之类的关系的表示，如图 6.16(来自 GloVe)所示。

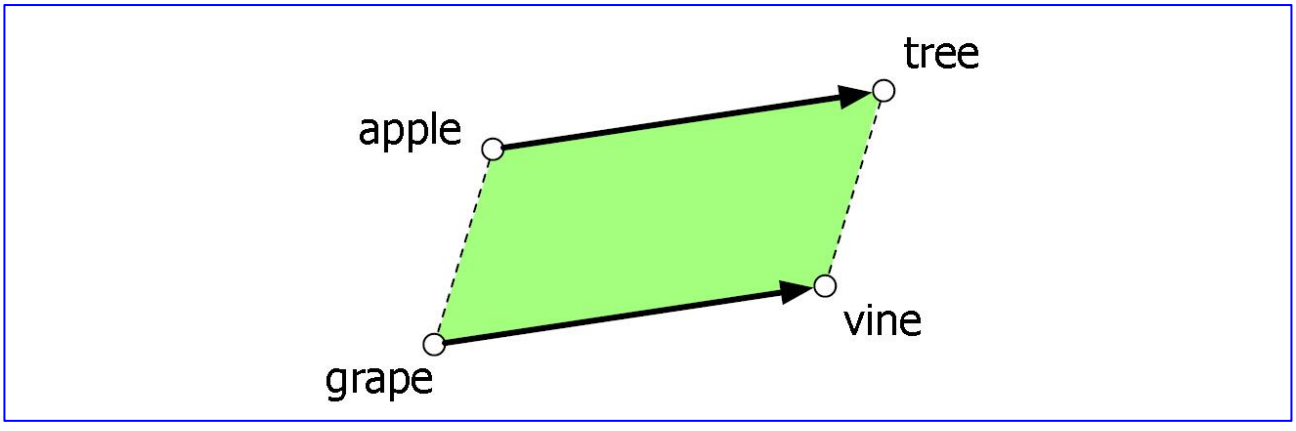


图 6-15：类比问题的平行四边形模型

图注：类比问题的平行四边形模型(Rumelhart and Abrahamson, 1973)：通过$\overrightarrow{apple}$减$\overrightarrow{tree}$加$\overrightarrow{grape}$，就可定位$\overrightarrow{vine}$。

对于 a：b :: a*：b*问题，意味着算法：已知 a、b 和 a*，要求必须找到 b*；因此，平行四边形方法为：

$$\hat{b}^* = \underset{x}{\text{argmax}}\ \text{distance}(x, a^* - a + b) \tag{6.41}$$

其中：距离函数定义为余弦距离或欧几里得距离。

有一些注意事项。例如，平行四边形算法在 word2vec 或 GloVe 嵌入空间中返回的最接近值通常实际上不是 b*，而是 3 个输入单词之一或它们的词素变体(即 cherry：red :: potato：x 返回 potato 或者 potatoes 而不是 brown)，因此必须明确排除这些内容。此外，如果任务涉及频繁的单词、小距离和某些关系(例如，国家与其首都的关系，或者动词/名词与其变体形式的关系)进行嵌入，则嵌入空间的效果会很好；但是，如果采用平行四边形方法进行嵌入，则对其他关系的效果不太好(Linzen 2016，Gladkova 等人 2016，Ethayarajh 等人 2019a)，甚至 Peterson 等人(2020)认为，平行四边形方法通常太简单了，以至于不能对人类形成这种类比的认知过程进行建模。

6.10.1. 嵌入和历史语义

嵌入是一种有用的工具，通过计算多个嵌入空间(每个嵌入空间来自特定时期的文本)来研究意义如何随时间变化。例如，图 6.17 显示了过去两个世纪英语单词含义变化的可视化，通过从历史语料库(如谷歌 N-grams, Lin 等人 2012b)和美国历史英语语料库(Davies, 2012)中为每十年建立单独的嵌入空间来计算。

6.11. 偏见和嵌入

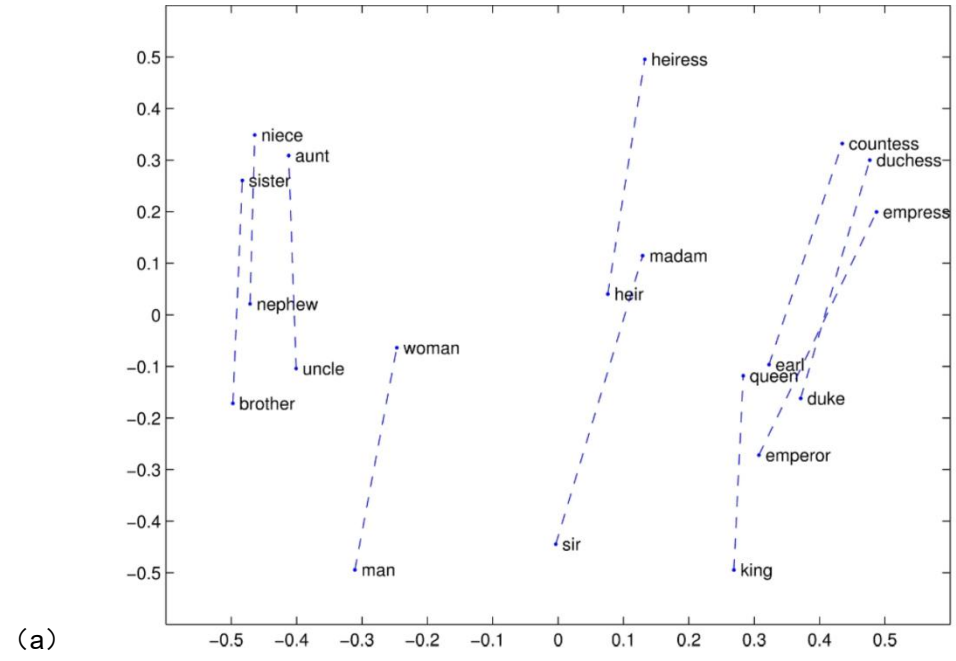
除了他们从文本中学习单词含义的能力外，嵌入，唉，也复制了潜在的文本中的隐性偏见和刻板印象。如前一节所示，嵌入可以大致模拟关系相似性:'queen'作为最接近'king'-'man'+'woman' 暗示了类似的含义:man:woman::king:queen。但是这些同样的内嵌类比也表现出性别刻板印象。例如，Bolukbasi 等人(2016)发现，在新闻文本上训练的 word2vec 嵌入中，与"男人"-"计算机程序员"+"女人"最接近的职业是"家庭主妇"，嵌入同样表明，"父亲"是"医生"，就像"母亲"是"护士"。这可能导致 Crawford(2017)和 Blodgett 等人(2020)所说的**分配伤害(allocational harm)**，即系统不公平地将资源(工作或信用)分配给不同的群体。例如，使用嵌入算法作为寻找潜在程序员或医生的搜索的一部分，可能会错误地减少带有女性名字的文档的份量。

事实证明，嵌入不仅反映了其输入的统计信息，而且还**放大(amplify)**了偏见。性别术语在嵌入空间中的性别比输入文本统计中的性别(Zhao 等人 2017，Ethayarajh 等人 2019b，Jia 等人 2020)更多，而且偏见比实际劳动就业统计数据中的夸张程度更大(Garg 等人 2018)。

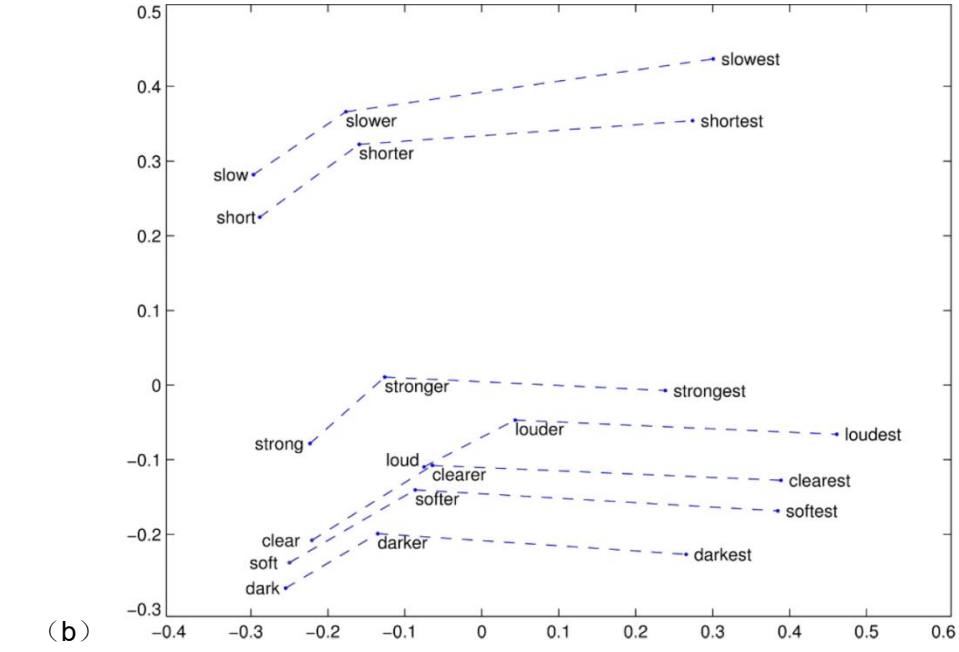
嵌入也编码了内隐联想，这是人类推理的一种属性。内隐联想测试(Greenwald 等人，1998)通过测量人们在不同类别中标记词语的潜伏期的差异，来衡量人们在概念(如"花"或"昆虫")和属性(如"愉快"和"不愉快")之间的联系[8]。使用这种方法，已证明美国民众将非裔美国人的名字与不愉快的单词联系在一起(比欧裔美国人的名字更多)，男性名字与数学联系在一起，女性名字与艺术联系在一起，而老年人的名字则与讨厌的单词联系在一起(Greenwald 等人 1998；Nosek 等人 2002a，Nosek 等人 2002b)。Caliskan 等人(2017)使用 GloVe 向量和余弦相似度而不是人类潜伏期复制了所有这些隐式关联的发现。例如，非裔美国人的名字如"Leroy"和"Shaniqua"中含有更多令人不愉快的词，而欧裔美国人的名字("Brad"，"Greg"，"Courtney")中含有更多令人愉快的词。这些与嵌入有关的问题是**表述伤害(representational harm)**的一个例子(Crawford 2017, Blodgett 等人 2020)，这是一种由系统贬低甚至忽视某些社会群体所造成的伤害。因此，任何使用词语情感的嵌入式感知算法都可能加剧对非裔美国人的偏见。

最近的研究集中在尝试消除这些偏见的方法上，例如通过开发一种嵌入空间的转换来消除性别刻板印象，但保留定义的性别(Bolukbasi 等人 2016，Zhao 等人 2017)或改变训练程序(Zhao 等人，2018b)。然而，尽管这些类型的**去偏见(debiasing)**可以减少嵌入中的偏见，但它们并不能消除这种偏见(Gonen debiasing 和 Goldberg，2019)，这仍然是一个未解决的问题。

[8] 粗略地说，如果人类将"花朵"与"不愉快"联系起来，将"昆虫"与"不愉快"联系起来，那么当他们被指示：按下一个绿色按钮表示"花朵"（雏菊，鸢尾花，丁香花)和"令人愉快的单词"（爱，笑声，愉悦)；按红色按钮表示"昆虫"（跳蚤，蜘蛛，蚊子)和"不愉快的单词"（虐待，仇恨，丑陋)时，他们的行动要快于如下不协调的情况：一个红色的按钮用于表示"花朵"和"不愉快的单词"；一个绿色的按钮用于表示"昆虫"和"令人愉悦的单词"。



（a）



（b）

图 6-16：GloVe 向量空间的关系属性

图注：(a) “$\overrightarrow{king}-\overrightarrow{man}+\overrightarrow{woman}$”接近“$\overrightarrow{queen}$”；(b)偏移似乎捕获了比较形态和最高级形态(Pennington 等人，2014)。

历史的嵌入也被用来衡量过去的偏见。Garg 等人(2018)使用历史文本中的嵌入项来衡量 20 世纪职业嵌入项和不同种族或性别名称嵌入项之间的关联(例如，女性姓名与男性姓名与职业词如“图书管理员”或“木匠”的相对余弦相似性)。他们发现，余弦与这些职业中女性或种族群体的经历百分比相关。历史嵌入也重复了关于种族刻板印象的旧调查;1933 年的实验参与者倾向于将“勤劳”或“迷信”等形容词与中国民族等联系起来，这与 20 世纪 30 年代文本训练的嵌入词与中国姓氏之间的余弦值有关。研究人员还记录了历史上的性别偏见，比如与能力相关的形容词(“聪明”、“明智”、“体贴”、“足智多智”)的男性词汇比女性词汇的余弦值更高，并且表明这种偏见自 1960 年以来一直在慢慢减少。我们将在后面的章节中回到这个关于偏见在自然语言处理中的作用的问题。

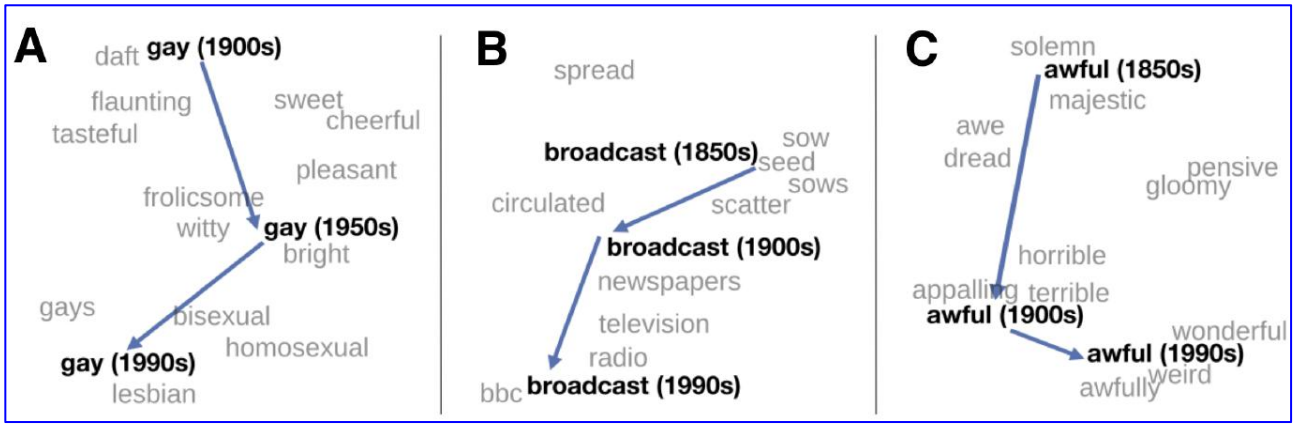


图 6-17：使用 word2vec 进行 t-SNE 可视化

图注：使用 word2vec 向量对 3 个英语单词的语义变化进行 t-SNE 可视化。每个词的现代意义和灰色上下文词都是根据最新(现代)时间点嵌入空间计算得出的。较早的点是根据较早的历史嵌入空间计算得出的。可视化显示了“同性恋”(gay)一词的变化，从与“开朗(cheerful)”或“轻快(frolicsome)”相关的含义到指同性恋(homosexuality)，广播(broadcast)的现代“传播(transmission)”意义由最初的“播种(sowing seeds)”意义发展，“awful”一词由“充满敬畏(full of awe)的含义转贬为“可怕的或骇人听闻的(terrible or appalling)”的含义(Hamilton 等人，2016b)。

6.12. 评估向量模型

向量模型最重要的评估指标是任务的外在评估，即在一个自然语言处理任务中使用向量，并观察这是否比其他模型提高性能。

尽管如此，内在评估是有用的。最常见的度量标准是测试他们在相似度上的表现，计算算法的单词相似度分数和人类赋予的单词相似度评级之间的相关性。WordSim-353 (Finkelstein 等人，2002)是一个常用的评分集合，从 0 到 10，用于 353 个名词对;例如(飞机，汽车)的平均分是 5.77 分。SimLex-999 (Hill 等人，2015)是一个更困难的数据集，它量化了相似性(杯子，马克杯)而不是相关性(杯子，咖啡)，并包括具体和抽象的形容词、名词和动词对。托福(TOEFL)考试数据集是一个 80 道题的集合，每道题包含一个目标词和 4 个额外的单词选择;任务是选择哪个是正确的同义词，例如:Levied is closest in meaning to: imposed, believed, requested, correlated(Landauer 和 Dumais, 1997)。所有这些数据集都提供没有上下文的单词。

更现实一点的是包含上下文的内在相似性任务。斯坦福上下文词相似度(SCWS)数据集(Huang 等人，2012)和上下文词(WiC)数据集(Pilehvar 和 camjo - collados, 2019)提供了更丰富的评估场景。SCWS 给出人工判断(在它们的句子上下文里的 2,003 对单词中)，而 WiC 给出目标单词(在两个意义相同或不同的句子上下文中)，参见第 18.5.3 节。**语义文本相似性(semantic textual similarity)**任务(Agirre 等人 2012；Agirre 等人 2015)评估句子级相似度算法的性能，该算法由一组成对的句子组成，每对句子均带有人类标记的相似性评分。

用于评估的另一个任务是类比任务，在第 6.10 节讨论，其中系统必须解决形式为 “a is to b as a* is to b*”的情况，给定 a、b 和 a*，必须找到 b*(Turney 和 Littman，2005)。为此任务创建了许多元组(Mikolov 等人 2013，Mikolov 等人 2013b，Gladkova 等人 2016)，涵盖了形态(city:cities::child:children)，词汇关系(leg:table::spout::teapot)和百科全书关系(Beijing:China::Dublin:Ireland)，以及摘自 SemEval-2012 Task 2 数据集中的 79 种不同关系(Jurgens 等人，2012)。

所有的嵌入算法都有内在的可变性。例如，由于初始化的随机性和随机的负抽样，像 word2vec 这样的算法即使在同一个数据集上也可能产生不同的结果，并且一个集合中的单个文档可能会强烈地影响结果的嵌入(Hellrich 和 Hahn 2016, Antoniak 和 Mimno 2018)。因此，当嵌入用于研究特定语料库中的单词关联时，最好的做法是通过引导(bootstrap)抽样对文档训练多个嵌入，并对结果进行平均(Antoniak 和 Mimno, 2018)。

6.13. 总结

•在向量语义中，单词被建模为向量，即高维空间中的一个点，也称为**嵌入**。在本章中，我们集中于**静态嵌入**，在每个单词中都将其映射到固定的嵌入。

•向量语义模型分为两类：**稀疏**和**密集**。在稀疏模型中，每个维度对应于词汇表 V 中的一个单词，而单元格是**共现计数**的函数。**术语-文档**矩阵在词汇表中的每个单词(**术语**)都有一行，在每个文档中都有一列。**单词-上下文**或**术语-术语**矩阵有一行用于词汇表中的每个(目标)单词，有一列用于词汇表中的每个上下文术语。稀疏加权通常有两种：根据每个单元格的词频和逆文档频率的 **tf-idf** 加权；针对单词-上下文矩阵的 **PPMI**(逐点正互信息)加权。

•密集矢量模型的维数为 50–1000。**Word2vec** 算法(例如 skip-gram)是计算密集嵌入的一种流行方法。**Skip-gram** 训练逻辑回归分类器，以计算出两个词“可能在文本附近出现”的概率。这个概率是从两个单词的嵌入之间的点积计算出来的。

•Skip-gram 使用随机梯度下降训练分类器，其方法是学习嵌入。该嵌入具有两个点积：一个是邻近词的高点积，另一个是噪声词的低点积。

•其他重要的嵌入算法包括 **GloVe**，一种基于词共现概率比率的方法。

•无论是使用稀疏向量还是密集向量，单词和文档的相似性都是通过向量之间的**点积**函数来计算的。两个向量的余弦—一种归一化的点积—是这种度量中最流行的。

6.14. 文献和历史说明

向量语义学的概念产生于 20 世纪 50 年代三个不同领域的研究:语言学、心理学和计算机科学，它们都对模型的基本方面做出了贡献。

在 20 世纪 50 年代的语言学理论中，含义与词语在语境中的分布有关的观点被广泛传播，分布论者如 Zellig Harris、Martin Joos 和 J. R. Firth，符号学家如 Thomas Sebeok。正如 Joos(1950)所说，

> the linguist's “meaning” of a morpheme. . . is by definition
>
> the set of conditional probabilities of its occurrence in context with all other morphemes.
>
> (语言学家的语素的“含义”…被定义为：它在与所有其他语素的上下文中发生的条件概率集。)

一个单词的含义可以作为一个多维语义空间中的一个点来建模的想法来自于像 Charles E. Osgood 这样的心理学家，他一直在研究人们是如何通过将快乐/悲伤或硬/软等尺度分配给单词的含义来做出反应的。

Osgood 等人(1957)提出，一般来说，一个词的含义可以建模为多维欧氏空间中的一个点，两个词之间的含义相似性可以建模为这些点在空间中的距离。

1950 年代和 1960 年代初期的最终知识渊源是当时称为机械索引的领域，现在称为信息检索。在被称为信息检索的向量空间模型(Salton 1971，Sparck Jones 1986)中，研究人员展示了用向量来定义词的含义的新方法(Switzer，1965)，并基于测度重新定义了词相似性的方法。互信息(Giuliano，1965)和 idf(Sparck Jones，1972)之类的词之间的统计关联关系，表明文档的含义可以用与单词相同的向量空间表示。

分布思维方式的某些哲学基础来自哲学家维特根斯坦(Wittgenstein)的后期著作，他对是否为每个单词建立含义定义的完全形式化理论持怀疑态度，而相反认为“单词的含义是它在语言中的使用”(Wittgenstein，1953，PI 43)。也就是说，维特根斯坦不使用某种逻辑语言来定义每个单词，也不使用符号或真值，而是应该通过人们在日常交互中如何使用单词来表达和理解它来定义一个单词 ，从而预示了语言学和自然语言处理中向具身和经验模型的发展(Glenberg 和 Robertson 2000, Lake 和 Murphy 2020, Bisk 等人 2020，Bender 和 Koller 2020)。

更为密切相关的是通过离散特征向量来定义单词的想法，该特征至少可以追溯到笛卡尔和莱布尼兹(Wierzbicka 1992，Wierzbicka 1996)。到 20 世纪中叶，从 Hjelmslev(Hjelmslev，1969)(最初是 1943 年)的工作开始，并在早期生成语法模型中得以实现(Katz 和 Fodor，1963 年)，这种想法产生了用**语义特征**来

表示含义，代表某种原始含义的符号。例如，母鸡，公鸡或小鸡(hen, rooster, or chick,)之类的词有一些共同点(它们都描述了鸡)而又有一些不同之处(它们的年龄和性别)，可表示为：

hen +female, +chicken, +adult
rooster -female, +chicken, +adult
chick +chicken, -adult

然而，含义向量模型用来定义单词的维度，只是抽象地与手工构建的少量固定数量的维度概念相关。

尽管如此，仍有一些研究试图表明，嵌入模型的某些维度确实有助于某些特定的含义构成方面，比如这些早期的语义特征。

使用密集向量来建模单词的含义以及实际上是术语嵌入的方法，源于潜在语义索引(LSI)模型(Deerwester 等人，1988)，改写为 LSA(潜在语义分析)(Deerwester 等人，1990)。在 LSA 中，将奇异值分解(SVD)应用于术语文档矩阵(每个单元通过对数频率加权并通过熵进行归一化)，然后将前 300 个维用作 LSA 嵌入。奇异值分解(SVD)是一种用于查找数据集最重要维度的方法，这些维度是数据变化最大的维度。LSA 随后迅速被广泛应用：作为认知模型 Landauer 和 Dumais(1997)，以及诸如拼写检查(Jones 和 Martin，1997)，语言建模(Bellegarda 1997，Coccaro 和 Jurafsky 1998，Bellegarda 2000)之类的形态学归纳(Schone 和 Jurafsky 2000，Schone 和 Jurafsky 2001b)，多字表达(MWE)(Schone 和 Jurafsky 2001a)和论文评分(Rehder 等人 1998)。Schutze(1992b)同时开发了相关模型并将其应用于词义消歧。在 Schutze 等人(1995)的逻辑回归文档转发器中，LSA 还导致最早使用嵌入来表示概率分类器中的单词。Schutze(1992b)在 LSA 之后不久就提出了将 SVD 放在术语-术语矩阵(而不是术语-文档矩阵)上作为 NLP 含义模型的想法。Schutze 将 SVD 产生的低秩(97 维)嵌入应用于词义消歧的任务，分析了由此产生的语义空间，并提出了诸如丢弃高阶维的可能技术。参见 Schutze(1997a)。

在早期 SVD 工作的基础上，出现了许多替代矩阵模型，包括概率潜在语义索引(PLSI) (Hofmann, 1999)、潜在 Dirichlet 分配(LDA) (Blei 等人，2003)和非负矩阵分解(NMF) (Lee 和 Seung, 1999)。

LSA 社区似乎首先在 Landauer 等人的语言中使用了“嵌入”一词。(1997 年)，其数学含义的一种变体是从一个空间或数学结构到另一空间或数学结构的映射。在 LSA 中，单词嵌入似乎描述了从稀疏计数向量的空间到 SVD 密集向量的潜在空间的映射。尽管该词本来是指从一个空间到另一个空间的映射，但它已被转喻以表示在潜在空间中产生的密集向量。从这个意义上说，我们目前使用的是这个词。

在接下来的十年里，Bengio 等人(2003)和 Bengio 等人(2006)表明，神经语言模型也可以用于开发嵌入，作为单词预测任务的一部分。然后，Collobert 和 Weston (2007)， Collobert 和 Weston(2008)，以及 Collobert 等人(2011)证明了嵌入可以用来表示一些自然语言处理任务中的单词含义。Turian 等人(2010)比较了不同类型嵌入在不同 NLP 任务中的价值。Mikolov 等人(2011)表明，循环神经网络可以作为语言模型。Mikolov 等人(2013)提出了简化这些神经网络语言模型的隐含层以创建 skip-gram(也包括 CBOW)算法的想法。Mikolov 等人(2013a)提出了负采样训练算法。有许多关于静态嵌入及其参数化的调查(Bullinaria 和 Levy 2007, Bullinaria 和 Levy 2012, Lapesa 和 Evert 2014, Kiela 和 Clark 2014, Levy 等人 2015)。

参见 Manning 等人(2008)对向量在信息检索中的作用有更深入的理解，包括如何将查询与文档进行比较，tf-idf 的更多细节，以及缩放到非常大的数据集的问题。见 Kim(2019)关于 word2vec 的清晰而全面的教程。Cruuse(2004)是关于词汇语义的一个有用的介绍性语言文本。

6.15. 练习

(无)

7. 神经网络和神经语言模型

> “[M]achines of this character can behave in a very complicated manner when the number of units is large.”
>
> Alan Turing (1948) “Intelligent Machines”, page 6

神经网络是用于语言处理的基本计算工具，并且是一个非常古老的工具。它们之所以被称为神经元，是因为它们的起源在于麦卡洛克-皮茨神经元(McCulloch 和 Pitts，1943)，这是人类神经元的简化模型，可以作为一种计算元素，可用命题逻辑来描述。但是，语言处理中的现代用法不再汲取这些早期生物学灵感。

取而代之的是，现代的神经网络是由小型计算单元组成的网络，每个计算单元采用输入值的向量并产生单个输出值。在本章中，我们介绍了应用于分类的神经网络。我们介绍的体系结构称为**前馈网络(feedforward network)**，因为计算是从一个层(由很多单元组成)到下一个层迭代进行的。现代神经网络的使用通常称为**深度学习(deep learning)**，因为现代网络通常很深(具有许多层)。

神经网络与逻辑回归具有许多相同的数学原理。但是，神经网络比逻辑回归具有更强大的分类器，实际上，可以证明一个最小的神经网络(技术上具有单个“隐藏层”)可以学习任何功能。

神经网络分类器在另一个方面不同于逻辑回归。通过逻辑回归，我们通过基于领域知识开发多种丰富的特征模板，将回归分类器应用于许多不同的任务。在使用神经网络时，更常见的是避免大多数使用丰富的手工衍生特征，而建立以原始单词为输入并学习归纳特征的神经网络，作为学习分类过程的一部分。我们在第 6 章中看到了用于嵌入的这种表示学习的示例。非常深的网络特别擅长表示学习。因此，深度神经网络是解决大规模问题的正确工具，可以提供足够的数据来自动学习特征。

在本章中，我们将介绍前馈网络作为分类器，并将它们应用于语言建模的简单任务：为单词序列分配概率并预测即将出现的单词。在随后的章节中，我们将介绍神经模型的许多其他方面，例如**递归神经网络(recurrent neural networks，RNN)**和 **Transformer**(第 9 章)，上下文嵌入(如 **BERT**)(第 10 章)以及**编解码器(encoder-decoder)**模型和**注意力(attention)**(第 11 章)。

7.1. 单元

神经网络的构建块是单个计算单元。一个单元将一组实数值作为输入，对其进行一些计算，然后产生输出。

从本质上讲，神经单元正在对其输入进行加权求和，其中的另一个项称为偏差项(bias term)。给定一组输入 $x_1 \dots x_n$，一个单元具有一组相应的权重 $w_1 \dots w_n$ 和偏差 b，因此加权总和 z 可以表示为：

$$z = b + \sum_i w_i x_i \tag{7.1}$$

通常，使用向量符号表示此加权和会更方便； 从线性代数中回想起，向量本质上只是一个数字列表或数字数组。因此，我们将以权重向量 w，标量偏差 b 和输入向量 x 来讨论 z，并将总和替换为方便的点积：

$$z = w \cdot x + b \tag{7.2}$$

正如公式 7.2 中定义的，z 只是一个实数。

最终，神经单元将非线性函数 f 应用于 z(而不是使用 x 的线性函数 z)作为输出。我们将此功能的输出称为单元 a 的**激活(activation)**值。由于我们仅对单个单元建模，因此节点的激活实际上是网络的最终输出，通常将其称为 y。因此，值 y 定义为：

$$y = a = f(z)$$

我们将在下面讨论三种流行的非线性函数 f( ) (sigmoid，tanh，和修正的线性 ReLU)，但从教学角度

来看，从 **sigmoid** 函数开始比较方便，因为我们在第 5 章看到过它:

$$y = \sigma(z) = \frac{1}{1+e^{-z}} \tag{7.3}$$

Sigmoid(如图 7.1 所示)具有许多优点。它将输出映射到[0,1]范围内，这对于将异常值压向 0 或 1 非常有用。而且它是可微的，正如我们在 5.8 节中看到的那样，它对于学习很方便。

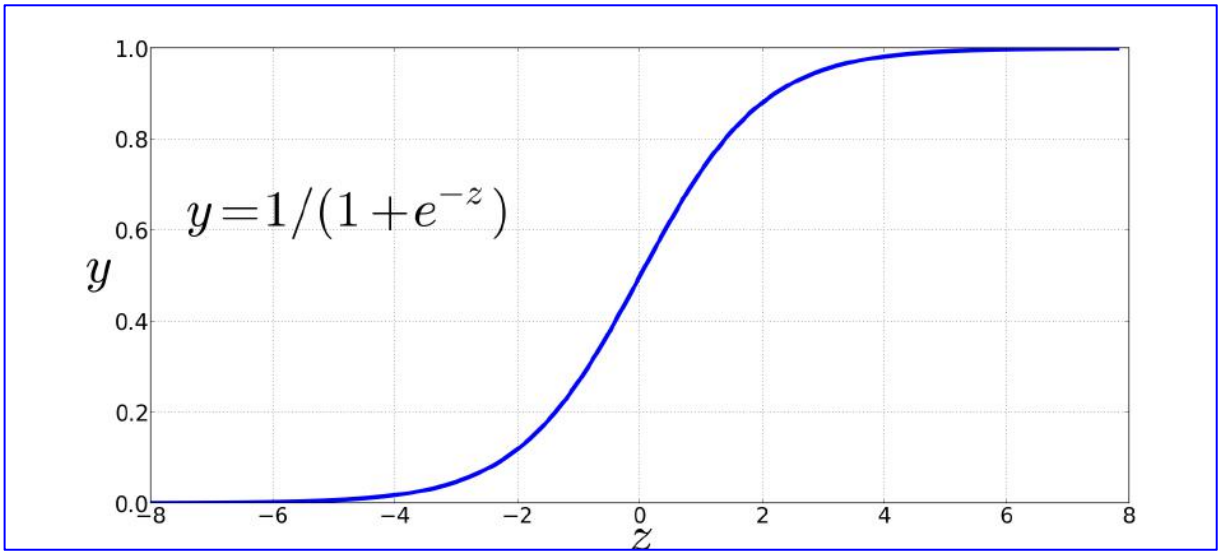


图 7-1：Sigmoid 函数

图注：Sigmoid 函数接受一个实数，并将其映射到范围[0,1]。它在 0 附近几乎是线性的，但是离群值被压向 0 或 1。

将公式 7.2 代入公式 7.3 得到一个神经单元的输出:

$$y = \sigma(w \cdot x + b) = \frac{1}{1+\exp(-(w \cdot x + b))} \tag{7.4}$$

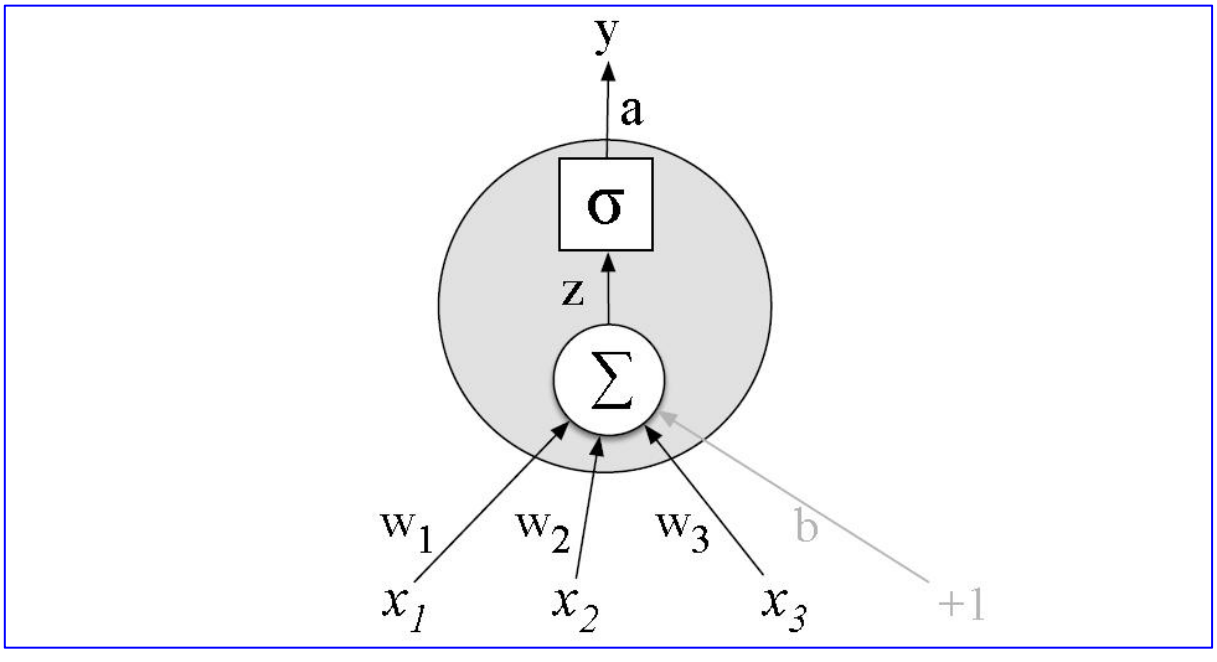


图 7-2：一个神经单元

图注：一个神经单元，取 3 个输入 x_1，x_2 和 x_3(以及一个偏差 b，我们将其表示为固定在+1 上的输入的权重)，并产生输出 y。我们包括一些方便的中间变量：总和 z 的输出和 S 型曲线 a 的输出。在这种情况下，单位 y 的输出与 a 相同，但是在更深的网络中，我们将保留 y 表示整个网络的最终输出，而保留 a 作为单个节点的激活。

图 7.2 显示了基本神经单元的最终示意图。在此示例中，该单元采用 3 个输入值 x_1，x_2 和 x_3，并计算一个加权和，将每个值乘以一个权重(分别为 w_1，w_2 和 w_3)，将它们加到偏差项 b，然后将累加结果通过 S 型函数传递，以得出介于 0 和 1 之间的数字的最终结果。

让我们来看一个例子，只是为了获得直觉。假设我们有一个具有以下权重向量和偏差的单元：

$$w = [0.2,0.3,0.9]$$

$$b = 0.5$$

该单元将使用以下输入向量：

x = [0.5,0.6,0.1]

结果输出 y 为：

$$y = \sigma(w \cdot x + b) = \frac{1}{1+e^{-(w \cdot x + b)}} = \frac{1}{1+e^{-(.5*.2+.6*.3+.1*.9+.5)}} = \frac{1}{1+e^{-0.87}} = .70$$

实际上，sigmoid 通常不用作激活函数。图 7.3a 所示的 **tanh** 函数非常相似，但几乎总是更好。tanh 是 S 型的一种变体，范围从-1 到+1：

$$y = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{7.5}$$

最简单的激活函数，也许是最常用的激活函数，是修正的线性单元，也称为 **ReLU**，如图 7.3b 所示。若 x 为正数，则它与 x 相同；否则为 0：

$$y=\max(x,0) \tag{7.6}$$

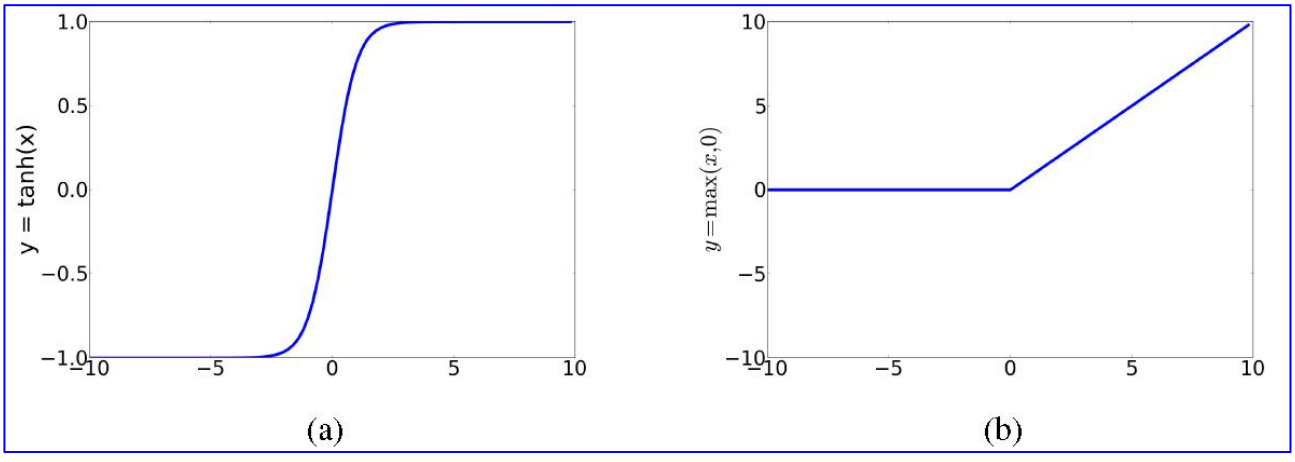


图 7-3：tanh 和 ReLU 激活函数

这些激活函数具有不同的属性，这些属性使它们对于不同的语言应用程序或网络体系结构很有用。例如，tanh 函数具有很好的特性，可以平滑地进行微分，并将离群值映射到均值。另一方面，修正函数具有很好的特性，这是由于它非常接近线性。在 S 型或 tanh 函数中，很高的 z 值会导致 y 值的**饱和(saturated)**，即非常接近 1，并且导数非常接近 0。零导数会引起学习问题，因为我们将在 7.4 节中看到，我们将通过向后传播误差信号并乘以网络各层的梯度(偏导数)来训练网络； 几乎为 0 的梯度会导致误差信号变得越来越小，直到它变得太小而无法用于训练为止，这个问题称为**消失梯度(vanishing gradient)**问题。修正函数没有这个问题，因为 ReLU 对高 z 值的导数是 1，而不是非常接近 0。

7.2.XOR 问题

在神经网络的历史早期，人们已经意识到神经网络的力量，就像激发它们的真实神经元一样，来自将这些单元组合成更大的网络。

Minsky 和 Papert(1969)证明了多层网络需求的最巧妙的证明之一，即单个神经单元无法计算其输入的某些非常简单的功能。考虑计算两个输入(例如 AND，OR 和 XOR)的基本逻辑功能的任务。提醒一下，

这是这些函数的真值表：

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

此示例首先针对**感知器(perceptron)**显示，该感知器是一个非常简单的神经单元，具有二进制输出并且不具有非线性激活函数。感知器的输出 y 为 0 或 1，其计算方式如下(使用与公式 7.2 相同的权重 w，输入 x 和偏差 b)：

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases} \tag{7.7}$$

构建一个可计算其二元输入的逻辑 AND 和 OR 功能的感知器非常容易；图 7.4 显示了必要的权重。

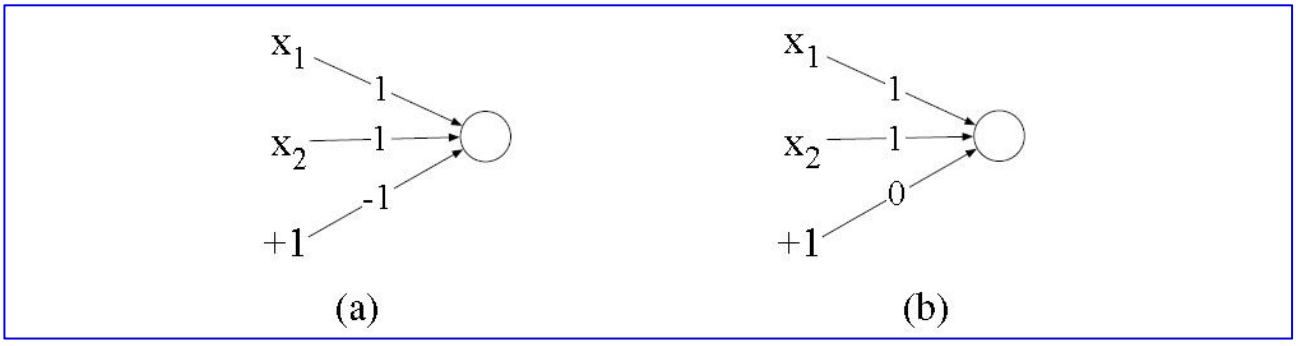


图 7-4：逻辑函数的感知器

图注：用于计算逻辑函数的感知器的权重 w 和偏差 b。输入显示为 x_1 和 x_2，偏差显示为特殊节点，其值+1 与偏差权重 b 相乘。(a)逻辑 AND，显示权重 $w_1 = 1$ 和 $w_2 = 1$，偏差权重 $b = -1$。(b)逻辑 OR，显示权重 $w_1 = 1$ 和 $w_2 = 1$，偏差权重 $b = 0$。这些权重/偏差仅仅是实现函数的无数可能的权重和偏差集之一。

但是事实证明，不可能构建感知器来计算逻辑 XOR(异或)！(值得花点时间尝试一下！)

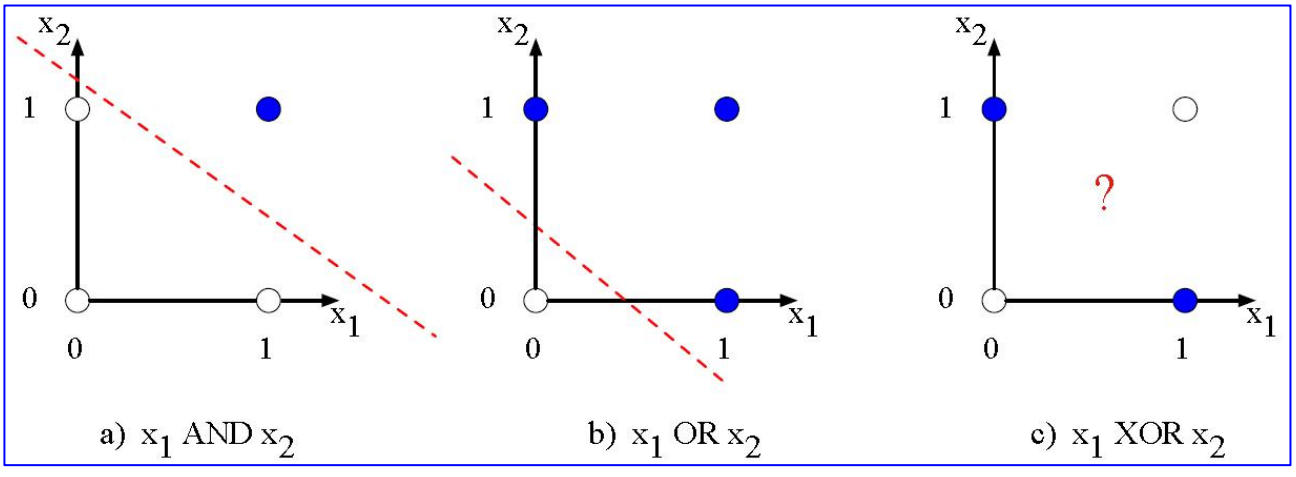


图 7-5：函数 AND，OR 和 XOR 的分类线

图注：函数 AND，OR 和 XOR 在 x 轴上表示为输入 x_1，在 y 轴上表示为输入 x_2。实心圆圈表示感知器输出为 1，白色圆圈感知器输出为 0。无法绘制正确区分 XOR 的两个类别的线。图样来自 Russell 和 Norvig(2002)。

重要结果背后的直觉取决于对感知器是线性分类器的理解。对于二维输入 x_1 和 x_2，感知方程 $w_1x_1 + w_2x_2 + b = 0$ 是直线的方程(我们可以通过将其放入标准线性格式中来看到它：$x_2 = (-w_1/w_2)x_1 + (-b/w_2)$)。该线充当二维空间中的**决策边界(decision boundary)**，在该二维空间中，将输出 0 分配给该线一侧的所有输入点，将输出 1 分配给该线另一侧的所有输入点。如果我们有 2 个以上的输入，则决策边界将变为超平面而不是直线，但是想法是相同的，将空间分为两类。

图 7.5 显示了可能的逻辑输入(00、01、10 和 11)，以及由一个可能的一组参数组成的与门(AND)、或门(OR)的分类线。注意，根本没有办法画出将异或门(XOR)的正例(01 和 10)与负例(00 和 11)分开的线。我们说 XOR 不是**线性可分(linearly separable)**函数。当然，我们可以用曲线或其他函数绘制边界，但不能绘制一条直线。

7.2.1. 解决方案：神经网络

虽然 XOR 函数不能由单个感知器计算，但可以由单元的分层网络计算。我们来看看 Goodfellow 等人(2016)的示例。使用两层基于 ReLU 的单元计算 XOR。图 7.6 显示了一个由两层神经单元处理输入的图形。中间层(称为 h)具有两个单元，对于正确计算 XOR 函数的每个 ReLU，显示了一组权重和偏差。

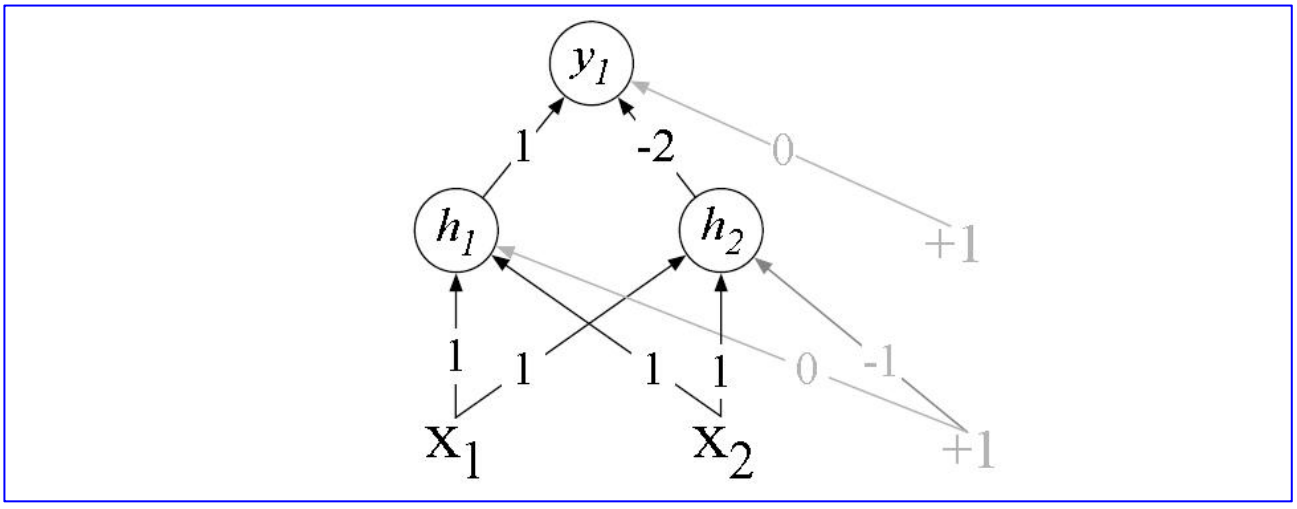


图 7-6：Goodfellow 等人的 XOR 解决方案

图注：Goodfellow 等人(2016)的 XOR 解决方案。共有三个 ReLU 单元，分为两层；称之为 h_1、h_2(h 代表隐藏层)和 y_1。如前所述，箭头上的数字表示每个单元的权重 w，我们将偏差 b 表示为固定为+1 的单元的权重，偏差权重/单元为灰色。

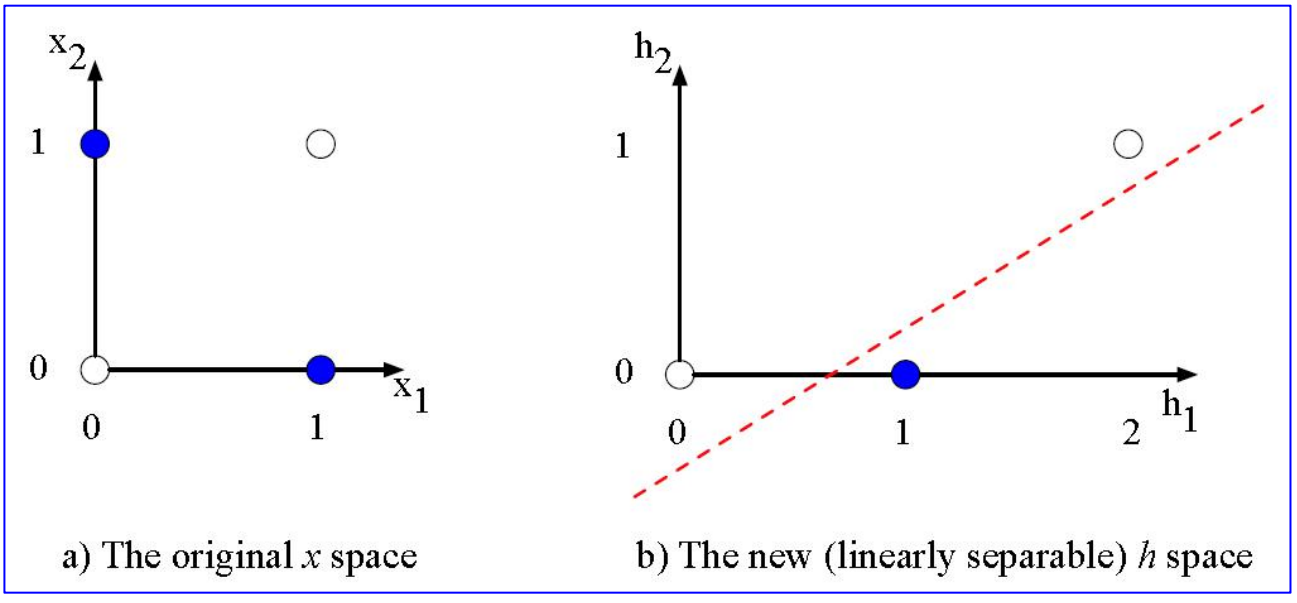


图 7-7：隐含层形成输入的新表示

图注：图(a)显示了原始输入 x 的表示，图(b)显示了隐藏层 h 的表示。注意，输入点[0 1]已经与输入点[1 0]折叠，这使得线性分离异或(XOR)的正负情况成为可能。根据 Goodfellow 等人(2016)。

让我们来看看输入 x = [0 0]会发生什么。如果我们将每个输入值乘以适当的权重，再求和，然后加上偏差 b，我们得出矢量[0 -1]，然后应用修正的线性变换将 h 层的输出设为[0 0] 。现在，我们再次乘以权重、求和并加上偏差(在这种情况下为 0)，得出的值为 0。读者应该对其余 3 个可能的输入对进行计算，以查看输入[0 1]和[1 0]的 y 值为 1，输入[0 0]和[1 1]的 y 值为 0。

查看中间结果(两个隐藏节点 h_1 和 h_2 的输出)也很有启发。在上一段中，我们证明了输入 x = [0 0]的 h 矢量为[0 0]。图 7.7b 显示了所有 4 个输入的 h 层的值。请注意，两个输入点 x = [0 1]和 x = [1 0](XOR 输出= 1 的两种情况)的隐藏表示被合并到单个点 h = [1 0]。通过合并，可以很容易地线性区分异或(XOR)的正例和负例。换句话说，我们可以将网络的隐藏层视为输入的表示形式。

在此示例中，我们仅在图 7.6 中规定了权重。但是对于真实的例子，将使用第 7.4 节介绍的误差反向传播算法自动学习神经网络的权重。这意味着隐藏层将学习形成有用的表示。这种直觉，即神经网络可以自动学习输入的有用表示形式，是它们的主要优势之一，这一点我们将在以后的章节中反复讨论。

请注意，解决 XOR 问题的方法需要具有非线性激活函数的单元网络。由简单线性(感知器)单元组成的网络无法解决 XOR 问题。这是因为，由多层纯线性单元组成的一个网络，始终可以被简化为(例如,显示在计算上相同)一个单层线性单元(具有适当权重)，而且我们已经显示(视觉上,在图 7.5 中)了单个单元不能解决 XOR 问题。

7.3. 前馈神经网络

现在，让我们简单地介绍一下最简单的神经网络，即**前馈网络(feedforward network)**。前馈网络是一个多层网络，其中的单元之间没有循环连接。每层单元的输出将传递到下一个较高层的单元，而没有输出被传递回到较低层。(在第 9 章中，我们将介绍具有循环的网络，称为循环神经网络。)

出于历史原因，多层网络，尤其是前馈网络，有时也称为**多层感知器(multi-layer perceptrons，MLP)**。这是一种技术上的误称，因为现代多层网络中的单元不是感知器(感知器是纯线性的，但是现代网络是由具有非线性的单元(如 S 型)组成的)，但在某种程度上，这个名称就固定下来了。

简单前馈网络具有三种节点：输入单元，隐藏单元和输出单元。图 7.8 示出了一个图片。

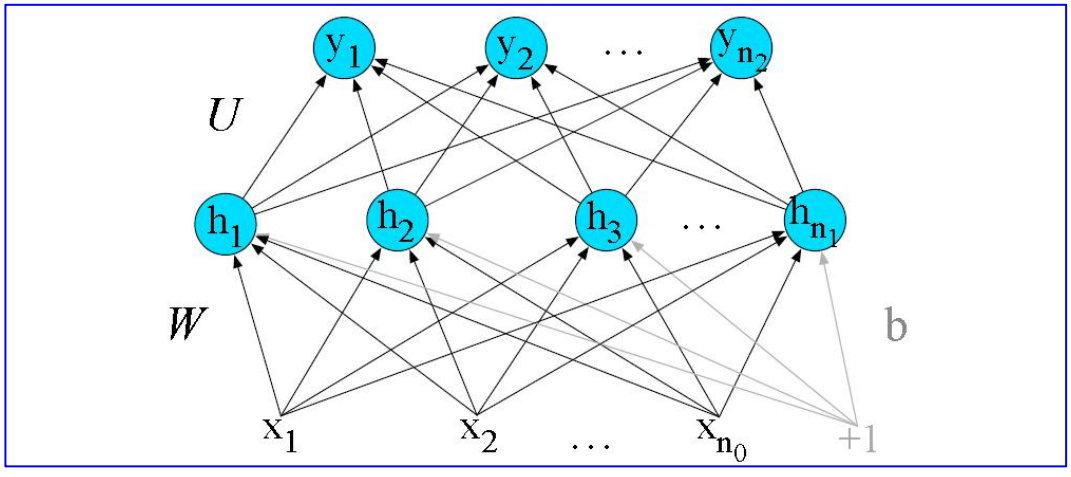


图 7-8：一个简单的 2 层前馈网络

图注：具有一个隐藏层，一个输出层和一个输入层(枚举层时通常不计算输入层)。

输入单元只是标量值，如图 7.2 所示。

神经网络的核心是由隐藏单元组成的隐藏层，每个隐藏单元都是第 7.1 节所述的神经单元，对输入进行加权求和，然后应用非线性。在标准体系结构中，每一层都是**全连接(full connected)**，这意味着每一层中的每个单元都将前一层中所有单元的输出作为输入，并且来自两个相邻层的每对单元之间都有一个连接。因此，每个隐藏单元对所有输入单元进行累加求和。

回想一下，单个隐藏单元的参数为 w(权重向量)和 b(偏差标量)。我们通过将每个单位 i 的权重向量 w_i 和偏差标量 b_i 组合为单个权重矩阵 W 和整个层的单个偏差向量 b 来表示整个隐藏层的参数(见图 7.8)。权重矩阵 W 的每个元素 W_{ji} 代表从第 i 个输入单元 x_i 到第 j 个隐藏单元 h_j 的连接的权重。

对整个层的权重使用单个矩阵 W 的优点是，现在可以通过简单的矩阵运算非常有效地完成前馈网络的隐藏层计算。实际上，该计算仅包括三个步骤：将权重矩阵与输入向量 x 相乘，加上偏差向量 b，然后应用激活函数 g(例如上面定义的 S 型，tanh 或 ReLU 激活函数)。

因此，使用 S 形函数 σ，隐藏层的输出矢量 h 如下：

$$h = \sigma(Wx+b) \tag{7.8}$$

注意，我们在这里对一个向量应用σ函数，而在公式 7.3 中，它是对一个标量应用的。因此，我们允许σ(·)，以及实际上任何激活函数 g(·)适用于向量元素，因此 $g[z_1,z_2,z_3] = [g(z_1),g(z_2),g(z_3)]$。

让我们介绍一些常数来表示这些向量和矩阵的维数。我们将输入层称为网络的第 0 层，并用 n_0 表示输入的数量，因此 x 是维度为 n_0 的实数的矢量，或者更正式地说 $x \in R^{n_0}$是维数$[n_0,1]$的列向量。我们将隐藏层称为第 1 层，将输出层称为第 2 层。该隐藏层的维数为 n_1，因此 $h \in R^{n_1}$和 $b \in R^{n_1}$(因为每个隐藏单元可以采用不同的偏差值)，并且权重矩阵 W 具有维数 $W \in R^{n_1 \times n_0}$，即$[n_1,n_0]$。

花点时间说服自己，在公式 7.8 中的矩阵乘法将计算每个 h_j 的值为$\sigma(\sum_{i=0}^{n_0} w_{ji}x_i + b_j)$。

正如我们在 7.2 节中所看到的，结果值 h(对于隐藏的还是对于假设的)形成了输入的表示。输出层的作用是采用新的表示形式 h 并计算最终输出。该输出可能是实数值，但是在许多情况下，网络的目标是做出某种分类决策，因此我们将重点介绍分类情况。

如果我们正在执行情感分类之类的二元任务，那么我们可能只有一个输出节点，其值 y 是正面情感与负面情感的概率。如果我们正在进行多元分类，例如分配词类标签，则每个潜在的词类可能有一个输出节点，其输出值是该词类的概率，并且所有输出节点的值之和必须为 1。因此，输出层给出了横跨(across)输出节点的概率分布。

让我们看看这是怎么发生的。与隐藏层一样，输出层具有权重矩阵(简称为 U)，但是某些模型在输出层中不包含偏差矢量 b，因此在此示例中，我们将通过消除偏差矢量来简化操作。权重矩阵乘以其输入向量(h)以产生中间输出 z。

$$z = Uh$$

有 n_2 个输出节点，因此 $z \in R^{n2}$，权重矩阵 U 具有维度 $U \in R^{n_2 \times n_1}$，元素 U_{ij} 是从隐藏层的单元 j 到输出层的单元 i 的权重。

但是，z 不能作为分类器的输出，因为 z 是实数值的向量，而我们需要的分类是概率的向量。有一个方便的函数可以对实值向量进行归一化，即将其归一化为将其编码为概率分布的向量(所有数字都在 0 到 1 之间，并且总和为 1)：softmax 函数(第 5 章)。对于向量为 d 的向量，softmax 定义为：

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \quad 1 \le i \le d \tag{7.9}$$

因此，例如给定向量 z = [0.6，1.1，-1.5，1.2，3.2，-1.1]，softmax(z)为[0.055，0.090，0.0067，0.10，0.74，0.010]。

您可能还记得，在第 5 章的逻辑回归中，softmax 正是根据实值数字(从权重乘以特征求和而来)的向量来创建概率分布的方法。

这意味着我们可以将具有一个隐藏层的神经网络分类器视为构建向量 h，该向量 h 是输入的隐藏层表示，然后对网络在 h 中发展的特征进行标准逻辑回归。相比之下，在第 5 章中，主要通过特征模板手动设计功能。因此，神经网络类似逻辑回归，但是有两点不同：(a)有许多层，因为深度神经网络就像是逻辑回归分类器的一层又一层；(b)不是通过特征模板形成特征，而是网络的前几层自己引导特征表示。

以下是具有单个隐藏层的前馈网络的最终方程式，该方程式采用输入向量 x，输出概率分布 y，并由权重矩阵 W 和 U 以及偏置向量 b 进行参数化：

$$\begin{aligned} h &= \sigma(Wx+b) \\ z &= Uh \\ y &= \text{softmax}(z) \end{aligned} \tag{7.10}$$

我们将此网络称为 2 层网络(传统上，在对层进行编号时，我们不对输入层进行计数，而对输出层进行计数)。因此，根据此术语，逻辑回归是一个 1 层网络。

现在，让我们设置一些符号，以便更轻松地讨论深度大于 2 的更深层网络。我们将在方括号中使用上标来表示层号，输入层的编号从 0 开始。因此，$W^{[1]}$表示(第一)隐藏层的权重矩阵，$b^{[1]}$表示(第一)隐藏层的偏差矢量。n_j 表示第 j 层的单元数。我们将使用 g(·)代表激活函数，中间层往往是 ReLU 或 tanh，输出层

通常为 softmax。我们将使用 $a^{[i]}$表示第 i 层的输出，使用 $z^{[i]}$表示权重和偏差 $W^{[i]}\ a^{[i-1]}+ b^{[i]}$的组合。第 0 层用于输入，因此我们通常将输入 x 称为 $a^{[0]}$。

因此，我们可以将公式 7.10 中的 2 层网络重新表示为:

$$\begin{aligned} z^{[1]} &= W^{[1]}a^{[0]} + b^{[1]} \\ a^{[1]} &= g^{[1]}(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} &= g^{[2]}(z^{[2]}) \\ \hat{y} &= a^{[2]} \end{aligned} \tag{7.11}$$

注意，使用这种符号，每一层的计算公式都是相同的。给定输入向量 $a^{[0]}$，在 n 层前馈网络中计算前向步长的算法因此很简单：

for i **in** 1..n

$z^{[i]} = W^{[i]}\ a^{[i-1]} + b^{[i]}$

$a^{[i]} = g^{[i]}(z^{[i]})$

$\hat{y} = a^{[n]}$

激活函数 g(·)通常在最后一层是不同的。因此，对于多元分类，$g^{[2]}$可能是 softmax；对于二进制分类，$g^{[2]}$可能是 sigmoid。而 ReLU 或 tanh 可能是内部层的激活函数 g(·)。

更换偏置单元 在描述网络时，我们经常会使用略微简化的符号表示完全相同的函数，而无需引用显式偏置节点 b。相反，我们向每个值始终为 1 的层添加一个虚拟节点 a_0。因此，输入层 0 层将具有一个虚拟节点$a_0^{[0]}= 1$，第 1 层将具有$a_0^{[1]}= 1$，并且以此类推。该虚拟节点仍然具有关联的权重，并且该权重代表偏差值 b。例如：

不使用以下公式：

$$h = \sigma(Wx+b) \tag{7.12}$$

而使用以下公式：

$$h = \sigma(Wx) \tag{7.13}$$

但是现在，我们的向量 x 不再具有 n 个值：$x = x_1, \ldots, x_n$，而是 n+1 个值，其中第 0 个虚拟值 $x_0 = 1$：$x = x_0, \ldots, x_n$。

不是像下面那样计算每个 h_j：

$$h_j = \sigma\left(\sum_{i=1}^{n_0} w_{ji}x_i + b_j\right), \tag{7.14}$$

而是改用：

$$\sigma\left(\sum_{i=0}^{n_0} w_{ji}x_i\right), \tag{7.15}$$

其中值 W_{j0} 代替了 b_j。图 7.9 显示了可视化。

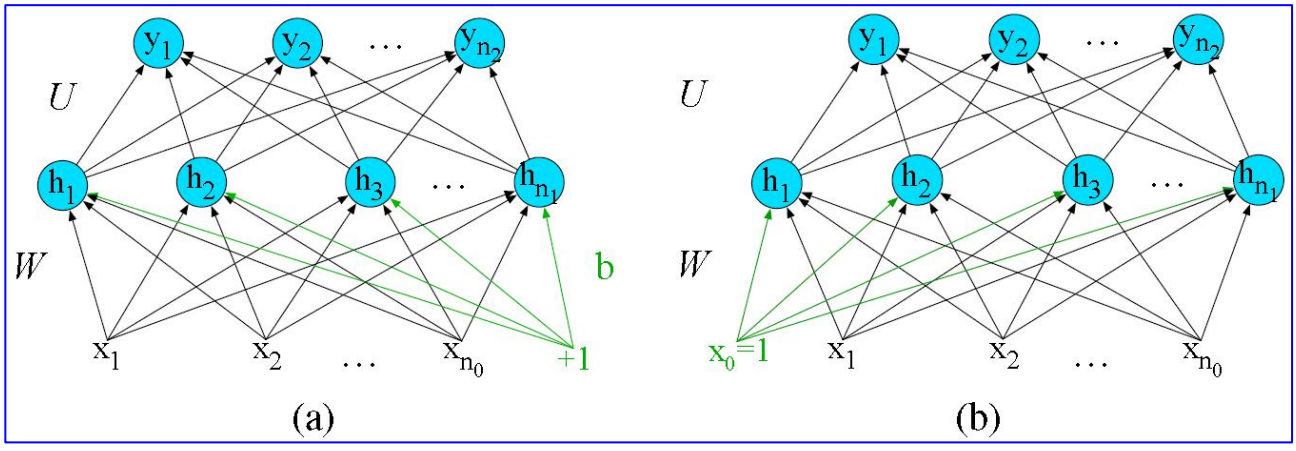


图 7-9：用 x_0 替换偏置节点 b

在下一节中，我们将继续在学习示例中将偏差显示为 b，但是在本书的其余部分中，我们将切换为这种简化的表示法，而没有明确的偏差项。

7.4. 训练神经网络

前馈神经网络是有监督机器学习的一个实例，其中我们知道每个观测值 x 的正确输出 y。系统通过公式 7.11 产生的结果是ŷ，即系统对真实 y 的估计。训练过程的目标是学习每个层 i 的参数 $W^{[i]}$和 $b^{[i]}$，使每个训练观察的ŷ尽可能接近真实 y。

一般来说，我们都是通过在第 5 章中介绍的逻辑回归方法来完成这一切的，所以读者在继续之前应该熟悉那一章。

首先，我们需要一个**损失函数(loss function)**来对系统输出和**黄金输出(gold output)**之间的距离进行建模，通常使用用于逻辑回归的损失函数，即**交叉熵损失(cross-entropy loss)**。

其次，为了找到使损失函数最小的参数，我们将使用第 5 章中介绍的梯度下降优化算法。

第三，梯度下降需要知道损失函数的梯度，该向量包含损失函数相对于每个参数的偏导数。这是神经网络的学习比逻辑回归更复杂的部分。在逻辑回归中，对于每个观察值，我们都可以直接计算损失函数相对于单个 w 或 b 的导数。但是对于神经网络来说，在许多层中都有数百万个参数，当损失附加到后面的层时，如何计算第一层中某些权重的偏导数要困难得多。我们如何分摊所有这些中间层的损失？

答案是称为**误差反向传播(error backpropagation)**或**逆向微分(reverse differentiation)**的算法。

7.4.1. 损失函数

在神经网络中使用的交叉熵损失和我们在逻辑回归中看到的是一样的。事实上，如果将神经网络用作二进制分类器，将 sigmoid 置于最后一层，则损失函数与我们在公式 5.11 中看到的逻辑回归完全相同:

$$L_{CE}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})] \tag{7.16}$$

如果神经网络被用作多元分类器怎么办？令 y 为代表真实输出概率分布的 C 类上的向量。这里的交叉熵损失是：

$$L_{CE}(\hat{y},y) = -\sum_{i=1}^{C} y_i \log\hat{y}_i \tag{7.17}$$

我们可以进一步简化该方程式。假设这是一项**硬分类(hard classification)**任务，这意味着只有一个类别是正确的类别，并且每个类别的 y 中都有一个输出单元。如果真实类别是 i，则 y 是一个向量，其中 $y_i = 1$ 且 $y_j = 0 \ \ \forall j = i$。像这样的一个向量，其中只有一个值为 1，其余值均为 0，被称为**独热(one-hot)**向量。除真实类别的项之外，公式 7.17 中的各项总和将为 0，即：

$$\begin{aligned} L_{CE}(\hat{y},y) &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\}\log\hat{y}_i \\ &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\}\log\hat{p}(y=k|x) \\ &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\}\log\frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \end{aligned} \tag{7.18}$$

因此，交叉熵损失只是对应于正确类的输出概率的对数，因此我们也称其为**负对数似然损失(negative log likelihood loss)**:

$$L_{CE}(\hat{y},y) = -\log\hat{y}_i, \quad \text{(where } i \text{ is the correct class)} \tag{7.19}$$

插入公式 7.9 中的 softmax 公式，其中 K 为类别的数量：

$$L_{CE}(\hat{y},y) = -\log \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{(where } i \text{ is the correct class)} \tag{7.20}$$

7.4.2. 计算梯度

我们如何计算该损失函数的梯度？计算梯度需要损失函数相对于每个参数的偏导数。对于只有一个权重层和 S 形输出的网络(这就是逻辑回归)，我们可以简单地使用等式 7.21(并在 5.8 节中得出)中用于逻辑回归的损失导数：

$$\begin{aligned} \frac{\partial L_{CE}(w,b)}{\partial w_j} &= (\hat{y}-y)x_j \\ &= (\sigma(w\cdot x+b)-y)x_j \end{aligned} \tag{7.21}$$

或者对于一个有一个隐含层和 softmax 输出的网络，我们可以使用公式 5.37 中 softmax 损失的导数:

$$\begin{aligned} \frac{\partial L_{CE}}{\partial w_k} &= (\mathbb{1}\{y=k\}-p(y=k|x))x_k \\ &= \left(\mathbb{1}\{y=k\}-\frac{\exp(w_k\cdot x+b_k)}{\sum_{j=1}^{K}\exp(w_j\cdot x+b_j)}\right)x_k \end{aligned} \tag{7.22}$$

但是，这些导数仅对一个权重层(最后一个权重层)提供正确的更新！ 对于深度网络，计算每个权重的梯度要复杂得多，因为我们计算的是权重参数的导数，这些参数出现在网络较早(very early)的层，即便损失只在网络的较后(very end)端计算。

解决此梯度的方法是一种称为**误差反向传**的算法(Rumelhart 等人，1986)。虽然反向传播是专门为神经网络发明的，但事实证明它与一种称为逆向微分的更通用的过程是一样的，后者取决于**计算图**的概念。

7.4.3. 计算图

计算图是计算数学表达式的过程表示，其中的计算被分解成独立的操作，每个操作被建模为图中的一个节点。

考虑计算函数 L(a，b，c)= c(a + 2b)。如果我们明确每个分量的加法和乘法运算，并为中间输出添加名称(d 和 e)，则所得的一系列计算为：

d = 2*b
e = a+d
L = c*e

现在我们可以用一个图来表示它，用节点表示每个操作，有向边表示每个操作的输出作为下一个操作的输入，如图 7.10 所示。计算图最简单的用法是用给定的输入来计算函数的值。在图中，我们假设输入 a=3, b=1, c=-2，我们已经展示了向前传递的结果，计算结果 L(3,1，-2) = -10。在计算图的向前传递过程中，我们从左到右应用每个操作，将每个计算的输出作为下一个节点的输入来传递。

7.4.4. 计算图上的逆向微分

计算图的重要性来自向后传递，它用于计算权重更新所需的导数。在这个例子中，我们的目标是计算输出函数 L 对每个输入变量的导数，即∂L/∂a，∂L/∂b，∂L/∂c。导数∂L/∂a，告诉我们 a 中的一个小变化对 L 的影响有多大。

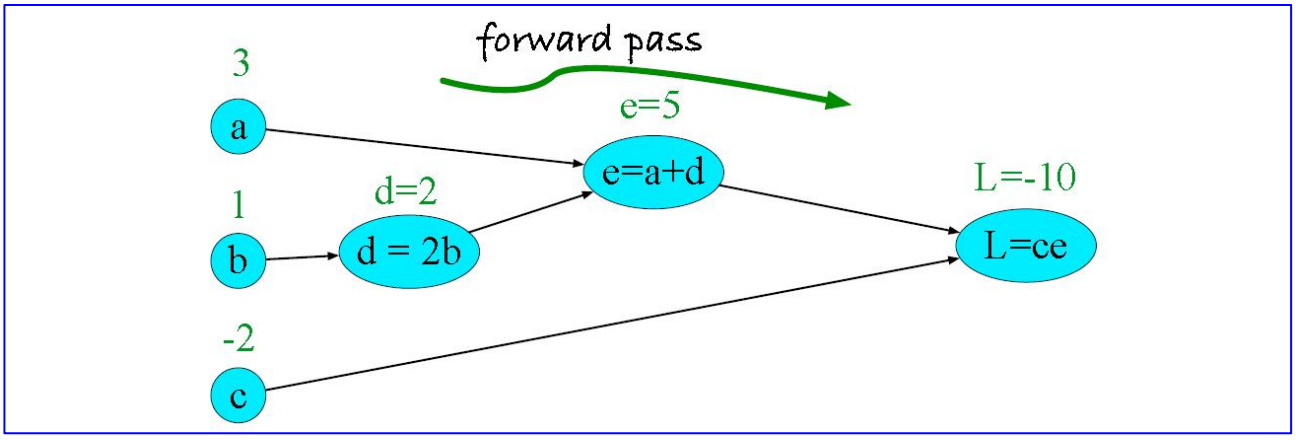


图 7-10：函数 L(a,b, c) = c(a+2b)的正向传递计算图

图注：输入节点 a = 3，b = 1，c = - 2，表示 L 的正向传递计算。

逆向微分在微积分中利用了链式规则。假设我们计算一个复合函数 f(x) = u(v(x))的导数。f(x)的导数等于 u(x)对 v(x)的导数乘以 v(x)对 x 的导数：

$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx} \tag{7.23}$$

链式规则扩展到两个以上的函数。如果计算复合函数 f(x) = u(v(w(x)))的导数，则 f(x)的导数为:

$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dw} \cdot \frac{dw}{dx} \tag{7.24}$$

现在我们来计算我们需要的三个导数。因为在计算图 L = ce 中，我们可以直接计算 ∂ L/ ∂ c 的导数:

$$\frac{\partial L}{\partial c} = e \tag{7.25}$$

对于其他两个，我们需要使用链式规则:

$$\begin{aligned} \frac{\partial L}{\partial a} &= \frac{\partial L}{\partial e}\frac{\partial e}{\partial a} \\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial b} \end{aligned} \tag{7.26}$$

公式 7.26 因此需要 5 个中间导数：∂L/∂e、∂L/∂c、∂e/∂a、∂e/∂d、∂d/∂b，它们如下(利用"和的导数是导数的和"这一事实):

$$\begin{aligned} L = ce &: \quad \frac{\partial L}{\partial e} = c, \frac{\partial L}{\partial c} = e \\ e = a + d &: \quad \frac{\partial e}{\partial a} = 1, \frac{\partial e}{\partial d} = 1 \\ d = 2b &: \quad \frac{\partial d}{\partial b} = 2 \end{aligned}$$

在反向过程中，我们沿着图的每条边从右到左计算这些偏导数，将必要的偏导数相乘，得到我们需要的最终导数。因此，我们首先用∂L/∂L = 1 来标注最后一个节点。向左移动，然后计算∂L/∂c 和∂L/∂e，以此类推，直到我们把图标注到输入变量为止。正向传递已经很方便地计算出了我们需要的正向中间变量(如 d 和 e)的值来计算这些导数。图 7.11 显示了反向传播。在每个节点上，我们需要计算关于父节点的局部偏

导数，将它乘以从父节点传递下来的偏导数，然后传递给子节点。

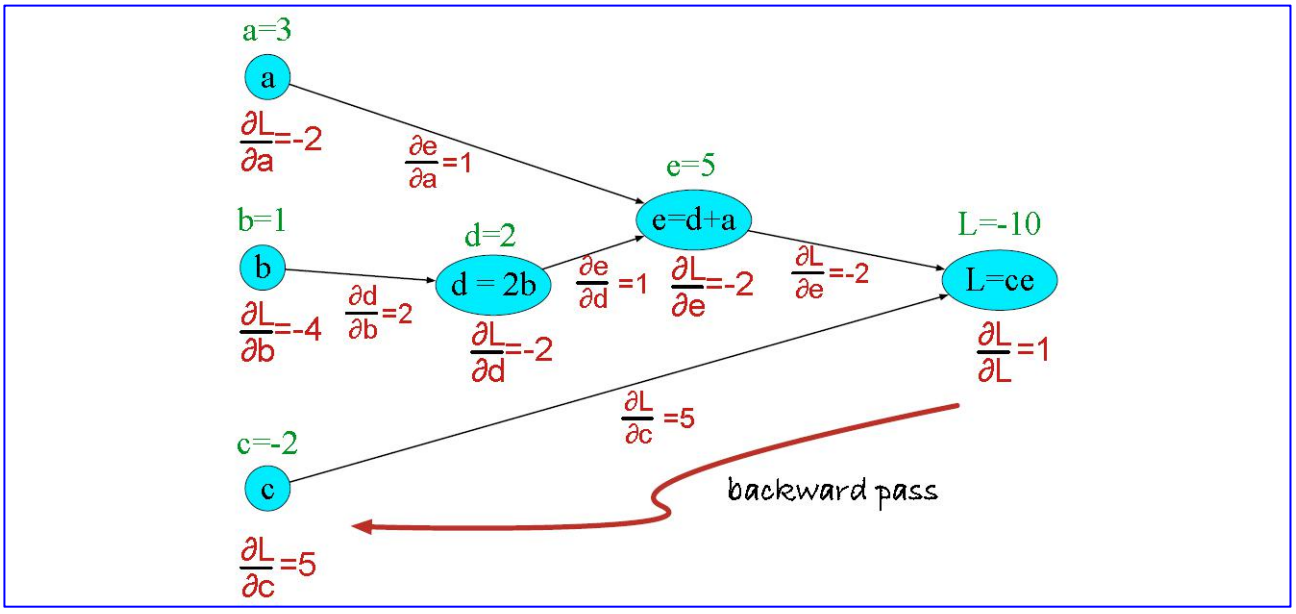


图 7-11：函数 L(a,b, c) = c(a+2b)的反向传递计算图

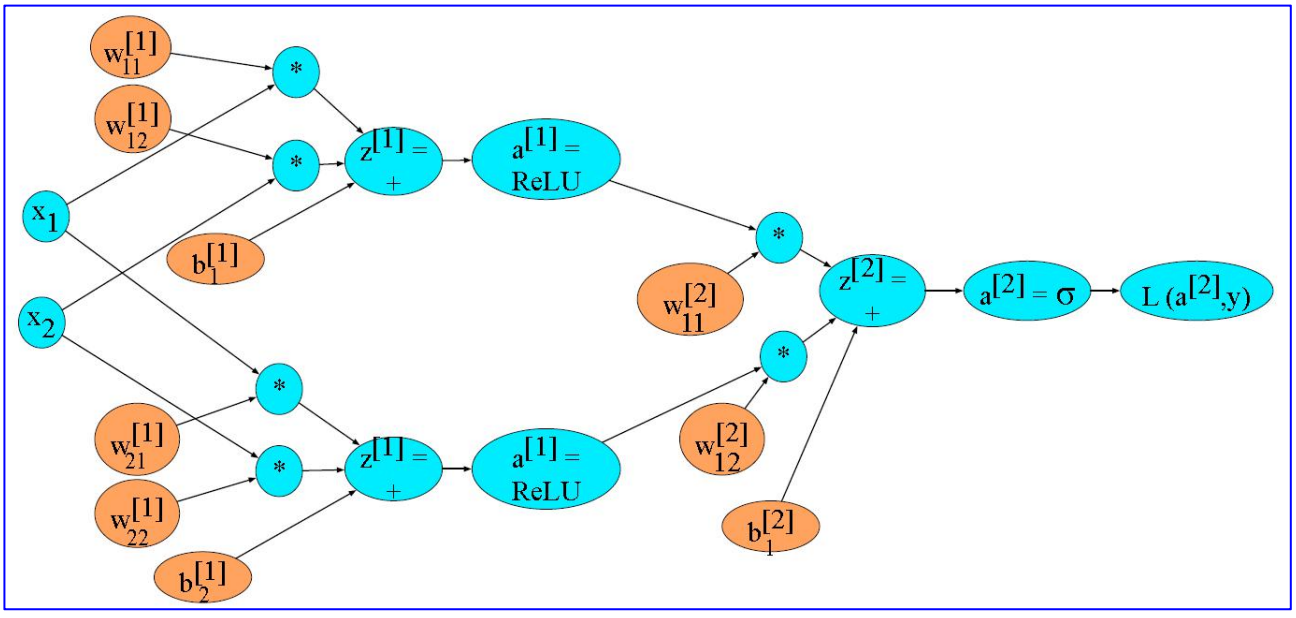


图 7-12：简单 2 层神经网络的计算图

图注：具有两个输入维度和两个隐藏维度的简单 2 层神经网络(1 个隐含层)的计算示例图。

神经网络的逆向微分

当然，真实神经网络的计算图要复杂得多。

图 7.12 显示了一个 $n_0=2, n_1=2, n_2=1$ 的两层神经网络的示例计算图，假设是二分类，因此为了简单起见使用了 sigmoid 输出单元。计算图计算的函数为:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \text{ReLU}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$
$$\hat{y} = a^{[2]} \tag{7.27}$$

需要更新的权重(我们需要知道损失函数的偏导数)用橙色表示。为了进行反向传递，我们需要知道图

中所有函数的导数。我们已经在 5.8 节看到了 sigmoid σ的导数:

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \tag{7.28}$$

我们还需要其他激活函数的导数。tanh 的导数为:

$$\frac{d\tanh(z)}{dz} = 1-\tanh^2(z) \tag{7.29}$$

ReLU 的导数是：

$$\frac{d\,\mathrm{ReLU}(z)}{dz} = \begin{cases} 0 & for \quad x<0 \\ 1 & for \quad x \geq 0 \end{cases} \tag{7.30}$$

7.4.5. 学习的更多细节

神经网络中的优化是一个非凸优化问题，比逻辑回归更复杂，并且由于这个和其他原因，有许多成功学习的最佳实践。

对于逻辑回归，我们可以初始化梯度下降，使所有的权值和偏差都为 0。而在神经网络中，我们需要用小的随机数初始化权值。将输入值归一化使其均值和单位方差为 0 也很有帮助。

使用各种形式的**正则化**来防止**过拟合**。最重要的一项是**丢弃(dropout)**：在训练期间从网络中随机丢弃一些单元及其连接(Hinton 等人 2012； Srivastava 等人 2014)。**超参数(hyperparameters)**的调整也很重要。神经网络的参数是权重 W 和偏差 b；这些是通过梯度下降来学习的。超参数是算法设计者选择的东西。最佳值是在 devset 上调整的，而不是通过训练集上的梯度下降学习来调整的。超参数包括学习率 η、小批量尺寸、模型体系结构(层数、每层隐藏节点数、激活函数的选择)、如何进行正则化等。梯度下降本身也有许多架构变体，例如 Adam(Kingma 和 Ba，2015)。

最后，大多数现代神经网络都是使用计算图形形式来构建的，这使得在基于向量的 GPU(图形处理单元)上进行梯度计算和并行化变得容易和自然。PyTorch (Paszke 等人，2017)和 TensorFlow (Abadi 等人，2015)是其中最受欢迎的两种。感兴趣的读者应该查阅神经网络教科书以了解更多的细节;在本章的最后给出了一些建议。

7.5. 神经语言模型

作为神经网络的第一个应用，让我们考虑语言建模：根据先前的单词上下文预测即将到来的单词。

基于神经网络的语言模型比第三章的 n-gram 语言模型有很多优点。其中之一是神经语言模型不需要平滑，它们可以处理更长的历史，它们可以在类似单词的上下文中进行概括。对于给定大小的训练集，神经语言模型比 n-gram 语言模型具有更高的预测精度。此外，神经语言模型是我们将介绍的用于机器翻译、对话和语言生成等任务的许多模型的基础。

另一方面，这种改进的性能是有代价的:神经网络语言模型训练起来比传统语言模型慢得多，因此对于许多任务来说，n-gram 语言模型仍然是正确的工具。

在本章中，我们将描述简单的前馈神经语言模型，这是本吉奥(Bengio 等人，2003)首先介绍的。现代神经语言模型通常不是前馈的，而是循环的，使用我们将在第 9 章中介绍的技术。

前馈神经 LM 是一种标准前馈网络，它以时间 t 处的一些先前单词(w_{t-1}，w_{t-2} 等)的表示作为输入，并输出可能的下一个单词的概率分布。因此，就像 n-gram 的 LM 一样，前馈神经 LM 通过基于 N 个先前单词进行近似，从而在给定整个现有上下文 $P(w_t \mid w_{1:t-1})$的情况下近似单词的概率：

$$P(w_t|w_1,\ldots,w_{t-1}) \approx P(w_t|w_{t-N+1},\ldots,w_{t-1}) \tag{7.31}$$

在下面的例子中，我们将使用一个 4-gram 的例子，展示一个网络来估计概率 $P(w_t = i|w_{t-1},w_{t-2},w_{t-3})$。

7.5.1. 嵌入

在神经语言模型中，先前上下文是由前一个单词的嵌入来表示的。将先前上下文表示为嵌入，而不是像 n-gram 语言模型中使用的精确词汇，这允许神经语言模型比 n-gram 语言模型更好地泛化到不可见的数据。例如，假设我们在训练中见过这样的句子:

I have to make sure that the cat gets fed.

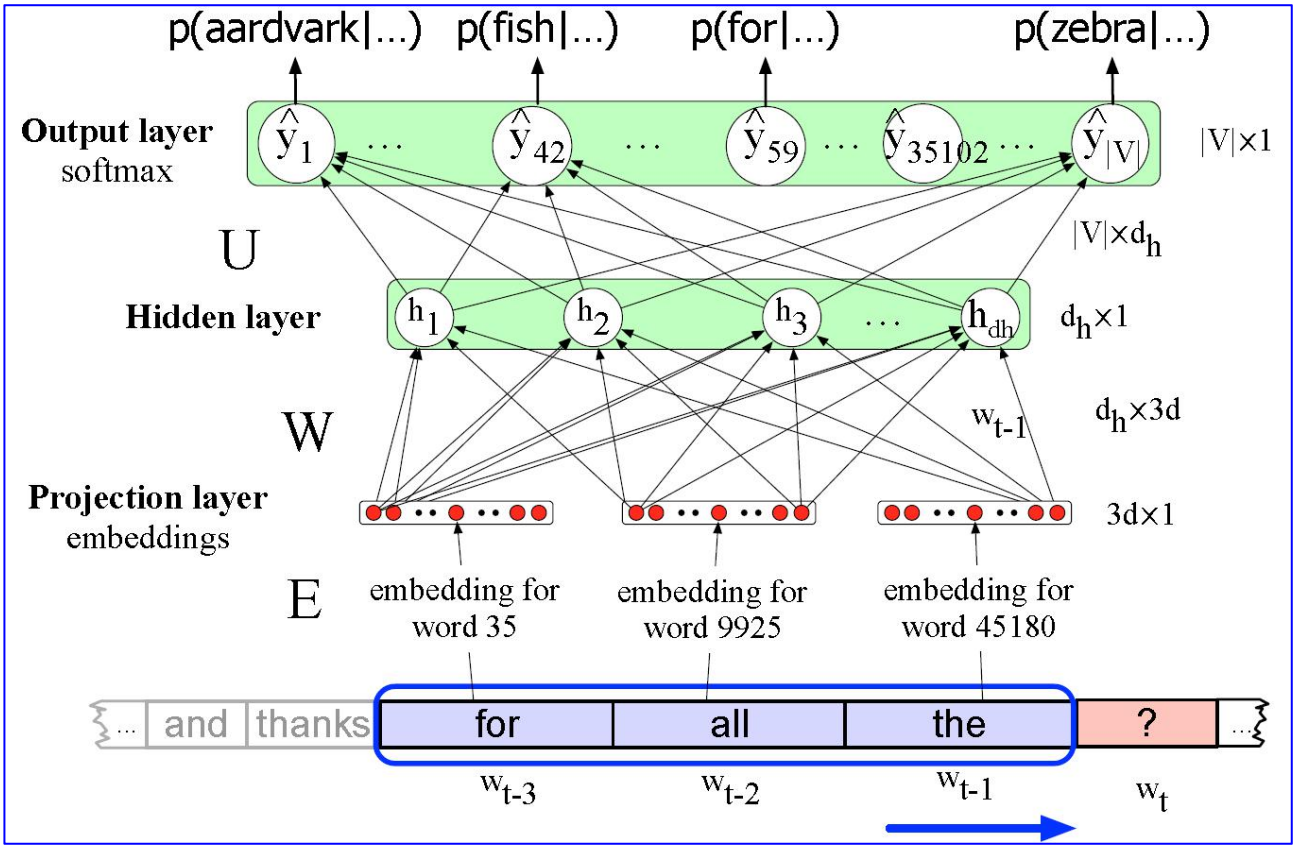


图 7-13：前馈神经语言模型的简化视图

图注：在文本中移动的前馈神经语言模型的简化视图。在每个时间步长 t，网络获取 3 个上下文词，将每个上下文词转换为 d 维嵌入，并将 3 个嵌入拼接在一起，以获得网络的 $1\times N_d$ 单元输入层 x。首先将这些单元乘以权重矩阵 W，其次逐个元素应用激活函数以生成隐藏层 h，然后将其乘以另一个权重矩阵 U。最后，softmax 输出层在每个节点 i 预测以下概率：下一个单词 w_t 将是词汇表 V 中的单词 V_i。(此图片是简化的，因为它假定我们只是在嵌入字典 E 中查找每个单词的 d 维嵌入向量，并通过诸如 word2vec 之类的算法预先计算得出。)

但从未见过“dog”之后的“gets fed”。我们的测试集带有前缀“I forgot to make sure that the dog gets”。下一个词是什么？一个 n-gram 语言模型将在“that the cat gets”之后预测“fed”，而不是在“that the dog gets”之后预测“fed”。但是神经 LM，知道“cat”和“dog”具有相似的嵌入，即使在看到“dog”之后，也能够从“cat”上下文中泛化出足够高的概率给“fed”。

让我们看看这在实践中是如何工作的。现在，假设我们已经有一个嵌入字典 E，它为词汇表 V 中的每个单词提供了嵌入。

图 7.13 是简化的前馈神经语言模型(N = 3)的示意图。我们在时间 t 有一个移动窗口，带有一个嵌入向量，表示三个先前单词(单词 w_{t-1}，w_{t-2} 和 w_{t-3})中的每个单词。这 3 个向量被拼接在一起以产生 x，即神经网络的输入层，其输出是 softmax，其概率分布在单词上。因此，y_{42}(输出节点 42 的值)是下一个单词 w_t 为 V_{42}(词汇表 V 中索引号为 42 的单词)的概率。

图 7.13 所示的模型是足够的，假设我们已经通过类似第 6 章的 word2vec 方法分别学习了嵌入。

如果有一种算法已经学习了输入词的嵌入表示，那么依赖于这种算法的训练就这被称为**预训练(pretraining)**。如果这些预训练的嵌入对您的目的足够了，那么这就是您所需要的。

但是，通常我们想在训练网络的同时学习嵌入。当为网络设计的任务(情感分类，翻译或解析)对构成良好表示的内容施加了严格的约束时，这是正确的。

因此，让我们展示一种允许学习嵌入的架构。为此，我们将在网络上添加一个额外的层，然后将误差一直传播到嵌入向量，首先从具有随机值的嵌入开始，然后逐渐向合理的表示形式发展。

为了使它在输入层起作用，而不是预先训练的嵌入，我们将把 N 个先前的词表示为长度为|V|的独热向量，即，词汇表中每个词的维度为 1。在独热向量中，只有一个等于 1 的元素(在维度上对应于该单词在词汇表中的索引位置)，而所有其他元素均为 0。

假设“toothpaste”是词汇表里的索引 5，$x_5 = 1$，$x_i = 0 \ \forall i \neq 5$，如下图所示:

[0 0 0 0 1 0 0 ... 0 0 0 0]
1 2 3 4 5 6 7 ... ... |V|

图 7.14 显示了 LM 训练中学习嵌入所需的额外层数。在这里，N=3 个上下文单词被表示为 3 个独热向量，通过嵌入矩阵 E 的 3 个实例完全拼接到嵌入层。

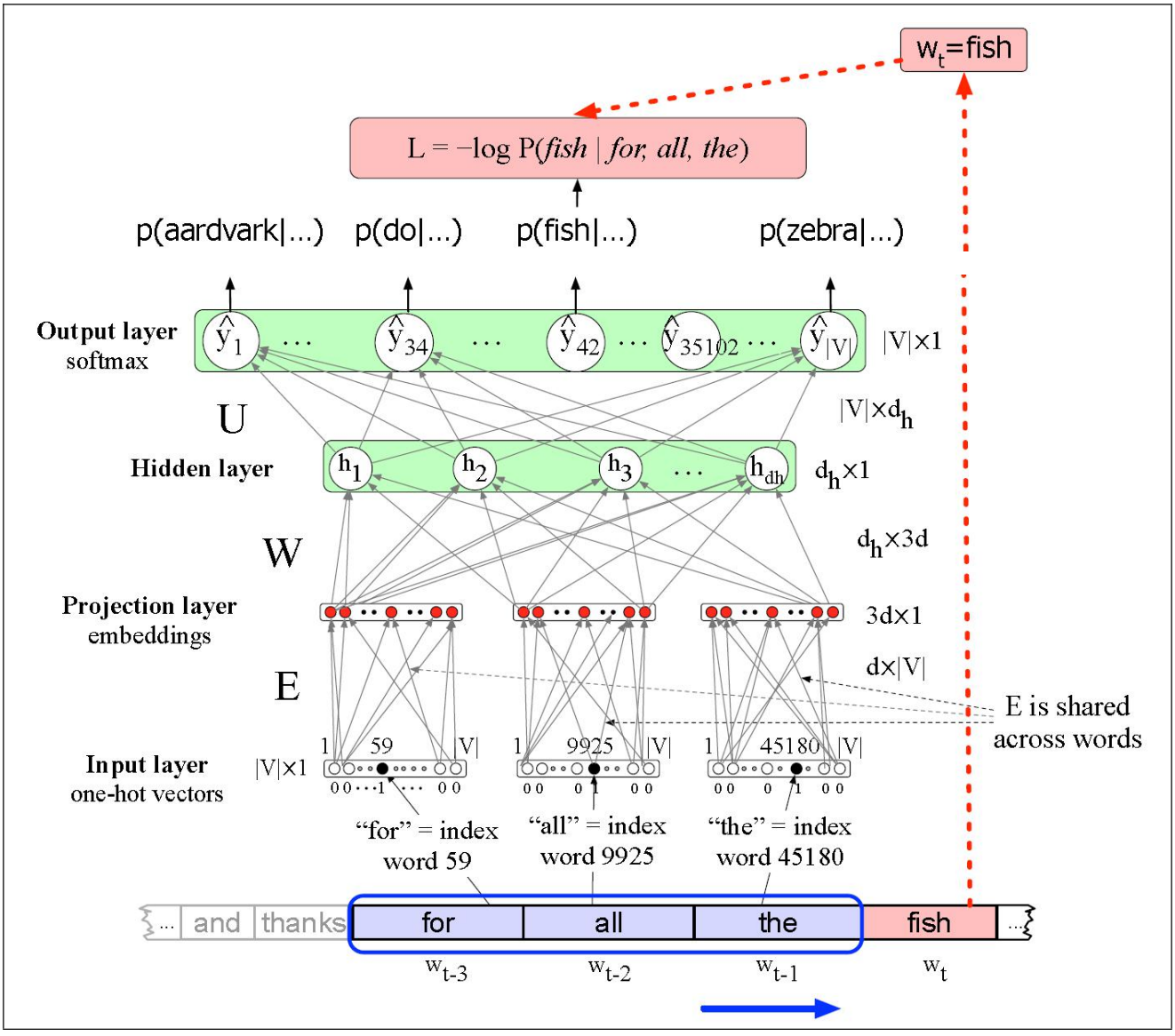


图 7-14：学习嵌入的所有方法

图注：请注意，嵌入矩阵 E 在三个上下文单词之间共享。

请注意，我们不想学习用于将前面的 3 个单词中的每个单词映射到投影层的单独的权重矩阵，我们希望在这三个单词之间共享一个嵌入字典 E。这是因为随着时间的流逝，许多不同的单词将以 w_{t-2} 或 w_{t-1} 的形式出现，而我们只想用一个矢量表示每个单词，无论它出现在哪个上下文位置。因此，嵌入权重矩阵 E 的每个单词都有一列，每个列向量的维数均为 d，因此维数为 d×|V|。

让我们看一下图 7.14 的正向传递。

1.**从 E 中选择三个嵌入**：给定先前三个词，我们查找它们的索引，创建 3 个独热向量，然后将每个独热向量乘以嵌入矩阵 E。考虑 w_{t-3}。将“the”(索引 35)的独热向量乘以嵌入矩阵 E，得到第一隐藏层的第一部分，即**投影层(projection layer)**。由于输入矩阵 E 的每一行只是一个单词的嵌入，而输入是单词 V_i 的单列向量 x_i，因此输入 w 的投影层将为 $Ex_i = e_i$，即单词 i 的嵌入。然后，我们将上下文单词的三个嵌入拼接起来。

2. **乘以 W**:我们现在乘以 W(并添加 b)并通过 ReLU(或其他)激活函数得到隐藏层 h。

3. **乘以 U**:现在 h 乘以 U。

4. **应用 softmax**:在 softmax 之后，输出层的每个节点 i 估算概率 $P(w_t=i|w_{t-1},w_{t-2},w_{t-3})$。

总而言之，如果我们用 e 表示投影层，该投影层是通过将三个上下文向量的三个嵌入拼接起来而形成的，则神经语言模型的等式变为：

$$e = (Ex_1, Ex_2, ..., Ex) \tag{7.32}$$

$$h = \sigma(We + b) \tag{7.33}$$

$$z = Uh \tag{7.34}$$

$$\hat{y} = \text{softmax}(z) \tag{7.35}$$

7.5.2. 训练神经语言模型

为了训练模型，即设置所有参数θ = E,W,U,b，我们做梯度下降(图 5.5)，在计算图上使用误差反向传播来计算梯度。因此，训练不仅设置了网络的权值 W 和 U，而且当我们预测即将到来的单词时，我们学习每个单词的嵌入 E，它能最好地预测即将到来的单词。

一般来说，训练的过程是将一个很长的文本作为输入，拼接所有的句子，从随机权重开始，然后迭代地在文本中移动，预测每个单词的 w_t。对每个单词 w_t，我们使用交叉熵(负对数似然)损失。回想一下这个的一般形式(从公式 7.19 中重复得到)是:

$$L_{CE}(\hat{y}, y) = -\log \hat{y}_i, \quad \text{(where } i \text{ is the correct class)} \tag{7.36}$$

对于语言建模，类是词汇表中的单词，所以$\hat{y}_i$ 在这里表示模型赋给正确的下一个单词 w_t 的概率:

$$L_{\text{CE}} = -\log p(w_t|w_{t-1}, ..., w_{t-n+1}) \tag{7.37}$$

将此损失从步骤 s 到 s+1 的随机梯度下降的参数更新为:

$$\theta^{s+1} = \theta^s - \eta \frac{\partial -\log p(w_t|w_{t-1}, ..., w_{t-n+1})}{\partial \theta} \tag{7.38}$$

这个梯度可以在任何标准的神经网络框架中计算，然后通过θ = E,W,U,b 反向传播。

训练参数以使损失最小化将不仅产生了语言建模算法(单词预测器)，而且还产生了一组新的嵌入 E，可用作其他任务的单词表示。

7.6. 总结

•神经网络由神经单元构建而成，最初灵感来自人类**神经单元**，但现在只是一个抽象的计算设备。

•每个神经单元将输入值乘以一个权重向量，添加一个偏差，然后应用一个非线性激活函数，如 sigmoid, tanh，或矫正线性(ReLU)。

•在一个**全连接**的**前馈**网络中，第 i 层的每个单元都与第 i+1 层的每个单元相连，不存在循环。

•神经网络的力量来自于早期层学习表示的能力，这些表示可以被网络的后续层利用。

•神经网络通过**梯度下降**等优化算法进行训练。

•**误差反向传播**，**计算图**上的逆向微分，用于计算网络损失函数的梯度。

•**神经语言模型**使用神经网络作为概率分类器，计算给定前 n 个单词的下一个单词的概率。

•神经语言模型可以使用预训练的**嵌入**，也可以在语言建模过程中从零开始学习嵌入。

7.7. 文献和历史说明

神经网络起源于 20 世纪 40 年代的 McCulloch-Pitts 神经元(McCulloch 和 Pitts, 1943)，这是人类神经元的一种简化模型，可以用命题逻辑来描述。到 20 世纪 50 年代末和 60 年代初，一些实验室(包括康奈尔大学的弗兰克 • 罗森布拉特和斯坦福大学的伯纳德 • Widrow)发展了神经网络的研究;这一阶段见证了感知机的发展(Rosenblatt, 1958)，以及阈值转变为偏差，一种我们仍在使用的符号(Widrow 和 Hoff, 1960)。

当单个感知器单元无法模拟像 XOR 这样简单的功能时，神经网络领域就衰落了(Minsky 和 Papert, 1969)。虽然在接下来的 20 年里仍有少量工作在进行，但直到 20 世纪 80 年代，当建立更深入的网络(如误差反向传播)的实用工具变得广泛时，该领域才出现了重大复兴(Rumelhart 等人，1986)。在 1980 年代各种神经网络和相关架构开发，特别是应用于心理学和认知科学(Rumelhart 和 McClelland 1986b, McClelland 和 Elman 1986, Rumelhart 和 McClelland 1986a, Elman 1990),**联结主义(connectionist)**一词或并行分布式处理被经常使用(Feldman 和 Ballard 1982, Smolensky 1988)。这一时期发展起来的许多原则和技术是现代工作的基础，包括分布式表示(Hinton, 1986)、循环网络(Elman, 1990)和合成性张量的使用(Smolensky, 1990)。

到 20 世纪 90 年代，更大的神经网络也开始应用于许多实际的语言处理任务，如手写识别(LeCun 等人，1989)和语音识别(Morgan 和 Bourlard，1990)。到 21 世纪初，计算机硬件的改进以及优化和训练技术的进步，使训练更大、更深层次的网络成为可能，从而产生了现代术语“深度学习”(Hinton 等人 2006, Bengio 等人 2007)。我们将在第 9 章和第 26 章讨论更多相关的历史。

关于这个问题有许多优秀的书籍。Goldberg(2017)对自然语言处理的神经网络进行了卓越的研究。对于一般的神经网络，请参阅 Goodfellow 等人(2016)和 Nielsen(2015)。

7.8. 练习

(无)

8. 词类和命名实体的序列标签

To each word a warbling note
每一个词都有一个颤音
A Midsummer Night's Dream, V.I

亚历山大的狄奥尼修斯·萨克斯(Dionysius Thrax，公元前 100 年)，或也许是其他人(很久以前)，写了一份希腊文的语法轮廓("techné")，总结了当时的语言知识。这项工作是现代语言词汇量惊人的来源，其中包括句法(syntax)、双元音(diphthong)、附着词(clitic)和类推(analogy)。还包括八个**词类(parts of speech，POS)**的描述：名词，动词，代词，介词，副词，连词，分词和冠词。尽管早期的学者(包括 Aristotle 和 Stoics)都有自己的词类清单，但正是 Thrax 的这八个词类成为后来 2000 年欧洲语言描述的基础。(一直到我们童年时期的 Schoolhouse Rock 教育电视节目播放，其中就有关于八个词类的歌曲，例如已故的 Bob Dorough 的《连接词的连接》)。两千年来，词类的经久不衰说明了它们在人类语言模型中的中心地位。

专有名称是另一个重要且古老的语言学类别。虽然词类通常分配给单词或词素，但专有名称通常是一个完整的多词短语，例如名称"玛丽·居里"，"纽约市"或组织"斯坦福大学"。粗略地说，我们将使用**命名实体(named entity)**一词来指代任何可以使用专有名称的东西：一个人，一个位置，一个组织，尽管我们将看到，这个术语通常被扩展到包括本质上不是实体的东西。

词类(POS)和命名实体是了解句子结构和意义的有用线索。知道一个词是名词还是动词告诉我们可能的邻近词(英语中的名词前面有限定词和形容词，动词前面有名词)和句法结构(动词与名词有依存关系)，使词类标注成为句法分析的一个关键方面。知道一个命名实体(如华盛顿)是一个人的名字、一个地方的名字还是一所大学的名字，对于许多自然语言理解任务(如回答问题、立场检测或信息提取)来说非常重要。

在本章中，我们将介绍词类标注的任务，获取单词序列并为每个单词分配词类(如名词 NOUN 或动词 VERB)，以及**命名实体识别(NER)**的任务，为词或短语指定标签，例如 PERSON(人名)，LOCATION(地名)或 ORGANIZATION(组织名)。

我们为输入单词序列中的每个单词 x_i 分配标签 y_i，以使输出序列 Y 与输入序列 X 具有相同长度的此类任务称为**序列标签(sequence labeling)**任务。我们将介绍经典的序列标记算法，一种是**生成式--隐马尔可夫模型(HMM)**，另一种是**区分式--条件随机场(CRF)**。在接下来的章节中，我们将介绍基于 RNN 和 Transformers 的现代序列标记器。

8.1. (大部分)英语单词的分类

直到现在，我们一直在相当自由地使用**名词(noun)**和**动词(verb)**等词类术语。在本节中，我们给出更完整的定义。虽然词类确实有语义倾向——例如，形容词经常描述 properties(属性)和名词经常描述 people(人)——词类的定义是根据它们与相邻单词的语法关系或词缀的形态学属性。

词类分为两大类:**封闭类(closed class)**和**开放类(open class)**。封闭类是那些成员关系相对固定的类，例如介词——新介词很少被创造出来。相比之下，名词和动词是开放类——像 iPhone 或 fax 这样的新名词和动词不断被创建或借用。封闭类词通常是**虚词(function words)**，如 of, it, and, 或者 you，这些词往往很短，经常出现，在语法中经常有结构化用法。

世界上的语言有四种主要的开放类：名词(包括专有名词)、动词、形容词和副词，以及较小的感叹词开放类。英语拥有全部五种开放类，尽管并非每种语言都有。

名词(nouns)是指人、地点或事物的词，但也包括其他词。**普通名词(common nouns)**包括像 cat 和 mango 这样的具体术语，像 algorithm 和 beauty 这样的抽象术语，以及类似动词的术语，就像 pacing 在句子 His pacing to and fro became quite annoying 之中。英语中的名词可能与限定词(a goat, its bandwidth)带所有格(IBM's annual revenue)一起出现，并且可能以复数形式出现(goats, abaci)。包括英语在内的许多

语言都将普通名词分为**可数名词(count nouns)**和**物质名词(mass nouns)**。可数名词可以单数和复数形式出现(goat/goats, relationship/relationships)，也可以计数(one goat, two goats)。当某物被概念化为一个同质群体时，就使用物质名词。所以 snow, salt, 和 communism 是不可数名词(不能说*two snows 或*two communisms)。**专有名词(proper noun)**，如 Regina、Colorado 和 IBM，是特定人员或实体的名称。

	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	PUNCT	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

图 8-1：英语单词的两大类和 17 个词类

图注：**通用依赖(Universal Dependencies，UD)**标记集中的 17 个词类(Nivre 等人，2016a)。可以添加特性来进行更细粒度的区分(使用诸如数字、大小写、确定性等属性)。

动词(verb)指的是动作和过程，包括主要的动词，如 draw, provide 和 go。英语动词有变化(非第三人称单数 eat，第三人称单数 eats，进行时 eating，过去分词 eaten)。虽然许多学者认为所有的人类语言都有名词和动词的类别，但也有人认为有些语言，比如廖内印尼语(Riau Indonesian)和汤加语(Tongan)，甚至没有做出这种区分(Broschart 1997; Evans 2000; Gil 2000)。

形容词(adjective)通常描述名词的属性或性质，如颜色(white, black)、年龄(old, young)和价值(good, bad)，但有些语言没有形容词。例如，在韩语中，英语形容词对应的单词是动词的一个子类，所以英语中的形容词“beautiful”在韩语中就像动词一样，意思是“to be beautiful”。

副词(adverb)是一种大杂烩。本例中所有斜体字都是副词:

Actually, I ran *home extremely quickly yesterday*

副词通常修饰某物(通常是动词，因此称为副词，但也包括其他副词和整个动词短语)。**方向(directional)**副词或**位置(locative)**副词(home, here，downhill)表示某个动作的方向或位置；**程度(degree)**副词(extreme, very, somewhat)表示某种行为、过程或性质的程度；**方式(manner)**副词(slowly, slinkily, delicately)描述某种行为或过程的方式；**时间(temporal)**副词用来描述某个动作或事件发生的时间(yesterday, Monday)。

感叹词(interjections)(oh, hey, alas, uh, um)是一个规模较小的公开类，也包括问候语(hello, goodbye)和问题回答(yes, no, uh-huh)。

介词(preposition)出现在名词之前，因此称为介词。它们可以表示空间或时间关系，无论是字面上的(on it, before then, by the house)或隐喻上的(on time, with gusto, beside herself)，以及像 Hamlet was written by Shakespeare 中所写的标记施事者这样的关系。

小品词(particle)类似于介词或副词，可与动词连用。小品词通常具有与它们所类似的介词不同的扩展含义，如 she turned the paper over 中的小品词 over。动词和小品词作为一个单独的单元叫做**短语动词**

(phrasal verb)。短语动词的意义通常是非组合的——不能从动词和小品词的单独意义来预测。因此，turn down 意为 reject (拒绝)，rule out 意为 eliminate(排除)，go on 意为 continue(继续)。

限定词(determiner)像 this 和 that(this chapter, that page)可以标记一个英语名词短语的开始。**冠词(article)**如 a、an、the 等是一类标示名词话语属性的限定词，其使用频率很高：the 是书面语中最常见的单词，紧跟其后的是 a 和 an。

连词(conjunction)连接两个短语、从句或句子。**协调**连词如 and, or, 以及 but 连接两个同等地位的元素。当其中一个元素具有某种嵌入状态时，使用**从属**连词。例如，在“I thought that you might like some milk”中的从属连词 that 将主句 I thought 与从句 you might like some milk 连接起来。这个从句之所以被称为从属句，是因为整个从句是主要动词 thought 的“内容”。以这种方式将动词连接到其论元(argument)的从属连词(如 that)也称为**补语(complementizer)**。

代词(pronoun)是指实体或事件的简写。**人称(personal)代词**指的是人或实体(you, she, I, it, me, 等)。**主有(possessive)代词**是人称代词的形式，可以表示实际的占有，也可以更多地表示人和某物(my, your, his, her, its, one's, our, their)之间的抽象关系。**Wh 代词**(what, who, whom, whom)用于某些疑问形式，或者作为补语(Frida, who married Diego…)。**辅助(auxiliary)**动词标示主要动词的语义特征，如时态、是否完成(体)、是否否定(极性)、动作是必要的、可能的、建议的还是需要的(情绪)。英语中的辅助动词包括**系动词(copula verb)**be、两个动词 do 和 have、以及**情态动词(modal verb)**，这些情态动词用来表示与主要动词所描述的事件有关的语气：can 表示能力或可能性，may 表示允许或可能, must 表示必要性。

一个特定于英语的标记集，45 个标记的 Penn Treebank 标记集(Marcus 等人，1993)，如图 8.2 所示，已经被用于标记许多句法注释语料库，如 Penn Treebank 语料库，因此值得了解。

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	“to”	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential ‘there’	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*’s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one’s*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

图 8-2：Penn Treebank 词类标记

下面我们展示一些根据 UD 和 Penn 标记集来标记每个单词的例子。请注意，Penn 标记集区分了动词的时态和分词，并且有一个特殊的标记用于英语中存在式 there 结构。注意，由于 New England Journal of Medicine(《新英格兰医学杂志》)是一个专有名词，两个标记集都将其组成名词标记为 NNP，包括 journal 和 medicine，否则可能被标记为普通名词(NOUN/NN)。

(8.1) There/PRO/EX are/VERB/VBP 70/NUM/CD children/NOUN/NNS there/ADV/RB ./PUNC/.

(8.2) Preliminary/ADJ/JJ fifindings/NOUN/NNS were/AUX/VBD reported/VERB/VBN in/ADP/IN today/NOUN/NN’s/PART/POS New/PROPN/NNP England/PROPN/NNP Journal/PROPN/NNP of/ADP/IN Medicine/PROPN/NNP

8.2. 词类标记

词类标记(part-of-speech tagging)是为文本中的每个单词分配词类的过程。输入是(符记化的)单词和标签集的序列 x_1，x_2，…，x_n，输出是标签的序列 y_1，y_2，…，y_n，每个输出 y_i 确切对应于一个输入 x_i ，如图 8.3 直观所示。

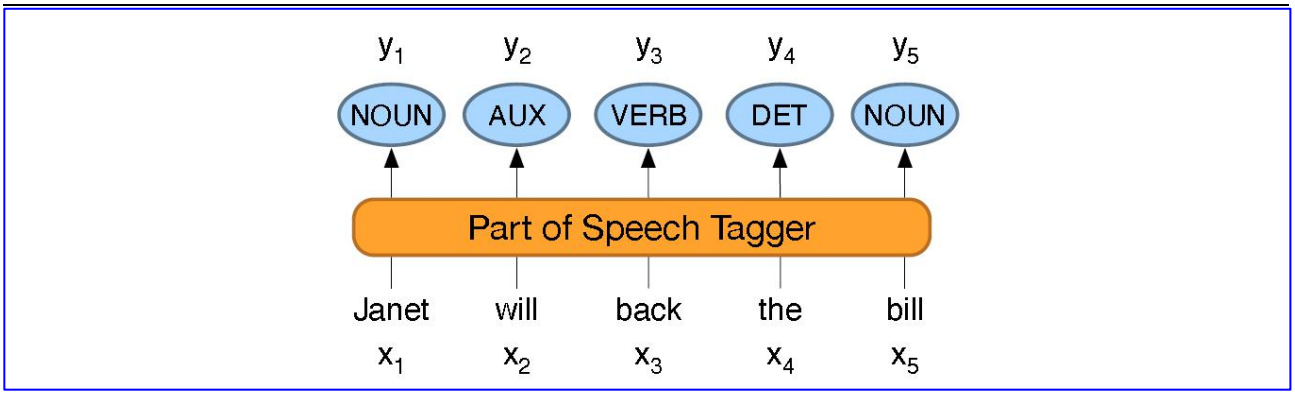


图 8-3：词类标记的任务

图注：从输入单词 x_1，x_2，...，x_n 映射到输出 POS 标记 y_1，y_2，...，y_n。

标记是一项**消除歧义(disambiguation)**的任务;单词是有**歧义(ambiguous)**的——有不止一种可能的词类——我们的目标是为这种情况找到正确的标记。例如，book 可以是动词(book that flight)，也可以是名词(hand me that book)。That 可以是一个限定词(Does that flight serve dinner)或补语(I thought that your flight was earlier)。POS 标记的目标是**消解(resolve)**这些歧义，为上下文选择适当的标记。

词类标记算法的准确性(匹配人类黄金标签的测试集标记的百分比)非常高。一项研究发现，通用依赖树库中 15 种语言的准确性超过 97% (Wu 和 Dredze, 2019)。各种英文树库的准确率也达到 97%(不管哪种算法；HMMs，CRFs, BERT 的表现也差不多)。97%的数字也反映了人类在这项任务中的表现，至少对英语 (Manning, 2011)来说是这样的。

Types:		WSJ	Brown
Unambiguous	(1 tag)	44,432 (86%)	45,799 (85%)
Ambiguous	(2+ tags)	7,025 (14%)	8,050 (15%)
Tokens:			
Unambiguous	(1 tag)	577,421 (45%)	384,349 (33%)
Ambiguous	(2+ tags)	711,780 (55%)	786,646 (67%)

图 8-4：标记歧义

图注：Brown 和 WSJ 语料库中的标记歧义(Treebank-3 中 45-标记的标记集)。

在接下来的几节中，我们将介绍该任务的算法，但首先让我们探索一下该任务。到底有多难？图 8.4 表明大多数单词类型(85-86％)是明确的(Janet 总是 NNP，hesitantly 总是 RB)。但是，歧义词虽然只占词汇量的 14-15％，但却很常见，而在运行文本中，有 55-67％的单词符记是歧义的。特别有歧义的通用词包括：back，down，put 和 set；以下是单词 back 的 6 个不同词类的示例：

earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** debt
I was twenty-one **back/RB** then

尽管如此，许多单词很容易消除歧义，因为它们不同的标记的可能性并不相同。例如，a 可以是限定词，也可以是字母 a，但限定词的意义更有可能是 a。

这个想法提出了一个有用的基准：给定一个有歧义的单词，选择在训练语料库中**最频繁(most frequent)**的标记。这里有一个关键概念:

最频繁的类基线(Most Frequent Class Baseline)：始终将分类器与基准进行比较，至少与最频繁的分类基准一样好(将每个符记分配给最常在训练集中出现的类)。

最频繁的标记基线的准确度约为 92%[1]。因此，基线与现有技术水平(SOTA)和人类最高水平(97%)的差异仅为 5%。

8.3. 命名实体和命名实体标记

词类标注可以告诉我们，像 Janet、Stanford University 和 Colorado 这样的词都是专有名词;专有名词是这些词的语法属性。但从语义的角度来看，这些专有名词指的是不同类型的实体:Janet 是一个人，Stanford University 是一个组织，…Colorado 多是一个地点。

概括地说，命名实体是可以用专有名称引用的任何东西：一个人，一个位置，一个组织。**命名实体识别(named entity recognition，NER)**的任务是查找构成专有名称的文本范围并标记实体的类型。最常见的是四个实体标签：**PER**(人员)，**LOC**(位置)，**ORG**(组织)或 **GPE**(地缘政治实体)。但是，术语“实体”通常会扩展为包括本身不是实体的事物，包括日期，时间和其他类型的时间表达式，甚至包括价格之类的数字表达式。这是 NER 标记器的输出示例：

Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flflights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as[LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

文本中有 13 次提及命名实体：5 个组织，4 个地点，2 个时间，1 个人，1 次钱。图 8.5 显示了典型的通用命名实体类型。许多应用程序还需要使用特定的实体类型，如蛋白质、基因、商业产品或艺术品。

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

图 8-5：通用命名实体类型及其引用

命名实体标记是许多自然语言理解任务中有用的第一步。在情感分析中，我们可能想了解消费者对特定实体的情感。实体是回答问题或将文本连接到诸如 Wikipedia 之类的结构化知识源中的信息的有用的第一步。而且，命名实体标记对于构建语义表示的自然语言理解任务(如提取事件和参与者之间的关系)也是至关重要的。

词类标注与命名实体识别的不同：词类标注不会因每个单词获得一个标记而产生分段(segmentation)问题；命名实体识别的任务是查找并标记文本的片段(span)，这在一定程度上是困难的，部分原因是分段的含糊性。我们需要确定什么是实体，什么不是实体，以及边界在哪里。实际上，文本中的大多数单词都不会被命名为实体。另一个困难是由类型歧义引起的。提及**肯尼迪(JFK)**可以指一个人、纽约的机场或美利坚合众国周围的任何学校、桥梁和街道。图 8.6 给出了这种交叉类型混淆的一些示例。

[PER Washington] was born into slavery on the farm of James Burroughs.
[ORG Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [LOC Washington] for what may well be his last state visit.
In June, [GPE Washington] passed a primary seatbelt law.

图 8-6：使用 Washington 名称时类型歧义的示例

像 NER 这样的跨度识别问题的序列标记的标准方法是 BIO 标记(Ramshaw 和 Marcus，1995)。这种方法使我们可以通过捕获边界和命名实体类型的标签，将 NER 视为逐字序列标记任务。考虑以下句子：

[PER **Jane Villanueva**] of [ORG **United**] , a unit of [ORG **United Airlines Holding**] ,
said the fare applies to the [LOC **Chicago**] route.

[1] 在《华尔街日报》语料库的英语部分，测试了 22-24 部分。

图 8.7 显示了用 BIO 标记表示的相同摘录，以及称为 IO 标记和 BIOES 标记的变体。在 BIO 标记中，我们将用标签 B 来标签任何一个符记处于一个片段(span)的开始(Begin)，用标签 I 来标签任何一个符记处于一个片段的内部(Inside)，用标签 O 来标签任何一个符记处于一个片段的外部(Outside)。虽然只有一个 O 标签，但每个命名实体类都将有不同的 B 和 I 标签。因此，标记的数量为 2n + 1 个，其中 n 是实体类型的数量。BIO 标记可以表示与括号符号完全相同的信息，但是具有的优势是，我们可以用与词类标记相同的简单序列建模方式来表示任务：为每个输入词 x_i 分配单个标记 y_i：

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

图 8-7：NER 序列模型的 IO-BIO-BIOES 标记

我们还展示了两种变体标记方案：IO 标记，它通过消除 B 标记而丢失了一些信息； BIOES 标记，它在片段的结尾处(End)添加了结束标记 E，而在仅包含一个字的片段添加了范围标记 S。序列标签器(HMM、CRF、RNN、Transformer 等)被训练去给文本中的每个符记贴上标签，这些标记表示特定类型的命名实体的存在(或不存在)。

8.4. HMM 词类标记

在本节中，我们介绍第一个序列标记算法，即隐马尔可夫模型，并展示如何将其应用于词类标记。回想一下，序列标记器是一个模型，其工作是将标记分配给序列中的每个单元，从而将观察序列映射到相同长度的标记序列。HMM 是经典模型，它引入了序列建模的许多关键概念，我们将在更现代的模型中再次看到它们。

HMM 是一种概率序列模型：给定一个单元序列(单词、字母、语素、句子，等等)，它计算可能的标签序列的概率分布，并选择最佳的标签序列。

8.4.1. 马尔可夫链

HMM 基于增强马尔可夫链。**马尔可夫链(Markov chain)**是一个模型，它告诉我们有关随机变量，状态序列的概率的信息，每个状态都可以从集合中取某个值。这些集合可以是单词，标记或表示任何内容(例如天气)的符号。马尔可夫链有一个很强的假设，即如果我们要按顺序预测未来，那么重要的是当前状态。当前状态之前的所有状态都不会影响未来，除非通过当前状态。假如预测明天的天气，您可以检查今天的天气，但不允许您查看昨天的天气。

更正式地，考虑状态变量 q_1，q_2，...，q_i 的序列。马尔可夫模型将**马尔可夫假设(Markov assumption)**体现在此序列的概率上：在预测未来时，过去并不重要，只有现在重要。

$$\textbf{Markov Assumption:}\quad P(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1}) \tag{8.3}$$

图 8.8a 显示了一个马尔可夫链，用于为一系列天气事件分配概率，其词汇表由 HOT，COLD 和 WARM 组成。状态在图中表示为节点，转移及其概率表示为边。转移是概率：离开(leaving)给定状态的弧的值必须总和为 1。图 8.8b 显示了一个马尔可夫链，用于将概率分配给单词 w_1 ... w_t。这个马尔可夫链应该很熟悉。实际上，它代表了一种 bigram 语言模型，每个边都表达了概率 $p(w_i|w_j)$！给定图 8.8 中的两个模型，我们可以为词汇表中的任何序列分配一个概率。

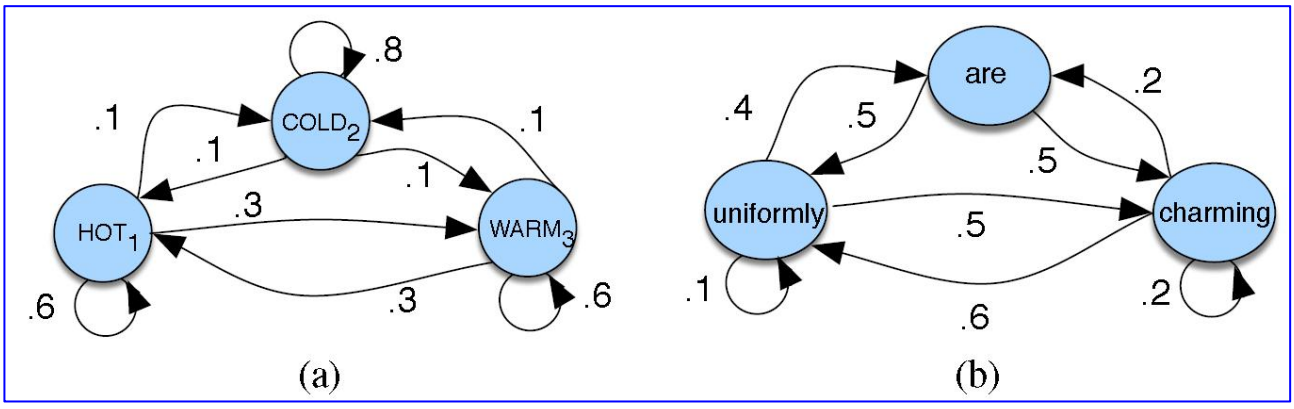


图 8-8：天气和单词的马尔可夫链

图注：用于天气(a)的马尔可夫链和用于单词(b)的马尔可夫链，显示状态和转换。起始分布 π 是必需的；将(a)设置为 π = [0.1，0.7，0.2]表示从状态 2(cold 开始的概率为 0.7)，从状态 1(hot)开始的概率为 0.1，依此类推。

形式上，马尔可夫链是由以下组成部分指定的:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

在你继续之前，使用图 8.8a 中的样本概率(π =[0.1,0.7,0.2])来计算以下每个序列的概率:

(8.4) hot hot hot hot

(8.5) cold hot cold hot

这些概率的差异告诉您图 8.8a 中编码的真实天气事实吗？

8.4.2. 隐马尔可夫模型

当我们需要计算一系列可观察事件的概率时，马尔可夫链很有用。但是，在许多情况下，我们感兴趣的事件是隐藏的：我们不会直接观察它们。例如，我们通常不会在文本中观察到词类标记。相反，我们看到单词，并且必须从单词序列中推断出标记。我们称这些标记为**隐藏(hidden)**标记，因为它们未被观察到。

隐马尔可夫模型(Hidden Markov Model，HMM)允许我们讨论观察到的事件(例如我们在输入中看到的单词)和隐藏事件(例如词类标记)，我们将它们视为概率模型中的因果关系。HMM 由以下组件指定：

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

一阶隐马尔可夫模型实例化了两个简化假设。

第一，与一阶马尔可夫链一样，特定状态的概率仅取决于之前的状态:

$$\textbf{Markov Assumption:} \quad P(q_i|q_1,...,q_{i-1}) = P(q_i|q_{i-1}) \tag{8.6}$$

第二，输出观测值 o_i 的概率仅取决于已产生的(produced)观测值 q_i 的状态，而不取决于任何其他状态或任何其他观测值：

$$\textbf{Output Independence:} \quad P(o_i|q_1,\ldots q_i,\ldots,q_T,o_1,\ldots,o_i,\ldots,o_T) = P(o_i|q_i) \tag{8.7}$$

8.4.3. HMM 标记器的成分

让我们从查看 HMM 标记器的各个部分开始，然后我们将看到如何使用它进行标记。HMM 有两个成分：A 和 B 概率。

A 矩阵包含标记**转换概率** $P(t_i \mid t_{i-1})$，表示给定先前标记的情况下标记出现的概率。例如，情态动词 will 很可能紧跟动词的基本形式 VB(例如 race)，因此我们希望这种概率很高。

我们通过计数(在标签语料库中看到第一个标记的次数中)第一个标记后面跟随第二个标记的频率，来计算此转移概率的最大似然估计：

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})} \tag{8.8}$$

例如，在 WSJ 语料库中，若情态动词发生 13124 次，其后是 VB 出现 10471 次，则 MLE 估计为：

$$P(VB|MD) = \frac{C(MD,VB)}{C(MD)} = \frac{10471}{13124} = .80 \tag{8.9}$$

在返回到解码算法之前，让我们浏览一个示例，看看如何估计这些概率并在示例标记任务中使用。

在 HMM 标记中，概率是通过计算标记的训练语料库来估计的。

在本例中，我们将使用标记的 WSJ 语料库。

B **发射概率** $P(w_i|t_i)$表示，在给定标签(比如 MD)的条件下，它与给定单词(比如 will)相关联的概率。发射概率的 MLE 为：

$$P(w_i|t_i) = \frac{C(t_i,w_i)}{C(t_i)} \tag{8.10}$$

在 WSJ 语料库的 13124 例 MD 中，它与 will 有 4046 次关联:

$$P(will|MD) = \frac{C(MD,will)}{C(MD)} = \frac{4046}{13124} = .31 \tag{8.11}$$

我们在第四章看到过这种贝叶斯模型;回想一下，这个可能的术语不是问“哪个是单词 will 最有可能的标记?”这就是后验概率 P(MD|will)。相反，P(will|MD)回答了一个稍微违反直觉的问题:“如果我们要生成 MD，那么这个情态动词(modal，MD)有多大可能是 will?”

HMM 词类标记器中三个状态的 HMM 的 A 转移概率和 B **观察概率**如图 8.9 所示。完整的标记器将为每个标记赋予一个状态。

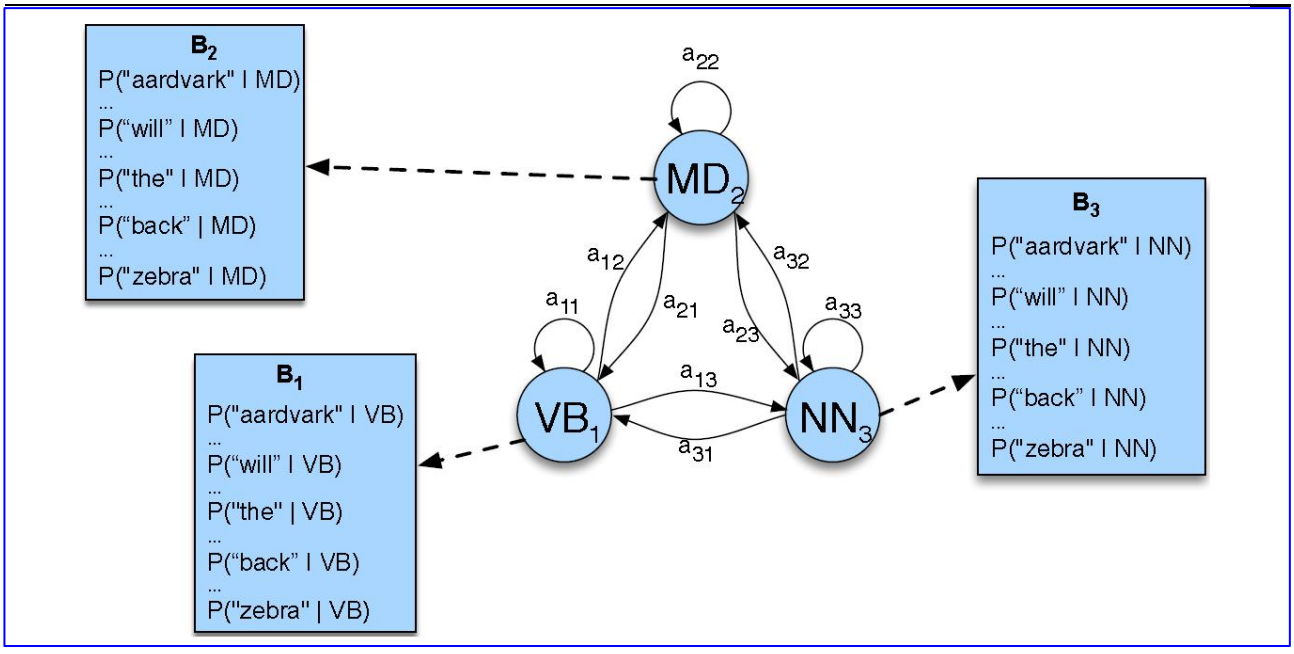


图 8-9：HMM 表示的两个部分

图注：两个部分：用于计算先验概率的 A **转移概率**，以及与每个状态相关的 B **观察概率**，每个可能的观察词都有一个似然度。

8.4.4. HMM 标记作为解码

对于任何包含隐藏变量的模型(例如 HMM)，确定与观察序列相对应的隐藏变量序列的任务称为**解码(decoding)**。更正式地说，

解码：给定 HMM λ=(A,B)和观测序列 $O=o_1,o_2,...,o_T$ 作为输入，找出状态 $Q=q_1q_2 ...q_T$ 的最可能序列。

对于词类标记，HMM 解码的目标是,在给定 n 个单词 $w_1 ... w_n$ 的观察序列的情况下，选择最有可能的标记序列 $t_1 ... t_n$：

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1 \dots t_n | w_1 \dots w_n) \tag{8.12}$$

在 HMM 中，我们将使用贝叶斯规则来代替计算:

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} \frac{P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n)}{P(w_1 \dots w_n)} \tag{8.13}$$

进一步，我们将公式 8.13 化简，去掉分母 $P(w_1^n)$:

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n) \tag{8.14}$$

HMM 标记器作了两个进一步简化的假设。

第一个假设，单词出现的概率仅取决于其自身的标记，并且与相邻的单词和标记无关：

$$P(w_1 \dots w_n | t_1 \dots t_n) \approx \prod_{i=1}^{n} P(w_i | t_i) \tag{8.15}$$

第二个假设，即 bigram 假设，一个标记的概率只依赖于前一个标记，而不是整个标记序列;

$$P(t_1 \dots t_n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}) \tag{8.16}$$

将公式 8.15 和公式 8.16 代入公式 8.14，可以得到从一个 bigram 标记器获得的最可能的标记序列:

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n) \approx \underset{t_1 \ldots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}} \tag{8.17}$$

公式 8.17 的两个部分正好对应于我们刚才定义的 **B 发射概率**和 **A 转移概率!**

8.4.5. 维特比(Viterbi)算法

HMM 的解码算法是图 8.10 所示的**维特比(Viterbi)**算法。作为动态规划的实例，维特比类似于第 2 章中的动态规划最小编辑距离算法。

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
$bestpath \leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

图 8-10：用于找到最佳标记序列的维特比算法

图注：给定观察序列和 HMM $\lambda = (A, B)$，该算法返回通过 HMM 的状态路径，该路径将最大似然分配给观察序列。

维特比算法首先建立一个概率矩阵或**格子(lattice)**，在状态图中，每个观察 o_t 用一列表示，每个状态用一行表示。因此，在单个**组合自动机(combined automaton)**中，每个列都有一个用于每个状态 q_i 的单元格。图 8.11 显示了句子 Janet will back the bill 的这个格子的直观感受。

对于格子中的每个单元(cell)，用 $v_t(j)$代表一个概率：在给定 HMM λ的条件下，在看到前 t 次观察并经过最可能的状态序列 $q_1,\ldots,q_{t-1}$ 后，HMM 处于状态 j 的概率。每个单元格 $v_t(j)$的值是通过递归地采用最可能导致我们到达此单元的路径来计算的。形式上，每个单元的概率:

$$v_t(j) = \max_{q_1,\ldots,q_{t-1}} P(q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j|\lambda) \tag{8.18}$$

我们通过取所有可能的先前状态序列 $\max_{q_1,\ldots,q_{t-1}}$ 之中的最大值来表示最可能的路径。与其他动态规划算法一样，维特比递归地填充每个单元。假设我们已经计算了在时间 t-1 处于每个状态的概率，那么我们将通过最有可能导致通往当前单元的路径扩展来计算维特比概率。对于在时间 t 的给定状态 q_j，值 $v_t(j)$计算为:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \tag{8.19}$$

为扩展前面的路径，计算 t 时刻的维特比概率，公式 8.19 中相乘的三个因子是：

$v_{t-1}(i)$	先前时间步的先前维特比路径概率
a_{ij}	从先前状态 q_i 到当前状态 q_j 的转移概率
$b_j(o_t)$	给定当前状态 j 的观测符号 o_t 的状态观测似然度

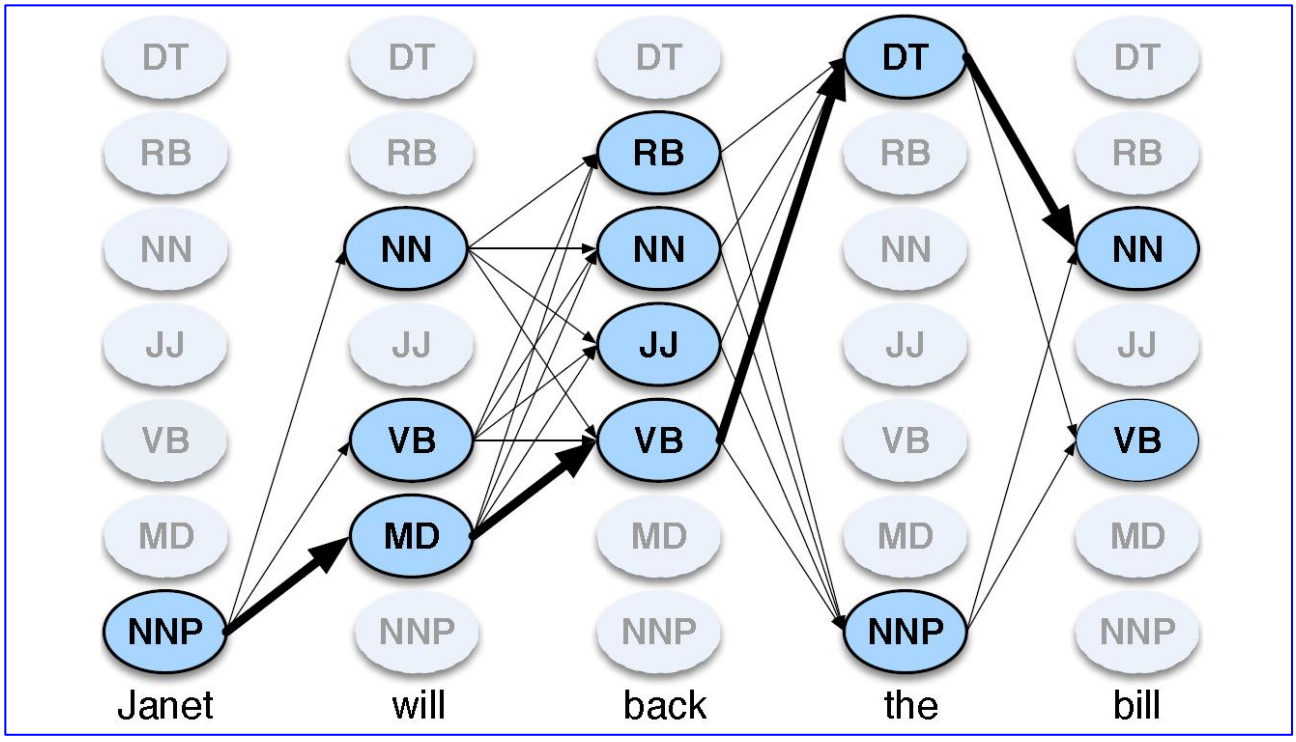


图 8-11：“Janet will back the bill”的格子示意图

图注：显示每个单词可能的标记(q_i)，并通过隐藏状态突出显示与正确标记序列相对应的路径。根据 B 矩阵生成特定单词的概率（例如，限定词 DT 将被实现为 Janet 的概率），概率为零的那些状态（词类）被标为灰色。

8.4.6. 通过一个例子

让我们标记这个句子“ Janet will back the bill”； 目标是正确的标记系列(另请参见图 8.11)：

让 HMM 由图 8.12 和图 8.13 中的两个表来定义。图 8.12 列出了隐藏状态(词类标记)之间转换的 a_{ij} 概率。图 8.13 表示了 $b_i(o_t)$概率，即给定标记的单词的观察似然度。这张表(稍微简化)来自《华尔街日报》语料库中的统计数据。因此，单词 Janet 只作为一个 NNP 出现，back 有 4 种可能的词类，单词 the 可以作为限定词出现，也可以作为一个 NNP 出现(在诸如“Somewhere Over the Rainbow”这样的标题中，所有的单词都被标记为 NNP)。

(8.20) Janet/NNP will/MD back/VB the/DT bill/NN

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

图 8-12：转移概率 A

图注：从 WSJ 语料库未经平滑计算的 A 转移概率 $P(t_i|t_{i-1})$。行被标记为条件作用事件，故 P(VB|MD) 为 0.7968。

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

图 8-13：观察似然度 B

图注：从 WSJ 语料库计算得到的观察似然度 B，不进行平滑处理，略有简化。

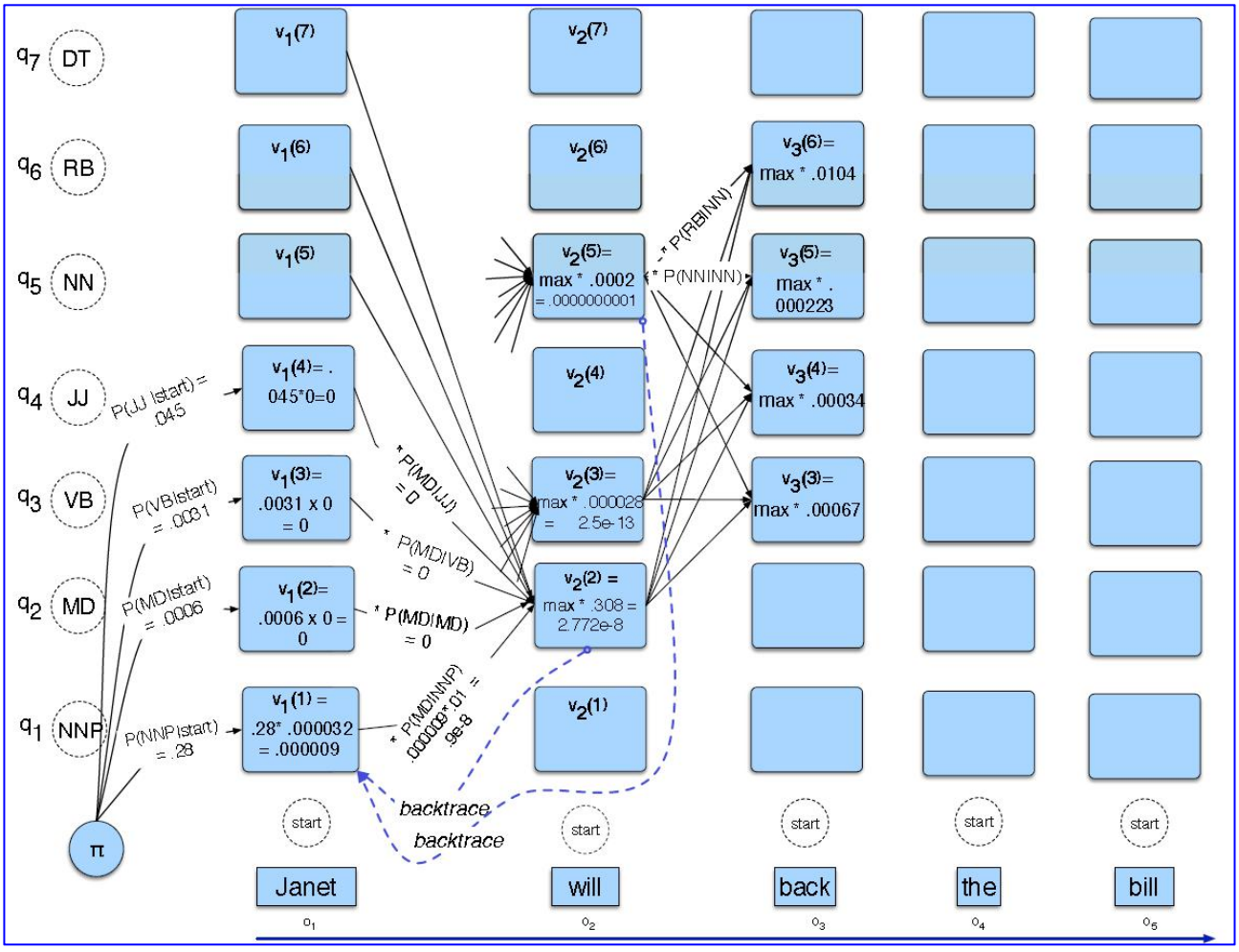


图 8-14：维特比算法示意图

图注：维特比算法的各个状态列中的前几项。每个单元保持到目前为止最佳路径的概率，并保持指向该路径上一个单元的指针。我们仅填写了第 1 列和第 2 列；为了避免混乱，将大多数值为 0 的单元留空。剩下的作为练习留给读者。单元填充完毕后，从结束状态回溯，我们应该能够重建正确的状态序列 NNP MD VB DT NN。

图 8.14 显示了我们在图 8.11 中看到的草图的充实版本，该维特比格子用于为观察序列“ Janet will back the bill”计算最佳隐藏状态序列。

有 N = 5 个状态列。我们从第 1 列(针对单词 Janet)开始，设置 Viterbi 值：转移概率与观察似然度的乘积。其中，转移概率来自图 8.12 的行值；观察似然度来自图 8.13 的列值。因为观察似然度中有很多零，所以 Viterbi 值也有很多零，读者应该在图 8.14 中找到这一点。

接下来，将更新 will 列中的每个单元。对于每个状态，我们根据公式 8.19 计算从上一列到当前单元的所有路径扩展的最大值，从而计算 viterbi[s,t]的值。我们已经显示了 MD、VB 和 NN 单元的值。每个单元从前一列中获得 7 个值中的最大值，乘以适当的转移概率；正如在本例中所发生的那样，它们中的大多数都是前一列中的零。剩下的值乘以相关的观察似然度，取(微不足道的)最大值。在本例中，最终值 2.772e-8 来自前一列的 NNP 状态。读者应该填入图 8.14 中格子的其余部分，然后回溯，看看 Viterbi 算法是否返回黄金状态序列 NNP MD VB DT NN。

8.5. 条件随机场(CRF)

虽然 HMM 是一个有用和强大的模型，但它需要大量的增强来实现较高的精度。例如，在词类标注中，就像在其他任务中一样，我们经常会遇到**未知单词(unknown words)**:专有名称和首字母缩写经常出现，甚至新的常见名词和动词也以惊人的速度进入语言。可能有办法添加任意特征来帮助解决此问题，这可能很棒，也许基于大写或词法(以大写字母开头的单词很可能是专有名词，以-ed 结尾的单词倾向于过去时态

(VBD 或 VBN)， 等等)，或者知道前一个或后一个单词可能是一个有用的特征(如果前一个单词是 the，则当前标记不太可能是动词)。

尽管我们可以尝试破解 HMM，想办法把这些东西都整合进去，但一般来说，HMM 之类的生成模型很难以干净的方式直接向模型中添加任意特征。我们已经看到了一种以原理的方式组合任意特征的模型：对数线性模型，如第 5 章的逻辑回归模型！但是逻辑回归不是序列模型，它为单个观察分配一个类。

幸运的是，有一个基于对数线性模型的区分式序列模型：**条件随机场(conditional random field，CRF)**。我们将在此处介绍线性链 CRF，最常用于语言处理的 CRF 版本，以及条件与 HMM 紧密匹配的 CRF 版本。

假设我们有一组输入单词 X =x_1^n = $x_1 \ldots x_n$，并想计算一个输出标记序列 Y = y_1^n = $y_1 \ldots y_n$。

在 HMM 中，为了计算最大化 P(Y|X)的最佳标记序列，我们依赖于贝叶斯规则和似然度 P(X|Y):

$$\begin{aligned}\hat{Y} &= \operatorname*{argmax}_Y p(Y|X) \\ &= \operatorname*{argmax}_Y p(X|Y)p(Y) \\ &= \operatorname*{argmax}_Y \prod_i p(x_i|y_i) \prod_i p(y_i|y_{i-1}) \end{aligned} \tag{8.21}$$

在 CRF 中，我们直接计算后验 p(Y|X)，训练 CRF 来区分可能的标记序列：

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}} P(Y|X) \tag{8.22}$$

但是，CRF 不会在每个时间步为每个标签计算概率。取而代之的是，CRF 在每个时间步上都会计算一组相关特征的对数线性函数，并将这些局部特征汇总并归一化以产生整个序列的全局概率。

让我们更正式地介绍 CRF，再次使用 X 和 Y 作为输入和输出序列。CRF 是对数线性模型，它在给定整个输入(单词)序列 X 的情况下，为所有可能的整个输出(标记)序列 Y 分配概率。我们可以把 CRF 看作是多元逻辑回归对单个符记所做的工作的一个巨大版本。回想一下，规则多元逻辑回归中的特征函数 f 将符记 x 和标签 y 的元组映射到特征向量中。在 CRF 中，函数 F 将整个输入序列 X 和整个输出序列 Y 映射到特征向量。假设我们有 K 个特征，每个特征的权重为 w_k：

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)} \tag{8.23}$$

通常我们也可以通过将分母化为函数 Z(X)来描述同一个方程:

$$p(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \tag{8.24}$$

$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right) \tag{8.25}$$

我们将这些 K 函数称为 F_k(X，Y)**全局特征(global features)**，因为每个函数都是整个输入序列 X 和输出序列 Y 的一个属性。我们计算的方法：把它们分解成 Y 中每个位置 i 的**局部特征(local features)**之和。

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \tag{8.26}$$

线性链 CRF 中的每个局部特征 f_k 均允许使用当前输出符记 y_i，先前输出符记 y_{i-1}，整个输入字符串 X(或其任何子部分)和当前位置 i。仅取决于当前和先前输出符记 y_i 和 y_{i-1} 的约束是线性链 CRF 的特点。正如我们将看到的，这种限制使得可以使用 HMM 中高效的 Viterbi 和 Forward-Backwards 算法版本。

相比之下，通用 CRF 允许某个特征去充分利用任何输出符记，因此对于依赖远程输出符记的决策任务(如 y_{i-4})来说，这是必要的。通用 CRF 需要更复杂的推理，并且很少用于语言处理。

8.5.1. CRF POS 标记器的特性

让我们详细看看其中的一些特征，因为使用区分式序列模型的原因是它更容易合并大量的特征[2]。

同样，在线性链 CRF 中，位置 i 的每个局部特征 f_k 可以依赖于(y_{i-1}, y_i,X,i)中的任何信息。因此，一些代表常见情况的合法(legal)特征可能如下:

L{x_i=*the*, y_i=DET}.

L(y_i= PROPN, x_{i+1} = *Street*, y_{i-1} = NUM}

L{y_i=VERB, y_{i-1} =AUX}

为简单起见，我们假设所有 CRF 特征值都为 1 或 0。上面，我们明确使用符号 L{x}表示“如果 x 为 true，则为 1；否则为 0”。从现在开始，我们在定义特征时将不再使用 L，但是您可以假定每个特征都隐含地具有该特征。

尽管使用什么特征的想法是由系统设计人员手工完成的，但具体的特征是通过使用我们在第 5 章中简要提到的**特征模板(feature templates)**自动填充的。下面是一些模板，只使用来自 y_{i-1}, y_i,X,i 的信息:

$$\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$$

这些模板会自动从训练和测试集中的每个实例中填充特征集。因此，对于我们的示例 Janet/NNP will/MD back/VB the/DT bill/NN,，当 x_i 是单词 back 时，将生成以下特征，并具有值 1(我们为它们分配了任意特征数字编号):

$\mathbf{f_{3743}}$: $\mathbf{y_i = VB}$ **and** $\mathbf{x_i = back}$

$\mathbf{f_{156}}$: $\mathbf{y_i = VB}$ **and** $\mathbf{y_{i-1} = MD}$

$\mathbf{f_{99732}}$: $\mathbf{y_i = VB}$ **and** $\mathbf{x_{i-1}}$ **= will and** $\mathbf{x_{i+2}}$ **= bill**

重要的是要具有有助于未知单词的特征。最重要的特征之一就是字形(word shape)功能，它通过将小写字母映射为“*x*”，将大写字母映射为“X”，将数字映射为“d”，并保留标点符号来表示单词的抽象字母模式。因此，例如 I.M.F 将映射到 X.X.X，而 DC10-30 将映射到 XXdd-dd。第二类较短的词形特征也被使用。在这些特征中，删除了连续的字符类型，因此所有大写字母的单词都映射到 X，首字母大写字母的单词映射到 Xx，DC10-30 会映射到 Xd-d，但是 I.M.F 仍会映射到 X.X.X。前缀和后缀特征也很有用。总之，这是一些示例特征模板，可帮助您解决未知单词：

x_i contains a particular prefifix (perhaps from all prefifixes of length ≤ 2)
x_i contains a particular suffifix (perhaps from all suffifixes of length ≤ 2)
x_i's word shape
x_i's short word shape

例如，well-dressed 这个词可能会生成以下非零特征值:

prefifix(x_i) = w

prefifix(x_i) = we

suffifix(x_i) = ed

suffifix(x_i) = d

[2] 因为在 HMM 中所有的计算都是基于两个概率 P(标记|标记)和 P(单词|标记)，如果我们想在标记过程中包含一些知识来源，我们必须找到一种方法将知识编码成这两个概率中的一个。每次我们添加一个特征，我们就必须做很多复杂的条件设置，随着我们有越来越多的这样的特征，这变得越来越难。

word-shape(x_i) = xxxx-xxxxxxx

short-word-shape(x_i) = x-x

对于训练集中看到的每个单词，都会计算已知单词模板；还可以为训练中的所有单词或仅针对频率低于某个阈值的训练单词计算未知单词特征。已知单词模板和单词签名特征的结果是一个非常大的特征集。通常，使用特征临界值，如果特征值在训练集中的计数小于 5，则将其丢弃。

请记住，在 CRF 中，我们不会为每个局部特征 fk 学习权重。相反，我们首先将整个句子中每个局部特征(例如特征 f_{3743})的值相加，以创建每个全局特征(例如 F_{3743})。然后将这些全局特征乘以权重 w_{3743}。因此，为了训练和推理，即使每个句子的长度不同，也总是有 K 个权重的 K 个特征的固定集合。

8.5.2. NER 的 CRF 特征

NER 的 CRF 使用了与 POS 标记器非常相似的特征，如图 8.15 所示。

identity of w_i , identity of neighboring words
embeddings for w_i , embeddings for neighboring words
part of speech of w_i , part of speech of neighboring words
presence of w_i in a gazetteer
w_i contains a particular prefifix (from all prefifixes of length ≤ 4)
w_i contains a particular suffifix (from all suffifixes of length ≤ 4)
word shape of w_i , word shape of neighboring words
short word shape of w_i , short word shape of neighboring words
gazetteer features

图 8-15：基于特征的 NER 系统的典型特性

地名词典(gazetteer)是一个对位置特别有用的特征，它是地名词典名称的列表，通常为具有详细地理和政治信息的位置提供数百万个条目[3]。可以将其实现为二进制特征，以指示短语出现在列表中。可以使用其他相关资源，例如来自美国人口普查局的名称列表[4]，也可以使用其他实体词典，例如公司或产品列表，尽管它们可能不像地名词典那样有用(Mikheev 等人,1999)。

一个名为实体符记 L'Occitane 的样本将生成以下非零特征值(假设 L'Occitane 既不在地名词典中也不在人口普查中)。

prefifix(x_i) = L	suffifix(x_i) = tane
prefifix(x_i) = L’	suffifix(x_i) = ane
prefifix(x_i) = L’O	suffifix(x_i) = ne
prefifix(x_i) = L’Oc	suffifix(x_i) = e
word-shape(x_i) = X’Xxxxxxxx	short-word-shape(x_i) = X’Xx

图 8.16 演示了在前面的示例中添加词类标记和一些形状信息的结果。

Words	POS	Short shape	Gazetteer	BIO Label
Jane	NNP	Xx	0	B-PER
Villanueva	NNP	Xx	1	I-PER
of	IN	x	0	O
United	NNP	Xx	0	B-ORG
Airlines	NNP	Xx	0	I-ORG
Holding	NNP	Xx	0	I-ORG
discussed	VBD	x	0	O
the	DT	x	0	O
Chicago	NNP	Xx	1	B-LOC
route	NN	x	0	O
•	•	•	0	O

图 8-16：示例句子的一些 NER 特征

图注：假定芝加哥和维拉纽瓦被列为地名词典中的位置，假设特征仅取值 0 或 1，因此第一个 POS 特征将表示为 L{POS = NNP}。

[3] www.geonames.org

[4] www.census.gov

8.5.3. CRF 的推理和训练

对于给定的输入 X，我们如何找到最好的标记序列Ŷ? 我们从公式 8.22 开始:

$$\begin{aligned}\hat{Y} &= \underset{Y\in\mathcal{Y}}{\operatorname{argmax}} P(Y|X) \\ &= \underset{Y\in\mathcal{Y}}{\operatorname{argmax}} \frac{1}{Z(X)}\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \qquad (8.27) \\ &= \underset{Y\in\mathcal{Y}}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1},y_i,X,i)\right) \qquad (8.28) \\ &= \underset{Y\in\mathcal{Y}}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1},y_i,X,i) \qquad (8.29) \\ &= \underset{Y\in\mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1},y_i,X,i) \qquad (8.30)\end{aligned}$$

如上所述，我们可以忽略 exp 函数和分母 Z(X)，因为 exp 不会改变 argmax，并且对于给定的观测序列 X，分母 Z(X)是恒定的。

我们应该如何解码才能找到最佳标记序列ŷ？ 与 HMM 一样，我们将转向 Viterbi 算法，该算法之所以有效，是因为与 HMM 一样，线性链 CRF 在每个时间步均仅依赖一个先前的输出符记 y_{i-1}。

具体而言，这涉及用适当的值填充 N×T 数组，并在进行过程中保持反向指针。与 HMM Viterbi 一样，在填充表格时，我们只需从最后一列中的最大值向后跟踪指针即可检索所需的标签集。

HMM Viterbi 的必要更改只与我们如何填充每个单元格有关。回想一下公式 8.19 中 Viterbi 方程的递推步骤，对于状态 j 和时间 t ，计算 Viterbi 值如下：

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t); 1 \le j \le N, 1 < t \le T \qquad (8.31)$$

其中 HMM 的实现如下：

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \mathrm{P}(s_j \mid s_i) P(o_t \mid s_j); 1 \le j \le N, 1 < t \le T \qquad (8.32)$$

CRF 只需要对后一个公式作微小改变，用 CRF 特征替换 a 和 b 的先验概率和似然概率:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t); 1 \le j \le N, 1 < t \le T \qquad (8.33)$$

CRF 中的学习依赖于我们为逻辑回归提供的相同的监督学习算法。给定一系列观察值、特征函数和相应的输出，我们使用随机梯度下降法训练权重，以最大程度地提高训练语料库的对数似然度。线性链 CRF 的局部性质意味着可以使用 CRF 版本的前向后向算法(请参阅附录 A)来有效地计算必要的导数。与逻辑回归一样，L1 或 L2 正则化也很重要。

8.6. 评估 NER

词类标记器通过标准的准确性度量进行评估。命名实体识别器通过召回率、精度和 F1 度量进行评估。回想一下，召回率是正确标记的响应数与应该标记的总数之比； 精度是正确标记的响应数与标记的总数之比； F 度量是两者的调和均值。

为了了解两个 MT 系统 F1 得分的差异是否具有显著性差异，我们使用**配对引导**(paired bootstrap)检

验，或类似的随机化检验(第 4.9 节)。

对于命名实体，实体而不是单词是响应的单位。因此，在图 8.16 的示例中，两个实体 Jane Villanueva 和 United Airlines Holding 以及所讨论的非实体都将被视为单个响应。

这里有一个导致评估出现一些问题的事实：命名实体标记具有文本分类或词类标记等任务中不存在的分割成分。例如，一个将 Jane 而不是 Jane Villanueva 标记为人的系统会导致两个错误，即 O 的假阳性和 I-PER 的假阴性。另外，将实体用作响应单元，而将单词用作训练单元意味着训练条件和测试条件之间不匹配。

8.7. 更多细节

在本节中，我们将总结数据和模型的其余一些细节。

首先，总结数据。由于我们介绍的算法是受监督的，因此拥有贴过标签的数据对于训练和测试至关重要。存在多种用于词类标记和/或 NER 的数据集。撰写本文时，通用依赖(UD)数据集(Nivre 等人，2016b)具有 POS 标签的语料库，支持 92 种语言，英语、中文和阿拉伯语的 Penn 树库也是如此。OntoNotes 具有用英语、中文和阿拉伯语标签命名实体的语料库(Hovy 等人，2006)。命名实体标记的语料库在特定领域也可用，例如用于生物医学(Bada 等人，2012)和文学著作(Bamman 等人，2019)。

其次，总结模型，有以下几个方面。

8.7.1. 双向性

所呈现的 CRF 和 HMM 架构的一个问题是，这些模型只能从左到右运行。尽管维特比算法仍然允许当前决策受到将来决策的间接影响，但如果有关单词 w_i 的决策可以直接使用有关未来标记 t_{i+1} 和 t_{i+2} 的信息，则将提供更大的帮助。

或者，可以通过使用多次遍历将任何序列模型转换为双向模型。例如，第一遍将仅使用左侧已经消歧单词的词类特征。在第二遍中，可以使用所有单词的标记，包括右边的单词。或者，标记器可以运行两次，一次从左至右，一次从右至左。在维特比解码中，贴标器将选择两个序列的较高得分(从左到右或从右到左)。双向模型对于神经模型是相当标准的，正如我们将在第 9 章中介绍的 biLSTM 模型所看到的那样。

8.7.2. 基于规则的方法

尽管机器学习(神经或 CRF)序列模型是学术研究中的常态，但 NER 的商业方法通常基于实用的列表和规则组合，而监督的机器学习的数量则较少(Chiticariu 等人，2013)。例如，在 IBM System T 架构中，用户使用正式查询语言指定用于标记任务的声明性约束，包括正则表达式、字典、语义约束和其他操作符，系统将这些操作符编译成有效的提取器(Chiticariu 等人，2018)。

一种常见的方法是对文本进行重复的基于规则的传递，从精度很高但召回率很低的规则开始，在随后的阶段，使用机器学习方法，将第一次传递的输出考虑进去。

1. 首先，使用高精度规则来标记明确提及的实体。
2. 其次，搜索与之前检测到的名称匹配的子字符串。
3. 再次，使用特定于应用程序的名称列表来查找可能特定领域的提及。
4. 最后，应用有监督的序列标签技术，使用来自之前的阶段的标记作为附加特征。

基于规则的方法也是最早的词类标记方法。基于规则的标记器，如英语约束语法系统(Karlsson 等人 1995, voutiainen 1999)，使用了 20 世纪 50 年代和 60 年代发明的两阶段形式主义:一个包含成千上万词干条目的形态学分析器返回一个单词的所有词类。然后，对输入的句子应用数以千计的约束，以排除与上下文不一致的词类。

8.7.3. 形态丰富语言的 POS 标记

处理诸如捷克语、匈牙利语和土耳其语等形态丰富的语言时，必须增强标记算法。

这些富有成效的单词形成过程会为这些语言带来大量词汇：匈牙利语的 25 万个**单词符记(word token)**

语料库的**单词类型(word type)**是类似大小的英语语料库的两倍多(Oravecz 和 Dienes，2002)，而一千万个单词符记的土耳其语的语料库包含的单词类型是同等大小的英语语料库的四倍(Hakkani-Tur 等人，2002)。大词汇量意味着许多未知词，这些未知词会导致多种语言(包括捷克语、斯洛文尼亚语、爱沙尼亚语和罗马尼亚语)的性能显著下降(HajiHac，2000)。

高度变形的语言也比英语在单词形态上编码的信息更多，例如大小写(主格，宾格，属格)或性别(男性，女性)。因为此信息对于诸如解析和共指解析之类的任务很重要，所以形态丰富的语言的词类标记器需要用大小写和性别信息标记单词。因此，形态丰富的语言的标记集是形态标记的序列，而不是单个原始标记。这是一个土耳其语的示例，其中 izin 一词具有三种可能的形态学/词类标记和含义(Hakkani-Tur 等人，2002)：

1. Yerdeki izin temizlenmesi gerek. — iz + Noun+A3sg+Pnon+Gen
   The trace on the flfloor should be cleaned.
2. Üzerinde parmak izin kalmis¸ — iz + Noun+A3sg+P2sg+Nom
   Your fifinger print is left on (it).
3. Içeri girmek için izin alman gerekiyor. — izin + Noun+A3sg+Pnon+Nom
   You need permission to enter.

使用像 Noun + A3sg + Pnon + Gen 这样的形态学分析序列作为词类标记会大大增加词类的数量，因此标记集可以比我们在英语中看到的 50-100 个标记大 4 到 10 倍。使用如此大的标记集，需要对每个单词进行形态学分析，以生成该单词的可能形态学标记序列(词类标记)列表。标记器的作用是消除这些标记之间的歧义。由于形态分析器可以接受未知词干并仍然正确地对词缀进行分段，因此该方法还可以帮助处理未知词。

8.8. 总结

本章介绍了**词类**和**命名实体**，以及部分**词类标记**和**命名实体识别**的任务:

•语言通常具有少量的**封闭类**单词，这些单词非常频繁、模棱两可、充当**虚词**，而**开放类**单词则是**名词、动词、形容词**。存在各种词类**标记集**，介于 40 到 200 个标记之间。

•**词类标记**是为词的每个序列分配词类标签的过程。

•**命名实体**是指专有名词的单词，主要指人、地方和组织，但扩展到许多其他类型，而不仅仅是严格的实体甚至不是专有名词。

•**序列建模**的两种常见方法是**生成式**方法(**HMM** 标记)和**区分式**方法(**CRF** 标记)。在以下各章中，我们将看到一种神经方法。

•HMM 标记器中的概率估计就是在贴标的训练语料库上的最大似然估计。**Viterbi 算法**用于**解码**，以找到最可能的标签序列。

•**条件随机场**或 **CRF 标记器**训练对数线性模型，该模型可以根据对输出标记、先验输出标记、整个输入序列和当前时间步的条件，在给定观察序列的情况下，选择最佳标记序列。它们使用 Viterbi 算法进行推理，选择最佳标记序列，并使用前向后向算法(请参见附录 A)进行训练。

8.9. 文献和历史说明

最早的词类标记器可能是 Zellig Harris 的转换和话语分析项目(TDAP)中的一部分，该项目于 1958 年 6 月至 1959 年 7 月在宾夕法尼亚大学(Harris, 1962)实施，尽管早期的系统使用的是词类词典。TDAP 使用了 14 条手写的词类歧义规则;词类标签序列的使用和单词标签的相对频率预示着现代算法的发展。解析器本质上是一个有限状态转录机的级联;看看 Joshi 和 hope(1999)和 Karttunen(1999)的重新实施。

Klein and Simmons(1963)的计算语法编码器(CGC)具有三个成分：词典，形态分析器和上下文歧义消除器。1500 字的小词典仅列出了虚词和其他不规则词。形态分析器使用曲折后缀和派生后缀来分配词类类别。将这些词遍历单词以产生候选词类，然后通过依赖于周围不含歧义词的岛，通过一组 500 个上下文规则对歧义词进行歧义消除。例如，一条规则说，在 ARTICLE 和 VERB 之间，唯一允许的序列是 ADJ-NOUN，NOUN-ADVERB 或 NOUN-NOUN。TAGGIT 标记器(Greene 和 Rubin，1971)使用的结构与 Klein 和 Simmons(1963)相同，但词典更大，标记更多(87)。TAGGIT 用于布朗语料，根据 Francis 和 Kucera(1982，p.9)的说法，准确地标记了 77％的语料。剩下的布朗语料则被手工标记。所有这些早期算法均基于两阶段架构，首先用字典为每个单词分配一组潜在的词类，然后用手写的消歧规则列表将这些消歧规则筛选为每个单词的单个词类。

Stolz 等人(1965)在标记中使用了概率，Bahl 和 Mercer(1976)绘制了一个完整的带有维特比解码的概率标记。Lancaster-Oslo/Bergen(LOB)语料库，相当于英国的布朗语料库，在 20 世纪 80 年代早期被贴上了 CLAWS 标签(Marshall 1983;Marshall 1987;Garside 1987)，一个近似于简化 HMM 标记器的概率算法。该算法使用标记的 bigram 概率，但不是存储每个标记的单词的似然度，该算法将标记掩码(masked)为罕见(P(标记|单词)< 0.01)很少(P(标记|单词)< 0.10)或通常频繁(P(标记|单词)> 0.10)。

DeRose(1988)开发了一种准 HMM 算法，包括使用动态规划，虽然计算 P(t|w)P(w)而不是 P(w|t)P(w)。同年，Church(1988)的概率 PARTS 标记器(1989)可能是第一个实现的 HMM 标记器，在 Church(1989)中描述正确，尽管 Church(1988)也错误地将计算描述为 P(t|w)P(w)，而不是 P(w|t)P(w)。Church (p.c.)解释说，为了教学目的，他简化了这一概念，因为使用概率 P(t|w)让这个概念看起来更容易理解，就像“以几乎标准的形式存储词汇”一样。

后来，标记者明确地介绍了隐马尔可夫模型的使用(Kupiec 1992;Weischedel 等人 1993;Schutze 和 Singer 1994)。Merialdo(1994)表明，完全无监督的 EM 不能很好地完成标记任务，依赖手工标签的数据是重要的。Charniak 等人(1993)指出了最常见标记基线的重要性;我们上面给出的 92.3%的数字来自 Abney 等人(1999)。请参阅 Brants(2000)了解 HMM 标记器的实现细节，包括对三元语法上下文的扩展，以及使用复杂的未知单词特征;它的性能仍然接近最先进的标记器。

Ratnaparkhi(1996)引入了用于 POS 标记的对数线性模型，他引入了一个名为 MXPOST 的系统，该系统实现了最大熵马尔可夫模型(MEMM)，这是 CRF 的简单版本。大约在同一时间，序列标签器被应用于命名实体标记的任务，首先是 HMM(Bikel 等人，1997)和 MEMM(McCallum 等人，2000)，然后是 CRF(Lafferty 等人，2001)， 它们也被应用于 NER(McCallum 和 Li，2003)。随后对特征进行了广泛的探索(Zhou 等人，2005)。NER 的神经方法主要来自 Collobert 等人(2011)的开创性结果，他在卷积网络上应用了 CRF。具有基于单词和字符的嵌入作为输入的 BiLSTM 紧随其后，并成为 NER 的标准神经算法(Huang 等人 2015; Ma 和 Hovy 2016; Lample 等人 2016)，随后是 Transformers 和 BERT 的最新使用。

用字母后缀代替未知单词的想法已经很古老了;早期的 Klein 和 Simmons(1963)系统检查了所有长度为 1-5 的最终字母后缀。第 8.5.1 节描述的未知词特征主要来自 Ratnaparkhi (1996)，Toutanova 等人(2003)和 Manning(2011)进行了补充。

最先进的 POS 标记器使用神经算法，双向 RNN 或像 BERT 这样的 Transformer;见第九章和第十章。HMM (Brants 2000;Thede 和 Harper 1999)和 CRF 标记器的准确性可能会稍微低一些。

Manning(2011)调查了一个高效标记器中剩下的 2.7%的错误(Toutanova 等人，2003)。他认为，这些剩余的错误中有三分之一或一半是由于训练数据中的错误或不一致造成的，三分之一可能可以用更丰富的

语言模型来解决，而其余的任务则不够明确或不清楚。

监督式标记在很大程度上依赖于由专家手工标签的领域内训练数据。放松这一假设的方法包括 Christodoulopoulos 等人(2010)总结的将单词聚类为**像词类的类(part-of-speech-like classes)**的无监督算法，以及结合有标签和无标签数据的方法，例如通过联合训练(Clark 等人 2003，Søgaard 2010)。

参见 Householder(1995)关于词类的历史注释，以及 Sampson(1987)和 Garside 等人(1997)关于 Brown 和其他标记集的起源。

8.10. 练习

8.1 Find one tagging error in each of the following sentences that are tagged with the Penn Treebank tagset:

1. I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN
2. Does/VBZ this/DT flight/NN serve/VB dinner/NNS
3. I/PRP have/VB a/DT friend/NN living/VBG in/IN Denver/NNP
4. Can/VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS

8.2 Use the Penn Treebank tagset to tag each word in the following sentences from Damon Runyon's short stories. You may ignore punctuation. Some of these are quite difficult; do your best.

1. It is a nice night.
2. This crap game is over a garage in Fifty-second Street. . .
3. . . . Nobody ever takes the newspapers she sells . . .
4. He is a tall, skinny guy with a long, sad, mean-looking kisser, and a mournful voice.
5. . . . I am sitting in Mindy's restaurant putting on the gefillte fish, which is a dish I am very fond of, . . .
6. When a guy and a doll get to taking peeks back and forth at each other, why there you are indeed.

8.3 Now compare your tags from the previous exercise with one or two friend's answers. On which words did you disagree the most? Why?

8.4 Implement the "most likely tag" baseline. Find a POS-tagged training set, and use it to compute for each word the tag that maximizes $p(t|w)$. You will need to implement a simple tokenizer to deal with sentence boundaries. Start by assuming that all unknown words are NN and compute your error rate on known and unknown words. Now write at least five rules to do a better job of tagging unknown words, and show the difference in error rates.

8.5 Build a bigram HMM tagger. You will need a part-of-speech-tagged corpus. First split the corpus into a training set and test set. From the labeled training set, train the transition and observation probabilities of the HMM tagger directly on the hand-tagged data. Then implement the Viterbi algorithm so you can decode a test sentence. Now run your algorithm on the test set. Report its error rate and compare its performance to the most frequent tag baseline.

8.6 Do an error analysis of your tagger. Build a confusion matrix and investigate the most frequent errors. Propose some features for improving the performance of your tagger on these errors.

8.7 Develop a set of regular expressions to recognize the character shape features described on page 164.

8.8 The BIO and other labeling schemes given in this chapter aren't the only possible one. For example, the B tag can be reserved only for those situations where an ambiguity exists between adjacent entities. Propose a new set of BIO tags for use with your NER system. Experiment with it and compare its performance with the schemes presented in this chapter.

8.9 Names of works of art (books, movies, video games, etc.) are quite different from the kinds of named entities we've discussed in this chapter. Collect a list of names of works of art from a particular category from a Web-based source (e.g., gutenberg.org, amazon.com, imdb.com, etc.). Analyze your list and give examples of ways that the names in it are likely to be problematic for the techniques described in this chapter.

8.10 Develop an NER system specific to the category of names that you collected in the last exercise. Evaluate your system on a collection of text likely to contain instances of these named entities.

9. 深度学习架构用于序列处理

Time will explain.

Jane Austen, Persuasion

语言是一种固有的时间现象。当我们理解和产生口语时，我们处理不确定长度的连续输入流。即使在处理书面文本时，我们通常也按顺序处理。语言的时代性反映在我们使用的隐喻中;我们谈论的是对话流、新闻源和推特流，所有这些都唤起了这样一种观念：语言是一个随时间展开的序列。

这种时间性质反映在我们用于处理语言的算法中。例如，当应用于词类标记问题时，Viterbi 算法通过一次输入一个单词来逐步实现自己的方式，从而沿途收集信息。另一方面，我们为情感分析而研究的机器学习方法和其他文本分类任务不具有这种时间性，它们假定可以同时访问其输入的所有方面。前馈神经网络尤其如此，包括它们在神经语言模型中的应用。这些完全连接的网络使用固定大小的输入以及关联的权重来一次捕获示例的所有相关方面。这使得难以处理长度可变的序列，并且无法捕获语言的重要时间方面的信息。

这些问题的解决方法是神经语言模型采用的滑动窗口方法。这些模型通过接受固定大小的符记窗口作为输入来运行。长度大于窗口大小的序列通过遍历输入进行预测来处理，最终结果是跨越输入的一系列预测。重要的是，在一个窗口中做出的决定不会影响后续的决定。从第 7 章图 7-13 复制而来的图 9.1，描述了使用这种方法的神经语言模型的操作，该方法的窗口大小为 3。在这里，根据输入“for all the”，我们将预测下一个单词。通过将窗口一次向前滑动一个单词来预测后续单词。

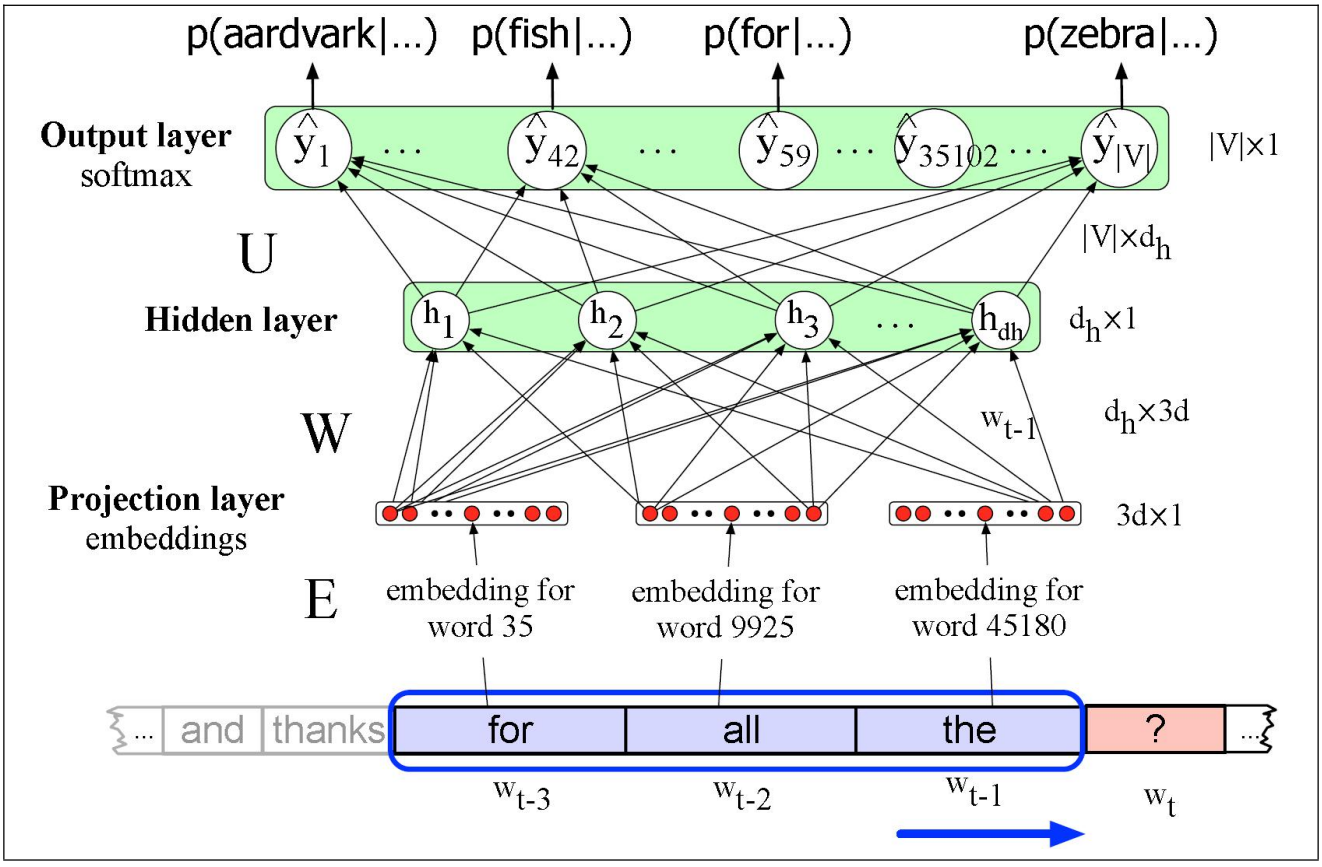


图 9-1：前馈神经语言模型的简化视图

图注：在文本中移动的前馈神经语言模型的简化视图。在每个时间步 t，网络获取 3 个上下文词，将每个上下文词转换为 d 维嵌入，并将 3 个嵌入拼接在一起，以获得网络的 $1\times N_d$ 单元输入层 x。网络的输出是词汇表上的概率分布，代表相对于每个单词为下一个可能单词的模型置信度。

由于多种原因，这种通用方法存在问题。首先，它与我们早期的马尔可夫 N-gram 方法一样，具有主要的缺点，即它限制了可以从中提取信息的上下文。上下文窗口之外的任何内容都不会影响所做出的决定。这是一个问题，因为有许多任务需要访问可能与处理发生地点(point)任意距离的信息。其次，窗口的使用使网络很难学习像成分等现象引起的系统模式。例如，在图 9.1 中，短语“all the”出现在两个单独的窗口中：首先显示为窗口中的第二和第三位置，然后再次出现在下一步中显示为第一和第二位置，从而迫使网络学习两种不同模式（这两种模式原本是一样的）。

本章涵盖了旨在解决这些挑战的两个紧密相关的深度学习架构：递归神经网络(RNN)和 Transformer 网络。两种方法都具有直接处理语言的顺序性质的机制，这些机制允许它们在不使用任何固定大小的窗口的情况下处理可变长度的输入，并捕获和利用语言的时间性质。

9.1. 语言模型回顾

在本章中，我们将主要通过概率语言模型的镜头(lens)来探讨这两种架构。回顾第三章，概率语言模型在给定某些先前上下文的情况下预测序列中的下一个单词。例如，如果前面的上下文是“Thanks for all the”，并且我们想知道下一个单词是“fish”的可能性，我们将计算：

P(fish|Thanks for all the)

语言模型让我们能够为每个可能的下一个单词分配这样的条件概率，给我们整个词汇的分布。我们也可以将这些条件概率与链式规则结合起来分配给整个序列的概率:

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i|w_{<i})$$

这个公式产生了广泛的序列标记应用，正如我们将看到的，它提供了一个明确的训练目标，该目标基于模型对序列中下一个单词的预测有多好。

我们已经看到了两种实例化概率语言模型的方法:第 3 章的 N-gram 模型和第 7 章的带滑动窗口的前馈神经网络。不幸的是，这两种方法都受到如下公式中马尔可夫假设的约束：

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{(n-N+1):(n-1)})$$

也就是说，预测基于一个大小为 N 的固定的先前上下文;在此之前发生的任何输入都与结果无关。我们在本章中探索的方法将会放宽这个假设，允许模型使用更大的上下文。

我们通过检查语言模型对与训练数据来自同一来源的看不见的数据进行预测的能力来评估语言模型。直观地讲，好的模型是为看不见的数据分配更高概率的模型。为了使这种直觉具体化，我们使用困惑度作为模型质量的度量。相对于看不见的测试集，模型θ的困惑度(PP)是模型分配给它的概率，并通过其长度进行归一化。

$$PP_\theta(w_{1:n}) = P(w_{1:n})^{\frac{1}{n}}$$

受信息论启发，观察困惑度的另一种方法是熵。

$$\begin{aligned} PP(w_{1:n}) &= 2^{H(w_{1:n})} \\ &= 2^{-\frac{1}{n}\sum_1^n log_2 m(w_n)} \end{aligned}$$

在这个公式中，指数中的值是我们当前模型相对于真实分布的交叉熵。

评估语言模型的另一种方法是使用它来生成新的序列。生成的序列反映训练数据的程度表明了模型的质量。我们在第 3 章中了解了如何通过采用同时由 Claude Shannon(Shannon，1951)和心理学家 George Miller 和 Selfridge(1950)提出的技术来做到这一点。首先，我们根据一个单词是否适合作为一个序列的开头对一个单词进行随机采样以开始一个序列。在对第一个单词进行采样之后，我们对以先前选择为条件的其他单词进行采样，直到达到预定长度，或者生成序列符记的结尾。今天，这种方法称为**自动回归生成(autoregressive generation)**。我们将介绍其在诸如机器翻译和文本摘要之类的问题中的实际应用。

9.2. 循环神经网络

循环神经网络(RNN)是在其网络连接中包含循环的任何网络，即任何一个单元的值直接或间接取决于其自身较早的输出作为输入的网络。尽管功能强大，但难以推理和训练。但是，在通用的循环网络类别中，有一些受约束的体系结构已被证明在应用于口语和书面语言时非常有效。在本节中，我们考虑一类称为**Elman Networks**(Elman，1990)的循环网络或**简单循环网络(simple recurrent networks)**。这些网络本身是有用的，并且是更复杂方法的基础，例如本章稍后讨论的**长短期记忆(LSTM)**网络。展望未来，当我们使用术语 RNN 时，我们将指的是这些更简单、更受约束的网络。

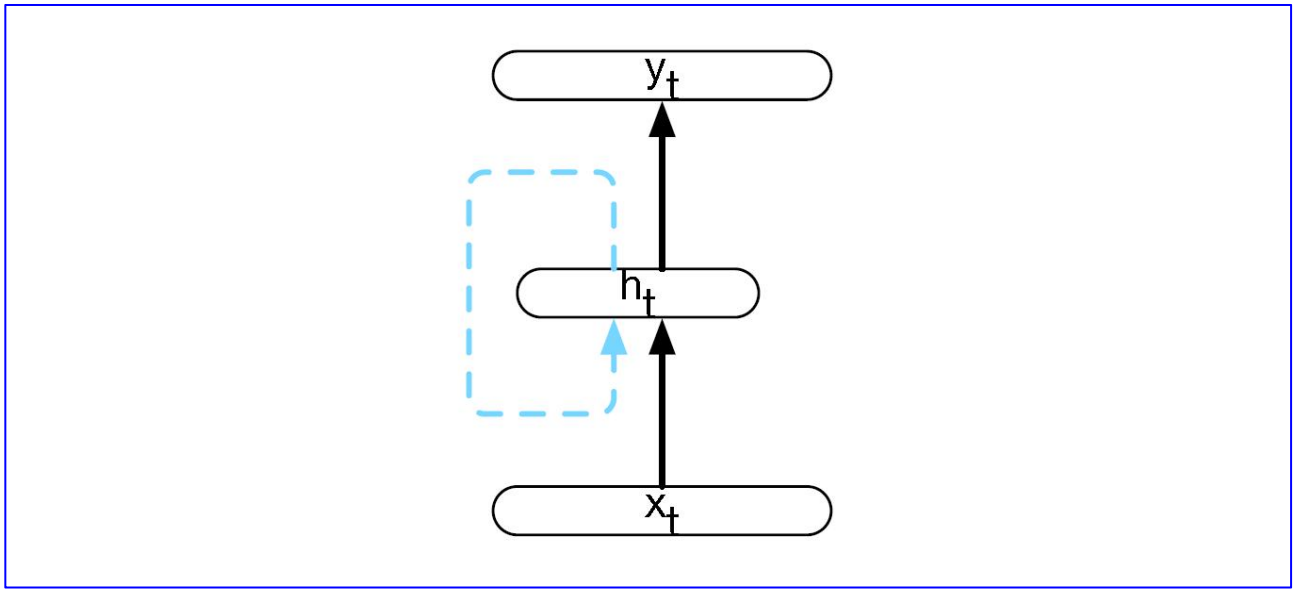


图 9-2：Elman 的简单循环神经网络

图注：Elman 的简单循环神经网络(Elman, 1990)。隐藏层包括一个循环连接，作为其输入的一部分。即隐含层的激活值既取决于当前输入，也取决于上一个时间步的隐含层激活值。

图 9.2 说明了 RNN 的结构。与普通前馈网络一样，将代表当前输入 x_t 的输入向量乘以权重矩阵，然后传递给非线性激活函数以计算隐藏单元层的值。然后将此隐藏层用于计算相应的输出 y_t。与我们以前的基于窗口的方法不同，通过一次向网络展示一项来处理序列。与前馈网络的主要区别在于图中虚线所示的循环连接。此连接将输入增加到隐藏层的计算中，该隐藏层的值“来自前一个时间点”。

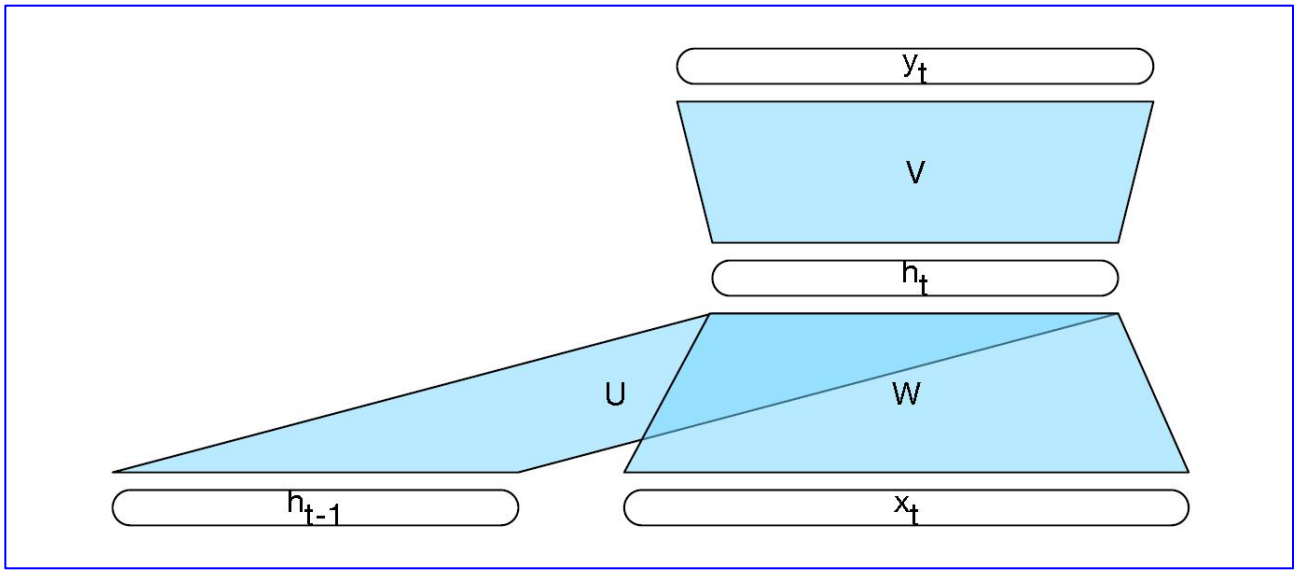


图 9-3：前馈网络的简单循环神经网络

上一个时间步中的隐藏层提供了一种记忆或上下文形式，该形式对较早的处理进行编码并通知要在较晚的时间点做出的决策。至关重要的是，这种方法不会在此先验上下文中施加固定长度的限制。前一个隐藏层中嵌入的上下文包括延伸回到序列开头的信息。

添加这个时间维度会使 RNN 看起来比非循环体系结构更复杂。但实际上，它们并没有那么不同。给定输入向量和前一个时间步的隐含层的值，我们仍然执行第七章中介绍的标准前馈计算。为了看到这一点，考虑图 9.3，它阐明了循环的性质，以及它如何在隐藏层的计算中发挥影响(factors)。最重要的变化是新的一组权值 U，它连接了从上一个时间步的隐藏层到当前的隐藏层。这些权重决定了网络如何利用过去的上下文来计算当前输入的输出。与网络中的其他权值一样，这些连接是通过反向传播进行训练的。

9.2.1. RNN 中的推理

RNN 中的前向推理(将输入序列映射到输出序列)与我们在前馈网络中看到的几乎相同。为了计算输入 x_t 对应的输出 y_t，我们需要隐藏层 h_t 的激活值。为了计算这一点，我们将输入 x_t 乘以权重矩阵 W，并将上一个时间步 h_{t-1} 的隐藏层乘以权重矩阵 U。我们将这些值加在一起，然后将它们传递给合适的激活函数 g，以得出当前隐藏层的激活值 h_t。一旦有了隐藏层的值，就可以进行通常的计算以生成输出向量。

$$h_t=g(Uh_{t-1}+Wx_t)$$
$$y_t=f(Vh_t)$$

这里值得仔细指定输入层、隐藏层和输出层的维数，以及权值矩阵，以确保这些计算是正确的。

输入层、隐藏层和输出层的尺寸分别为 d_{in}、d_h 和 d_{out}。因此，我们的三个参数矩阵是:

$$W \in \mathbb{R}^{d_h \times d_{in}}, \quad U \in \mathbb{R}^{d_h \times d_h}, \quad V \in \mathbb{R}^{d_{out} \times d_h}$$

在软分类的常见情况下，计算 y_t 由 softmax 计算组成，该计算提供了可能的输出类别上的概率分布。

$$y_t=\text{softmax}(Vh_t)$$

在 t 时刻的计算要求从 t-1 时刻开始的隐含层的值，这一事实要求一种从序列开始到结束的增量推理算法，如图 9.4 所示。通过及时展开网络，也可以看出简单循环网络的顺序性质，如图 9.5 所示。在这个图中，每个时间步都复制了不同的单元层，以说明它们随着时间的推移会有不同的值。然而，不同的权重矩阵在时间上是共享的。

```
function FORWARDRNN(x, network) returns output sequence y
h_0 ←0
for i←1 to LENGTH(x) do
    h_i←g(U h_{i-1} + W x_i)
    y_i← f(V h_i)
return y
```

图 9-4：简单循环网络中的正向推理

图注：矩阵 U，V 和 W 在时间上是共享的，而 h 和 y 的新值在每个时间步中计算。

9.2.2. 训练

与前馈网络一样，我们将使用训练集，损失函数和反向传播来获得调整这些循环网络中权重所需的梯度。如图 9.3 所示，我们现在有 3 组权重要更新：W，从输入层到隐藏层的权重；U，从前一个隐藏层到当前隐藏层的权重；V，从隐藏层到输出层的权重。

图 9.5 突出了我们在前馈网络中不需要担心的两个问题。首先，为了计算 t 时刻输出的损失函数，我们需要从 t-1 时刻开始的隐含层。第二，t 时刻的隐含层影响 t 时刻的输出和 t+1 时刻的隐含层(因此也影响 t +1 时刻的输出和损失)。由此可以得出，要评估 h_t 产生的误差，我们需要知道它对当前输出以及随后输出的影响。

针对这种情况量身定制反向传播算法会导致在 RNN 中训练权重的两遍算法。第一遍，我们执行前向推断，计算 h_t 和 y_t，在每个时间步累加损失，在每个步骤保存隐藏层的值以供下一时间步使用。第二遍，我们以相反的顺序处理序列，随着时间的推移计算所需的梯度，计算并保存误差项，以供在时间上向后的

每一步用于隐藏层。这种通用方法通常称为**时间反向传播(Backpropagation Through Time)**(Werbos 1974，Rumelhart 等人 1986，Werbos 1990)。

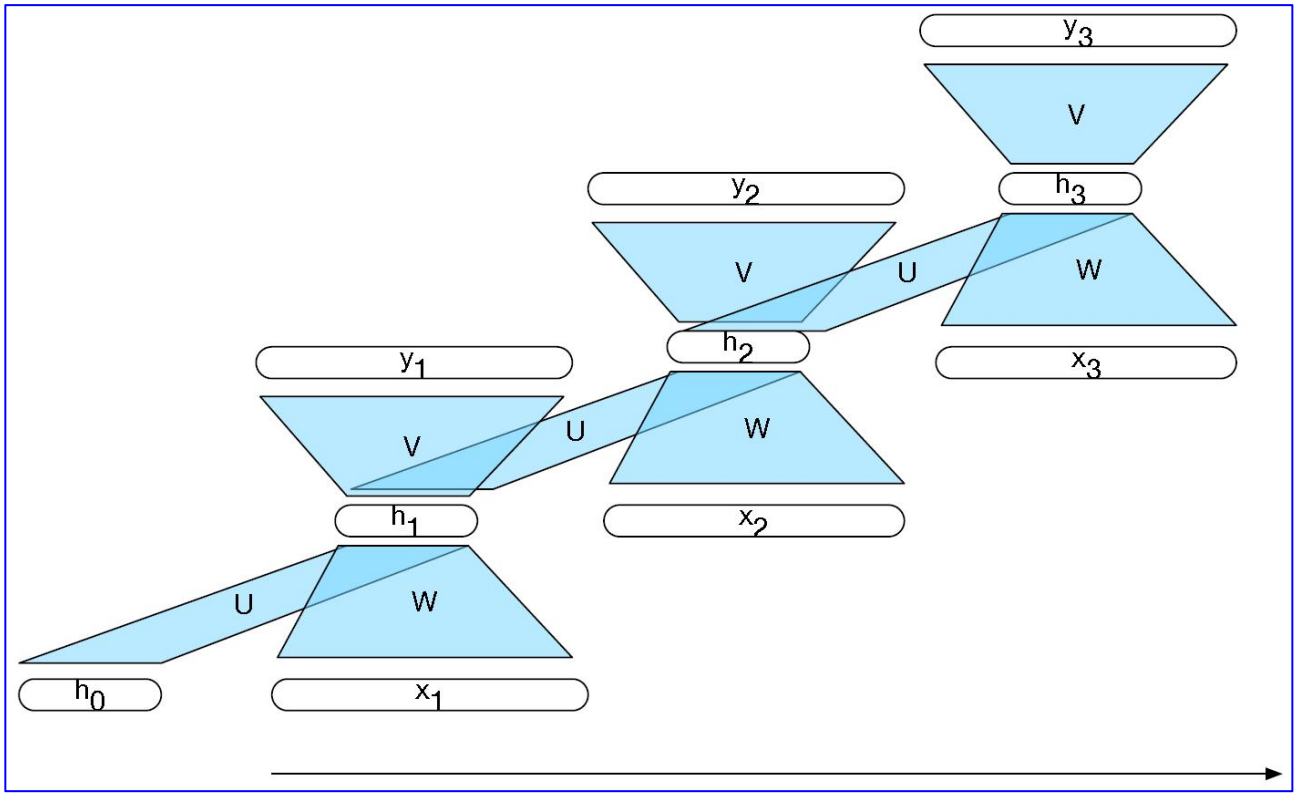


图 9-5：在时间上展开的简单循环神经网络

图注：每个时间步都复制网络层，而所有时间步都共享权值 U、V 和 W。

幸运的是，有了现代的计算框架和足够的计算资源，就不需要专门的方法来训练 RNN。如图 9.5 所示，将循环网络显式展开为前馈计算图可消除任何显式循环，从而可以直接训练网络权重。在这种方法中，我们提供了一个模板，用于指定网络的基本结构，包括输入、输出和隐藏层的所有必要参数、权重矩阵以及要使用的激活和输出函数。然后，当呈现特定的输入序列时，我们可以生成特定于该输入的展开前馈网络，并使用该图通过普通的反向传播执行前向推断或训练。

对于涉及更长输入序列的应用程序，例如语音识别，字符级处理或连续输入流，展开整个输入序列可能不可行。在这些情况下，我们可以将输入展开为可管理的固定长度段，并将每个段视为不同的训练项目。

9.2.3. RNN 作为语言模型

基于 RNN 的语言模型一次处理一个单词的序列，试图通过使用当前单词和之前的隐藏状态作为输入来预测序列中的下一个单词(Mikolov 等人，2010)。N-gram 模型中固有的有限上下文约束被避免了，因为隐藏状态包含了关于前面所有单词的信息，一直到退回到序列的开始。

循环语言模型中的前向推理完全按照第 9.2.1 节中的描述进行。输入序列 x 由单词嵌入组成，这些单词嵌入表示为大小为|V|×1 的独热向量。并且输出预测 y 表示为向量，表示在词汇表上的概率分布。在每个步骤中，模型都使用单词嵌入矩阵 E 检索当前单词的嵌入，然后将其与上一步中的隐藏层进行组合以计算新的隐藏层。然后，该隐藏层用于生成输出层，该输出层将穿过 softmax 层以在整个词汇表上生成概率分布。也就是说，在时间 t：

$$e_t = E^T x_t$$
$$h_t = g(Uh_{t-1}+We_t)$$
$$y_t = \text{softmax}(Vh_t)$$

给定 h 中提供的证据，可以将 Vh 产生的向量视为词汇量的一组分数。将这些分数传递给 softmax 会将分数归一化为概率分布。给定 y，则词汇中特定单词 i 的概率(即下一个单词)就是 y 的相应分量。

$$P(w_{t+1} = i \mid w_{1:t}) = y_t^i$$

由此可以得出，整个序列的概率就是序列中每一项的概率的乘积。

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i \mid w_{1:i-1})$$
$$= \prod_{i=1}^{n} y_{w_i}^i$$

为了将 RNN 作为一种语言模型进行训练，我们使用一个文本语料库作为训练材料，并结合名为**教师强制(teacher forcing)**的训练方案。使用交叉熵作为损失函数，其任务是最小化预测训练序列中下一个单词的误差。回想一下，交叉熵损失度量的是预测概率分布与正确分布之间的差异。

$$L_{CE} = -\sum_{w \in V} y_w^t \log \hat{y}_w^t$$

在语言建模的情况下，正确的分布 y 来自于知道下一个单词。这表示为对应词汇表的一个独热向量，其中实际下一个单词的条目为 1，所有其他条目为 0。因此，语言建模的交叉熵损失是由模型分配给正确的下一个单词的概率决定的。具体来说，在时刻 t, CE 损失是分配给训练序列中下一个单词的负对数概率。

$$L_{CE}(\hat{y}_t, y_t) = -\log \hat{y}_{w_{t+1}}^t \quad (9.1)$$

在实践中，通过梯度下降调整网络中的权值，使训练序列的平均 CE 损失最小。图 9.6 说明了这个训练方案。

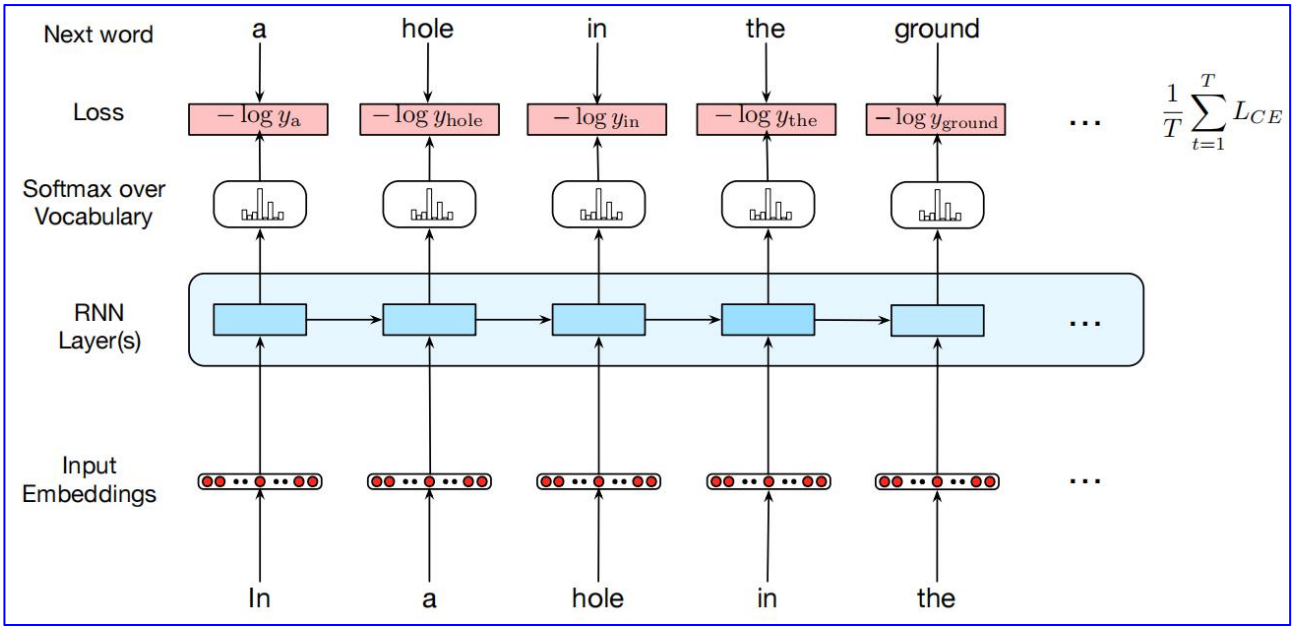


图 9-6：训练 RNN 作为语言模型

细心的读者可能已经注意到，输入的嵌入矩阵 E 和最后一层矩阵 V(V 的输出馈入 softmax 函数)非常相似。E 的行表示训练过程中学习到的词汇中每个单词的嵌入度，目的是让含义和功能相似的单词具有相似的嵌入度。由于这些嵌入的长度对应于隐藏层 d_h 的大小，因此嵌入矩阵形状 E 为$|V| \times d_h$。

最终层矩阵 V 提供了一种对词汇表中每个单词的似然度进行评分的方法，该方法通过 V_h 的计算，给出存在于网络最终隐藏层中的证据。这要求它也具有维度$|V| \times d_h$。也就是说，V 的行提供了第二组学习的单词嵌入，这些单词嵌入捕获了单词含义和功能的相关方面。这就引出了一个明显的问题：两者是否都需要？**权重绑定(Weight Tying)**是一种方法，它免除了此冗余，并在输入层和 softmax 层使用一组嵌入。也就是说，E=V。为了做到这一点，我们将最终隐藏层的维数设置为相同的 d_h，(或添加一个额外的投影层以执行相同的操作)，并简单地对两个层使用相同的矩阵。除了提供改善的困惑度结果之外，这种方法还大大减少了模型所需的参数数量。

基于 RNN 的语言模型生成

就像第 3 章中的概率莎士比亚生成器一样，了解语言模型的一个有用方法是使用训练过的模型来生成随机的新句子。这个过程和第 3.3 节描述的基本一样。

•首先，从 softmax 分布的输出中选取一个单词作为第一个输入，这是使用句子标记开头<s>作为第一个输入的结果。

•使用第一个单词的词嵌入作为下一个时间步的网络输入，然后以同样的方式采样下一个单词。

•继续生成，直到采样到句子标记</s>或达到固定长度限制为止。

这种技术被称为自回归生成，因为每一步生成的词都是以网络在前一步中选择的词为条件的。图 9.7 说明了这种方法。在这个图中，RNN 的隐藏层和循环连接的细节隐藏在蓝色块中。

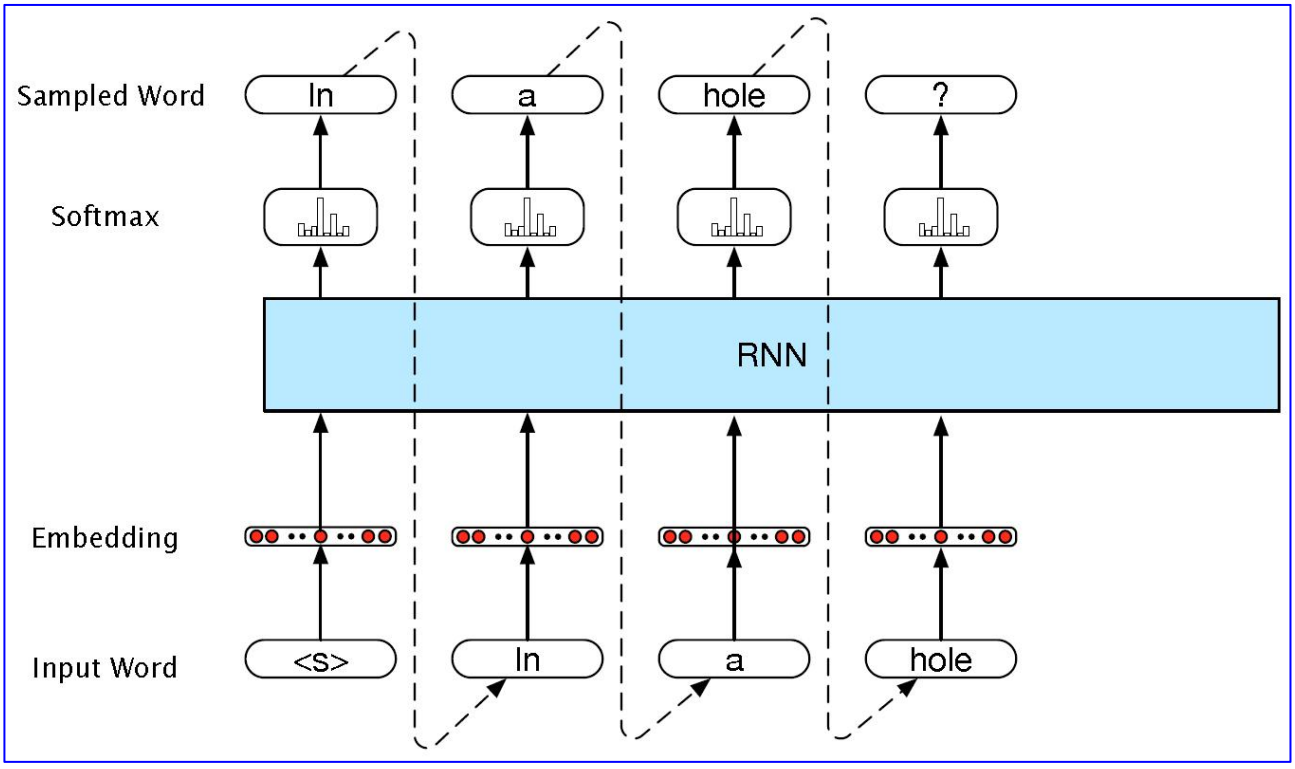


图 9-7：基于 RNN 的神经语言模型自回归生成

虽然这是一个有趣的练习，但该体系结构启发了最新的应用方法，例如机器翻译，摘要和问答。这些方法的关键是在适当的上下文中填充生成组件。也就是说，除了简单地使用<s>来开始工作之外，我们还可以提供更丰富的适合任务的上下文。在第 9.4 节中，我们将在基于 Transformer 的语言模型的背景下讨论上下文生成在摘要问题上的应用。

9.2.4. RNN 的其它应用

事实证明，循环神经网络是一种有效的方法，可用于语言建模，序列标签任务(例如词类标记)以及序列分类任务(例如情感分析和主题分类)。正如我们将在第 11 章中看到的那样，它们构成了摘要，机器翻译和问答的逐个序列方法的基础。

序列标签

在序列标签中，网络的任务是将一个从固定的小型标签集中选择的标签分配给序列的每个元素。序列标签的规范示例包括词类标记和命名实体识别，已在第 8 章中详细讨论了。在 RNN 序列标签方法中，输入是词嵌入，而输出是由给定标记集上的 softmax 层生成的标记概率 ，如图 9.8 所示。

在该图中，每个时间步的输入都是与输入符记相对应的预训练词嵌入。RNN 块是一个抽象概念，代表展开的简单循环网络，该网络由每个时间步的输入层，隐藏层和输出层以及组成该网络的共享 U，V 和 W 权重矩阵组成。网络在每个时间步的输出表示为一个分布，该分布置于 softmax 层生成的 POS 标签集上。

为了为给定的输入生成标签序列，我们对输入序列进行前向推理，并在每个步骤中从 softmax 中选择最可能的标记。由于我们在每个时间步都使用 softmax 层在输出标签集上生成概率分布，因此我们将在训

练过程中再次采用交叉熵损失。

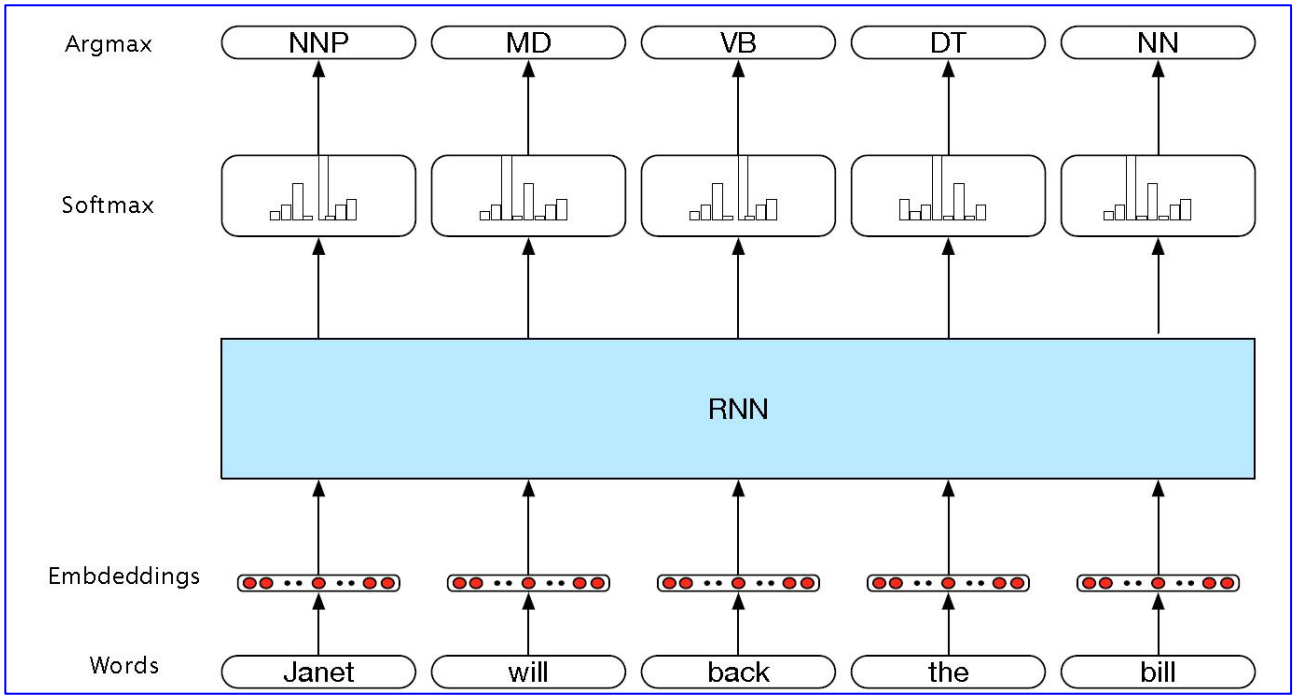


图 9-8：使用简单的 RNN 将词类标记当作序列标签

图注：预先训练的单词嵌入用作输入，并且 softmax 层在每个时间步上输出的词类标记上都提供了概率分布。

9.2.5. RNN 用于序列分类

RNN 的另一个用途是对整个序列进行分类，而不是对其中的符记进行分类。在第 4 章中，我们已经对情感分析进行了讨论。其他示例包括文档级主题分类，垃圾邮件检测，针对客户服务应用程序的邮件路由以及欺骗检测。在所有这些应用中，文本序列被分类为属于少数类别之一。

要在此设置中应用 RNN，将要分类的文本一次通过一个单词传递到 RNN，并在每个时间步生成一个新的隐藏层。文本的最后元素 h_n 的隐藏层被用来构成整个序列的压缩表示。在最简单的分类方法中，h_n 用作后续前馈网络的输入，该网络通过 softmax 在可能的类别上选择类别。图 9.9 说明了这种方法。

注意，在这种方法中，最后一个元素之前的序列中的单词没有中间输出。因此，没有与这些元素相关的损失项。取而代之的是，用于训练网络中权重的损失函数完全基于最终的文本分类任务。具体来说，softmax 的输出和前馈分类器的输出，连同交叉熵损失一起，驱动了训练的进行。

分类的误差信号通过前馈分类器中的权值一直反向传播到其输入，然后再反向传播到前面第 9.2.2 节中描述的 RNN 中的三组权值。这种简单循环网络与前馈分类器的组合是我们深度神经网络的第一个例子。使用来自下游应用程序的损失在整个网络中调整权值的训练方案称为**端到端训练(end-to-end training)**。

9.2.6. 堆叠和双向 RNN

从图 9.9 所示的序列分类体系结构可以看出，循环网络是非常灵活的。通过将展开的计算图的前馈性质与作为公共输入和输出的向量相结合，可以将复杂网络视为可以以创造性方式组合的模块。本节介绍在 RNNs 语言处理中使用的两种更常见的网络架构。

堆叠 RNNs

到目前为止，在我们的示例中，RNN 的输入由单词或字符嵌入(矢量)的序列组成，而输出则是可用于预测单词、标记或序列标签的矢量。但是，没有什么可以阻止我们将一个 RNN 的整个输出序列用作到另一个 RNN 的输入序列。**堆叠(Stacked)RNN** 由多个网络组成，其中一层的输出充当下一层的输入，如图 9.10 所示。

它已经在许多任务中得到了证明，堆叠 RNN 可以比单层网络的性能更好。这一成功的原因之一与网络在不同的抽象层上诱导(induce)表示的能力有关。就像早期的人类视觉系统检测边缘,然后用于发现更大的区域和形状,多层网络的初始层可以诱导表示,作为进一步有用的抽象层,表示可能在一个 RNN 难以诱导。

堆栈的最优数量是针对每个应用和每个训练集而定的。但随着堆栈数量的增加，训练成本迅速上升。

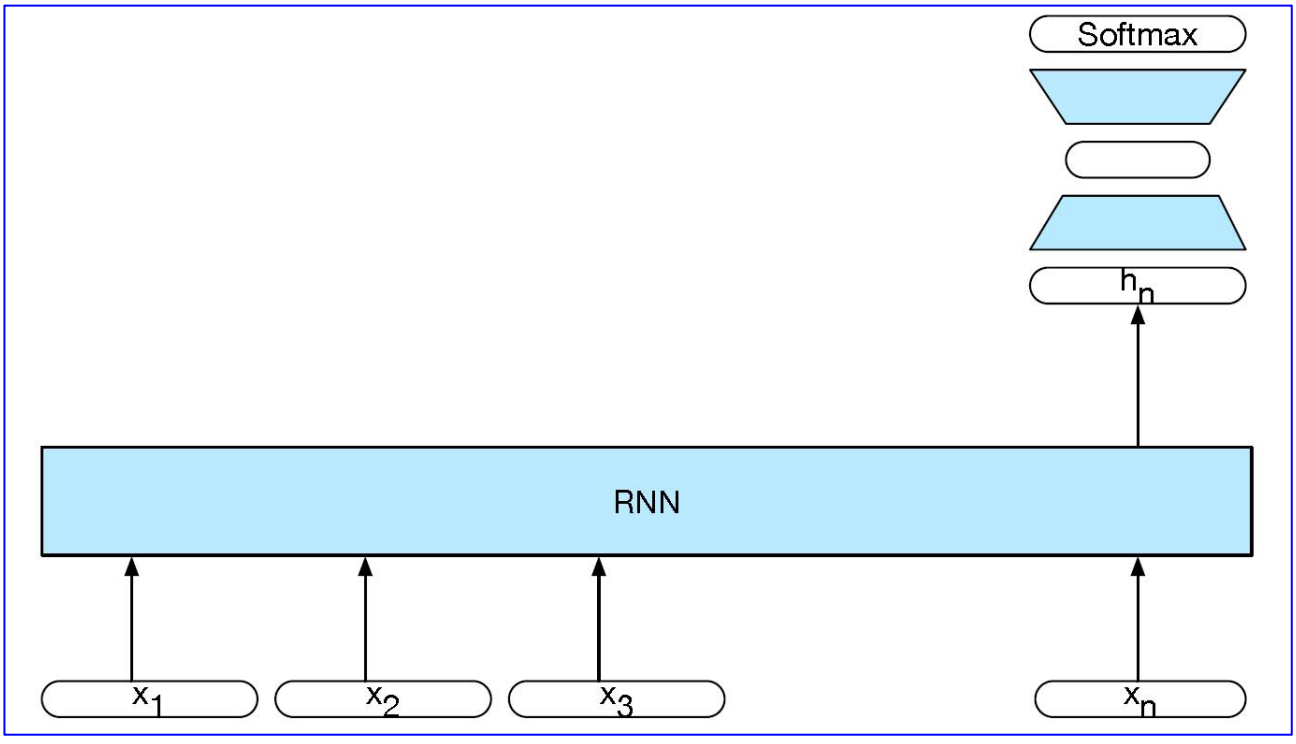


图 9-9：使用简单的 RNN 和前馈网络进行序列分类

图注：来自 RNN 的最终隐藏状态用作执行分类的前馈网络的输入。

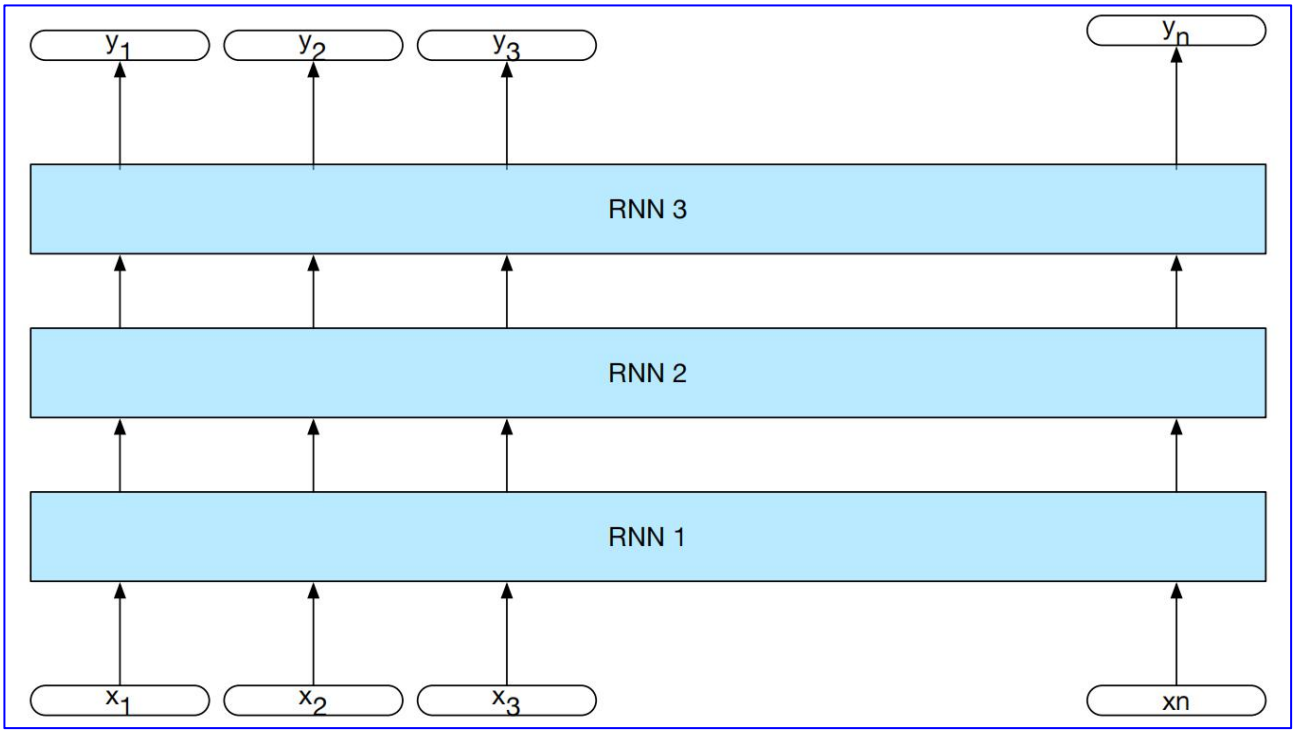


图 9-10：堆叠式 RNN

图注：较低一级的输出作为较高一级的输入，最后一个网络的输出作为最终输出。

双向 RNNs

在简单的循环网络中，给定时间 t 的隐藏状态展现网络在序列中直到该点为止所知道的一切。即，在时间 t 的隐藏状态是从开始到时间 t 的输入的函数的结果。我们可以将其视为当前时间左侧的网络上下文。

$$h_t^f = \text{RNN}_{\text{forward}}(x_1^t)$$

其中h_t^f对应于 t 时刻的正常隐藏状态，表示网络从序列到该点收集到的所有信息。

在许多应用程序中，我们可以一次访问整个输入序列。我们可能会问，利用当前输入右边的上下文是否有帮助。恢复此类信息的一种方法是，使用与我们一直在讨论的完全相同的网络，在输入序列上反向训练 RNN。通过这种方法，时间 t 的隐藏状态现在表示有关当前输入右侧的序列的信息。

$$h_t^b = \text{RNN}_{\text{backward}}(x_t^n)$$

这里，隐藏状态h_t^b表示我们已经识别的从 t 到序列末尾的所有信息。

将正向网络和反向网络相结合产生了**双向 RNN**(Schuster 和 Paliwal, 1997)。一个双向 RNN 由两个独立的 RNN 组成，一个从开始到结束处理输入，另一个从结束到开始处理输入。然后，我们将两个网络的输出组合成一个单独的表示形式，在每个时间点捕捉输入的左上下文和右上下文。

$$h_t = h_t^f \oplus h_t^b$$

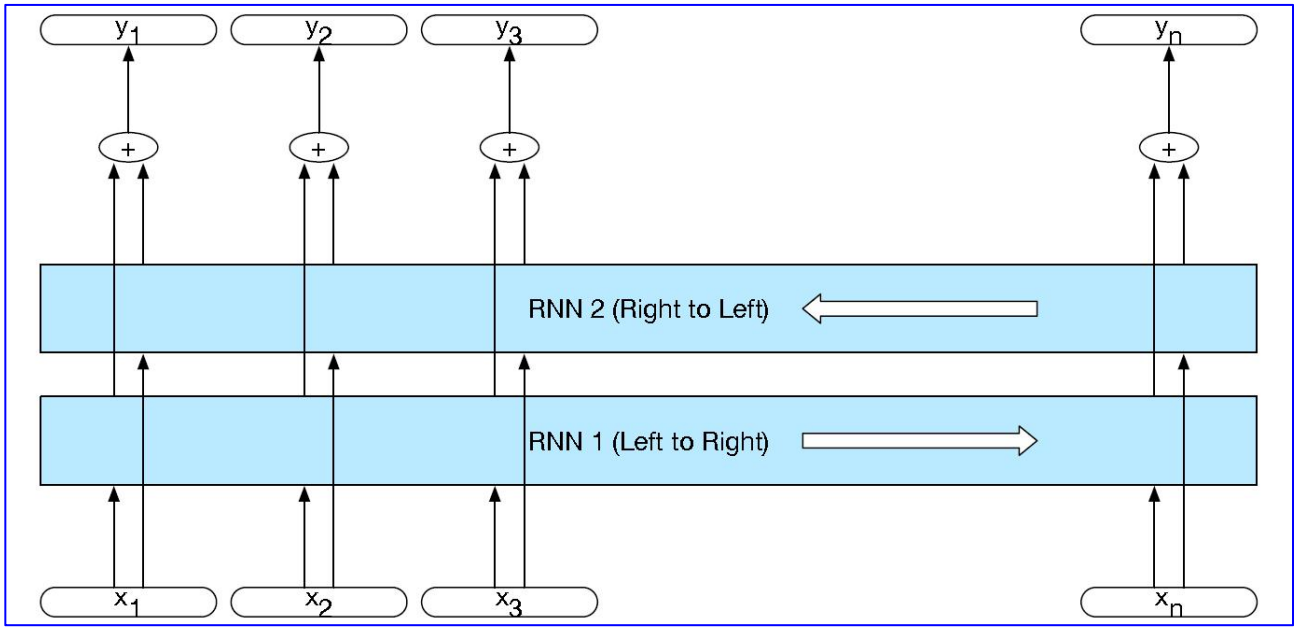


图 9-11：一个双向 RNN

图注：分别的模型在前进和后退方向进行训练，每个模型在每个时间点的输出拼接起来，以表示该时间点的事件状态。包裹在前向和后向网络周围的方框强调了这种架构的模块化本质。

图 9.11 说明了一个双向网络，其中前向和后向通过的输出是拼接在一起的。组合前向和后向上下文的其他简单方法包括逐元素加法或乘法。因此，每个时间步的输出都会在当前输入的左侧和右侧捕获信息。在序列标签应用中，这些拼接的输出可以用作本地标签决策的基础。

双向 RNN 也被证明对序列分类非常有效。回顾图 9.10，对于序列分类，我们使用 RNN 的最终隐藏状态作为后续前馈分类器的输入。这种方法的困难在于，最终状态自然会反映出比句子开头更多的有关句子结尾的信息。双向 RNN 为这个问题提供了一个简单的解决方案。如图 9.12 所示，我们仅将前进和后退路径中的最终隐藏状态组合在一起，并将其用作后续处理的输入。同样，拼接是组合两个输出的常用方法，但也使用逐元素求和、相乘或求平均。

9.3.在 RNN 中管理上下文：LSTM 和 GRU

在实践中，有一个非常困难的训练 RNN 的任务：要求网络利用远离当前处理点的信息。尽管可以访问整个前面的序列，但是，被编码在隐藏状态里的信息，往往是相当局部的，而且与输入序列的最新部分和最新决策更相关。然而，通常情况下，遥远的信息对于许多语言应用程序都是至关重要的。为此，请在

语言建模的上下文中考虑以下示例。

(9.2) The flights the airline was cancelling were full.

分配一个高概率给“airline”后面的“was”是很直截了当的，因为“airline”为语法上的单数一致性提供了强大的局部上下文。但是，很难给“were”分配适当的概率，这不仅是因为复数“flights”距离很远，而且还因为介入的上下文涉及单数成分。理想情况下，网络应该能够保留有关复数“flights”的遥远信息，直到需要它为止，同时仍能正确处理序列的中间部分。

RNN 无法传递关键信息的原因之一是隐藏层和权重(权重决定了隐藏层中的值)，它们同时被要求执行两个任务：提供对当前决策有用的信息，以及更新和转发未来决策所需的信息。

训练 SRN 的第二个困难来自需要反向传播误差信号。从 9.2.2 节回想起，时间 t 的隐藏层会参与下一计算步，因此会导致下一步骤的损失。结果，在训练的后退过程中，隐藏层要经受重复乘法，这由序列的长度确定。该过程的一个常见结果是，梯度最终被驱动为零-所谓的**消失梯度(vanishing gradients)**问题。

为了解决这些问题，已经设计了更复杂的网络体系结构来显式管理随时间推移维护相关上下文的任务。更具体地说，网络需要学会忘记不再需要的信息，并记住仍然需要做出决策的信息。

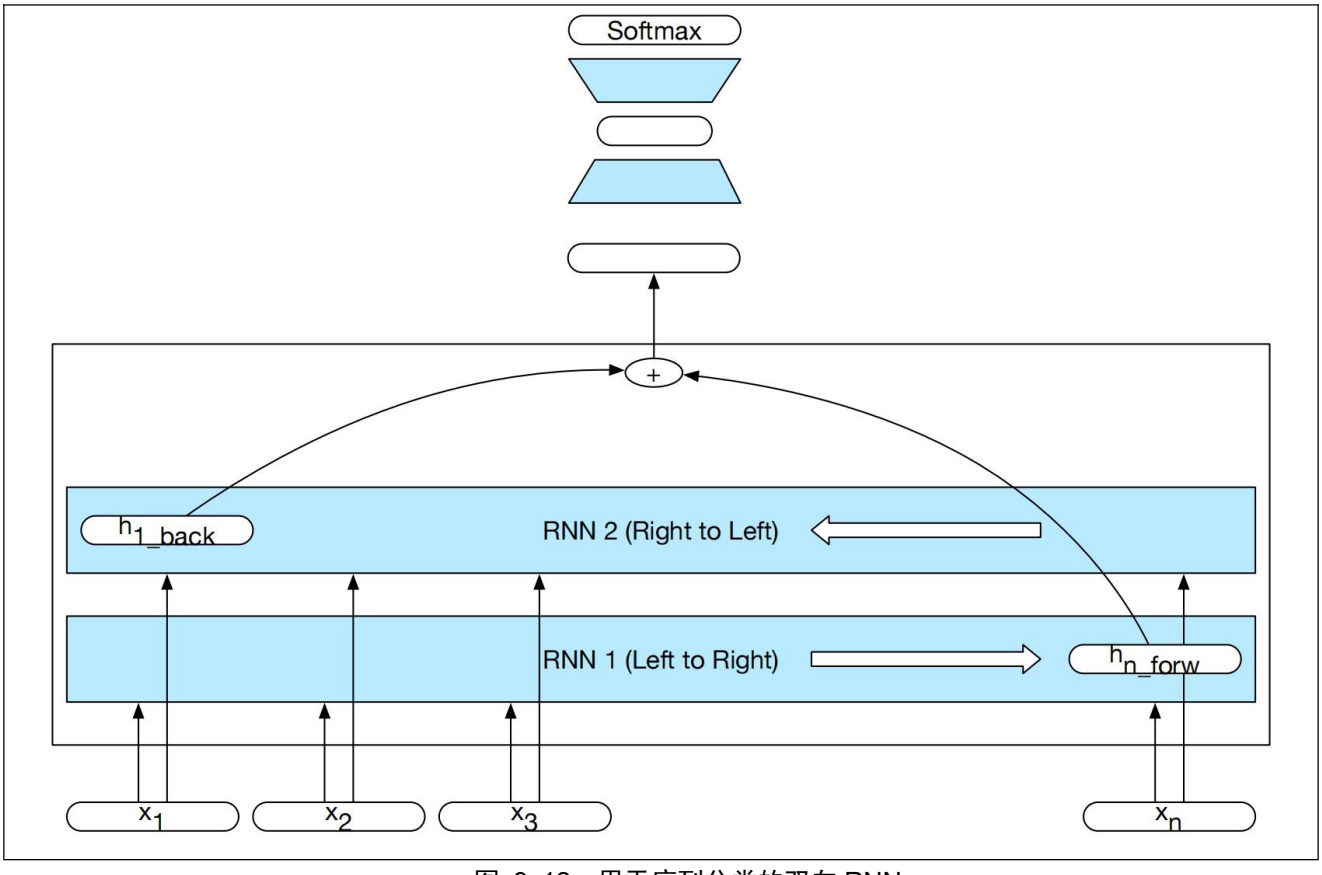


图 9-12：用于序列分类的双向 RNN

图注：向前和向后遍历的最终隐藏单元被组合以代表整个序列。该组合表示用作后续分类器的输入。

9.3.1. 长短期记忆(LSTM)

长短期记忆(LSTM)网络(Hochreiter 和 Schmidhuber，1997 年)将上下文管理问题分为两个子问题：从上下文中删除不再需要的信息，以及添加以后进行决策时可能需要的信息。解决这两个问题的关键是学习如何管理此上下文，而不是将策略硬编码到体系结构中。LSTM 通过首先在体系结构中添加一个显式的上下文层(除了通常的循环隐藏层之外)，并通过使用专用的神经单元，这些单元利用**门(gate)**来控制信息流入和流出，这些单元组成了网络层。这些门是通过使用其他权重实现的，这些权重依次对输入、先前的隐藏层和先前的上下文层进行操作。

LSTM 中的门具有相同的设计模式。每一个都由前馈层组成，其后是 S 型激活函数，然后是与该选通的层进行逐点乘法。选择 S 型作为激活函数是因为它倾向于将其输出推到 0 或 1。将其与逐点乘法相结合

具有类似于二进制**掩码(mask)**的效果。一些值几乎保持不变(即=1)，这些值是指：在选通的层中，与掩码中近似 1 的值对齐的那些值；另一些值基本被抹去(即=0)，这些值是指：与较低值对应的那些值。

我们要考虑的第一个门是**遗忘门(forget gate)**。这个门的目的是从上下文中删除不再需要的信息。遗忘门计算先前状态的隐藏层和当前输入的加权和，并通过一个 sigmoid。然后将此掩码与上下文向量相乘，以从上下文中删除不再需要的信息。

$$f_t = \sigma(U_f\, h_{t-1} + W_f\, x_t)$$

$$k_t = c_{t-1} \odot f_t$$

下一个任务是计算我们需要从之前的隐藏状态和当前输入中提取的实际信息——这与我们在所有循环网络中使用的基本计算方法相同。

$$g_t = tanh(U_g\, h_{t-1} + W_g\, x_t) \tag{9.3}$$

接下来，我们为**加法门(add gate)**生成掩码，以选择要添加到当前上下文的信息。

$$i_t = \sigma(U_i\, h_{t-1} + W_i\, x_t) \tag{9.4}$$

$$j_t = g_t \odot i_t \tag{9.5}$$

接下来，我们将它添加到修改后的上下文向量中，以获得新的上下文向量。

$$c_t = j_t + k_t \tag{9.6}$$

我们将使用的最后一个门是**输出门(output gate)**，它用于决定当前隐藏状态需要什么信息(而不是为未来的决策需要保留什么信息)。

$$o_t = \sigma(U_o\, h_{t-1} + W_o\, x_t) \tag{9.7}$$

$$h_t = o_t \odot tanh(c_t) \tag{9.8}$$

$$\tag{9.9}$$

图 9.13 给出了单个 LSTM 单元的完整计算。给定各种门的适当权值，LSTM 接受上下文层、前一时间步的隐藏层以及当前输入向量作为输入。然后它生成更新的上下文和隐藏向量作为输出。隐藏层 h_t 可以作为堆叠 RNN 中后续层的输入，也可以为网络的最后一层生成输出。

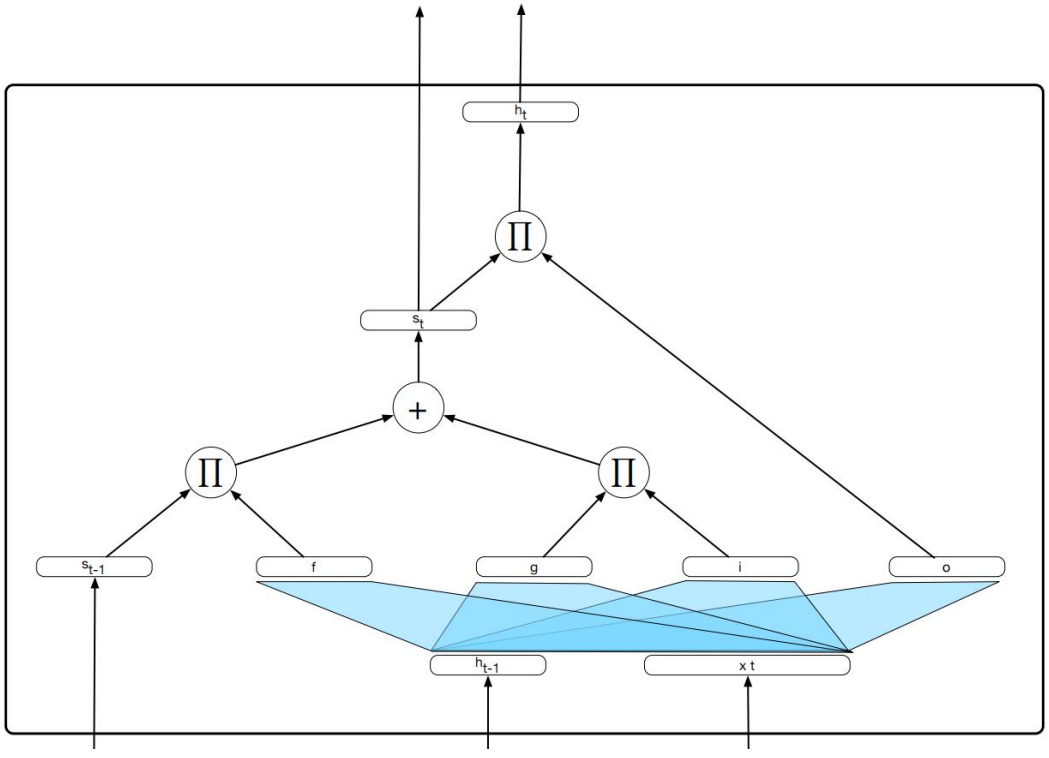


图 9-13：单个 LSTM 单元的计算图

图注：每个单元的输入由当前输入 x_t、之前的隐藏状态 h_{t-1} 和之前的上下文 c_{t-1} 组成。输出是一个新的隐藏状态 h_t 和一个更新的上下文 c_t。

9.3.2. 门控循环单元(GRU)

LSTMs 为我们的循环网络引入了相当多的附加参数。现在我们有 8 组权重需要学习(即每个单元中的 4 个门的 U 和 W)，而在简单的回归单元中我们只有 2 个。这些额外的参数增加了训练成本。门控循环单元(GRUs)(Cho 等人，2014)通过免除使用单独的上下文向量，并通过将门的数量减少到 2 个(重置门 r 和更新门 z)来减轻这一负担。

$$r_t = \sigma(U_r\, h_{t-1} + W_r\, x_t) \tag{9.10}$$

$$z_t = \sigma(U_z\, h_{t-1} + W_z\, x_t) \tag{9.11}$$

与 LSTM 一样，在这些门的设计中使用 S 形，产生的结果类似二进制掩码值。若掩码值≈0，则阻止；若掩码值≈1，则放行。重置门的目的是确定先前隐藏状态的哪些方面与当前上下文相关，以及哪些内容可以忽略。这是通过将 r 与先前的隐藏状态的值进行逐元素乘法来实现的。然后，我们在计算时间 t 的新隐藏状态的中间表示时使用此掩码值。

$$\tilde{h}_t = tanh(U(r_t \odot h_{t-1}) + W\, x_t) \tag{9.12}$$

更新门 z 的工作是确定将在新的隐藏状态下直接使用此新状态的哪些方面，以及需要保留先前状态的哪些方面以备将来使用。这可以通过使用 z 中的值在旧的隐藏状态和新的隐藏状态之间进行插值来实现。

$$h_t = (1-z_t)h_{t-1} + z_t\tilde{h}_t \tag{9.13}$$

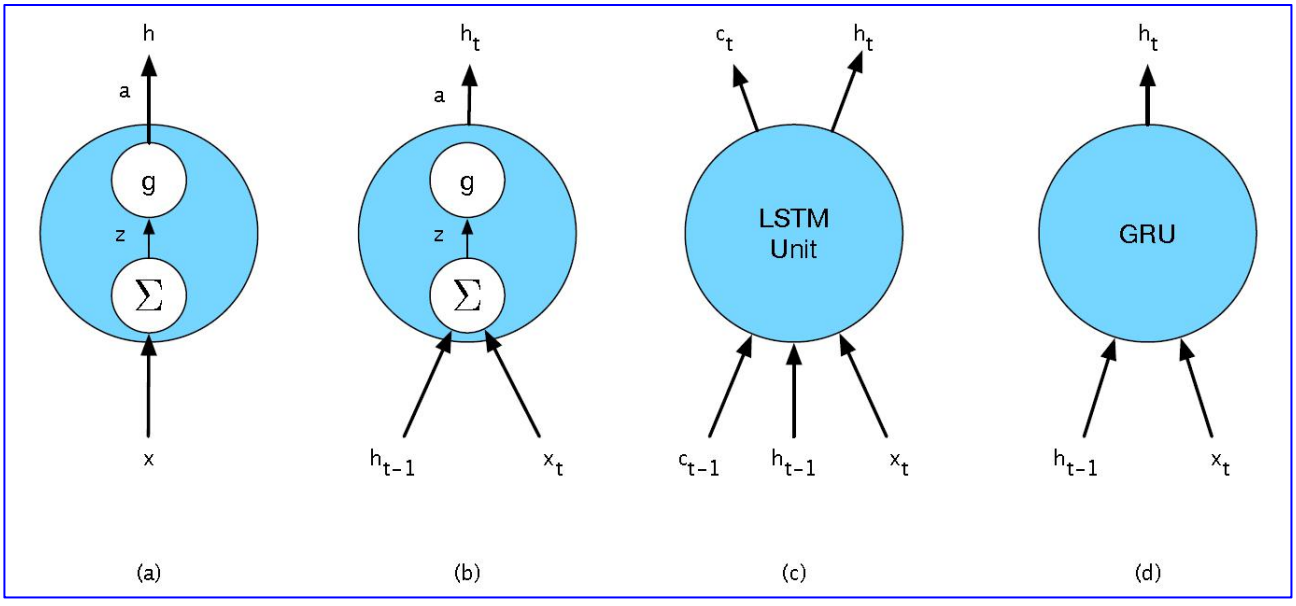


图 9-14：FF、SRN、LSTM 和 GRU

图注：(a)前馈(FF)、(b)简单循环网络(SRN)、(c)长短期记忆(LSTM)和(d)门循环单元(GRU)中使用的基本神经单元。

9.3.3. 门控单元、层和网络

LSTMs 和 GRUs 中使用的神经单元明显比基本的前馈网络中使用的神经单元复杂得多。幸运的是，这种复杂性被封装在基本处理单元中，允许我们保持模块化，并轻松地试验不同的架构。要了解这一点，考虑图 9.14，它说明了与每种单元相关的输入和输出。

最左边的(a)是基本前馈单元，由一组权值和一个激活函数决定其输出，在层中排列时，层内单元之间没有连接。(b)表示简单循环网络中的单元。现在有两个输入和一组额外的权重。然而，仍然有一个单独的激活函数和输出。右侧的 LSTM(c)和 GRU(d)单元增加了复杂性，被封装在这些单元本身中。在基本循环单元(b)上，LSTM 唯一的附加外部复杂性是附加上下文向量作为输入和输出的存在。GRU 单元具有与简单循环单元相同的输入和输出架构。

这种模块化是 LSTM 和 GRU 单元功能强大和广泛应用的关键。LSTM 和 GRU 单元可以替换为第 9.2.6 节中描述的任何网络体系结构。而且，与简单的 RNN 一样，可以将利用门控单元的多层网络展开为深度前馈网络，并通过反向传播以常规方式对其进行训练。

9.4. Self-Attention 网络：Transformer

尽管 LSTM 能够缓解由于 RNN 的循环而导致的远程信息丢失的问题，但潜在的问题仍然存在。通过扩展的一系列循环连接向前传递信息会导致相关信息的丢失和训练上的困难。此外，循环网络的固有顺序性质抑制了并行计算资源的使用。这些考虑导致了 **Transformers** 的发展：一种序列处理方法，该方法消除了循环连接，并返回到使人联想起第 7 章中所述的完全连接网络的体系结构。

Transformers 将输入向量($x_1, \ldots, x_n$)的序列映射到相同长度的输出向量($y_1, \ldots, y_n$)的序列。Transformers 由网络层的堆栈组成，这些网络层由简单的线性层、前馈网络以及围绕它们的自定义连接组成。除了这些标准组件之外，Transformers 的关键创新是使用**自注意力(self-attention)**层。我们将首先描述自注意力的工作原理，然后再回到如何适应更大的 Transformer 模块。自注意力允许网络直接从任意大小的上下文中提取和使用信息，而无需像 RNN 中那样通过中间的循环连接传递信息。在本章中，我们将重点关注自注意力在语言建模和自回归生成问题上的应用，这些问题要使用过去的上下文。在后面的章节中，我们将返回自注意力和 Transformer 的更广泛应用。

图 9.15 说明了单个因果关系或向后看的自注意力层中的信息流。与整个 Transformer 一样，自注意力层将输入序列(x_1，…，x_n)映射到相同长度(y_1，…，y_n)的输出序列。在处理输入中的每个项(item)时，模型可以访问所有输入(包括正在考虑的输入)，但是无法访问有关当前输入之外的输入信息。此外，为每个项执行的计算独立于所有其他计算。第一点确保我们可以使用这种方法创建语言模型，并将它们用于自回归生成；第二点意味着我们可以轻松地并行化此类模型的正向推理和训练。

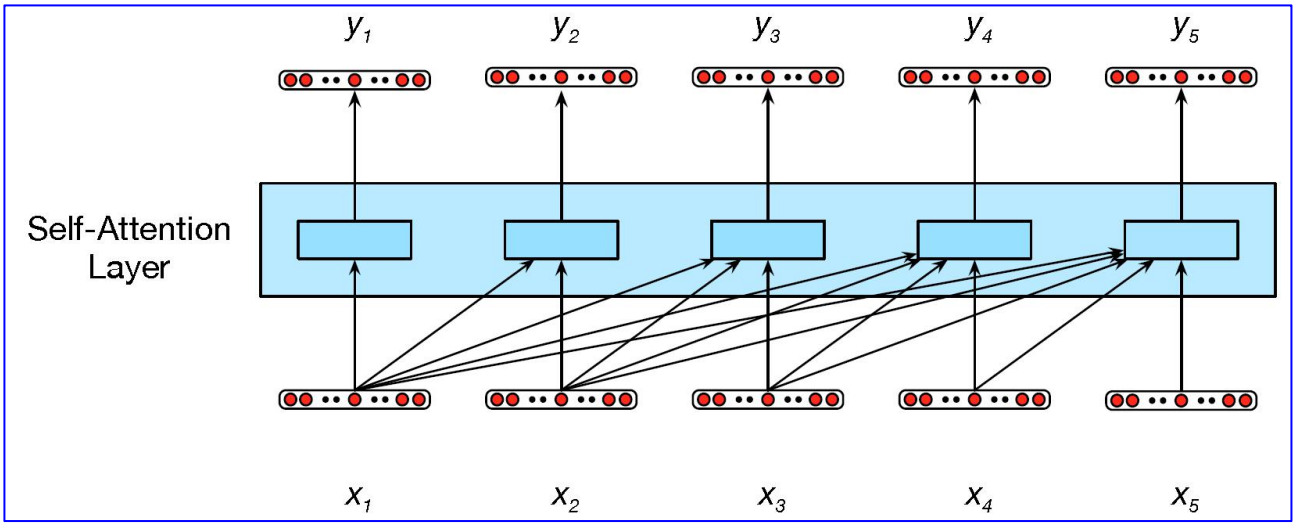


图 9-15：Self-Attention 模型

图注：因果(或掩盖)自注意力模型中的信息流。在处理序列中的每个元素时，模型都会处理直到当前(包括当前)的所有输入。与 RNN 不同，每个时间步的计算都独立于所有其他步骤，因此可以并行执行。

基于注意力的方法的核心是能够将感兴趣的项与其他项的集合进行比较，从而揭示它们在当前上下文中的相关性。在自注意力的情况下，这组比较是针对给定序列中的其他元素的。然后，将这些比较的结果用于计算当前输入的输出。例如，返回到图 9.15，y_3 的计算基于输入 x_3 与它的先前元素 x_1 和 x_2 以及 x_3 本身之间的一组比较。自注意力层中元素之间比较的最简单形式是点积。为了进行其他可能的比较，我们将这些比较的结果称为得分。

$$SCORE(x_i, x_j) = x_i \cdot x_j \tag{9.14}$$

点积的结果是在-∞到∞范围内的标量值，该值越大，正在比较的向量越相似。继续我们的示例，计算 y_3 的第一步将是计算三个分数：x_3x_1，x_3x_2 和 x_3x_3。然后，为了有效利用这些得分，我们将使用 softmax 对它们进行归一化，以创建权重向量 α_{ij}，它表示每个输入与输入元素 i(当前注意力焦点)的比例相关性。

$$\alpha_{ij} = \text{softmax}(score(x_i, x_j)), \quad \forall j \le i \tag{9.15}$$

$$= \frac{exp(score(x_i, x_j))}{\sum_{k=1}^{i} exp(score(x_i, x_k))}, \qquad \forall j \le i \tag{9.16}$$

给定α中的比例分数，然后将目前所见的输入求总和，按各自的α值加权，生成一个输出值 y_i。

$$y_i = \sum_{j \le i} \alpha_{ij} x_j \tag{9.17}$$

公式 9.14 至 9.17 中体现的步骤表示基于注意力的方法的核心：在某些情况下对相关项进行的一组比较，对这些分数进行归一化以提供概率分布，然后使用该分布进行加权求总和。输出 y 是对输入进行直接计算的结果。

不幸的是，这种简单的机制没有提供学习的机会，一切都直接基于原始输入值 x。特别是，没有机会学习单词可以有助于较长输入表示的多种方式。为了进行这种学习，Transformers 以一组权重矩阵的形式包含其他参数，这些参数对输入嵌入进行操作。为了激发这些新参数，请考虑每个输入嵌入在注意力过程中扮演的不同角色(即 Q、K、V)。

- 和其他所有*先前输入*比较时，扮演***当前注意力的焦点***角色，称此为 query(即 Q)。
- 与*当前注意力焦点*比较时，扮演***先前输入***角色，称此为 key(即 K)。
- 用于计算*当前注意力焦点*的输出时，扮演***值***角色，称此为 value(即 V)。

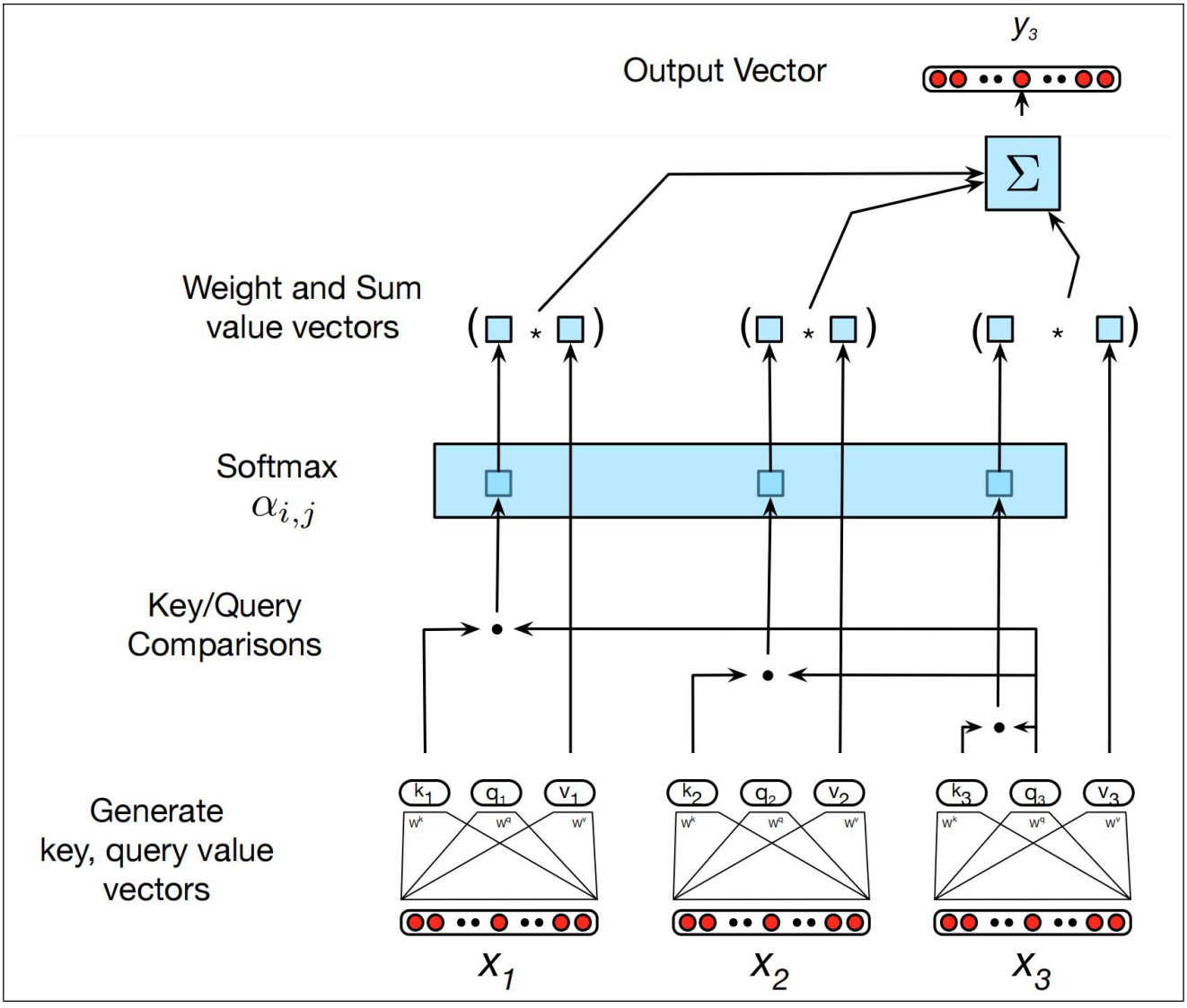


图 9-16：自注意力的计算

图注：使用因果自注意力，对序列中第三个元素的值进行计算。

为了捕获输入嵌入在每个步骤中扮演的不同角色，Transformer 引入了三组权重，我们将其称为 W^Q、W^K 和 W^V。这些权重将用于计算每个输入 x 的线性转换，其结果值将用于后续计算中各自的角色。

$$q_i = W^Q x_i; \qquad k_i = W^K x_i; \qquad v_i = W^V x_i$$

给定大小为 d_m 的输入嵌入，这些矩阵的维数分别为 $d_q \times d_m$, $d_k \times d_m$, $d_v \times d_m$。在原始 Transformer 著作(Vaswani 等人,2017)中，d_m 为 1024，d_q 和 d_v 为 64。

给定这些预测，即当前注意力焦点 x_i 与先前上下文中的元素 x_j 之间的分数，由其 Q 向量 q_i 和先前元素 K 向量 k_j 之间的点积组成。让我们更新之前的比较计算以反映这一点。

$$SCORE(x_i, x_j) = q_i \cdot k_j \tag{9.18}$$

随之产生的 softmax 计算结果α_{ij} 保持不变，但是 y_i 的输出计算现在基于值向量 v 的加权和。

$$y_i = \sum_{j \le i} \alpha_{ij} v_j \tag{9.19}$$

图 9.16 演示了在计算序列的第三个输出 y_3 时的这种计算。

在计算α_{ij} 时出现的实际考虑是由于将点积与 softmax 中的指数结合起来用作比较。点积的结果可以是任意大(正或负)值。对如此大的值求幂可能会导致数值问题，并导致训练过程中梯度的实际损失。为了避免这种情况，需要以适当的方式缩放点积。缩放点积方法将点积的结果除以与嵌入大小相关的因子，然后再将其通过 softmax。一种典型的方法是将点积除以查询(Q)和键(K)向量的维数的平方根，从而导致我们再次更新评分函数。

$$SCORE(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}} \tag{9.20}$$

自注意力过程的描述是从在特定时间点计算单个输出的角度出发的。但是，由于每个输出 y_i 都是独立计算的，因此可以通过使用有效的矩阵乘法例程来并行化整个过程，方法是将输入的嵌入内容打包到单个矩阵中，并将其乘以键(K)、查询(Q)和值(K)矩阵，以生成包含所有键、查询和值向量的矩阵。

$$Q = W^Q X; \quad K = W^K X; \quad V = W^V X$$

给定这些矩阵，我们可以通过在单个矩阵乘法中将 K 和 Q 相乘来同时计算所有必要的 query-key 比较。更进一步，我们可以缩放这些分数，取 softmax，然后将结果乘以 V，从而将整个序列的整个自注意力步骤减少到以下计算：

$$SelfAttention(Q, K, V) = \text{softmax}\frac{QK^T}{\sqrt{d_k}}V \tag{9.21}$$

不幸的是，这个过程有点过头了，因为 QK^T 中的比较计算会为每个查询值对每个键值(包括查询后面的键值)产生一个分数。这在语言建模的设置中是不合适的，因为如果您已经知道下一个单词，那么猜测它是非常简单的。为了解决这个问题，比较矩阵的上三角部分中的元素被清零(设为-∞)，从而消除序列后面的单词的任何知识。

Transformer 模块

自注意力的计算是所谓的 Transformer 模块的核心，该模块除了自注意力层之外，还包括其他前馈层、残余连接和归一化层。图 9.17 展示了一个典型的 Transformer 模块，该模块由一个单一的注意力层，一个完全连接的前馈层，一个残余连接和每个层之后的归一化层组成。然后可以像堆叠 RNN 一样堆叠这些块。

多头注意力

句子中不同的单词可以同时以多种不同的方式相互关联。例如，动词及其在句子中的论元之间可以保持明显的句法、语义和话语关系。单个 Transformer 模块很难学习捕获其输入之间的所有不同种类的并联关系。Transformer 使用多头自注意力层解决了这个问题。这些层是一组自注意力层，称为头，它们位于层中，平行于模型中相同深度的层，每个层都有自己的一组参数。给定这些不同的参数集，每个头都可以学习在同一抽象级别上存在的输入之间的关系的不同方面。

为了实现此概念，在自注意层中的每个头 i 都拥有自己的一组键，查询和值矩阵：W_i^K，W_i^Q 和 W_i^V。它们被用于将每个头的输入分别投影到层 x_i，其余的自注意力计算保持不变。具有 h 个头的多头层的输出包含相同长度的 h 个向量。为了在进一步处理中使用这些矢量，将它们组合在一起，然后减小到原始输入维度 d_m。这是通过拼接来自每个头的输出，然后使用另一个线性投影将其减小到原始输出维度来实现的。

$$MultiHeadAttn(Q, K, V) = W^O(head_1 \oplus head_2 \ldots \oplus head_h)$$

$$head_i = SelfAttention(W_i^Q X, W_i^K X, W_i^V X)$$

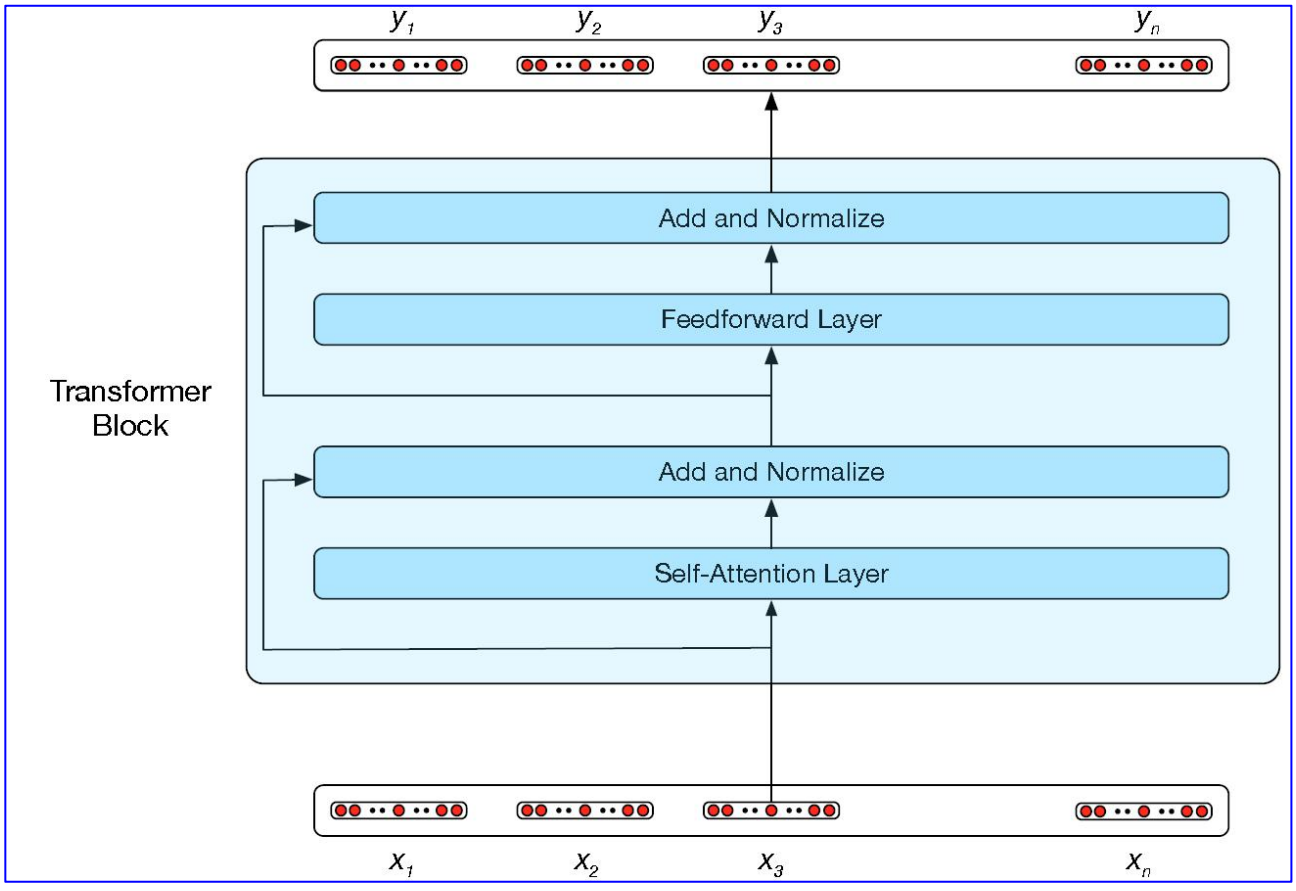


图 9-17：典型的 Transformer 模块

图 9.18 用 4 个自注意力头说明了这种方法。该多头层取代了图 9.17 中所示的 Transfomer 模块中的单个自注意力层，Transfomer 模块的其余部分及其前馈层、残余连接层和层归一化保持不变。

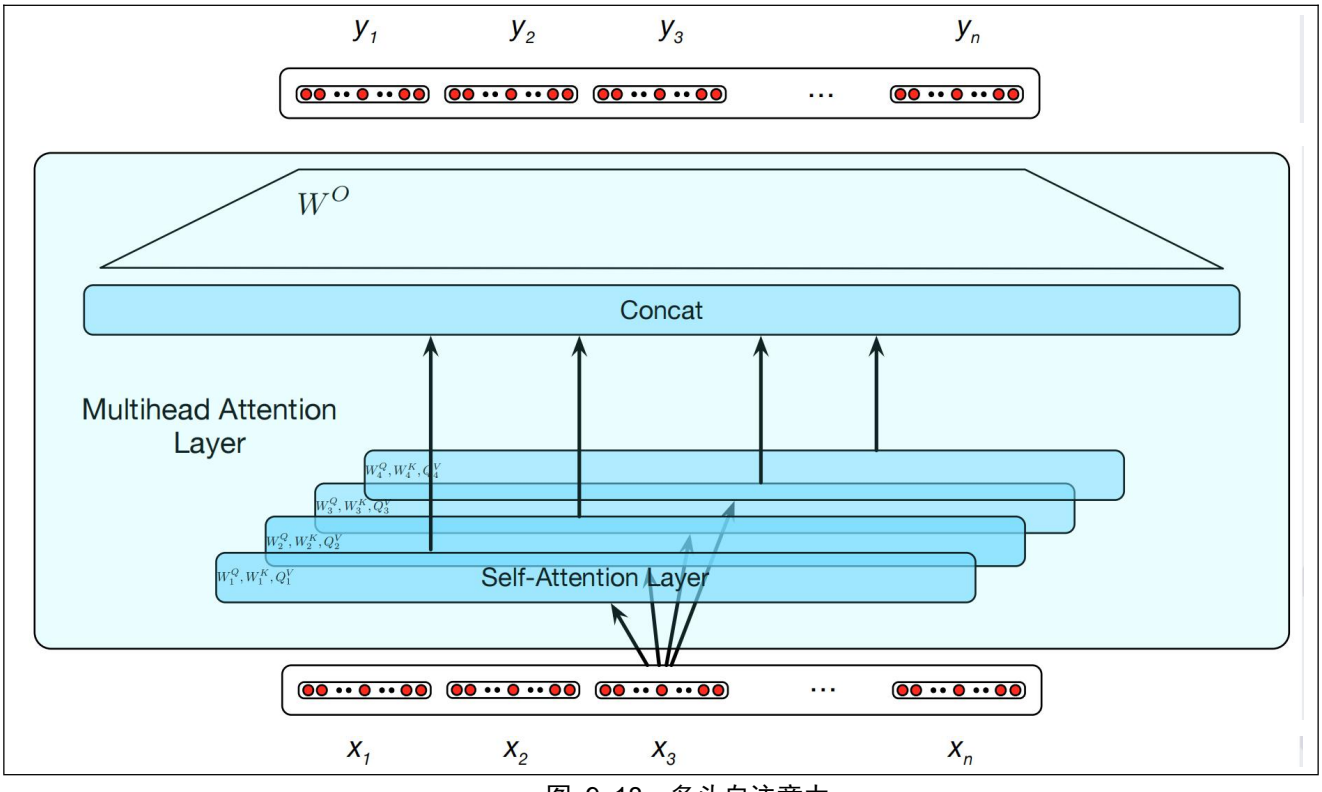


图 9-18：多头自注意力

图注：每个多头自注意层力都有自己的一组键(K)、查询(Q)和值(V)权重矩阵。每个层的输出被拼接起来，然后向下投影到 d_{model} 中，从而产生正确大小的输出。

位置嵌入

利用 RNN，有关输入顺序的信息被纳入模型的性质。不幸的是，Transformer 并非如此。没有什么可以让这种模型利用有关输入序列元素相对或绝对位置的信息。从以下事实可以看出这一点：如果您在前面说明的注意力计算中对输入的顺序进行了打乱，您将得到完全相同的答案。为了解决这个问题，将 Transformer 输入与特定于输入序列中每个位置的**位置嵌入(positional embedding)**相结合。

我们在哪里得到这些位置嵌入？ 一种简单有效的方法是从随机初始化的嵌入开始，这些嵌入对应于每个可能的输入位置，直到某个最大长度。例如，就像有一个单词 fish 嵌入一样，我们也为位置 3 嵌入了一个词。与单词嵌入一样，这些位置嵌入是在训练过程中与其他参数一起学习的。为了产生捕获位置信息的输入嵌入，我们只需将每个输入的词嵌入添加到其对应的位置嵌入中。这种新的嵌入用作进一步处理的输入。

这种方法的潜在问题是，我们输入中的初始位置将有大量的训练样本，而在长度极限之外的位置相应的样本会更少。因此，后面的这些嵌入可能训练不足，并且在测试过程中可能无法很好地泛化。位置嵌入的另一种方法是选择一个静态函数，该函数以捕获位置之间固有关系的方式将整数输入映射到实值向量。也就是说，它捕获了一个事实：相比于输入中的位置 4 与位置 17 的关系，位置 4 与位置 5 的关系更紧密。原始 Transformer 工作中使用了具有不同频率的正弦和余弦函数的组合。

9.4.1. Transformer 作为自回归语言模型

现在，我们已经了解了 Transformers 的所有主要组件，下面我们来研究如何通过半监督学习将它们部署为语言模型。为此，我们将像基于 RNN 的方法一样继续进行操作：给定纯文本训练语料库，训练模型用教师强制方法来预测序列中的下一个单词。图 9.19 说明了一般方法。在每个步骤中，给定所有前面的单词，最终的 Transformer 层会在整个词汇表上产生输出分布。在训练过程中，分配给正确单词的概率用于计算序列中每个项的交叉熵损失。与 RNN 一样，训练序列的损失是整个序列的平均交叉熵损失。

请注意，该图与先前的基于 RNN 的版本之间的关键区别如图 9.6 所示。给定隐藏状态的计算循环，此处输出的计算和每一步的损失本质上都是串行的。使用 Transformer，可以并行处理每个训练项，因为序列中每个元素的输出是分别计算的。训练后，我们可以计算所得模型的困惑度，或像基于 RNN 的模型一样自动回归生成新颖的文本。

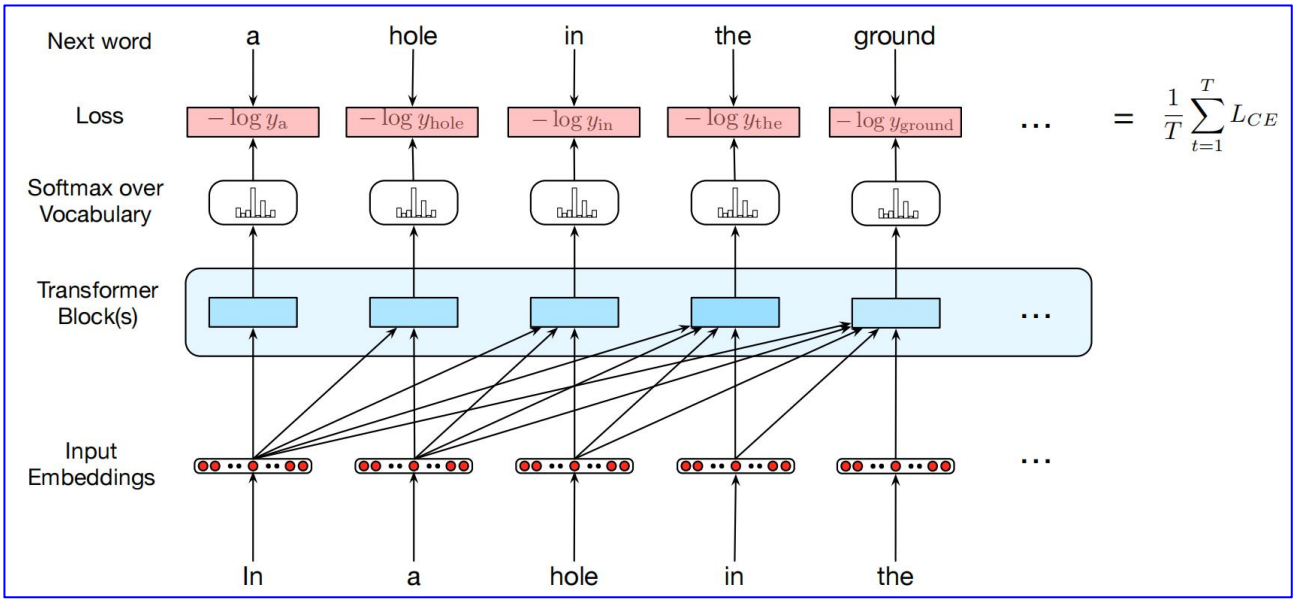


图 9-19：将 Transformer 训练为语言模型

上下文生成和摘要

自回归生成的一个简单变体是许多实际应用的基础，它使用先验上下文来准备自回归生成过程。图

9.20 用**文本补全(text completion)**任务说明了这一点。在这里，标准语言模型被赋予某些文本的前缀，并被要求为其生成可能的补全。请注意，随着生成过程的进行，模型可以直接访问头道(priming)上下文以及所有其自身随后生成的输出。在每个时间步整合整个较早上下文和生成的输出的能力是这些模型强大功能的关键。

文本摘要(text summarization)是基于上下文的自回归生成的实际应用。在这里，任务是获取全长文章，并提供有效的摘要。为了训练基于 Transformer 的自回归模型来执行此任务，我们从一个语料库开始，该语料库由全长文章及其相应的摘要组成。图 9.21 显示了来自广泛使用的摘要语料库的此类数据的示例，该摘要语料库由 CNN 和 Daily Mirror 新闻文章组成。

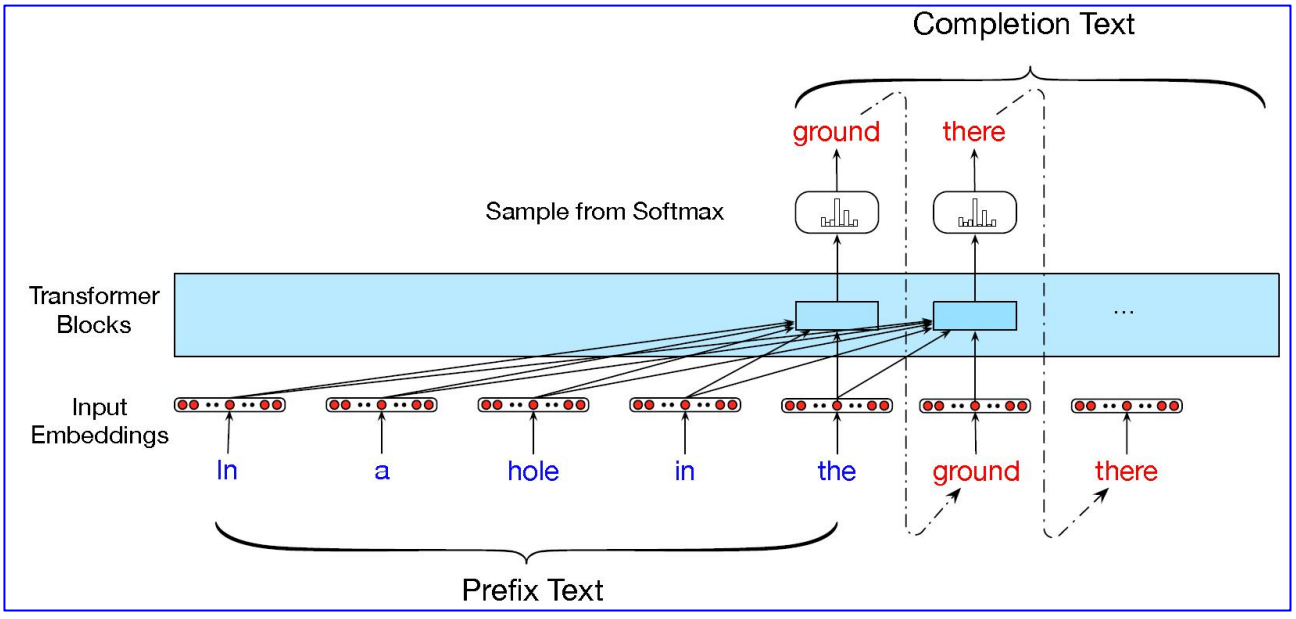


图 9-20：Transformers 的自回归文本补全

Original Article

The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $89, self-styled entrepreneur KyleWaring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says.

But not if you live in New England or surrounding states. “We will not ship snow to any states in the northeast!” saysWaring’s website, ShipSnowYo.com. “We’re in the business of expunging snow!”

His website and social media accounts claim to have filled more than 133 orders for snow – more than 30 on Tuesday alone, his busiest day yet. With more than 45 total inches, Boston has set a record this winter for the snowiest month in its history. Most residents see the huge piles of snow choking their yards and sidewalks as a nuisance, but Waring saw an opportunity.

According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling

deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston.

He joked about shipping the stuff to friends and family in warmer states, and an idea was born. His business slogan: “Our nightmare is your dream!” At first, ShipSnowYo sold snow packed into empty 16.9-ounce water bottles for $19.99, but the snow usually melted before it reached its destination…

Summary

KyleWaring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.

图 9-21：CNN/每日邮报语料库的文章和摘要示例

图注：来自(Hermann 等人, 2015b), (Nallapati 等人, 2016)。

将 Transformers 应用于摘要的一种出乎意料的有效方法是将摘要添加到语料库中的每篇全长文章中，并用独特的标记将两者分开。更正式地说，训练语料库中的每个文章摘要配对$(x_1, \ldots, x_m)$，$(y_1, \ldots, y_n)$都转换为单个训练实例$(x_1, \ldots, x_m, \delta, y_1, \ldots y_n)$，总长度为 n + m +1。将这些训练实例视为长句子，然后像我们之前所做的那样，用教师强制方法来训练自回归语言模型。

经过培训后，以特殊标记结尾的完整文章将用作上下文，引导生成过程并且产生摘要，如图 9.22 所示。请注意，与 RNN 相比，该模型在整个过程中都可以访问原始文章以及新生成的文本。

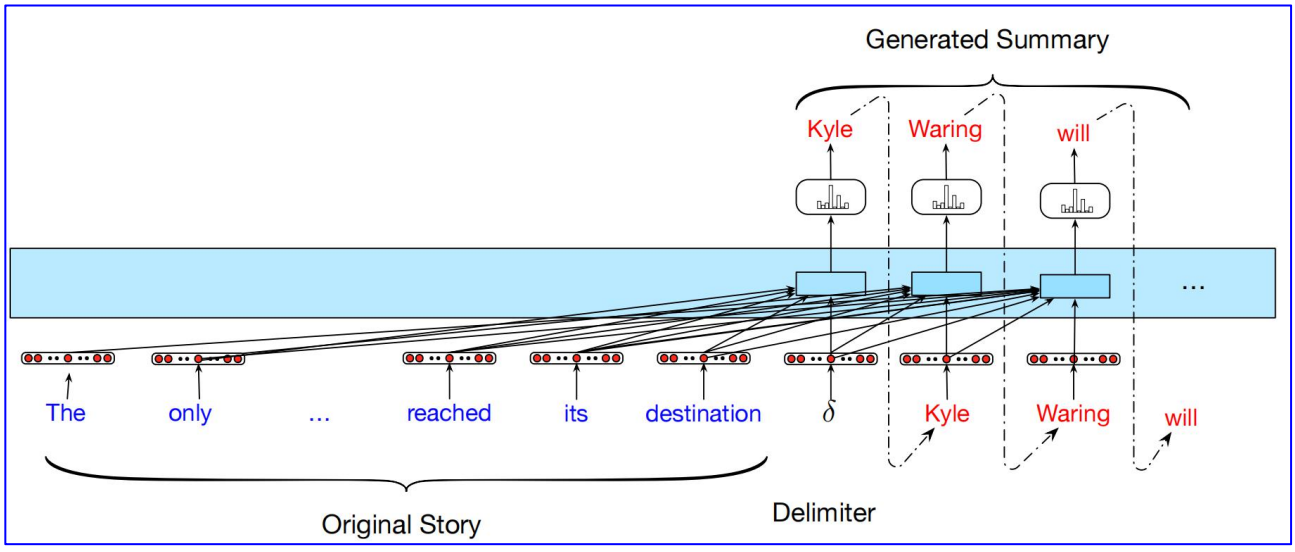


图 9-22：Transformers 用于摘要

正如我们将在后面的章节中看到的那样，此简单方案的各种变体是成功进行文本到文本应用程序(包括机器翻译、摘要和问答)的基础。

9.5. 语言模型的潜在危害

大型神经语言模型表现出许多在第 4 章和第 6 章中讨论的潜在危害。每当将语言模型用于文本生成时(如在 Web 搜索查询完成或电子邮件的预测键入的辅助技术中)，都可能会出现问题(Olteanu 等人，2020)。

例如，语言模型可以生成有毒的语言。Gehman 等人(2020)表明，许多种类的完全无毒的提示仍然可以导致大型语言模型输出仇恨言论和辱骂。Brown 等人(2020)和 Sheng 等人(2019)表明，大型语言模型生成的句子显示出对少数族裔身份(如黑人或同性恋)的消极态度。

实际上，语言模型因其训练数据的分布而在许多方面存在偏见。Gehman 等人(2020 年)表明，大型语言模型训练数据集包括从禁令站点刮取的有毒文字。除了毒性问题之外，发达国家的作者还不成比例地生成了互联网数据，许多大型语言模型都基于 Reddit 的数据进行训练，其作者偏向于男性和年轻人。这种偏倚的人口样本，可能会使一代人偏离到代表性不足的观点或主题。此外，正如我们在第 6 章中看到的嵌入模型一样，语言模型可以放大训练数据中的人口统计特征和其他偏差。

语言模型也可以是生成用于错误信息、网络钓鱼、激进和其他对社会有害的活动的文本的工具(Brown 等人，2020)。(McGuffie 和 Newhouse，2020)显示了大型语言模型如何生成模仿在线极端分子的文本，并有加剧极端主义运动及其激进和招募的风险。

最后，还有重要的隐私问题。和其他机器学习模型一样，语言模型也会泄露其训练数据的信息。

因此，对手可以从语言模型中提取个人训练数据短语，如个人的姓名、电话号码和地址(Carlini 等人 2020 年，使用 Henderson 等人 2017 年介绍的技术)。如果在私人数据集上训练大型语言模型，比如电子健康记录(EHRs)，这就是一个问题。

如何减轻这些危害是自然语言处理中一个重要但尚未解决的研究问题。额外的对无毒子语料的预训练(Gururangan 等人，2020)似乎在一定程度上减少了语言模型产生有毒语言的倾向(Gehman 等人，2020)。分析用于预训练大型语言模型的数据对于理解生成过程中的毒性和偏见以及隐私都很重要，因此，语言模型包括**数据表**或**模型卡**(第 4.10 节)非常重要，数据表提供了用于训练它们的语料库上的完全可复制的信息。

9.6. 总结

本章介绍了**循环神经网络(RNN)**的概念，以及如何将它们应用于语言问题。以下是我们总结的要点:

•在**简单的循环神经网络(SRN)**中，序列每次都作为一个元素自然地处理。

•神经单元在特定时间点的输出既基于当前输入，也基于来自上一时间步的隐藏层的值。

•RNN 可以通过**反向传播算法(**称为时间反向传播(BPTT))的直接扩展来训练。

•RNN 的基于通用语言的应用程序包括：

–概率语言建模，其中模型将概率分配给序列，或序列的下一个元素(在给定前面单词的情况下)。

–使用经过训练的语言模型**自动回归生成**。

–序列标签，为序列的每个元素分配一个标签，如词类标记一样。

–序列分类，将整个文本分配给一个类别，例如垃圾邮件检测、情感分析或主题分类。

•简单的循环网络常会失败，因为要成功地训练它们来解决随时间推移保持有用梯度的问题非常困难。

•更复杂的门控架构，如 **LSTM** 和 **GRU**，被设计来克服这些问题，通过显式管理任务，决定在其隐藏和上下文层中记住和忘记什么。

9.7. 文献和历史说明

在 1980 年代，在圣地亚哥加州大学的并行分布式处理(PDP)小组的背景下，对这里讨论的简单 RNN 的类型进行了有影响的研究。这项工作大部分是针对人类认知建模的，而不是针对实际的 NLP 应用的(Rumelhart 和 McClelland 1986 ，McClelland 和 Rumelhart1986)。

Elman(1990)引入了在前馈网络(Elman 网络)的隐藏层使用循环的模型。Jordan(1986)用输出层的循环研究了类似的架构，Mathis 和 Mozer(1995)在隐藏层之前添加了一个循环上下文层。在(Rumelhart 和 McClelland，1986)中讨论了将循环网络展开为等效前馈网络的可能性。

在认知建模工作的同时，RNN 在信号处理和语音社区的连续领域得到了广泛的研究(Giles 等人，1994)。Schuster 和 aliwal(1997)引入了双向 RNN，并描述了 TIMIT 音子转录任务的结果。

尽管从理论上讲很有趣，但是训练 RNN 和在长序列上管理上下文的困难阻碍了实际应用的进展。随着 Hochreiter 和 Schmidhuber(1997)引入 LSTM，这种情况发生了变化。在信号处理和语言处理的边界任务上表现出令人印象深刻的性能提升，包括音子识别(Graves 和 Schmidhuber，2005)，手写识别(Graves 等人，2007)和最重要的语音识别(Graves 等人，2013b)。

由于 Collobert 和 Weston(2008)以及 Collobert 等人(2011)的工作，人们对将神经网络应用于实际的自然语言处理问题的兴趣大增。这些努力利用了词汇嵌入、卷积网络和端到端的训练。他们在许多标准的共享任务中，包括词类标记、分块、命名实体识别和语义角色标记，在没有使用人工工程特征的情况下，展示了接近最先进的性能。

将 LSTM 与基于 word2vec(Mikolov 等人，2013)和 GLOVE(Pennington 等人，2014)的词嵌入的预训练集合相结合的方法迅速主导了许多常见任务：词类标记(Ling 等人，2015)，句法分块(Søgaard 和 Goldberg，2016)，并通过 IOB 标记(Chiu 和 Nichols 2016，Ma 和 Hovy 2016)，意见挖掘(Irsoy 和 Cardie，2014)，语义角色标记(Zhou 和 Xu，2015a)，AMR 解析(Foland 和 Martin，2016)。与早期统计机器学习的快速发展一样，这些进展是由于 CONLL、SemEval 提供的训练数据和其他共享任务，以及 Ontonotes (Pradhan 等人，2007b)和 PropBank (Palmer 等人，2005)等共享资源的可用性而实现的。

9.8. 练习

(无)

10. 上下文嵌入

Placeholder
(空空如也)

10.1. 上下文概述

(空空如也)

11. 机器翻译和编码器-解码器模型

> “I want to talk the dialect of your people.
> It’s no use of talking unless people understand what you say.”
> (“我想说你们民族的方言。说话是没有用的，除非人们能听懂你说的话。”)
>
> Zora Neale Hurston, *Moses, Man of the Mountain* 1939, p. 121

这一章介绍了**机器翻译(machine translation MT)**，使用计算机将一种语言翻译成另一种语言。

当然，总的来说，翻译，如文学或诗歌的翻译，是一项困难的、迷人的和强烈的人类努力，就像人类在其他领域的创造力一样丰富。



因此，目前形式的机器翻译着重于许多非常实际的任务。机器翻译当前最常见的用法是用于**信息访问(information access)**。我们可能希望翻译一些网上说明，例如最喜欢的菜谱或组装家具的步骤。或者，我们可能想阅读报纸上的文章，或者从在线资源(如 Wikipedia 或政府网页)中获取外语信息。用于信息访问的 MT 可能是 NLP 技术最常见的用途之一，仅 Google Translate(如上所示)每天就在 100 多种语言之间翻译成千亿个单词。

机器翻译的另一个常见用途是帮助人工翻译。MT 系统通常用于产生草稿翻译，并由人工翻译在**后期编辑(post-editing)**阶段进行固定。此任务通常称为**计算机辅助翻译(computer-aided translation，CAT)**。CAT 通常用作**本地化(localization)**的一部分：使内容或产品适应特定语言社区的任务。

最后，MT 的最新应用是满足即时的人际交流需求。这包括**增量翻译**(incremental translation)，在整个句子完成之前实时翻译语音，这在同声传译中通常使用。例如，可以使用以图像为中心的翻译来将手机摄像头图像上的文本的 OCR 用作 MT 系统的输入，以翻译菜单或路牌。

MT 的标准算法是**编码器-解码器(encoder-decode)**网络，也称为**序列到序列(sequence to sequence)**网络，该体系结构可以用 RNN 或 Transformer 实现。我们在之前的章节中已经看到 RNN 或 Transformer 体系结构可用于进行分类(例如，将句子映射到正或负的情绪标记以进行情绪分析)，或可用于进行序列标签(例如，给输入句子中的每个词都分配一个词类或命名实体标记)。对于词类标记，请记住输出标记直接与每个输入词相关联，因此我们可以将标记建模为每个输入词 x_t 的输出 y_t。

编码器-解码器或序列-序列模型用于不同类型的序列建模，其中输出序列是整个输入序列器的一个复杂函数;我们必须从一个输入单词或符记序列映射到一个标记序列，这些标记不仅是单个单词的直接映射。

机器翻译就是这样一个任务：目标语言的单词不一定在数量或顺序上与源语言的单词一致。考虑把下面的英语句子翻译成日语。

(11.1)

English:	*He*	*wrote*	*a letter*	*to*	*a friend*
Japanese:	*tomodachi*	*ni*	*tegami-o*		*kaita*
	friend		to letter		wrote

请注意，句子的元素在不同语言中的位置非常不同。在英语中，动词位于句子的中间；而在日语中，动词 kaita 位于结尾。日语句子不需要代词 he，而英语则需要。

语言之间的这种差异可能相当复杂。在下面这个来自联合国的实际句子中，请注意这个中文句子(我们用红色标出了汉字的逐字注解)和对应的英文句子之间的许多变化。

(11.2) 大会/General Assembly 在/on 1982 年/1982 12 月/December 10 日/10 通过了/adopted 第 37 号/37th 决议/resolution，核准了/approved 第二次/second 探索/exploration 及/and 和平 peaceful 利用/using 外层空间/outer space 会议/conference 的/of 各项/various 建议/suggestions。

On 10 December 1982 , the General Assembly adopted resolution 37 in which it endorsed the recommendations of the Second United Nations Conference on the Exploration and Peaceful Uses of Outer Space .

注意英语和汉语有很多不同的方式。例如，顺序在主要方面有所不同。名词短语的中文顺序是“peaceful using outer space conference of suggestions”，而英语则是“ suggestions of the ... conference on peaceful use of outer space”。而且顺序略有不同(日期顺序不同)。英语在很多地方都要求(而中文没有要求)the，并增加了一些中文中不需要的细节(比如“in which”和“it”)。中文并没有在语法上标记复数名词(不像在英语中“recommendations”中带有“-s”)，因此中文必须使用修饰词“各项/various”来表明不仅有一个推荐。英语有些单词大写，有些单词不大写。

编码器-解码器网络在处理这类复杂的序列映射情况方面非常成功。事实上，编码器-解码器算法不仅适用于 MT;对于涉及两个序列之间复杂映射的许多其他任务来说，这是最先进的技术。这些技术包括：

摘要(summarization)(我们从长文本映射到它的摘要，如一个标题或**提炼(abstract)**)，对话(我们将用户所说的内容映射到对话系统应该响应的内容)，语义解析(我们将单词的字符串映射到语义表示形式，就像逻辑或 SQL)，还有许多其他内容。

我们将在第 11.2 节中介绍该算法，并在接下来的章节中给出模型的重要组成部分，如波束搜索解码，我们将讨论 MT 如何被评估，并介绍流行的 BLEU 度量。

但首先，在下一节中，我们开始总结 MT 的语言背景：考虑翻译任务时重要的语言之间的关键差异。

11.1. 语言差异与类型学

人类语言的某些方面似乎是通用的，适用于每一种语言，或者在统计学上是普遍存在的，适用于大多数语言。语言作为人类交际系统的功能作用产生了许多共性。例如，每种语言似乎都有用来指人、谈论吃喝、表示礼貌或不礼貌的词。还有结构上的语言共性；例如，每一种语言似乎都有名词和动词(第 8 章)，都有提问题的方法，或发出命令，表明同意或不同意的语言机制。

然而，语言在许多方面也存在差异，理解这种**翻译差异(translation divergence)**的原因将有助于我们建立更好的机器翻译模型。我们经常区分两种差异：必须逐一处理的**独特(idiosyncratic)**和词汇差异(“狗”一词在语言之间的差异很大)；我们可以用一般方式建模的**系统(systematic)**差异(许多语言将动词放在直接宾语之前；其他将动词放在直接宾语之后)。对这些系统的跨语言相似性和差异性的研究被称为语言**类型学(typology)**。这一类型学部分概述了一些影响机器翻译的类型学事实;感兴趣的读者还应该看看 WALS，即《世界语言结构地图集》，它给出了许多关于语言的类型学事实(Dryer 和 Haspelmath, 2013)。

11.1.1. 词序类型学

正如我们在上面比较英语和日语的例子中所暗示的那样，语言在简单陈述句中的动词、主语和宾语的基本词序上有所不同。例如，德语、法语、英语和普通话都是 **SVO**(主语-动词-宾语)语言，这意味着动词往往位于主语和宾语之间。相比之下，印地语和日语是 **SOV** 语言，这意味着动词往往出现在基本从句的末尾，而爱尔兰语和阿拉伯语是 **VSO** 语言。两种具有相同基本词序类型的语言通常有其他相似之处。例如，VO 语言通常有介词，而 OV 语言通常有后置词。

让我们更详细地看看(11.3)例子。在这个 SVO 英语句子中，动词 write 后面跟着它的宾语 a letter 和介词短语 to a friend，其中介词 to 后面跟着它的论元 a friend。阿拉伯语的 VSO 顺序，动词也放在宾语和

介词之前。相比之下，在接下来的日本例子中，这些顺序都颠倒了;动词前面有论元，后置词跟在论元后面。

(11.3)

English:	*He wrote*	*a letter to*	*a friend*
Japanese:	*tomodachi*	*ni tegami-o*	*kaita*
	friend	to letter	wrote
Arabic:	*katabt*	*risāla*	*li ṣadq*
	wrote	letter	to friend

其他种类的排序首选项因语言而异。在某些 SVO 语言(例如英语和普通话)中，形容词倾向于出现在动词之前，而在其他语言(例如西班牙语和现代希伯来语)中，则形容词出现在名词之后：

(11.4) **Spanish** *bruja verde* **English** *green witch*

图 11.1 给出了其他词序差异的例子。所有这些语言之间的词序差异都会导致翻译问题，这就要求系统在生成输出时进行大量的结构重新排序。

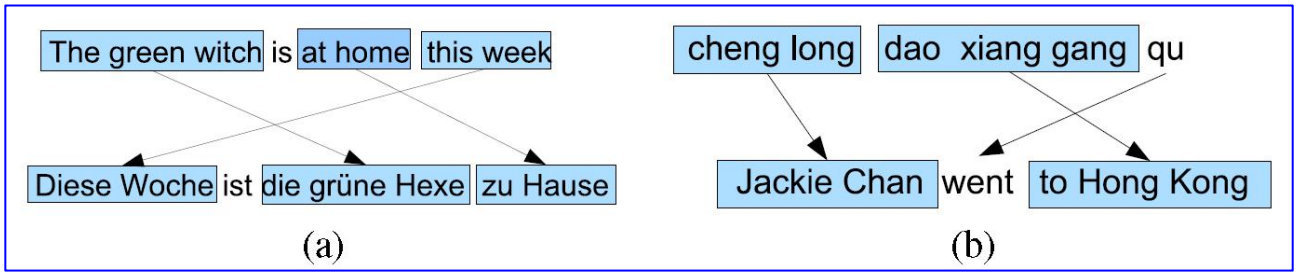


图 11-1：其他词序差异的例子

图注：(a)在德语中，副词出现在最开始的位置，而在英语中则更自然一些，时态动词出现在第二位置。

(b)在汉语中，表达目标的介词短语通常在动词前(pre-verbally)出现，这与英语不同。

11.1.2. 词汇差异

当然，我们也需要将单个单词从一种语言翻译成另一种语言。对于任何翻译，合适的词都可能根据上下文而有所不同。例如，英语源语言的单词 bass 在西班牙语中可以作为鱼 lubina 或乐器 bajo 出现。德语用两个截然不同的词来表示在英语中被称为 wall 的东西:Wand 表示建筑物内的墙，Mauer 表示建筑物外的墙。在英语中， brother 一词用于任何男性同胞，而中文和许多其他语言在 older brother 和 younger brother 中分别使用不同的词(普通话分别用“哥哥”和“弟弟”)。在所有这些情况下，翻译英文中的 bass、wall 或 brother 都需要一种专业化，消除单词不同用法的歧义。因此，机器翻译和词义消歧(第 18 章)这两个领域是紧密相连的。

有时，一种语言比另一种语言对单词选择施加更多的语法约束。我们在上面看到英语标记名词是单数还是复数。而普通话没有这些标记。例如，法语或西班牙语会在形容词上标记语法性别，因此将英语翻译成法语需要指定形容词类别。

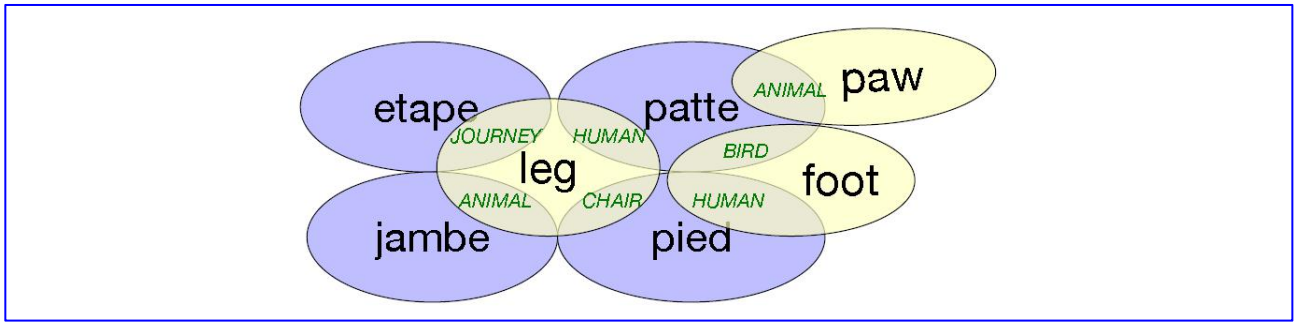


图 11-2：英语和法语翻译之间的复杂重叠

语言在词汇上划分概念空间方面的差异可能比这种一对多的翻译问题更复杂，从而导致多对多映射。例如，图 11.2 总结了 Hutchins 和 Somers(1992)在将英语的 leg、foot 和 paw 翻译成法语时所讨论的一些复杂性。例如，当 leg 用于形容一种动物时，它被翻译成法语 jambe;如果是关于行程的 leg，则我们使用法语的 etape；如果是关于椅子的 leg，则我们使用法语的 pied。

此外，一种语言可能会有**词汇空白(lexical gap)**，即在没有解释性脚注的情况下，任何单词或短语都不能在另一种语言中表达一个单词的含义。例如，英语中没有一个单词能够与汉语中的 xi'ao(孝)或者日语中的 oyaḱok(孝顺)相对应(在英语中，必须处理一些尴尬的词组，例如 filial piety(孝顺)或 loving child(可爱的孩子)，或者是 good son/daughter(好儿女))。

最后，语言在事件的概念特性如何映射到特定单词方面存在系统差异。Talmy(1985，1991)指出，语言可以通过在动词或“**外围词(satellites)**”上标记运动方向和运动方式来表征：助词、介词短语或副词短语。例如，在英语 a bottle floating out of a cave 中表示方向的是助词 out，而在西班牙语中，表示方向的是动词：

(11.5)
English: *The bottle flfloated out.*
Spanish: *La botella salió flotando.*
The bottle exited floating.

动词框架(verb-framed)语言在动词上标记运动的方向(让外围词标记运动的方式)，如西班牙语 acercarse 的“approach(接近)”，alcanzar 的“reach(到达)”，entrar 的“enter(进入)”，salir 的“exit(退出)”。**外围词框架(satellite-framed)**语言在外围词上标记运动的方向(留下动词来标记运动的方式)，如英语 crawl out(爬行)，float off(漂浮)，jump down(跳下来)，run after(追赶)。像日语、泰米尔语以及罗曼语、闪米特语和玛雅语系的许多语言都是动词结构的;汉语和非罗曼语系的印欧语言，如英语、瑞典语、俄语、印地语和波斯语，都是外围词框架(Talmy 1991, Slobin 1996)。

11.1.3. 形态类型学

在**形态学(Morphologically)**上，语言通常具有两个维度的变异特征。第一个维度是每个单词的语素数量，从**孤立(isolating)**语言(如越南语和广东话，每个单词通常有一个语素)，到**多综合(polysynthetic)**语言(如西伯利亚尤皮克语(“爱斯基摩语”)，一个单词可能有很多语素，相当于英语的整个句子)。第二个维度是语素可分段的程度，范围从诸如土耳其语(语素具有相对清晰的边界)之类的**粘着(agglutinative)**语言，到诸如俄语之类的**融合(fusion)**语言(其中单个词缀可以混淆多个词素，例如 stolom 中的 -om)(table-SG-INSTR-DECL1)，它融合了工具格、单数和第一变格的不同形态范畴。

在词法丰富的语言之间进行翻译需要处理低于单词级别的结构，因此，现代系统通常使用子词模型，如 11.7.1 节中的 wordpiece 或 BPE 模型。

11.1.4. 指代密度

最后，语言在与它们往往忽略的事物相关的类型维度上有所变化。有些语言，比如英语，要求我们在谈论话语中给定的指代物时使用显式代词。然而，在其他语言中，我们有时可以省略代词，如下面的西班牙语例子所示[1]:

(11.6) [E1 jefe]$_i$ dio con un libro. Ø$_i$ Mostró a un descifrador ambulante.
[The boss] came upon a book. [He] showed it to a wandering decoder.

可以省略代词的语言称为**代词省略(pro-drop)**语言。即使在代词省略语言中，省略频率也存在明显差异。例如，日语和中文比西班牙语往往会忽略很多。跨语言的这种变化的维度称为**指代密度(referential density)**的维度。我们说倾向于使用更多代词的语言比使用更多零(即不使用代词,零指代)的语言具有更高的指代密度。指代稀疏语言，例如中文或日语，需要听众进行更多推理工作以恢复先行词，也称为**冷语言(cold language)**。更加明确的语言以及使听众更容易使用的语言称为**热语言(hot language)**。热和冷是从马歇尔·麦克卢汉(Marshall McLuhan，1964)的区别中借来的，例如电影等热媒体为观看者填充了许多细节，而漫画等冷媒体则需要读者做更多的推论性工作来填充表示形式(Bickel,2003)。

把大量的代词省略语言(如汉语或日语)翻译成非代词省略语言(如英语)可能很困难，因为模型必须以某种方式识别每个零指代，并恢复正在谈论的人或事，以便插入适当的代词。

[1] 这里我们使用Ø表示法;我们将在第 22 章进一步介绍并讨论这个问题。

11.2. 编码器-解码器模型

编码器-解码器网络或序列-序列网络，是能够生成上下文相关的任意长度的输出序列的模型。编码器-解码器网络已被广泛应用于包括机器翻译、摘要、问答和对话的应用。

这些网络背后的关键思想是用编码器网络来接受输入序列并为其创建上下文表示，通常称为上下文。然后将该表示传递给解码器以生成特定于任务的输出序列。图 11.3 说明了该体系结构：

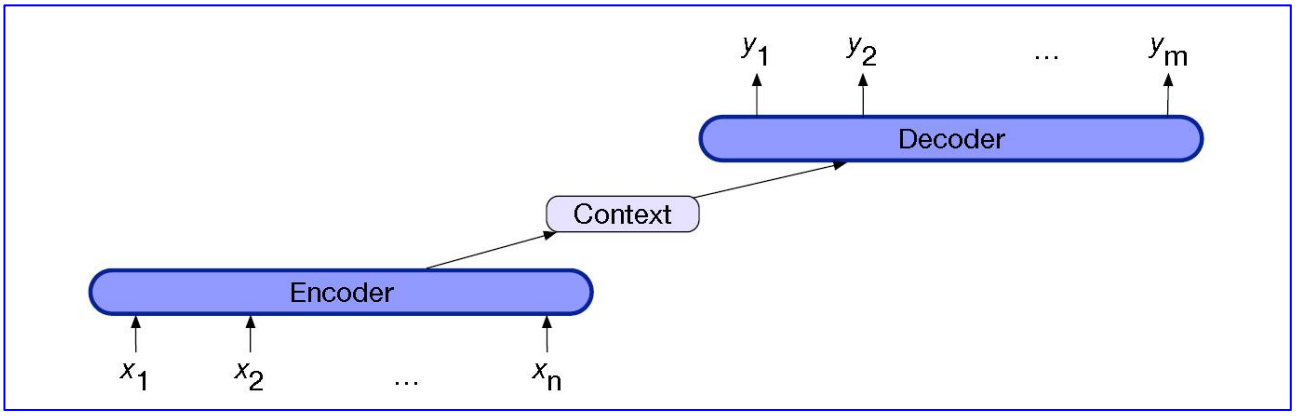


图 11-3：编码器-解码器架构

图注：上下文是输入的隐藏表示的函数，解码器可以以多种方式使用它。

编解码器网络由三部分组成:

1. **编码器**，它接受输入序列x_1^n，并生成相应的上下文表示序列h_1^n。LSTMs、GRUs、CNNs、Transformers 都可以作为编码器。

2. **上下文**，它是向量 c，是h_1^n的函数，它将输入的本质传达给解码器。

3. **解码器**，它接受 c 作为输入，生成任意长度的隐藏状态序列h_1^m，从中可以得到相应的输出状态序列y_1^m。就像编码器一样，解码器可以用任何一种序列结构来实现。

11.3. 带 RNN 的编解码器

让我们从描述一个基于一对 RNN 的编码器-解码器网络开始[2]。回想一下第 9 章中计算序列 y 的概率 p(y)的条件 RNN 语言模型。与任何语言模型一样，我们可以将概率分解如下:

$$\mathbf{p(y) = p(y_1)p(y_2 \mid y_1)p(y_3 \mid y_1, y_2)...P(y_m \mid y_1, ..., y_{m-1})} \tag{11.7}$$

在特定时间 t，我们通过语言模型传递 t-1 符记的前缀，使用前向推理产生一个隐藏状态序列，以前缀的最后一个单词对应的隐藏状态结束。然后使用前缀的最终隐藏状态作为生成下一个符记的起点。

更正式地讲，如果 g 是激活函数(如 tanh 或 ReLU)，其参数是时间 t 的输入和时间 t-1 的隐藏状态，并且 f 是一组可能的词汇项的 softmax，则在时间 t 的输出 y_t 和隐藏状态 h_t 计算为：

$$h_t = g(h_{t-1}, x_t) \tag{11.8}$$

$$y_t = f(h_t) \tag{11.9}$$

我们只需做一点改动就可以将这种语言模型(具有自动回归生成)转变为翻译模型(可以将一种语言的源文本翻译成另一种语言的目标文本)：在**源(source)**文本的末尾添加一个句子分隔标记，然后简单地拼接**目标(target)**文本。在第 9 章中，当我们考虑使用 Transformer 语言模型通过训练条件语言模型来进行摘要时，我们简要介绍了句子分隔符记的想法。如果我们调用(call)源文本 x 和目标文本 y，则按以下方式计算概率 p(y|x)：

$$\mathbf{p(y \mid x) = p(y_1 \mid x)p(y_2 \mid y_1, x)p(y_3 \mid y_1, y_2, x)...P(y_m \mid y_1, ..., y_{m-1}, x)} \tag{11.10}$$

图 11.4 显示了简化版本的编码器-解码器模型的设置(我们将在下一节看到完整的模型，其中需要**注意力(attention)**)。

图 11.4 显示了英语原文(the green witch arrived)，句子分隔符记(<s>)和西班牙语译文(llegó la bruja

[2] 稍后我们将看到如何使用成对的 Transformer;甚至可以为编码器和解码器使用不同的架构。

verde)。为了翻译源文本，我们通过网络进行正向推理以生成隐藏状态，直到到达源末尾为止。然后我们开始自回归生成，要求(asking for)隐含层上下文中的单词(从源输入的末尾到句子末尾的标记)。后继单词的计算条件是：先前的隐藏状态和最后一个生成单词的嵌入。

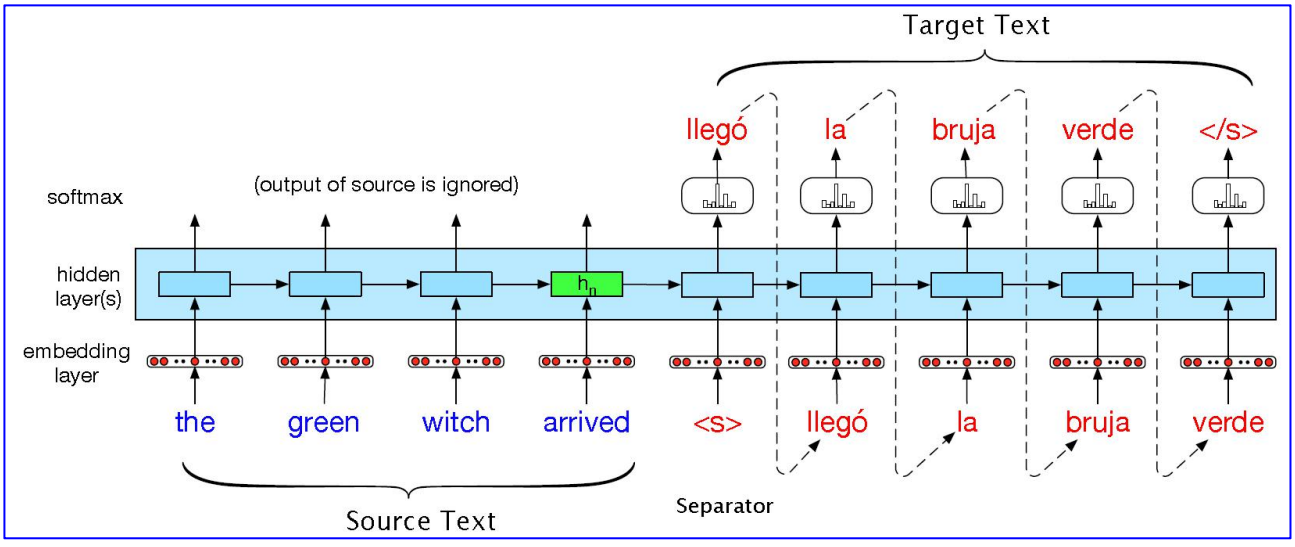


图 11-4：基本的 RNN 编码器-解码器的机器翻译

图注：在基本 RNN 版本的编码器-解码器方法中，翻译单个句子(推理时间)以进行机器翻译。源语句和目标语句之间用分隔符记连接，解码器使用编码器上次隐藏状态中的上下文信息。

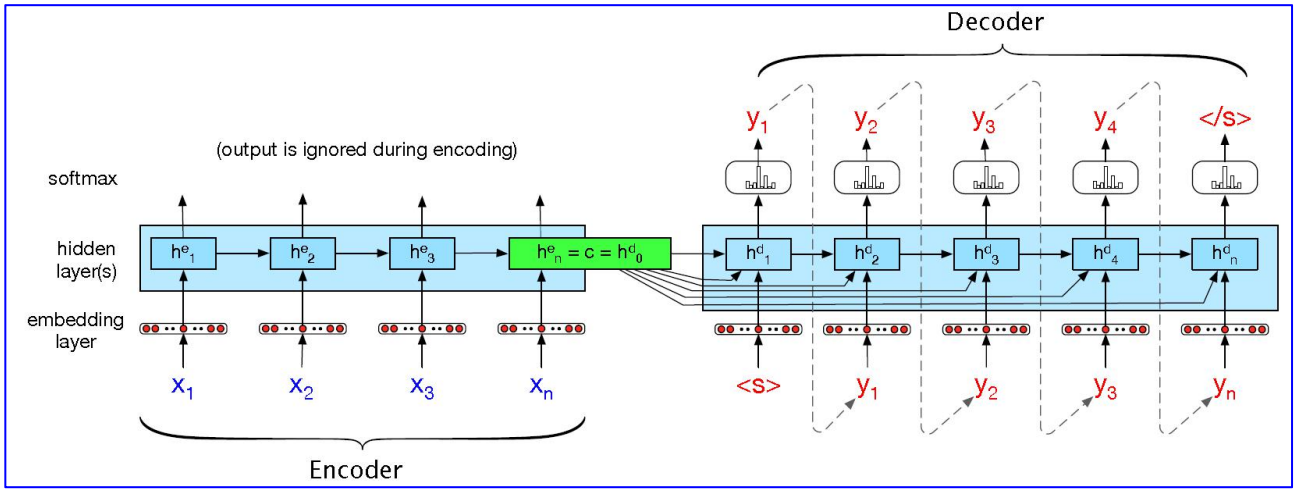


图 11-5：正式的 RNN 编解码器的机器翻译

图注：在基于 RNN 的基本编码器-解码器基础结构中，在推理时翻译句子的更正式版本。编码器 RNN 的最终隐藏状态 h^e_n 作为解码器的上下文，在解码器 RNN 中当作 h^d_0 角色。

让我们在图 11.5 中对此模型进行形式化和概括化。(为了使事情变得简单，我们将在需要时使用上标 e 和 d 来区分编码器和解码器的隐藏状态。)左侧的网络元素处理输入序列 x 并组成**编码器(encoder)**。虽然我们的简化图仅显示了编码器的单个网络层，但堆叠体系结构是标准的，其中将堆叠顶层的输出状态作为最终表示。广泛使用的编码器设计利用了堆叠的 biLSTM，其中，如第 9 章所述，将来自前向和后向传递的顶层隐藏状态进行了拼接，以提供每个时间步的上下文表示。

编码器的全部目的是生成输入的上下文表示。该表示体现在编码器 h^e_n 的最终隐藏状态中。然后将该表示形式，也称为**上下文(context)** c，传递给解码器。

右侧的**解码器(decoder)**网络采用此状态，并使用它来初始化解码器的第一个隐藏状态。即，第一解码器 RNN 单元使用 c 作为其先前的隐藏状态 h^d_0。解码器自动回归地生成一个输出序列，一次生成一个元素，直到生成序列结束标记。每个隐藏状态的计算基础是：先前的隐藏状态和在先前状态中生成的输出。

到目前为止描述的这种方法的一个缺点是，上下文向量 c 的影响将随着输出序列的生成而减弱。

一种解决方案是将上下文向量 c 作为参数添加到当前隐藏状态的计算中，使其在解码过程的每一步都可用，使用如下公式(如图 11.6 所示):

$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c) \tag{11.11}$$

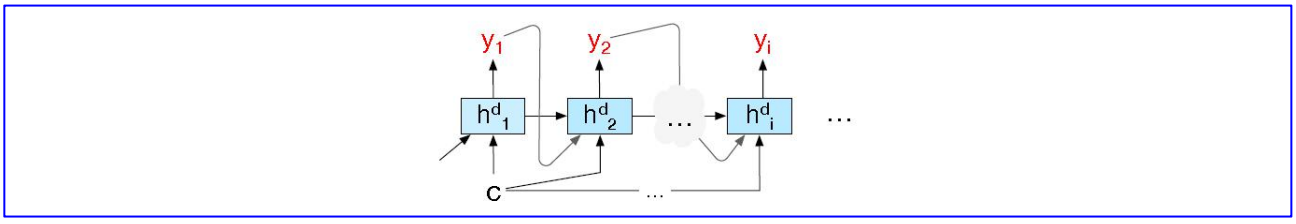


图 11-6：解码器的隐藏状态

图注：允许解码器的每个隐藏状态(不仅是第一个解码器状态)受编码器产生的上下文 c 的影响。

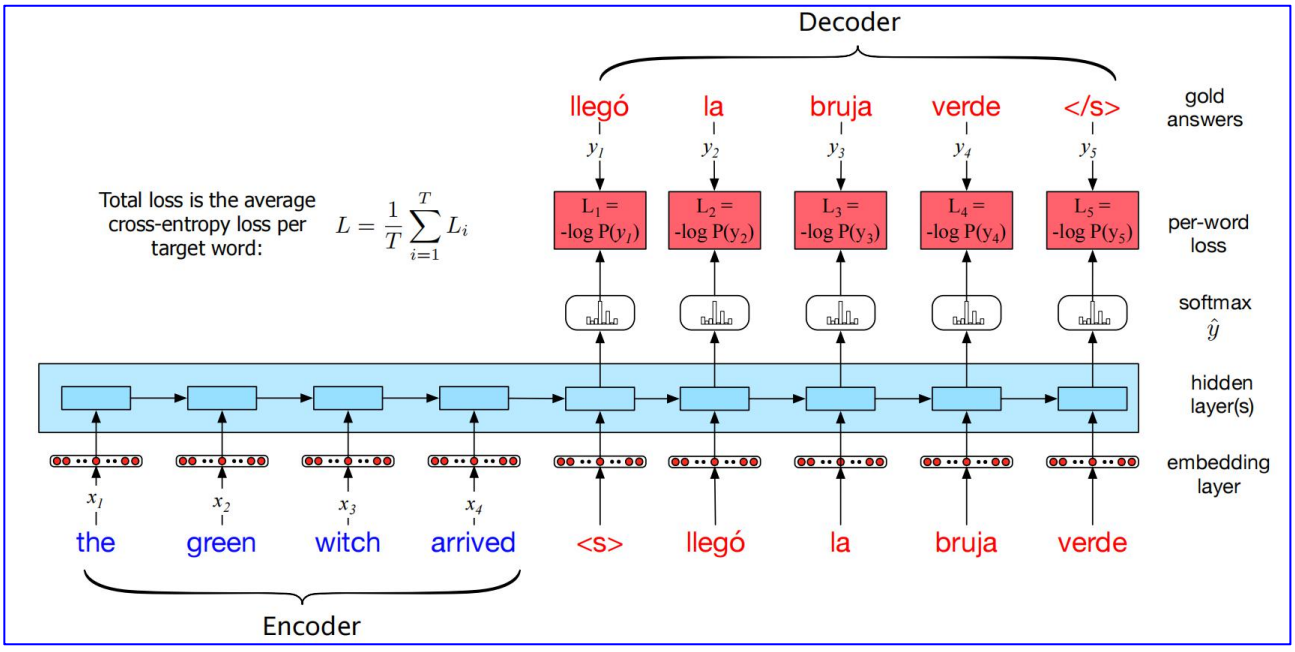


图 11-7：训练的 RNN 编解码器的机器翻译

图注：训练基本的 RNN 编码器-解码器机器翻译方法。请注意，在解码器中，我们通常不会传递模型的 softmax 输出$\hat{y}_t$，而是使用教师强制将每个输入强制为正确的训练黄金值。我们在解码器中通过$\hat{y}$计算 softmax 输出分布，以计算每个符记的损失，然后可以对其求平均值来计算句子的损失。

现在，我们已准备好在基本的编解码器模型中查看此版本解码器的完整方程式，并在每个解码时间步都有可用的上下文。回顾，g 是 RNN 的某种替代品，而$\hat{y}_{t-1}$是在上一步中从 softmax 采样的输出的嵌入：

$$\begin{aligned} c &= h_n^e \\ h_0^d &= c \\ h_t^d &= g(\hat{y}_{t-1}, h_{t-1}^d, c) \\ z_t &= f(h_t^d) \\ y_t &= \text{softmax}(z_t) \end{aligned} \tag{11.12}$$

最后，如前所述，每个时间步的输出 y 包括一组可能的输出(在语言建模或 MT 情况下的词汇)上的 softmax 计算。我们通过在 softmax 输出的基础上取 argmax，在每个时间步计算最可能的输出：

$$\hat{y}_t = \text{argmax}_{w \in V} P(w \mid x, y_1 \ldots y_{t-1}) \tag{11.13}$$

还有很多方法可以让模型更强大。例如，我们可以通过调节输出层 y 来帮助模型跟踪哪些已经生成，哪些还没有生成，方法不仅是对隐藏状态h^d_t和上下文 c 进行调节，还可以对在前一个时间步生成的输出$\hat{y}_{t-1}$进行调节:

$$y_t = \text{softmax}(\hat{y}_{t-1}, z_t, c) \tag{11.14}$$

11.3.1. 训练编码器-解码器模型

编码器-解码器架构是端到端的训练，就像第 9 章的 RNN 语言模型一样。每个训练示例都是一对字符串组成的元组，一个源和一个目标。通过连接分隔符记，这些源-目标偶对现在可以作为训练数据。

对于 MT，训练数据通常由一组句子及其翻译组成。这些可以从对齐句子偶对的标准数据集中得出，我们将在第 11.7.2 节中讨论。有了训练集后，训练本身便会像使用任何基于 RNN 的语言模型一样进行。向网络提供源文本，然后从分隔符记开始进行自回归训练以预测下一个单词，如图 11.7 所示。

请注意，在每个时间步上，训练(图 11.7)和推理(图 11.4)之间的差异。推理期间的解码器将其自己的估计输出$\hat{y}_t$作为下一个时间步 x_{t+1} 的输入。因此，随着解码器不断产生更多的符记，它将倾向于越来越多地偏离黄金目标语句。因此，在训练中，更常见的是在解码器中使用**教师强制(teacher forcing)**。教师强制意味着我们迫使系统使用训练中的黄金目标符记作为下一个输入 x_{t+1}，而不是让它依赖于(可能是错误的)解码器输出$\hat{y}_t$。这样可以加快训练速度。

11.4. 注意力

编码器-解码器模型的简单性在于它将编码器(构建源文本的表示)与解码器(使用此上下文生成目标文本)清晰地分离开来。在我们目前描述的模型中，这个上下文向量是 h_n，源文本最后(第 n 个)时间步的隐藏状态。因此，这个最终的隐藏状态就像一个**瓶颈(bottleneck)**(图 11.8)：它必须表示源文本的所有含义，因为解码器对源文本的唯一了解就是这个上下文向量中的内容。句子开头的信息，特别是长句子的信息，可能不能在上下文向量中得到同样好的表示。

注意力机制是解决瓶颈问题的一种方法，它允许解码器从编码器的所有隐藏状态(而不仅仅是最后一个隐藏状态)中获取信息。

在注意力机制中，就像在普通编码器-解码器模型中一样，上下文向量 c 是单个向量，它是编码器隐藏状态的函数，即 $c= f(h_1^n)$。由于隐藏状态的数量随输入大小的变化而变化，因此我们无法将编码器隐藏状态向量的整个张量直接用作解码器的上下文。注意力的想法是通过对所有编码器隐藏状态h_1^n进行加权求和来创建单个固定长度矢量 c。

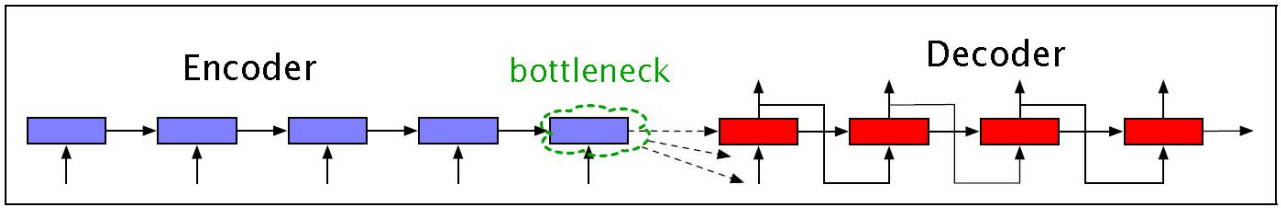


图 11-8：编码器的隐藏状态-瓶颈

图注：要求上下文 c 仅为编码器的最终隐藏状态，迫使来自整个源句的所有信息都要通过这个表示的瓶颈。

权重用于关注源文本的特定部分，该部分与当前由解码器生成的符记相关。因此，由注意力机制产生的上下文向量是动态的，对于解码中的每个符记来说都是不同的。

因此，注意力将静态上下文向量替换为动态上下文，这个动态上下文是解码期间从编码器隐藏状态在每个点上动态地导出的向量。每个解码步骤 i 都会重新生成这个上下文向量 c_i，并在推导过程中考虑到编码器的所有隐藏状态。然后，我们通过对当前解码器隐藏状态的计算(以及先前的隐藏状态和解码器生成的先前输出)进行调整，使此上下文在解码期间可用。该方程为(见图 11.9)：

$$\boldsymbol{h}_i^d = g(\hat{y}_{i-1}, \boldsymbol{h}_{i-1}^d, \boldsymbol{c}_i)$$

计算 c_i 的第一步是计算每个编码器状态的聚焦度，以及每个编码器状态与 h^d_{i-1} 中捕获的解码器状态的相关度。我们通过在解码过程中的每个状态 i 上计算每个编码器状态 j 的得分(h^d_{i-1}，h^e_j)来捕获相关度。

这种最简单的分数称为**点积注意力(dot-product attention)**，将相关度实现为相似度：通过计算解码器隐藏状态与编码器隐藏状态之间的点积，来测量它们之间的相似度：

$$\mathit{SCORE}(\boldsymbol{h}_{i-1}^d, \boldsymbol{h}_j^e) = \boldsymbol{h}_{i-1}^d \cdot \boldsymbol{h}_j^e \tag{11.15}$$

从这个点积得到的分数是一个标量，反映两个向量之间的相似度。所有编码器隐藏状态的这些分数的

向量为我们提供了每个编码器状态与解码器当前步骤的相关度。

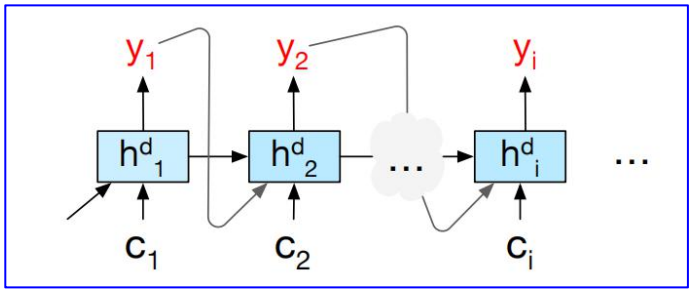


图 11-9：注意力机制的动态上下文

图注：注意力机制允许解码器的每个隐藏状态看到不同的动态上下文，这是所有编码器隐藏状态的函数。

为了利用这些得分，我们将使用 softmax 对其进行归一化，以创建权重向量α_{ij}，该权重告诉我们每个编码器隐藏状态 j 与先前的隐藏解码器状态 h^d_{i-1} 的比例相关度。

$$\alpha_{ij} = \text{softmax}(SCORE(\boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{i-1}}, \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{j}}),\ \forall j \in e) \tag{11.16}$$

$$= \frac{exp(\boldsymbol{SCORE}(\boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{i-1}}, \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{j}}))}{\sum_k exp(\boldsymbol{SCORE}(\boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{i-1}}, \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{k}}))} \tag{11.17}$$

最后，给定α中的分布，我们可以通过对所有编码器隐藏状态进行加权平均来计算当前解码器状态的固定长度上下文向量。

$$\mathbf{c}_i = \sum_j \alpha_{ij} \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{j}} \tag{11.18}$$

这样，我们最终得到了一个固定长度的上下文向量，它考虑了来自整个编码器状态的信息，并在解码的每一步动态更新以反映解码器的需求。图 11.10 说明了一个编码器-解码器网络，着重于一个上下文向量 $\mathbf{c}_i$ 的计算。

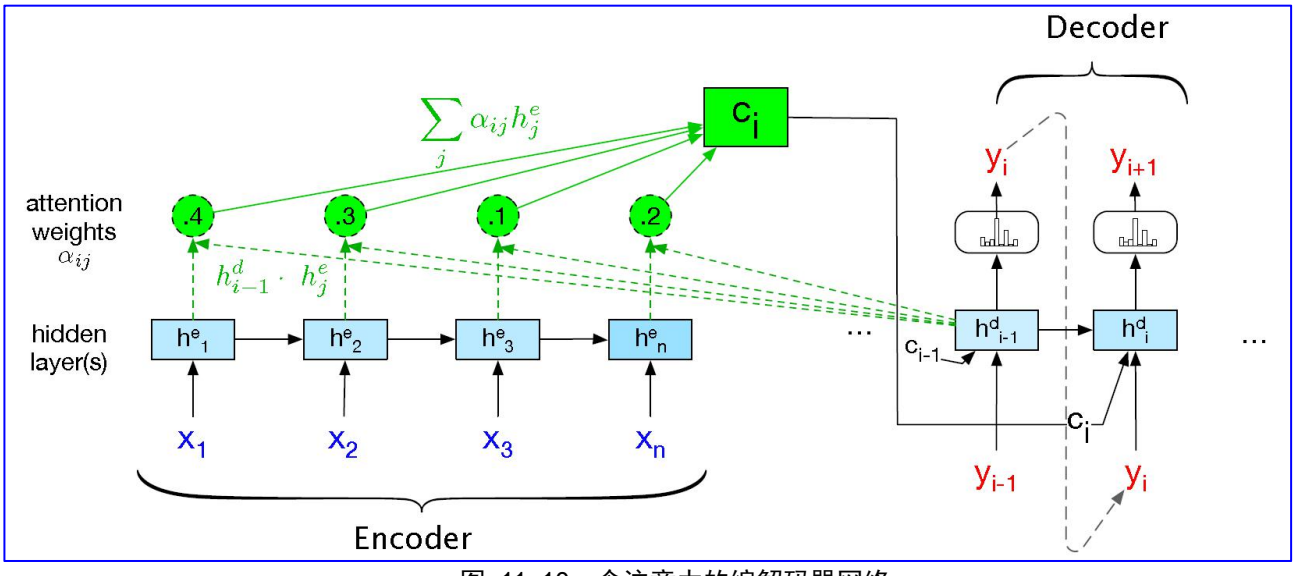


图 11-10：含注意力的编解码器网络

图注：图中重点是 c_i 的计算。上下文值 c_i 是计算 h^d_i 的输入之一。它是通过对所有编码器隐藏状态的加权和来计算的，每个隐藏状态的加权值都是它们与之前的解码器隐藏状态 h^d_{i-1} 的点积。

还可为注意力模型创建更为复杂的评分函数。不用简单的点积注意力，我们可得到一个更强大的函数，该函数通过参数化得分与它自己的一组权值 W_s，计算每个编码器隐藏状态与解码器隐藏状态的相关度：

$$\boldsymbol{SCORE}(\boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{i-1}}, \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{j}}) = \boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{i-1}} \boldsymbol{W}_{\boldsymbol{S}} \boldsymbol{h}^{\boldsymbol{e}}_{\boldsymbol{j}}$$

然后在正常的端到端训练过程中训练的权值 W_s，使网络能够学习解码器和编码器状态之间的哪些相似点对当前应用是重要的。这种双线性模型还允许编码器和解码器使用不同的维数向量，而简单的点积注意力则要求编码器和解码器隐藏状态具有相同的维数。

11.5. 波束搜索

我们上面给出的用于生成翻译的解码算法有一个问题(正如我们在第 9 章中介绍的从条件语言模型生成的自回归生成)。回想一下那个算法:在解码的每个时间步，通过计算一组可能的输出(在语言建模或 MT 的情况下是词汇表)的 softmax 来选择输出 y_t，然后选择概率最高的符记(argmax):

$$\hat{y}_t = \text{argmax}_{w \in V} P(w|x, y_1 \ldots y_{t-1}) \quad (11.19)$$

在每一步中选择一个最可能产生的符记称为**贪心(greedy)**解码;贪心算法是一种做出局部最优选择的算法，不管它事后是否会被证明是最好的选择。

事实上，贪心搜索并不是最优的，而且可能找不到最高的平移概率。问题是，现在对解码器来说看起来不错的符记，后来可能证明是错误的选择!

让我们通过查看搜索树来了解这一点，**搜索树(search tree)**是解码器在搜索最佳翻译时所做选择的图形表示。在搜索树中，我们将解码问题视为启发式状态空间搜索，并系统地探索可能输出的空间。在这样的搜索树中：分支是操作，即生成符记的操作；节点是状态，即生成特定前缀的状态。我们正在寻找最佳的动作序列，也就是概率最高的目标字符串。图 11.11 用一个虚构的例子演示了这个问题。注意，最可能的序列是 ok ok(概率为.4*.7*1.0)，但贪心的搜索算法将无法找到它，因为它错误地选择 yes 作为第一个单词，因为它具有最高的局部概率。

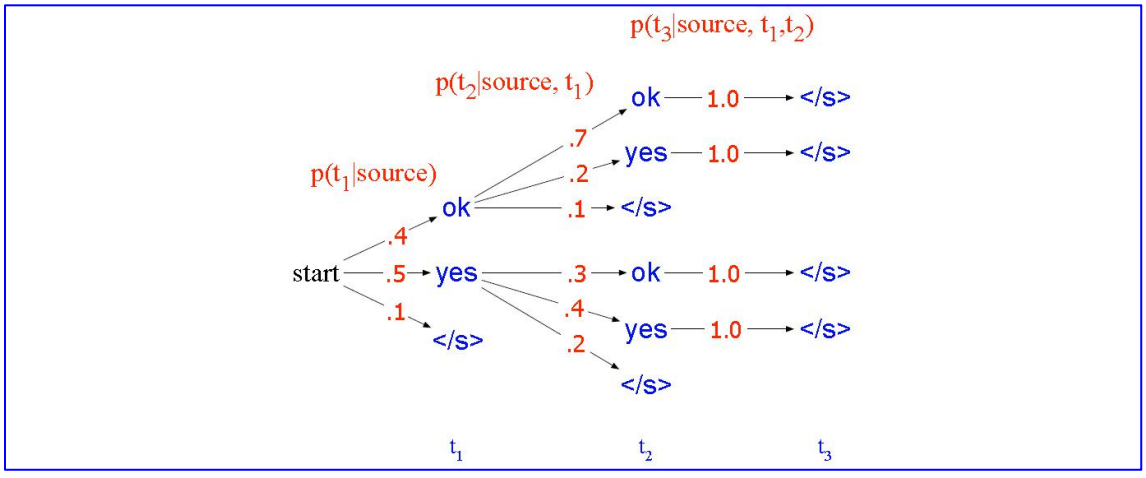


图 11-11：搜索树

图注：在给定源字符串的情况下，根据词汇 V = {yes，ok，</s>}生成目标字符串 T = t_1，t_2，...的搜索树，显示了从该状态生成每个符记的概率。贪心搜索会在第一步中选择 yes，然后再选择 yes 而不是全局最可能的顺序。

回想一下在第 8 章中，对于词类标记，我们使用了动态规划搜索(Viterbi 算法)来解决这个问题。不幸的是，动态规划不适用于输出决策之间存在长距离依存关系的生成问题。唯一能保证找到最佳解的方法是穷举搜索:计算每一个 V^T 可能的句子(对于某个长度值 T)的概率，这显然太慢了。

相反，在 MT 解码和其他序列生成问题中，通常使用一种称为**波束搜索(beam search)**的方法。在波束搜索中，我们在每一步都保留 k 个可能的符记，而不是在每个时间步都选择最优的符记来生成。这种固定大小的记忆足迹(memory footprint)k 称为**波束宽度(beam width)**，就像手电筒光束的隐喻一样，可以将其参数化为更宽或更窄。

因此，在解码的第一步，我们计算整个词汇表的 softmax，为每个单词分配一个概率。然后我们从这个 softmax 输出中选择 k 个最佳选项。这些初始输出的 k 是搜索边界，这 k 个初始词被称为假设。这个假设就是一个输出序列，也就是即时翻译及其概率。

在随后的步骤中，k 个最佳假设的每一个都被传递给不同的解码器，每个解码器在整个词汇表上生成一个 softmax，以将假设扩展到每个可能的下一个符记。这 k* V 个假设中的每一个都由 $P(y_i|x, y_{<i})$评分：当前单词选择的概率乘以导致该单词的路径的概率。然后，我们将 k*V 个假设删减为 k 个最佳假设，因此在搜索的边界上永远不会超过 k 个假设，并且也不会超过 k 个解码器。

图 11.12 以束宽为 2 说明了这一过程。

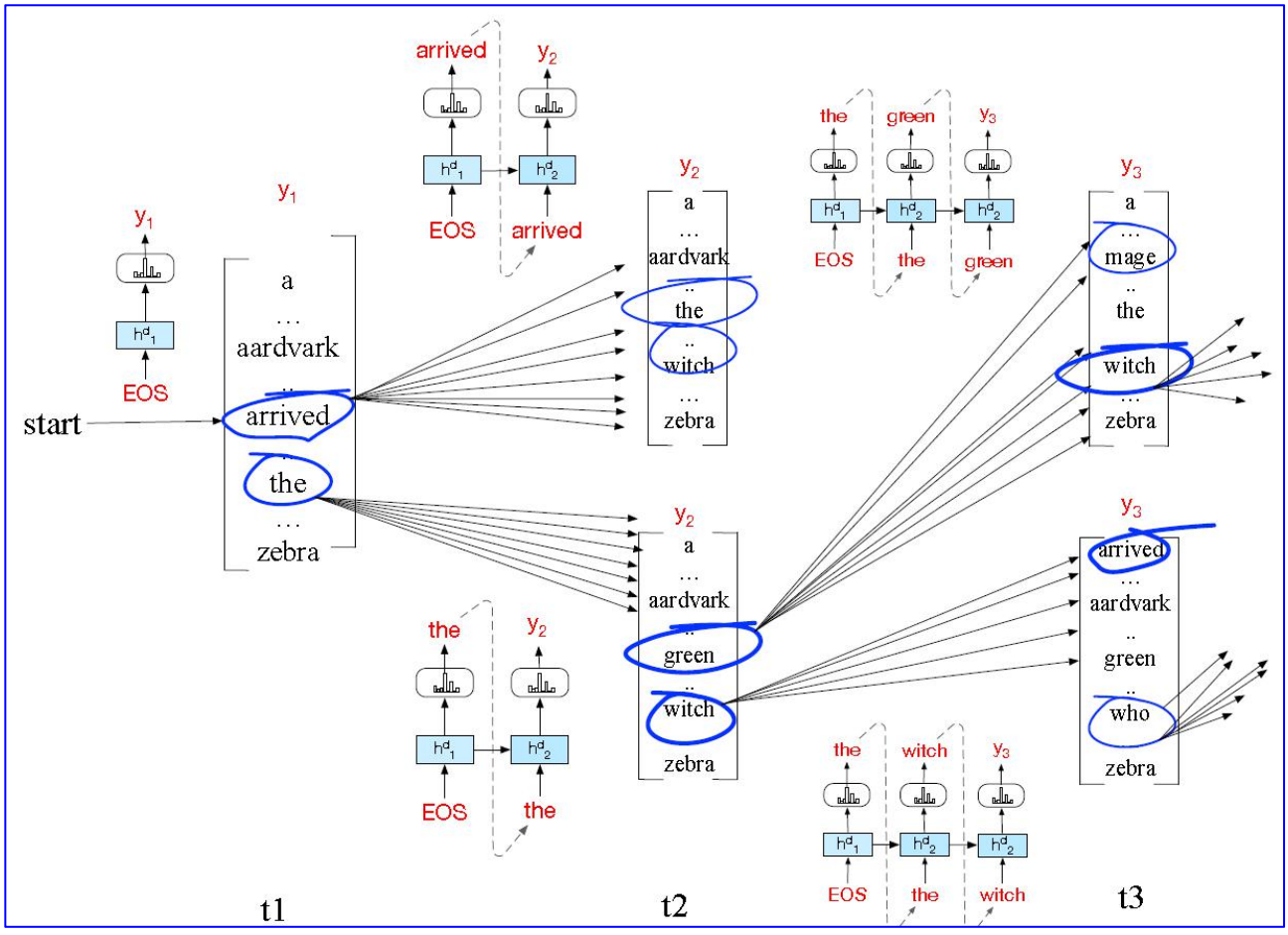


图 11-12:波束搜索解码的原理

图注：波束宽度 k=2 时的波束搜索解码。在每个时间步，我们选择 k 个最好的假设，计算每个假设的 V 个可能的扩展，给得到的 k*V 个可能的假设评分，并选择最好的 k，继续。在时间 t1，边界被解码器初始状态的 2 个最佳选项填满:arrived 和 the。然后我们扩展每一个假设，计算到目前为止所有假设的概率(arrived the, arrived aardvark, the green, the witch)并计算最好的 2 个(在本例中是 the green 和 the witch)作为下一步要扩展的搜索边界。在弧线上，我们展示了解码器，这些解码器对扩展词进行评分(尽管为了简单起见，我们没有显示在每一步输入的上下文值 c_i)。

这个过程会一直持续，直到生成一个</s>，这表明已经找到了一个完整的候选输出。在这一点上，完整的假设从边界移除，波束的尺寸减小了 1。搜索将继续进行，直到波束减少到 0，结果是 k 个假设。

让我们看看详细的评分是如何工作的，根据每个节点的对数概率来评分。回想一下公式 11.10，我们可以使用概率链式规则将 p(y|x)分解为每个单词在其之前上下文下的概率的乘积，我们可以将其转换为对数的总和(对于长度为 t 的输出字符串):

$$\begin{aligned} score(y) &= \log P(y|x) \\ &= \log(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)...P(y_t|y_1,...,y_{t-1},x)) \\ &= \sum_{i=1}^{t} \log P(y_i|y_1,...,y_{i-1},x) \end{aligned} \tag{11.20}$$

因此，在每一步中，为了计算部分转换的概率，我们只需将迄今为止前缀转换的对数概率与生成下一个符记的对数概率相加。图 11.13 用一些简单编造的概率给出了图 11.12 中的例句的得分。对数概率是负的或 0，两个对数概率的最大值是较大的那个(接近于 0)。

图 11.14 给出了算法。一个问题产生于这样一个事实，即完整的假设可能有不同的长度。因为模型通常为较长的字符串分配较低的概率，朴素的算法也会为 y 选择较短的字符串。这在早期的解码步骤中不是问题;由于波束搜索的宽度优先性，所有被比较的假设都具有相同的长度。通常的解决方法是对每个假设应用某种形式的长度归一化，例如简单地将负对数概率除以字数:

$$\begin{aligned} score(y) &= -\log P(y|x) \\ &= \frac{1}{t}\sum_{i=1}^{t} -\log P(y_i|y_1,...,y_{i-1},x) \end{aligned} \tag{11.21}$$

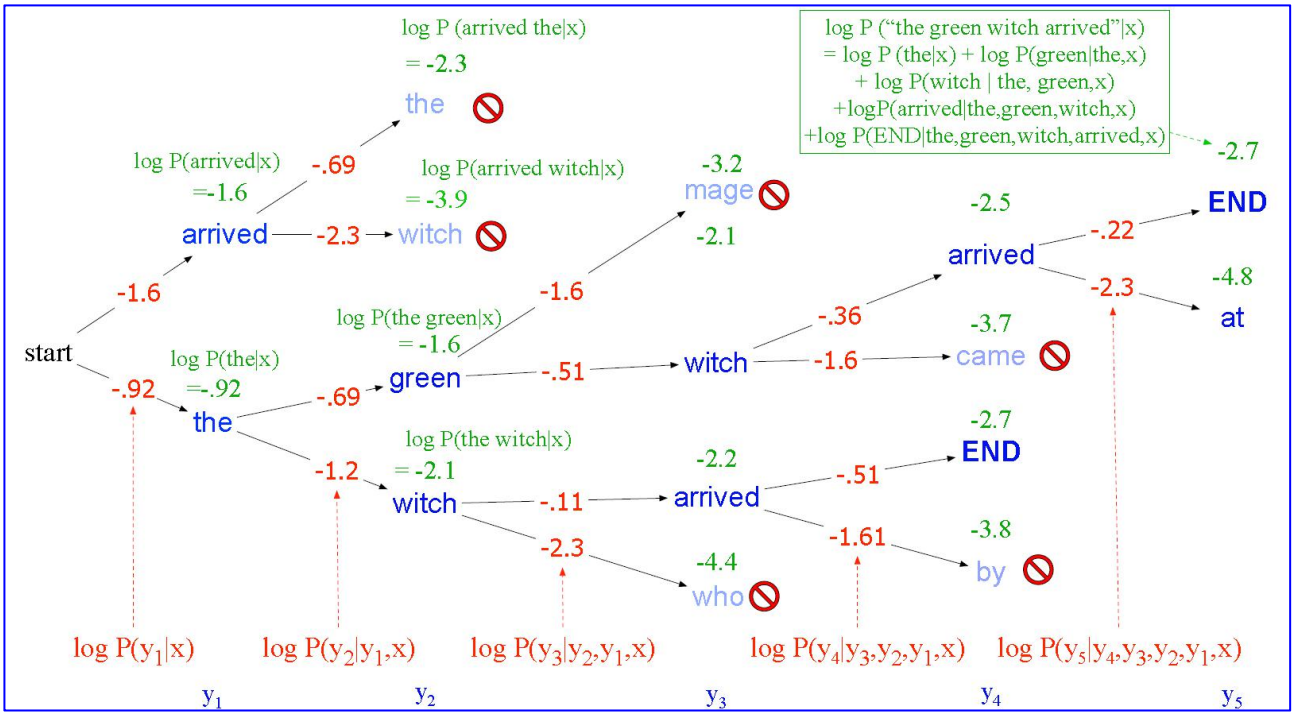


图 11-13：波束搜索解码评分

图注：波束宽度 k = 2 时的波束搜索解码评分。我们通过增加生成下一个字符的对数概率来保持波束中每个假设的对数概率。只有最上面的 k 条路径被扩展到下一步。

波束搜索在大型 MT 系统的产品中很常见，通常波束宽度 k 在 5 到 10 之间。我们怎么处理得出的 k 个假设?在某些情况下，我们只需要 MT 算法中最好的假设，所以我们可以取返回值。在其他情况下，我们的下游应用程序可能需要查看所有 k 个假设，因此我们可以将它们全部(或一个子集)和它们各自的分数传递给下游应用程序。

11.6. 编解码器与 Transformers

TBD(待定)

11.7. 构建 MT 系统的实用细节

11.7.1. 符记化

机器翻译系统通常使用一个固定的词汇表，生成这个词汇表的常用方法是使用第 2 章中概述的 BPE 或词片(wordpiece)算法。通常，源语言和目标语言使用共享词汇表，这使得将符记(如名称)从源语言复制到目标语言非常容易，因此我们在包含源语言和目标语言数据的语料库上构建词片/BPE 词汇表。词片在每个符记的开头使用一个特殊的符号;以下是谷歌 MT 系统的符记化结果(Wu 等人，2016):

words: Jet makers feud over seat width with big orders at stake

wordpieces: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

第二章详细介绍了 BPE 算法;以下是关于词片算法的更多细节，它有一个训练语料库和一个期望的词汇大小 V，并按照以下步骤进行:

1. 用字符(例如 Unicode 字符的子集，将所有剩余字符折叠为一个特殊的未知字符符记)初始化 wordpiece 词典。

2. 重复，直到有 V 个词片:

(a)在训练语料库上训练 n-gram 语言模型，使用当前的一组词片。

(b)考虑将当前词汇中的两个词连接在一起而产生的可能的新词片。选择最能增加训练语料库语言模型概率的新词片。

通常使用 8K 到 32K 的词汇量。

```
function BEAMDECODE(c, beam width) returns best paths
 y0, h0←0
 path←()
 complete paths←()
 state←(c, y0, h0, path)    ;initial state
 frontier← ⟨state⟩          ;initial frontier

 while frontier contains incomplete paths and beamwidth > 0
      extended_frontier ⟨⟩
      for each state ∈ frontier do
           y←DECODE(state)
           for each word i ∈ Vocabulary do
                successor←NEWSTATE(state, i, yi)
                new_agenda←ADDTOBEAM(successor, extended_frontier, beam_width)

      for each state in extended frontier do
           if state is complete do
                complete_paths←APPEND(complete_paths, state)
                extended_frontier←REMOVE(extended_frontier, state)
                beam_width←beam_width - 1
      frontier←extended_frontier

 return completed_paths

 function NEWSTATE(state, word, word_prob) returns new state

 function ADDTOBEAM(state, frontier, width) returns updated frontier
  if LENGTH(frontier) < width then
      frontier←INSERT(state, frontier)
  else if SCORE(state) > SCORE(WORSTOF(frontier))
      frontier←REMOVE(WORSTOF(frontier))
      frontier←INSERT(state, frontier)
  return frontier
```

图 11-14：波束搜索解码算法伪代码

11.7.2. MT 语料库

机器翻译模型是在**平行语料库(parallel corpus，bitext)**上进行训练的，这种文本以两种(或更多)语言出现。平行语料库数量庞大。有些是政府的语料库;欧洲语料库(Koehn, Europarl 2005)从欧洲议会的会议记录中提取，包含来自 21 种欧洲语言的 40 万至 200 万个 e 句子。联合国平行语料库包含联合国六种官方语言(阿拉伯语、汉语、英语、法语、俄语，西班牙语)约 1000 万个句子。其他平行语料库来自电影和电视字幕，如**开放字幕(OpenSubtitles)**语料库(Lison 和 Tiedemann, 2016)，或一般的 web 文本，如 **ParaCrawl** 语料库，包含 23 种欧盟语言之间的 2.23 亿句对，以及从 CommonCrawl Banon 等人(2020)中提取的英语。

句子对齐

机器翻译的标准训练语料库是成对对齐的句子。当创建新的语料库时，例如对于资源不足的语言或新的领域，必须创建这些句子对齐。图 11.15 给出了一个假设的句子对齐示例。

给定两个相互翻译的文档，我们通常需要两个步骤来生成句子对齐:

- 一个成本函数，获取一段源句和一段目标句，返回一个分数，衡量这些句子被翻译的可能性有多大。
- 一个对齐算法，利用这些分数来找到文档之间的良好对齐。

由于可以诱导多语言句子嵌入(Artetxe 和 Schwenk, 2019)，这种嵌入的余弦相似度提供了一个自然的评分函数(Schwenk, 2018)。Thompson 和 Koehn(2019)分别从源文档和目标文档给出了以下两个句子或跨 x,y 之间的成本函数:

$$c(x,y)=\frac{(1-cos(x,y))nSents(x)nSents(y)}{\sum_{s=1}^{S}(1-cos(x,y_s))+\sum_{s=1}^{S}(1-cos(x_s,y))} \tag{11.22}$$

其中 nSents()给出了句子的数量(这使度量偏向于单个句子的许多对齐，而不是对齐非常大的跨度)。分母有助于归一化相似度，所以 $x_1,...,x_S$, $y_1,...,y_S$，是从相应的文档中随机抽取的句子。

通常使用动态规划作为对齐算法(Gale 和 Church, 1993)，这是对我们在第 2 章中介绍的最小编辑距离算法的简单扩展。

最后，通过去除有噪声的句对，对语料库进行清理。这可能涉及到删除低精度句对的手写规则(例如，删除太长、太短、有不同 URL 的句子，甚至是太相似(表明它们是拷贝而不是翻译)的句对)。或者这些句对可以通过多语言嵌入余弦值对其进行排序，而低评分对被丢弃。

E1: “Good morning," said the little prince.	F1: -Bonjour, dit le petit prince.
E2: “Good morning," said the merchant. E3: This was a merchant who sold pills that had been perfected to quench thirst.	F2: -Bonjour, dit le marchand de pilules perfectionnées quiapaisent la soif.
E4: You just swallow one pill a week and you won’t feel the need for anything to drink.	F3: On en avale une par semaine et l'on n'éprouve plus lebesoin de boire.
E5: “They save a huge amount of time," said the merchant.	F4: -C’est une grosse économie de temps, dit le marchand.
	F5: Les experts ont fait des calculs.
E6: “Fifty–three minutes a week."	F6: On épargne cinquante-trois minutes par semaine.
E7: “If I had fifty–three minutes to spend?" said the little prince to himself.	F7: “Moi, se dit le petit prince, si j'avais cinquante-trois minutes à dépenser, je marcherais tout doucement vers une fontaine..."
E8: “I would take a stroll to a spring of fresh water”	

图 11-15：英文和法文句子之间的对齐示例

图注：其中的句子摘自圣艾修伯里的《小王子》和一个假设的翻译。句子对齐采用句子 $e_1...e_n$ 和 $f_1...f_n$ 并找到相互翻译的最小句子集，包括 (e_1, f_1), (e_4-f_3), (e_5-f_4), (e_6-f_6) 等单个句子映射，以及 2-1 对齐 $(e_2/e_3, f_2)$，$(e_7/e_8, f_7)$ 和空对齐 (f_5)。

11.7.3. 反向翻译

由于平行语料库对于特定的语言或领域可能是有限的，我们经常缺乏训练机器翻译模型的数据。

然而，我们经常可以找到一个大的单语语料库，以增加较小的平行语料库可用。

反向翻译(backtranslation)是利用目标语言中的单语语料库，创建合成的平行语料库(bittext)的一种方法。在反向翻译中，我们在小 bitext 上训练一个中间目标-源的 MT 系统，将单语目标数据翻译成源语言。现在我们可以将这个合成 bitext(自然目标句子，与 MT 生成的源句子对齐)添加到我们的训练数据中，并重新训练我们的源-目标 MT 模型。例如，假设我们想把纳瓦霍语翻译成英语，但只有一小部分纳瓦霍语-英语文本，尽管我们当然可以找到大量的单语英语数据。我们使用小的 bitext 来建立一个 MT 引擎以另一种方式(从英语到纳瓦霍语)运行。一旦我们把单语英语文本翻译成纳瓦霍语，我们就可以把这个合成的纳瓦霍/英语文本添加到我们的训练数据中。

反向翻译有各种参数。一个参数是我们如何生成反向翻译数据;我们可以在贪心推理中运行解码器，也可以使用波束搜索。或者我们可以做抽样，或者**蒙特卡罗搜索(Monte Carlo search)**。在蒙特卡罗解码中，在每个时间步，我们并不总是生成 softmax 概率最大的单词，而是掷一个加权的骰子，根据它的 softmax 概率选择下一个单词。这就像我们在第 3 章中看到的从 n-gram 语言模型中生成随机句子的抽样算法一样。假设只有 4 个单词，t 时刻的 softmax 概率分布为(the: 0.6, green: 0.2, a: 0.1, witch: 0.1)。我们掷一个加权的骰子，4 个边的权重分别为 0.6、0.2、0.1 和 0.1，然后根据出现的边选择单词。另一个参数是反向翻译数据与自然 bitext 数据的比率;我们可以选择对 bitext 数据进行上采样(包括每个句子的多个副本)。

总的来说，反向翻译的效果出奇的好;一项估计表明，在反向翻译文本上训练的系统获得的收益约为在同等数量的自然文本上训练的 2/3(Edunov 等人，2018)。

11.8. 机器翻译评价

翻译可以从两个维度来评估，充分性和流利性。

充分性(adequacy):译文对源句的准确把握程度。有时称为忠诚或忠实。

流利性(fluency):翻译的目标语言的流利程度(是否符合语法、清晰、易读、自然)。

同时使用人工和自动评估指标。

11.8.1. 用人工评分员对机器翻译进行评价

最准确的评估是使用人工评分员在两个维度上评估每一种翻译(通常这些评分员是专门雇佣来评估的在线众包工作者)。

例如，在流利性维度上，我们可以询问机器翻译输出(目标文本)的可理解性、清晰性、可读性或自然性。我们可以给评分员一个等级，例如，从 1(完全无法理解)到 5(完全理解），或 1 到 100，并要求他们对 MT 输出的每个句子或段落进行评分。

我们可以做同样的事情来判断第二个维度，充分性，让评分者在一个量表上打分。如果我们有双语评分员，我们可以给他们源句和一个建议的目标句，并以 5 分或 100 分的评分标准，给目标句保留了多少源句中的信息。如果我们只有单语评分者，但我们对源文本有很好的人工翻译，我们可以给单语评分者人工参考翻译和目标机器翻译，并再次评估保留了多少信息。如果我们使用一个足够细粒度的量表，我们可以通过从评分中减去平均值并除以方差来标准化评分者。

另一种方法是进行排名:给评价者一对候选翻译，并询问他们更喜欢哪一个。虽然人工生成机器翻译输出的最佳评估，但运行人工评估可能耗时且昂贵。在下一节中，我们将介绍一种自动度量，尽管不如人类评估准确，但它被广泛使用，因为它可以快速评估潜在的系统改进，甚至可以用作训练的自动损失函数。

11.8.2. 自动评估：BLEU

最流行的机器翻译自动度量被称为**双语评估替补(BiLingual Evaluation Understudy，BLEU)**。BLEU(连同许多替代指标，例如：NIST、TER、精度和召回率、METEOR)是基于一个简单的直觉，源于 Miller 和 Beebe-Center(1958)的开创性工作：一个好的机器翻译往往会包含在同一句子的人工翻译中出现的单词和短语。

考虑一个来自平行语料库的测试集，其中每个源句子都有一个黄金的人类目标翻译和一个我们想要评估的候选 MT 翻译。BLEU 度量根据与人工翻译重叠的 n-gram 数量的函数对每个 MT 目标句进行排序。

图 11.16 显示了一个西班牙语源句子的两个候选翻译的直观感受，其中包含人工参考翻译。请注意，与候选 2 相比，候选 1 与参考翻译共享更多的 n-grams(在框中)，特别是更长的 n-grams(在深色框中)。

图 11.16 的直觉显示了一个句子，但 BLEU 实际上不是一个句子的得分，而是对整个候选翻译句子的评分。更正式地说，候选翻译句子的语料库的 BLEU 分数是所有句子的 n-gram 精度和整个语料库计算的简短惩罚的函数。

n-grams 精度是什么意思?考虑一个由单个句子组成的语料库。这个语料库的 unigram 精度是候选翻译中出现在参考翻译中的 unigram 符记的百分比，对于 bigrams 也是如此，最高可达 4-grams。图 11.16 中：候选 1 句子中有 19 个唯一的 unigrams（其中一些 unigrams 出现多次），总共有 26 个符记；参考译文中有 16 个唯一的 unigrams，总共 23 个符记(没有出现的有 3 个：voice，deposit，和 actions)。因此候选 1 语料库的 unigram 精度为 23/26 =0.88。

我们将这种 unigram 度量扩展到由许多句子组成的完整语料库，如下所示。对于分子，我们将参考译文中出现的所有 unigram 类型对每个句子的计数相加，然后将所有句子的计数相加。分母是所有候选句子中所有 unigrams 的总数。我们计算 unigrams、bigrams、trigrams 和 4-grams 的 n-grams 精度。因此，整个候选句语料库的 n-gram 精度 prec_n 为:

$$\text{prec}_n = \frac{\sum_{C\in\{Candidates\}}\sum_{n\text{-}gram\in C}\text{Count}_{\text{match}}(n\text{-}gram)}{\sum_{C'\in\{Candidates\}}\sum_{n\text{-}gram'\in C'}\text{Count}(n\text{-}gram')} \tag{11.23}$$

BLEU 通过取这四种 n-gram 精度的几何平均值来组合它们。

此外，过于简短的翻译也会受到 BLEU 的处罚。想象一下，我们的机器翻译引擎返回了图 11.16 中的例子中可怕的候选翻译 3:

(11.24) *for the*

因为 unigram 的 for 和 the，bigram 的 all the 都出现在人工参考中，仅 n-gram 精度就会给 candidate 3 一个很高的分数，因为它的 unigram 和 bigram 精度都达到了 1.0!

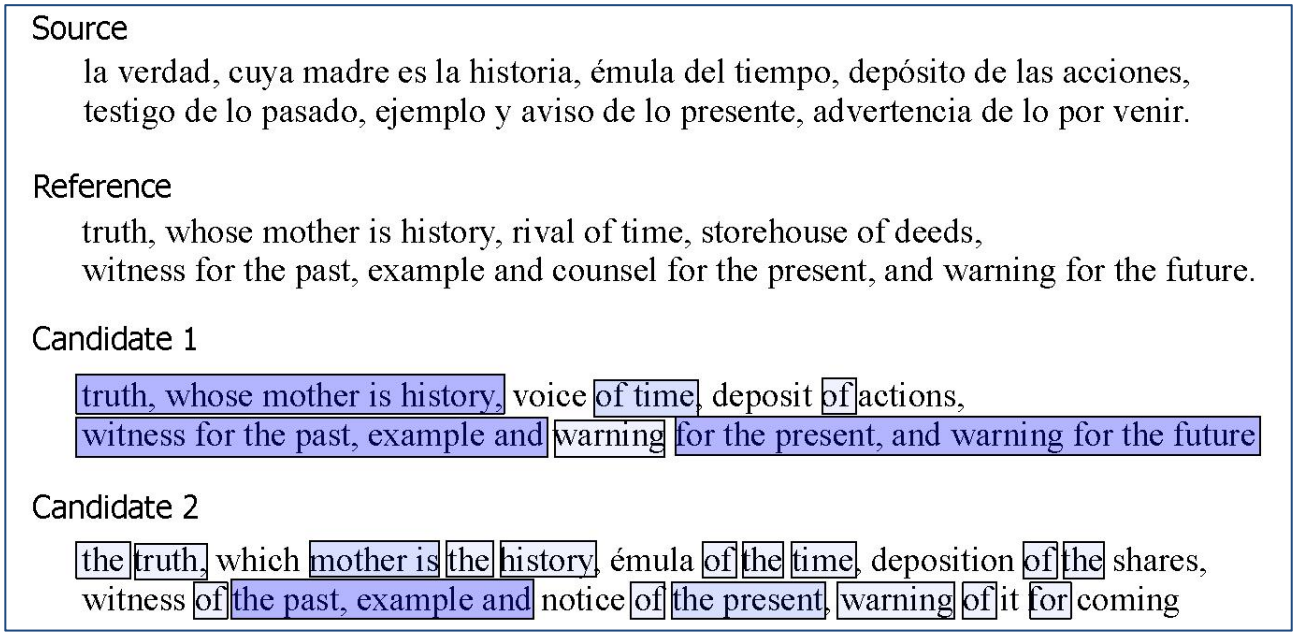


图 11-16：BLEU 的直觉

处理此问题的一种方法是将召回率与精度结合起来，但是 BLEU 选择了另一种方法：在整个语料库中添加简短惩罚，对产生平均比参考译文短的译文的系统进行处罚。令 sys_len 为所有候选翻译语句的长度之和，ref_len 为所有参考翻译语句的长度之和。如果候选译文比参考译文短，则我们指定简短惩罚 BP，这是它们的比率的函数：

$$\begin{aligned} \mathrm{BP} &= \min\left(1, \exp\left(1 - \frac{\mathrm{ref\_len}}{\mathrm{sys\_len}}\right)\right) \\ \mathrm{BLEU} &= \mathrm{BP} \times \left(\prod_{n=1}^{4} \mathrm{prec}_n\right)^{\frac{1}{4}} \end{aligned} \tag{11.25}$$

BLEU 的高级细节

上面的描述在许多方面得到了简化。BLEU 实际上使用的 n-gram 精度与公式 11.23 稍有不同。公式 11.23 有个缺点，就是会奖励有多余重复单词的候选人。图 11.17 显示了一个由 7 个单个单词 the 组成的病态的候选句子的例子，其 unigram 精度为 7/7!

为避免此问题，BLEU 使用了**改进的 n-gram 精度(modified n-gram precision)**度量。我们首先计算一个单词在任何单个参考译文中使用的最大次数。然后，每个**候选**单词的计数将被此最大**参考**计数限制。因此，在图 11.17 的示例中，修改后的字母组合精度为 2/7，因为参考句子 1 中 the 出现的最大次数为 2。

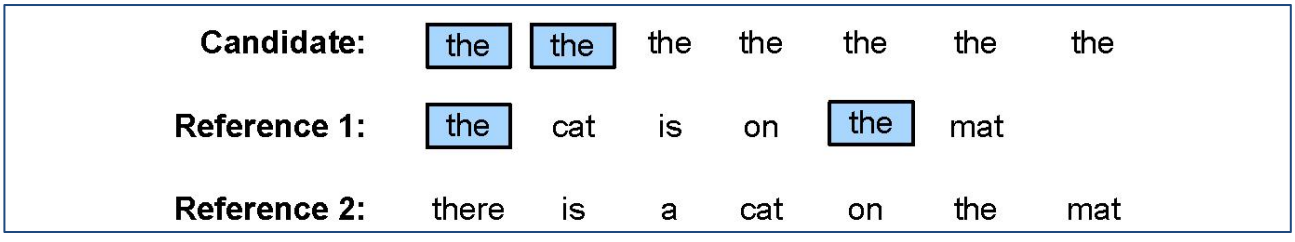


图 11-17：BLEU 使用的病态例句

图注：Unigram 的精度会不合理的高(7/7)。改良的 unigram 精度适当的低(2/7)。

为了计算整个测试集的分数，BLEU 首先为每个句子计算 n-gram 匹配，然后将所有候选句子的裁剪后的计数求和，最后除以测试集中的候选 n-gram 总数。如果我们将 $\text{Count}_{\text{match\_clipped}}$ 函数定义为表示“与参考匹配的所有 n-gram 的修剪计数”，那么 BLEU 使用的候选句子的整个语料库的精度 prec_n 为：

$$\text{prec}_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} \text{Count}_{\text{match\_clipped}}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} \text{Count}(n\text{-}gram')} \tag{11.26}$$

如果我们对一个源句子有多个人工参考翻译，BLEU 也可以很好地工作。

事实上，BLEU 在这种情况下工作得更好，因为源句子可以以多种方式合法地翻译，因此 n-gram 的精度将更加健壮。如果它出现在任何参考中,则我们只匹配 n-gram。对于简短惩罚，我们为每个候选句子选择长度最接近的参考句子来计算 ref_len。但在实践中，大多数翻译语料库只有一个人翻译来进行比较。

最后，实现 BLEU 需要对平滑和符记化的许多细节进行标准化;因此，建议使用标准的实现方法，如 SACREBLEU (Post, 2018)，而不是尝试从头实现 BLEU。

BLEU 的统计显著性检验

BLEU 分数主要用于比较两个系统，目的是回答以下问题：我们发明的特殊新算法是否改善了 MT 系统？ 要了解两个 MT 系统的 BLEU 得分之间的差异是否存在显著差异，我们使用配对的**启动(bootstrap)测试**或类似的随机测试。

为了使用启动测试获得单个 BLEU 得分的置信区间，请回顾第 4.9 节，我们采用测试集(或 devset)并通过重复采样并替换原始测试集来创建数千个伪测试集。现在，我们计算每个伪检验集的 BLEU 分数。如果我们降低得分最高的 2.5%和最低的 2.5%，其余得分将为我们的系统 BLEU 得分提供 95%的置信区间。

为了比较两个 MT 系统 A 和 B，我们绘制了同一组伪测试集，并计算每个伪测试集的 BLEU 分数。然后我们计算 A 的 BLEU 分数高于 B 的伪测试集的百分比。

BLEU 的局限性

虽然像 BLEU 这样的自动度量很有用，但它们有重要的局限性。BLEU 是非常局部的:一个大的短语被移来移去可能根本不会改变 BLEU 分数，而且 BLEU 也不能评估一个文档的跨句子属性，比如它的话语连贯性(第 22 章)。BLEU 和类似的自动度量在比较非常不同类型的系统时也做得很差，例如比较人工辅助翻译和机器翻译，或者不同的机器翻译体系结构之间的差异(Callison-Burch 等人，2006)。当评估单个系统的变更时，这种自动的度量可能是最合适的。

11.8.3. 自动评估：基于嵌入的方法

BLEU 度量是基于测量人类参考和候选机器翻译有共同之处的确切单词或 n-gram。然而，这一标准过于严格，因为好的翻译可能会使用替换词或释义。一个解决方案在早期的度量诸如 METEOR(Banerjee 和 Lavie, 2005)是允许同义词之间的参考 x 和候选$\tilde{x}$。最近的度量使用 BERT 或其他嵌入来实现这种直觉。

例如，在某些情况下，我们可能拥有对翻译质量进行人工评估的数据集。这样的数据集由元组(x，$\tilde{x}$，r)组成，其中 $x=(x_1, \ldots, x_n)$是参考翻译，$\tilde{x}=(\tilde{x}_1, \ldots, \tilde{x}_m)$是候选机器翻译 ，并且 $r \in R$ 是人工评分，表示相对于 x 的$\tilde{x}$的质量。给定这样的数据，像 BLEURT(Sellam 等人，2020)之类的算法通过将 x 和$\tilde{x}$传递给 BERT 版本(经过额外的预训练，然后在人工标签的句子上进行微调)，最后是训练以预测 r 的线性层。这种模型的输出与人工标签高度相关。

但是，在其他情况下，我们没有此类人工标记的数据集。在那种情况下，我们可以通过它们的嵌入相似性来测量 x 和$\tilde{x}$的相似性。例如，图 11.18 所示的 BERTSCORE 算法(Zhang 等人，2020 年)通过 BERT 传递参考 x 和候选对象$\tilde{x}$，计算每个符记 x_i 和$\tilde{x}_j$ 的 BERT 嵌入。每对符记(x_i，$\tilde{x}_j$)通过其余弦$(x_i \cdot \tilde{x}_j)/(|x_i||\tilde{x}_j|)$进行评分。$x$ 中的每个符记与$\tilde{x}$中的符记匹配以计算召回率，并且$\tilde{x}$中的每个符记与 x 中的符记作匹配以计算精度(每个符记都贪心地与对应句子中最相似的符记匹配)。BERTSCORE 提供精度和召回率(以及 F1)：

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j \qquad P_{\text{BERT}} = \frac{1}{|\tilde{x}|}\sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j \tag{11.27}$$

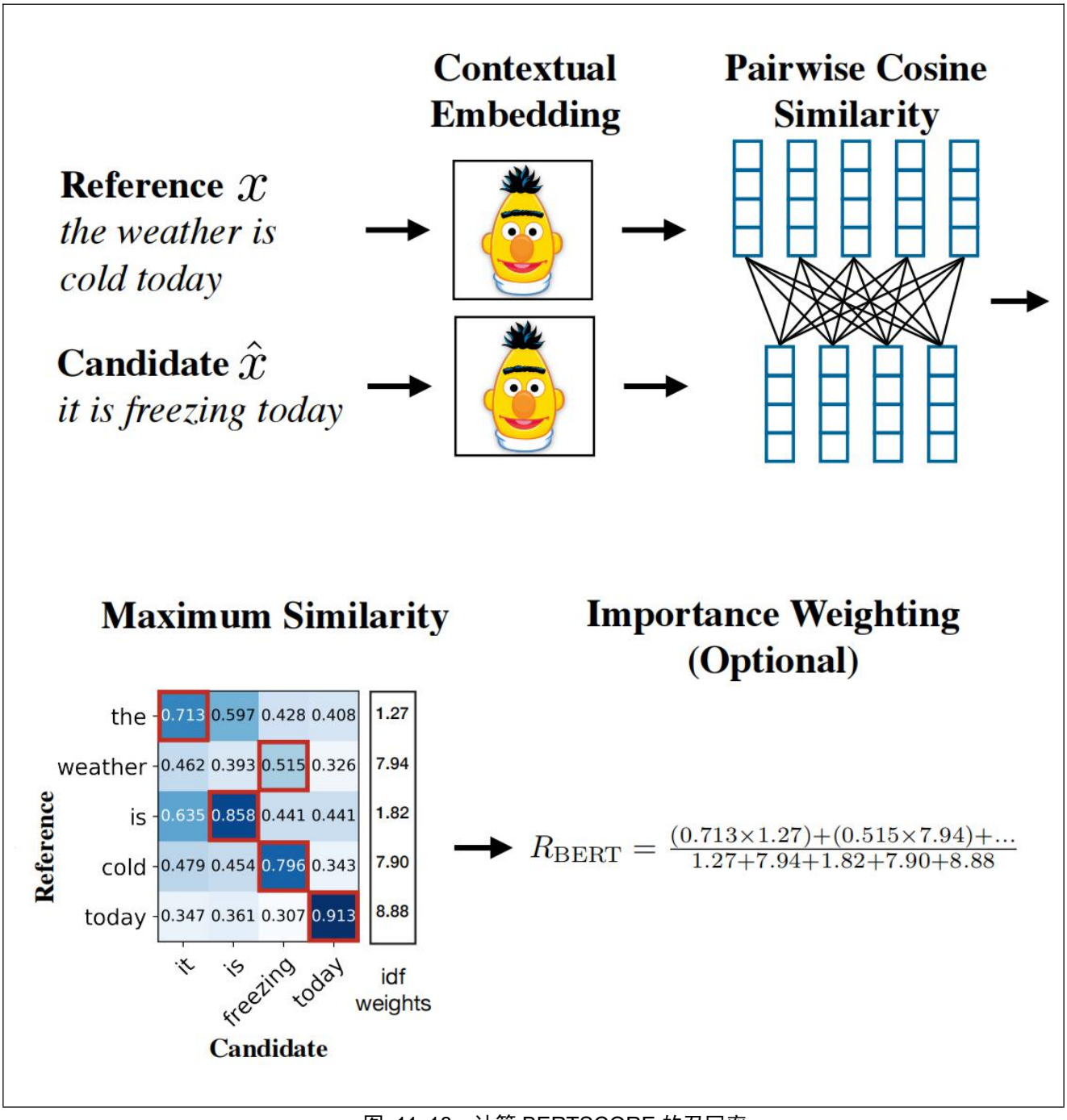


图 11-18：计算 BERTSCORE 的召回率

图注：根据 Zhang 等人(2020)中的图 1，借由参考 x 和候选$\tilde{x}$，计算 BERTSCORE 的召回率。他的版本显示了该度量的扩展版本，其中符记也由它们的 idf 值加权。

11.9. 偏见和道德问题

机器翻译引发了许多我们在前面章节中讨论过的伦理问题。例如，考虑 MT 系统将匈牙利语(有性别中立代名词ö)或西班牙语(经常省略代名词)翻译成英语(代名词是强制执行的，它们有语法上的性别)。当翻译一个没有指定性别的人的参考句子时，机器翻译系统通常默认为男性性别(Schiebinger 2014, Prates 等人，2019)。MT 系统通常根据我们在第 6.11 节中看到的文化刻板印象来分配性别。图 11.19 显示了来自(Prates 等人，2019)的示例，匈牙利语中的性别中立词“ö”，在“ö is a nurse”中被翻译成“she”，在“ö is a CEO”

被翻译成“he”。Prates 等人(2019)发现这些刻板印象不能完全由美国劳工统计中的性别偏见来解释，因为偏见被 MT 系统放大，代词被映射到男性或女性的概率要高于基于实际劳动就业统计数据的映射。

同样，最近的一项挑战集，WinoMT 数据集(Stanovsky 等人，2019)表明，当要求 MT 系统翻译描述非刻板性别角色的句子时，如“The doctor asked the nurse to help her in the operation（医生请护士帮助她做手术）”，MT 系统的表现更差。

Hungarian (gender neutral) source	English MT output
ő egy ápoló	she is a nurse
ő egy tudós	he is a scientist
ő egy mérnök	he is an engineer
ő egy pék	he is a baker
ő egy tanár	she is a teacher
ő egy vesküvőszervező	she is a wedding organizer
ő egy vezérigazgató	he is a CEO

图 11-19：匈牙利语译成英语时的性别问题

图注：当从匈牙利语等性别中立语言翻译成英语时，当前的机器翻译系统将人们从传统上以男性为主的职业解释为男性，而将传统上以女性为主的职业解释为女性(Prates 等人，2019)。

MT 中许多开放的道德问题需要进一步研究。其中之一是需要更好的指标来了解我们的系统不知道的内容。MT 系统可用于人工翻译人员不可用或延迟的紧急情况：在医学领域，当患者和医生使用不同语言时，可帮助翻译；在法律领域，帮助法官或律师与证人或被告沟通。为了“不造成伤害”，系统需要为候选译文分配**置信度(confidence)**值的方法，因此它们可以避免提供可能造成伤害的不正确译文。

另一个问题是需要低资源的算法，该算法可以与世界上绝大多数语言进行双向翻译，而没有大量并行文本可用于培训。MT 的跨语言迁移和多语言方法往往侧重于一种语言是英语的情况(Anastasopoulos 和 Neubig，2020)，这一事实使这一问题更加严重。∀等人(2020)提出了一个参与性设计过程，以鼓励讲这些**低资源语言(low-resourced languages)**的内容创建者、策展人和语言技术人员参与 MT 算法的开发。他们的方法使用在线小组、在线指导和在线基础设施，并且他们报告了有关开发针对低资源的非洲语言的 MT 算法的案例研究。

11.10. 总结

机器翻译是 NLP 最广泛的应用之一，而最早为 MT 开发的编码器-解码器模型是在 NLP 中广泛应用的一个关键工具。

•语言在结构和词汇上都有**差异性**，这使得翻译困难。

•**类型学**的语言领域研究这些差异;语言可以按照它们的位置沿着类型维度进行分类，比如动词是否在它们的对象之前。

•**编码器-解码器**网络由一个**编码器**网络组成，接受一个输入序列并创建一个上下文的表示。这个**上下文**表示然后被传递到一个**解码器**，以生成一个特定于任务的输出序列。

•**注意力**机制丰富了上下文向量，允许解码器从编码器的所有隐藏状态查看信息，而不仅仅是最后一个隐藏状态。

•编码器-解码器架构可以通过 RNNs 或者 Transformers 实现。

•对于解码器，选择在每个步骤中生成的单个最可能的符记称为**贪心**解码。

•在**波束搜索**中，我们在每一步都保留 k 个可能的符记，而不是在每个时间步都选择最优的符记来生成。这种固定大小的记忆足迹 k 称为**波束宽度**。

•机器翻译模型在**平行语料库**上进行训练，亦称 **bitext**，即一种出现在两种(或更多)语言中的文本。

•**反向翻译**是一种在目标语言中使用单语语料库的方法，它通过向后运行领航(pilot)MT 引擎来创建合成 bitexts。

•翻译质量的评估是通过衡量译文的**充分性**(对源语句子含义的把握程度)和**流利性**(在目标语言中的流利程度或自然程度)来进行的。人工评价是黄金标准，但像 **BLEU** 这样的自动评价指标(它测量单词或 n-gram 与人工翻译的重叠)，或基于嵌入相似性的更近期的指标，也很常用。

11.11. 文献和历史说明

在 20 世纪 40 年代末期，计算机诞生后不久，MT 就被认真地提出了(Weaver，1955)。1954 年，MT 系统原型的首次公开演示(Dostert, 1955)在新闻界引起了极大的兴奋(Hutchins, 1997)。接下来的十年见证了思想的大开花，预示了随后的大多数发展。但是这项工作是超前的，实现受到了限制，例如，在磁盘开发之前，没有好的方法来存储字典信息。

随着高质量的机器翻译被证明是难以实现的(Bar-Hillel, 1960)，在形式语言学和计算语言学的新领域中，有一种共识是需要更好的评价和更多的基础研究。这一共识在 1966 年著名的 ALPAC(自动语言处理咨询委员会)报告(Pierce 等人，1966)中达到了顶峰，该报告导致了 20 世纪 60 年代中期美国对机器翻译的资金大幅削减。由于机器翻译研究失去了学术地位，机器翻译和计算语言学协会从名称中删除了机器翻译。然而，一些 MT 开发者坚持了下来，出现了早期的工业引擎，如 Systran，以及早期的 MT 系统诸如 Meteo，将天气预报从英语翻译成法语(chanddioux，1976)。

在早期，MT 体系结构的空间跨越了三个通用模型。第一个是直接模型，直接翻译可能是最早发展起来的方法，该系统逐字逐句地翻译源语言文本中的每个单词。直接翻译使用大型双语词典，每个词典的条目都是一个小的程序，负责翻译一个单词。第二个是转移模型，在转移方法中，我们首先解析输入文本，然后应用规则将源语言解析转换为目标语言解析。然后，我们从解析树生成目标语言句子。第三个是国际语模型，在**国际语(interlingua)**方法中，我们将源语言文本分析为某种抽象含义表示形式，称为国际语。然后，我们从这种语言间的表示形式生成目标语言。可视化这三种早期方法的常见方法是图 11.20 中所示的**沃夸(Vauquois)**三角形。三角形显示了从直接方法到转移方法到国际语方法时所需的分析深度(在分析和生成端)都在逐渐增加，而所需的转移知识的数量却在逐渐减少。我们可以看到在三角形中向上移动的过程：从直接层的大量转移（几乎所有知识都是每个单词的转移知识），到转移层（仅针对解析树或主题角色的转移规则），到国际语层（没有特定的转移知识）。我们可以将编码器-解码器网络视为一种国际语的方法，注意可以将直接和转移集成在一起，从而使解码器可以直接访问单词或它们的表示形式。

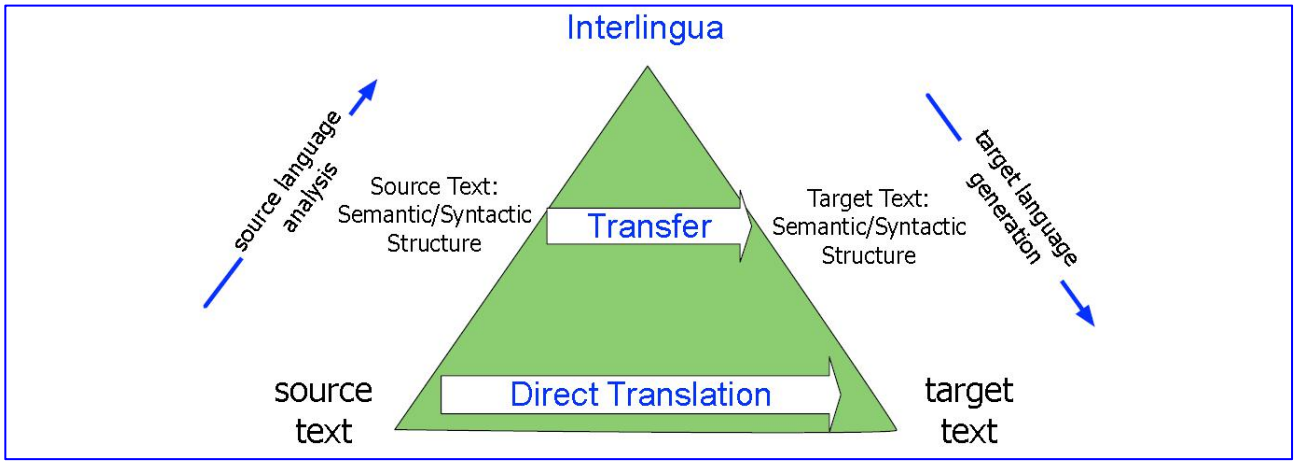


图 11-20：**沃夸**三角形

统计方法开始于 1990 年左右开始应用，首先是通过发展大型双语语料库,例如加拿大议会议事录中的 **Hansard** 语料库，并以法文和英文两种形式进行保存，然后通过网络的发展来实现。早期，许多研究人员表明，可以使用单词或句子长度之类的简单提示从双语语料库中提取成对的对齐句子(Kay 和 Roscheisen 1988；Gale 和 Church 1991；Gale 和 Church 1993； Kay and Roscheisen 1993)。

同时，IBM 小组直接利用用于语音识别的噪声信道模型，提出了用于统计 MT 的算法，该算法被称为 **Candide** 系统中的 **IBM 模型 1-5**。

这些算法(除了解码器)被详细地发表了——受到了部分资助这项工作的美国政府的鼓励——这给了他们在研究界巨大的影响(Brown 等人 1990, Brown 等人 1993)。到世纪之交，大多数关于机器翻译的学术研究都使用了统计噪声信道模型。通过开发公开可用的工具包，例如 GIZA 工具包(Och 和 Ney, 2003)，它实现了 IBM 模型 1-5 和 HMM 对齐模型，使得进展变得非常容易。

大约在世纪之交，一种扩展的方法被称为**基于短语的翻译(phrase-based translation)**，它基于短语对的归纳翻译(Och 1998, Marcu 和 Wong 2002, Koehn 等人 2003,Och 和 Ney 2004, Deng 和 Byrne 2005，等等)。一个对数线性公式(Och 和 Ney, 2004)在一种被称为**最小错误率训练(Minimum Error Rate Training)**或 MERT(Och, 2003)的方法中被训练来直接优化评价指标，如 BLEU(也来自语音识别模型)(Chou 等人，1993)。流行的工具包诸如**摩西(Moses)**被开发出来(Koehn 等人 2006，Zens 和 Ney 2007)。

在世纪之交也有一些基于句法结构的方法(第 12 章)。

基于**转导语法(transduction grammar)**(亦称**同步语法(synchronous grammar)**)的模型为不同语言的一对句子分配一个平行的句法树结构，其目标是通过在树上应用重新排序操作来翻译句子。从生成的角度来看，我们可以把转换语法看作是在两种语言中生成配对的对齐句子。一些最广泛使用的模型包括**倒装转导语法(inversion transduction grammar)**(Wu, 1996)和同步上下文无关语法(Chiang, 2005)。

现代历史上的编码器-解码器方法 HERE;(Kalchbrenner and Blunsom, 2013)，(Cho 等人，2014)，(Sutskever 等人，2014)，等等。波束搜索与人类语言处理有着有趣的关系; Meister 等人(2020)表明，波束搜索增强了文本中**均匀信息密度(uniform information density)**的认知特性。均匀信息密度是一种假设，即人类语言处理器倾向于在句子中均匀分布信息(Jaeger 和 Levy，2007)。

关于机器翻译评价的研究起步较早。Miller 和 Beebe-Center(1958)借鉴心理语言学的工作，提出了一些方法。其中包括使用完形填空(cloze)和香农(Shannon)任务来测量可理解性，以及与人工翻译的编辑距离的度量，这种直觉是所有现代自动评估指标(如 BLEU)的基础。ALPAC 报告包括了 John Carroll 进行的一项极具影响力的早期评估研究(Pierce 等人，1966)。Carroll 提出了不同的逼真度和可理解度的衡量标准，并让评分者以 9 分的量表为他们主观打分。

最近有关评估的工作主要集中在提出自动指标上，包括第 11.8.2 节(Papineni 等人，2002)中讨论的关于 BLEU 的工作，以及诸如 NIST(Doddington，2002)，TER(翻译错误率)等相关措施(Snover 等人，2006)，精度和召回率(Turian 等人，2003)和 METEOR(Banerjee 和 Lavie，2005)。Hutchins(1986)和(1997)对 MT 的早期历史进行了很好的调查。Nirenburg 等人(2002)是 MT 早期读物的集合。

有关类型学的介绍，请参见 Croft(1990)或 Comrie(1989)。

11.12. 练习

(无)

12. 成分语法

If on a winter ' s night a traveler by Italo Calvino
Nuclear and Radiochemistry by Gerhart Friedlander et al.
The Fire Next Time by James Baldwin
A Tad Overweight, but Violet Eyes to Die For by G. B. Trudeau
Sometimes a Great Notion by Ken Kesey
Dancer from the Dance by Andrew Holleran

Six books in English whose titles are not constituents, from Pullum (1991, p. 195)

语法研究有着古老的血统。帕尼尼(Panini)的梵文语法写于两千多年前，今天在教梵文时仍被引用。我们的单词**句法(syntax)**来自希腊语 sýntaxis，含义是“一起摆放或排列”，是指单词排列在一起的方式。在前面的章节中，我们已经看到了各种句法概念：单词序列的顺序(第 2 章)，这些单词序列的概率(第 3 章)以及使用词类类别作为单词的语法等价类(第 8 章))。在本章和接下来的三章中，我们介绍了远远超出这些简单方法的各种句法现象，以及用于以计算上有用的方式捕获它们的形式模型。

本章的大部分内容专用于上下文无关的语法。上下文无关语法是自然语言(以及计算机语言)句法的许多形式化模型的基础。因此，它们在许多计算应用程序中都发挥着作用，包括语法检查，语义解释，对话理解和机器翻译。它们足够强大，可以表达句子中各个单词之间的复杂关系，但在计算上也很容易处理，以至于存在有效的算法来解析句子(如第 13 章所示)。在第 16 章中，我们将展示它们如何为语义解释提供系统的框架。在这里，我们还介绍了词汇语法的概念，重点放在一个示例上，即**组合范畴语法(combinatory categorial grammar，CCG)**。

在第 14 章中，我们将介绍一个被称为**句法依存(syntactic dependencies)**的正式语法模型，它是这些成分语法的替代方案，我们将给出用于**依存解析(dependency parsing)**的算法。成分形式和依存形式对于语言处理都很重要。

最后，我们从一个叫做 ATIS(空中交通信息系统)的具有相对简单句子的领域(Hemphill 等人，1990)给出了英语语法的简要概述。ATIS 系统是一种早期的口语系统，用户可以通过表达像 I 'd like to fly to Atlanta 这样的句子来预订航班。

12.1. 成分

句法成分是指一组单词可以作为单个单元或成分。发展语法的一部分是建立语言成分的清单。英语中的单词是怎样组合在一起的?考虑名词短语(noun phrase)，它是至少包含一个名词的一系列单词。以下是一些名词短语的例子(thanks to Damon Runyon)：

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

我们有什么证据证明这些词可以组合在一起(或“构成成分”)? 有证据表明，它们都可以出现在相似的句法环境中，例如，出现在动词之前。

three parties from Brooklyn arrive. . .
a high-class spot such as Mindy's attracts. . .
the Broadway coppers love. . .
they sit

但是，虽然整个名词短语可以出现在动词之前，但组成名词短语的每个单词却不是这样。以下是不符

合语法的英语句子(记得我们用星号(*)来标记不符合语法的英语句子):

*from *arrive*. . .	*as *attracts*. . .
*the *is*. . .	*spot *sat*. . .

因此，要正确地描述这些单词在英语中的顺序，我们必须能够说“Noun Phrases can occur before verbs”这样的话。

其他类型的成分证据来自所谓的**前置(preposed)**或**后置(postposed)**构造。例如，在以下示例中，可以将 on September seventeenth 的介词短语放置在多个不同的位置，包括在开头(前置)或结尾(后置)：

On September seventeenth, I'd like to fly from Atlanta to Denver

I'd like to fly ***on September seventeenth*** from Atlanta to Denver

I'd like to fly from Atlanta to Denver ***on September seventeenth***

但是，虽然整个短语可以放在不同的位置，但是组成这个短语的每个单词不能是:

*On September, I'd like to fly seventeenth from Atlanta to Denver

*On I'd like to fly September seventeenth from Atlanta to Denver

*I'd like to fly on September from Atlanta to Denver seventeenth

12.2. 上下文无关语法(CFG)

使用英语和其他自然语言对成分结构进行建模的最广泛使用的形式系统是**上下文无关语法(Context-Free Grammar，CFG)**。上下文无关的语法也称为**短语结构语法(Phrase-Structure Grammars)**[1]，而它的形式化方法等效于**巴科斯-诺尔范式(Backus-Naur-Form，BNF)**。将语法建立在成分结构上的思想可以追溯到心理学家威廉·温德(Wilhelm Wundt，1900)，但直到乔姆斯基(Chomsky，1956)和巴克(Backus，1959)才正式成立。

上下文无关的语法由一组**规则**或**产生式**，以及单词和符号的**词表(lexicon)**组成。每个规则或产生式表示可以将语言的符号进行组合和排序的方式。例如，下面的结果表示一个 **NP**(或名词短语)可以由一个专有名词或一个限定词(Det)后跟一个**名词性词(Nominal)**组成，一个名词性词可以依次由一个或多个名词组成。

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun \mid Nominal\ Noun \end{aligned}$$

上下文无关的规则可以分层次地嵌入，因此我们可以将之前的规则与其他规则结合起来，如下面所示，这些规则表达了关于词表的事实:

$$\begin{aligned} Det &\rightarrow a \\ Det &\rightarrow the \\ Noun &\rightarrow flight \end{aligned}$$

CFG 中使用的符号分为两类。与语言中单词对应的符号(“The”、“nightclub”)称为**终端(terminal)**符号;词表是一组引入这些终端符号的规则。在这些终端上表达抽象的符号称为**非终端(non-terminal)**。在每个上下文无关的规则中，箭头(→)右边的项是一个由一个或多个终端和非终端组成的有序列表;箭头左边是一个表示某些集群或泛化的非终结符号。与词表中每个单词相关联的非终端是它的词汇范畴，或词类。

---

[1] 译者注：语言结构的两种观点:成分和依存(Constituency and Dependency)结构。

- 成分=短语结构语法=上下文无关语法(CFGs)

  短语结构将单词组织成嵌套的(nested)成分。

- 依存结构显示了哪些单词依赖于哪些其他单词。

摘自《Natural Language Processing with Deep Learning》CS224N/Ling284，Christopher Manning，Lecture 4: Dependency Parsing，Stanford / Winter 2021。

可以用两种方式来理解 CFG:一种是用来生成句子的工具，另一种是用来为给定的句子分配结构的工具。将 CFG 作为生成器来查看，我们可以将“→”箭头理解为“用右边的符号串重写左边的符号”。

So starting from the symbol:	*NP*
we can use our first rule to rewrite *NP* as:	*Det Nominal*
and then rewrite *Nominal* as:	*Det Noun*
and finally rewrite these parts-of-speech as:	*a flight*

我们说 a flight 字符串可以从非终端 NP 派生。因此，CFG 可用于生成一组字符串。规则扩展的这种序列称为单词字符串的**派生(derivation)**。通常用**解析树(parse tree)**来表示派生树(通常以根倒置显示)。图 12.1 显示了该派生树表示。

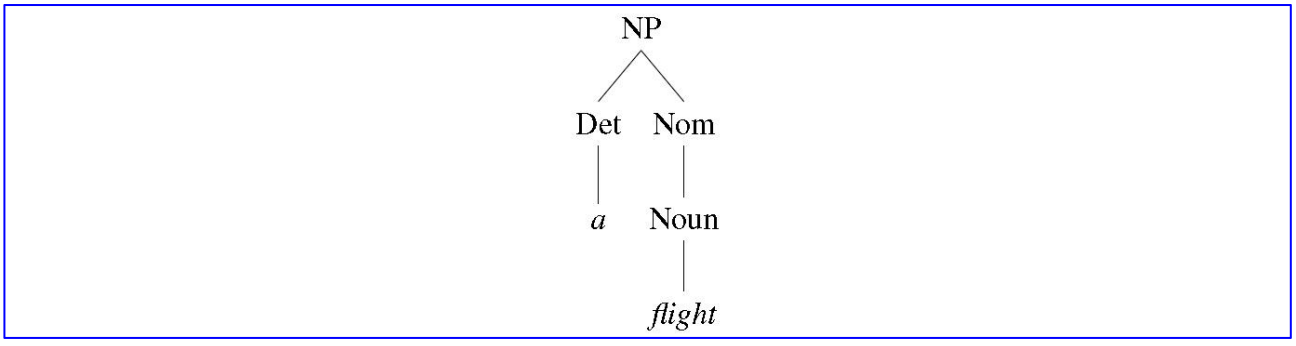


图 12-1：“A flight”的解析树

在图 12.1 所示的解析树中，我们可以说节点 NP **支配(dominate)**着树中的所有节点(Det, Nom, Noun, a, flight)。进一步说，它直接支配节点 Det 和 Nom。

CFG 定义的形式语言是从**指定的起始符号(designated start symbol)**派生的一组字符串。每个语法必须有一个指定的起始符号，通常称为 S。由于上下文无关的语法通常用于定义句子，因此 S 通常被解释为“句子”节点，从 S 派生的字符串集是英语的某些简化版本中的句子集。

让我们在清单上增加一些额外的规则。下面的规则表达了一个句子可以由一个名词短语和一个动词短语组成的事实:

S → NP VP I prefer a morning flight

英语中的动词短语是由一个动词和搭配的其他事物组成的;例如，有一种动词短语由一个动词后跟一个名词短语组成:

VP → Verb NP prefer a morning flight

或者动词后面可以跟名词短语和介词短语:

VP → Verb NP PP leave Boston in the morning

或者动词短语中可以有一个动词后跟一个介词短语:

VP → Verb PP leaving on Thursday

介词短语通常由一个介词跟一个名词短语组成。例如，在 ATIS 语料库中，一种常见的介词短语被用来表示位置或方向:

PP → Preposition NP from Los Angeles

PP 中的 NP 不一定是一个位置;PPs 经常和时间、日期连用，也可以和其他名词连用;它们可以任意复杂。以下是来自 ATIS 语料库的 10 个例子:

to Seattle	on these flights
in Minneapolis	about the ground transportation in Chicago
on Wednesday	of the round trip flight on United Airlines
in the evening	of the AP fifty seven flight
on the ninth of July	with a stopover in Nashville

图 12.2 给出了一个示例词表，图 12.3 总结了我们到目前为止看到的语法规则，我们将其称为 $\mathcal{L}_0$。请注意，我们可以使用表示“或者”的符号“|”来表示非终端符具有替代的、可能的展开方式。

我们可以使用这种语法生成这种“ATIS-language”的句子。我们从 S 开始，将它展开为 NP VP，然后随机选择 NP 的展开(假设是 I)，VP 的随机展开(假设是动词 NP)，以此类推，直到我们生成一个字符串 I prefer a morning flight。图 12.4 显示了一个表示 I prefer a morning flight 的完整派生的解析树。

我们还可以用一种更紧凑的格式表示解析树，称为**方括号表示法(bracketed notation)**；下面是图 12.4 中解析树方括号表示法:

(12.1) [S [NP [Pro I]] [VP [V prefer] [NP [Det a] [Nom [N morning] [Nom [N flight]]]]]]

类似于 $\mathcal{L}_0$ 的 CFG 定义了一种正式语言。我们在第 2 章中看到，正式语言是一组字符串。可以由语法派生的句子(单词的字符串)是由该语法定义的正式语言，称为**符合语法(grammatical)**句子。不能由给定的形式语法派生的句子不是由该语法定义的语言，被称为**不合语法的(ungrammatical)**。

Noun	→flights | breeze | trip | morning
Verb	→is | prefer | like | need | want | fly
Adjective	→cheapest | non-stop | first | latest | other | direct
Pronoun	→me | I | you | it
Proper-Noun	→Alaska | Baltimore | Los Angeles | Chicago | United | American
Determiner	→the | a | an | this | these | that
Preposition	→from | to | on | near
Conjunction	→and | or | but

图 12-2：$\mathcal{L}_0$ 的词汇表

Grammar Rules		Examples
S →	NP VP	I + want a morning flight
NP →	Pronoun	I
|	Proper-Noun	Los Angeles
|	Det Nominal	a + flight
Nominal →	Nominal Noun	morning + flight
|	Noun	flights
VP →	Verb	do
|	Verb NP	want + a flight
|	Verb NP PP	leave + Boston + in the morning
|	Verb PP	leaving + on Thursday
PP →	Preposition NP	from + Los Angeles

图 12-3：$\mathcal{L}_0$ 的语法以及每个规则的示例短语

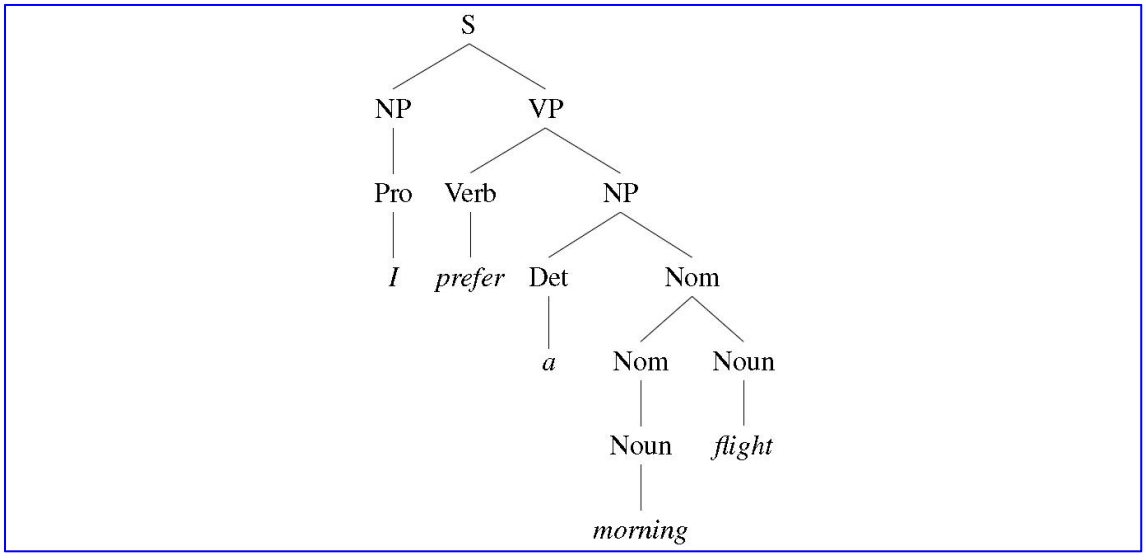


图 12-4：根据语法 $\mathcal{L}_0$ 的“I prefer a morning flight”的解析树

在“输入”和“输出”之间的强硬线（hard line）代表了所有形式语言的特征，但这只是自然语言如

何工作的非常简化的模型。这是因为确定给定句子是否属于给定自然语言(例如英语)的一部分通常取决于上下文。在语言学中，使用形式语言来对自然语言进行建模被称为**生成语法(generative grammar)**，因为该语言是由语法“生成”的一组可能的句子定义的。

12.2.1. 上下文无关语法的正式定义

我们通过对上下文无关的语法及其生成的语言的快速、正式的描述来结束本节。一个上下文无关的语法 G 由四个参数定义：N，Σ，R，S（技术上这是一个“4 元组”）。

> N 非终结符(或变量)的集合
> Σ终端符号(与N不相交)的集合
> R 规则的集合或生成式，每个生成式的形式为A→β，
> 其中：A 是非终端符号，
> β是由字符串(Σ∪N)*中无穷集合的符号构成的字符串
> S 是指定的起始符号，并且是N的成员

在本书的其余部分中，我们在讨论上下文无关语法的形式属性时(与解释关于英语或其他语言的特定事实相反)遵循以下约定：

诸如A、B和S的大写字母	非终端符号
S	起始符号
诸如α、β和γ的小写希腊字母	从(Σ∪N)*推导的符号串
诸如u、v和w的小写罗马字母	终端符号串

语言是通过派生的概念定义的。如果可以通过一系列规则应用程序将一个字符串重写为第二个字符串，则它会派生另一个字符串。根据 Hopcroft 和 Ullman(1979)，可以更正式地说：

> 如果A→β是R的一个生成式，α和γ是(Σ∪N)*集合中的任意字符串，
> 则我们就说，αAγ直接派生出αβγ，或αAγ ⇒ αβγ。

那么派生就是直接派生的概括：

令α_1，α_2，...，α_m为$(\Sigma \cup N)^*$，m≥1 中的字符串，使得

$$\alpha_1 \Rightarrow \alpha_2，\ \alpha_2 \Rightarrow \alpha_3，\ \ldots，\ \alpha_{m-1} \Rightarrow \alpha_m$$

我们说，α_1派生出α_m，或$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$。

然后，我们可以正式地将语法 G 生成的语言 $\mathcal{L}_G$ 定义为一组由可以从指定的开始符号 S 派生的终端符号组成的字符串集合。

$$\mathcal{L}_G = \{w | w \text{ is in } \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$$

将一串单词映射到它的解析树的问题称为**句法解析(syntactic parsing)**，我们在第 13 章中定义了成分解析的算法。

12.3. 一些英语语法规则

在这一节中，我们将介绍英语短语结构的几个方面;为了保持一致性，我们将继续关注来自 ATIS 领域的句子。由于篇幅有限，我们的讨论必然局限于重点内容。强烈建议读者参考一本好的英语语法参考书，如 Huddleston 和 Pullum(2002)。

12.3.1. 句级结构

在小语法 $\mathcal{L}_0$ 中，我们只为陈述性句子提供了一个句子级的构造，就像 I prefer a morning flight 一样。在大量的英语句子构造中，特别常见和重要的句子有**四个：陈述式(declaratives)**、**命令式(imperatives)**、

是非疑问式(yes-no questions)和 **wh-疑问式(wh-questions)**。

陈述式(declaratives)结构的句子有一个主语名词短语，后面跟着一个动词短语，比如“I prefer a morning flight”。这种结构的句子有很多不同的用法，我们将在第 24 章继续讨论。下面是一些来自 ATIS 领域的例子:

I want a flight from Ontario to Chicago
The flight should be eleven a.m. tomorrow
The return flight should leave at around seven p.m.

命令式(imperatives)结构的句子通常以动词短语开头，没有主语。之所以称为命令式命令，是因为它们几乎总是用于命令和建议。在 ATIS 域中，它们是对系统的命令。

Show the lowest fare
Give me Sunday's flights arriving in Las Vegas from New York City
List all flights between five and seven p.m.

我们可以用 S 展开的另一条规则来模拟这个句子结构:

S→VP

是非疑问式(yes-no questions)结构的句子经常(虽然不总是)用来问问题;它们以助动词开头，然后是主语 NP，最后是 VP。下面是一些例子。注意，第三个例子根本不是一个问题，而是一个请求;第 24 章讨论了这些疑问形式的使用，以执行不同的**语用(pragmatic)**功能，如询问、请求、或建议。

Do any of these flights have stops?
Does American's flight eighteen twenty five serve dinner?
Can you give me the same information for United?

这是规则：

$$S \rightarrow Aux\ NP\ VP$$

wh-疑问式(wh-questions)

我们在这里研究的最复杂的句子层次结构是各种各样的 wh 结构。它们之所以这样命名，是因为它们的成分之一是 **wh-短语(wh-phrase)**，也就是说，它包含了 wh-单词(who, whose, when, where, what, which, how, why)。这可以大致分为两类句子层次结构。

第一，**“wh-主语疑问式(wh-subject-question)”**结构与陈述式结构相同，只是第一个名词短语中包含了一些“wh”字。

What airlines fly from Burbank to Denver?
Which flights depart Burbank after noon and arrive in Denver by six p.m?
Whose flights serve breakfast?

这里有一条规则。练习 12.7 讨论组成 Wh-NP 的成分规则。

$$S \rightarrow Wh\text{-}NP\ VP$$

第二，在 **wh-非主语疑问式(wh-non-subject-question)**结构中，wh-短语不是句子的主语，因此句子包含了另一个主语。在这些类型的句子中，助动词出现在主语 NP 之前，就像在是非疑问式结构中一样。下面是一个带有示例规则的示例:

What flights do you have from Burbank to Tacoma Washington?

$$S \rightarrow Wh\text{-}NP\ Aux\ NP\ VP$$

像 wh-非主语疑问式的 Wh-NP 这样的结构包含了所谓的**长距离依存关系(long-distance dependencies)**，因为 Wh-NP“what flight”与 VP 中主要动词“have”在语义上相关的谓词相距甚远。在与上述语法规则兼容的某些解析和理解模型中，长距离依存关系(如 flights 和 have 之间的关系)被视为语义关系。在这样的模型中，弄清楚 flights 是 have 的论元的工作是在语义解释期间完成的。其他解析模型将 flights 和 have 之间的关系表示为语法关系，语法被修改为在动词后面插入一个称为踪迹(trace)或空范畴(empty category)的小标记。我们将在 12.4.1 章节中介绍宾州树库时讨论空范畴模型。

12.3.2. 从句和句子

在我们继续之前，我们应该澄清我们刚才描述的语法中 S 规则的地位。S 规则旨在解释作为话语基本单位的整个句子。然而，S 也可以出现在语法规则的右侧，因此可以嵌入到更大的句子中。很明显，S 不仅仅是作为一个单独的话语单位存在。句子结构(即 S 规则)与其他语法的区别在于它们在某种意义上是**完整的(complete)**。这样，它们就相当于一个从句的概念，而传统语法常常把从句描述为形成一个**完整的思想(complete thought)**。让“完整思想”这个概念更精确的一种方法是说 S 是一个解析树的节点，S 的主要动词在它下面有它所有的**论元(arguments)**。我们稍后再定义论元，但现在让我们看看图 12.4 中“I prefer a morning flight”的图解。动词 prefer 有两个论元：主语 I 和宾语 a morning flight。一个论元出现在 VP 节点下，但是另一个论元(主语 NP)只出现在 S 节点下。

12.3.3. 名词短语

我们的 $\mathcal{L}_0$ 语法介绍了英语中最常见的三种名词短语类型：代词、专有名词、NP→Det Nominal 结构。本节的重点是最后一种类型，因为这是大部分语法复杂性所在的地方。这些名词短语由一个**头部**（名词短语中的**中心词**）以及各种修饰语组成，这些修饰语可以出现在头部名词之前或之后。让我们仔细看看各个部分。

限定词

名词短语可以以简单的词汇限定词开头:

a stop　　the flights　　this flight
those flights　any flights　　some flights

限定词的作用也可以由更复杂的表达式来填充:

United’s flight
United’s pilot’s union
Denver’s mayor’s mother’s canceled flight

在这些例子中，限定词的作用由一个名词短语和一个 ’s 作为所有格标记组成的所有格表达来填充，如下面的规则所示。

$$Det \rightarrow NP's$$

这个规则是递归的(因为 NP 可以从 Det 开始)，这一事实帮助我们为上面的最后两个例子建模，在这两个例子中，一系列所有格表达式充当限定词。

在某些情况下，英语中限定词是可选的。例如，如果限定词修饰的名词是复数，则可以省略:

(12.2)　Show me *flights* from San Francisco to Denver on weekdays

正如我们在第 8 章中看到的，大量名词也不需要确定。记住，物质(mass)名词通常(不总是)涉及到被当作实体的东西(例如，water 和 snow)，不要使用不定冠词“a”，也不要做复数形式。许多抽象名词是集合名词(例如，music 和 homework)。ATIS 领域中的集合名词包括 breakfast, lunch 和 dinner:

(12.3)　Does this flight serve dinner?

名词性词(Nominal)

名词性词紧跟限定词，包含任何前、后中心名词修饰语。正如语法 $\mathcal{L}_0$ 中所示，在最简单的形式中，**名词性词**可以由一个名词组成。

$$Nominal \rightarrow Noun$$

正如我们将看到的，该规则还为各种递归规则的底层提供了基础，用于捕获更复杂的名词性词结构。

中心名词之前

在一个**名词性词**之中，许多不同种类的词类可以出现在中心名词之前，但在限定词(后限定词)之后。

这些词包括**基数词(cardinal numbers)**、**序数词(ordinal numbers)**、**数量修饰语(quantifiers)**和形容词。

基数的例子:

two friends one stop

序数词包括 first、second、third 等，也包括 next、last、past、other 和 another:

the first one the second leg the next day the last flight the other American flight

一些数量修饰语(many， (a) few, several)只出现在复数可数名词中:

many fares

形容词出现在量词之后，名词之前。

a *first-class* fare a *non-stop* flight

the *longest* layover the *earliest* lunch fligh

形容词也可以组合成一个短语，称为**形容词短语(adjective phrase，AP)**。AP 可以在形容词前加一个副词(形容词的定义见第 8 章形容词和副词):

中心名词之后

中心名词后面可以接后修饰语。英语中常见的**名词性词**后修饰语有三种:

介词短语 all flights *from Cleveland*

非限定从句 any flights *arriving after eleven a.m.*

关系从句 a flight *that serves breakfast*

它们在 ATIS 数据库中特别常见，因为它们被用来标记航班的出发地和目的地。

下面是一些介词短语后修饰语的例子，用方括号表示每个短语的边界;注意，两个或更多的 PP 可以串成一个 NP:

all flights *[from Cleveland] [to Newark]*

arrival *[in San Jose] [before seven p.m.]*

a reservation *[on flight six oh six] [from Tampa] [to Montreal]*

这是一个新的名词类成分规则来解释后名词类成分的 PP：

$$Nominal \rightarrow Nominal\ PP$$

最常见的**非限定(non-finite)**后修饰语有三种：动名词(-ing)、-ed 动词和不定式动词。

之所以称为**动名词(gerundive)**后修饰语，是因为它们由动词短语组成，该动词短语以动词的动名词(-ing)形式开头。这里有些例子：

any of those *[leaving on Thursday]*

any flights *[arriving after eleven a.m.]*

flights *[arriving within thirty minutes of each other]*

我们可以使用新的非终端 GerundVP 来定义动名词修饰语的**名词性词**，如下所示：

$$Nominal \rightarrow Nominal\ GerundVP$$

我们可以通过复制所有 VP 产生式来制定 GerundVP 成分的规则，用 GerundV 代替 V。

$$GerundVP \rightarrow GerundV\ NP$$
$$| \ GerundV\ PP \mid GerundV \mid GerundV\ NP\ PP$$

GerundV 可以被定义为:

$$GerundV \rightarrow being \mid arriving \mid leaving \mid ...$$

下面用斜体印出来的短语是另外两种常见的非限定从句、不定式和-ed 形式的例子:

the last flight *to arrive in Boston*

I need to have dinner *served*

Which is the aircraft *used by this flight*?

名词后关系从句(更准确地说是限制性关系从句)是以**关系代词(relative pronoun)**(that 和 who 最常用)开头的从句。在下面的例子中，关系代词作内嵌动词的主语:

a flight *that serves breakfast*

flights *that leave in the morning*

the one *that leaves at ten thirty five*

我们可以添加如下规则来处理这些问题:

Nominal → Nominal RelClause

RelClause → (who | that) VP

关系代词也可以作为内嵌动词的宾语，如下面的例子;我们把编写这类更复杂的关系从句的语法规则的练习留给读者去做。

the earliest American Airlines flight that I can get

各种名词后修饰语可以组合在一起:

a flight *[from Phoenix to Detroit] [leaving Monday evening]*

evening flights *[from Nashville to Houston] [that serve dinner]*

a friend *[living in Denver] [that would like to visit me in DC]*

名词短语之前

修饰和出现在 NP 之前的词类称为前限定词，很多都与数字或数量有关；一个常见的前限定词是 all:

all the flights　　all flights　　all non-stop flights

图 12.5 中给出的名词短语的例子说明了当这些规则组合在一起时所产生的一些复杂性。

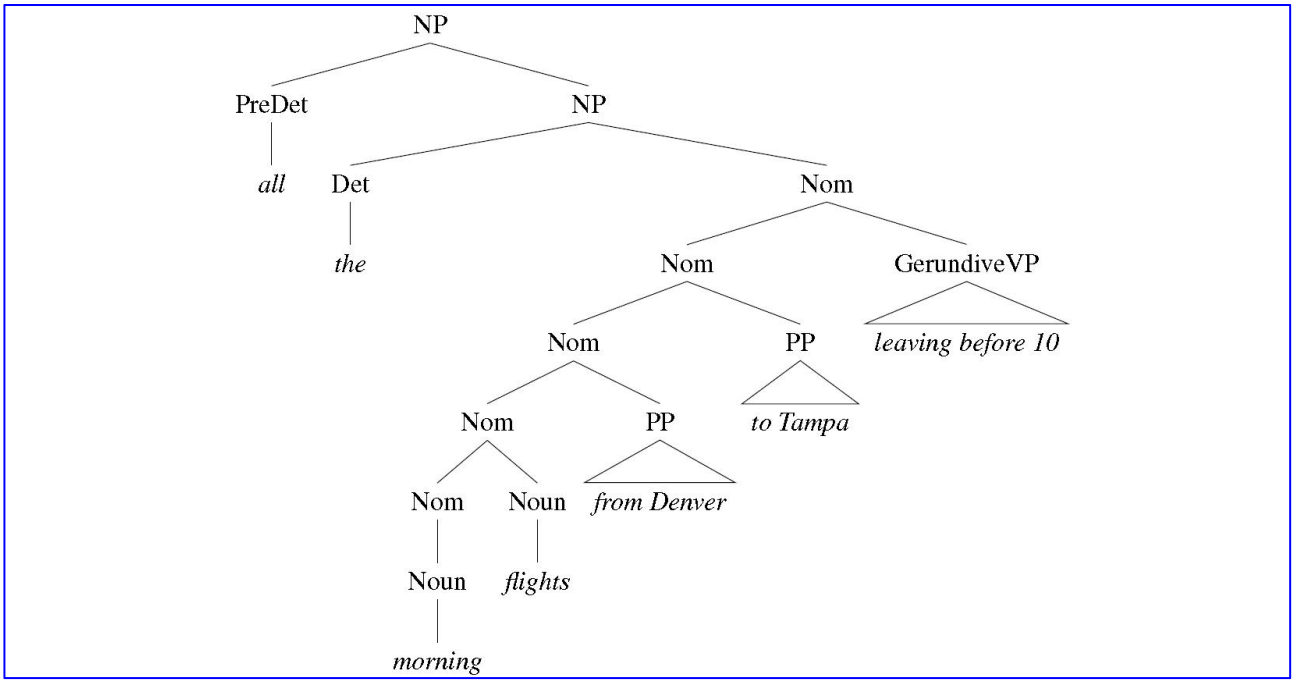


图 12-5：句子解析树的例子

图注：句子“all the morning flights from Denver to Tampa leaving before 10”的解析树。

12.3.4. 动词短语

动词短语由动词和一些其他成分组成。在我们目前建立的简单规则中，其他成分包括 NPs 和 PPs 以及两者的组合:

VP → Verb	disappear
VP → Verb NP	prefer a morning flight
VP → Verb NP PP	leave Boston in the morning
VP → Verb PP	leaving on Thursday

动词短语可以比这个复杂得多。许多其他成分，如整个内嵌的句子，可以跟在动词后面。这些被称为

句子补语(sentential complements):

You [$_{VP}$ [$_{V}$ said [$_{S}$ you had a two hundred sixty-six dollar fare]]

[$_{VP}$ [$_{V}$ Tell] [$_{NP}$ me] [$_{S}$ how to get from the airport to downtown]]

I [$_{VP}$ [$_{V}$ think [$_{S}$ I would like to take the nine thirty flight]]

这里有一条规则:

$$VP \rightarrow Verb\ S$$

类似地，VP 的另一个潜在成分是另一个 VP。像 want, would like, try, intend, need 这样的动词经常出现这种情况:

I want [$_{VP}$ to fly from Milwaukee to Orlando]

Hi, I want [$_{VP}$ to arrange three flights]

虽然一个动词短语可以有很多可能的成分，但并不是每个动词都与每个动词短语兼容。例如，动词 want 既可以与 NP 补语连用(I want a flight…)，也可以与不定式 VP 补语连用(I want fly to…)。相反，像 find 这样的动词就不能用 VP 补语(* I found to fly to Dallas)。

动词与不同种类的补语相容的观点是很古老的;传统语法区分了**及物(transitive)**动词，比如 find，它带一个直接宾语 NP(I found a flight)，和**不及物(intransitive)**动词，比如 disappear，它不带宾语(*I disappear a flight)。

传统语法将动词**次范畴化(subcategorize，再分类)**为两个范畴(及物和不及物)，而现代语法最多可将动词区分为 100 个次范畴。我们说像 find 这样的动词对 NP 进行了再分类，而像 want 这样的动词则对 NP 或非限定 VP 进行了再分类。我们也称这些成分为动词的补语(因此我们在上面使用了术语补语)。所以我们说 want 可以带一个 VP 补语。这些可能的补语集称为动词的次范畴化**框架(frame)**。讨论动词与其他成分之间关系的另一种方法是，将动词视为逻辑谓词，并将成分视为谓词的逻辑论元。因此，我们可以将谓词-论元关系视为 FIND(I，A FLIGHT)或 WANT(I，TO FLY)。当我们讨论动词语义的谓词演算表示法时，我们将在第 15 章中更多地讨论动词和论元的这种观点。图 12.6 给出了一组示例动词的次范畴化框架。

Frame	Verb	Example
$\emptyset$	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
NP PP_{with}	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VPto	prefer, want, need	I would prefer [$_{VPto}$ to go by United Airlines]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

图 12-6：一组实例动词的次范畴化框架

我们可以通过创建类动词的单独子类型来捕获动词及其补语之间的关联(例如：*Verb-with-NP-complement*, *Verb-with-Inf-VP-complement*, *Verb-with-S-complement*, 等等)

Verb-with-NP-complement → *find* | *leave* | *repeat* | ...

Verb-with-S-complement → *think* | *believe* | *say* | ...

Verb-with-Inf-VP-complement → *want* | *try* | *need* | ...

然后可以修改每个 VP 规则，以要求适当的动词子类型:

VP → *Verb-with-no-complement* disappear

VP → *Verb-with-NP-comp NP* prefer a morning flight

VP → *Verb-with-S-comp S* said there were two flights

这种方法的一个问题是规则数量的显著增加以及相关的通用性损失。

12.3.5. 并列关系

这里讨论的主要词组类型可以与 and、or、but 等**连词(conjunctions)**连用，形成同一类型的更大结构。例如，一个**并列(coordinate)**名词短语可以由两个由连词分隔的其他名词短语组成:

Please repeat [$_{NP}$ [$_{NP}$ the flights] *and* [$_{NP}$ the costs]]

I need to know [$_{NP}$ [$_{NP}$ the aircraft] *and* [$_{NP}$ the flight number]]

以下是允许这些结构的规则:

$$NP \rightarrow NP \text{ and } NP$$

注意，通过连词形成协调短语的能力经常被用作对成分的测试。考虑下面的例子，它们不同于上面的例子，因为它们没有第二个限定词。

Please repeat the [$_{Nom}$ [$_{Nom}$ flights] *and* [$_{Nom}$ costs]]

I need to know the [$_{Nom}$ [$_{Nom}$ aircraft] *and* [$_{Nom}$ flight number]]

这些短语可以连用的事实证明了我们一直在使用的潜在的名词性成分的存在。这里有一条规则:

$$\text{Nominal} \rightarrow \text{Nominal and Nominal}$$

下面的例子说明了 VPs 和 Ss 的连词:

What flights do you have [$_{VP}$ [$_{VP}$ leaving Denver] and [$_{VP}$ arriving in San Francisco]]

[$_{S}$ [$_{S}$ I'm interested in a flight from Dallas to Washington] and [$_{S}$ I'm also interested in going to Baltimore]]

VP 和 S 的连词规则与上面的 NP 规则是一致的。

$$\text{VP} \rightarrow \text{VP and VP}$$

$$\text{S} \rightarrow \text{S and S}$$

由于所有主要的短语类型都可以以这种方式连接，它也可以更普遍地表示这个连接事实;许多语法形式主义，如 GPSG (Gazdar 等人,1985 年)使用如下的**元规则(metarules)**来实现这一点:

$$X \rightarrow X \text{ and } X$$

这个元规则声明任何非终端符都可以与同一个非终端符连接，以产生相同类型的成分;变量 X 必须被指定为一个代表任何非终端符的变量，而不是非终端符本身。

12.4. 树库

由上下文无关的语法规则组成的足够健壮的语法可以用于将解析树分配给任何句子。这意味着可以构建一个语料库，其中每个句子都与相应的解析树配对。这样的句法注释语料库称为树库(treebank)。正如我们在第十三章中讨论的那样，树库在句法分析以及语言学对句法现象的研究中扮演着重要的角色。

已经创建了各种各样的树库，通常是通过使用解析器(在接下来的几章中描述的那种)自动解析每个句子，然后使用人类(语言学家)手动更正解析。**宾州树库(Penn Treebank)**项目(我们在第 8 章中介绍了它的 POS 标记集)已经从 Brown、Switchboard、ATIS 和华尔街日报的英语语汇中生成了树库，以及阿拉伯语和汉语的树库。许多树库使用我们将在第 14 章中介绍的依存表示，包括许多**通用依存**项目(Nivre 等人，2016b)的一部分。

12.4.1. 例子：宾州树库项目

图 12.7 显示了来自宾州树库 Brown 和 ATIS 部分的句子。注意词类标记的格式差异;这种微小的差异是常见的，在处理树库时必须加以处理。第八章定义了宾州树库词类标记集。对树使用 LISP 风格的圆括号表示法非常常见，类似于我们在(12.1)中看到的方括号表示法。对于那些不熟悉它的人，我们在图 12.8 中展示了一个标准的节点和线条的树表示。

图 12.9 显示了来自《华尔街日报》的树。这棵树显示了宾州树库的另一个特性:使用跟踪(trace)(-NONE-表示节点)来标记长距离依存关系或句法位移(syntactic movement)。例如，引语经常跟在引语动词后面，比如 say。但在这个例子中，“We would have to wait until we have collected on those assets”的引语出现在 he said 之前。空的 S 只包含节点-NONE-标记在 said 之后引语句子经常出现的位置。这个空节点(在

树库 II 和 III 中)被标记为索引 2，就像句子开头的引语 S 一样。这样的相互索引可能使一些解析器更容易恢复这样一个事实，即这个前面的或主题化的引语就是动词 said 的补语。类似的-NONE-节点表示在动词 to wait 之前没有句法主语;相反，主语是更前面的名词短语(NP)We。同样，它们都与索引 1 相互索引。

宾州树库 II 和树库 III 发布了新增的进一步的信息，以便更容易地恢复谓词和论元之间的关系。某些短语用标记(tag)来作标注(mark)，表明短语的语法功能(例如，表层主语、逻辑主题、强调、非 VP 谓语)，它在特定文本类别(标题、题目)中的存在，以及它的语义功能(时间短语、方位)(Marcus 等人 1994, Bies 等人 1995)。图 12.9 显示了-SBJ(表层主语)和-TMP(时间短语)标记的示例。图 12.8 还显示了-PRD 标记，它用于非 VPs 的谓语(图 12.8 中的谓语是一个 ADJP)。在第 14 章讨论依存语法和解析时，我们将回到语法功能的主题。

```
((S
  (NP-SBJ (DT That)
    (JJ cold) (, ,)
    (JJ empty) (NN sky) )
  (VP (VBD was)
    (ADJP-PRD (JJ full)
      (PP (IN of)
        (NP (NN fire)
          (CC and)
          (NN light) ))))
  (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

图 12-7：从 LDC 树库 3 版本解析出的句子

图注：图(a)中的句子来自 Brown 语料库，图(b)中的句子来自 ATIS 语料库。

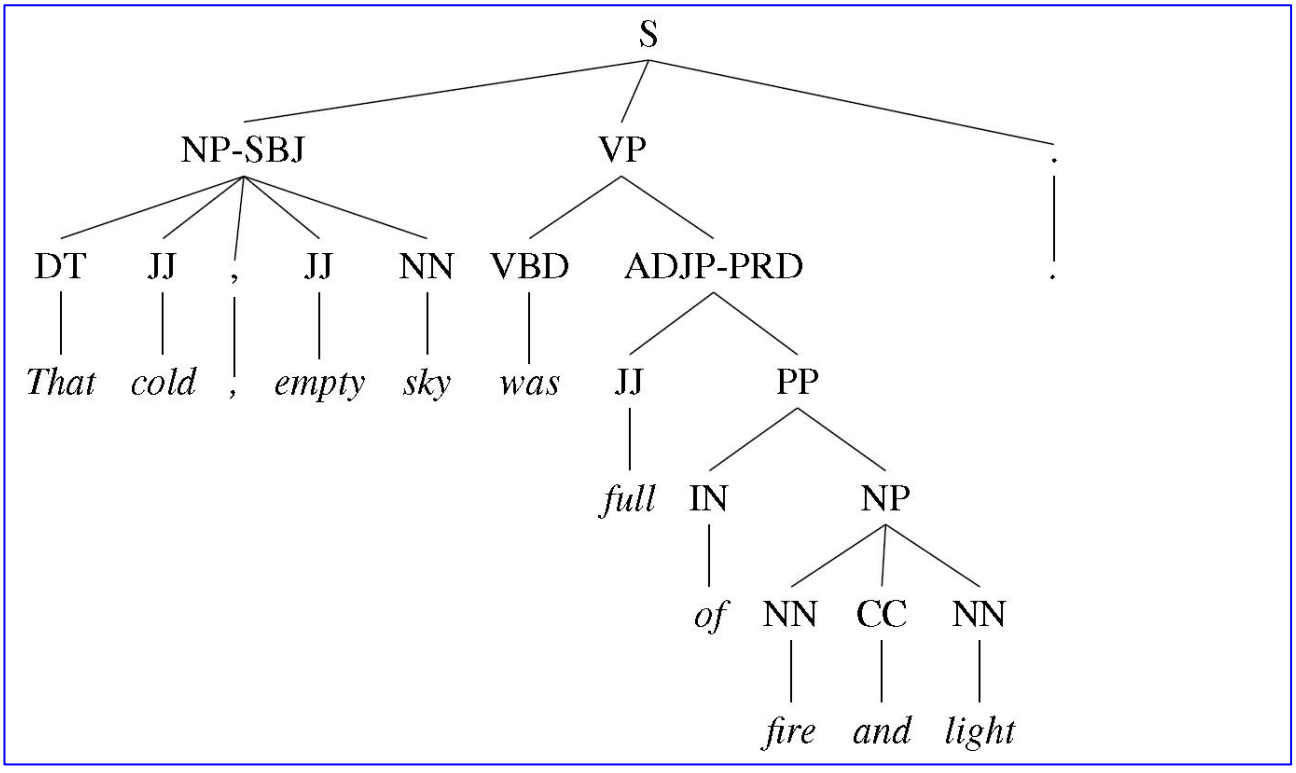


图 12-8：Brown 语料库句子对应的树形图

12.4.2. 树库作为语法

树库中的句子隐含地构成被注释的语料库所代表的语言的语法。例如，从图 12.7 和图 12.9 中三个已解析的句子中，我们可以提取出其中的每一条 CFG 规则。为简单起见，让我们去掉规则后缀(-SBJ 等)，结果生成的语法如图 12.10 所示。

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

图 12-9：来自 LDC 宾州树库的一句话

图注：摘自 LDC 宾州树库《华尔街日报》部分的一句话。请注意空的-NONE-节点的使用方法。

图注：从图 12.7 和图 12.9 中的三个树库句子中提取 CFG 语法规则和词法条目的示例。

用于解析宾州树库的语法相对平坦，从而产生非常多且非常长的规则。例如，在扩展 VPs 的大约 4500 条不同的规则中，有一些单独的规则适用于任意长度的 PP 序列和各种可能的动词论元排列:

VP → VBD PP

VP → VBD PP PP

VP → VBD PP PP PP

VP → VBD PP PP PP PP

VP → VB ADVP PP

VP → VB PP ADVP

VP → ADVP VB PP

以及更长的规则，比如：

VP → VBP PP PP PP PP PP ADVP PP

它来自于用斜体标记的 VP:

This mostly happens because we *go from football in the fall to lifting in the winter to football again in the spring*.

Grammar	Lexicon
S→NP VP .	PRP→we | he
S→NP VP	DT→the | that | those
S→“ S ” , NP VP .	JJ→cold | empty | full
S→-NONE-	NN→sky | fire | light | flight | tomorrow
NP→DT NN	NNS→assets
NP→DT NNS	CC→and
NP→NN CC NN	IN→of | at | until | on
NP→CD RB	CD→eleven
NP→DT JJ , JJ NN	RB→a.m.
NP→PRP	VB→arrive | have | wait
NP→-NONE-	VBD→was | said
VP→MD VP	VBP→have
VP→VBD ADJP	VBN→collected
VP→VBD S	MD→should | would
VP→VBN PP	TO→to
VP→VB S	
VP→VB SBAR	
VP→VBP VP	
VP→VBN PP	
VP→TO VP	
SBAR→IN S	
ADJP→JJ PP	
PP→IN NP	

图 12-10：从三个树库句子中提取 CFG 语法规则的示例

成千上万的 NP 规则包括：

NP → DT JJ NN

NP → DT JJ NNS

NP → DT JJ NN NN

NP → DT JJ JJ NN

NP → DT JJ CD NNS

NP → RB DT JJ NN NN

NP → RB DT JJ JJ NNS

NP → DT JJ JJ NNP NNS

NP → DT NNP NNP NNP NNP JJ NN

NP → DT JJ NNP CC JJ JJ NN NNS

NP → RB DT JJS NN NN SBAR

NP → DT VBG JJ NNP NNP CC NNP

NP → DT JJ NNS , NNS CC NN NNS NN

NP → DT JJ JJ VBG NN NNP NNP FW NNP

NP → NP JJ , JJ “SBAR” NNS

例如，最后两条规则来自以下两个名词短语:

[$_{DT}$ The] [$_{JJ}$ state-owned] [$_{JJ}$ industrial] [$_{VBG}$ holding] [$_{NN}$ company] [$_{NNP}$ Instituto] [$_{NNP}$ Nacional] [$_{FW}$ de] [$_{NNP}$ Industria] [$_{NP}$ Shearson’s] [$_{JJ}$ easy-to-fifilm], [$_{JJ}$ black-and-white] “[$_{SBAR}$ Where We Stand]” [$_{NNS}$ commercials]

从这种角度来看，宾州树库 III 的《华尔街日报》语料库是一个大型语法，它包含大约 100 万个单词，也有大约 100 万个非词汇规则**符记(token)**，由大约 17500 个不同的规则**类型(type)**组成。

关于树库语法的各种事实，比如大量的平面规则，给概率解析算法带来了问题。因此，对从树库中提取的语法进行各种修改是很常见的。我们在附录 C 中进一步讨论这些问题。

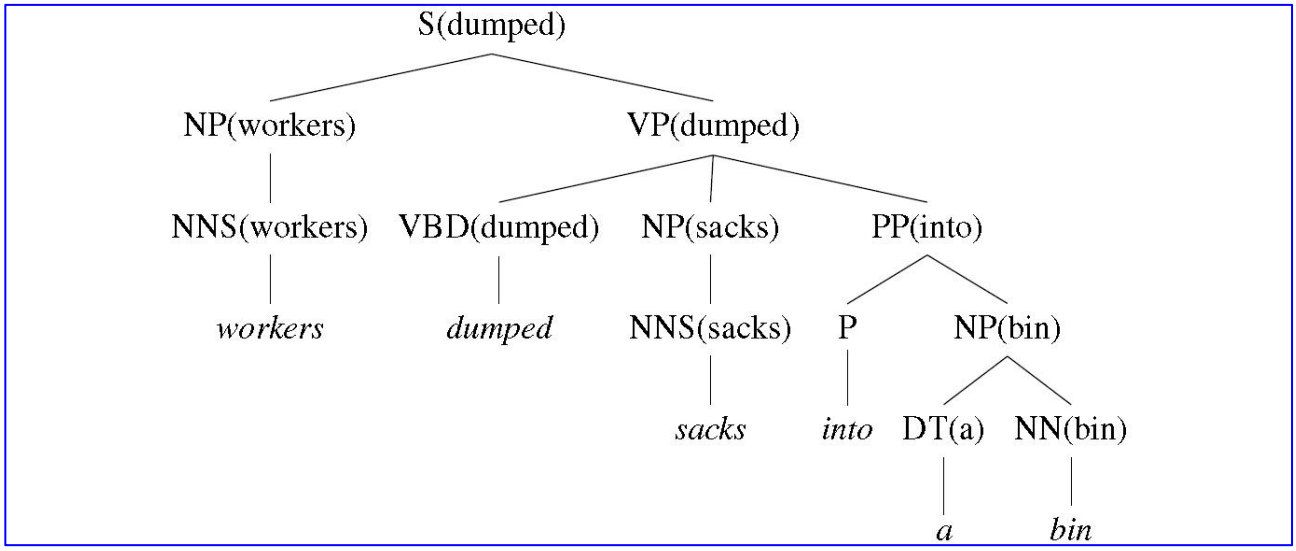


图 12-11：柯林斯(1999)的词汇树

12.4.3. 中心词及其发现

我们先前非正式地提出，语法成分可以与词汇**中心词(head)**相关联;N 是 NP 的中心词，V 是 VP 的中心词。每个成分都有一个中心词的想法可以追溯到 Bloomfield(1914 年)，这是我们将在第 14 章介绍的依存语法和依存解析的核心。中心词在概率分析(附录 C)和基于成分的语法形式中也很重要，如中心词驱动的短语结构语法(Pollard 和 Sag,1994)。

在词汇中心词的一个简单模型中，每个上下文无关的规则都与中心词相关联(Charniak 1997, Collins 1999)。短语的中心词是语法上最重要的单词。中心词被传递到解析树中;因此，解析树中的每个非终端符都用单个单词进行注释，这就是它的词汇中心词。图 12.11 显示了 Collins(1999)的这样一个树的例子，其中每个非终端符都用其中心词进行了注释。

为了生成这样的树，必须对每个 CFG 规则进行扩展，以标识右侧的一个成分作为中心词的孩子(child)。然后，节点的中心词被设置为它的中心词子节点的中心词。对于教科书上的例子来说，选择这些中心词的孩子是很简单的(NN 是 NP 的中心词)，但对于大多数短语来说是很复杂的，而且确实存在争议。(该补语或该动词应是不定式动词短语的中心词吗?)现代语言学的句法理论通常包括一个组件来定义中心词(Pollard 和 Sag, 1994))。

在大多数实际的计算系统中，都有另一种寻找中心词的方法。不是在语法本身中指定中心词规则，而是在树的上下文中为特定的句子动态识别中心词。换句话说，一旦解析了一个句子，就会遍历生成的树，用适当的中心词装饰每个节点。目前的大多数系统都依赖于一套简单的手写规则，例如一个实用的宾州树库语法，它在 Collins(1999)中给出，但最初是由 Magerman(1995)开发的。例如，找到 NP 中心词的规则如下(Collins, 1999, p. 238):

- If the last word is tagged POS, return last-word.
- Else search from right to left for the first child which is an NN, NNP, NNPS, NX, POS, or JJR.
- Else search from left to right for the first child which is an NP.
- Else search from right to left for the first child which is a $, ADJP, or PRN.
- Else search from right to left for the first child which is a CD.
- Else search from right to left for the first child which is a JJ, JJS, RB or QP.
- Else return the last word

从该集合中选取的其他规则如图 12.12 所示。例如 VP 规则为 VP→$Y_1 \cdots Y_n$，算法从 $Y_1 \cdots Y_n$ 左边开始寻找第一个类型为 TO 的 Y_i；如果没有找到 TO，则搜索 VBD 类型的第一个 Y_i；如果没有找到 VBN，它将搜索 VBN，以此类推。更多细节见 Collins(1999)。

Parent	Direction	Priority List
ADJP	Left	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	Right	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
PRN	Left	
PRT	Right	RP
QP	Left	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
S	Left	TO IN VP S SBAR ADJP UCP NP
SBAR	Left	WHNP WHPP WHADVP WHADJP IN DT S SQ SINV SBAR FRAG
VP	Left	TO VBD VBN MD VBZ VB VBG VBP VP ADJP NN NNS NP

图 12-12：Collins(1999)的一些中心词规则

图注：中心词规则也称为中心词渗透表(percolation table)。

12.5. 语法等价与乔姆斯基范式(CNF)

正式语言被定义为(可能是无限的)单词的字符串集合。这表明，我们可以通过询问两种语法是否生成相同的字符串集合来求(ask)它们是否等价。事实上，两个不同的上下文无关的语法可能生成同一种语言。

我们通常要区分两种语法等价:**弱等价(weak equivalence)**和**强等价(strong equivalence)**。如果两种语法生成相同的字符串集，并且为每个句子分配相同的短语结构(只允许重命名非终端符号)，那么这两种语法是强等价的；如果它们生成相同的字符串集合，但不为每个句子分配相同的短语结构，那么两个语法是弱等价的。

对于语法而言，拥有一个**范式(normal form)**通常是很有用的，范式中每个产生式都采用特定形式。例如，如果上下文无关语法是ϵ自由(ϵ-free)的，并且如果另外每个产生式均为 A→B C 或 A→a，则其形式为**乔姆斯基范式(Chomsky normal form，CNF)**(Chomsky,1963)。也就是说，每个规则的右侧要么具有两个非终端符，要么具有一个终端符。乔姆斯基范式语法是**二叉的(binary branching)**，即它们具有二叉树(向下到前词汇节点)。我们将在第 13 章的 CKY 解析算法中使用此二叉属性。

任何上下文无关的语法都可以转换为弱等价的乔姆斯基范式语法。例如,如下形式的规则：

$$A \rightarrow B\ C\ D$$

可以转换成以下两个 CNF 规则(练习 12.8 要求读者写出完整的算法):

$$A \rightarrow B\ X$$

$$X \rightarrow C\ D$$

有时使用二进制分支实际上可以产生更小的语法。例如，句子可能被描述为：

VP → VBD NP PP*

在宾州树库中由以下一系列规则表示:

VP → VBD NP PP

VP → VBD NP PP PP

VP → VBD NP PP PP PP

VP → VBD NP PP PP PP PP

...

但也可以由以下两个规则语法生成:

VP → VBD NP PP

VP → VP PP

一个生成符号 A 具有一个潜在的无限序列的符号 B，其规则是 A→A B，这种生成被称为**乔姆斯基邻接规则（Chomsky-adjunction）**。

12.6. 词汇语法

到目前为止提出的语法方法强调短语结构规则，同时使词汇的作用最小化。但是，正如我们在关于一

致性、次范畴化和长距离依赖项的讨论中所看到的那样，这种方法充其量带来了繁琐的解决方案，产生了多余、难以管理且脆弱的语法。为了克服这些问题，已经开发出了许多替代方法，它们都具有更好地利用词汇的共同主题。在更多与计算相关的方法中，包括词汇功能语法(LFG)(Bresnan，1982)，中心词驱动短语结构语法(HPSG)(Pollard 和 Sag，1994)，树连接语法(TAG)(Joshi，1985)，和组合范畴语法(CCG)。这些方法的不同之处在于它们的词汇化程度—它们依赖词汇而不是短语结构规则来捕捉语言事实的程度。

以下部分提供了 CCG 的介绍，这是一种由句法和语义考虑而激发的高度词汇化的方法，我们将在第 15 章中返回讨论。第 14 章讨论了依存语法，这是一种完全消除短语结构规则的方法。

12.6.1. 组合范畴语法(CCG)

在本节中，我们提供了范畴语法(Ajdukiewicz 1935, Bar-Hillel 1953)的概述：**组合范畴语法(combinational category grammar，CCG)** (Steedman 1996, Steedman 1989, Steedman 2000)。这是一种早期的词汇化语法模式，也是一种重要的现代扩展。

范畴方法由三个主要元素组成:一组**范畴**，一个将单词与范畴联系起来的**词表**，以及一组管理范畴如何在上下文中组合的**规则**。

范畴

范畴是原子元素或单一论元函数，当提供所需范畴作为论元时，返回论元作为值。更正式地说，我们可以定义 C，一组用于语法的范畴如下:

- $A \subseteq C$, where A is a given set of atomic elements
- $(X/Y), (X\backslash Y) \in C$, if $X, Y \in C$

这里显示的斜杠符号用于定义语法中的函数。它指定了期望论元的类型、期望找到它的方向以及结果的类型。因此，(X/Y)是一个函数，它于右侧寻找类型为 Y 的成分，并返回 X 的值;(X\Y)是相同的，除了它于左边寻求它的论元。

原子范畴的集合通常非常小，包括熟悉的元素，如句子和名词短语。功能范畴包括动词词组和复杂名词词组。

词表

词表由单词的范畴分配组成。这些分配可以是原子范畴，也可以是功能范畴，并且由于词汇模糊性，可以将单词分配给多个范畴。考虑以下示例词汇条目。

flight : *N*
Miami : *NP*
cancel : (*S\NP*)/*NP*

名词和专有名词(如 flight 和 Miami)被分配到原子范畴，反映了它们作为函数论元的典型作用。另一方面，像 cancel 这样的及物动词被赋予了范畴(S\NP)/NP:一个函数寻找它右边的 NP，并返回一个具有类型(S\NP)的函数作为它的值。反过来，这个函数可以与左边的 NP 组合，产生一个 S 作为结果。这捕获了第 12.3.4 节中讨论的次范畴化信息，但是这里的信息具有丰富的、计算上有用的内部结构。

像 give 这样的双及物动词，在动词后需要两个论元，就会有((S\NP)/NP)/NP:一个与 NP 结合的函数，产生另一个对应于及物动词(S\NP)/NP 范畴的函数，如上面给出的 cancel。

规则

范畴语法规则指定函数及其论元如何组合。以下两个规则模板构成了所有范畴语法的基础。

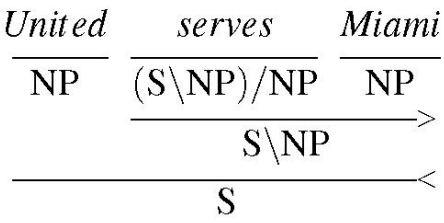
$$X/Y \quad Y \Rightarrow X \tag{12.4}$$

$$Y \quad X\backslash Y \Rightarrow X \tag{12.5}$$

第一个规则将一个函数应用到它右边的论元，而第二个规则从左边查找它的论元。我们将第一个称为前向函数应用程序，第二个称为后向函数应用程序。应用其中任何一个规则的结果都是指定为应用函数值

的类范畴。

给定这些规则和一个简单的词汇表，让我们来分析一下 United serves Miami 的句子。假设 serves 是一个具有范畴(S\NP)/NP 的及物动词，United 和 Miami 都是简单的 NP。同时使用前向和后向函数应用，推导过程如下:

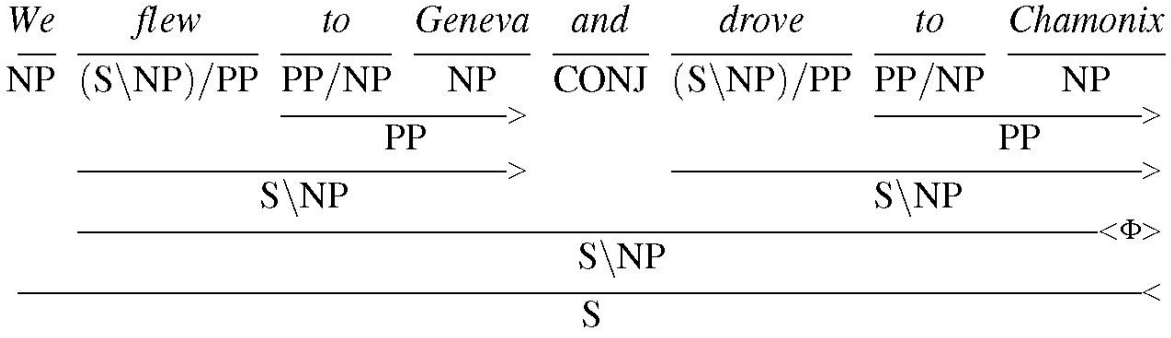


范畴语法的推动从单词开始，规则应用程序用横贯所涉及元素的水平线来说明，操作的类型在该行的右端表示。在此示例中，有两个函数应用程序：一个由>表示的前向函数应用程序，将动词 serves 应用到其右侧的 NP；一个由<表示的后向函数应用程序，将第一个结果应用于 NP United 在其左侧。

通过添加另一个规则，分类方法提供了一种直接的方式来实现前面在第 12.3.5 节描述的并列关系元规则。回想一下，英语允许同一类型的两个成分并列，从而产生同一类型的新成分。下面的规则提供了处理这类示例的机制。

$$X\ CONJ\ X \Rightarrow X \tag{12.6}$$

这条规则表明，当同一个范畴的两个成分被 CONJ 类型的成分分开时，它们可以组合成同一类型的单个更大的成分。下面的推导说明了该规则的使用。



通过连接操作符<Φ>将两个 S\NP 组合成一个较大的同类成分，再通过反向函数应用与主语 NP 组合。

这些例子说明了范畴语法方法的词汇性质。关于一种语言的语法事实很大程度上被编码在词表中，而语法规则被归结为三规则的集合。不幸的是，与传统的 CFG 规则相比，基本的分类方法并没有给我们更多的表达能力;它只是将信息从语法转移到词汇中。为了超越这些限制，CCG 可对函数进行操作。

第一对操作符是组合相邻函数。

$$X/Y\ Y/Z \Rightarrow X/Z \tag{12.7}$$

$$Y\backslash Z\ X\backslash Y \Rightarrow X\backslash Z \tag{12.8}$$

其中：第一个规则（12.7）称为**前向组合**(forward composition)规则，是一个在其右侧寻找类型为 Y 的参数的函数，可以应用于相邻的成分；第二个规则（12.8）称为**向后组合**(backward composition)规则，是一个提供 Y 作为结果的函数，该规则可以将这两个函数组合成一个函数，此函数的第一个成分为类型，第二个成分为参数。尽管这种表示有点笨拙，但第二条规则是一样的，除了我们要从左边而不是从右边查找相关的参数。CCG 图中的 B 表示这两种成分，并用"<"或">"表示方向。

第二对操作符是类型提升(type raising)。类型提升将简单的范畴提升到函数的地位。更具体地说，类型提升接受一个范畴，并将其转换为一个函数，该函数寻求一个以原始范畴作为论元的函数。下面的模式显示了两个版本的类型提升:一个用于右边的论元，一个用于左边的论元。

$$X \Rightarrow T/(T\backslash X) \tag{12.9}$$

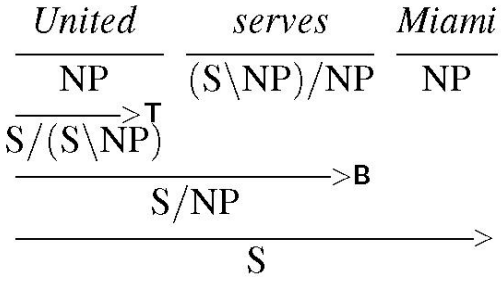
$$X \Rightarrow T\backslash(T/X) \tag{12.10}$$

这些规则中的范畴 T 可以对应于语法中已经存在的任何原子或函数范畴。

一个特别有用的类型提升示例将一个简单的主语位置的 NP 论元转换为一个可以与后面的 VP 组合的函数。为了了解这是如何运作的，让我们回顾一下之前 United serves Miami 的例子。与其将 United 分类为 NP，作为附加于 serve 的功能的一个论元，我们还不如使用类型提升，将其重新定义为一个函数，如下所示。

$$NP \Rightarrow S/(S\backslash NP)$$

将这个类型提升的成分与前向组合规则(12.7)相结合，可以实现以下替代前面推导的方法。



通过类型提升将 United 提升为 S/(S\NP)，我们可以将它与及物动词 serve 组合起来，以产生完成推导所需的(S/NP)函数。

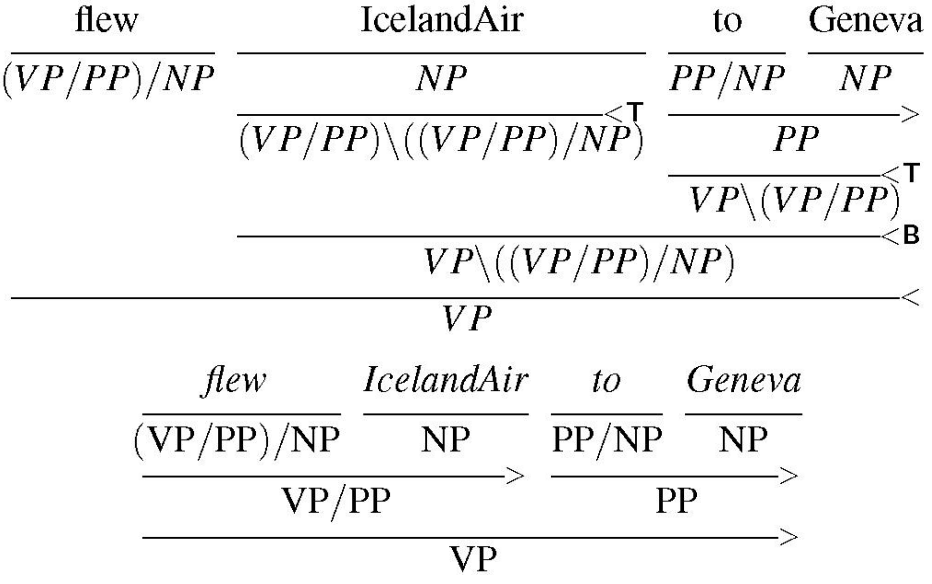
关于这个推导有几个有趣的地方需要注意。首先，它提供了从左到右逐字推导的方法，这更接近于反映人类处理语言的方式。这使得 CCG 成为一个特别适合于心理语言学研究的框架。第二，这一推导涉及到中间分析单元 United services 的使用，这与英语中的传统成分不相对应。这种使用**非成分(non-constituent)**的能力使 CCG 能够处理非适当成分的短语的并列关系，如下面的示例所示。

(12.11) We flew IcelandAir to Geneva and SwissAir to London.

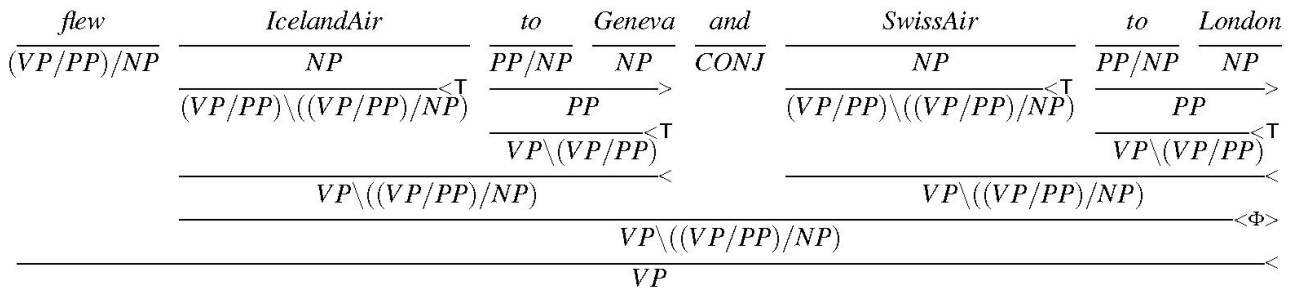
这里，正在并列的部分是 IcelandAir to Geneva 和 SwissAir to London，这些短语通常不被认为是成分，从动词短语 flew IcelandAir to Geneva 的标准推导中可以看出。

在这一推导中，没有单个成分对应于 IcelandAir to Geneva，因此没有机会使用<Φ>操作符。注意，复杂的 CCG 范畴可能会有点麻烦，因此我们将在本文和以下推导中使用 VP 作为(S\NP)的简写。

下面的替代推导通过使用后向类型提升(12.10)和后向函数组合(12.8)来提供所需的元素。



通过对 SwissAir to London 应用相同的分析，满足了<Φ>操作符的需求，产生了原始示例(12.11)的如下派生。

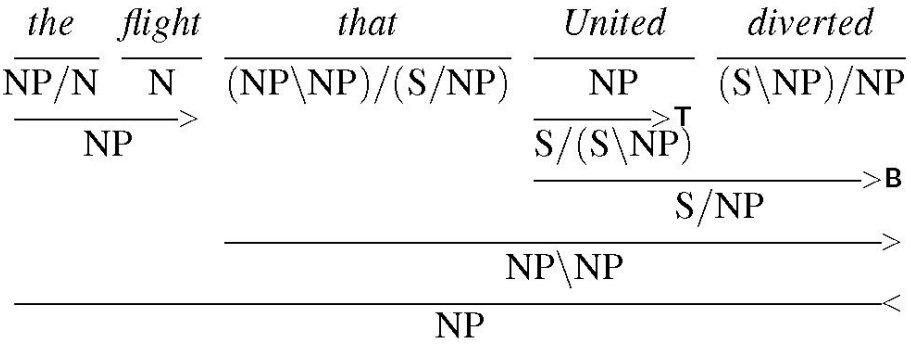


最后，让我们看看如何使用这些高级操作符来处理长距离依存关系(也称为句法位移或提取)。正如第 12.3.1 节所提到的，远距离依存关系产生于许多英语结构，包括 wh-疑问句、关系从句和话题化(topicalization)。这些结构的共同之处是出现在与其通常的或预期的位置不同的地方的成分。以下面的关系从句为例。

the flight that United diverted

其中，divert 是一个及物动词，它有两个 NP 论元，主语 NP 在左，直接宾语 NP 在右；因此它的范畴是(S\NP)/NP。然而，在这个例子中，直接宾语 the flight 已经被“移动”到从句的开头，而主语 United 保持在它的正常位置。我们需要的是一种方法来合并主语论元，同时处理 the flight 不是在其预期位置的事实。

下面的推导实现了这一点，同样是通过结合使用类型提升和函数组合。



正如我们在前面的例子中所看到的，这个推导的第一步是将 United 的类型提升到 S/(S\NP)范畴，允许它把 diverted 与前向组合联合起来。这种组合的结果是 S/NP，它保留了一个事实，即我们仍然在寻找一个 NP 来填补缺失的直接对象。第二个关键部分是词汇范畴被分配给单词 that：(NP\NP)/(S/NP)。此函数查找缺少其右侧论元的动词短语，并将其转换为 NP，以查找左侧缺少元素的 NP，恰好是我们找到 the flight 的位置。

CCG 树库

与短语结构方法一样，树库在基于 CCG 的解析方法中也起着重要作用。CCGBank(Hockenmaier 和 Steedman，2007 年)是最大、使用最广泛的 CCG 树库。它是通过基于规则的方法自动翻译宾州树库中的短语结构树而创建的。该方法成功翻译了宾州树库中超过 99%的树，从而产生 48,934 个句子与 CCG 派生词配对。它还提供了 1200 多个范畴的 44000 个单词的词汇。附录 C 将讨论如何使用这些资源来训练 CCG 解析器。

12.7. 总结

本章通过使用**上下文无关语法**介绍了一些句法上的基本概念。

•在许多语言中，连续单词的集合充当一个组或**成分**，可以用**上下文无关语法**(也称为**短语结构语法**)来建模。

•上下文无关的语法由一组(groups)**规则**或**产生式**组成，用一组**非终端**符和一组**终端**符来表达。形式上，特定的上下文无关语言是可以从特定的上下文无关语法**派生**的字符串集合。

•**生成式语法**是语言学中形式语言的传统名称，用于建模自然语言的语法。

•英语中有许多句子级语法结构;**陈述式、命令式、是非疑问式和 wh-疑问式**是四种常见的类型;这些可以用上下文无关的规则来建模。

•一个英语名词短语的**中心名词**前面可以有**限定词、数字、数量修饰语**和**形容词短语**，中心名词后面可以有一些**后修饰语**;**动名词**和**不定式** VP 是常见的可能性。

•英语中**主语**在人称和数方面与主谓动词**一致**。

•动词可以根据其期望的**补语**类型**次范畴化**(再分类)。简单的次范畴有**及物性**和**不及物性**;大多数语法包括比这些更多的类范畴。

•**树库**的分析句子存在于许多类型的英语和许多语言。可以用树搜索工具搜索树库。

•任何上下文无关的语法都可以转换为**乔姆斯基范式（CNF）**，即每个规则的右边都有两个非终端符或一个终端符。

•词汇语法更加重视词表的结构，减轻了纯短语结构规则的负担。

•**组合范畴语法(CCG)**是一种重要的与计算相关的词汇化方法。

12.8. 文献和历史说明

根据 Percival(1976)的说法，将开创性的句子分解为成分的层次结构的思想出现在开创性的心理学家威廉·温特(Wiltt Wundt)的沃克心理学中(Wundt，1900)：

> *...den sprachlichen Ausdruck für die willkürliche Gliederung einer Gesammtvorstellung in ihre in logische Beziehung zueinander gesetzten Bestandteile*
>
> (将一个整体思想任意划分为相互之间具有逻辑关系的成分的语言表达)

Leonard Bloomfield 在其早期著作《语言研究导论》(Bloomfield，1914)中将 Wundt 的成分概念纳入了语言学。到他后来的著作《语言》(Bloomfield，1933)出版时，当时称为“**直接成分分析（immediate-constituent analysis，ICA）**”的语法已成为美国公认的句法研究方法。相比之下，传统的欧洲语法可以追溯到古典时期，它定义了单词之间的关系，而不是成分之间的关系，欧洲句法学家则将重点放在这种依存语法上，这是第 14 章的主题。

美国结构主义在寻找“发现程序”的过程中看到了直接成分的许多具体定义，这是一种描述语言句法的方法论算法。通常，这些尝试试图抓住这样的直觉，即“直接成分的主要标准是等级(degree)，其中把组合行为（combinations behave）当作简单的单元”(Bazell 1966，第 284 页)。在最具体的定义中，最著名的是哈里斯(Harris)的关于与单个单元的分布相似性的想法，并带有可替代性测试。本质上，该方法通过尝试用简单的结构替换可能的成分来将结构分解为成分，如果一个简单的形式(例如人)的替换可以替换为更复杂的集合的结构(例如紧张的年轻人)，那么紧张的年轻人很可能是一个成分。哈里斯的测试是直觉的开始，即成分是一种等价类。

这种层级成分思想的第一个形式化是 Chomsky(1956)中定义的**短语结构语法**，并在 Chomsky(1957)和 Chomsky(1975)中进一步扩展(并反对)。从那时起，大多数生成式语言学理论至少部分地基于上下文无

关语法或它们的概括(例如，中心词驱动短语结构语法(Pollard 和 Sag，1994)，词汇功能语法(Bresnan，1982) ，极简主义计划(Chomsky，1995 年)和建构语法(Kay 和 Fillmore，1999 年)，等等)； 这些理论中有许多使用了称为 **X-bar 图式(x-bar schemata)**的上下文无关逻辑示意图模板，该模板也依赖于句法中心词的概念。

在乔姆斯基的最初工作后不久，上下文无关语法被 Backus(1959)和 Naur 等人(1960)在他们对 ALGOL 编程语言的描述中独立地重新发明；Backus(1996)指出他受到了 Emil Post 作品的影响，Naur 的作品独立于他(Backus)自己的作品。在这些早期工作之后，大量的自然语言处理的计算模型都是基于上下文无关的语法，因为早期开发了高效的算法来解析这些语法(见第 13 章)。

有各种各样的 CFGs 扩展类，其中许多用于处理句法中的长距离依存关系。(其他语法将远程依赖项视为语义相关，而不是语法相关(Kay 和 Fillmore 1999, Culicover 和 Jackendoff 2005))。

一种扩展的形式主义是树邻接语法(TAG)(Joshi，1985 年)。TAG 的主要数据结构是树，而不是规则。树有两种：初始树和辅助树。例如，初始树可能表示简单的句子结构，而辅助树则将递归添加到树中。树由替换和附加两种操作组合而成。附加操作处理长距离依赖性。有关更多详细信息，请参见 Joshi(1985)。树邻接语法是上下文敏感语言家族的成员。

我们在第 12.4 节提到了另一种处理远程依存关系的方法，这种方法基于使用空类别(empty categories)和共索引(co-indexing)。宾州树库使用了这个模型，它(在各种 Treebank 语库中)借鉴了扩展标准理论和极简主义(Radford, 1997)。

对英语语法感兴趣的读者应该获得三大参考英语语法之一:Huddleston 和 Pullum (2002)， Biber 等人(1999)， Quirk 等人(1985)。

有许多从不同角度介绍句法的好的入门教科书。Sag 等人(2003)从**生成式**的角度对句法进行了介绍，重点研究了中心词驱动型短语结构语法中短语结构规则、统一和类型层次的使用。Van Valin, Jr.和 La Polla(1997)从**功能式**角度介绍了跨语言数据和句法结构的功能动机。

12.9. 练习

12.1 Draw tree structures for the following ATIS phrases:

1. Dallas
2. from Denver
3. after five p.m.
4. arriving in Washington
5. early flights
6. all redeye flights
7. on Thursday
8. a one-way fare
9. any delays in Denver

12.2 Draw tree structures for the following ATIS sentences:

1. Does American Airlines have a flight between five a.m. and six a.m.?
2. I would like to fly on American Airlines.
3. Please repeat that.
4. Does American 487 have a first-class section?
5. I need to fly between Philadelphia and Atlanta.
6. What is the fare from Atlanta to Denver?
7. Is there an American Airlines flight from Philadelphia to Dallas?

12.3 Assume a grammar that has many *VP* rules for different subcategorizations, as expressed in Section 12.3.4, and differently subcategorized verb rules like *Verb-with-NP-complement*. How would the rule for ostnominal relative clauses (12.4) need to be modified if we wanted to deal properly with examples like *the earliest flight that you have*? Recall that in such examples the pronoun *that* is the object of the verb *get*. Your rules should allow this noun phrase but should correctly rule out the ungrammatical *S *I get*.

12.4 Does your solution to the previous problem correctly model the NP *the earliest flight that I can get*? How about *the earliest flight that I think my mother wants me to book for her*? Hint: this phenomenon is called **long-distance dependency**.

12.5 Write rules expressing the verbal subcategory of English auxiliaries; for example, you might have a rule *verb-with-bare-stem-VP-complement → can*.

12.6 *NP*s like *Fortune ' s office* or *my uncle ' s marks* are called **possessive** or **genitive** noun phrases. We can model possessive noun phrases by treating the sub-NP like *Fortune ' s* or *my uncle ' s* as a determiner of the following head noun. Write grammar rules for English possessives. You may treat *' s* as if it were a separate word (i.e., as if there were always a space before *' s*).

12.7 Page 238 discussed the need for a *Wh-NP* constituent. The simplest *Wh-NP* is one of the *Wh-pronouns* (*who, whom, whose, which*). The Wh-words *what* and *which* can be determiners: *which four will you have?*, *what credit do you have with the Duke?* Write rules for the different types of *Wh-NP*s.

12.8 Write an algorithm for converting an arbitrary context-free grammar into Chomsky normal form.

13. 成分解析

One morning I shot an elephant in my pajamas.
How he got into my pajamas I don't know.

一天早晨，我穿着睡衣射杀了一头大象。
我不知道他是怎么进入我的睡衣的。

Groucho Marx, *Animal Crackers*, 1930

句法分析是将句法结构指定给句子的任务。本章重点关注成分结构，即由第 12 章中描述的那种上下文无关的语法分配的结构。在下一章中，我们将介绍依存解析，这是一种可选的解析结构。

可以在诸如语法检查等应用程序中使用解析树:不能解析的句子可能有语法错误(或至少难以阅读)。

解析树可以是语义分析表示的中间阶段(我们将在第 16 章中介绍)，因此在**问答**等应用程序中发挥作用。例如，要回答以下问题：

Which flights to Denver depart before the Seattle flight?

我们将需要知道发问者想要一个飞往丹佛的航班清单，而不是飞往西雅图的航班，并且解析结构(知道 to Denver 修饰 flights，以及 which flights to Denver 是 depart 的主语)可以帮助我们。

我们首先讨论歧义及其存在的问题，然后给出 Cocke-Kasami-Younger(CKY)算法(Kasami 1965，Younger 1967)，这是一种用于语法分析的标准动态规划方法。我们已经看到了其他动态规划算法，例如最小编辑距离(第 2 章)和 Viterbi(第 8 章)。

原始 CKY 算法返回一个句子的一组解析树的有效表示，但没有告诉我们哪一棵解析树是正确的。为此，我们需要使用每种可能成分的分数来增加 CKY。我们将看到如何使用基于神经跨度的解析器来做到这一点。并且，我们将介绍其他方法，例如用于 CCG 解析的超级标记，部分解析方法(用于需要对输入进行表层语法分析就足够的情况)以及用于评估解析器准确性的标准度量标准集。

13.1. 歧义

歧义是句法分析器面临的最严重的问题。第八章介绍了**词类歧义**和**词类消歧**的概念。这里，我们引入了一种新的歧义类型，称为**结构歧义(structural ambiguity)**，用一个新的玩具(toy)语法 $\mathcal{L}_1$ 来说明，如图 13.1 所示，它向上一章的 $\mathcal{L}_0$ 语法添加了一些规则。

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

图 13-1：$\mathcal{L}_1$ 微型英语语法和词汇

当语法可以为一个句子分配多个解析时，就会出现结构歧义。格劳乔·马克思(Groucho Marx)在《动物

饼干》(Animal Crackers)中扮演船长斯波尔丁(Captain Spaulding)时的那句名言是模棱两可的，因为 in my pajamas 中的短语既可以是以 elephant 为中心词的 NP 短语的一部分，也可以是以 shot 为中心词的动词短语的一部分。图 13.2 用 $\mathcal{L}_1$ 中的规则说明了这两种对格劳乔·马克思路线的分析。

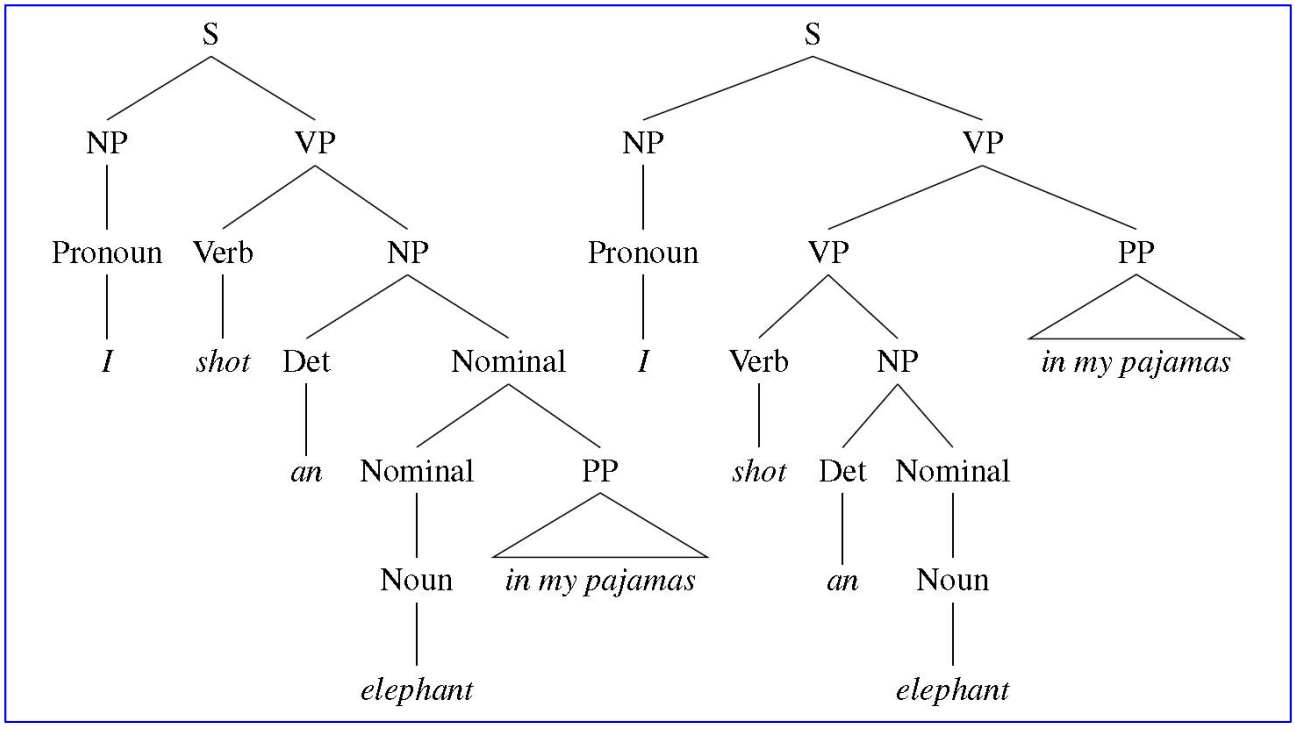


图 13-2：歧义句的两棵解析树

图注：左边的解析对应的是幽默的解读“大象穿着睡衣”，右边的解析对应的是斯波尔丁上尉“穿着睡衣开枪”。

很恰当地说，结构歧义(appropriately enough)具有多种形式。两种常见的歧义是**附着歧义(attachment ambiguity)**和**并列连接歧义(coordination ambiguity)**。如果一个特定的成分可以在一个以上的位置附加到解析树上，则句子具有附着歧义。格劳乔·马克思句子是 PP-附着歧义的一个例子。各种状语短语也受这种歧义的影响。例如，在以下示例中，flying to Paris 的动名词 VP 可以是主语为 the Eiffel Tower 的动名词从句的一部分，也可以是修饰以 saw 为中心词的 VP 的附加语(adjunct)：

(13.1) We saw the Eiffel Tower flying to Paris.

在并列连接歧义中，短语可以通过诸如 and 的连词连接起来。例如，词组 old men and women 可以被括为[old [men and women]]，指的是老男人和老女人，或[old men]和[women]，在这种情况下，只有男人是老的。这些歧义在真实的句子中以复杂的方式结合在一起，比如下面这个来自布朗语料库的新闻句子:

(13.2) President Kennedy today pushed aside other White House business to devote all his time and attention to working on the Berlin crisis address he will deliver tomorrow night to the American people over nationwide television and radio.

这个句子有一些歧义，尽管它们在语义上是不合理的，需要仔细阅读才能看到它们。最后一个名词短语可以被解析为[nationwide [television and radio]]或[[nationwide television] and radio]。“pushed aside”的直接宾语应该是“other White House business”，但也可以是一个奇怪的短语(“other White House business to devote all his time and attention to working”)(这个结构诸如 Kennedy affirmed [his intention to propose a new budget to address the deficit])。那么，这个短语 on the Berlin crisis address he will deliver tomorrow night to the American people 可能是对 pushed 这个动词的附加修饰词。一种类似于 over nationwide television and radio 的 PP 可以连接到任何更高的 VPs 或 NPs(例如，它可以修饰 people 或者 night)。

对于自然出现的句子，有许多语法正确但语义不合理的解析器，这是一个影响所有解析器的令人讨厌的问题。幸运的是，下面的 CKY 算法旨在有效地处理结构歧义。在下一节中我们将看到，我们可以使用神经方法来增强 CKY，通过**句法消歧(syntactic disambiguation)**来选择一个正确的解析。

13.2. CKY 解析：一种动态规划方法

动态规划为解决语法中的歧义问题提供了一个强大的框架。回想一下，动态规划方法系统地填充子问题的解决方案表。完整的表包含解决整个问题所需的所有子问题的解决方案。在语法解析的情况下，这些子问题表示输入中检测到的所有成分的解析树。

动态规划的优势来自于语法规则的上下文无关性质—一旦在输入的一个片段中发现了一个成分，我们就可以记录它的存在，并使它可以在任何可能需要它的后续派生中使用。这提供了时间和存储效率，因为子树可以在表中查找，而不是重新分析。本节介绍 CKY 算法，这是最广泛使用的基于动态规划的解析方法。**图表解析(chart parsing)**(Kaplan 1973, Kay 1982)是一种相关的方法，动态规划方法通常被称为图表解析方法。

13.2.1. CNF 转换

CKY 算法要求语法首先采用乔姆斯基范式(CNF)。回想一下第 12 章，CNF 中的语法被限制为 A→B C 或 A→w 形式的规则。也就是说，每个规则的右边必须扩展为两个非终端符或单个终端符。将语法限制为 CNF 并不会导致表达性的任何损失，因为任何上下文无关的语法都可以转换为相应的 CNF 语法，该语法接受与原始语法完全相同的一组字符串。

让我们从将一个通用的 CFG 转换为用 CNF 表示的 CFG 的过程开始。假设我们正在处理一个ϵ-free 语法，那么在任何通用语法中都需要解决三种情况：①右边混合了终端和非终端的规则；②右边只有一个非终端的规则；③右边长度大于 2 的规则。

①对于混合终端和非终端的规则的补救措施是简单地引入一种新的虚拟非终端，只覆盖原始终端。例如，不定式动词短语 INF-VP→to VP 的规则将被 INF-VP→TO VP 和 TO→to 这两条规则所取代。

②右边有一个单独的非终端符的规则称为**单位产生式(unit productions)**。我们可以消除单位产生式规则，方法是用最终导致的所有非单位产生式规则的右侧，来重写原始规则的右侧。更正式地讲，如果 $A \overset{*}{\Rightarrow} B$ 由一个或多个单位产生式的链组成，并且 $B \rightarrow \gamma$ 是我们语法中的非单位产生式，则我们对语法中的每个这样的规则加 $A \rightarrow \gamma$ 并丢弃所有中间单位产生式。正如我们在玩具语法中所展示的那样，这可能会导致语法的实质性**扁平化(flattening)**，并最终导致生成的树中的终端达到相当高的水平。

③右边大于 2 的规则通过引入新的非终端符来归一化，这些非终端符将较长的序列分散到几个新规则上。正式地说，如果我们有这样的规则

$$A \rightarrow B\ C\ \gamma$$

我们用一个新的非终端符替换了最左边的一对非终端符，并引入了一个新的产生式，结果产生了以下新规则:

$$A \rightarrow X1\ \gamma$$
$$X1 \rightarrow B\ C$$

在右侧较长的情况下，我们简单地迭代这个过程，直到原来触及的(offending)的那个规则被长度为 2 的规则所取代。替换最左边的一对非终端符完全是任意的;任何产生二元规则的系统方案都足够了。

在我们现有的语法中，规则 S→Aux NP VP 将被 S→X1 VP 和 X1→ Aux NP 这两个规则所取代。整个转换过程可以总结如下:

1.将所有符合的规则复制到新语法中。

2.将规则中的终端转换为虚拟非终端。

3.转换单位产生式。

4.将所有规则变为二元规则并将它们添加到新语法中。

$\mathcal{L}_1$ Grammar	$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

图 13-3：$\mathcal{L}_1$ 语法及其到 CNF 的转换

图注：请注意，虽然这里没有显示它们，但是所有来自 $\mathcal{L}_1$ 的原始词汇条目也没有变化。

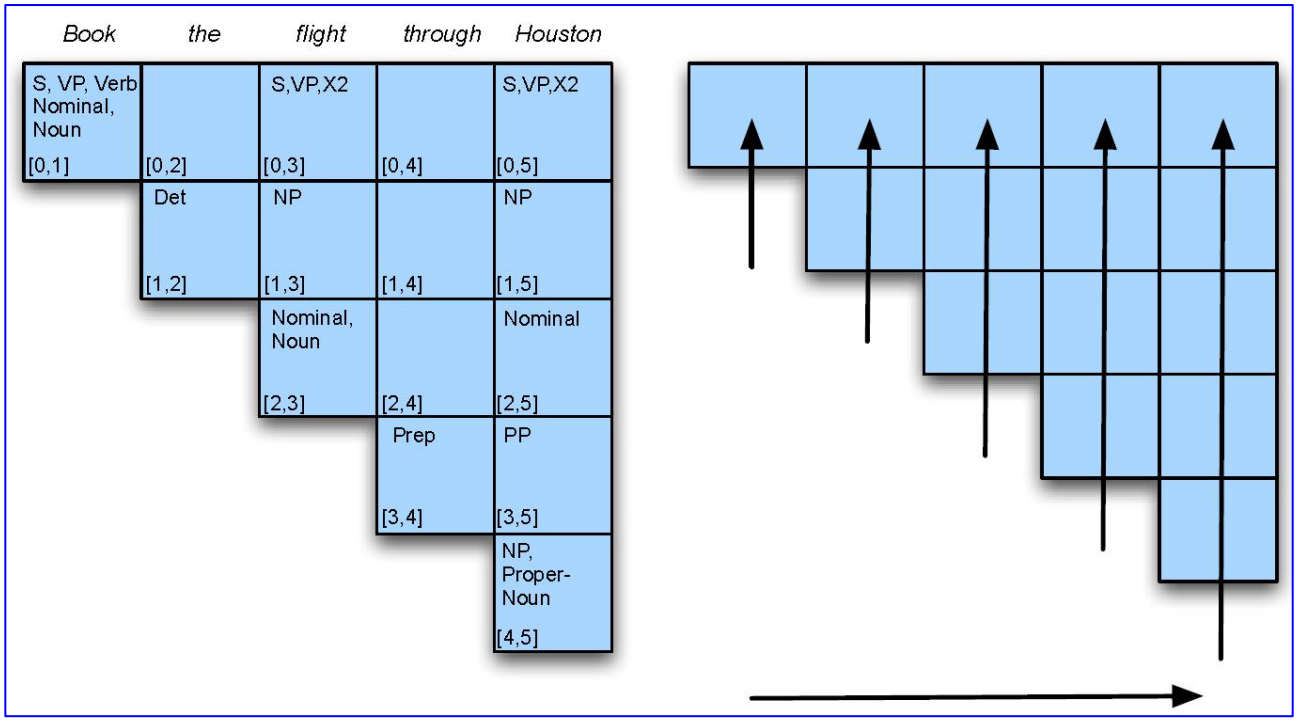


图 13-4：句子 Book the flight through Houston 的完整的解析表

图 13.3 显示了将整个转换过程应用到第 13.1 节介绍的 $\mathcal{L}_1$ 语法的结果。注意，这个图没有显示原始的

词汇规则;由于这些原始的词汇规则已经在 CNF 中，它们都原封不动地保留到新的语法中。然而，图 13.3 显示了各个不同的地方，在这些地方，消除单位产生式的过程，实际上创建了新的词汇规则。例如，在转换后的语法中，所有原来的动词都被提升为 VP 和 S。

13.2.2. CKY 识别

现在我们的语法是在 CNF 中，解析树中词类级别之上的每个非终端节点将有两个子节点。一个二维矩阵可以用来编码整个树的结构。对于长度为 n 的句子，我们将使用(n+1)×(n+1)矩阵的上三角部分。这个矩阵中的每一个单元[i, j]都包含一个非终端点集合，这些非终端点代表了所有成分输入的位置 i 到 j 的成分。由于我们的索引方案从 0 开始，很自然地将索引视为指向输入单词之间的空隙的指针(例如 $_0$ Book $_1$ that $_2$ flight $_3$)。这些间隙通常称为**围栏(fencepost)**，比喻为栅栏段之间的柱子。因此，表示整个输入的单元格位于矩阵中的位置[0,n]。

由于我们表中的每个非终端项在解析中都有两个孩子节点，因此对于由项[i，j]表示的每个成分，输入中必须有一个位置 k，这个 k 可以将[i，j]拆分为两个部分，使得 i<k <j。给定这样的位置 k，第一个成分[i,k]必须沿着第 i 行并处于项[i,j]的左侧某处，而第二个成分[k,j]则应沿第 j 列并位于其下方。为了更加具体，请考虑以下示例及其完整的解析矩阵，如图 13.4 所示。

(13.3) Book the flight through Houston.

矩阵中的超对角线行(superdiagonal row)包含输入中每个单词的词类。该超对角线上方的随后的(subsequent)对角线包含的成分将覆盖输入中所有长度增加的跨度。

```
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
            table[j − 1, j] ← table[j − 1, j] ∪ A
    for i ← from j − 2 down to 0 do
        for k ← i + 1 to j − 1 do
            for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
                    table[i,j] ← table[i,j] ∪ A
```

图 13-5：CKY 算法

在这种设置下，CKY 识别包括以正确的方式填充解析表。为此，我们将以自底向上的方式进行，以便在填充任何单元格[i, j]时，该单元包含可能有贡献于此条目(即左侧的单元格和下方的单元格)的部分，而且该单元已经被填充了。图 13.5 给出的算法每次填充上三角矩阵一列，从左到右，每列从下到上填充，如图 13.4 右侧所示。该方案保证在每个时间点我们都有所需的所有信息(在左边，因为左边的所有列都已经填满了；在下面，因为我们正在从下到上填满)。它也反映了在线处理，因为从左到右填充列对应着一次处理一个单词。

图 13.5 给出的算法的最外层循环在一个列上迭代，第二个循环在一个行上迭代，剖析从下到上进行。最内层循环的目的是覆盖所有位置，在这些位置上，输入信息中从 i 到 j 的子字符串可能被一分为二。当 k 横跨在可拆分字符串的位置时，我们认为成对的单元格将沿第 i 行向右移动，沿第 j 列向下移动。

图 13.6 说明了填充单元[i，j]的一般情况。在每次这样的拆分时，算法都会考虑是否可以按照语法中的规则认可的方式组合两个单元格的内容。如果存在这样的规则，则将其左侧的非终端符输入到表中。

图 13.7 显示了在读取 Houston 之后，表的第 5 列中的 5 个单元格是如何填充的。箭头指出了用于向表中添加条目的两个跨度(span)。注意，在单元[0,5]这种操作表明，对于这个输入，有三个可选的解析：一个是 PP 修饰 flight；一个是 PP 修饰 booking；一个是捕获到第二个论元(该论元在原始的 VP→Verb NP PP 规则中，但是该论元现在用 VP→X2 PP 规则间接地被捕获了)。

13.2.3. CKY 解析

图 13.5 中给出的算法是识别器，而不是解析器。为了使其成功，只需在单元[0，n]中找到一个 S。要将其转换为能够返回给定输入的所有可能分析的解析器，我们可以对算法进行两个简单的更改：第一个更改是增加表中的条目，以使每个非终端符都与指向其来源的表条目的指针配对(或多或少，如图 13.7 所示)；第二个更改是允许将同一非终端的多个版本输入到表中(再次如图 13.7 所示)。进行了这些更改后，完成的表将包含给定输入的所有可能的解析。返回的任意单个的解析包含从单元格[0，n]中选择的一个 S，然后从表中递归地检索其组成成分。

返回一个句子的每个解析可能没有用，因为解析的数目可能是指数的。在下一节中，我们将介绍如何仅检索最佳解析。

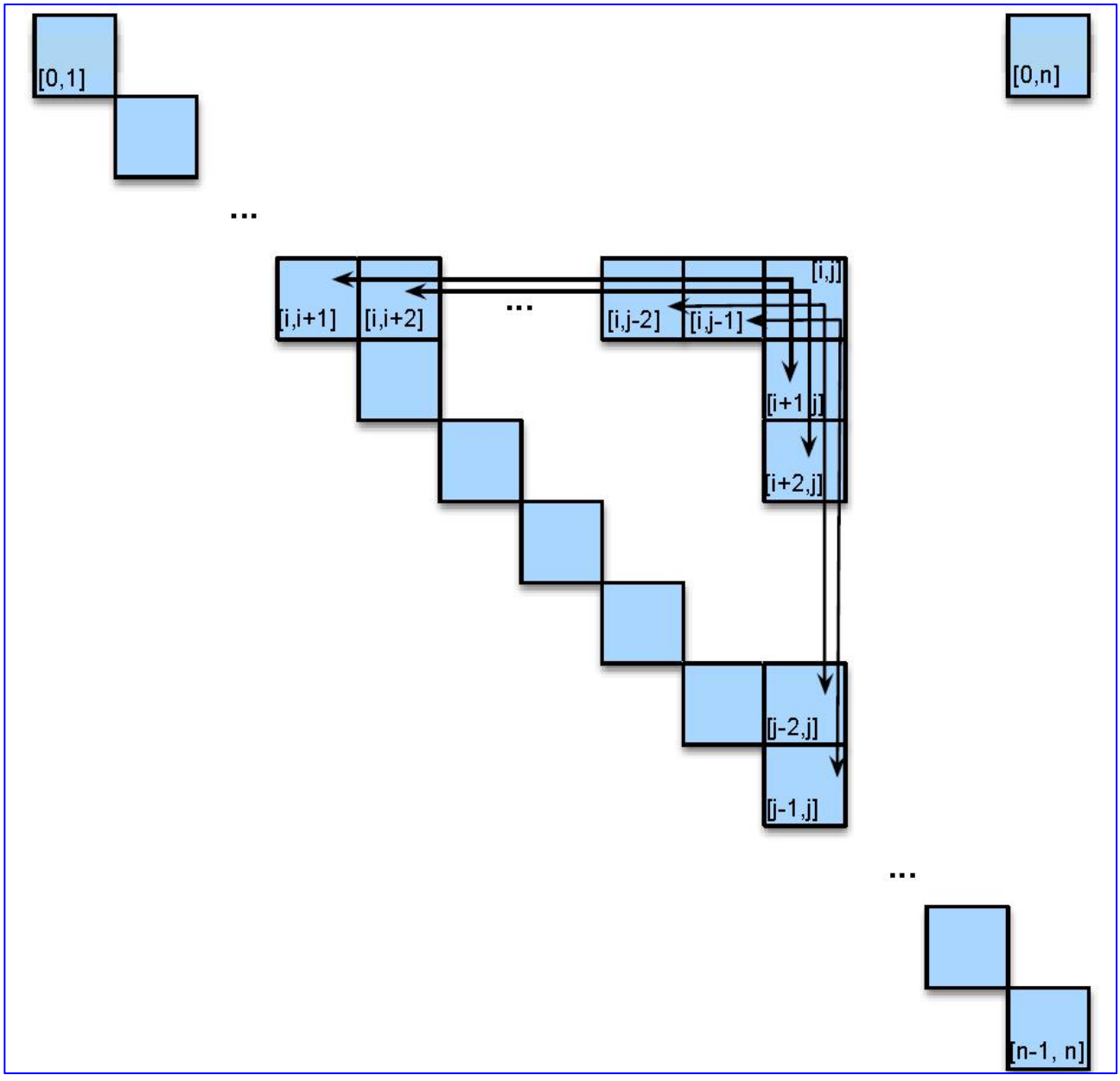


图 13-6：填充 CKY 表中第[i, j]个单元格的所有方法

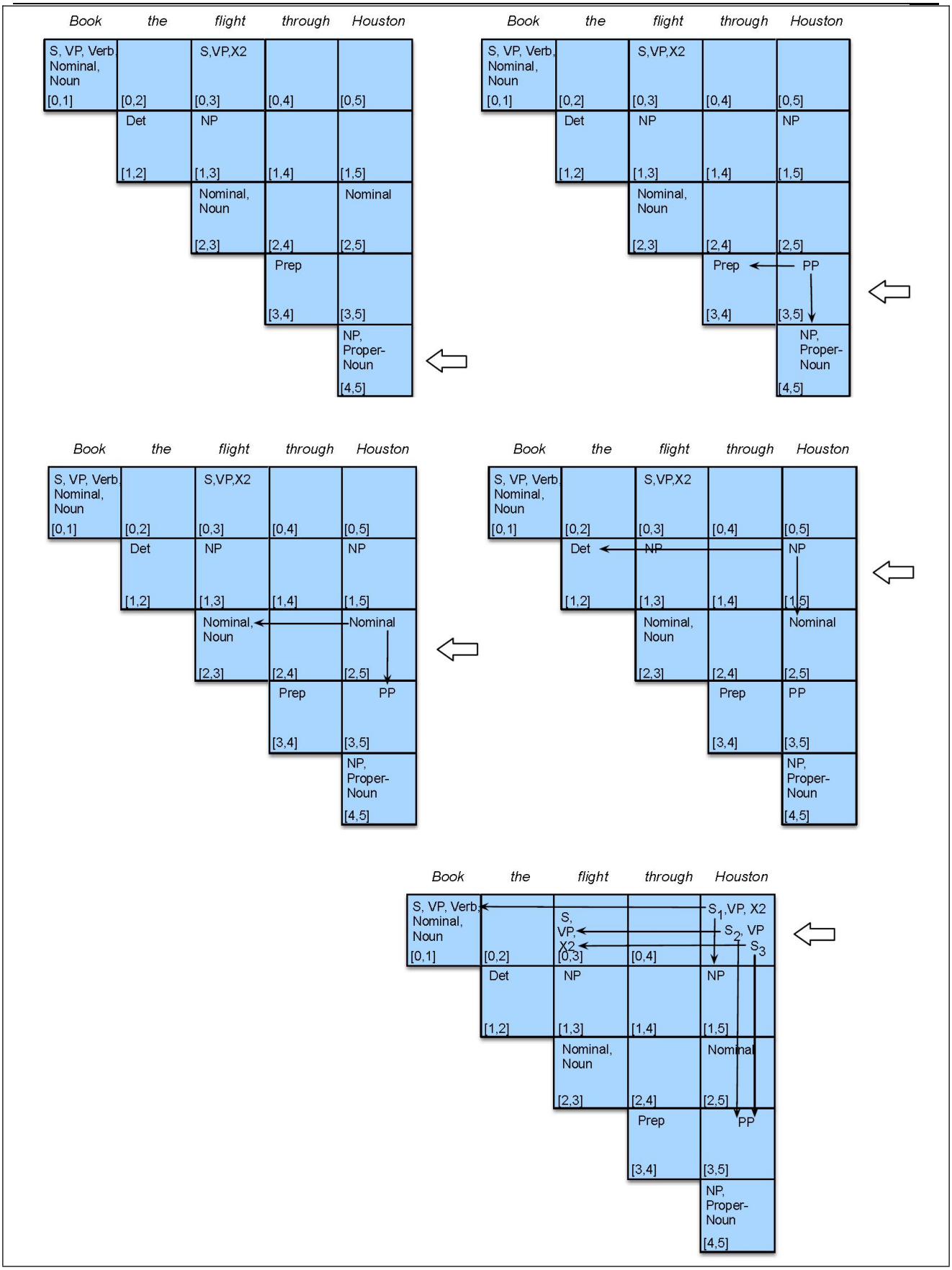


图 13-7：读取 Houston 单词后填充第 5 列的单元格

13.2.4. CKY 实践

最后，我们应该注意到，虽然对 CNF 的限制在理论上并不构成问题，但在实践中却带来了一些重要的问题。显然，就目前的情况来看，我们的解析器返回的树，与友善的句法人员提供给我们的树，在语法上不一致。除了让语法开发人员不高兴之外，转换到 CNF 将使任何句法驱动的语义分析方法复杂化。

解决这些问题的一种方法是保留足够的信息，将树转换回原始语法，作为解析的后处理步骤。对于长度大于 2 的规则所使用的转换来说，这是微不足道的。简单地删除新的伪非终端并提升它们的子节点，就可以恢复原来的树。

对于单位产生式，事实证明，更改基本 CKY 算法以直接处理它们，比存储恢复正确树所需的信息要方便得多。练习 13.3 要求您进行更改。附录 C 介绍的许多概率解析器都使用以这种方式更改的 CKY 算法。

13.3. 基于跨度的神经成分解析

虽然到目前为止我们看到的 CKY 解析算法在枚举一个句子的所有可能解析树方面做得很好，但它有一个很大的问题:它没有告诉我们哪个解析是正确的!也就是说，它不会在可能的解析中消除歧义。为了解决消歧问题，我们将使用 CKY 算法的一个简单的神经扩展。这种解析算法，通常称为**基于跨度的成分解析(span-based constituency parsing)**，或**神经 CKY**，它的直觉是训练神经分类器为每个成分分配分数，然后使用 CKY 的修改版本组合这些成分分数，以找到得分最高的解析树。这里我们将描述 Kitaev 等人(2019)的算法版本。

13.3.1. 计算跨度分数

让我们首先考虑一下位于围栏位置 i 和 j 之间(带有非终端符号标签 l)的成分(我们将其称为**跨度(span)**)。我们将构建一个分类器，为这个成分跨度分配一个分数 s(i,j,l)。图 13.8 概述了该体系结构。

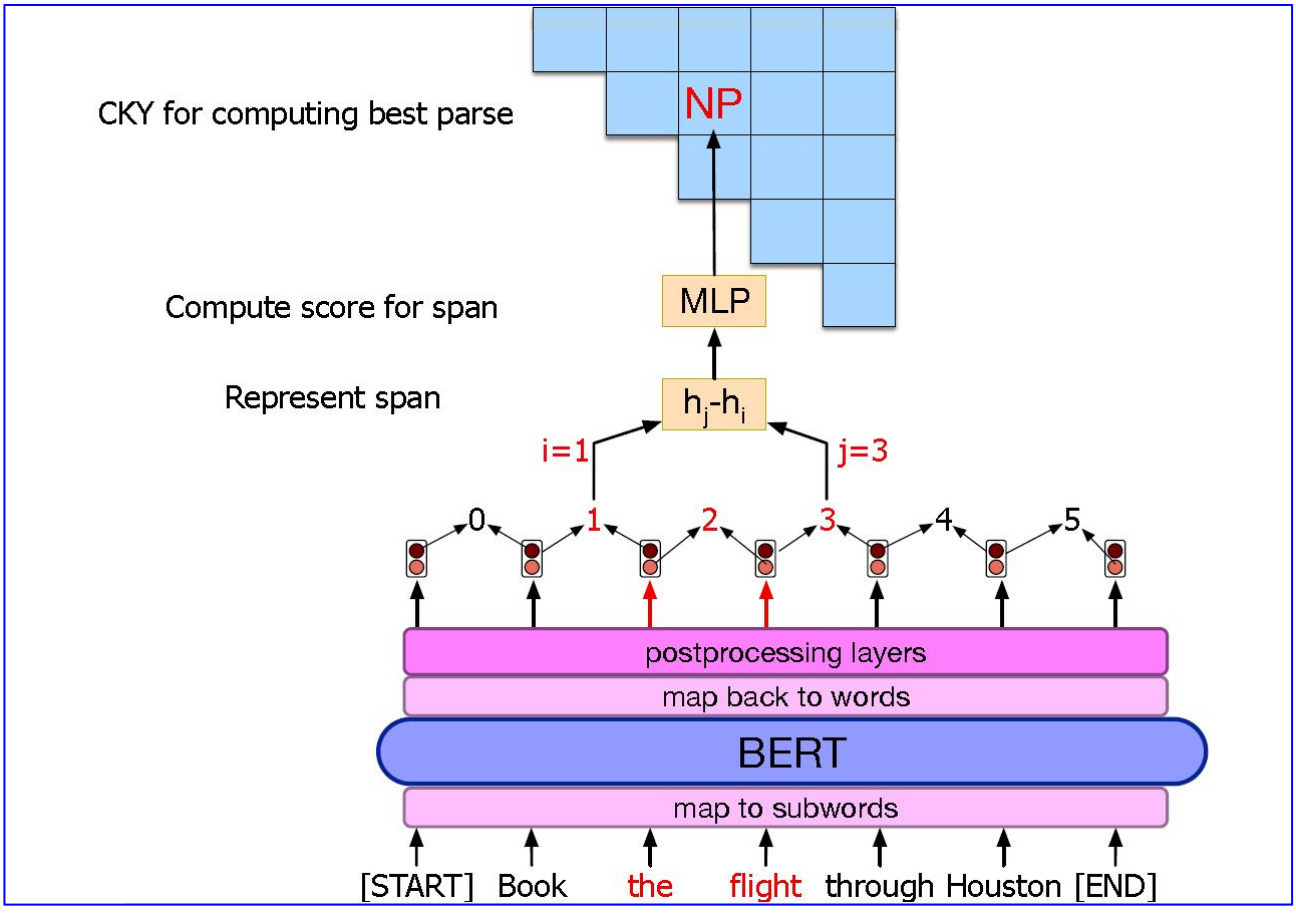
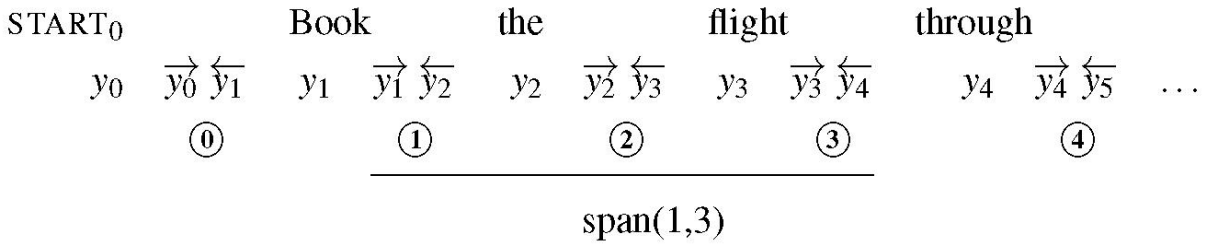


图 13-8：计算“the flight”跨度分数的简化大纲

通过将输入的单词符记传递给像 BERT 这样的预训练语言模型来嵌入它们。由于 BERT 是在子词(词片 wordpiece)符记而不是单词的级别上运行的，因此我们首先需要将 BERT 的输出转换为单词表示形式。一种标准的执行方法是简单地使用最后一个子词单元作为单词的表示形式(使用第一个子词单元似乎效果相当好)。然后可以将嵌入穿过某些后处理层；例如，Kitaev 等人(2019)使用 8 层 Transformer。

然后将所得的单词编码器输出的结果 y_t 用于计算跨度得分。首先，我们必须将单词编码(由单词位置索引)映射到跨度编码(由围栏杆索引)。为此，我们用两个单独的值表示每个围栏杆。直觉是单词右侧的跨度端点表示与单词左侧的跨度端点不同的信息。通过将 y_t 分为两半，我们将每个单词输出 y_t 转换为在此围栏杆结束的跨度值 $\overrightarrow{y}_t$，并为在此围栏杆开始的跨度的转换为跨度值 $\overleftarrow{y}_t$。然后，每一个跨度从一个双向量围栏杆延伸到另一个，如下面的 the flight 表示为跨度(1,3):



一个传统的表示跨度的方法，最初是为基于 RNN 的模型开发的(Wang 和 Chang，2016)，但也扩展到了 Transformers，它采用了开始和结束嵌入之间的差异，即表示跨度(i,j)通过从 j 的嵌入中减去 i 的嵌入。在这里，我们通过拼接每个围栏组件的差异来表示跨度：

$$v(i,j) = [\overrightarrow{y_j} - \overrightarrow{y_i} \; ; \; \overleftarrow{y_{j+1}} - \overleftarrow{y_{i+1}}] \tag{13.4}$$

然后将跨度向量 v 通过一个 MLP 跨度分类器，该分类器有两个全连接层和一个 ReLU 激活函数，其输出维数为可能的非终端标签数:

$$s(i, j, \bullet) = W_2 \, \text{ReLU}(\text{LayerNorm}(W_1 v(i,j))) \tag{13.5}$$

然后 MLP 为每个可能的非终端输出一个分数。

13.3.2. 集成跨度分数到解析中

现在，我们对每个标签的成分跨度 s(i,j,l)都有一个分数。但是我们需要为整个解析树评分。形式上，树 T 表示为|T|这样的标签集合。第 t^{th} 个跨度从位置 i_t 开始，到位置 j_t 结束，标签为 l_t:

$$T = \{(i_t, j_t, l_t) : t = 1, : : : , |T|\} \tag{13.6}$$

因此，一旦我们对每个跨度都有了一个分数，解析器就可以通过对其成分跨度的分数求和来计算整个树 s(T)的分数:

$$s(T) = \sum_{(i,j,l) \in T} s(i,j,l) \tag{13.7}$$

我们可以选择最终的解析树作为得分最高的树:

$$\hat{T} = \underset{T}{\text{argmax}} \, s(T) \tag{13.8}$$

产生最有可能的解析的最简单方法是为每个跨度贪心地选择得分最高的标签。这个贪心的方法不能保证产生一棵树，因为一个跨度的最佳标签可能不适合一个完整的树。然而，在实践中，贪心的方法往往会找到树;Gaddy 等人(2018)在他们的实验中发现，95%的预测包围(predicted bracketings)形成有效的树。

尽管如此，更常见的做法是使用 CKY 算法的变体来找到完整的解析。Gaddy 等人(2018)定义的变量如下。让我们将 $s_{best}(i,j)$定义为最佳子树跨度(i,j)的分数。对于长度为 1 的跨度，我们选择最佳标签：

$$s_{\text{best}}(i, i+1) = \max_{l} s(i, i+1, l) \tag{13.9}$$

对于其他跨度(i, j)，递归为:

$$s_{\text{best}}(i,j) = \max_{l} s(i,j,l) + \max_{k}[s_{\text{best}}(i,k) + s_{\text{best}}(k,j)] \tag{13.10}$$

对更详细的关于基于跨度的句法分析，包括基于边界的训练算法，请参见 Stern 等人(2017)、Gaddy 等人(2018)、Kitaev 和 Klein(2018)、Kitaev 等人(2019)。

13.4. 评估解析器

有一个标准的工具可以用来评估为一个句子分配单个解析树的解析器，这个工具就是**解析值(PARSEVAL)**指标(Black 等人，1991)。PARSEVAL 指标用来衡量两种成分的相像程度，这两种成分是：**假设解析**，手工标签的**参考解析**。因此 PARSEVAL 要求测试集中的每个句子都有一个人工标签的参考(或“黄金标准”)解析，我们通常从诸如宾州树库中提取这些参考解。

如果在参考解析 C_r 中的一个成分具有相同的起点、终点和非终端符，那么在一个句子 s 的假设解析 C_h 中的一个成分将被标记为正确的。然后我们可以测量精度和召回率，就像我们已经看到的任务(如命名实体标记):

$$\textbf{labeled recall: } = \frac{\textbf{\# of correct constituents in hypothesis parse of } s}{\textbf{\# of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision: } = \frac{\textbf{\# of correct constituents in hypothesis parse of } s}{\textbf{\# of total constituents in hypothesis parse of } s}$$

像往常一样，我们经常报道两者的结合，F_1:

$$F_1 = \frac{2PR}{P+R} \tag{13.11}$$

我们还为每个句子 s 使用了一个新的度量，即交叉括号数：

交叉括号数(cross-brackets):参考解析具有括号诸如((A B) C)、而假设解析具有括号诸如(A(B C))的成分的数量。

为了比较使用不同语法的解析器，PARSEVAL 指标包括一个规范化算法，用于删除可能是语法特定的信息(辅助，前不定式“to”等)并计算简化分数(Black 等人，1991)。PARSEVAL 指标的规范实现称为 **evalb**(Sekine 和 Collins，1997)。

13.5. 部分解析

许多语言处理任务不需要所有输入的复杂、完整的解析树。对于这些任务，输入句子的**部分解析(partial parse)**或**浅解析(shallow parse)**可能就足够了。例如，信息提取系统通常不会从文本中提取所有可能的信息:它们只是对文本中可能包含有价值信息的部分进行识别和分类。

部分解析的一种被称为**组块分析(chunking)**。组块是对句子中扁平的、不重叠的片段进行识别和分类的过程，这些片段构成了基本的非递归短语，这些短语对应于主要内容词的词类：名词短语、动词短语、形容词短语和介词短语。在一篇文章中找到所有的基本名词短语的任务是特别常见的。由于组块文本缺乏层次结构，在给定的例子中，一个简单的括号标记法就足以表示组块的位置和类型:

(13.12) [NP The morning flight] [PP from] [NP Denver] [VP has arrived.]

这种括号标记法明确了组块中涉及的两个基本任务:分段(寻找组块的不重叠区域)和标签(为发现的组块分配正确的标记)。有些输入词可能不是任何组块的一部分，特别是在基本 NP 这样的任务中:

(13.13) [NP The morning flight] from [NP Denver] has arrived.

构成语法基本短语的内容取决于应用程序(以及短语是否来自树库)。然而，大多数系统都遵循一些标准准则。首先也是最重要的是，给定类型的基本短语不会递归地包含相同类型的任何成分。消除这种递归留给我们的问题是确定非递归短语的边界。在大多数方法中，基本短语包括短语的标题词，以及成分内的任何中心词之前的材料，而至关重要的是排除任何中心词之后的材料。消除中心词之后的修饰符也就排除

了解决附加歧义的需要。这种排斥确实导致了某些奇怪的现象，比如 PP 和 VP 通常都是由中心词组成的。因此，从 a flight from Indianapolis to Houston 将减少到如下:

(13.14) [NP a flight] [PP from] [NP Indianapolis][PP to][NP Houston]

组块算法通常是通过监督学习来完成的，从带注释的训练数据中训练 BIO 序列标记器，我们在第 8 章中见过。回想一下，在 BIO 标记中，我们有一个标记用于每个组块类型的开始(B)和内部(I)，还有一个符记用于任何组块的外部(O)。下面的例子显示了(13.12)的括号标记法被重新框定为标记任务:

(13.15) *The morning flight from Denver has arrived*
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

带有基本 NP 标记的相同句子说明了 O 标记的作用。

(13.16) *The morning flight from Denver has arrived.*
B_NP I_NP I_NP O B_NP O O

由于注释工作既昂贵又耗时，组块通常依赖于现有的树库，如宾州树库，从句子的完整解析成分中提取语法短语，找到合适的中心词，然后包括中心词左侧的材料，忽略右侧的文本。这有点容易出错，因为它依赖于第 12 章中描述的中心词发现规则的准确度。

给定一个训练集，任何序列模型都可以用来进行组块：CRF、RNN、Transformer 等。与对命名实体标记符的评估一样，组块的评估通过将组块输出与人工注释器提供的黄金标准答案进行比较，使用精度、召回率和 F1。

13.6. CCG 解析

CCG 之类的词汇化语法框架提出了一些问题，而我们一直在讨论的基于短语的方法并不是特别适合。为了快速回顾，CCG 由三个主要部分组成：一组**范畴**，将单词与范畴相关联的**词表**，以及一组用于控制范畴如何在上下文中组合的**规则**。范畴可以是诸如 S 和 NP 之类的原子元素，也可以是诸如(S\NP)/NP 之类的用于指定及物动词范畴的函数。规则指定函数、论元和其他函数的组合方式。例如，以下规则模板(前向和后向函数应用程序)指定函数将函数应用于其论元的方式。

$$X/Y\ Y \Rightarrow X$$
$$Y\ X\backslash Y \Rightarrow X$$

第一个规则将一个函数应用到它右边的论元，而第二个规则从左边查找它的论元。应用其中任何一个规则的结果都是指定为应用函数值的范畴。出于讨论的目的，我们将依赖这两个规则，以及第 12 章中描述的前向和后向组合规则和类型提升。

13.6.1. CCG 中的歧义

在解析中，处理歧义是成功的 CCG 解析的关键。由于大量复杂的词汇范畴与语法规则的普遍性相结合，导致了歧义，从而导致了 CCG 句法分析的困难。为了了解在范畴框架中出现的歧义，我们可以考虑以下例子。

(13.17) United diverted the flight to Reno.

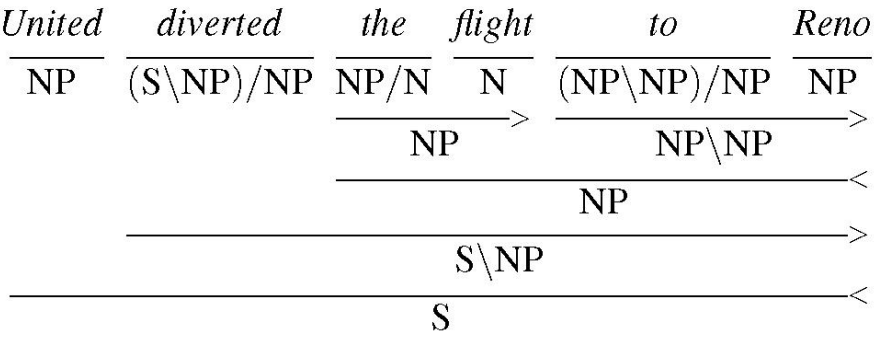
在这个例子中，我们对 the flight 的作用的理解取决于介词短语 to Reno 是作为 the flight 的修饰语，还是作为整个动词短语的修饰语，还是作为动词 divert 的潜在第二个论元。在上下文无关的语法方法中，这种歧义将表现为在语法中的下列规则中进行选择。

$$Nominal \rightarrow Nominal\ PP$$
$$VP \rightarrow VP\ PP$$
$$VP \rightarrow Verb\ NP\ PP$$

在短语结构方法中，我们将简单地将单词 to 分配到类别 P 中，允许它与 Reno 结合形成一个介词短语。随后语法规则的选择将决定最终的推导。在范畴方法中，我们可以将 to 与不同的范畴联系起来，以反映它可能与句子中的其他元素相互作用的方式。然后，相当抽象的组合规则将整理出哪些推导是可能的。因此，

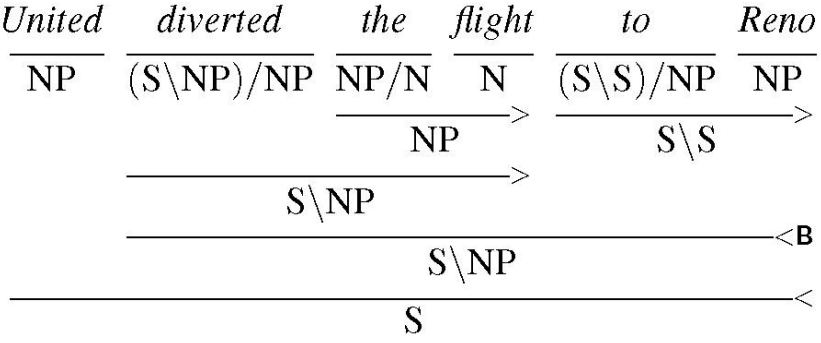
歧义的来源不是来自语法，而是来自词汇。

让我们通过考虑这个例子的几种可能的推导来看看它是如何工作的。为了捕捉介词短语 to Reno 修饰 the flight 的情况，我们将介词 to 指定为范畴(NP\NP)/NP，这就产生了以下推导。

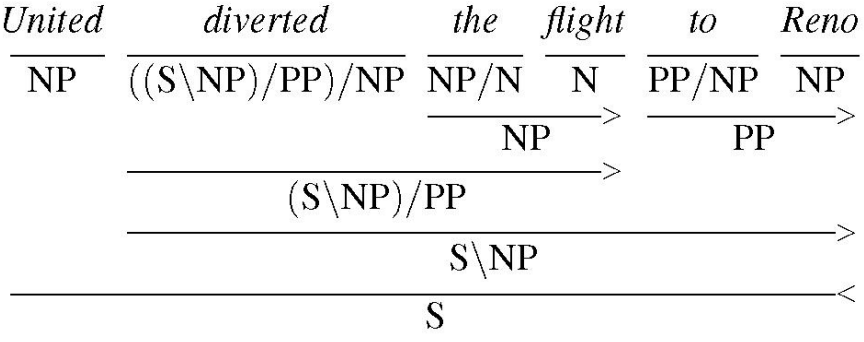


这里，分配给 to 的范畴期望找到两个论元:一个在传统介词的右边，另一个在左边，与要修改的 NP 相对应。

或者，我们可将 to 分配到范畴(S\S)/NP，这允许下面的推导，其中 to Reno 修改了前面的动词短语。



第三种可能是把 divert 看作一个双及物动词，将其归入((S\NP)/PP)/NP 范畴，而把 to Reno 看作一个简单的介词短语。



尽管 CCG 解析器仍然会因语法规则的选择而产生歧义，包括第 12 章中讨论的那种伪歧义，但应该清楚的是，词汇范畴的选择是 CCG 解析中要解决的主要问题。

13.6.2. CCG 解析框架

由于成分语法中的规则要么是二元的，要么是一元的，基于 CKY 算法的自底向上的表格方法应该直接适用于 CCG 解析。不幸的是，每个单词都有大量可用的词汇范畴，再加上 CCG 组合规则的混乱性，导致添加到解析表中的成分数量激增(大多数无用)。管理这种僵尸成分爆炸的关键是准确评估和利用每个单词最可能的词汇范畴——这个过程被称为超级标记。

以下各节描述了使用超级标记的两种 CCG 解析方法。第 13.6.4 节介绍了一种通过使用 A*算法将解析过程构造为启发式搜索的方法。接下来的部分简要介绍了一种更为传统的基于分类器的方法，该方法通过使用自适应超级标记来管理搜索空间的复杂性，该过程会反复考虑越来越多的标记，直到找到解析为止。

13.6.3. 超级标记

第八章介绍了词类标记的任务，即为句子中的每个词指定正确的词汇范畴的过程。**超级标记(supertagging)**是高度词汇化语法框架的相应任务，在这种框架中，指定的标记通常指示一个句子的大部分推导过程。

CCG 超级标记器依赖于树库(如 CCGbank)来提供词汇范畴的整体集合，以及词汇中每个单词的允许范畴分配。CCGbank 包含超过 1000 个词汇范畴，然而，在实践中，大多数超级标记器将其标记集限制为在训练语料库中出现至少 10 次的标记。这使得词典中总共有大约 425 个词汇范畴可供使用。请注意，即使这个较小的数字与宾州树库标记集使用的 45 个 POS 类型相比也很大。

与传统的词类标记一样，构建 CCG 超级标记器的标准方法是使用监督机器学习从手工注释的训练数据构建序列标记器。要找到给定句子中最有可能的标记序列，最常见的方法是使用神经序列模型，RNN 或 Transformer。

但是，也可以使用第 8 章中所述的 CRF 标记模型，并使用类似的功能：当前单词 w_i、其周围 l 个单词内的单词、本地 POS 标记和字符后缀、以及前一时间步长的超级标签，通过最大化训练语料的对数似然性进行训练，并通过第 8 章中所述的 Viterbi 算法进行解码。

不幸的是，大量可能的超级标记与较高的单词歧义相结合，导致朴素的 CRF 算法的错误率对于解析器中的实际应用而言太高了。单个最佳标记序列t̂通常包含太多不正确的标记，无法进行有效的解析。为了克服这个问题，我们取而代之的是在输入中每个单词的可能超级标记上返回概率分布。下表说明了一个简单句子的示例分布，其中每一列代表给定单词“在输入句子的上下文中(in the context of the input sentence)”每个超级标记的概率。“ ...”代表每个单词可能剩余的所有超级标记。

United	serves	Denver
N/N: 0.4	$(S\backslash NP)/NP$: 0.8	NP: 0.9
NP: 0.3	N: 0.1	N/N: 0.05
S/S: 0.1	...	...
$S\backslash S$: .05		
...		

要获得每个可能的单词/标签成对的概率，我们需要将该位置包含该标记的所有超级标记序列的概率相加。这可以通过附录 A 中描述的用于训练 CRF 的前向-后向算法来实现。

13.6.4. 使用 A*算法的 CCG 解析

A*算法是一种启发式搜索方法，利用议程来寻找最优解。代表部分解决方案的搜索状态将基于成本函数添加到议程中，并在每次迭代中选择成本最低的选项进行进一步探索。当代表一个完整解的状态第一次从议程中被选择时，它保证是最优的，搜索结束。

A*成本函数 f(n)用于有效引导搜索到解决方案。f-成本包含两个部分：

g(n)，由状态 n 表示的局部解的精确成本；

h(n)，使用状态 n 的解的启发式近似成本。

当 h(n)满足不会高估实际成本的条件时，A*会找到最优解。毫不奇怪，启发式方法越接近实际成本，A*就越能有效地找到解决方案，而无需探索解决方案空间的大部分。

当应用于解析时，搜索状态对应于代表完成成分的边。每条边都指定一个成分的开始和结束位置、其语法范畴以及其 f-成本。此处，g 分量代表边的当前成本，h 分量代表完成利用该边的推导的成本估算。使用 A*进行短语结构解析源自 Klein 和 Manning(2003)，而此处介绍的 CCG 方法基于 Lewis 和 Steedman(2014)的工作。

使用来自超级标记的信息，一个议程和一个分析表被状态初始化了，这些状态代表了输入中每个单词的所有可能词汇类范畴及其 f-成本。主循环从议程中删除了成本最低的边，并进行测试以查看其是否完整

的推导。如果它反映了一个完整的推导，则将其选择为最佳解决方案，然后循环终止；否则，将根据适用的 CCG 规则生成新状态，分配成本，并将其输入议程以等待进一步处理。循环继续进行，直到发现完整的推导；或议程已结束，表明解析失败。该算法在图 13.9 中给出。

启发式函数

在为 A*搜索定义启发式函数之前，我们需要确定如何评估 CCG 推导的质量。我们将简化假设，即 CCG 推导的概率只是在推导中分配给单词的超级标记的概率的乘积，而忽略了推导中使用的规则。更正式地说，给定包含超级标记序列 T 的句子 S 和推导 D，我们有：

$$P(D,S) = P(T,S) \tag{13.18}$$

$$= \prod_{i=1}^{n} P(t_i|s_i) \tag{13.19}$$

为了更好地适应传统的 A*方法，我们更倾向于使用成本函数给状态评分，越低越好(即，我们试图最小化推导的成本)。为了达到这个目的，我们将使用负对数概率进行推导;这将得到以下方程式，我们将使用该方程式为完整的 CCG 推导打分。

$$P(D,S) = P(T,S) \tag{13.20}$$

$$= \sum_{i=1}^{n} -\log P(t_i|s_i) \tag{13.21}$$

```
function CCG-ASTAR-PARSE(words) returns table or failure

  supertags ← SUPERTAGGER(words)
  for i ← from 1 to LENGTH(words) do
      for all {A | (words[i], A, score) ∈ supertags}
          edge ← MAKEEDGE(i − 1, i, A, score)
          table ← INSERTEDGE(table, edge)
          agenda ← INSERTEDGE(agenda, edge)
  loop do
      if EMPTY?(agenda) return failure
      current ← POP(agenda)
      if COMPLETEDPARSE?(current) return table
      table ← INSERTEDGE(chart, edge)
      for each rule in APPLICABLERULES(edge) do
          successor ← APPLY(rule, edge)
          if successor not ∈ in agenda or chart
              agenda ← INSERTEDGE(agenda, successor)
          else if successor ∈ agenda with higher cost
              agenda ← REPLACEEDGE(agenda, successor)
```

图 13-9：基于 A*算法的 CCG 解析

根据这个模型，我们可以定义 f-成本如下所示。一条边的 f-成本是两个分量的总和：

g(n)，即由这条边表示的跨度的成本；

h(n)，即完成包含这条边的推导所需成本的估计

(这些通常被称为内部成本和外部成本)。

我们将用公式 13.21 定义一条边的 g(n)。也就是说，它只是构成跨度的超级标记成本的总和。

对于 h(n)，我们需要一个接近但从不高估最终推导的实际成本的分数。满足这一要求的一个简单的启发式方法是假定外部跨度内的每个单词都将被分配其最可能的超级标记。如果这些是在最终推导中使用的标记，那么它的分数将等于启发式的分数。如果在最终的推导中使用任何其他的标签，f-成本会更高，因为新的标记必须有更高的成本，从而保证我们不会高估。

综上所述，我们得出以下关于一条边的合适 f-成本的定义。

$$
\begin{aligned}
f(w_{i,j},t_{i,j}) &= g(w_{i,j})+h(w_{i,j}) \qquad (13.22)\\
&= \sum_{k=i}^{j} -\log P(t_k|w_k)+\\
&\quad \sum_{k=1}^{i-1} \min_{t\in tags}(-\log P(t|w_k))+\sum_{k=j+1}^{N} \min_{t\in tags}(-\log P(t|w_k))
\end{aligned}
$$

举个例子，在下面的例子中，有一条边表示带有超级标记 N 的单词 serve。

(13.23) United serves Denver.

这条边的 g-成本就是这个标记的负的对数概率，-$\log_{10}$(0.1)或者是 1。外部的 h-成本由 United 和 Denver 最乐观的超级标记分配组成，分别为 N/N 和 NP。因此，这条边的 f-成本为 1.443。

一个例子

图 13.10 显示了此示例的初始议程和完整解析的进度。使用来自超级标记的信息初始化议程和分析表后，它将从议程中选择最佳的边—United 条目带有标记 N/N 和 f-成本 0.591。该条边不构成完整的解析，因此可通过应用所有相关的语法规则来生成新的状态。在这种情况下，采用前向应用程序于 United：N/N 和 serves：N，会导致创建边 United serves：N [0,2]，1.795 加到议程中。

向前跳过，在第三次迭代中，将代表曼联为丹佛服务的完整推导的一条边 United serves Denver, S[0,3], 0.716 添加到议程中。但是，该算法不会在这一点上终止，因为此边的成本(0.716)并未将其置于议程的顶部。相反，代表 Denver 的边与范畴 NP 被弹出。这导致在议程上增加了另一条边(类型提升 Denver)。只有在弹出并处理此边之后，表示完整推导的早期状态才会上升到议程的顶部，在那里弹出、测试目标并作为解决方案返回。

A *方法的有效性反映在图 13.10 中的状态着色以及最终的解析表中。以蓝色显示的边(包括未明确显示的所有初始词汇范畴分配)反映了搜索空间中从未进入议程顶部的状态，因此，从不对最终表做出任何贡献。这与 PCKY 方法相反，在 PCKY 方法中，解析器针对输入中所有可能的跨度，系统地使用所有可能的成分来填充解析表，并使用无助于最终分析的大量成分填充表。

13.7. 总结

本章介绍成分解析。以下是主要观点的总结:

- **结构歧义(structural ambiguity)**是解析器面临的重大问题。结构歧义的常见来源包括 **PP-附着(PP-attachment)**，**并列连接歧义(coordination ambiguity)** 和 **名词短语括号歧义(noun-phrase bracketing ambiguity)**。
- **动态规划(dynamic programming)**解析算法(例如 CKY)使用局部解析表来有效解析歧义语句。
- **CKY** 将语法形式限制为 Chomsky 范式(CNF)。
- 解析器使用以下三个指标进行评估：**标签召回率(labeled recall)**，**标签精度(labeled precision)**和 **交叉括号数(cross-brackets)**。
- **部分解析(Partial parsing)**和**组块(chunking)**是用于识别文本中浅层句法成分的方法。它们通过在语法注释的数据上训练的序列模型来解决。

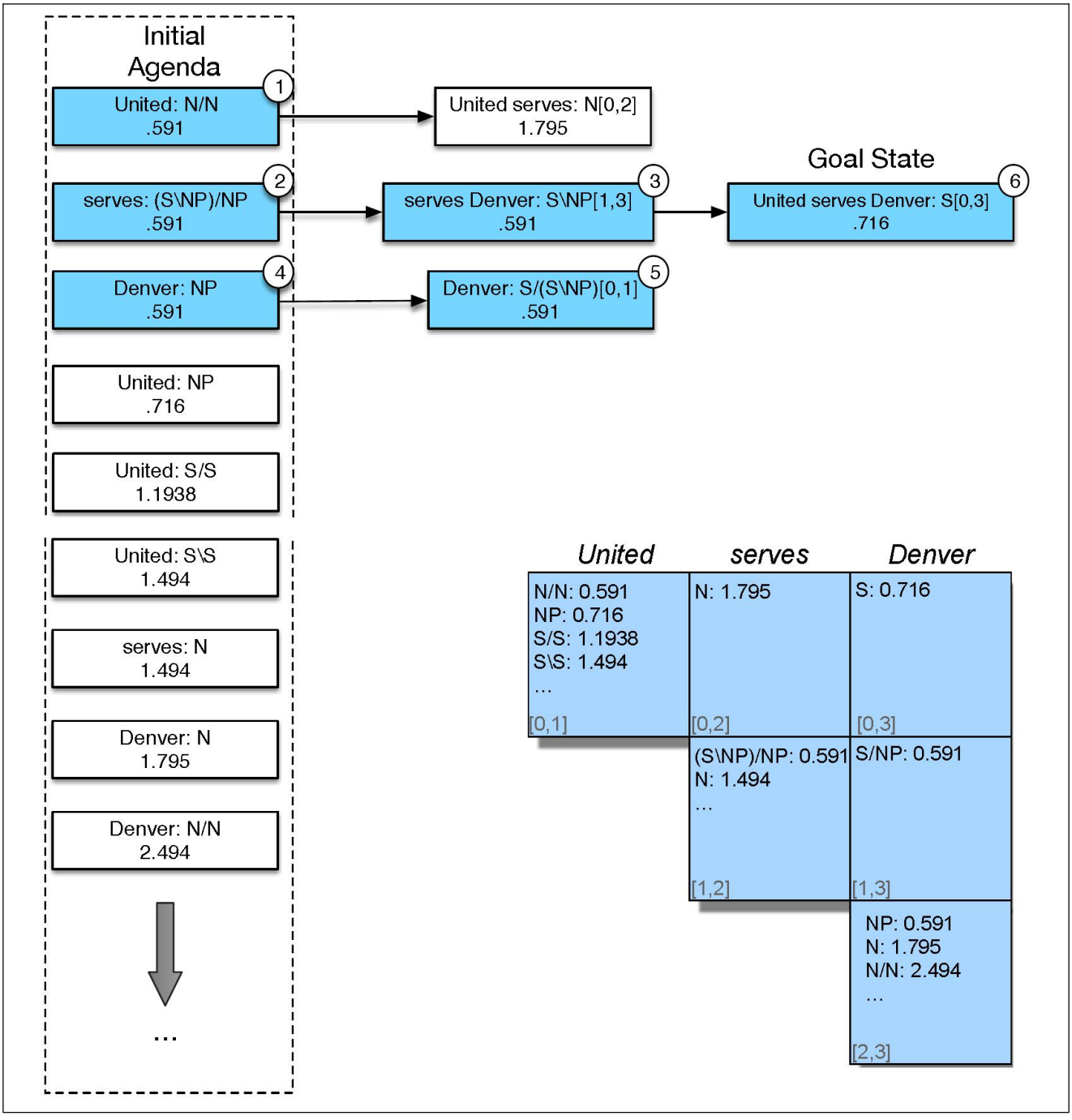


图 13-10：A*搜索“United serves Denver”的示例

图注： 蓝色框上带圆圈的数字表示从议程中弹出状态的顺序。每个状态的成本均基于使用负 log10 概率的 f-成本。

13.8. 文献和历史说明

关于编译器的历史，Knuth 写道:

In this field there has been an unusual amount of parallel discovery of the same technique by people working independently.

(在这一领域中，独立工作的人们对同一技术的并行发现数量之多是不寻常的。)

好吧，也许并不罕见，因为多重发现是科学的常态。但是，肯定有足够多的并行出版物，使这一历史错误地出于简洁的目的，即只对每种算法进行了有特色的早期提及。有兴趣的读者可以参阅 Aho 和 Ullman(1972)。

自底向上解析似乎是由 Yngve(1955)首先描述的，他给出了一个广度优先的自底向上解析算法，作为机器翻译过程说明的一部分。至少 Glennie(1960)、Irons(1961)、Kuno 和 Oettinger(1963)描述了(大概是独立的)自上而下的解析和翻译方法。动态规划解析同样具有独立发现的历史。根据 Martin Kay (个人通讯)的说法，John Cocke 在 1960 年首次实现了一个包含 CKY 算法的根(roots)的动态规划解析器。后来的工作扩展和形式化了算法，并证明了它的时间复杂度(Kay 1967, Younger 1967, Kasami 1965)。相关的格式良好的子字符串表(**WFST**)似乎是由 Kuno(1965)独立提出的，它是一种数据结构，用于存储解析过程中所有以前计算的结果。基于对 Cocke 工作的概括，Kay(1967)(和 Kay 1973)独立地描述了类似的数据结构。Earley 的博士论文(Earley 1968, Earley 1970)描述了动态规划在句法分析中的自顶向下应用。Sheil(1976)证明了 WFST 和 Earley 算法的等价性。Norvig(1991)表明，在任何带有存储化函数的语言(如 LISP)中，只要将存储化操作包装在一个简单的自顶向下解析器周围，就可以捕获动态规划所提供的效率。

虽然在解析的早期历史中，通过有限状态自动机的级联进行解析是很常见的(Harris, 1962)，但不久之后，焦点就转移到了完全 CFG 解析。Church(1980)主张回归有限状态语法作为自然语言理解的处理模型;其他早期的有限状态解析模型包括 Ejerhed(1988)。

解析算法的经典参考文献是 Aho 和 Ullman (1972);虽然这本书的重点是计算机语言，但大多数算法都已应用于自然语言。一本好的编程语言教科书，如 Aho 等人(1986)也很有用。

13.9. 练习

13.1 Implement the algorithm to convert arbitrary context-free grammars to CNF. Apply your program to the L_1 grammar.

13.2 Implement the CKY algorithm and test it with your converted L_1 grammar.

13.3 Rewrite the CKY algorithm given in Fig. 13.5 on page 264 so that it can accept grammars that contain unit productions.

13.4 Discuss the relative advantages and disadvantages of partial versus full parsing.

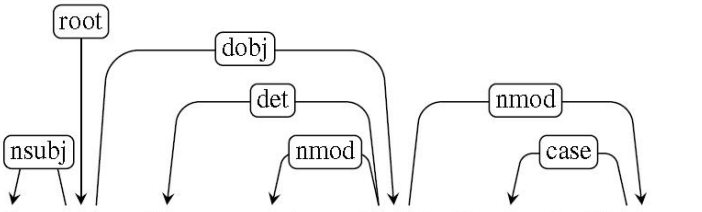
13.5 Discuss how to augment a parser to deal with input that may be incorrect, for example, containing spelling errors or mistakes arising from automatic speech recognition.

13.6 Implement the PARSEVAL metrics described in Section 13.4. Next, use a parser and a treebank, compare your metrics against a standard implementation. Analyze the errors in your approach.

14. 依存解析

前两章的重点是上下文无关的语法以及它们在自动生成基于成分的表示中的使用。在这里，我们提出了另一组语法形式主义，称为**依存语法(dependency grammars)**，这是相当重要的当代语音和语言处理系统。在这些形式主义中，短语成分和短语结构规则并不直接起作用。相反，一个句子的句法结构仅由句子中的词(或 lemmas，即词元)和包含在词之间的一组相关的定向二元语法关系来描述。

下面的图说明了使用依存关系解析社区中常用的标准图形化方法进行的依存关系风格分析。



(14.1)

单词之间的关系，在句子上方，用带方向、带标签的圆弧(从中心词到依存)来表示。我们称其为**类型依存(typed dependency)**结构，因为标签是从语法关系的固定目录中提取的。它还包括一个根节点，它显式地标记出树的根，即整个结构的中心词(head)。

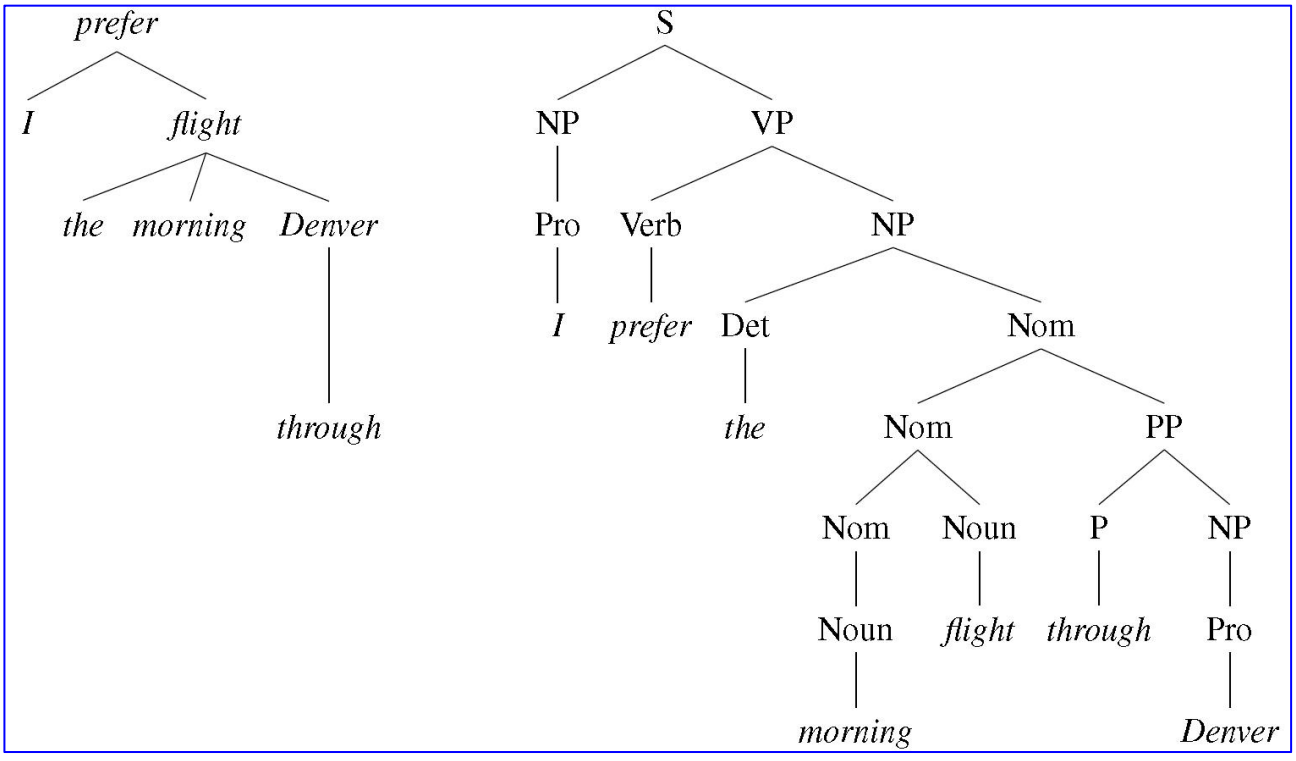


图 14-1：一个依存类型的解析及其相应的基于成分的分析

图注：针对句子：I prefer the morning flight through Denver。

图 14.1 显示了与树一样的依存关系分析，以及第十二章中给出的相应的短语结构分析。注意在依存解析中没有对应于短语成分或词汇范畴的节点;依存解析的内部结构仅由句子中词汇项之间的有向关系构成。这些关系直接编码了通常隐藏在更复杂的短语结构解析中的重要信息。例如，动词 prefer 的论元直接连接依存结构，而它们与主要动词的连接在短语结构树中更遥远。类似地，morning 和 Denver，flight 的修饰语，在依存结构中直接与之关联。

依存语法的主要优点是它们能够处理形态丰富且具有相对**自由词序(free word order)**的语言。例如，

捷克语的单词顺序比英语的灵活得多。语法宾语可能出现在位置状语之前或之后。对于语法树中可能出现此类副词短语的每个可能位置，短语结构语法都需要一个单独的规则。基于依存的方法只有一种连接类型，该类型代表此特定的状语关系。因此，依存语法方法从单词顺序信息(仅表示解析所需的信息)中抽象出来。

基于依存关系的方法的另一个实际动机是，依赖于中心词的关系为谓词及其论元之间的语义关系提供了近似值，从而使谓词直接可用于许多应用程序，例如共指解析、问题回答和信息提取。基于成分的解析方法提供了类似的信息，但通常必须通过第 12 章中讨论的查找规则等技术从树中提取出这些信息。

在下面几节中，我们将更详细地讨论在依存解析中使用的关系清单，以及这些依存结构的形式基础。然后，我们将继续讨论用于自动生成这些结构的主要算法系列。最后，我们将讨论如何评估依存分析器，并指出在语言处理应用程序中使用它们的一些方法。

14.1. 依存关系

语法关系(grammatical relation)的传统语言学概念为包含这些依存结构的二元关系提供了基础。这些关系的论元包括一个**中心词(head)**和一个**依存(dependent)**。在成分结构的背景下，我们已经在第 12 章和附录 C 中讨论了中心词的概念。在那里，一个成分的中心词是较大成分的中心组织词(例如，名词短语中的主要名词，或动词短语中的动词)。成分中的其余单词是其中心词的直接或间接依存项。在基于依存的方法中，通过将中心词直接连接到依存于它们的单词，绕过对成分结构的需求，从而使中心词-依存关系很明确。

除了指定中心词-依存偶对之外，按照依存相对于中心词扮演的角色，依存语法还允许我们进一步分类语法关系或语法功能的种类。诸如主语，直接宾语和间接宾语之类的熟悉概念都是我们想到的那种关系。在英语中，这些概念与句子中的位置和成分类型都紧密相关(但又不确定)，因此这些概念在短语结构树中发现的信息有点多余。但是，在更灵活的语言中，直接基于这些语法关系编码的信息至关重要，因为基于短语的成分句法几乎没有帮助。

毫不奇怪，语言学家已经发展出了关系分类，远远超出了我们熟悉的主语和宾语的概念。虽然理论与理论之间有相当大的差异，但有足够的共性，使开发一个计算上有用的标准成为可能。**通用依存(Universal Dependency)**项目(Nivre 等人 2016b)提供了一个依存关系清单，这些依存关系是语言驱动的、计算有用的和跨语言适用的。图 14.2(de Marneffe 等人，2014)显示了该工作的关系子集。

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

图 14-2：选自通用依存集合的依存关系

图 14.3 提供了一些例子来说明所选的关系。

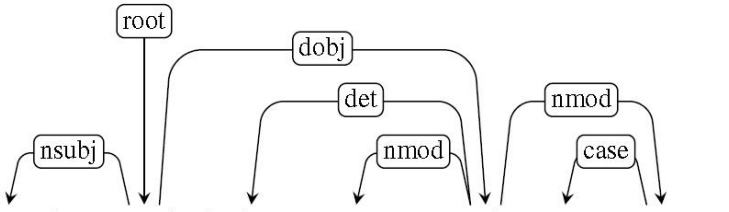
通用依存关系方案中所有关系的动机都超出了本章的范围，但是经常使用的关系的核心集可以分为两类：从句关系，描述关于谓语(通常是动词)的句法作用 ；修饰符关系，这些关系对单词可以修改其中心词

的方式进行了分类。

Relation	Examples with head and dependent
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which flight was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

图 14-3：核心通用依存关系的例子

考虑以下示例语句：



(14.2)

从句关系 NSUBJ 和 DOBJ 标识谓词 cancel 的主语和直接宾语，而 NMOD，DET 和 CASE 关系表示名词 Flight 和 Houston 的修饰语。

14.2. 依存形式

在它们最一般的形式中，我们正在讨论的简单有向图的依存结构是 G=(V,A)，其中：V 是顶点(Vertices)的集合；A 弧线(Arcs)的集合（由有序的、成对的顶点组成）。

在大多数情况下，我们将假定顶点集合 V，完全对应于给定句子中的单词集合。然而，它们也可能对应于标点符号，或者在处理形态复杂的语言时，顶点集可能由词干和词缀组成。弧线集合 A 捕获了 V 中元素之间的中心词-依存和语法功能的关系。

这些依存结构的进一步约束是特定于潜在的语法理论或形式主义的。更常见的限制是结构必须是连接的，有指定的根节点，并且是非循环的或平面的。与本章讨论的解析方法最相关的是对根树(rooted tree)的常见的、计算驱动的限制。也就是说，**依存树(dependency tree)**是满足以下约束的有向图:

1.有单个指定的根节点，没有传入的弧线。

2.除了根节点之外，每个顶点都有一条传入的弧线。

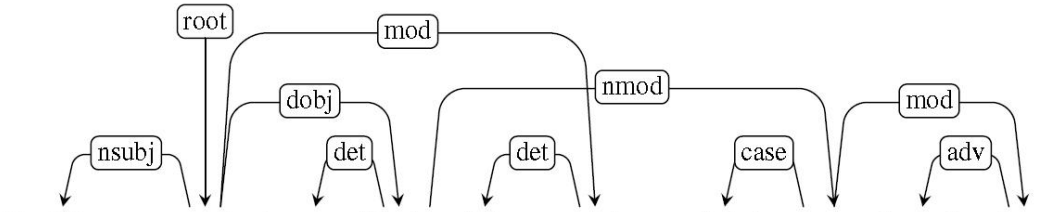
3.在 V 中，从根节点到每个顶点都有一条唯一的路径。

总的来说，这些约束确保每个单词都只有一个中心词，依存结构是连接的，并且只有一个根节点，从这个根节点可以沿着一个唯一的有向路径到达句子中的每个单词。

14.2.1. 投射性

投射性的概念强加了一个附加的约束，该约束源自输入中的单词顺序。如果句子中从中心词到从属关联词之间有一条路径，那么从中心词到从属关联词的弧线就是投射的。如果一个依存树上的所有弧线都是投射的，那么这个依存树就是投射的。到目前为止，我们看到的所有依存树都是投射的。然而，有许多完全有效的结构会导致非投射树，特别是在词序相对灵活的语言中。

考虑以下示例。



(14.3)

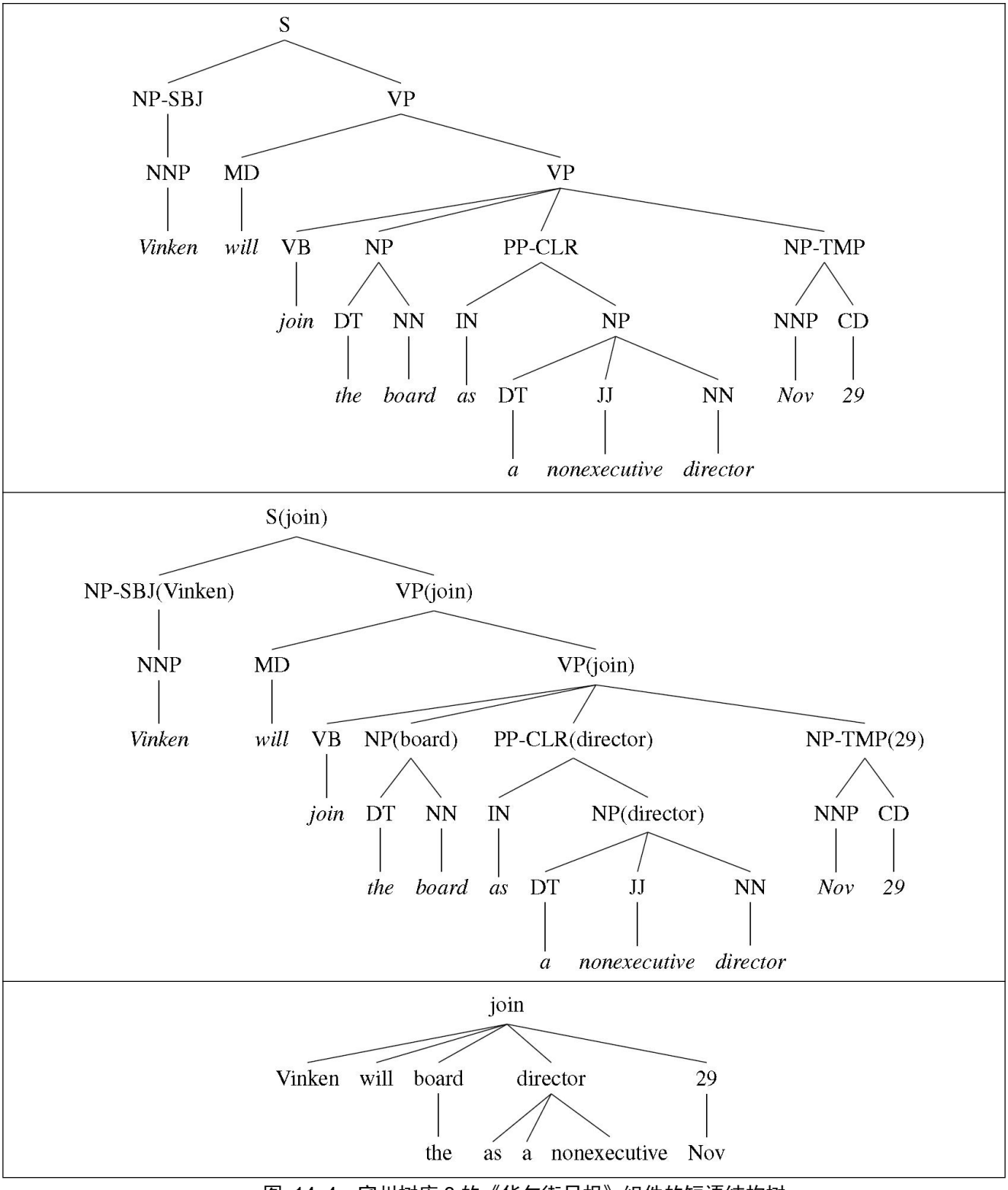


图 14-4：宾州树库 3 的《华尔街日报》组件的短语结构树

在这个例子中，从 flight 到它的修饰词 was 的弧线是非投射的，因为从 flight 到 this 和 morning 中间的单词之间没有路径。正如我们从这张图中看到的，投射性(和非投射性)可以通过我们绘制树的方式检测到。如果一个依存树没有交叉的边，那么它就是投射的。在这种情况下，如果不交叉将 morning 连接到其中心词的弧线，那么就无法将 flight 连接到其依存 was。

我们对投射性的关注来自两个相关问题。首先，最广泛使用的英语依存树库是通过使用查找规则从短语结构树库中自动得出的(第 12 章)。这样生成的树是从上下文无关的语法生成的，故可保证是投射的。

其次，最广泛使用的解析算法家族存在计算限制。第 14.4 节中讨论的基于转移的方法只能产生投射树，因此任何具有非投射结构的句子都必然包含一些错误。这个限制是第 14.5 节中描述的更灵活的基于图的解析方法的动机之一。

14.3. 依存树库

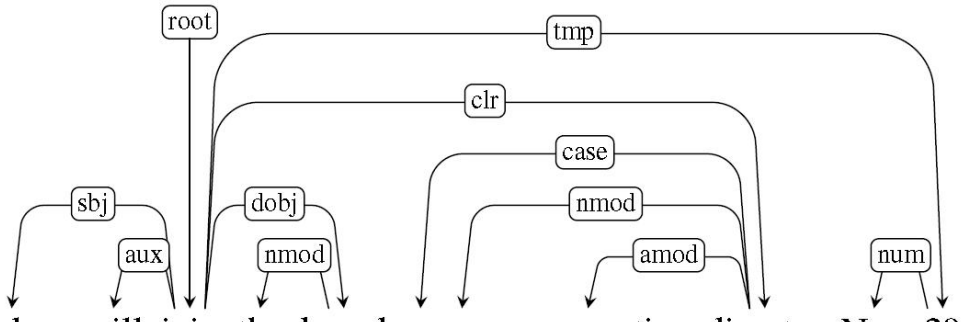
与基于成分的方法一样，树库在依存解析器的开发和评估中扮演着关键角色。依存树库的创建使用了与第 12 章中讨论的方法类似的方法——让人类注释器直接为给定的语料库生成依存结构，或者使用自动解析器提供初始解析，然后让注释器手动更正这些解析器。我们还可以使用确定性过程，通过使用中心词规则将现有的基于成分的树库转换为依存树。

在大多数情况下，已经为形态丰富的语言(如捷克语、印地语和芬兰语)创建了直接注释的依存树库，这些语言可以使用依存语法方法，其中捷克语的布拉格依赖树库(Bejcek 等人，2013)是最著名的成果。主要的英国依存树库主要是从现有资源中提取的，如宾州树库的《华尔街日报》部分(Marcus 等人，1993 年)。最近的 OntoNotes 项目(Hovy 等人 2006, Weischedel 等人 2011)扩展了这种方法，超越了传统的新闻文本，包括会话电话语音、网络日志、usenet 新闻组、广播和英语、汉语和阿拉伯语的脱口秀。

从成分到依存结构的转换过程有两个子任务:识别结构中所有的中心词-依存关系，并为这些关系识别正确的依存关系。第一个任务很大程度上依赖于第 12 章中讨论的中心词规则的使用，该规则最初是为词汇化概率分析器开发的(Magerman 1994, Collins 1999, Collins 2003)。下面是 Xia 和 Palmer(2001)提出的一个简单而有效的算法。

1. 使用适当的中心词规则在短语结构中标记每个节点的中心词孩子节点。
2. 在依存结构中，使每个非中心词孩子节点的中心词依赖于中心词孩子节点的中心词。

当短语结构解析以语法关系和功能标记的形式包含附加信息时(如宾州树库的情况)，这些标记可用于标记结果树中的边。当应用到图 14.4 中的解析树时，该算法将产生示例 14.4 中的依存结构。



(14.4)

这些提取方法的主要缺点是它们受到原始成分树中所呈现信息的限制。其中最重要的问题是未能将形态信息与短语结构树相结合，不能很容易地表示非投射结构，以及大多数名词短语缺乏内部结构，这反映在大多数树库语法中普遍使用的扁平化规则。由于这些原因，除了英语之外，大多数依存树库都是直接使用人类注释器开发的。

14.4. 基于转移的依存解析

我们的第一种依存解析方法是由一种称为**位移减少(shift-reduce)**解析的基于堆栈的方法驱动的，这

种方法最初是为分析编程语言而开发的(Aho 和 Ullman, 1972)。这种经典方法简单而优雅，使用上下文无关的语法、堆栈和要解析的符记列表。输入符记依次移动到堆栈上，堆栈的顶部两个元素与语法规则的右侧进行匹配;当找到一个匹配的元素时，匹配的元素将在堆栈上被匹配规则左边的非终端符替换(减少)。在为依存解析调整这种方法时，我们放弃了显式使用语法，并更改了减少操作，以便在一个单词和它的中心词之间引入依存关系，而不是向解析树添加非终端符。更具体地说，减少操作被替换为两个可能的操作:断言堆栈顶部的单词和它下面的单词之间的中心词-依存关系，或者反之亦然。图 14.5 演示了这种解析器的基本操作。

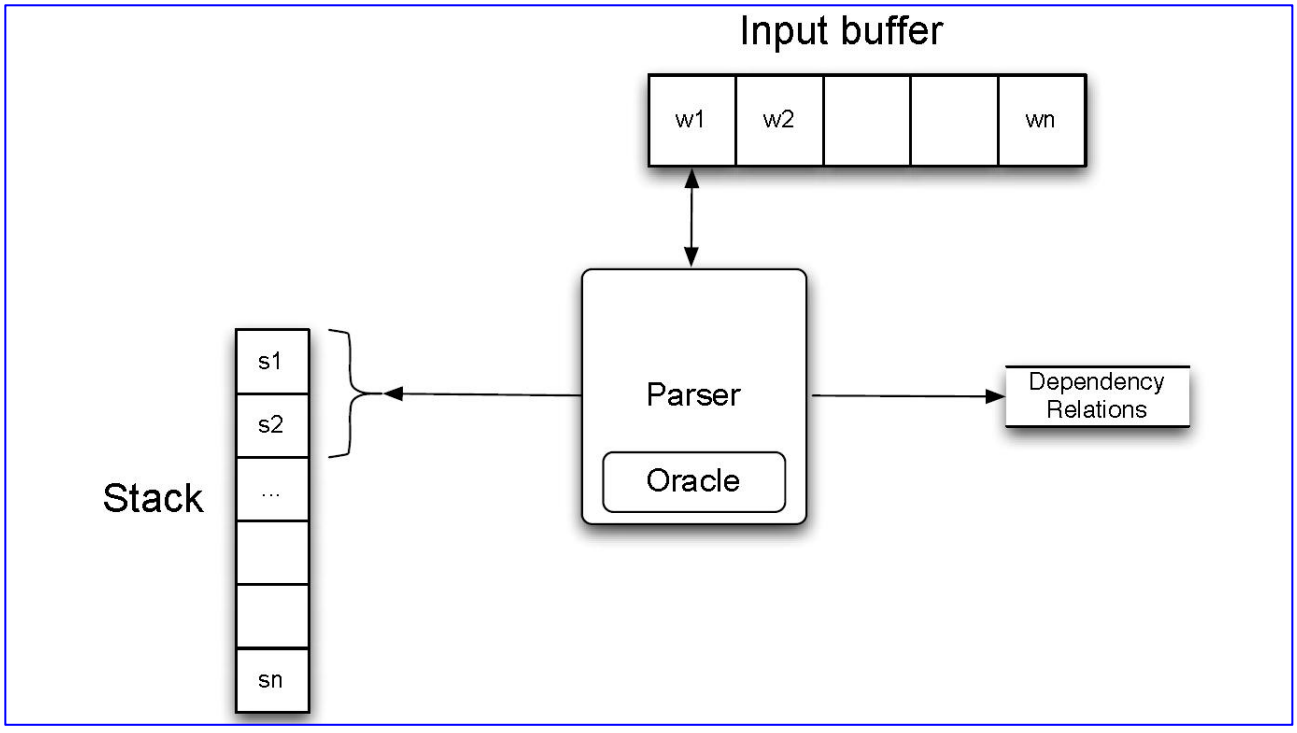


图 14-5：基于转移的基本解析器

图注：解析器检查堆栈的顶部两个元素，并根据检查当前配置的预言机选择一个操作。

基于转移的解析中的一个关键概念是**配置(configuration)**，它由堆栈、单词的输入缓冲区(或者符记)、以及一组表示依赖树的关系组成。在此框架下，解析过程由一系列通过可能配置空间的转移组成。这个过程的目标是找到一个最终的配置，其中所有的单词都已被解释，并且合成了一个适当的依存树。

为了实现这样的搜索，我们将定义一组转移操作符，当把它们应用于配置时，将产生新的配置。在这种设置下，我们可以将解析器的操作视为在配置空间中搜索从开始状态到期望的目标状态的一系列转移。在这个过程的开始，我们创建了一个初始配置，在该配置中，堆栈包含根(ROOT)节点，单词列表用句子中的单词集或词元化的符记进行初始化，并且创建一个空的关系集来表示解析。在最终的目标状态中，堆栈和单词列表应该是空的，关系集将表示最终的解析。

在基于转移的解析的标准方法中，用于产生新配置的操作符非常简单，并且与通过从左到右单次检查输入中的单词来创建依存关系树时可能采取的直观操作相对应 (Covington,2001):

•指定当前的单词作为之前看到的单词的中心词，

•指定一些以前看到的单词作为当前单词的中心词，

•或者推迟对当前单词的处理，将其添加到存储中以便后续处理。

为了使这些操作更精确，我们将创建三个转移操作符，它们将对堆栈顶部的两个元素进行操作：

•LEFTARC：在堆栈顶部的单词与其正下方的单词之间建立中心词-依存关系，从堆栈中删除低位字；

•RIGHTARC：在堆栈中的第二个单词和顶部的单词之间建立中心词-依存关系，删除堆栈顶部的单词；

•SHIFT：从输入缓冲区的开头删除单词并将其推入堆栈。

这组特定的操作符实现了基于过渡的解析的**弧线标准(arc standard)**方法(Covington 2001，Nivre 2003)。此方法有两个显著的特征：过渡操作符仅在堆栈顶部声明元素之间的关系，并且一旦分配了元素

的中心词，就将其从堆栈中移除，无法进行进一步处理。就像我们将看到的那样，还有其他的转移系统可以演示不同的解析行为，但是弧线标准方法非常有效并且易于实现。

为了确保正确地使用这些操作符，我们需要为它们的使用添加一些先决条件。首先，因为根据定义，根节点不能有任何传入的弧线，所以我们将添加一个限制，即当根是堆栈的第二个元素时，不能应用 LEFTARC 操作符。其次，两个减少操作符都要求在堆栈上应用两个元素。有了这些转移操作符和先决条件，基于转移的解析器的规范就相当简单了。图 14.6 给出了基本算法。

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)         ; choose a transition operator to apply
    state ← APPLY(t, state)   ; apply it, creating a new state
  return state
```

图 14-6：通用的基于转移的依存关系解析器算法

在每一步中，解析器都要咨询**预言机(oracle)**(我们稍后再讨论这个问题)，该预言机提供了给定当前配置的正确转移操作符。然后将该操作符应用到当前配置，生成一个新配置。当句子中的所有单词都被消费完，并且根节点是堆栈上唯一剩下的元素时，进程结束。

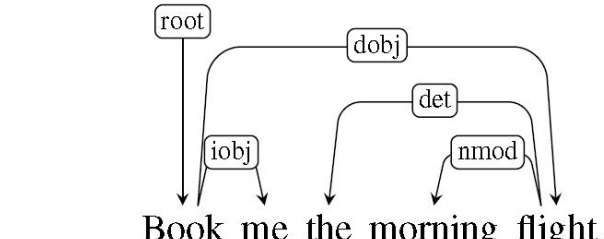
基于转移的解析器的效率从算法中可以明显看出。复杂性在句子长度上是线性的，因为它基于从左到右的单一遍历句子中的单词。更具体地说，每个单词必须先移到堆栈上，然后再减少。

请注意，与第 12 和 13 章中讨论的基于动态规划和基于搜索的方法不同，该方法是一种简单的贪心算法-预言机在每个步骤中都提供了一个选择，并且解析器继续该选择，没有探索其他选择，没有使用回溯，最后返回单个解析。

图 14.7 演示了解析器的操作，其中的转移序列将导致下面示例的解析。

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

图 14-7：基于转移的解析的跟踪



(14.5)

让我们考虑第 2 步的配置状态，在单词 me 被压入堆栈之后。

Stack	Word List	Relations
[root, book, me]	[the, morning, flight]	

这里应用的正确操作符是 RIGHTARC，它将 book 指定为 me 的中心词，并将 me 从堆栈中弹出，从而产生以下配置。

Stack	Word List	Relations
[root, book]	[the, morning, flight]	(book → me)

在多次使用 SHIFT 和 LEFTARC 操作符之后，第 6 步的配置如下所示:

Stack	Word List	Relations
[root, book, the, morning, flight]	[]	(book → me)

这里，所有剩余的字都已经传递到堆栈上，剩下要做的就是应用适当的减少操作符。在当前配置中，我们使用 LEFTARC 操作符产生以下状态。

Stack	Word List	Relations
[root, book, the, flight]	[]	(book → me) (morning ← flight)

此时，这个句子的解析由以下结构组成。



(14.6)

在检查图 14.7 中的序列时，有几件重要的事情需要注意。首先，给定的序列不是唯一可能导致合理解析的序列。通常，可能有不止一条路径导致相同的结果，而且由于歧义，可能有其他转移序列导致不同的同样有效的解析。

其次，我们假设预言机总是在解析的每个点上提供正确的操作符——这个假设在实践中不太可能是正确的。因此，鉴于这种算法的贪心本性，不正确的选择将导致不正确的解析，因为解析器没有机会返回并寻求其他选择。第 14.4.2 节将介绍几种技术，这些技术允许基于转移的方法更充分地探索搜索空间。

最后，为了简单起见，我们在没有依存关系标签的情况下演示了这个示例。要生成标记树，我们可以用依存项标签参数化 LEFTARC 和 RIGHTARC 操作符，如 LEFTARC(NSUBJ)或 RIGHTARC(DOBJ)。这等价于将原来的三个转移操作符的集合扩展为一个包含所使用的依存关系集合中的每个关系的 LEFTARC 和 RIGHTARC 操作符的集合，再加上一个额外的 SHIFT 操作符。当然，这使得预言机的工作更加困难，因为它现在有更多的操作符可供选择。

14.4.1. 创建一个预言机(Oracle)

最先进的(SOTA)基于转移的系统使用监督机器学习方法来训练扮演预言机角色的分类器。给定适当的训练数据，这些方法学习从配置映射到转移操作符的函数。

与所有有监督的机器学习方法一样，我们将需要访问适当的训练数据，我们将需要提取对做出决策的特征有用的特征。这个训练数据的源将是包含依存树的代表性树库。这些特性将包括我们在第 8 章的词类标记中遇到的许多相同的特性，以及附录 C 中用于统计解析模型的特性。

生成的训练数据

让我们从图 14.6 中的算法重新审视预言机，以充分了解学习问题。预言机将配置作为输入，并返回转移操作符作为输出。因此，要训练分类器，我们将需要与转移操作符(即 LEFTARC，RIGHTARC 或 SHIFT)

配对的配置。不幸的是，树库将整个句子与相应的树配对，因此它们无法直接提供我们所需的内容。

为了生成所需的训练数据，我们将以一种巧妙的方式采用基于预言机的解析算法。我们将向预言机提供要解析的训练语句以及来自树库的相应参考解析。为了产生训练实例，我们将通过运行算法并依靠新的**训练预言机(training oracle)**来为每个连续的配置提供正确的转移操作符，从而模拟解析器的操作。

要了解其工作原理，首先让我们回顾一下解析器的操作。它以默认的初始配置开始，其中堆栈包含ROOT，输入列表仅是单词列表，关系集为空。LEFTARC 和 RIGHTARC 操作符分别将堆栈顶部的单词之间的关系添加到给定句子累积的关系集。由于我们对每个训练句子都有一个黄金标准的参考解析，因此我们知道对于给定句子哪些依存关系有效。因此，当解析器逐步执行一系列配置时，我们可以使用参考解析来指导操作符的选择。

更精确地说，给定一个参考解析和配置，预言机的训练过程如下:

•如果在给定参考解析和当前配置的情况下产生正确的中心词-依存关系，则选择 LEFTARC;

•否则，如果(1)在给定参考解析的情况下产生正确的中心词-依存关系，并且(2)已分配了堆栈顶部单词的所有从属关系，则选择 RIGHTARC;

•否则，选择 SHIFT。

需要限制选择 RIGHTARC 操作符，以确保在未将所有单词分配给其依赖项之前，不会从堆栈中弹出一个单词，从而避免该单词被进一步处理。

更正式地说，在训练期间，预言机可以访问以下信息：

•当前配置有一个栈和一组依存关系 R_c

•一个由顶点 V 和依存关系 R_p 组成的参考解析

有了这些信息，预言机会选择如下的转换:

LEFTARC(r): **if** $(S_1\ r\ S_2) \in R_p$

RIGHTARC(r): **if** $(S_2\ r\ S_1) \in R_p$ **and** $\forall r', w\ s.t. (S_1\ r'\ w) \in R_p$ **then** $(S_1\ r'\ w) \in R_c$

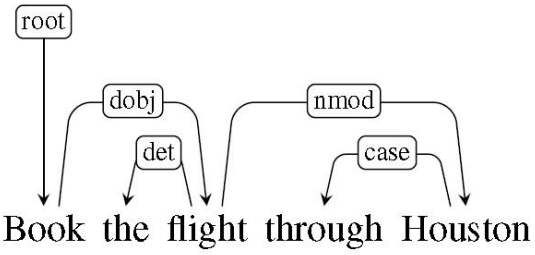
SHIFT: **otherwise**

让我们通过图 14.8 所示的示例来完成这个过程的步骤。

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

图 14-8：用参考解析生成训练项

图注：通过使用给定的参考解析来模拟解析，生成由配置/预测动作对组成的训练项



(14.7)

在第 1 步，LEFTARC 在初始配置中不适用，因为它断言一个关系(root←book)不在参考答案中;RIGHTARC 确实断言了包含在最终答案(root → book)中的关系，然而 book 还没有附加到它的任何依存项上，所以我们不得不服从（defer），留下 SHIFT 作为唯一可能的行动。在接下来的两个步骤中也存在相同的条件。在步骤 3 中，LEFTARC 被选中以连接 the 到其中心词。

现在考虑第 4 步中的情况。

Stack	Word buffer	Relations
[root, book, flight]	[through, Houston]	(the ← flight)

在这里，我们可能想在 book 和 flight 之间添加一个依存关系，该关系出现在参考解析中。但现在这样做可以防止 Houston 的后续附着，因为 flight 将从堆栈中删除。幸运的是，选择 RIGHTARC 的先决条件阻止了此选择，我们再次将 SHIFT 留为唯一可行的选择。其余选择完成了此示例所需的一组操作符。

回顾一下，我们通过在参考依存树的上下文中模拟解析器的操作，从树库中获得由配置-转移偶对组成的适当训练实例。我们可以确定地在每个训练示例中记录正确的解析器操作，从而创建我们需要的训练集。

特征

生成了适当的训练实例(配置-转移偶对)之后，我们需要从配置中提取有用的特性，以便训练分类器。用于训练基于转移的系统的特征因语言、体裁和所使用的分类器类型而不同。例如，取决于正在处理的语言，诸如主语或直接宾语上的案例标记之类的语态特征可能或多或少重要。也就是说，我们已经看到的关于词类标记和部分解析的基本特性已经被证明在训练各种语言的依存解析器时非常有用。词形、词元和词类都是强有力的特征，就像中心词以及与中心词的依存关系一样。

在基于转移的解析框架中，需要从组成训练数据的配置中提取这些特性。回想一下，配置由三个元素组成:堆栈、缓冲区和当前关系集合。原则上，这些元素的任何属性都可以用通常的训练方法表示为特征。然而，为了避免稀疏性和鼓励泛化，最好将学习算法集中在解析过程中每个点上最有用的决策制定方面。因此，基于转移的解析的特征提取的重点是堆栈的顶层、靠近缓冲区前端的单词以及已经与这些元素相关联的依存关系。

通过将简单的特征(如单词形式或词类)与配置中的特定位置相结合，我们可以使用我们在情感分析和词类标注中已经遇到的特征模板(feature template)的概念。特征模板允许我们从一个训练集自动生成大量的特定特征。作为一个例子，考虑以下基于配置中的单个位置的特征模板。

$$\langle s_1.w,op\rangle,\ \langle s_2.w,op\rangle\langle s_1.t,op\rangle,\ \langle s_2.t,op\rangle$$
$$\langle b_1.w,op\rangle,\ \langle b_1.t,op\rangle\langle s_1.wt,op\rangle \quad (14.8)$$

在这些示例中，单个特征被表示为位置.属性(location.property)，其中 s 表示堆栈，b 表示单词缓冲区，r 表示关系集合。位置的个别属性包括 w 表示词形，l 表示词元，t 表示词类。例如，对应于堆栈顶部的单词形式的特征将被表示为 $s_1.w$，以及位于缓冲区 $b_1.t$ 前端的词类部分。我们也可以通过连接将单个的特征组合成更具体的特征，这可能证明是有用的。例如，$s_1.wt$ 指定的特性表示单词的形式与堆栈顶部单词的词类相连接。最后，op 代表所讨论的训练示例的转移操作符(即，训练实例的标签)。

让我们考虑一下上面给出的一组简单的单个元素特征模板，这些模板在下面的中间(intermediate)配置的上下文中，这些中间配置来自于一个训练预言机示例 14.2。

Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled → United) (flights → morning) (flights → the)

这里正确的转移是 SHIFT(在继续之前你应该说服自己)。将我们的一组特性征模板应用到这个配置中，将产生以下一组实例化的特征。

$$\langle s_1.w = \text{flights},op = \text{shift}\rangle \quad (14.9)$$

<s_2.w = canceled,op = shift>
<s_1.t = NNS,op = shift>
<s_2.t = VBD,op = shift>
<b_1.w = to,op = shift>
<b_1.t = TO,op = shift>
<s_1.wt = flightsNNS,op = shift>

考虑到左右圆弧转换操作在堆栈的顶部两个元素上，将这些位置的属性组合起来的特征更加有用。例如，像 $s_1.t \circ s_2.t$ 这样的特征将堆栈顶部的单词的词类标记与它下面的单词的标记连接起来。

$$\langle s_1.t \circ s_2.t = NNSVBD, op = shift \rangle \tag{14.10}$$

毫不奇怪，如果两个属性是有用的，那么三个或更多应该更好。

图 14.9 给出了一组已经使用的基本特征模板(Zhang 和 Clark 2008, Huang 和 Sagae 2010, Zhang 和 Nivre 2011)。注意，这些特征中有一些使用了动态特征，例如在解析过程的早期步骤中预测的中心词和依存关系等特性，而不是从输入的静态属性派生的特征。

Source	Feature templates		
One word	$s_1.w$	$s_1.t$	$s_1.wt$
	$s_2.w$	$s_2.t$	$s_2.wt$
	$b_1.w$	$b_1.w$	$b_0.wt$
Two word	$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
	$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
	$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

图 14-9：训练基于转移的依存分析器的标准特征模板

图注：s_n 是指堆栈上的一个位置，b_n 是指单词缓冲区中的一个位置，w 是指输入的单词形式，t 是指输入的词类部分。

学习

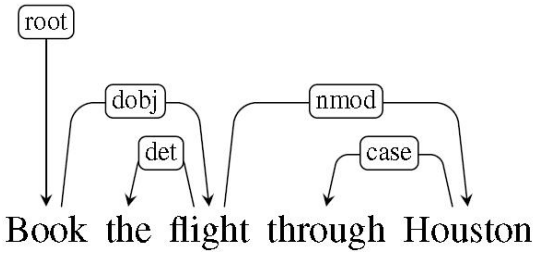
多年来，训练基于转移的依存分析器的主要方法是多元逻辑回归和支持向量机，这两种方法都可以有效地利用上一节中描述的大量稀疏特征。最近，第 8 章中描述的神经网络或深度学习方法已成功应用于基于转移的解析(Chen 和 Manning, 2014)。这些方法消除了对复杂的、手工制作的特性的需要，并且在克服通常与训练基于转移的解析器相关的数据稀疏性问题方面特别有效。

14.4.2. 基于转移解析中的高级方法

基本的基于转移的方法可通过多种方式加以阐述，通过解决该方法中一些最明显的缺陷来提高性能。

替代转移系统

上面描述的弧线标准转换系统只是许多可能的系统之一。一种常用的替代方法是弧线急转(arc eager)系统。弧线急转方法得名于它能够比弧线标准方法更快地断言右向(rightward)关系。为了看到这一点，让我们回顾一下例子 14.7 中的弧线标准轨迹，在这里重复。



在这个分析中考虑 book 和 flight 之间的依存关系。如图 14.8 所示，尽管 book 和 flight 在第 4 步更早地到达堆栈上，但弧线标准方法将在第 8 步确定这种关系。这个关系在这一点上不能被捕获的原因是由于存在后名词类(postnominal)修饰语 through Houston。在弧线标准的方法中，依存项一旦被分配了中心

词就会被从堆栈中移除。如果 flight 在步骤 4 中被指定 book 作为它的中心词，它将不能再担任 Houston 的中心词。

虽然这个延迟在这个例子中不会造成任何问题，但通常一个单词等待被分配到它的中心词的时间越长，出错的机会就越多。弧线急转系统解决了这个问题，它允许单词尽可能早地连接到它们的中心词，在所有依赖于它们的后续单词被看到之前。这是通过对 LEFTARC 和 RIGHTARC 操作符的微小更改和添加一个新的 REDUCE 操作符来实现的。

•LEFTARC:断言在输入缓冲区前面的单词和堆栈顶部的单词之间的中心词-依存关系;弹出堆栈。

•RIGHTARC:在堆栈顶部的单词和输入缓冲区前面的单词之间断言一个中心词-依存关系;将输入缓冲区前面的单词移到堆栈上。

•SHIFT:从输入缓冲区的前面移除单词，并将其压入堆栈。

•REDUCE:弹出堆栈。

LEFTARC 和 RIGHTARC 操作符应用于堆栈的顶部和输入缓冲区的前端，而不是像弧线标准方法中那样应用于堆栈的顶部两个元素。现在，RIGHTARC 操作符将依赖项从缓冲区移动到堆栈，而不是删除它，从而使它可以作为后续单词的中心词。新的 REDUCE 操作符从堆栈中移除顶部元素。这些更改加在一起允许一个单词被急切地分配它的中心词，并且仍然允许它作为后续依存的中心词。图 14.10 所示的轨迹说明了这个例子的新决策序列。

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, the, flight, through, houston]	RIGHTARC	(root → book)
1	[root, book]	[the, flight, through, houston]	SHIFT	
2	[root, book, the]	[flight, through, houston]	LEFTARC	(the ← flight)
3	[root, book]	[flight, through, houston]	RIGHTARC	(book → flight)
4	[root, book, flight]	[through, houston]	SHIFT	
5	[root, book, flight, through]	[houston]	LEFTARC	(through ← houston)
6	[root, book, flight]	[houston]	RIGHTARC	(flight → houston)
7	[root, book, flight, houston]	[]	REDUCE	
8	[root, book, flight]	[]	REDUCE	
9	[root, book]	[]	REDUCE	
10	[root]	[]	Done	

图 14-10：使用弧线急转转移操作符的处理轨迹

图注：针对句子：Book the flight through Houston

除了演示弧线急转转换系统之外，这个示例还演示了整个基于转移方法的强大功能和灵活性。我们能够在无需对底层解析算法做任何更改的情况下切换到一个新的转移系统。这种灵活性的发展导致了一组不同的转移系统,解决句法和语义的不同方面，包括:分配词类标记(Choi 和 Palmer，2011),允许生成非投射依存结构(Nivre, 2009)，分配语义角色(Choi 和 Palmer, 2011b)，以及解析包含多种语言的文本(Bhat 等人，2017)。

波束搜索

前面讨论过的基于转移的方法的计算效率源于这样一个事实，即它只通过一个句子，贪心地做出决定而不考虑其他选择。当然，这也是其最大弱点的根源——一旦做出了决定，就无法撤销，即使在之后的句子中出现了压倒性的证据。另一种方法是系统地探索备选决策序列，在这些选择中选择最好的。这种搜索的关键问题是管理大量的潜在序列。波束搜索通过将宽度优先搜索策略与启发式过滤器相结合来实现这一点，启发式过滤器修剪搜索边界以保持在一个固定大小的波束宽度内。

在将波束搜索应用于基于转移的解析时，我们将详细说明图 14.6 中给出的算法。我们不会在每次迭代中选择单个最佳转移操作符，而是将所有适用的操作符应用于议程上的每个状态，然后对结果进行评分。然后，在波束中必须有空间的约束下，我们将这些新配置中的每一个都添加到边界。只要议程的大小在指定的波束宽度内，我们就可以向议程添加新的配置。一旦议程达到极限，我们将仅添加比议程上最差的配置更好的新配置(删除最糟糕的元素，使我们保持在极限之内)。最后，为了确保我们能检索到议程上的最

佳状态，只要议程上存在非最终状态，while 循环就会继续。

波束搜索方法需要一个比我们使用的贪心算法更详细的评分概念。在这里，我们假设使用监督机器学习训练的分类器将作为一个预言机，根据从当前配置中提取的特征选择最佳的转移操作符。不管具体的学习方法是什么，这个选择都可以看作是给所有可能的转移打分，然后选出最好的一个。

$$\hat{T}(c) = argmaxScore(t, c)$$

使用波束搜索，我们现在正在搜索决策序列的空间，因此根据整个历史来为配置评分是有意义的。更具体地说，我们可以将新配置的分数定义为其前身的分数加上用于生成该配置的操作符的分数。

$$\begin{aligned} ConfigScore(c_0) &= 0.0 \\ ConfigScore(c_i) &= ConfigScore(c_{i-1}) + Score(t_i, c_{i-1}) \end{aligned}$$

这个分数用于筛选议程和选择最终答案。基于转移的句法分析的波束搜索新版本如图 14.11 所示。

```
function DEPENDENCYBEAMPARSE(words, width) returns dependency tree

  state ← {[root], [words], [], 0.0}     ;initial configuration
  agenda ← ⟨state⟩                       ;initial agenda

  while agenda contains non-final states
    newagenda ← ⟨⟩
    for each state ∈ agenda do
      for all {t | t ∈ VALIDOPERATORS(state)} do
        child ← APPLY(t, state)
        newagenda ← ADDTOBEAM(child, newagenda, width)
    agenda ← newagenda
  return BESTOF(agenda)

function ADDTOBEAM(state, agenda, width) returns updated agenda

  if LENGTH(agenda) < width then
    agenda ← INSERT(state, agenda)
  else if SCORE(state) > SCORE(WORSTOF(agenda))
    agenda ← REMOVE(WORSTOF(agenda))
    agenda ← INSERT(state, agenda)
  return agenda
```

图 14-11：波束搜索应用于基于转移的依存解析

14.5. 基于图的依存解析

基于图的依存解析方法在可能的树空间中搜索给定的句子，搜索最大化某个分数的树。这些方法将搜索空间编码为有向图，并利用图论的方法在空间中搜索最优解。更正式地说，给定一个句子 S，我们要在 $\mathcal{G}_S$ 中寻找最好的依存树，即这个句子的所有可能的树的空间，来最大化某个分数。

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}}\, score(t, S)$$

与附录 C 中讨论的上下文无关解析的概率方法一样，树的总体得分可以看作是树各部分得分的函数。本节的重点是**边因子(edge-factored)**方法，其中树的分数基于组成树的边的分数。

$$score(t,S) = \sum_{e \in t} score(e)$$

使用基于图的方法有几个动机。首先，与基于转移的方法不同，这些方法能够生成非投射树。尽管对英语来说，投射性不是一个重要的问题，但对世界上许多语言来说，它绝对是一个问题。第二个动机与解析的准确性有关，尤其是对较长的依存关系而言。从经验上看，基于转移的方法在依存关系较短的情况下具有较高的准确性，但随着中心词与依存之间距离的增加，准确性显著下降(McDonald 和 Nivre, 2011)。基于图的方法通过对整个树进行评分来避免这个困难，而不是依赖贪心的局部决策。

下一节研究一种广泛研究的方法，该方法基于对加权有向图使用**最大生成树(maximum spanning tree，MST)**算法。然后我们讨论通常用于评分树的特征，以及用于训练评分模型的方法。

14.5.1. 解析

这里描述的方法使用一个有效的贪心算法在有向图中寻找最优生成树。给定一个输入句子，它首先构造一个全连接的、有权的、有方向的图，其中顶点是输入词，有向边代表所有可能的中心词-依存赋值。一个附加的根节点包含了指向所有其他顶点的出边。图中的权重反映了由训练数据生成的模型提供的每个可能的中心词-依存的得分。给定这些权重，从根产生的该图的最大生成树表示句子的首选依赖项解析。图 14.12 所示的 Book that flight 的有向图，与所需解析对应的最大生成树用蓝色表示。为了便于说明，这里我们将重点讨论无标记的依赖项解析。第 14.5.3 节将讨论基于图的标记解析方法。

在描述算法之前，考虑关于有向图及其生成树的两种直觉是有用的。第一个直觉是从这样一个事实开始的:一个生成树中的每个顶点都有一条传入的边。由此可以得出，每一个生成树的连通分量也会有一条进来的边。第二种直觉是，边的分数的绝对值对确定最大生成树并不重要。相反，重要的是进入每个顶点的边的相对权值。如果我们从每条进入一个给定顶点的边中减去一个常量，这对最大生成树的选择没有影响，因为每棵可能的生成树都会减少相同的数量。

算法本身的第一步非常简单。对于图中的每个顶点，选择一个得分最高的传入边(代表可能的中心词赋值)。如果这些边形成了一棵生成树，那么我们就完成了。更正式地说，给定原全连通图 G = (V,E)，一个子图 T = (V,F)是一棵生成树，如果它没有环且每个顶点(除根之外)都有一条入边。如果贪心选择过程产生了这样一棵树，那么它就是可能的最佳树。

不幸的是，这种方法并不总是生成树，因为所选的边集可能包含循环。幸运的是，在另一种多重发现的情况下，有一种直接的方法可以消除贪心选择阶段产生的循环。Chu 和 Liu(1965)和 Edmonds(1967)独立开发了一种方法，它从贪心选择开始，然后是一个优雅的递归清理阶段，消除了循环。

清理阶段首先通过从所有进入顶点的边的分数减去进入每个顶点的最大边的分数来调整图中的所有权值。这就是前面提到的直觉发挥作用的地方。我们对边的值进行了缩放，这样循环中边的权值就不会对任何可能生成树的权值产生影响。从进入顶点的每条边减去权值最大的边的值，结果在贪心选择阶段选中的所有边的权值为零，包括循环中涉及的所有边。

在调整了权重之后，该算法通过选择一个循环并将其折叠成一个新节点来创建一个新图。进入或离开循环的边被更改，以便它们现在进入或离开新折叠的节点。不循环的边被包括进来，在循环中的边被丢弃。

现在，如果我们知道这个新图的最大生成树，我们就得到了消去这个循环所需要的东西。最大生成树的边指向表示折叠循环的顶点，告诉我们删除哪条边以消除循环。我们如何找到这个新图的最大生成树呢?我们递归地把算法应用到新图上。这将产生一个生成树或一个带有循环的图。只要遇到循环，递归就可以继续。当每次递归完成时，我们展开折叠的顶点，恢复循环中的所有顶点和边(要删除的单条边除外)。

将所有这些放在一起，最大生成树算法包括贪心的边选择、边成本的重新评分和需要时的递归清理阶段。完整算法如图 14.13 所示。

图 14.14 通过 Book that flight 示例逐步介绍了算法。该图的第一行说明了贪心的边选择，其中选择的边以蓝色显示(对应于算法中的集合 F)。这导致了 that 和 flight 之间的循环。右图显示了使用输入每个节点的最大值的缩放后的权重。

第二行显示两个内容：将 that 和 flight 之间的循环分解到单个节点(标记为 tf)；新缩放的成本递归。递归中的贪心选择步骤生成了一棵生成树，它将 root 连接到 book，并生成一条边，将 book 连接到收缩节点。将收缩的节点展开，我们可以看到这样一条边，这条边对应原始图中从 book 到 flight 的边。这反过来告诉我们应该放下哪条边来消除循环。

在任意有向图上，此版本的 CLE 算法运行时间为 O(mn)，其中 m 为边数，n 为节点数。因为这个算法的特殊应用始于构造一个全连接图 $m = n^2$，产生 $O(n^3)$的运行时间。Gabow 等人(1986)提出了一个运行时间为 O(m+nlogn)的更高效的实现。

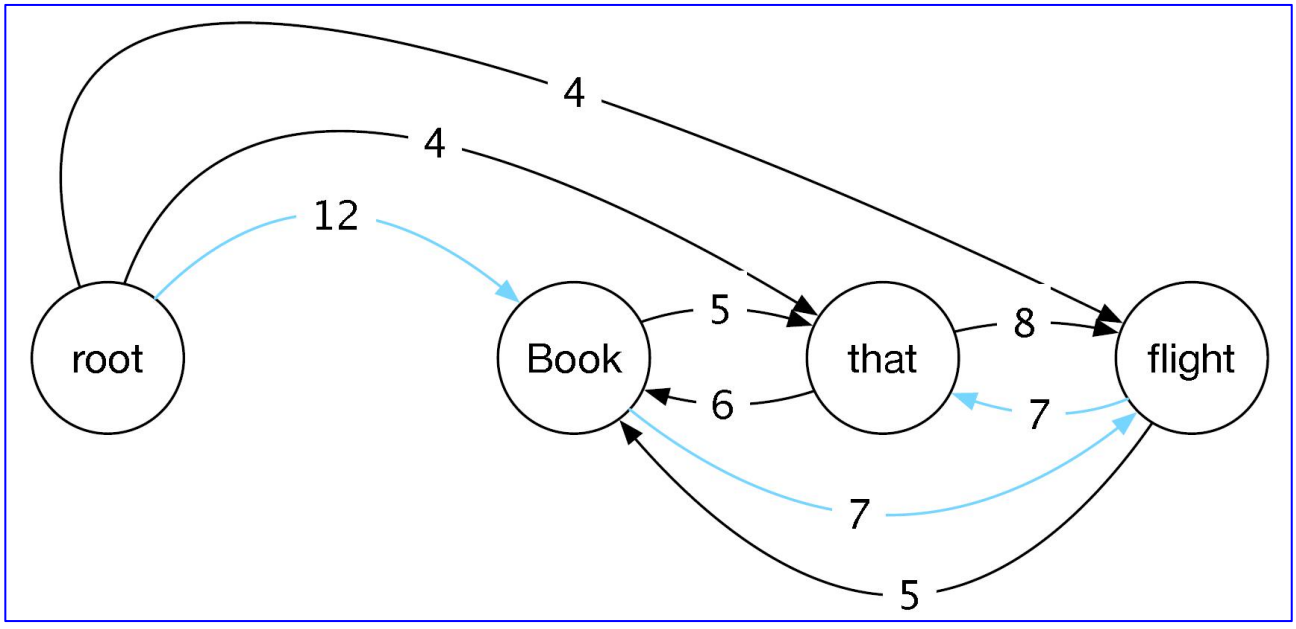


图 14-12：Book that flight 的初始根有向图

```
function MaxSpanningTree(G=(V,E), root, score) returns spanning tree

    F ← []
    T' ← []
    score' ← []
    for each v ∈ V do
        bestInEdge ← argmax_{e=(u,v)∈E} score[e]
        F ← F ∪ bestInEdge
        for each e=(u,v) ∈ E do
            score'[e] ← score[e] − score[bestInEdge]

        if T=(V,F) is a spanning tree then return it
        else
            C ← a cycle in F
            G' ← Contract(G, C)
            T' ← MaxSpanningTree(G', root, score')
            T ← Expand(T', C)
            return T

function Contract(G, C) returns contracted graph

function Expand(T, C) returns expanded graph
```

图 14-13：在加权有向图中寻找最大生成树的 CLE 算法

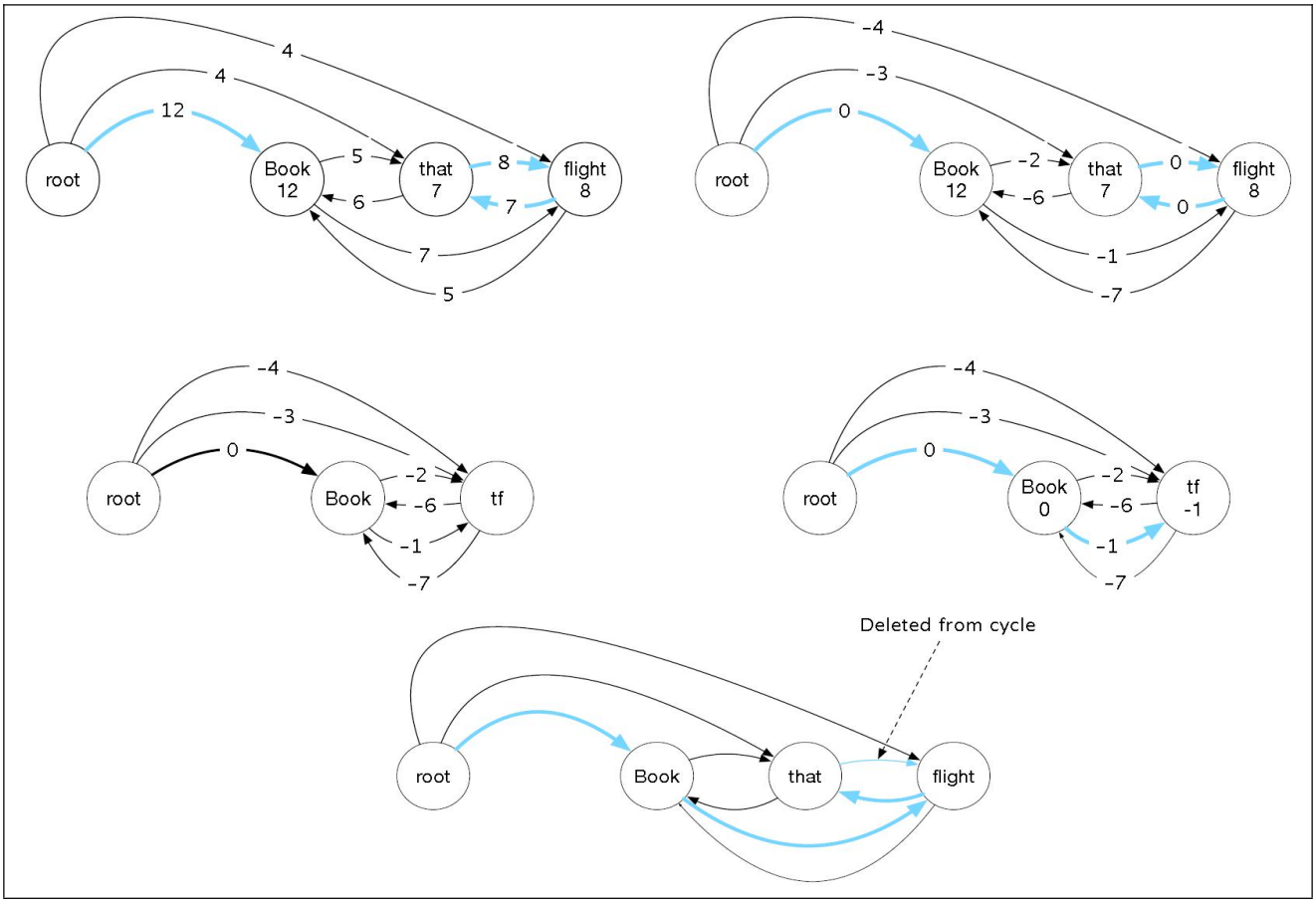


图 14-14：基于图的 CLE 算法示例

14.5.2. 特征和训练

给定一个句子 S 和一个候选树 T，边因子的解析模型将树的分数降低为构成树的边的分数之和：

$$score(S,T) = \sum_{e \in T} score(S,e)$$

每个边分数可以依次减少为从它提取的特征的加权之和：

$$score(S,e) = \sum_{i=1}^{N} w_i f_i(S,e)$$

或者更简洁：

$$score(S,e) = w \cdot f$$

有了这个公式，我们在训练解析器时面临两个问题:识别相关的特征和找到用于为这些特征评分的权重。用于训练边因子模型的特征与训练基于转移的解析器的特征相似(如图 14.9 所示)。这并不奇怪，因为在这两种情况下，我们都试图在单一关系的上下文中捕捉关于中心词和它们的依存的信息。总结前面的讨论，常用的特征包括:

- 词形、词元、中心词及其依存的词类。
- 从词前、词后和词间的上下文中衍生出相应的特征。
- 单词嵌入。
- 依存关系本身。
- 关系的方向(向右或向左)。

•从中心词到依存的距离。

与基于转移的方法一样，也经常使用预先选定的这些特征组合。

给定一组特征，我们的下一个问题是学习一组与每个特征对应的权值。与前面章节中讨论的许多学习问题不同，这里我们没有训练一个模型来将训练项与类标签或解析器动作关联起来。相反，我们试图训练一个模型，让正确的树比错误的树得分更高。对于这样的问题，一个有效的框架是使用基于推理的学习(inference-based learning)结合感知器学习规则。在这个框架中，我们使用初始权值的初始随机集从训练集中解析一个句子(即执行推理)。如果结果解析与训练数据中相应的树相匹配，我们就不对权重做任何处理。否则，我们会在错误解析中找到那些在参考解析中不存在的特征，然后根据学习速率降低它们的权重。我们对训练数据中的每句话都递增地这样做，直到权值收敛。

最先进的多语言解析算法是基于循环神经网络(RNN) (Zeman 等人 2017, Dozat 等人 2017)。

14.5.3. 基于图解析的高级问题

（待定）

14.6. 评估

与基于短语结构的解析一样，依存解析器的评估是通过测量它们在测试集上的工作情况来进行的。一个明显的度量标准是**精确匹配(exact match，EM)**——有多少句子被正确解析。这个度量相当悲观，大多数句子都被标错了。这样的度量不够细粒度，不足以指导开发过程。我们的度量标准必须足够敏感，以判断是否进行了实际的改进。

由于这些原因，评估依存解析器最常见的方法是有标签的和未标签的附着正确度。有标签的附着是指将一个词恰当地赋值给它的中心词，并建立正确的依存关系。未标签的附着只是查看分配的中心词的正确性，忽略了依存关系。给定一个系统输出和相应的参考解析，正确度就是输入中单词的百分比，这些单词被赋予了正确的中心词和正确的关系。这些度量标准通常称为**有标签的附着得分(LAS)**和**未标签的附着得分(UAS)**。最后，我们可用**标签正确度分数(LS)**，即带有正确标签的符记的百分比，忽略关系来自何处。

作为一个例子，考虑下面图 14.15 中所示的例子的参考解析和系统解析。

(14.11) Book me the flight through Houston.

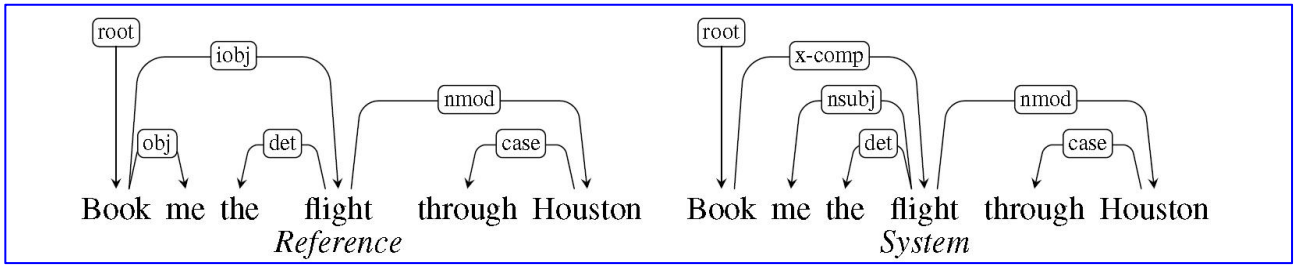


图 14-15：参考解析和系统解析示例

图注：句子 Book me the flight through Houston 的参考和系统解析，结果是 LAS 为 2/3,UAS 为 5/6。

系统正确地找到参考解析中存在的 6 个依存关系中的 4 个，并接收到 LAS 的 2/3。然而，系统发现的两个错误关系中的一个存在于 book 和 flight 之间，它们在参考解析中是中心词-依存关系;因此，该系统的 UAS 为 5/6。

除了附着评分以外，我们还可能对系统在整个开发语料库上对特定种类的依存关系(例如 NSUBJ)的性能表现有兴趣。在这里，我们可以利用第 8 章中介绍的精度和召回率概念，测量这两个指标：系统标记为 NSUBJ 的正确关系的百分比(精度)；系统实际上发现的开发集中存在的 NSUBJ 关系的百分比(召回率)。我们可以使用混淆矩阵来跟踪每种依存类型与另一种依存类型混淆的频率。

14.7. 总结

本章介绍了**依存语法**和**依存解析**的概念。以下是我们总结的要点:

•在**基于依存句法**方法中，一个句子的结构是用一组二元关系来描述的，这些关系保持在一个句子中的两个单词之间。在依存关系分析中没有直接编码更大的成分概念。

•**依存结构**中的关系捕获了句子中单词之间的**中心词-依存**关系。

•**基于依存的分析**为进一步的语言处理任务提供了直接有用的信息，包括信息提取、语义解析和问答。

•**基于转移的解析**系统采用贪心的基于堆栈的算法来创建依存结构。

•创建依存结构的基于图的方法是基于图论中**最大生成树**方法的使用。

•**基于转移**和**基于图**的方法都是使用**监督机器学习**技术开发的。

•树库提供训练这些系统所需的数据。**依存树库**可以由人工注释人员直接创建，也可以通过对短语结构树库的自动转换来创建。

•**依存解析器**的评估基于对保留开发和测试语料库测量的**有标签**的和**未标签**的**正确度分数**。

14.8. 文献和历史说明

基于依存的语法方法比相对较新的短语结构或成分语法要古老得多，这些语法多年来一直是理论和计算语言学的主要焦点。它起源于古希腊和印度的语言传统。当代的依存语法理论都大量借鉴了Tesniere(1959)的著作。最具影响力的依存语法框架包括意义文本理论(MTT) (Melcuk, 1988)、单词语法(Hudson, 1984)、功能生成描述(FDG) (Sgall 等人，1986)。这些框架在许多方面有所不同，包括它们处理形态、句法、语义和语用因素的程度和方式，它们对多层表示的使用，以及用于分类依存关系的一组关系。

戴维·海斯(David Hays)领导的兰德(RAND)公司在机器翻译方面的早期工作首次将依存语法的自动解析引入了计算语言学。这一依存解析的工作与成分解析的工作密切相关，并明确使用语法来指导解析过程。在这个早期阶段之后，依存解析的计算工作在接下来的几十年里仍然是断断续续的。在这一时期，英语依存解析器的显著实现包括连接语法(Sleator 和 Temperley, 1993)、约束语法(Karlsson 等人，1995)和MINIPAR (Lin, 2003)。

依存解析在 20 世纪 90 年代后期随着大型基于依存的树库的出现以及本章中描述的数据驱动方法的出现而重新流行起来。Eisner(1996)基于源于宾州树库的双元语法，开发了一种高效的依存解析动态规划方法。Covington(2001)在当前基于转移的方法基础上，逐字引入了确定性方法。Yamada 和 Matsumoto(2003)、Kudo 和 Matsumoto(2002)引入了位移-减少范式，以及以支持向量机的形式使用监督机器学习来进行依存分析。

Nivre(2003)定义了现代的、确定的、基于转移的依存解析方法。Nivre 和他的同事随后的工作形式化和分析了许多转移系统、训练方法和处理非投射语言的方法的性能(Nivre 和 Scholz 2004, Nivre 2006, Nivre 和 Nilsson 2005, Nivre 等人 2007, Nivre 2007)。

基于图的最大生成树方法是由 McDonald 等人（2005）引入的。

训练和评估依存英语解析器的最早数据来源来自第 12 章中描述的 WSJ 宾州树库 (Marcus 等人，1993)。为使用概率解析而开发的中心词-寻找(head-finding)规则的使用，促进了从基于短语的解析中自动提取依存解析(Xia 和 Palmer, 2001)。

长期运行的布拉格依存树库项目(PDT)(Hajic, 1998)是直接注释具有多层次形态、句法和语义信息的语料库的最重要的努力。目前的 PDT 3.0 包含超过 150 万个符记(Bejcek 等人，2013)。

通用依存(UD) (Nivre 等人，2016b)是一个旨在为跨语言的依存树库注释创建一致框架的项目，其目标是促进世界各地语言的解析器开发。UD 注释方案由几个不同的努力发展而来，包括斯坦福依存 (de Marneffe 等人 2006, de Marneffe 和 Manning 2008, de Marneffe 等人 2014)、谷歌的通用词类标记(Petrov 等人，2012)和 Interset 国际语的形态学的标记集(Zeman, 2008)。在这项努力的支持下，超过 90 种语言的树库已经被注释，并以单一一致的格式提供(Nivre 等人，2016b)。

自然语言学习会议(CoNLL)多年来进行了一系列与依存解析相关的有影响力的共享任务(Buchholz 和 Marsi 2006, Nilsson 等人 2007, Surdeanu 等人 2008, Hajic 等人 2009)。最近的评估侧重于解析器对丰富形态语言(Seddah 等人，2013 年)和非规范语言形式(如社交媒体、文本和口语语言)的鲁棒性(Petrov 和 McDonald, 2012)。Choi 等人(2015)提出了 10 个依存解析器的性能分析，包括一系列指标，以及可靠、健壮的解析器评估工具。

14.9. 练习

(无)

15. 句子含义的逻辑表示

ISHMAEL: *Surely all this is not without meaning.*

Herman Melville, *Moby Dick*

在这一章中，我们介绍了语言表达的含义可以通过被称为含义表示(meaning representations)的形式结构来表达。考虑一些需要某种形式的语义处理的任务，比如通过阅读手册学习使用新软件，通过阅读菜单决定在餐馆点什么菜，或者遵循菜谱。完成这些任务需要将语言元素与必需的非语言知识联系起来。阅读菜单，决定点什么菜，给去哪里吃饭的建议，根据食谱，生成新的食谱，所有这些都需要关于食物和它的准备，人们喜欢吃什么，以及餐馆是什么样的知识。通过阅读使用手册来学习使用软件，或提供使用软件的建议，需要了解该软件以及类似的应用程序、电脑和用户。

在本章中，我们假设语言表达具有含义表示，这些含义表示与用来表示世界上日常常识的东西是相同的。创建这种表示并将其分配给语言输入的过程称为**语义解析(semantic parsing)**或语义分析，而设计含义表示和相关语义解析器的整个过程称为**计算语义(computational semantics)**。

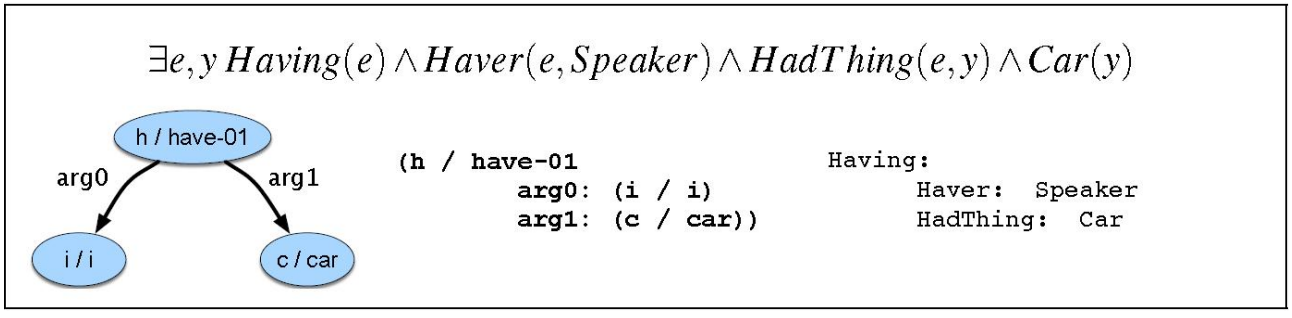


图 15-1：含义表示的采样器

图注：一个符号列表，两个有向图，和一个记录结构:I have a car 的含义表示的采样器。

以图 15.1 为例，它用四种常用的含义表示语言展示了 I have a car 这句话的示例含义表示。

第一行用**一阶逻辑(First-Order Logic)**说明了一个句子，在第 15.3 节中将详细介绍。

有向图及其对应的文本形式是**抽象含义表示(Abstract Meaning Representation，AMR)**形式的一个例子(Banarescu 等人，2013)。

右边是基于框架或**槽填充(slot-filler)**的表示，将在第 15.5 节和第 17 章中再次讨论。

虽然这些方法之间有很大的差异，但它们都有一个共同的概念，即含义表示由一组符号或表示词汇表组成的结构组成。当适当地安排时，这些符号结构被认为是对应于对象，对象的属性，以及对象之间的关系在某些事件的状态被表示或推理。在这种情况下，所有四种表示法都使用了与说话者、一辆汽车和一种关系相对应的符号，这种关系表示两者之间的占有关系。

重要的是，在所有这些方法中，这些表示至少可以从两个不同的角度来看待：作为特定语言输入 I have a car 的含义表示，以及作为某些世界事务状态表示。正是这种双重视角使这些表示能够将语言输入与世界以及我们对世界的认识联系起来。

在接下来的部分中，我们提供一些背景知识：我们对一种含义表示语言的渴望，并保证这些表示将实际执行我们需要它们执行的操作-提供与所表示的事务状态的对应关系。在 15.3 节中，我们介绍了一阶逻辑，这是历史上研究自然语言语义的主要技术，并在 15.4 节中介绍了如何将其用于捕获英语中事件和状态的语义。然后，第 16 章介绍了语义解析的技术：给定语言输入，生成这些形式的含义表示。

15.1. 表示的计算需要

让我们考虑一下为什么需要含义表示，以及它们应该为我们做什么。为了集中讨论，让我们考虑一个基于知识库向游客提供餐馆建议的系统。

①可证性

考虑下面这个简单的问题:

(15.1) Does Maharani serve vegetarian food?

要回答这个问题，我们必须知道它问的是什么，并且知道它问的问题针对 Maharini 来说是真还是假。

可验证性(verifiability)是系统比较表示形式所描述的事务状态与某个知识库中建模的世界中事务状态的能力。例如，我们需要某种表示形式诸如 Serves(Maharani，VegetarianFood)，系统可以将其与特定餐厅的事实知识库相匹配，如果找到与该命题相匹配的表示形式，则可以回答 yes；否则，如果对当地餐馆的了解是完整的，则必须说 No，或者说它不知道它是否知道自己的知识是不完整的。

②非歧义表示

语义学，就像我们所研究过的所有其他领域一样，容易出现歧义。单词和句子在不同的上下文中具有不同的含义。考虑下面的例子:

(15.2) I wanna eat someplace that's close to ICSI

这句话可能意味着说话者想在附近的某个地方吃饭，或者根据哥斯拉作为说话者的解释，说话者可能想吃附近的某个地方。这句话含糊不清； 单个语言表达可以具有两种含义之一。但是我们的含义表示本身不能是模棱两可的。输入含义的表示应没有任何歧义，以便系统可以推理表示(这意味着一件事或另一件事)来决定如何回答。

与歧义密切相关的概念是模糊性(vagueness)：其中含义表示使含义的某些部分未被充分说明。模糊性不会引起多种表示。考虑以下请求：

(15.3) I want to eat Italian food.

虽然意大利菜可以提供足够的信息来提供推荐，但对于用户真正想吃什么，它却很模糊。对于这个短语的含义的模糊表示可能适合于某些目的，而对于其他目的可能需要更具体的表示。

③典范形式(canonical form)

这一学说认为不同的输入意味着相同的东西应该有相同的含义表示。这种方法极大地简化了推理，因为系统只需要处理潜在广泛表达式的单一含义表示。

针对表达方式(15.1)，考虑以下替代方案:

(15.4) Does Maharani have vegetarian dishes?

(15.5) Do they have vegetarian food at Maharani?

(15.6) Are vegetarian dishes served at Maharani?

(15.7) Does Maharani serve vegetarian fare?

尽管这些替代方案使用不同的单词和句法，但我们希望它们映射到单一的典范含义表示。如果它们都不同，假设系统的知识库只包含对这一事实的单一表示，那么大多数表示都不会匹配。当然，我们可以在知识库中存储相同事实的所有可能的替代表示，但是这样做将导致保持知识库的一致性方面的巨大困难。

典范形式的确会使语义解析的任务复杂化。我们的系统必须得出这样的结论:vegetarian fare, vegetarian dishes, 和 vegetarian food 指的是同一件事，在这里，having 和 served 是等价的，所有这些解析结构仍然指向相同的含义表示。或者考虑以下两个例子:

(15.8) Maharani serves vegetarian dishes.

(15.9) Vegetarian dishes are served by Maharani.

尽管 serve 论元的位置不同，但系统仍必须通过利用语法知识(例如主动句和被动句结构之间的关系)将 Maharani 和 vegetarian dishes 分配给两个示例中的相同角色。

④推理和变量

对于更复杂的请求，例如：

(15.10) Can vegetarians eat at Maharani?

此请求得到的答案与其他请求的答案相同，不是因为它们的意思相同，而是因为素食者所吃的食物与素食餐厅所提供的食物之间存在常识性的联系。这是关于世界的事实。我们需要在知识库中将此请求的含义表示与关于世界的事实联系起来。系统必须能够使用**推理(inference)**-根据输入的含义表示及其背景知识得出有效的结论。系统必须有可能得出关于命题真实性的结论，这些结论没有在知识库中明确表示，但是从逻辑上可以从存在的命题中得出。

现在考虑以下一些更复杂的请求：

(15.11) I'd like to find a restaurant where I can get vegetarian food

此要求并不涉及任何特定的餐厅;该用户想要了解一家不知名的素食餐厅的信息。由于没有餐厅的名字，简单的搭配是行不通的。回答这个请求需要使用**变量(variable)**，使用如下表示:

$$Serves(x, VegetarianFood) \tag{15.12}$$

只有当变量 x 可以被知识库中的某个对象替换，并且整个命题能够匹配时，匹配才会成功。然后可使用替代变量的概念来满足用户的请求。任何含义表示语言都必须能够处理这类不确定的参考。

⑤表现力

最后，含义表示方案必须具有足够的表现力，以处理广泛的主题，理想情况下应是任何明智的自然语言语段。尽管对于任何单个表示系统来说，这可能都太过期望了，但是如 15.3 节中所述，一阶逻辑具有足够的表现力，可以处理很多需要表示的东西。

15.2. 模型理论语义学

含义表示语言是什么意思?此语言允许它们满足这些需求，弥合形式表示与表示(告诉我们一些关于世界事务状态的事情)之间的鸿沟。

答案是**模型(model)**。模型是一种正式的构造，它代表了世界上事务的特定状态。含义表示语言中的表达式可以映射到模型的元素，如对象、对象的属性和对象之间的关系。如果模型准确地捕捉了我们感兴趣的事实，那么含义表示和模型之间的一致映射就提供了含义表示和现实世界之间的桥梁。模型提供了一种惊人的简单和强大的方法来将表达式建立在含义表示语言中。

首先,一些术语。含义表示的词汇表由两部分组成:非逻辑词汇表和逻辑词汇表。非逻辑词汇表由对象、属性和关系的开放式名称集合组成，这些名称组成了我们试图表现的世界。它们以谓词、节点、连接上的标签或框架中插槽的标签的形式出现在各种方案中。逻辑词汇表由符号、操作符、量词、连接等的封闭集合组成，它们为在给定的含义表示语言中组合表达式提供了正式的手段。

非逻辑词汇表的每个元素都必须在模型中有一个**指示(denotation)**，这意味着每个元素都对应于模型中一个固定的、定义良好的部分。让我们从对象开始。模型的**领域(domain)**是被表示的对象的集合。每个不同的概念、类别或个体都表示领域中的一个独特元素。

我们通过指示具有该属性的领域元素来表示模型中对象的属性;也就是说，属性指示集合。红色属性的指示是我们认为是红色的东西的集合。类似地，对象之间的关系表示一组参与关系的领域元素的有序列表或元组：已婚关系的指示是已婚的域对象对的集合。这种处理属性和关系的方法被称为**外延的(extensional)**，因为我们通过它们的外延和它们的指示来定义概念。总结一下:

- 对象指示领域元素
- 属性指示领域元素的集合
- 关系指示领域元素的元组集合

我们现在需要一个映射，将我们的含义表示映射到相应的指示：一个函数，将我们的含义表示的非逻辑词汇映射到模型中的适当指示。我们将这种映射称为**解释(interpretation)**。

让我们回到我们的餐馆建议应用程序，让它的领域包含餐馆、顾客、关于顾客喜好和厌恶的事实，以及关于餐馆的事实，例如它们的菜肴、典型成本和噪音水平。为了开始填充我们的领域 D，让我们假设我们正在处理由非逻辑符号指定的四个顾客:Matthew、Franco、Katie 和 Caroline，表示四个唯一的领域元素。我们用常数 a、b、c、d 来表示这些领域元素。我们故意为我们的领域元素使用没有含义的、非助记符的名称，以强调这样一个事实：无论我们对这些实体的了解是什么，都必须来自模型的形式属性，而不是来自符号的名称。再继续,让我们假设我们的应用程序包括三个餐厅,设定为 Frasca、Med、和 Rio 的含义表示，以指示领域元素的 e、f 和 g。最后,让我们假设我们在处理三个菜系 Italian、Mexican、和 Eclectic,在我们的模型中用 h、i 和 j 来指示。

像 Noisy 这样的属性指示我们领域中已知是嘈杂的餐馆子集。两两（Two-place）关系概念，例如个别顾客喜欢(Like)哪个餐厅，表示领域对象的有序偶对或元组。而且，由于我们决定在模型中将菜系表示为对象，所以我们可以将哪个餐厅提供(serve)哪些菜系作为一组元组。图 15.2 给出了一种可能的情况。

Domain	$\mathcal{D} = \{a, b, c, d, e, f, g, h, i, j\}$
Matthew, Franco, Katie and Caroline	a, b, c, d
Frasca, Med, Rio	e, f, g
Italian, Mexican, Eclectic	h, i, j
Properties	
Noisy	$Noisy = \{e, f, g\}$
Frasca, Med, and Rio are noisy	
Relations	
Likes	$Likes = \{\langle a, f\rangle, \langle c, f\rangle, \langle c, g\rangle, \langle b, e\rangle, \langle d, f\rangle, \langle d, g\rangle\}$
Matthew likes the Med	
Katie likes the Med and Rio	
Franco likes Frasca	
Caroline likes the Med and Rio	
Serves	$Serves = \{\langle f, j\rangle, \langle g, i\rangle, \langle e, h\rangle\}$
Med serves eclectic	
Rio serves Mexican	
Frasca serves Italian	

图 15-2：餐厅世界的模型

有了这个简单的方案，我们就可以通过参考相应模型中的适当指示来使含义表示法成为基础。例如，我们可以通过两种映射来评估一个声称 Matthew likes the Rio(或者 The Med serves Italian)的表示：

第一种映射：将含义表示中的对象映射到其对应的领域元素；

第二种映射：将含义表示中的任何连接、谓词或槽映射到模型中的适当关系。

更具体地讲，我们可以通过两种映射验证一个断言“Matthew likes Frasca”的表示：

第一种映射：首先使用我们的解释功能将符号 Matthew 映射到其符号 a，将 Frasca 映射到 e，

第二种映射：将 Likes 关系映射到适当的元组集合。然后，我们检查元组集合中是否存在元组<a，e>。在这种情况下，如果模型中存在该元组，那么我们可以得出结论 Matthew likes Frasca 是真的；如果模型中不存在该元组，那么我们就不能得出结论。

这是非常直接的——我们使用集合和集合上的操作来建立我们的含义表示的表达式。当然，更有趣的部分是当我们考虑更复杂的例子，如以下:

(15.13) Katie likes the Rio and Matthew likes the Med.

(15.14) Katie and Caroline like the same restaurants.

(15.15) Franco likes noisy, expensive restaurants.

(15.16) Not everybody likes Frasca.

对于这样的例子，我们为表示的含义建立基础的简单方案是不够的。这些示例的合理含义表示不会直

接映射到单个实体、属性或关系。相反，它们涉及诸如连词、等式、量化变量和否定等复杂问题。为了评估这些表述是否与我们的模型一致，我们必须将它们拆开，评估各部分，然后从各部分的含义确定整体的含义。

考虑上面的第一个例子。此示例的含义表示将包括两个不同的命题，它们表示单个用户的偏好，并与某种隐式或显式的连接操作符相结合。我们的模型没有将模型中所有顾客和餐馆的成对偏好编码起来的关系，也不需要这样做。从我们的模型中我们知道 Matthew likes the Med，而 Katie likes the Rio(也就是说，元组<a, f>和<c,g>是由 likes 关系表示的集合的成员)。我们真正需要知道的是如何处理操作符的语义。如果我们为英语单词 and 假设最简单的语义，那么如果模型中的每个成分都为真，则整个语句为真。在这种情况下，两个成分都为真，因为适当的元组出现了，因此整个句子为真。

我们在这个例子中所做的是为假设的连接操作符在某些含义表示中提供**真值条件语义学(truth-conditional semantics)**。也就是说，我们提供了一种方法，可以通过部分的含义(通过咨询模型)来确定一个复杂表达式的真值，通过咨询真值表来确定一个操作符的意义。含义表示语言是有真值条件的，因为它们对我们如何从复合句的部分含义来确定复合句的含义给出了正式的说明。特别地，我们需要知道正在使用的含义表示模式的整个逻辑词汇的语义。

请注意，尽管这种情况如何发生的细节取决于所使用的特定含义表示的细节，但应该清楚的是，评估此类例子的真值条件除了我们已经讨论过的简单集合操作之外，不涉及任何其他内容。我们在下一节讨论一阶逻辑的语义时再讨论这些问题。

15.3. 一阶逻辑

一阶逻辑(First-Order Logic，FOL)是一种灵活的、易于理解的、在计算上易于处理的含义表示语言，它满足了 15.1 节中给出的许多要求。它为可验证性、推理和表现力需求提供了良好的计算基础，以及良好的模型理论语义。

FOL 的另一个吸引人的特点是，对于事物应该如何被表示，FOL 很少做出具体的承诺，而它所做的那些都是前面提到的许多方案所共有的:被表示的世界由对象、对象的属性和对象之间的关系组成。

本节的其余部分将介绍 FOL 的基本句法和语义，然后描述 FOL 在事件表示中的应用。

15.3.1. 一阶逻辑的基本要素

让我们通过首先检查 FOL 的各种原子元素，然后展示如何组合它们来创建更大的含义表示来探索 FOL。

图 15.3 为我们将要使用的特定 FOL 句法提供了一个完整的上下文无关的语法，这是本节的路线图。

首先，我们来研究**术语(term)**的概念，即用于表示对象的 FOL 设备(device)。从图 15.3 可以看出，FOL 提供了三种表示这些基本构件的方法：常量、函数和变量。这些手段中的每一个都可以被认为是在考虑中的对象的指定对象。

FOL 中的**常量(constants)**指的是被描述世界中的特定对象。这些常量通常被描述为单个大写字母，如 A 和 B，或者首字母大写单词，这常常让人联想到专有名词，如 Maharani 和 Harry。像编程语言常量一样，FOL 常量只指向一个对象。但是，对象可以有多个引用它们的常量。

FOL 中的**函数(function)**对应的概念在英语中通常表示为所有格或者属格(genitives)，如 Frasca's location。这样一个表达式的 FOL 翻译可能如下所示：

$$LocationOf(Frasca) \tag{15.17}$$

FOL 函数在语法上与单论元谓词相同。但是，重要的是要记住，虽然它们看起来像谓词，但实际上它们是术语，因为它们指的是唯一的对象。函数提供了一种方便的方法来引用特定的对象，而不必将命名常量与它们关联起来。这在许多命名对象(如餐馆)具有独特概念(如与之相关的位置)的情况下尤其方便。

变量(variable)是我们引用对象的最终 FOL 机制。变量被描述为单个小写字母，我们可以断言并推断对象，而不必引用任何特定的命名对象。这种对匿名对象进行声明的能力有两种:对特定未知对象进行声明，以及对任意对象世界中的所有对象进行声明。我们将在介绍量词(使变量有用的 FOL 元素)后，再讨论变量。

$$
\begin{aligned}
Formula &\rightarrow AtomicFormula \\
&\mid Formula\ Connective\ Formula \\
&\mid Quantifier\ Variable, \ldots\ Formula \\
&\mid \neg\ Formula \\
&\mid (Formula) \\
AtomicFormula &\rightarrow Predicate(Term, \ldots) \\
Term &\rightarrow Function(Term, \ldots) \\
&\mid Constant \\
&\mid Variable \\
Connective &\rightarrow \wedge \mid \vee \mid \Longrightarrow \\
Quantifier &\rightarrow \forall \mid \exists \\
Constant &\rightarrow A \mid VegetarianFood \mid Maharani \cdots \\
Variable &\rightarrow x \mid y \mid \cdots \\
Predicate &\rightarrow Serves \mid Near \mid \cdots \\
Function &\rightarrow LocationOf \mid CuisineOf \mid \cdots
\end{aligned}
$$

图 15-3：一阶逻辑表示的句法的上下文无关语法规范

现在我们有了引用对象的方法，我们可以继续讨论用于声明对象之间的关系的 FOL 机制。谓词是指(或称)给定领域中固定数量的对象之间的关系的符号。回到第 15.1 节非正式介绍的例子，一个合理的 FOL 表示 Maharani serves vegetarian food 可能看起来像下面的公式:

$$Serves(Maharani,VegetarianFood) \tag{15.18}$$

这个 FOL 语句断言 serve 是一个双位置谓词，位于常量 Maharani 和 vegeanfood 所表示的对象之间。下面是一个非常典型的句子，例如 Maharani is a restaurant，这说明了谓词的不同用法:

$$Restaurant(Maharani) \tag{15.19}$$

这是一个使用单个位置(one-place)谓词的示例，它不是用于关联多个对象，而是用于断言单个对象的一个属性。在这种情况下，它编码了 Maharani 的类别成员。

通过引用对象、断言关于对象的事实以及将对象彼此关联起来的能力，我们可以创建基本的复合表示。这些表示对应于图 15.3 中的原子公式层面。然而，这种组合复杂表示的能力并不仅限于使用单个谓词。更大的组合表示也可以通过使用**逻辑连接词(logical connectives)**组合在一起。从图 15.3 可以看出，逻辑连接词让我们通过使用三种操作符中的一种来连接逻辑公式，从而创建更大的表示。例如，考虑下面的 BERP 语句和它的一种可能的表示:

(15.20) I only have five dollars and I don't have a lot of time.

$$Have(Speaker,FiveDollars) \wedge \neg Have(Speaker,LotOfTime) \tag{15.21}$$

这个例子的语义表示是通过使用∧和¬操作符，以一种简单的方式从单个从句的语义建立起来的。请注意，图 15.3 中的语法的递归性质允许通过使用这些连接词创建无限个逻辑公式。因此，与句法一样，我们可以使用一个有限的手段来创建无限个表示。

15.3.2. 变量和量词

现在，我们已经具备了一切必要的机制，可以回到我们先前关于变量的讨论上来。如上所述，在 FOL 中变量以两种方式使用:引用特定的匿名对象和引用集合中的所有对象。这两种用法是通过使用称为**量词(quantifiers)**的操作符实现的。对 FOL 来说，两个基本的操作符是：

存在量词(existential quantifier)，用∃表示“存在”，发音为“there exists”；

全称量词(universal quantifier)，用∀表示“对于所有的”，发音为“for all”。

英语中经常出现不定名词短语来表示存在性量化变量的需要。考虑下面的例子:

(15.22) a restaurant that serves Mexican food near ICSI.

这里引用的是具有特定属性的特定类别的匿名对象。以下是对这一短语含义的合理解释:

$$\exists x Restaurant(x) \wedge Serves(x,MexicanFood) \wedge Near((LocationOf(x),LocationOf(ICSI)) \quad (15.23)$$

这个句子开头的存在量词告诉我们如何在这个句子的上下文中解释变量 x。非正式地说，这个句子要为真，就必须至少有一个对象。因此，如果我们用它来代替变量 x，那么得到的句子就会为真。例如，如果 AyCaramba 是 ICSI 附近的一家墨西哥餐厅，那么将 AyCaramba 代入 x 的结果，逻辑公式如下:

$$Restaurant(AyCaramba) \wedge Serves(AyCaramba,MexicanFood) \wedge Near((LocationOf(AyCaramba),LocationOf(ICSI)) \quad (15.24)$$

根据∧操作符的语义，如果它的三个组成原子公式都为真，则这个句子为真。如果它们存在于系统的知识库中，或者可以从知识库中的其他事实推断出来，那么它们将依次为真。

全称量词的使用还具有基于将已知对象替换为变量的解释。全称量词的替代语义从字面上理解为“对于一切(far all)”。∀操作符指出，要使所讨论的逻辑公式正确，用知识库中的任何对象替换通用量化变量都应得出一个真实公式。这与∃操作符形成了鲜明的对比，∃操作符只要求单个有效的替换语句为 ture。

考虑以下示例：

(15.25) All vegetarian restaurants serve vegetarian food

该句子的合理表示应类似于以下内容：

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x,VegetarianFood) \quad (15.26)$$

要使这个句子为真，每个已知对象对 x 的替换都必须产生一个为真的句子。我们可以把所有可能的替代品的集合分为由素食餐厅组成的对象集合和由其他所有东西组成的集合。让我们首先考虑这样一种情况:被取代的对象实际上是一家素食餐厅;一个这样的替换将产生下列句子:

$$VegetarianRestaurant(Maharani) \Rightarrow Serves(Maharani,VegetarianFood) \quad (15.27)$$

如果我们假设我们知道结果从句:

$$Serves(Maharani,VegetarianFood) \quad (15.28)$$

为真，那么这句话整体上也一定为真。因为前提和结果的值都为真，因此，根据图 15.4 的前两行，句子本身的值为真。这个结果将是相同的所有可能的替代术语代表素食餐厅 x。

然而，请记住，要使这个句子为真，它必须对所有可能的替换都为真。当我们从一组不是素食餐厅的对象中考虑替换时，会发生什么呢？考虑一个非素食餐厅，比如 AyCaramba 来替代变量 x:

$$VegetarianRestaurant(AyCaramba) \Rightarrow Serves(AyCaramba,VegetarianFood)$$

因蕴涵(implication)的前提是 False，故我们可以从图 15.4 确定句子始终为 True，再次满足∀约束。

请注意，AyCaramba 可能仍然会在没有成为素食餐厅的情况下提供素食。还要注意，尽管我们选择了示例，但是对于可以用这种推理代替 x 的对象没有隐含的绝对限制。换句话说，x 不受餐厅或与餐厅有关的概念的限制。考虑以下替换：

$$VegetarianRestaurant(Carburetor) \Rightarrow Serves(Carburetor,VegetarianFood)$$

在这里，前提仍然是假的，所以规则在这种不相关的替换下仍然是真。回顾一下，逻辑公式中的变量必须是存在的(∃)或全称的(∀)量化的。为了满足一个存在的量化变量，至少有一个替换必须产生一个真实的句子。为了满足一个全称的量化变量，所有的替换都必须产生真实的句子。

15.3.3. Lambda 表示法

我们需要完成对 FOL 的讨论的最后一个元素称为λ表示法(lambda notation)(Church, 1940)。这种表示法提供了一种从完全指定的 FOL 公式中抽象出来的方法，这种方法对语义分析特别有用。λ表示法扩展了 FOL 的句法，包括以下形式的表达式:

$$\lambda x.P(x) \quad (15.29)$$

这样的表达式由希腊符号λ组成，后面跟着一个或多个变量，然后是一个利用这些变量的 FOL 公式。这些λ表达式的有用性基于将它们应用于逻辑项的能力，从而产生新的 FOL 表达式，其中形式参数变

量绑定到指定的项。这个过程被称为**λ化简(λ-reduction)**，由λ变量的简单文本替换和λ的移除组成。下面的表达式说明了一个λ表达式对常数 A 的应用，以及对该表达式进行λ化简的结果:

$$\lambda x.P(x)(A) \tag{15.30}$$
$$P(A)$$

这种技术的一个重要而有用的变化是使用一个λ表达式作为另一个λ表达式的主体，如下所示:

$$\lambda x.\lambda y.Near(x, y) \tag{15.31}$$

这个相当抽象的表达可以解释为某物接近另一物的状态。下面的表达式说明了用这种嵌入λ表达式进行单个λ应用和随后的化简:

$$\lambda x.\lambda y.Near(x, y)(Bacaro) \tag{15.32}$$
$$\lambda y.Near(Bacaro, y)$$

这里的重点是得到的表达式仍然是λ表达式;第一个化简限定了变量 x，去掉了外部的λ，从而揭示了内部表达式。正如所期望的那样，这个λ表达式可以反过来应用到另一项，从而得到一个完全指定的逻辑公式，如下所示：

$$\lambda y.Near(Bacaro, y)(Centro) \tag{15.33}$$
$$Near(Bacaro,Centro)$$

这种通用技术称为**柯里化(currying)**(Schonfinkel, 1924)[1]，是一种将带有多个论元的谓词转换为单个论元谓词序列的方法。正如我们在第 16 章中所展示的，λ符号提供了一种方法，当谓词的论元没有全部作为解析树中谓词的子项一起出现时，可以将它们递增地收集到谓词中。

15.3.4. 一阶逻辑的语义

在 FOL 知识库中表示的各种对象、属性和关系通过它们与被建模的外部世界中的对象、属性和关系的对应来获得它们的含义。我们可以使用第 15.2 节介绍的模型理论方法来实现这一点。回想一下，这种方法使用简单的集合理论概念来提供从含义表示中的表达式到正在建模的事件状态的真条件映射。我们可以将这种方法应用到 FOL 中，遍历图 15.3 中的所有元素，并指定每个元素应如何解释。

我们可以从断言我们世界中的对象(FOL 术语)表示领域中的元素开始，并断言原子公式是作为属性的领域元素集捕获的，还是作为关系的元素元组集合捕获的。作为一个例子，考虑以下情况:

(15.34) Centro is near Bacaro.

要在 FOL 中捕获这个例子的含义，需要识别与句子中的各种语法元素对应的术语和谓词，并创建逻辑公式，以捕获句子中的单词和句法所暗示的关系。对于这个例子，这样的努力可能会产生如下结果:

$$Near(Centro,Bacaro) \tag{15.35}$$

这个逻辑公式的含义取决于术语 Centro 和 Bacaro 所表示的领域元素是否包含在由当前模型中的谓词 Near 所表示的关系所表示的元组中。

对包含逻辑连接词的公式的解释是基于公式中成分的含义和它们所包含的连接词的含义的结合。图 15.4 给出了图 15.3 中所示逻辑操作符的解释。

∧(and，与)和¬(not，非)操作符的语义相当简单，至少与相应英语术语的某些含义相关。然而，值得指出的是，∨(or，或)操作符与相应的英文单词不以同样的方式析取，而且⇒(imply,蕴涵)操作符只是松散地以任何有关含义或因果的常识概念为基础。

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Longrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

图 15-4：真值表给出了各种逻辑连接词的语义

[1] Currying 是标准的术语，尽管 Heim 和 Kratzer(1998)提出了一个关于 Currying 的术语“Schonkfinkelization”的有趣的争论，因为 Curry 后来建立在 Schonfinkel 的工作之上。

我们需要处理的最后一点涉及变量和量词。回想一下，在我们的基于集合的模型中没有变量，只有领域的元素和它们之间的关系。我们可以利用第 15.3.2 节介绍的替换概念，为有变量的公式提供基于模型的解释。如果变量的替换项的结果公式在模型中是真实的，那么涉及∃的公式是真实的，。包含∀的公式在所有可能的替换下都必须是正确的。

15.3.5. 推理

一种含义表示语言必须支持推理，以便向知识库中添加有效的新命题，或者确定没有明确包含在知识库中的命题的真实性(第 15.1 节)。本节简要讨论一种由 FOL 提供的最广泛实施的推理方法。

取式推理(Modus ponens)是一种推理形式，与非正式的 if-then 推理相对应。我们可以抽象地定义取式推理如下，其中α和β应作为 FOL 公式:

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta} \tag{15.36}$$

这样的模式表明，可以通过某种形式的推理从该行上面的公式推导出该行下面的公式。取式推理表明，如果一个蕴涵规则的左手边是真的，那么右手边的规则就可以推断出来。在接下来的讨论中，我们将把一个蕴涵的左手边称为前提，右手边称为结果。

对于取式推理的典型用法，考虑下面的例子，它使用了上一节的规则:

$$\frac{VegetarianRestaurant(Leaf) \qquad \forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)}{Serves(Leaf, VegetarianFood)} \tag{15.37}$$

在这里，公式 VegetarianRestaurant(Leaf)与规则的前提相匹配，因此我们可以用取式推理的方法得出 Serves(Leaf ,VegetarianFood)的结论。

取式推理的实际应用有两种方式：正向链和反向链。

在**正向链(forward chaining)**系统中，取式推理使用正是如上所述的方式。当单个事实添加到知识库时，使用取式推理来触发所有适用的蕴涵规则。在这种安排中，只要向知识库中添加一个新的事实，就会找到并应用所有适用的蕴涵规则，每一个都导致向知识库中添加新的事实。这些新的命题反过来也可以用来提出适用于它们的蕴涵规则。这一过程继续进行，直到无法推断出进一步的事实为止。

正向链方法的优点是，事实将在需要时出现在知识库中，因为从某种意义上说，所有推理都是提前执行的。这可以大大减少回答后续查询所需的时间，因为它们都应该是简单的查询。这种方法的缺点是，可能会推断和存储永远不需要的事实。

在**反向链(backward chaining)**系统中，取式推理用逆向思维的方式来证明被称为查询的特定命题。第一步是通过确定查询公式是否存在于知识库中来判断它是否为真。如果不是，那么下一步就是搜索知识库中存在的适用的蕴涵规则。适用的规则是规则的结果与查询公式匹配的规则。如果存在这样的规则，那么只要其中任何一个规则的前提都可以证明为真，就可以证明该查询。这可以通过将前提作为新查询进行反向连接来递归地执行。Prolog 编程语言是一个实现这种策略的反向链系统。

为了了解这是如何工作的，让我们假设我们被要求验证命题 Serves(Leaf ,VegetarianFood)的真实性，假设公式(15.37)中的直线以上给出的事实。由于这个命题在知识库中不存在，因此开始搜索一个适用的规则，结果是上面给出的规则。在用常数 Leaf 代替变量 x 之后，我们的下一个任务是证明这个规则的前提，VegetarianRestaurant(Leaf)，当然，这是我们给定的事实之一。

请注意，区分从查询到已知事实的反向链推理和从已知结果到未知前提的反向推理是非常重要的。具体地说，通过反向推理，我们的意思是，如果一个规则的结论为真，我们就假设前提也为真。例如，让我们假设我们知道 Serves(Leaf ,VegetarianFood)为真。由于这个事实与我们的规则的结果相匹配，我们可能会反向推理到 VegetarianRestaurant(Leaf)为真的结论。

虽然反向链是一种合理的推理方法，但反向推理是一种无效的、但通常有用的似是而非的推理形式。从结论到前提的似是而非的推理被称为**诱导法(abduction)**，正如我们在第 22 章中所展示的那样，在解释人们在分析扩展话语时做出的许多推论时，通常是有用的。

虽然向前推理和向后推理都是可靠的，但两者都不是**完备的(complete)**。这意味着有一些有效的推论是系统不能单独使用这些方法来发现的。幸运的是，有一种替代推理技术，称为**归结法(resolution)**，它是可靠的和完备的。不幸的是，基于归结法的推理系统在计算代价上比正向或反向链系统要昂贵得多。因此，在实践中，大多数系统使用某种形式的连接，并给知识库开发人员增加了负担，以允许得出必要的推断的方式对知识进行编码。

15.4. 事件和状态表示

我们希望捕获的许多语义包括状态和事件的表示。状态是在一段时间内保持不变的条件或属性，事件表示某些事件状态的变化。状态和事件的表示都可能涉及大量的参与者、内容(props)、时间和地点。

到目前为止，我们使用的事件和状态的表示都是由单个谓词组成的，这些谓词具有与给定示例相关联的所有角色所需的论元。例如，Leaf served vegetarian fare 的表示由一个谓词组成，该谓词带有执行服务的实体和被服务的事物的论元。

$$Serves(Leaf, VegetarianFare) \tag{15.38}$$

这种方法假设用于表示事件动词的谓词具有与该动词的句法子范畴框架中存在的相同数量的论元。不幸的是，情况显然并非总是如此。来看看动词 eat 的以下例句:

(15.39) I ate.

(15.40) I ate a turkey sandwich.

(15.41) I ate a turkey sandwich at my desk.

(15.42) I ate at my desk.

(15.43) I ate lunch.

(15.44) I ate a turkey sandwich for lunch.

(15.45) I ate a turkey sandwich for lunch at my desk.

显然，为表示 eat 意义的谓词选择正确数量的论元是一个棘手的问题。这些例子以一组不同的句法形式、位置和组合来介绍五个不同的论元或角色。不幸的是，FOL 中的谓词有固定的**元数(arity)**——它们接受固定数量的论元。

为了解决这个问题，我们引入了事件变量(event variable)的概念，以允许我们对特定事件作出断言。为此，我们可以重构事件谓词，使其具有一个存在量化变量作为其第一个也是唯一的论元。使用这个事件变量，我们可以引入其他谓词来表示我们拥有的关于事件的其他信息。这些谓词将事件变量作为它们的第一个论元，相关的 FOL 术语作为它们的第二个论元。下面的公式用我们前面讨论过的 15.40 的含义表示来说明这个方案。

$$\exists e\ Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TurkeySandwich)$$

这里，量化变量 e 代表吃的事件，用于将事件谓词与通过命名角色 Eater 和 Eaten 提供的核心信息绑定。为了处理更复杂的示例，我们只需添加额外的关系来捕获所提供的信息，对于 15.45,如下所示：

$$\exists e\ Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TurkeySandwich) \wedge Meal(e, Lunch) \wedge Location(e, Desk) \tag{15.46}$$

这种事件表示被称为**新戴维森(neo Davidsonian)**事件表示(Davidson 1967, Parsons 1990)，以哲学家 Donald Davidson 引入事件变量的概念(Davidson, 1967)命名。总而言之，在新戴维森主义的事件表示方法中:

- 用谓词捕获事件，谓词以单个事件变量作为论元。
- 没有必要为一个给定的谓语指定一个固定数量的论元;相反，只要输入中提供的角色和填充项可以被粘在一起。
- 除了输入中提到的角色外，没有更多的角色是假定的。

•可以满足共享相同谓词的密切相关输入之间的逻辑连接，而不需要额外的推理。

这种方法仍然给我们留下了一个问题，即确定所需的谓词集合，以表示与特定事件(如 Eater 和 Eaten)相关的角色，以及更一般的概念(如位置和时间)。我们将在第十章更详细地回到这个问题。

15.4.1. 时间表示

在我们对事件的讨论中，我们没有认真处理捕获所表示事件应该发生的时间的问题。将这些信息以一种有用的形式表示出来就是**时序逻辑(temporal logic)**的领域。本讨论介绍了时序逻辑的最基本问题，并简要讨论了人类语言传递时间信息的方法，其中包括**时态逻辑(tense logic)**，动词时态传递时间信息的方式。关于时间表达式表示和分析的鲁棒方法的更详细的讨论将在第 17 章中介绍。

最直接的时间理论认为，时间不可阻挡地向前流动，事件与时间上的点或间隔有关，如在时间轴上。我们可以将不同的事件排列在时间轴上;如果时间从第一个事件流向第二个事件，则一个事件先于另一个事件。在大多数理论中，伴随这些概念的是当前时刻的概念。将这一概念与时间顺序关系相结合，就产生了我们熟悉的过去、现在和未来的概念。

许多方案都可以表示这种时间信息。这里展示的是一个相当简单的，它停留在我们一直在追求的具体化事件的 FOL 框架内。考虑以下例子:

(15.47) I arrived in New York.

(15.48) I am arriving in New York.

(15.49) I will arrive in New York.

这些句子都指同一种事件，只是动词的时态不同。在我们当前的事件表示方案中，所有三个事件都将共享以下类型的表示，它缺乏任何时间信息:

$$\exists e Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \quad (15.50)$$

动词的时态提供的时间信息可以利用预测关于事件的附加信息变量 e。具体地说,我们可以添加时间变量代表间隔对应于事件,事件的终点,时间谓词有关这个终点当前时间的动词的时态。对于我们的 arriving 例子，这种方法产生了以下表示:

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge EndPoint(i,n) \wedge Precedes(n,Now)$$

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge MemberOf(i,Now)$$

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge EndPoint(i,n) \wedge Precedes(Now,n)$$

该表示引入了一个变量来表示与事件相关的时间间隔，以及一个变量来表示该时间间隔的结束。前面的二元谓词 Precedes 表示第一个时间点论元在时间上先于第二个时间点;常量 Now 指的是当前时间。对于过去的事件，间隔的结束点必须在当前时间之前。类似地，对于未来的事件，当前时间必须在事件结束之前。对于当前发生的事件，当前时间包含在事件间隔内。

不幸的是，简单的动词时态和时间点之间的关系绝不是直接的。考虑以下例子:

(15.51) Ok, we fly from San Francisco to Boston at 10.

(15.52) Flight 1390 will be at the gate an hour now.

在第一个例子中，动词 fly 的现在时指将来发生的事，而在第二个例子中，将来时指过去发生的事。

当我们考虑其他动词时态时，会出现更多的复杂情况。考虑以下例子:

(15.53) Flight 1902 arrived late.

(15.54) Flight 1902 had arrived late.

尽管两者都涉及过去的事件，但以同样的方式来描述它们似乎是错误的。第二个例子似乎有另一个未命名的事件潜伏在背景中(例如，1902 航班已经到达晚了，这时发生了另一件事)。为了解释这种现象，Reichenbach(1947)引入了**参考点(reference point)**的概念。在我们简单的时间模式中，当前时刻等同于**说话(utterance)**的时间，并用作事件发生时的参考点(在之前、在此时或之后)。在 Reichenbach 的方法中，

参考点的概念是分离的说话时间和事件时间。下面的例子说明了这种方法的基础:

(15.55) When Mary's flight departed, I ate lunch.

(15.56) When Mary's flight departed, I had eaten lunch.

在这两个例子中，吃饭(eating)事件都发生在过去，也就是说，在说话之前。然而，在第一个例子中的动词时态表明吃饭事件在航班起飞时就开始了，而第二个例子表明吃饭是在航班起飞前完成的。因此，用 Reichenbach 的术语来说，起飞(departure)事件指定了参考点。这些事实可以通过与吃饭和起飞事件相关的额外限制得到满足。在第一个示例中，参考点在吃饭事件之前，在第二个示例中，吃饭在参考点之前。图 15.5 展示了 Reichenbach 对初级英语时态的研究方法。练习 15.6 要求你用 FOL 写出这些例子。

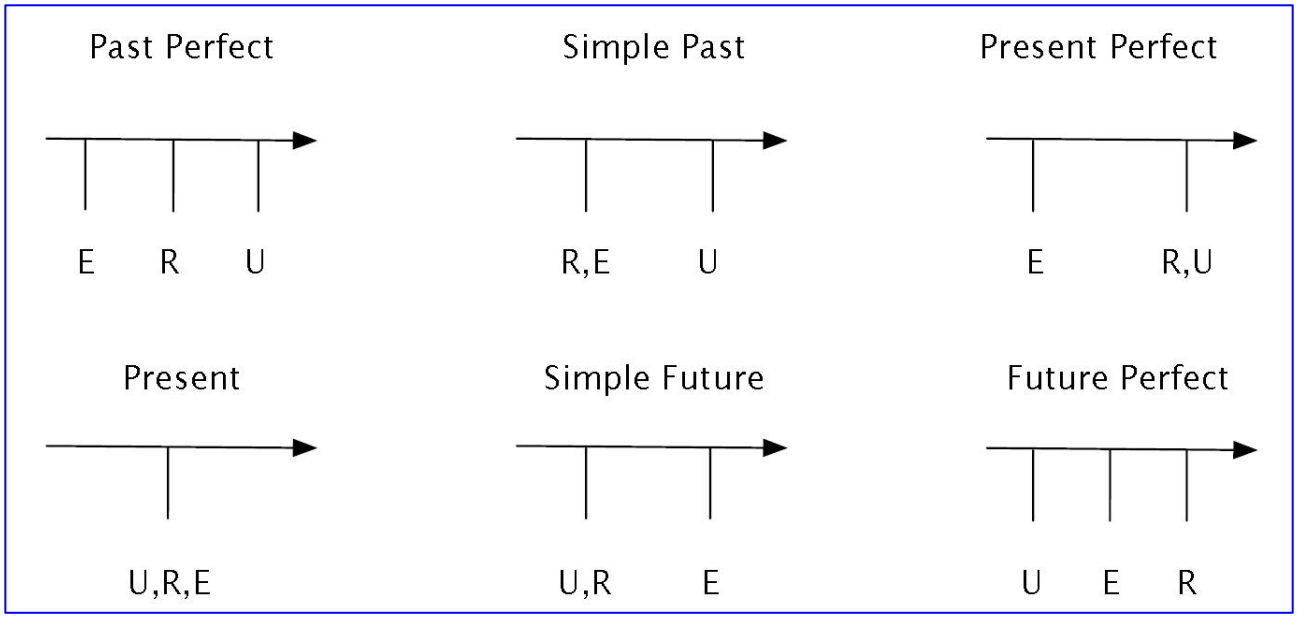


图 15-5：Reichenbach 的研究方法应用于各种英语时态

图注：在这些图中，时间从左向右流动，E 表示事件时间，R 表示参考时间，U 表示说话时间。

这次讨论集中在过去、现在和未来的广义概念上，以及它们是如何被各种英语动词时态所表示的。当然，语言还有很多其他的方式来传递时间信息，包括时间表达式:

(15.57) I'd like to go at 6:45 in the morning.

(15.58) Somewhere around noon, please.

正如我们在第 17 章中所展示的，这种时态表达式的语法对于信息提取和问题回答应用程序具有相当大的实际重要性。

最后，我们应该注意到，系统的概念组织反映在这样的例子中。特别地，时间表达在英语中经常用空间术语来表达，正如这些例子中 at、in、somewhere 和 near 的各种用法所说明的(Lakoff 和 Johnson 1980, Jackendoff 1983)。像这样的隐喻组织，一个领域用另一个领域系统地表达，在世界上的语言中非常常见。

15.4.2. 体

在最后一节中，我们讨论了用描述事件说话的时间来表示事件时间的方法。在本节中,我们强调**体(aspect)**的概念,涉及聚类相关的话题,包括一个事件是否已经结束或正在进行,它是否被概念化为在某个时间点或某个时间间隔内发生，以及世界上是否有任何特定状态是由于此而发生的。基于这些概念和相关概念，传统上将事件表达式分为以下四个示例中所示的四个通用类：

(静态体)**Stative:** I know my departure gate.

(活动体)**Activity:** John is flying.

(完成体)**Accomplishment:** Sally booked her flight.

(达成体)**Achievement:** She found her gate.

尽管 Aristotle(亚里士多德)讨论了该分类的最早版本，但此处介绍的版本归功于 Vendler(1967)。

静态体(stative)表达式表示事件参与者在给定时间点具有特定属性或处于某种状态的概念。因此，这些表达式可以被认为是在单个时间点上捕捉世界的一个方面。考虑下面的 ATIS 示例。

(15.59) I like Flight 840 arriving at 10:06.

(15.60) I need the cheapest fare.

(15.61) I want to go first class.

在这样的例子中，由主语指示的事件参与者可以被视为在特定的时间点对某件事有经验。经验者是否在之前或将来处于同样的状态是未知的。

活动体(activity)表达式描述由参与者承担的事件，并且没有特定的终点。与静态体不同，活动体被看作是在一定时间段内发生的，因此与单个时间点无关。考虑以下例子:

(15.62) She drove a Mazda.

(15.63) I live in Brooklyn.

这两个例子都说明了主语在一段时间内正在或已经从事动词所指定的活动。

最后一个体类，**达成体表示(achievement expression)**，与**完成体**相似，因为这些表示会导致一种状态。考虑以下:

(15.64) She found her gate.

(15.65) I reached New York.

与完成体不同，达成体事件被认为是瞬间发生的，不等同于任何特定的活动导致的状态。更具体地说，这些例子中的事件之前可能有扩展的 searching 或者 traveling 事件，但与 find 和 reach 直接对应的事件被认为是**时间点(points)**，而不是**时间段(intervals)**。

注意，由于完成体和达成体都是导致状态的事件，所以它们有时被描述为单一体类的子类。这种组合类的成员被称为**终结体可能事件(telic eventities)**。

15.5. 描述逻辑

正如本章开头所指出的，已经发明了相当多的表示方案来捕捉语言说话(utterances)的含义。现在人们普遍认为，在这些不同的方法中表示的含义，原则上可以相对容易地在 FOL 中转换成等效的语句。困难在于，在许多这些方法中，语句的语义都是按程序定义的。也就是说，含义来自于解释它的系统。

描述逻辑(DL)是一种努力，旨在更好地指定这些早期结构化网络表示的语义，并提供一个特别适合于某些领域建模的概念框架。在形式上，术语描述逻辑是指一系列逻辑方法，它们对应于不同的 FOL 子集。对描述逻辑的可表达性施加的限制可确保各种关键类型推理的易处理性。但是，我们的重点是描述逻辑的建模方面，而不是计算复杂性问题。

在使用描述逻辑为应用领域建模时，重点是表示有关类别、属于那些类别的个人以及这些个人之间可以保持的关系的知识。组成特定应用领域的一组类别或概念称为它的**专业术语(terminology)**。知识库中包含专业术语的部分传统上称为 **TBox**。这与包含有关个人事实的 **ABox** 相反。该专业术语通常被安排到一个称为**本体知识体系(ontology)**的层次结构中，该本体捕获类别之间的子集/超集关系。

回到我们前面的烹饪领域，我们使用一阶谓词表示领域概念，如 Restaurant(x);等效的 DL 省略了这个变量，因此餐厅类别被简单地写成 **Restaurant**[2]。为了捕捉特定的领域元素(如 Frasca)是餐厅这一事实，我们断言 Restaurant (Frasca)与在 FOL 中断言的方式基本相同。这些类别的语义的指定方式与前面第 15.2 节中介绍的完全相同：像 Restaurant 这样的类别只是表示一组作为餐馆的领域元素。

在特定域中指定了感兴趣的类别后，下一步就是将它们排列成层次结构。捕获专业术语中存在的层次关系有两种方法：我们可以直接断言与层次相关的类别之间的关系；或者也可以为概念提供完整的定义，然后依靠推理来提供层次关系。这些方法之间的选择取决于所使用类别的用途以及为许多自然发生的类别制定精确定义的可行性。我们将在这里讨论第一个选项，并在本节稍后部分返回到定义的概念。

为了直接指定层次结构，我们可以断言专业术语中适当概念之间的**包含(subsumption)**关系。

[2] DL 语句通常采用 sans serif 字体排版。在这里，我们将遵循这种惯例，在给出与 DL 语句等价的 FOL 时，回到我们的标准数学符号。

包含关系常写成 C⊑ D，读作 C 被 D 包含，意即类别 C 的所有成员也是类别 D 的成员。毫不奇怪，这个关系的形式语义是由一个简单的集合关系提供的；在集合 C 中的任何定义域元素也在集合 D 中。

在 TBox 中添加以下语句将断言所有的餐馆都是商业场所，此外，还有各种各样的餐馆子类型。

$$\text{Restaurant} \sqsubseteq \text{CommercialEstablishment} \tag{15.66}$$

$$\text{ItalianRestaurant} \sqsubseteq \text{Restaurant} \tag{15.67}$$

$$\text{ChineseRestaurant} \sqsubseteq \text{Restaurant} \tag{15.68}$$

$$\text{MexicanRestaurant} \sqsubseteq \text{Restaurant} \tag{15.69}$$

像这样的本体知识体系通常用图 15.6 所示的图来说明，其中包含关系由表示类别的节点之间的连接来指示。

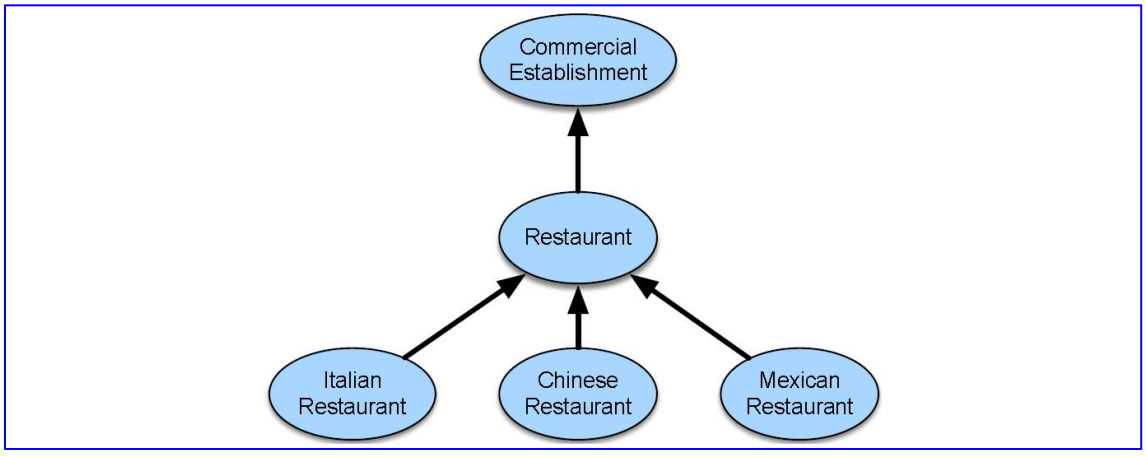


图 15-6：一组包含关系的图形网络表示

注意，正是这样的语义网络图的模糊性质促使了描述逻辑的发展。例如，从该图中我们无法确定给定的类别集合是详尽的还是不相交的。也就是说，我们无法确定这些是我们在该领域中要处理的所有餐厅，还是可能还有其他餐厅。我们也无法确定单个餐厅是否必须仅属于这些类别之一，或者是否有可能(例如，餐厅既可以是意大利餐厅又可以是中国餐厅)。上面给出的 DL 语句的含义更加透明； 他们只是断言类别之间的一系列归属关系，而没有关于覆盖范围或相互排斥的主张。

如果一个应用程序需要覆盖范围和不相交的信息，那么这些信息必须明确地说明。捕获这类信息的最简单方法是使用否定和析取(disjunction)操作符。例如，下面的断言会告诉我们，中国餐馆不可能也是意大利餐馆。

$$\text{ChineseRestaurant} \sqsubseteq \textbf{not}\ \text{ItalianRestaurant} \tag{15.70}$$

指定一组包含一个类别的子概念可以通过析取来实现，如下所示:

$$\text{Restaurant} \sqsubseteq (\textbf{or}\ \text{ItalianRestaurant}\ \text{ChineseRestaurant}\ \text{MexicanRestaurant}) \tag{15.71}$$

图 15.6 所示的层次结构几乎没有告诉我们其中的概念。我们当然不知道什么是餐厅，更不用说意大利的、中国的、或者昂贵的。我们需要的是对成为这些类别中的一员意味着什么的额外断言。在描述逻辑中，这样的语句以被描述的概念和领域中的其他概念之间的关系的形式出现。为了与它在结构化网络表示中的起源保持一致，描述逻辑中的关系通常是二元的，通常被称为角色或角色-关系。

为了了解这种关系是如何运作的，让我们考虑一下本章前面讨论的一些关于餐馆的事实。我们将使用 hasCuisine 关系来获取关于餐馆提供哪种食物的信息，使用 hasPriceRange 关系来获取特定餐馆的价格倾向。我们可以使用这些关系来更具体地说明我们不同类别的餐馆。让我们从我们的意大利餐厅概念开始。作为第一种近似说法，我们可能会说一些没有争议的事情，比如意大利餐馆提供意大利美食。为了捕获这些概念，让我们首先添加一些新的概念到我们的专业术语中来代表各种各样的美食。

$$\text{MexicanCuisine} \sqsubseteq \text{Cuisine} \qquad \text{ExpensiveRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{ItalianCuisine} \sqsubseteq \text{Cuisine} \qquad \text{ModerateRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{ChineseCuisine} \sqsubseteq \text{Cuisine} \qquad \text{CheapRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{VegetarianCuisine} \sqsubseteq \text{Cuisine}$$

接下来，让我们修改我们早期版本的意大利餐厅，以捕捉美食信息。

$$\text{ItalianRestaurant} \sqsubseteq \text{Restaurantu} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.72}$$

正确读取这个表达式的方法是，类别意大利餐厅中的个体既包含在类别餐厅中，也包含在存在从句定义的未命名类中——提供意大利美食的实体集合。用 FOL 表示的等价表述是：

$$\forall x \mathit{ItalianRestaurant}(x) \rightarrow \mathit{Restaurant}(x) \tag{15.73}$$
$$\wedge (\exists y \mathit{Serves}(x;y) \wedge \mathit{ItalianCuisine}(y))$$

该 FOL 转换应使上面给出的 DL 断言做什么和不做什么都很清楚。特别是，他们并没有说被归类为意大利餐馆的领域实体不能被加入其他关系，例如价格昂贵甚至是中国美食。至关重要的是，他们对我们知道确实提供意大利美食的领域实体并没有多说。实际上，对 FOL 转换的检查清楚地表明，我们不能根据其特征推断任何新实体都属于该类别。我们所能做的最好的事情就是推断有关餐厅的新事实，我们被明确告知餐厅是该类别的成员。

当然，推断具有特定特征的个体的类别成员资格是我们需要支持的常见且关键的推理任务。这使我们回到了用专业术语创建层次结构的另一种方法：实际上以类别成员资格的必要和充分条件的形式提供对我们正在创建的类别的定义。在这种情况下，我们可能会明确地为 ItalianRestaurant 提供一个定义，即提供意大利美食的餐厅，将 ModerateRestaurant 提供为价格范围适中的餐厅。

$$\text{ItalianRestaurant} \equiv \text{Restaurant} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.74}$$

$$\text{ModerateRestaurant} \equiv \text{Restaurant} \sqcap \text{hasPriceRange:ModeratePrices} \tag{15.75}$$

虽然我们前面的陈述为这些类别的成员提供了必要条件，但这些陈述同时提供了必要和充分条件。

最后，让我们来看看表面上类似的素食餐厅的例子。显然，素食餐厅是提供素食的餐厅。但他们不只是提供素食，这是他们提供的全部。我们可以通过在我们之前对 VegetarianRestaurants 的描述基础上增加一个全称量词的形式来适应这种限制，如下所示:

$$\text{VegetarianRestaurant} \equiv \text{Restaurant} \tag{15.76}$$
$$\sqcap \exists \text{hasCuisine:VegetarianCuisine}$$
$$\sqcap \forall \text{hasCuisine:VegetarianCuisine}$$

推理

与描述逻辑对类别、关系和个体的关注并行的是对逻辑推理的受限子集的处理关注。DL 推理系统强调包含和实例检查这两个紧密耦合的问题，而不是使用 FOL 允许的全部推理。

包含(Subsumption)，作为一种推理形式，是基于专业术语中断言的事实来确定两个概念之间是否存在超集/子集关系的任务。相应地，**实例检查(instance checking)**要求(ask)，根据我们所知道的关于个体和专业术语的事实，个体是否可以成为特定类别的成员。包含和实例检查背后的推理机制不仅仅是检查专业术语中显式声明的包含关系。它们必须明确地使用有关专业术语的关系信息进行推理，以推断出适当的包含和成员关系。

回到我们的餐厅领域，让我们使用以下语句添加一种新型餐厅：

$$\text{IlFornaio} \sqsubseteq \text{ModerateRestaurant} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.77}$$

鉴于此，我们可能会问，IlFornaio 连锁餐厅是否可以归类为意大利餐厅或素食餐厅。更准确地说，我们可以对我们的推理系统提出以下问题:

$$\text{IlFornaio} \sqsubseteq \text{ItalianRestaurant} \tag{15.78}$$

$$\text{IlFornaio} \sqsubseteq \text{VegetarianRestaurant} \tag{15.79}$$

第一个问题的答案是肯定的，因为 IlFornaio 符合我们为 ItalianRestaurant 分类所指定的标准：这是一家餐厅，因为我们将其明确分类为 ModerateRestaurant，它是 Restaurant 的子类型，并且满足 has.Cuisine 类型限制，因为我们已经直接断言了。

第二个问题的答案是否定的。回想一下，我们对素食餐厅的标准包含两个要求：它必须提供素食，而且它只能提供素食。我们目前对 IlFornaio 的定义在这两个方面都失败了，因为我们还没有断言任何声称 IlFornaio 提供素食的关系，而我们断言的关系 hasCuisine.ItalianCuisine 与第二个标准相矛盾。

基于基本包含推论的相关推理任务是，在给定有关专业术语类别的事实的情况下，得出该专业术语的

蕴涵层次结构(implied hierarchy)。该任务大致对应于将包含操作符重复应用到成对的专业术语中。给定我们当前的语句集合，可以推断出图 15.7 中所示的扩展层次结构。您应该使自己确信，该图仅包含所有包含的连接，而根据我们的当前知识，这些连接应该存在。

实例检查的任务是确定一个特定的个体是否可以被分类为特定类别的成员。这个过程以关系和明确的直言陈述的形式获取已知的关于特定个体的信息，然后将这些信息与已知的关于当前专业术语的信息进行比较。然后，它返回个体可以归属的最特定类别的列表。

作为一个分类问题的例子，假设我们被告知是一家提供意大利美食的餐厅。

Restaurant(Gondolier)

hasCuisine(Gondolier,ItalianCuisine)

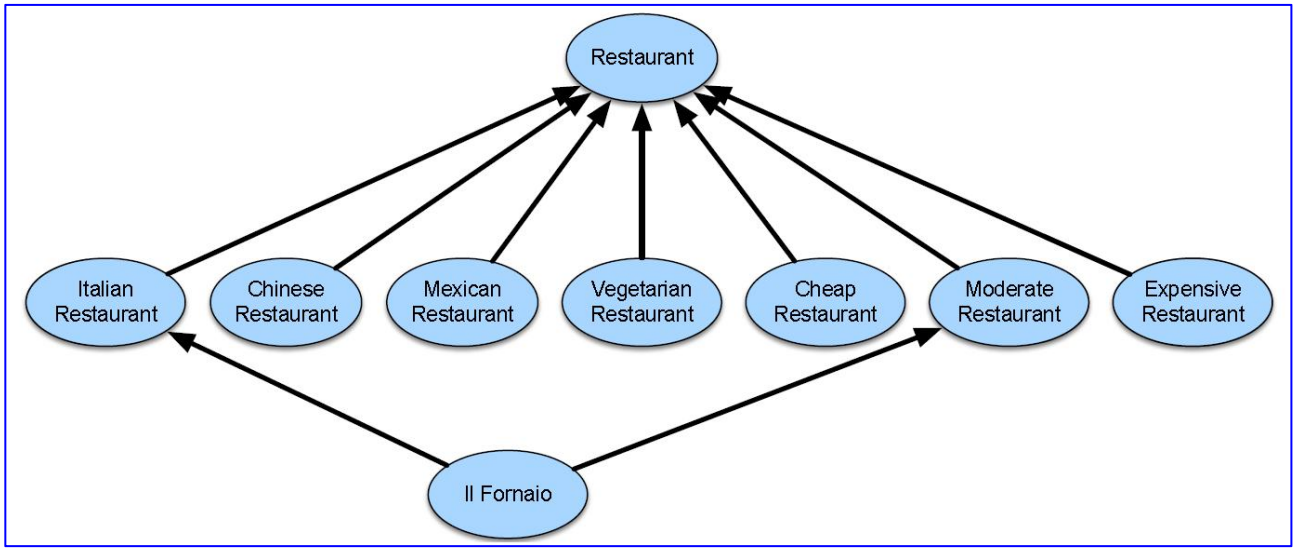


图 15-7：完整包含关系集的图形网络表示

图注：给定 TBox 中的当前断言集，餐饮领域中完整包含关系集的图形网络表示

在这里，我们被告知 Gondolier 这个术语所指的实体是一家餐厅，供应意大利食物。根据这个新信息和我们目前 TBox 的内容，我们可能会有理由问一下，这是一家意大利餐厅，是一家素食餐厅，还是价格适中。

假设前面给出了定义性声明，我们确实可以将 Gondolier 归类为意大利餐厅。也就是说，我们所获得的信息符合该类别成员资格的必要和充分条件。与 IlFornaio 类别一样，此个体也无法满足针对 VegetarianRestaurant 规定的条件。最后，Gondolier 可能还算是一家价格适中的餐厅，但由于我们对其价格一无所知，因此我们目前无法透露。这意味着在给定我们目前的知识的情况下，对查询 ModerateRestaurant(Gondolier)的答案将为假，因为它缺少所需的 hasPriceRange 关系。

包含、实例检查以及实际应用程序所需的其他类型的推理的实现，根据所使用的描述逻辑的表达性而不同。然而，对于即使是中等能力的描述逻辑，主要的实现技术是基于满足性方法的，而满足性方法又依赖于本章前面介绍的基于模型的底层语义。

网络本体语言和语义网

迄今为止，描述逻辑最引人注目的角色是作为语义网开发的一部分。语义网是一项持续的努力，它提供了一种正式指定网络内容语义的方法(Fensel 等人，2003)。这项工作的一个关键成分涉及为各种感兴趣的应用程序领域创建和部署本体。用来表示这些知识的含义表示语言是**网络本体语言(Web Ontology Language，OWL)** (McGuiness 和 van Harmelen, 2004)。OWL 包含了一个描述逻辑，大致对应于我们在这里描述的逻辑。

15.6. 总结

本章介绍了**含义**的**表示**方法。以下是本章的一些重点内容:

•在计算语言学中，一个主要的意义方法涉及到创建**形式含义表示**，以捕获语言输入含义的相关内容。这些表示的目的是弥合语言与世界常识知识之间的鸿沟。

•定义这些表示的句法和语义的框架被称为**含义表示语言**。在自然语言处理和人工智能中使用了各种各样的这种语言。

•这样的表示需要能够支持语义处理的实际计算需求。这些要求包括:确定**命题的真值**、支持**无歧义表示**、表示**变量**、支持**推理**以及足够的**表现力**。

•人类语言有各种各样用来传递含义的特性。其中最重要的是传递**谓词-论元结构**的能力。

•**一阶逻辑**是一种易于理解的，易于计算的含义表示语言，它提供了含义表示语言所需的大部分功能。

•可以在 FOL 中捕获语义表示的重要元素，包括状态和事件。

•可以在 FOL 框架内捕获**语义网络**和**框架**。

•现代**描述逻辑**由完整的一阶逻辑的有用且易计算的子集组成。描述逻辑的最主要用途是语义网规范中使用的**网络本体语言(OWL)**。

15.7. 文献和历史说明

声明性含义表示在自然语言处理中的最早计算性使用是在问答系统的背景下(Green 等人 1961,Raphael 1968, Lindsey 1963)。这些系统对回答问题所需的事实采用了特别的表示。然后，问题被翻译成一种可以与知识库中的事实相匹配的形式。Simmons(1965)概述了这些早期的努力。

Woods(1967)研究了在回答问题时使用 FOL-like 表示，作为当时使用的临时表示的替代。Woods(1973)在具有里程碑意义的月球(Lunar)系统中进一步发展和扩展了这些想法。有趣的是，在 Lunar 中使用的表示既有真值条件，也有程序语义。Winograd(1972)在他的 SHRDLU 系统中采用了一种基于微规划语言的类似表示。

在此期间，对语言和记忆认知模型感兴趣的研究人员一直在研究各种形式的联想网络表示。Masterman(1957)是第一个将类似语义网络的知识表示方法用于计算的人，尽管语义网络通常被认为是 Quillian(1968)的作品。语义网络框架的大量工作是在这一时期进行的(Norman 和 Rumelhart 1975, Schank 1972, Wilks 1975c, Wilks 1975b, Kintsch 1974)。正是在这一时期，一些研究人员开始将 Fillmore 的案例角色概念(Fillmore, 1968)纳入他们的表述中。Simmons(1973)是最早将案例角色作为自然语言处理表示的一部分的使用者。

Woods(1975)和 Brachman(1979)为了弄清语义网络的真正含义而进行的详细分析，导致了许多更复杂的类网络语言的发展，包括 KRL (Bobrow 和 Winograd, 1977)和 KL-ONE (Brachman 和 Schmolze, 1985)。随着这些框架变得越来越复杂和定义得越来越好，很明显，它们是 FOL 的受限变体，外加专门的索引推理过程。在 Brachman 和 Levesque(1985)的著作中可以找到一个有用的论文集，涵盖了大部分这方面的工作。Russell 和 Norvig(2002)描述了这些代表性努力的现代视角。

在生成时代，语言学为自然语言句子分配语义结构的努力始于 Katz 和 Fodor(1963)的工作。它们简单的基于特征的表示方式的局限性，以及逻辑对当时许多语言学问题的自然契合，很快导致了各种谓词-参论元结构作为首选语义表示方式的采用(Lakoff 1972a, McCawley 1968)。随后 Montague(1973)将真值条件模型理论框架引入语言学理论，使形式句法理论与广泛的形式语义框架更加紧密地结合在一起。Dowty 等人(1981)和 Partee(1976)对 Montague 语义学及其在语言学理论中的作用有很好的介绍。

事件作为具体化对象的表示应归功于 Davidson(1967)。这里提出的方法明确地使事件的参加者身份化，这归功于 Parsons(1990)。

目前大多数时间推理的计算方法都是基于 Allen 的时间间隔概念(Allen, 1984);参见第 17 章。

Meulen(1995)提供了对时态和体的现代处理。Davis(1990)描述了 FOL 的使用来代表广泛的常识性领域的知识，包括数量、空间、时间和信念。

van Benthem 和 ter Meulen(1997)对逻辑和语言进行了全面的论述。Lyons(1977)是一个经典的语义文本。McCawley(1993)是一本不可或缺的教科书，涵盖了逻辑和语言的广泛主题。Chierchia 和 McConnell-Ginet(1991)也从语言学的角度广泛地涵盖了语义问题。Heim 和 Kratzer(1998)是从当前生成理论的角度写的一个更近期的文本。

15.8. 练习

15.1 Peruse your daily newspaper for three examples of ambiguous sentences or headlines. Describe the various sources of the ambiguities.

15.2 Consider a domain in which the word *coffee* can refer to the following concepts in a knowledge-based system: a caffeinated or decaffeinated beverage, ground coffee used to make either kind of beverage, and the beans themselves. Give arguments as to which of the following uses of coffee are ambiguous and which are vague.

1. I’ve had my coffee for today.
2. Buy some coffee on your way home.
3. Please grind some more coffee.

15.3 The following rule, which we gave as a translation for Example 15.25, is not a reasonable definition of what it means to be a vegetarian restaurant.

$\forall x \mathit{VegetarianRestaurant}(x) \Rightarrow \mathit{Serves}(x, \mathit{VegetarianFood})$

Give a FOL rule that better defines vegetarian restaurants in terms of what they serve.

15.4 Give FOL translations for the following sentences:

1. Vegetarians do not eat meat.
2. Not all vegetarians eat eggs.

15.5 Give a set of facts and inferences necessary to prove the following assertions:

1. McDonald’s is not a vegetarian restaurant.
2. Some vegetarians can eat at McDonald’s.

Don’t just place these facts in your knowledge base. Show that they can be inferred from some more general facts about vegetarians and McDonald’s.

15.6 For the following sentences, give FOL translations that capture the temporal relationships between the events.

1. When Mary’s flight departed, I ate lunch.
2. When Mary’s flight departed, I had eaten lunch.

15.7 On page 316, we gave the representation *Near*(*Centro,Bacaro*) as a translation for the sentence *Centro is near Bacaro*. In a truth-conditional semantics, this formula is either true or false given some model. Critique this truth conditional approach with respect to the meaning of words like *near*.

16. 计算语义和语义解析

Placeholder

(空空如也)

16.1. 计算语义概述

(空空如也)

17. 信息提取

I am the very model of a modern Major-General,
I've information vegetable, animal, and mineral,
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical...

Gilbert and Sullivan, *Pirates of Penzance*

假设您是一家跟踪航空公司股票的投资公司的分析师。您的任务是确定航空公司宣布的票价上涨与次日股票走势之间的关系(如果有的话)。股票价格的历史数据很容易获得，但航空公司的公告呢?您至少需要知道航空公司的名称、建议涨价的性质、公告的日期，以及其他航空公司可能的回应。幸运的是，这些都可以在像这样的新闻文章中找到:

Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lowercost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

本章介绍了从文本中提取有限种类语义内容的技术。这个**信息提取(information extraction，IE)**将嵌入文本中的非结构化信息转化为结构化数据，例如填充关系数据库以实现进一步处理。

我们从**关系提取(relation extraction)**的任务开始：在文本实体之间查找和分类语义关系。这些通常是二元关系，例如子女关系，就业关系，部分整体关系和地理空间关系。关系提取与填充关系数据库有着紧密的联系。确实，**知识图谱(knowledge graphs)**，结构化关系知识的数据集，是搜索引擎向用户提供信息的常见方式。

接下来，我们将讨论三个与事件相关的任务。**事件提取(event extraction)**是寻找这些实体参与的事件，例如，在我们的示例文本中，United 和 American 的票价增加以及 aid 和 cite 的报告事件。需要使用**事件共指(event coreference)**(第 22 章)来确定文本中提到的哪个事件引用的是同一个事件;在我们正在运行的示例中，increase 的两个实例和短语 the move 都指向同一个事件。

为了弄清文本中的事件何时(when)发生，我们提取**时间表达式(temporal expressions)**，例如一周中的几天(Friday 和 Thursday)，**相对表达式(relative expressions)**，例如 two days from now 或者 next year，以及时间诸如 3:30 PM，这些表达式必须标准化为特定的日历日期或一天中的时间，以便及时定位事件。在我们的示例任务中，这将使我们能够将 Friday 与 United's 宣布时间连接起来，将 Thursday 与前一天的加价连接起来，并产生一个时间表，在该时间表中，United's 宣布会跟随加价，而 American's 宣布会跟随这两个事件。

最后，许多文本描述了反复出现的陈规定型事件或情况。**模板填充(template filling)**的任务是在文档中查找此类情况并填写模板槽。这些槽填充符可以包含直接从文本中提取的文本段，也可以包括通过额外处理从文本元素中推断出的诸如时间、数量或本体(ontology)实体之类的概念。

我们的航空公司文字是这种陈规定型情况的一个例子，因为航空公司经常提高票价，然后等待竞争对手是否跟进。在这种情况下，我们可以将 United 确定为最初提高票价的领头航空公司，将票价定为$6，将 Thursday 为加价日期，将 American 定为紧随其后的航空公司，从而产生如下所示的填充模板。

```
FARE-RAISE ATTEMPT: [ LEAD AIRLINE:    UNITED AIRLINES
                      AMOUNT:          $6
                      EFFECTIVE DATE:  2006-10-26
                      FOLLOWER:        AMERICAN AIRLINES ]
```

17.1. 关系提取

让我们假设我们已经在示例文本中检测到了命名实体(可能使用了第 8 章的技术)，并想要辨别存在于检测到的实体之间的关系:

> Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flflights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

例如，文本告诉我们，Tim Wagner 是 American Airlines 的发言人，United 是 UAL Corp.的子公司，而 American 是 AMR 的子公司。这些二元关系是更一般的关系的实例，例如在新闻风格的文本中相当常见的部分(part-of)或雇佣(employs)关系。图 17.1 列出了 ACE 关系抽取评估中使用的 17 种关系，图 17.2 给出了一些样本关系。我们还可以提取更多特定于领域的关系，比如航线的概念。例如，从本文中我们可以得出结论，United 有飞往 Chicago、Dallas、Denver、和 San Francisco 航线。

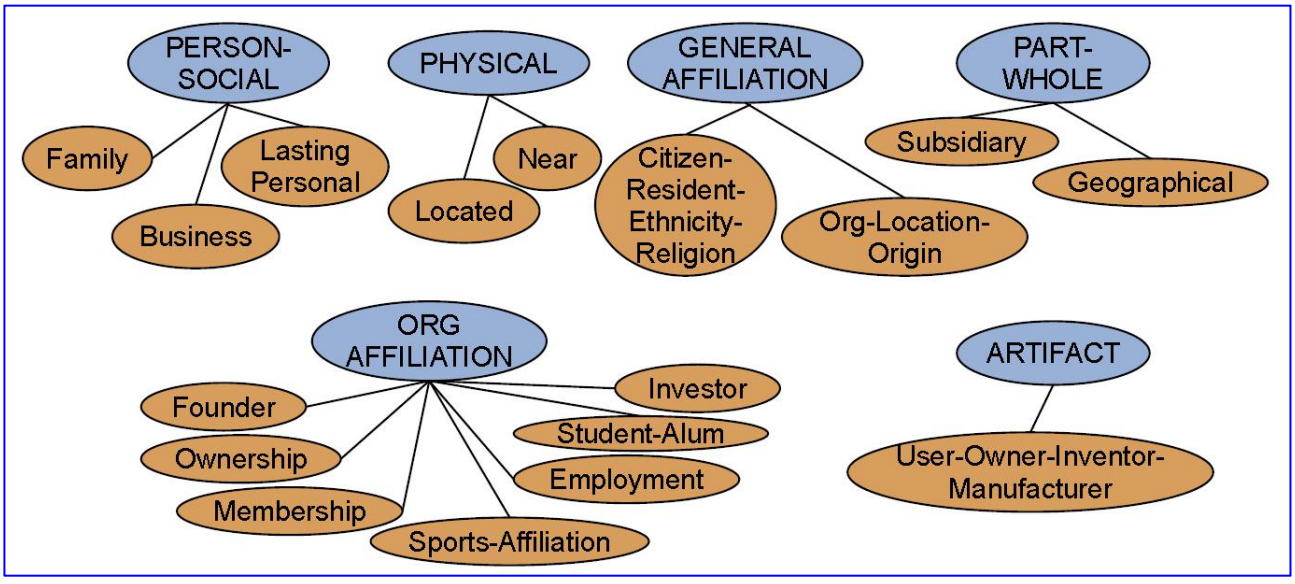


图 17-1：ACE 关系提取任务中使用的 17 个关系

Relations	Types	Examples
Physical-Located	PER-GPE	He was in Tennessee
Part-Whole-Subsidiary	ORG-ORG	XYZ, the parent company of ABC
Person-Social-Family	PER-PER	Yoko's husband John
Org-AFF-Founder	PER-ORG	Steve Jobs, co-founder of Apple...

图 17-2：带有示例的语义关系及其所涉及的命名实体类型

这些关系与我们在第 15 章中介绍的用于奠定逻辑形式含义基础的模型理论概念非常吻合。也就是说，一个关系由领域元素上的一组有序元组组成。在大多数标准的信息提取应用程序中，领域元素对应于文本中出现的命名实体，对应于共指消解产生的底层实体，或者对应于从领域本体中选择的实体。图 17.3 显示了一组实体和关系的基于模型的视图，这些实体和关系可以从我们正在运行的示例中提取出来。

请注意这个模型理论视图是如何包含 NER 任务的;命名实体识别对应于一类一元(unary)关系的识别。

也为许多其他领域定义了关系集。例如 UMLS，来自美国国家医学图书馆的统一医学语言系统，其网络定义了 134 个广泛的主题类别、实体类型和实体之间的 54 种关系，例如:

Domain	$\mathcal{D}$ = {a,b,c,d,e,f,g,h,i }
United, UAL, American Airlines, AMR	a,b,c,d
Tim Wagner	e
Chicago, Dallas, Denver, and San Francisco	f,g,h,i
Classes	
United, UAL, American, and AMR are organizations	Org = {a,b,c,d}
Tim Wagner is a person	Pers= {e}
Chicago, Dallas, Denver, and San Francisco are places	Loc = {f,g,h,i}
Relations	
United is a unit of UAL American is a unit of AMR	PartOf = {<a,b> , <c,d>}
Tim Wagner works for American Airlines	OrgAff = {<c,e>}
United serves Chicago, Dallas, Denver, and San Francisco	Serves= {<a,f >, <a,g>, <a,h>, <a,i>}

图 17-3：示例文本中基于模型的关系和实体视图

Entity	Relation	Entity
Injury	disrupts	Physiological Function
Bodily Location	location-of	Biologic Function
Anatomical Structure	part-of	Organism
Pharmacologic Substance	causes	Pathological Function
Pharmacologic Substance	treats	Pathologic Function

给出这样一个医学的句子:

(17.1) Doppler echocardiography can be used to diagnose left anterior descending artery stenosis in patients with type 2 diabetes

因此，我们可以提取 UMLS 关系:

Echocardiography, Doppler Diagnoses *Acquired stenosis*

Wikipedia 还提供了大量的关系，这些关系是从与某些 Wikipedia 文章相关的**信息框(infoboxes)**，结构化表格中得出的。例如，斯坦福大学的 Wikipedia 信息框包含结构化事实，例如 state="California"或 President="Marc Tessier-Lavigne"。这些事实可以转化为诸如 president-of 或者 located-in 之类的关系。或在称为**资源描述框架(Resource Description Framework，RDF)**的元语言(metalanguage)中建立关系。RDF **三元组(triple)**是实体关系实体的元组，称为**主-谓-宾(subject-predicate-object)**表达式。这是一个示例 RDF 三元组：

subject	predicate	object
Golden Gate Park	location	San Francisco

例如，众包的(crowdsourced) DBpedia (Bizer 等人，2009)是从 Wikipedia 派生出来的本体，包含超过 20 亿个 RDF 三元组。另一个来自维基百科信息箱的数据集 Freebase (Bollacker 等人，2008)，现在是 Wikidata Freebase 的一部分(Vrandecic 和 Krotzsch, 2014)，具有人们和他们的国籍、地点以及他们所包含的其他地点之间的关系。

WordNet 或其他本体提供有用的本体关系，表达词或概念之间的层次关系。例如，WordNet 在类之间有 is-a 或**上位词(hypernym)**关系：

Giraffe is-a ruminant is-a ungulate is-a mammal is-a vertebrate ...

(长颈鹿— 反刍动物 — 有蹄动物 — 哺乳动物 — 脊椎动物…)

WordNet 在个体和类别之间也有实例(Instance-of)关系，例如，San Francisco 与 city 之间的实例关系。提取这些关系是扩展或构建本体的重要步骤。

最后，有大量的数据集，其中包含利用它们的关系来做手工标记的句子，用于训练和测试关系提取器。TACRED 数据集(Zhang 等人，2017)包含关于特定人或组织的关系三元组的 106,264 个例子，以从年度 TAC 知识库人口(TAC KBP)挑战中抽取的新闻和 web 文本的句子标注。TACRED 包含 41 种关系类型(如

per:出生城市，org:子公司，org:成员，per:配偶)，加上一个无关系(no relation)标记;示例如图 17.4 所示(Zhang 等人，2017)。大约 80%的例子被标注为没有关系;有足够的负面数据对于训练有监督的分类器是很重要的。

Example	Entity Types & Label
Carey will succeed Cathleen P. Black, who held the position for 15 years and will take on a new role as chairwoman of Hearst Magazines, the company said.	PERSON/TITLE Relation: per:title
Irene Morgan Kirkaldy, who was born and reared in Baltimore, lived on Long Island and ran a child-care center in Queens with her second husband, Stanley Kirkaldy.	PERSON/CITY Relation: per:city of birth
Baldwin declined further comment, and said JetBlue chief executive Dave Barger was unavailable.	Types: PERSON/TITLE Relation: no relation

图 17-4：来自 TACRED 数据集的例句和标签

SemEval 2010 任务 8 也生成了一个标准数据集，用于检测名词性词(nominals)之间的关系(Hendrickxdr 等人，2009)。数据集有 10717 个示例，每个示例都有一对名词性词(无类型)，手工标记为 9 个直接关系中的一个，如产品-生产者(product-producer)(工厂生产套装)或组件-整体(component-whole)(我的公寓有一个大厨房)。

17.2. 关系提取算法

关系提取算法主要有五类：手写模式，有监督的机器学习，半监督(通过自举和远程监督)以及无监督。我们将在接下来的部分中介绍每一个。

17.2.1. 使用模式提取关系

最早也是最常见的关系提取算法是词汇-句法模式，最先由 Hearst (1992a)开发，因此通常称为 **Hearst 模式**。考虑下面的句子:

> Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use.

Hearst 指出，大多数人类读者不会知道什么是 Gelidium，但是无论它们是什么，他们都可以轻易地推断出 Gelidium 是一种红藻(的下位词(hyponym))。她建议以下词汇-句法模式：

$$NP_0 \text{ such as } NP_1\{, NP_2 : : : , (and|or)NP_i\}, i \geq 1 \quad (17.2)$$

蕴涵了以下语义：

$$\forall NP_i, i \geq 1, hyponym(NP_i, NP_0) \quad (17.3)$$

让我们推断：

$$\text{hyponym(Gelidium, red algae)} \quad (17.4)$$

NP {, NP}* {,} (and|or) other NP_H	temples, treasuries, and other important civic buildings
NP_H such as {NP,}* {(or|and)} NP	red algae such as Gelidium
such NP_H as {NP,}* {(or|and)} NP	such authors as Herrick, Goldsmith, and Shakespeare
NP_H{,} including {NP,}* {(or|and)} NP	common-law countries, including Canada and England
NP_H{,} especially {NP}* {(or|and)} NP	European countries, especially France, England, and Spain

图 17-5：用于找到上位词的词汇-句法模式

图注：使用{}标记可选性的手工建立的用于找到上位词的词汇-句法模式(Hearst 1992a，Hearst 1998)。

图 17.5 显示了 Hearst (1992a, 1998)提出的推断下位词关系的五种模式;我们已经展示了 NPH 作为父/下位词。基于模式的方法的现代版本通过添加命名实体约束来扩展它。例如，如果我们的目标是回答“谁在哪个组织担任什么职务?”，我们可以使用下面的模式:

PER, POSITION of ORG:
George Marshall, Secretary of State of the United States

PER (named|appointed|chose|etc.) PER Prep? POSITION
Truman appointed Marshall Secretary of State

PER [be]? (named|appointed|etc.) Prep? ORG POSITION
George Marshall was named US Secretary of State

手工构建的模式具有精度高的优点，可以针对特定的领域进行定制。另一方面，它们的召回率低，而且要为所有可能的模式创建它们需要大量的工作。

17.2.2. 通过监督学习提取关系

有监督的机器学习方法来提取关系遵循一个现在应该很熟悉的方案。选择一组固定的关系和实体，用这些关系和实体手工标注训练语料库，然后使用标注的文本训练分类器来标注看不见的测试集。

图 17.6 所示的最简单的方法是:(1)查找成对的命名实体(通常在同一个句子中)。(2)对每一对应用关系分类。分类器可以使用任何监督技术(逻辑回归，RNN，Transformer，随机森林等)。

可选的中间过滤分类器可以通过对给定的命名实体是否相关(通过任何关系)进行二元决策来加快处理速度。它的训练对象是直接从标注的语料库中的所有关系中提取的正示例，以及从没有标注关系的句子内实体对中生成的负示例。

```
function FINDRELATIONS(words) returns relations
  relations ← nil
  entities ← FINDENTITIES(words)
  forall entity pairs <e1, e2> in entities do
    if RELATED?(e1, e2)
        relations← relations+CLASSIFYRELATION(e1, e2)
```

图 17-6：查找并分类文本中实体之间的关系

基于特征的监督关系分类器。让我们考虑基于特征的分类器的样本特征(如逻辑回归或随机森林)，从这句话中对 American Airlines(提及 1，或 M1)和 Tim Wagner(提及 2，或 M2)之间的关系进行分类:

(17.5) **American Airlines**, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said

这些包括单词特征(如嵌入，或独热，无论词干与否):

•M1 和 M2 的中心词及其连接:

Airlines Wagner Airlines-Wagner

•M1 和 M2 中的词袋和 bigrams:

American, Airlines, Tim, Wagner, American Airlines, Tim Wagner

•特定位置的单词或 bigrams:

M2: -1 spokesman

M2: +1 said

•在 M1 和 M2 之间的词袋或 bigrams:

a, AMR, of, immediately, matched, move, spokesman, the, unit

命名实体特征：

•命名实体类型及其连接：

(M1: ORG, M2: PER, M1M2: ORG-PER)

•M1 和 M2 的实体级别(来自集合 NAME, NOMINAL, PRONOUN)：

M1: NAME [it or he would be PRONOUN]

M2: NAME [the company would be NOMINAL]

•论元之间的实体数(在本例中，AMR 为 1)：

句法结构是一个有用的信号，通常表示为实体之间的树中遍历的依存或成分语法路径。

•M1 和 M2 之间的成分路径：

NP ↑ NP ↑ S ↑ S ↓ NP

•依存树路径：

Airlines $\leftarrow_{subj}$ matched $\leftarrow_{comp}$ said $\rightarrow_{subj}$ Wagner

神经监督关系分类器 用于关系提取的神经模型同样将任务视为监督分类。让我们考虑一个应用于 TACRED 关系提取数据集和任务的典型系统(Zhang 等人，2017)。在 TACRED 中，我们会得到一个句子和其中的两个区段(span)：一个主语，指的是一个人或组织；一个宾语，指的是任何其他实体。任务是从 42 个 TAC 关系中分配一个关系，或者没有关系。

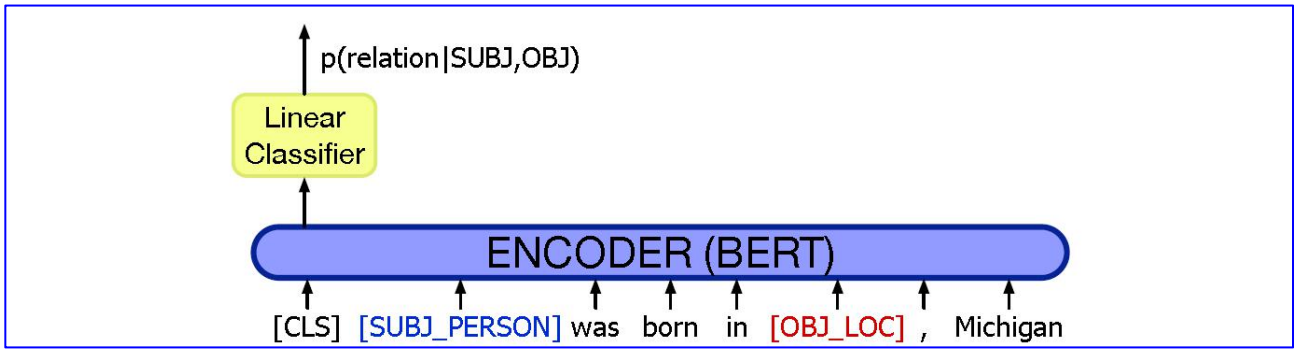


图 17-7：关系提取作为编码器上的线性层(本例中为 BERT)

图注：将输入中的 subject 和 object 实体替换为其 NER 标签(Zhang 等人 2017，Joshi 等人 2020)。

典型的 Transformer-encoder 算法(如图 17.7 所示)仅采用像 BERT 这样的经过预训练的编码器，并在句子表示的顶部添加一个线性层(例如 BERT [CLS]符记)，该线性层被微调为 1-of-N 分类器分配 43 个标签之一。BERT 编码器的输入被部分去混音(de-lexified)。在输入中，主语和宾语实体由其 NER 标记替换。这有助于防止系统过拟合单个词汇项目(Zhang 等人，2017)。当使用 BERT 型 Transformers 进行关系提取时，它有助于使用 RoBERTa (Liu 等人，2019)或 SPANbert (Joshi 等人，2020)等 BERT 版本，这些版本不包含由[SEP]符记分隔的两个序列，而是从单个长句序列中形成输入。

一般来说，如果测试集与训练集足够相似，并且有足够的手工标签数据，监督关系提取系统可获得较高的准确性。但是，标记一个庞大的训练集是非常昂贵的，监督模型是脆弱的：它们不能很好地泛化到不同的文本类型。由于这个原因，关系提取的很多研究都集中在我们接下来要讨论的半监督和非监督方法上。

17.2.3. 通过自举半监督提取关系

监督式机器学习假设我们有很多标签数据。不幸的是，这很贵。但是，假设我们只有一些高精度的**种子模式(seed patterns)**(如第 17.2.1 节中所述)，或者一些**种子元组(seed tuples)**。这足以**自举(bootstrap)**一个分类器!自举技术通过获取种子对中的实体，然后查找包含这两个实体的句子(在 web 上，或我们使用的任何数据集上)。从所有这些句子中，我们提取和概括实体周围的上下文，以学习新的模式。图 17.8 勾画了一个基本算法。

例如，假设我们需要创建一个航空公司/中转站(hub)配对的列表，并且我们只知道 Ryanair 在 Charleroi 有一个中转站。我们可以利用这个种子事实来发现新的模式，通过在语料库中发现其他提及这个关系的情况。我们搜索 Ryanair、Charleroi 和附近的 hub。也许我们能找到下面的句子:

(17.6) Budget airline Ryanair, which uses Charleroi as a hub, scrapped all weekend flights out of the airport.

(17.7) All flights in and out of Ryanair's hub at Charleroi airport were grounded on Friday...

(17.8) A spokesman at Charleroi, a main hub for Ryanair, estimated that 8000 passengers had already been affected.

```
function BOOTSTRAP(Relation R) returns new relation tuples
  tuples← Gather a set of seed tuples that have relation R
  iterate
    sentences← find sentences that contain entities in tuples
    patterns← generalize the context between and around entities in sentences
    newpairs← use patterns to grep for more tuples
    newpairs← newpairs with high confidence
    tuples← tuples + newpairs
  return tuples
```

图 17-8：从种子实体对中通过自举来学习关系

从这些结果中，我们可以使用实体提及(mention)之间的单词上下文，在提及一个之前的单词，在提及两个之后的单词，以及两个提及的命名实体类型，也许还有其他特征，来提取如下的一般模式:

从这些结果中，我们可以使用实体提及之间的词的上下文、第一次提及之前的词、第二次提及之后的词、两次提及的命名实体类型以及其他特征，来提取一般模式，例如:

/ [ORG], which uses [LOC] as a hub /

/ [ORG]'s hub at [LOC] /

/ [LOC], a main hub for [ORG] /

这些新模式可以用于搜索其他元组。

引导系统也为新的元组分配**置信度值(confidence values)**，以避免**语义漂移(semantic drift)**。在语义漂移中，一个错误的模式会导致语义漂移错误元组的引入，进而导致问题模式的产生以及所提取关系"漂移"的含义。考虑以下示例：

(17.9) Sydney has a ferry hub at Circular Quay.

如果作为一个正面的例子被接受，这个表达式可能会导致错误地引入元组<Sydney,CircularQuay>。基于此元组的模式可能会将更多错误传播到数据库中。

模式的置信度值基于两个因素之间的平衡：模式相对于当前元组的性能，以及模式在文档集合中产生的匹配项数量上的生产率。更正式地说，给定文档集合 D，当前元组 T 和提议的模式 p，我们需要跟踪两个因素：

•hits(p)：T 在 D 中查找时匹配的 T 中的元组集

•finds(p)：p 在 D 中找到的元组的总数

下面的方程平衡了这些考虑因素(Riloff 和 Jones, 1999)。

$$Conf_{RlogF}(p) = \frac{|hits(p)|}{|finds(p)|} log(|finds(p)|) \tag{17.10}$$

这个度量标准通常被归一化以产生一个概率。

我们可以通过结合所有与 D 中的元组匹配的模式 P'来支持对新元组的置信度的评估(Agichtein 和 Gravano，2000 年)。结合这种证据的一种方法是"noisy-or"。假定给定的元组由 P 中的模式子集支持，每个模式都有自己的置信度，如上所述。在 noise-or 模型中，我们做出两个基本假设。首先，要使一个提议的元组为假，其所有支持模式都必须是错误的；其次，其单个失败的来源都是独立的。如果我们将置信度度量宽松地视为概率，则任何单个模式 p 失败的概率为 1-Conf(p)；元组所有支持模式错误的可能性是其个别失败概率的乘积，留给我们的等式是我们对一个新的元组的置信度。

$$Conf(t) = 1 - \prod_{p \in P'} (1 - Conf(p)) \tag{17.11}$$

为自举过程中的新模式和元组接受设置保守的置信度阈值，有助于防止系统偏离目标关系。

17.2.4. 通过远程监督提取关系

尽管使用关系标签手工标记文本的生产成本很高，但是有一些方法可以找到训练数据的间接来源。**远程监督(distant supervision)**方法(Mintz 等人，2009)结合了自举和监督学习的优点。远程监督使用一个大型数据库来获取大量的种子样本(而不是少量的种子)，从所有这些样本中创建大量的噪声模式特征，然后将它们组合到一个监督分类器中。

例如，假设我们正在学习人们和他们出生城市之间的 place-of-birth 关系。在基于种子的方法中，我们可能只有 5 个例子可以开始。但基于维基百科的数据库，如 DBPedia 或 Freebase，有成千上万的例子来表达许多关系;包括超过 100,000 个 place-of-birth 的例子，(<Edwin Hubble, Marshfield>，<Albert Einstein, Ulm>，等等，)。下一步是在大量的文本上运行命名实体标记，Mintz 等人(2009)使用了维基百科上的 80 万篇文章，并提取出所有有两个与元组匹配的命名实体的句子，如下所示:

...Hubble was born in Marshfield...

...Einstein, born (1879), Ulm...

...Hubble's birthplace in Marshfield...

现在可以从这个数据中提取训练实例，每个相同的元组都有一个训练实例<relation, entity1, entity2>。因此，每一个都有一个训练实例:

<born-in, Edwin Hubble, Marshfield>

<born-in, Albert Einstein, Ulm>

<born-year, Albert Einstein, 1879> 等等。

然后我们可以应用基于特征或神经的分类。对于基于特征的分类，标准监督关系提取特征，如两个提及的命名实体标签，提及之间的词和依存路径，以及相邻的词。每个元组都有从许多训练实例中收集来的特征;单个训练实例诸如(<born-in,Albert Einstein, Ulm>)的特征向量将具有来自许多提及 Einstein 和 Ulm 的不同句子的词汇和语法特征。

因为远程监督有非常大的训练集，它也能够使用非常丰富的特征，这些特征是这些个体特征的结合。因此，我们将提取数以千计的模式，将实体类型与中间的单词或依存路径连接起来，就像以下这些:

PER was born in LOC

PER, born (XXXX), LOC

PER's birthplace in LOC

回到我们正在运行的例子，这个句子:

(17.12) American Airlines, a unit of AMR, immediately matched the move, spokesman Tim Wagner said

我们将学习丰富的连接特征，像这样:

M1 = ORG & M2 = PER & nextword= "said" & path= *NP* ↑ *NP* ↑ *S* ↑ *S* ↓ *NP*

结果是一个有监督的分类器，它有大量丰富的特征用于检测关系。由于不是每个测试句子都有一个训练关系，分类器还需要能够将一个示例标记为无关系(no-relation)。该标签的训练方法是随机选择在任何 Freebase 关系中不出现的实体对，为它们提取特征，并为每个这样的元组构建特征向量。最终的算法如图 17.9 所示。

```
function DISTANT SUPERVISION(Database D, Text T) returns relation classifier C
  foreach relation R
    foreach tuple (e1,e2) of entities with relation R in D
      sentences← Sentences in T that contain e1 and e2
      f ← Frequent features in sentences
      observations← observations + new training tuple (e1, e2, f, R)
  C← Train supervised classifier on observations
  return C
```

图 17-9：关系提取的远程监督算法

图注：神经分类器将跳过特征集 f。

远程监督与我们所研究的每一种方法都有优点。像监督分类，远程监督使用一个分类器与许多特征，并监督详细的手工创建的知识。与基于模式的分类器一样，它可对实体之间的关系使用高精度的证据。实际上，远程监控系统学习模式就像早期关系提取器手工构建的模式一样。例如，Snow 等人(2005)的 is-a 或上位词抽取系统使用 WordNet 中成对的上位词/下位词 NP 作为远程监督，然后从大量文本中学习新模式。他们的系统精确地引入了 Hearst（1992a）的 5 种原始模板模式，但是还引入了 70,000 种其他模式，包括以下 4 种：

NP_H like NP	*Many hormones like leptin...*
NP_H called NP	*...using a markup language called XHTML*
NP is a NP_H	Ruby is a programming language...
NP, a NP_H	*IBM, a company with a long...*

同时使用大量功能的能力意味着，与基于种子的系统中模式的迭代扩展不同，它没有语义漂移。像无监督分类一样，它不使用标记的文本训练语料库，因此它对训练语料库中的体裁问题不敏感，并且依赖大量的未标记数据。远程监督还具有以下优势：可以创建不需要元组的训练元组以便与神经分类器一起使用。

远程监控的主要问题是它往往产生低精度的结果，因此目前的研究集中在提高精度的方法。此外，远程监督只能帮助提取已经存在足够大的数据库的关系。要提取没有数据集的新关系或新领域的关系，必须使用纯无监督的方法。

17.2.5. 通过无监督提取关系

无监督关系提取的目的是在没有标签的训练数据，甚至没有任何关系列表的情况下，从网络中提取关系。这个任务通常被称为**开放信息抽取(open information extraction，Open IE)**。在 Open IE 中，关系只是简单的单词串(通常以动词开头)。例如，ReVerb 系统(Fader 等人，2011)从一个句子 s 中提取关系要通过 4 个步骤:

1. 在 s 上运行一个词类标记器和实体分块器。
2. 对于 s 中的每个动词，找到以一个动词开始并满足语法和词汇约束的单词 w 的最长序列，合并相邻的匹配项。
3. 对于每个短语 w，找到最左边的名词短语 x，它不是关系代词，wh-词或存在的词“there”。在右侧找到最接近的名词短语 y。
4. 使用一个置信度分类器将置信度 c 赋给关系 r = (x,w, y)并返回它。

关系只有在满足句法和词汇限制的情况下才被接受。句法约束确保它是一个以动词开头的序列，也可能包含名词(以轻动词如 make、have 或 do 开头的关系通常表达与名词关系的核心，如 have a hub in):

V | VP | VW*P
V = verb particle? adv?
W = (noun | adj | adv | pron | det)
P = (prep | particle | inf. marker)

词汇约束基于字典 D，该字典用于修剪非常罕见的长关系字符串。直觉是要消除候选关系，这些关系在足够数量的不同论元类型下不会出现，因此可能是不好的例子。该系统首先在 5 亿条网络句子上脱机运行上述关系抽取算法，并抽取出规范化后出现的所有关系列表(去除词形变化、助动词、形容词和副词)。如果每个关系 r 包含至少 20 个不同的论元，则将其添加到字典中。Fader 等人(2011)使用了 170 万个规范化关系的字典。

最后，使用逻辑回归分类器计算每个关系的置信度值。训练分类器的方法是:选取 1000 个随机的 web 句子，运行抽取器，并手工标签每个抽取的关系是正确的还是错误的。然后，使用关系和周围单词的特征，对手工标签的数据训练置信度分类器。图 17.10 显示了一些用于分类的样本特征。

例如下面的句子：

(17.13) United has a hub in Chicago, which is the headquarters of United Continental Holdings.

有关系短语“has a hub in”和“is the headquarters of”(它也有 has 和 is，但更长的短语优先)。步骤 3 在“has a hub in”的左边发现“United”，右边发现“Chicago”，跳过“which”，在“is the headquarters of”的左边发现“Chicago”。最终输出为:

r1: <United, has a hub in, Chicago>

r2: <Chicago, is the headquarters of, United Continental Holdings>

无监督关系抽取的最大优点是它能够处理大量的关系，而不需要预先指定它们。缺点是需要将这些大型字符串集合映射为某种规范形式，以便添加到数据库或其他知识源。目前的方法过于注重用动词表达的关系，因此会遗漏很多**名词性词(nominally)**表达的关系。

(x,r,y) covers all words in s
the last preposition in r is *for*
the last preposition in r is *on*
len(s)≤10
there is a coordinating conjunction to the left of r in s
r matches a lone V in the syntactic constraints
there is preposition to the left of x in s
there is an NP to the right of y in s

图 17-10：分类器的功能

图注：这些功能为开放信息提取系统 REVERB 提取的关系分配可信度(Fader 等人，2011)。

17.2.6. 关系抽取的评价

通过使用带有人类注释的、黄金标准关系的测试集以及计算精度、召回率和 F-measure 来评估**监督**关系抽取系统。标签的精度和召回率要求系统正确地对关系进行分类，而未标签的方法只是测量系统检测相关实体的能力。

半监督(semi-supervised)和**无监督(unsupervised)**方法更难评估，因为它们从 web 或大型文本中提取了全新的关系。因为这些方法使用了大量的文本，通常不可能只在一个小的带标签的测试集中运行它们，因此不可能预先注释一组正确的关系实例。

对于这些方法，可以通过从输出中抽取关系的随机样本来近似(仅)精度，并让人检查每个关系的准确性。通常这种方法侧重于从正文中提取的元组，而不是提及的关系；系统不需要检测每一个提及的关系来得到正确的分数。相反，当系统完成时，评估是基于占用(occupying)数据库的元组集。也就是说，我们想知道系统是否能发现 Ryanair has a hub at Charleroi，我们并不在乎它发现了多少次，则估计精度为：

$$\hat{P} = \frac{\#\text{ of correctly extracted relation tuples in the sample}}{\text{total }\#\text{ of extracted relation tuples in the sample.}} \tag{17.14}$$

另一种方法给了我们一些关于召回率的信息是计算不同召回率级别的精度。假设我们的系统能够对它产生的关系进行排序(根据概率或置信度)，我们可以分别计算前 1000 个新关系、前 10000 个新关系、前 100,000 个新关系的精度，等等。在每一种情况下，我们从该集合中随机抽取样本。这将向我们展示当我们提取越来越多的元组时，精度曲线的表现。但是没有办法直接评估召回率。

17.3. 提取时间

时间和日期是一种特别重要的命名实体，它们在日历和个人助理应用程序的问题回答中扮演着重要角色。为了推断时间和日期，在提取这些**时间表达式(temporal expression)**之后，必须将它们**规范化(normalized)**--转换为标准格式，以便进行推断。在本节中，我们考虑时间表达式的提取和规范化。

17.3.1. 时间表达式提取

时间表达式指的是绝对时间点、相对时间、持续时间和它们的集合。**绝对(absolute)**时间表达式是那些可以直接映射到日历日期、一天中的时间或两者都映射的表达式。**相对(relative)**时间表达式通过一些其他的参考点映射到特定的时间(比如 a week from last Tuesday)。最后，**持续(duration)**时间表示以不同粒度级别(秒、分钟、天、周、世纪等)的时间跨度。图 17.11 列出了这些类别中的一些时间表达式示例。

时态表达是一种语法结构，以时间**词汇触发器(lexical triggers)**作为其中心词。词汇触发器可以是名词、专有名词、形容词和副词;完整的时间表达包括它们的短语投射:名词短语、形容词短语和状语短语。图 17.12 提供了一些示例。

Absolute	Relative	Durations
April 24, 1916	yesterday	four hours
The summer of '77	next semester	three weeks
10:15 AM	two weeks from yesterday	six days
The 3rd quarter of 2006	last quarter	the last three quarters

图 17-11：绝对、关系和持续时间表达式的例子

Category	Examples
Noun	morning, noon, night, winter, dusk, dawn
Proper Noun	January, Monday, Ides, Easter, Rosh Hashana, Ramadan, Tet
Adjective	recent, past, annual, former
Adverb	hourly, daily, monthly, yearly

图 17-12：时间词汇触发器示例

让我们来看看 TimeML 注释方案，其中时间表达式用 XML 标记 TIMEX3 和该标记的各种属性进行注释(Pustejovsky 等人 2005, Ferro 等人 2005)。下面的示例演示了这种模式的基本用法(我们将在 17.3.2 节讨论属性)。

> A fare increase initiated <TIMEX3>last week</TIMEX3> by UAL Corp's United Airlines was matched by competitors over <TIMEX3>the weekend</TIMEX3>, marking the second successful fare increase in <TIMEX3>two weeks</TIMEX3>.

时间表达式识别任务包括查找与这种时间表达式对应的所有文本区间的开始和结束。基于规则的时间表达式识别方法使用自动机级联来识别复杂程度不断增加的模式。符记首先被标记为词类，然后根据包含触发器词(例如，February)或类(例如，MONTH)的模式，从以前阶段的结果中识别出越来越大的组块。图 17.13 给出了一个来自基于规则的系统的片段(Verhagen 等人，2005 年)。

```
# yesterday/today/tomorrow
$string =~ s/(($OT+the$CT+\s+)?$OT+day$CT+\s+$OT+(before|after)$CT+\s+)?$OT+$TERelDayExpr$CT+
(\s+$OT+(morning|afternoon|evening|night)$CT+)?)/<TIMEX$tever TYPE=\"DATE\">$1
<\/TIMEX$tever>/gio;

$string =~ s/($OT+\w+$CT+\s+)<TIMEX$tever TYPE=\"DATE\"[^>]*>($OT+(Today|Tonight)$CT+)
<\/TIMEX$tever>/$1$4/gso;

# this (morning/afternoon/evening)
$string =~ s/(($OT+(early|late)$CT+\s+)?$OT+this$CT+\s*$OT+(morning|afternoon|evening)$CT+)/
<TIMEX$tever TYPE=\"DATE\">$1<\/TIMEX$tever>/gosi;
$string =~ s/(($OT+(early|late)$CT+\s+)?$OT+last$CT+\s*$OT+night$CT+)/<TIMEX$tever
TYPE=\"DATE\">$1<\/TIMEX$tever>/gsio;
```

图 17-13：来自 Tarsqi 中的 GUTime 时间标记系统的 Perl 片段

序列标记方法(Sequence-labeling approaches)遵循与命名实体标记相同的 IOB 方案，用 I、O 和 B 标记在 TIME3 分隔的时间表达式内部、外部或开头的单词，如下所示:

A fare increase initiated last week by UAL Corp's...

O O O O B I O O O

从符记及其上下文中提取特征，并训练统计序列标签器(可以使用任何序列模型)。图 17.14 列出了时间标记中使用的标准特性。

时间表达式识别器是用通常的召回率、精度和 F-测度来评估的。所有这些词汇化的方法的一个主要困难是避免触发误报(false positives，假阳性)的表达:

(17.15) *1984* tells the story of Winston Smith...

(17.16) ...U2's classic *Sunday* Bloody *Sunday*

Feature	Explanation
Token	The target token to be labeled
Tokens in window	Bag of tokens in the window around a target
Shape	Character shape features
POS	Parts of speech of target and window words
Chunk tags	Base phrase chunk tag for target and words in a window
Lexical triggers	Presence in a list of temporal terms

图 17-14：用于训练 IOB 风格时间表达式标记器的典型特征

17.3.2. 时间规范化

时间规范化(temporal normalization)是将时间表达式映射到特定时间点或持续时间的过程。时间点与日历日期、一天中的时间或两者都对应。持续时间主要由时间长度组成，但也可能包括关于开始点和结束点的信息。标准化时间用来自 ISO8601 编码时间值的 VALUE 属性表示(ISO8601, 2004)。图 17.15 复制了我们前面添加了 VALUE 属性的示例 4

```
<TIMEX3 id = ' ' t1 ' ' type ="DATE" value =" 2007-07-02 " functionInDocument ="CREATION TIME"
> July 2, 2007 </TIMEX3> A fare increase initiated <TIMEX3 id =" t2 " type ="DATE"
value =" 2007-W26" anchorTimeID=" t1 ">last week</TIMEX3> by United Airlines was
matched by competitors over <TIMEX3 id =" t3 " type ="DURATION" value ="P1WE"
anchorTimeID=" t1 "> the weekend </TIMEX3>, marking these cond successful fare
increase in <TIMEX3 id =" t4 " type ="DURATION" value ="P2W" anchorTimeID=" t1 "> two
weeks </TIMEX3>.
```

图 17-15：包含时间表达式的规范化值的 TimeML 标注

此文本的日期栏(或文档日期)是 *July 2, 2007*。这种表达式的 ISO 表示形式为 YYYY-MM-DD，在本例中为 2007-07-02。我们的示例文本中时间表达式的编码都是从这个日期开始的，并在此显示为 VALUE 属性的值。

正文中第一个时间表达式是指一年中某个特定的周。在 ISO 标准中，周的编号从 01 到 53，每年的第一个周是当年第一个星期四的那个周。这些星期用模版 YYYY-Wnn 表示。我们文件日期的 ISO 周是第 27 周;因此 last week 的值表示为"2007-W26"。

下一个时间表达式是周末(the weekend)。ISO 的周从星期一开始;因此，周末出现在一周的末尾，并且完全包含在一个星期内。周末被视为持续时间，因此 VALUE 属性的值必须是一个长度。持续时间是根据 Pnx 模式来表示的，其中 n 是一个表示长度的整数，x 表示单位，如 P3Y 表示三年，P2D 表示两天。在本例中，一个周末被捕获为 P1WE。在这种情况下，也有足够的信息来锚定这个特定的周末作为特定一周的一部分。这些信息编码在 ANCHORTIMEID 属性中。最后，短语 two weeks 也表示 P2W 表示的持续时间。关于各种时间注释标准还有很多内容——这里涉及的太多了。图 17.16 描述了表示其他时间和持续时间的一些基本方法。请参阅 ISO8601(2004)、Ferro 等人(2005)和 Pustejovsky 等人(2005)了解更多细节。

目前大多数时间规范化方法都是基于规则的(Chang 和 Manning 2012, Strotgen 和 Gertz 2013)。匹配时间表达式的模式与语义分析过程相关联。就像在第 16 章中介绍的规则到规则的组合方法一样，一个成分的含义是通过使用特定于该成分的方法从其各部分的含义计算出来的，尽管这里的语义组合规则涉及的是时间算法，而不是λ-微积分附件。

Unit	Pattern	Sample Value
Fully specified dates	YYYY-MM-DD	1991-09-28
Weeks	YYYY-Wnn	2007-W27
Weekends	PnWE	P1WE
24-hour clock times	HH:MM:SS	11:13:45
Dates and times	YYYY-MM-DDTHH:MM:SS	1991-09-28T11:00:00
Financial quarters	Qn	1999-Q3

图 17-16：表示不同时间和持续时间的 ISO 模式示例

完全确定(fully qualified)的日期表达式以某些常规形式包含年、月和日。表达式中的单位必须被检测到，然后放在相应 ISO 模式中的正确位置。下面的模式规范了像 April 24, 1916 这样的表达。

FQTE → Month Date , Year {Year.val -- Month.val -- Date.val}

非结束符(non-terminals)月、日和年表示已经被识别和分配语义值的成分，通过*.val 标号访问这些成分。这个 FQE 成分的值，在进一步处理期间，可以反过来通过 FQTE.val 访问 。

在真实文本中，完全确定的时间表达式非常罕见。新闻文章中的大多数时间表达式都是不完整的，并且通常是相对于文章的日期栏而言，是隐式锚定的，我们将其称为文档的**时间锚定(temporal anchor)**。可以相对于该时间锚来计算诸如 today, yesterday, 或者 tomorrow 之类的时间表达式的值。Today 的语义过程仅分配锚点，tomorrow 和 yesterday 的附着词分别从锚点增加一天和减去一天。当然，考虑到我们表示的月份、星期、天和一天中的时间的周期性，我们的时间运算过程必须使用适合所用时间单位的模(modulo)运算。

不幸的是，即使是像 the weekend 或 Wednesday 这样的简单表达式也会引入相当多的复杂性。在我们当前的例子中，the weekend 显然指的是文档日期之前一周的周末。但情况并非总是如此，如下面的例子所示。

(17.17) Random security checks that began yesterday at Sky Harbor will continue at least through the weekend.

在这种情况下，短语 the weekend 指的是锚定日期所在一周中的那个周末(即下一个周末)。表明这个意思的信息来自于 continue 时态，这个动词支配着 the weekend。

相对时间表达式使用时间算法处理，类似于 today 和 yesterday 使用的算法。文档日期表明我们的示例文章是 ISO week 27，因此 last week 的表达式规范化为当前周减 1。为了解决有歧义的 next 和 last 表达式，我们考虑从锚定日期到最近单位的距离。Next Friday 可以指的是靠近当前的下一个星期五，也可以指下一周的星期五，但是文档日期离星期五越近，则该短语越有可能跳过最接近的那个星期五。通过将语言和特定于领域的试探法编码到时间附着词中，可以处理此类歧义。

17.4. 提取事件及其时间

事件提取(event extraction)的任务是识别文本中提到的事件。在本任务中，提及的事件是任何表示事件或状态的表达式，该事件或状态可以在时间上分配给特定的点或间隔。下面示例文本的标记显示了该文本中的所有事件。

[EVENT Citing] high fuel prices, United Airlines [EVENT said] Friday it has [EVENT increased] fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately [EVENT matched] [EVENT the move], spokesman Tim Wagner [EVENT said]. United, a unit of UAL Corp., [EVENT said] [EVENT the increase] took effect Thursday and [EVENT applies] to most routes where it [EVENT competes] against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

在英语中，大部分事件都对应于动词，而动词大多是用来介绍事件的。然而，从我们的例子中可以看出，情况并非总是如此。事件可以由名词短语引入，如在 the move 和 the increase 中，而有些动词不能引入事件，如动词短语 took effect，它指的是事件开始的时间，而不是事件本身。同样，轻动词，如 make、take 和 have 也常常不能表示事件;对于轻动词来说，事件通常用名词性直接宾语(took a flight)来表示，这

些轻动词只是为名词的论元提供了一个句法结构。

根据目标的不同，存在不同版本的事件提取任务。例如，在 TempEval 共享任务(Verhagen 等人，2009)中，目标是提取事件和体(aspects)，比如它们的体和时间属性。事件被分类为动作、状态、**报告事件(reporting events)**(比如 say、report、tell、explain)、感知事件，等等。每个事件的体、时态和情态也需要提取出来。例如，示例文本中的各种 said 事件可以注释为(class=REPORTING, tense=PAST, aspect=PERFECTIVE)。

事件提取通常通过监督学习建模，通过带有 IOB 标记的序列模型检测事件，并使用多类分类器分配事件类和属性。基于特征的模型使用表面信息，如词类、词汇项和动词时态信息;见图 17.17。

Feature	Explanation
Character affixes	Character-level prefixes and suffixes of target word
Nominalization suffix	Character-level suffixes for nominalizations (e.g., -tion)
Part of speech	Part of speech of the target word
Light verb	Binary feature indicating that the target is governed by a light verb
Subject syntactic category	Syntactic category of the subject of the sentence
Morphological stem	Stemmed version of the target word
Verb root	Root form of the verb basis for a nominalization
WordNet hypernyms	Hypernym set for the target

图 17-17：在事件检测中基于规则和机器学习方法的常用特征

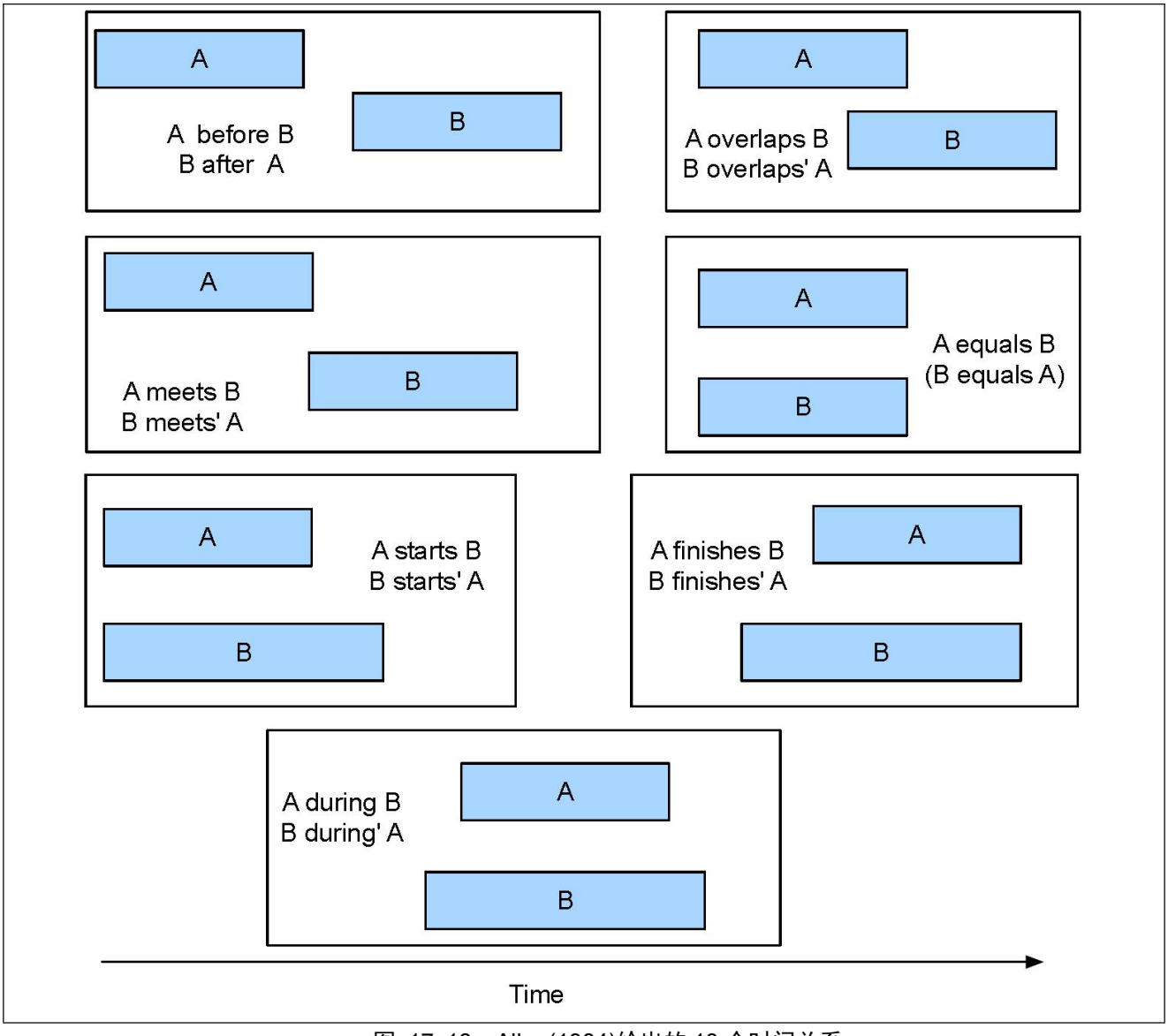


图 17-18：Allen(1984)给出的 13 个时间关系

17.4.1. 事件的时间顺序

在检测到文本中的事件和时态表达式之后，下一个逻辑任务是使用这些信息将事件整合到一个完整的时间轴中。这样的时间表对诸如问答和摘要等应用程序是有用的。这项雄心勃勃的任务是目前大量研究的主题，但超出了当前系统的能力。

一个稍微简单但仍然有用的任务是对文本中提及的事件和时间表达式进行部分排序。这样的排序可以提供与真正的时间表相同的许多好处。这种部分排序的一个示例是，在我们的示例文本中确定美国航空公司的票价上涨是在联合航空公司的票价上涨之后。确定这样的排序可以看作是一种二元关系检测和分类任务，类似于前面第 17.1 节中描述的任务。事件之间的时间关系被分类为图 17.18 (Allen, 1984)所示的 **Allen 关系(Allen relations)**标准集之一，使用第 17.1 节所示的基于特征的分类器，在 TimeBank 语料库上进行训练，特征包括单词/嵌入、解析路径、时态和体(aspect)。

时间库(TimeBank)语料库由注释了我们在本节中讨论过的许多信息的文本组成(Pustejovsky 等人，2003b)。时间库 1.2 包含从各种来源(包括宾州树库和 PropBank 集合)选择的 183 个新闻文章。

时间库语料中的每一篇文章都有在 TimeML 注释中明确标注的时间表达式和事件(Pustejovsky 等人，2003a)。除了时间表达式和事件之外，TimeML 注释还提供了事件和时间表达式之间的时间连接，后者指定了它们之间关系的性质。考虑下面的示例句子和图 17.19 中所示的相应标记，它们是从一个 TimeBank 文档中选择的。

(17.18) Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

如注释所示，该文本包括三个事件和两个时间表达式。这些事件都在发生(occurrence)类中，并被赋予了唯一的标识符，以便在进一步的注释中使用。时间表达式包括文章的创建时间(作为文档时间)和文本中的单个时间表达式。

除了这些注释之外，TimeBank 还提供了四个连接，使用图 17.18 中的 Allen 关系来捕捉文本中事件和时间之间的时间关系。下面是为这个例子注释的句子内时间关系。

- Soaring$_{e1}$ is **included** in the fiscal first quarter$_{t58}$
- Soaring$_{e2}$ is **before** 1989-10-26$_{t57}$
- Soaring$_{e3}$ is **simultaneous** with the bucking$_{e3}$
- Declining$_{e4}$ **includes** soaring$_{e1}$

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26" functionInDocument="CREATION_TIME">
10/26/89 </TIMEX3>
Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in <TIMEX3 tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the
fiscal first quarter </TIMEX3>, <EVENT eid="e3" class="OCCURRENCE">bucking</EVENT>
the industry trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT>
profits.
```

图 17-19：来自 TimeBank 语料库的示例

17.5. 模板填充

许多文本包含对事件的报道，以及可能的事件序列，这些报道往往与世界上相当常见的、刻板的情况相对应。这些抽象的情境或故事，与所谓的**脚本(scripts)**(Schank 和 Abelson, 1977)有关，由子事件、参与者和他们的角色的原型序列组成。这些脚本提供的强烈期望可以促进实体的适当分类，将实体分配到角色和关系中，以及最重要的是，对填充未说出的内容的推论的描绘。在最简单的形式中，这类脚本可以表示为由固定的槽集组成的模板，这些槽集接受属于特定类的**槽填充符(slot-fillers)**作为值。**模板填充(template filling)**的任务是找到调用特定脚本的文档，然后用从文本中提取的填充符来填充相关模板中的槽。这些槽填充符可以由直接从文本中提取的文本段组成，也可以由通过一些额外处理从文本元素中推断

出来的概念组成。

我们最初的航空公司故事中的填充模板可能如下所示。

```
FARE-RAISE ATTEMPT: [ LEAD AIRLINE:    UNITED AIRLINES
                      AMOUNT:          $6
                      EFFECTIVE DATE:  2006-10-26
                      FOLLOWER:        AMERICAN AIRLINES ]
```

该模板有四个槽(LEAD AIRLINE, AMOUNT, EFFECTIVE DATE, FOLLOWER)。下一节描述填充槽的标准序列标记方法。第 17.5.2 节接着描述了一个基于**有限状态转录机(fifinite-state transducers)**级联使用的旧系统，该系统旨在解决当前基于学习的系统尚未解决的更复杂的模板填充任务。

17.5.1. 模板填充的机器学习方法

在模板填充的标准范例中，我们得到了带有预定义模板及其槽填充符注释的文本跨度的训练文档。我们的目标是为输入中的每个事件创建一个模板，用文本跨度填充槽。

这个任务通常是通过训练两个独立的监督系统来建模的。

第一个系统决定模板是否存在于特定的句子中。这个任务被称为**模板识别(template recognition)**，或者有时，用一个可能令人困惑的术语来说，称为事件识别。模板识别可以被视为文本分类任务，从训练文档中标记的每个单词序列中提取特征，填补被检测的模板中的任何位置。可以使用通常的一组特征：符记、嵌入、单词形状、词类标记、语法块标记和命名实体标记。

第二个系统的工作是**角色填充符提取(role-filler extraction)**。训练一个单独的分类器来检测每个角色(LEAD-AIRLINE、AMOUNT，等等)。它可以是在已解析输入句子中的每个名词短语上运行的二元分类器，也可以是在单词序列上运行的序列模型。每个角色分类器都根据训练集合中的标签数据进行训练。同样，可以使用通常的特征集，但现在只针对单个名词短语或单个槽的填充符进行训练。

多个不相同的文本段可以用相同的槽标签进行标记。例如，在我们的示例文本中，字符串 United 或 United Airlines 可能被标记为 LEAD AIRLINE。这些并不是不兼容的选择，第 22 章中介绍的共指解析技术可以提供一个解决方案。

各种注释集合已经被用来评估这种模板填充方法，包括工作公告集合、论文会议电话、餐厅指南和生物文本。最近的工作侧重于在没有训练数据或甚至没有预定义模板的情况下提取模板，通过将模板归纳为一系列关联事件(Chambers 和 Jurafsky, 2011)。

17.5.2. 早期的有限状态模板填充系统

上面的模板相对简单。但是考虑一下生成一个模板的任务，该模板在这样的文本中包含所有信息(Grishman 和 Sundheim，1995)：

Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan. The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and “metal wood” clubs a month.

MUC-5 的“合资企业”任务(信息理解会议是一系列美国政府组织的信息提取评估)是生成描述合资企业的层次连接模板。图 17.20 显示了 FASTUS 系统产生的结构(Hobbs 等人，1997)。请注意，绑定模板的 ACTIVITY 槽的填充器本身是一个带有槽的模板。

早期处理这些复杂模板的系统是基于手写规则的转录机级联，如图 17.21 所示。

Tie-up-1		Activity-1:	
RELATIONSHIP	tie-up	COMPANY	Bridgestone Sports Taiwan Co.
ENTITIES	Bridgestone Sports Co. a local concern a Japanese trading house	PRODUCT START DATE	iron and “metal wood” clubs DURING: January 1990
JOINT VENTURE	Bridgestone Sports Taiwan Co.		
ACTIVITY	Activity-1		
AMOUNT	NT$20000000		

图 17-20：FASTUS 在输入文本后生成的模板

No.	Step	Description
1	Tokens	Tokenize input stream of characters
2	Complex Words	Multiword phrases, numbers, and proper names.
3	Basic phrases	Segment sentences into noun and verb groups
4	Complex phrases	Identify complex noun groups and verb groups
5	Semantic Patterns	Identify entities and events, insert into templates.
6	Merging	Merge references to the same entity or event

图 17-21：FASTUS 中的处理水平

图注：FASTUS 中的处理水平(Hobbs 等人，1997)。每一层提取特定类型的信息，然后传递到下一层。

前四个阶段使用手写的正则表达式和语法规则进行基本的符记化、分块和解析。然后，第 5 阶段使用基于 FST 的识别器识别实体和事件，并将识别的对象插入模板中适当的槽中。这个 FST 识别器基于如下手工构建的正则表达式(NG 表示名词组，VG 表示动词组)，它匹配上面新闻故事的第一个句子。

NG(Company/ies) VG(Set-up) NG(Joint-Venture) with NG(Company/ies) VG(Produce) NG(Product)

处理这两个句子的结果是五个草案模板(图 17.22)，然后必须合并到图 17.20 所示的单一层次结构中。在执行了相关解析之后，合并算法将合并两个可能描述相同事件的活动。

# Template/Slot	Value
1 RELATIONSHIP:	TIE-UP
ENTITIES:	Bridgestone Co., a local concern, a Japanese trading house
2 ACTIVITY:	PRODUCTION
PRODUCT:	“golf clubs”
3 RELATIONSHIP:	TIE-UP
JOINT VENTURE:	“Bridgestone Sports Taiwan Co.”
AMOUNT:	NT$20000000
4 ACTIVITY:	PRODUCTION
COMPANY:	“Bridgestone Sports Taiwan Co.”
STARTDATE:	DURING: January 1990
5 ACTIVITY:	PRODUCTION
PRODUCT:	“iron and “metal wood” clubs”

图 17-22：FASTUS 的阶段 5 生成的 5 个部分模板

图注：这些模板在阶段 6 中合并，以生成图 17.20 中所示的最终模板。

17.6. 总结

本章探讨了从文本中提取有限形式的语义内容的技术。

•**实体之间的关系**可以通过基于模式的方法提取：当有注释的训练数据可用时，可以使用**监督**学习方法；当有少量的**种子元组**或**种子模式**可用时，可以使用轻度**监督自举**方法；当有关系数据库可用时，可以使用**远程监督**、**无监督**或 **OPEN IE** 方法。

•通过结合统计学习和基于规则的方法对**时间表达式**进行检测和规范化，可以促进对时间的推理。

•使用经过时间和事件标记的数据(如 **TimeBank 语料库**)训练的序列模型和分类器，可以及时检测和排序**事件**。

•**模板填充**应用程序可以识别文本中的定型情况，并将文本中的元素分配给表示为**固定槽集合**的角色。

17.7. 文献和历史说明

最早的信息提取工作是在 Frump 系统的背景下处理模板填充任务(DeJong, 1982)。后来的研究受到了美国政府主办的 MUC 会议的推动(Sundheim 1991、Sundheim 1992、Sundheim 1993、Sundheim 1995)。早期的 MUC 系统，如 CIRCUS 系统 (Lehnert 等人，1991)和 SCISOR (Jacobs 和 Rau, 1990)，对后来的系统诸如 FASTUS (Hobbs 等人，1997)产生了很大的影响和启发。Chinchor 等人(1993)描述了 MUC 评价技术。

由于将系统从一个领域移植到另一个领域很困难，人们的注意力转向了机器学习方法。早期的 IE 监督学习方法(Cardie 1993, Cardie 1994, Riloff 1993, Soderland 等人 1995, Huffman 1996)关注于知识获取过程的自动化，主要用于有限状态的基于规则的系统。他们的成功，以及早期基于 HMM 的语音识别的成功，导致了序列标记的使用(HMMs：Bikel 等人 1997；MEMMs：McCallum 等人 2000；CRFs：Lafferty 等人 2001)，以及对特征的广泛探索(Zhou 等人，2005)。神经方法遵循了 Collobert 等人(2011)的开创性成果，他们在卷积网络上应用了 CRF。

通过使用共享基准数据集进行正式评估，继续刺激该领域的进展，包括 2000-2007 年对命名实体识别，关系提取和时间表达的自动内容提取(ACE)评估，**KBP**(Knowledge Base Population，知识基础群体)评估(Ji 等人 2010，Surdeanu 2013)的关系提取任务，例如槽填充(提取给定实体的年龄，出生地和配偶等属性(“slots”))和一系列 SemEval 研讨会(Hendrickx 等人，2009) 。

半监督关系提取最先由 Hearst (1992b)提出，并由 AutoSlog-TS (Riloff, 1996)、DIPRE (Brin, 1998)、SNOW-BALL (Agichtein 和 Gravano, 2000)和 Jones 等(1999)系统进行了扩展。远程监督算法描述从 Mintz 等人(2009),他们创造了“远程监督”这个词,但类似的想法发生在早期的系统中，该系统冠以“weakly labeled data(弱带安全标签的数据)”的名义(Craven 和 Kumlien(1999)和 Morgan 等人 2004，以及 Snow 等人 2005,Wu 和 Weld 2007)。这些系统还有很多扩展，其中包括 Wu 和 Weld(2010)、Riedel 等人(2010)和 Ritter 等人(2013)。开放 IE 系统包括 KNOWITALL(Etzioni 等人，2005)、TextRunner (Banko 等人，2007)和 REVERB (Fader 等人，2011)。还有一种通用模式，该模式结合了远程监督和 OPEN IE 的优势，参见 Riedel 等人(2013)。

HeidelTime (Strotgen 和 Gertz, 2013)和 SUTime (Chang 和 Manning, 2012)是可下载的时间提取和规范化系统。UzZaman 等人(2013)描述了 2013 年的 TempEval 挑战;Chambers(2013)和 Bethard(2013)给出了典型的方法。

17.8. 练习

17.1 Acronym expansion, the process of associating a phrase with an acronym, can be accomplished by a simple form of relational analysis. Develop a system based on the relation analysis approaches described in this chapter to populate a database of acronym expansions. If you focus on English **Three Letter Acronyms** (TLAs) you can evaluate your system’s performance by comparing it to Wikipedia’s TLA page.

17.2 A useful functionality in newer email and calendar applications is the ability to associate temporal expressions connected with events in email (doctor’s appointments, meeting planning, party invitations, etc.) with specific calendar entries. Collect a corpus of email containing temporal expressions related to event planning. How do these expressions compare to the kinds of expressions commonly found in news text that we’ve been discussing in this chapter?

17.3 Acquire the CMU seminar corpus and develop a template-filling system by using any of the techniques mentioned in Section 17.5. Analyze how well your system performs as compared with SOTA results on this corpus.

18. 单词意义和 WordNet

> **Lady Bracknell**. Are your parents living?
> **Jack**. I have lost both my parents.
> **Lady Bracknell.** To lose one parent, Mr. Worthing, may be regarded as a misfortune; to lose both looks like carelessness.
>
> Oscar Wilde, *The Importance of Being Earnest*

单词是有**歧义(ambiguous)**的:同一个词可以用来表示不同的东西。在第六章中，我们看到“mouse”这个词(至少)有两种意义:(1)一种小型啮齿动物，或者(2)一种手动控制光标的装置。“bank”这个词的意思是:(1)一个金融机构;(2)一个倾斜的堤岸。在上面的戏剧《认真的重要性》中，奥斯卡·王尔德扮演了“失去”的两种意义(放错地方的东西和遭受失去一个人的痛苦)。

我们说，mouse 或 bank 是**一词多义的(polysemous)**(源自希腊语 have many senses，poly 表示 many, sema 表示 sign 或 mark)[1]。**意义(sense)**(或词义)是一个单词的含义的一个方面(aspect)的离散表示。在本章中，我们将更详细地讨论词义，并介绍 WordNet，一个大型的在线**词典**(thesauruses)--表示词义的数据库--具有多种语言的版本。WordNet 也代表了意义之间的关系。例如，在狗和哺乳动物之间存在一个 IS-A 关系(狗是一种哺乳动物)，在引擎和汽车之间存在一个部分-整体关系(引擎是汽车的一部分)。

了解两个意义之间的关系在语言理解中起着重要的作用。考虑一下反义关系(antonymy)。如果两个词有相反的含义，那它们就是反义词，比如 long 和 short, up 和 down。区分这些元素对于语言理解非常重要(如果用户要求对话施事者将音乐调大，则不幸的是调低音乐)。但事实上，在 word2vec 这样的嵌入模型中，反义词很容易相互混淆，因为在嵌入空间中与单词最接近的单词(如 up)往往是它的反义词(如 down)。代表此关系的词典会有所帮助!

我们还引入了词义消歧(word sense disambiguation，WSD)，即确定在特定上下文中使用的单词的意义。我们将给出有监督的和无监督的算法来决定在特定的环境下意图使用哪种意义。这项任务在计算语言学和许多应用中有很长的历史。在回答问题时，如果我们知道哪一种 bat 意义是相关的，则我们可以对询问“bat care”的用户更有帮助。(用户是吸血鬼(vampire)吗?或者只是想打棒球(baseball)。)一个词的不同意义通常会有不同的翻译;在西班牙语中，动物 bat(蝙蝠)是一种 murcielago，而 baseball bat(棒球棒)是一种 bate，确实，词义算法可以帮助改进 MT (Pu 等人，2018)。最后，WSD 长期以来一直被用作评估自然语言理解模型的工具，而理解模型如何表示不同的词义是一个重要的分析方向。

18.1. 单词意义

意义是一个单词的含义的一个方面的离散表示。宽松地遵循词典编纂的传统，我们用上标来表示每种意义：bank[1] 和 bank[2]，mouse[1] 和 mouse[2]。在上下文中，很容易看到不同的含义：

mouse[1]:...1968 年，**鼠标**控制电脑系统。
mouse[2]:...像**老鼠**一样安静的动物
bank[1]:...**银行**可以将投资存放在托管账户中。
bank[2]:...随着东**岸**农业的迅速发展，这条河...

18.1.1. 定义单词意义

我们如何定义单词意义的含义(the meaning of a word sense)?我们在第六章介绍了将一个单词表示为语义空间中的一个嵌入点的标准计算方法。像 word2vec 或 GloVe 这样的嵌入模型的直觉是，一个单词

[1] 你可能还会看到**多义关系（polysemy)**的不同用法，指的是一个词的意思有某种语义关系的情况，而**同形关系（homonymy)**则用于意思之间没有关系的情况。

的意义可以通过它的共现来定义，即经常在附近出现的单词的数量。但这并没有告诉我们如何定义单词意义的含义。正如我们在第 10 章中看到的，像 BERT 这样的上下文嵌入更进一步，它提供了一个在文本上下文中表示单词含义的嵌入，我们将看到上下文嵌入是现代词义消歧算法的核心。

但首先，我们需要考虑**字典(dictionary)**和**词典(thesaurus)**提供的另一种定义意义的方式。一个是基于这样的事实，字典或词典给每一种意义的文本定义称为**注释(gloss)**。以下是 bank 的两种意义的注释:

1. financial institution that accepts deposits and channels the money into lending activities
2. sloping land (especially the slope beside a body of water)

注释并不是一种正式的含义表示;它们只是为人类而写的。请参考 American Heritage Dictionary (Morris, 1985)中的关于 right, left, red, 和 blood 的定义。

right	adj.	located nearer the right hand esp. being on the right when facing the same direction as the observer.
left	adj.	located nearer to this side of the body than the right.
red	n.	the color of blood or a ruby.
blood	n.	the red liquid that circulates in the heart, arteries and veins of animals.

注意这些定义中的**回环(circularity)**。right 的定义直接指向了它自己，而 left 的词条在短语 this side of the body 中包含了一个隐含的自我引用，这大概意味着 left。red 和 blood 的词条在定义上相互参照。对于人类来说，这样的词条是有用的，因为字典的用户对这些其他术语有充分的了解。

然而，尽管它们具有回环并且缺乏正式表示，但是注释仍然可以用于意义的计算建模。这是因为注释只是一个句子，我们可以从句子中计算出句子的嵌入来告诉我们一些关于意义的东西。字典经常会提供例句和注释，这些可以再次用来帮助建立一种意义的表示。

词典为定义一种意义提供的第二种方式是像字典定义一样，通过其与其他意义的关系来定义一种意义。例如，以上定义清楚地表明，right 和 left 是相似的词元，彼此之间存在某种交替或对立。同样，我们可以收集到 red 是一种颜色，而 blood 是一种液体。这种意义上的关系(IS-A 或反义词)在 WordNet 等在线数据库中明确列出。给定足够大的此类关系数据库，许多应用程序都具有执行关于词义的复杂语义任务的能力(即使它们并不真正从 left 知道自己的 right)。

18.1.2. 单词有多少种意义?

字典和词典给出离散的意义列表。相比之下，嵌入(无论是静态的还是上下文的)提供了一个连续的高维意义模型，它不会分解为离散的意义。

因此，创建词典取决于用于确定何时应以离散的意义来表示单词的不同用法的标准。如果两个意义具有独立的真实条件，不同的句法行为和独立的意义关系，或者它们表现出对立的意义，我们可能会认为它们是离散的。

考虑一下 WSJ 语料库中的动词 serve 的以下用法：

(18.1) They rarely serve red meat, preferring to prepare seafood.

(18.2) He served as U.S. ambassador to Norway in 1976 and 1977.

(18.3) He might have served his time, come out and led an upstanding life.

serving red meat 中的 serve，以及 serving time 中的 serve，显然具有不同的真值条件和预先设定。serve as ambassador 中的 serve 具有作为 NP 的独特子类结构。这些启发意味着可能有三种截然不同的 serve 意义。判定两种意义是否不同是一种实用的技术，该技术是将一个单词的两种用法合成到一个句子中。这种将相反意义的阅读知识(readings)结合起来的方法称为**轭合(zeugma)**。考虑以下例子:

(18.4) Which of those flights serve breakfast?

(18.5) Does Air France serve Philadelphia?

(18.6) ?Does Air France serve breakfast and Philadelphia?

我们用(?)来标记那些语义上有错误格式的例子。虚构的第三个例子(一个轭合案例)的奇怪之处表明，没有一种明智的方法可以使得 serve 的单独意义同时适用于 breakfast 和 Philadelphia。我们可以以此为

依据来证明在这种情况下 serve 有两种不同的意义。

字典倾向于使用许多细粒度的意义来捕捉细微的含义差异，考虑到字典的传统作用是帮助单词学习者，这是一种合理的方法。为了计算的目的，我们通常不需要这些细微的区别，所以我们经常将意义进行分组或聚类;我们已经在本章的一些例子中这样做了。实际上，将示例聚类为意义，或将意义聚类为更粗粒度的类别，是一项重要的计算任务，我们将在第 18.7 节中讨论。

18.2. 意义之间的关系

本节探讨意义之间的关系，特别是那些已经接受了大量计算研究的单词，如**同义关系(synonymy)**、**反义关系(antonymy)**和**上位关系(hypernymy)**。

同义关系

我们在第六章介绍过，当两个不同词(词元的两个意义相同或几乎相同时，我们说这两个意义是**同义词(synonym)**。同义词包括这样的对：

couch/sofa vomit/throw up filbert/hazelnut car/automobile

我们也提到过，在实践中，同义词这个词通常用来描述一种近似或粗略的同义词关系。此外，同义词实际上是一种意义之间的关系，而不是单词之间的关系。考虑单词 big 和 large。这两个词在下面的句子中看起来可能是同义词，因为我们可以在两个句子中互换 big 和 large，而且保留相同的含义:

(18.7) How big is that plane?

(18.8) Would I be flying on a large or small plane?

但是请注意下面的句子，我们不能用 large 来代替 big:

(18.9) Miss Nelson, for instance, became a kind of big sister to Benjamin.

(18.10) ?Miss Nelson, for instance, became a kind of large sister to Benjamin.

这是因为“big”这个词有“变老”或“长大”的意思，而“large”却没有这个意思。因此，我们说 big 和 large 的某些意义(几乎)是同义的，而其他的意义则不是。

反义关系

同义词是具有相同或相似意义的单词，而**反义词(antonym)**是具有相反意义的单词，例如：

long/short big/little fast/slow cold/hot dark/light rise/fall up/down in/out

如果两种意义定义了一种二元对立或在某种程度上处于对立的两端，那么它们可以是反义词。这就是长/短，快/慢，或大/小的情况，它们在长度或尺寸尺度的两端。另一组反义词，**可逆词(reversives)**，描述在相反方向上的变化或运动，如 rise/fall 或 up/down。

因此，反义词在一个方面的意义完全不同—它们在尺度上或方向上的位置—但在其他方面却非常相似，几乎所有其他方面的意义都相同。因此，自动区分同义词和反义词可能很困难。

分类关系

另一种联系词义的方法是**分类(taxonomically)**。一个词(或意义)是另一个词或意义的**下位词(hyponym)**，如果前者更具体，表示另一个词或意义的子类。例如 car 是 vehicle 的下位词，dog 是 animal 的下位词，mango 是 fruit 的下位词。反之，我们说 vehicle 是 car 的**上位词(hypernym)**，animal 是 dog 的上位词。不幸的是，这两个词(hypernym 和 hyponym)非常相似，因此很容易混淆;由于这个原因，**superordinate** 这个词经常被用来代替 hypernym。

Superordinate	vehicle	fruit	furniture	mammal
Subordinate	car	mango	chair	dog

我们可以更正式地定义上位关系，即由上位词表示的类外延地包含由下位词表示的类。因此，动物的类别包括所有的狗，而移动(moving)动作的类别包括所有的行走(walking)动作。上位关系也可以根据**蕴涵(entailment)**来定义。在此定义下，如果 A 的所有对象也都是 B，则意义 A 是意义 B 的下位词，因此，A 蕴涵 B 或$\forall x \quad A(x) \Rightarrow B(x)$。

下位关系/上位关系通常是一种传递关系。如果 A 是 B 的下位词，而 B 是 C 的下位词，则 A 是 C 的下

位词。

上位词/下位词结构的另一个名称是 IS-A 层次结构，其中我们说 A IS-A B 或 B 包含(subsumes)A。

上位关系在文本蕴涵或问题回答等任务中很有用；例如，知道白血病(leukemia)是一种癌症，在回答有关白血病的问题时肯定会很有用。

部份-整体关系

另一种常见关系是 **meronymy**，即**部分-整体(part-whole)**关系。腿是椅子的一部分;车轮是汽车的一部分。我们说 wheel 是 car 的**部分词(meronym)**, car 是 wheel 的**整体词(holonym)**。

结构化的一词多义

一个单词的意义也可以在语义上有联系，在这种情况下，我们称之为结构化一词多义(structured polysemy)。想想这个 bank 的意义:

(18.11) The bank is on the corner of Nassau and Witherspoon.

这个意义，可能是 bank[4]（银行），含义类似于“属于金融机构的建筑物”。这两种意义(组织和与组织相关的建筑)同时出现在许多其他词中(school, university, hospital 等)。因此，我们所代表的意义之间存在着一种系统的关系：

BUILDING ↔ ORGANIZATION

这种特殊的一词多义关系被称为隐喻(metonymy)。隐喻是使用一个概念或实体的一个方面来指实体的其他方面或实体本身。当我们用“白宫”这个短语来指办公室在白宫的政府时，我们就是在进行隐喻。其他常见的隐喻例子包括以下意义的成对关系:

AUTHOR *(Jane Austen wrote Emma)*	↔	**WORKS OF AUTHOR** *(I really love Jane Austen)*
FRUITTREE *(Plums have beautiful blossoms)*	↔	**FRUIT** *(I ate a preserved plum yesterday)*

18.3. WordNet：词汇关系数据库

在英语和许多其他语言中，最常用的语义关系资源是 **WordNet** 词汇数据库(Fellbaum, 1998)。英语 WordNet 由三个独立的数据库组成，分别是名词和动词，第三个是形容词和副词；不包括封闭类的单词。每个数据库都包含一组词元，每个词元都有一组语义注释。WordNet 3.0 版本有 117,798 个名词，11529 个动词，22,479 个形容词和 4,481 个副词。平均每个名词有 1.23 个意义，平均每个动词有 2.16 个意义。WordNet 可以从网上访问，也可以从本地下载。图 18.1 显示了名词和形容词 bass 的词元条目。

The noun “bass” has 8 senses in WordNet.
1. bass[1] - (the lowest part of the musical range)
2. bass[2], bass part[1] - (the lowest part in polyphonic music)
3. bass[3], basso[1] - (an adult male singer with the lowest voice)
4. sea bass[1], bass[4] - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass[1], bass[5] - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass[6], bass voice[1], basso[2] - (the lowest adult male singing voice)
7. bass[7] - (the member with the lowest range of a family of musical instruments)
8. bass[8] - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

图 18-1：名词 bass 的 WordNet 3.0 词条的一部分

请注意，名词有八种意义，形容词有一种意义，每一种意义都有**注释(gloss)**(字典式的定义)，这个意义的同义词列表，有时也有用法示例(如形容词意义所示)。WordNet 没有表示发音的信息，因此不区分 bass[4]、bass[5] 和 bass[8] 中的[b ae s]发音与其他发音[b ey s]的意义。

WordNet 意义的**近义词(near-synonym)**集称为**同义词集(synset)**；在 WordNet 中，同义词集是一个重要的原语。bass 的词条包括{bass[1], deep[6]}或{bass[6], bass voice[1], basso[2]}这样的同义词集。我们可以把同义词集看作是我们在第 15 章中讨论过的类型的概念。因此，WordNet 不是用逻辑术语来表示概念，而是将它们表示为可以用来表达概念的词义列表。下面是另一个同义词集的示例:

{chump[1], fool[2], gull[1], mark[9], patsy[1], fall guy[1],sucker[1], soft touch[1], mug[2]}

这个同义词集的注释描述如下:

Gloss: a person who is gullible and easy to take advantage of.

(容易受骗和容易被利用的人。)

注释是一个同义词集的属性，因此包含在同义词集之中的每个意义都有相同的注释，可以表达这个概念。因为它们共享语义，像这样的同义词集是与 WordNet 词条相关联的基本单位，因此，在 WordNet 中参与大多数词汇语义关系的是同义词集，而不是词形、词元或个体意义。

WordNet 还用从语义字段中抽取的词汇分类来标注每个同义词集，例如，图 18.2 所示的名词有 26 个类别，动词有 15 个类别(加上形容词的 2 个类别和副词的 1 个类别)。这些类别通常被称为**超意义(supersense)**，因为它们充当粗糙的语义类别或意义分组，当词义过于细粒度时，这可能是有用的(Ciaramita 和 Johnson 2003, Ciaramita 和 Altun 2006)。超意义也被定义为形容词(Tsvetkov 等人，2014)和介词(Schneider 等人，2018)。

Category	Example	Category	Example	Category	Example
ACT	*service*	GROUP	*place*	PLANT	*tree*
ANIMAL	*dog*	LOCATION	*area*	POSSESSION	*price*
ARTIFACT	*car*	MOTIVE	*reason*	PROCESS	*process*
ATTRIBUTE	*quality*	NATURAL EVENT	*experience*	QUANTITY	*amount*
BODY	*hair*	NATURAL OBJECT	*flower*	RELATION	*portion*
COGNITION	*way*	OTHER	*stuff*	SHAPE	*square*
COMMUNICATION	*review*	PERSON	*people*	STATE	*pain*
FEELING	*discomfort*	PHENOMENON	*result*	SUBSTANCE	*oil*
FOOD	*food*			TIME	*day*

图 18-2 ：超意义:WordNet 中名词的 26 个词汇分类

18.3.1. WordNet 中的意义关系

WordNet 表示上一节讨论的所有类型的意义关系，如图 18.3 和图 18.4 所示。

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	*breakfast*[1] → *meal*[1]
Hyponym	Subordinate	From concepts to subtypes	*meal*[1] → *lunch*[1]
Instance Hypernym	Instance	From instances to their concepts	*Austen*[1] → *author*[1]
Instance Hyponym	Has-Instance	From concepts to their instances	*composer*[1] → *Bach*[1]
Part Meronym	Has-Part	From wholes to parts	*table*[2] → *leg*[3]
Part Holonym	Part-Of	From parts to wholes	*course*[7] → *meal*[1]
Antonym		Semantic opposition between lemmas	*leader*[1] ⟺ *follower*[1]
Derivation		Lemmas w/same morphological root	*destruction*[1] ⟺ *destroy*[1]

图 18-3：WordNet 中的一些名词关系

Relation	Definition	Example
Hypernym	From events to superordinate events	*Fly*[9] → *travel*[5]
Troponym	From events to subordinate event	*walk*[1] → *stroll*[1]
Entails	From verbs (events) to the verbs (events) they entail	*snore*[1] → *sleep*[1]
Antonym	Semantic opposition between lemmas	*increase*[1] ⟺ *decrease*[1]

图 18-4：WordNet 中的一些动词关系

例如，WordNet 通过直接上位词和下位词的关系将每个同义词集与其直接更通用和更具体的同义词集相关联来表示下位关系(第 18.2 节)。可以遵循这些关系以产生具有更一般或更具体的同义词集的更长链。

图 18.5 显示了 bass[3] 和 bass[7] 的上位词链；更一般的同义词集在连续缩进的线上显示。

bass[3], basso (an adult male singer with the lowest voice)
=> singer, vocalist, vocalizer, vocaliser
=> musician, instrumentalist, player
=> performer, performing artist
=> entertainer
=> person, individual, someone...
=> organism, being
=> living thing, animate thing,
=> whole, unit
=> object, physical object
=> physical entity
=> entity

bass[7] (member with the lowest range of a family of instruments)
=> musical instrument, instrument
=> device
=> instrumentality, instrumentation
=> artifact, artefact
=> whole, unit
=> object, physical object
=> physical entity
=> entity

图 18-5：词元 bass 的两种不同意义的下位关系链

图注：请注意，这些链是完全不同的，只是会聚于非常抽象的层次 whole、unit。

WordNet 有两种分类实体:类和实例。实例是一个个体，一个专有名词，是一个独特的实体。例如，旧金山就是城市的一个例子。但城市是一个类，是自治市乃至地理位置的下位词。图 18.6 显示了 WordNet 的一个子图（(数据来自 Navigli 2016)），展示了许多关系。

18.4. 词义消歧

为一个词选择正确的意义的任务称为**词义消歧(word sense disambiguation)**，简称 WSD。WSD 算法以上下文中的一个词和一个固定的潜在词义清单作为输入，并输出上下文中正确的词义。

18.4.1. WSD:任务和数据集

在本节中，我们将介绍 WSD 的任务设置，然后转向算法。意义标记清单取决于任务。对于从英语翻译到西班牙语的上下文中的意义标记，英语单词的意义标记清单可能是不同西班牙语翻译的集合。对于医学文章的自动索引，意义标记清单可以是医学主题标题(Medical Subject Headings，MeSH)词典的词条集。或者，我们可以使用来自资源(如 WordNet)的意义集；或者，如果我们想要一个较粗的粒度集，则可以使用超意义(supersenses)。图 18.7 显示了单词 bass 的一些这样的示例。

在某些情况下，我们只需要消除一小部分单词的歧义。在这样的**词汇样本(lexical sample)**任务中，我们有一小部分预先选定的目标词和一些词汇表的意义清单。由于词汇集合和语义集合都很小，简单的监督分类方法工作得很好。

然而，更常见的是，我们有一个更难的问题，我们必须消除某些文本中所有单词的歧义。在这个**全词(all-words)**任务中，系统被给予一个完整文本和一个词汇表，其中每个词条都有一个意义清单，我们必须消除文本中的每个词的歧义(有时只是每个内容词)。全词任务类似于词类标注，不同之处在于它有一个更大的标记集，因为每个词元都有自己的标记集。这个更大的标记集的结果是数据稀疏性。

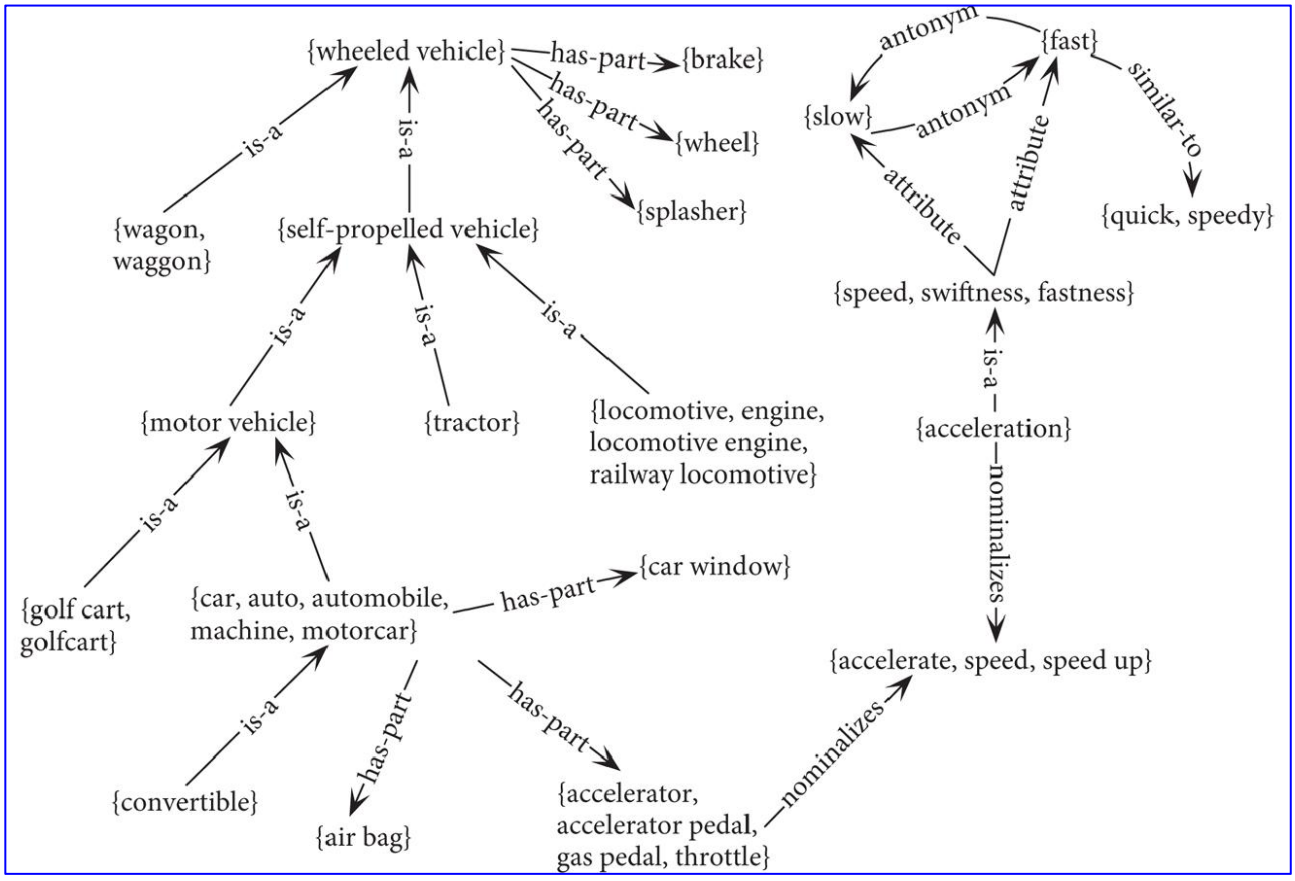


图 18-6：WordNet 作为一个图

WordNet Sense	Spanish Translation	WordNet Supersense	Target Word in Context
bass^4	lubina	FOOD	. . . fish as Pacific salmon and striped bass and. . .
bass^7	bajo	ARTIFACT	. . . play bass because he doesn't have to solo. . .

图 18-7：单词 bass 的一些可能的意义标记清单

受监督的全词消歧任务通常从**语义索引(semantic concordance)**(语料库)中进行训练，在这个语料库中，每个句子中的每个开放类词都用特定字典或词典(通常是 WordNet)中的词义进行标记。SemCor 语料库是 Brown 语料库的一个子集，由超过 226,036 个单词组成，这些单词是用 WordNet 意义手工标记的(Miller 等人 1993, Landes 等人 1998)。已经为 SENSEVAL 和 SemEval WSD 任务建立了其他意义标记的语料库，例如 SENSEVAL-3 Task 1 英语全词测试数据，包含 2282 个标注(Snyder 和 Palmer, 2004)或 SemEval-13 Task 12 数据集。大型语义索引在其他语言中也可用，包括荷兰语(Vossen 等人，2011)和德语(Henrich 等人，2012)。

这里有一个来自 SemCor 语料库的例子，显示了标记单词的 WordNet 意义数量;我们已经使用了标准的 WSD 表示法，其中一个下标标记了词类(Navigli, 2009):

(18.12) You will find$^{9}_{v}$ that avocado$^{1}_{n}$ is$^{1}_{v}$ unlike$^{1}_{j}$ other$^{j}_{j}$ fruit$^{1}_{n}$ you have ever$^{1}_{r}$ tasted$^{2}_{v}$

给定手工标记的测试集(如 fruit)中的每个名词、动词、形容词或副词，基于 semcord 的 WSD 任务是从 WordNet 中可能的意义中选择正确的意义。对于 fruit(水果)来说，这意味着要在正确答案“fruit^1_n”(种子植物成熟的生殖体)和其他两种意义的“fruit^2_n”(yield;产品的数量)和“fruit^3_n”(一些努力或行动的结果)之间作出选择。图 18.8 勾画了这个任务。

WSD 系统通常在本质上进行评估，通过计算 F1 来针对(against)在一个不提供的集合中手工标记的意义标记，例如上面讨论的 SemCor 语料库或 SemEval 语料库。

从标签语料库中为每个单词选择**最常见的意义(most frequent sense，MFS)**是一个令人惊讶的强大基线(Gale 等人，1992a)。对于 WordNet，这与第一个意义相对应，因为 WordNet 中的意义通常根据其

在 SemCor 意义标记语料库中的计数从最常见到最不常见排序。最常见的意义基线可以非常准确，因此经常用作默认值，在监督算法没有足够的训练数据时提供一个单词的意义。

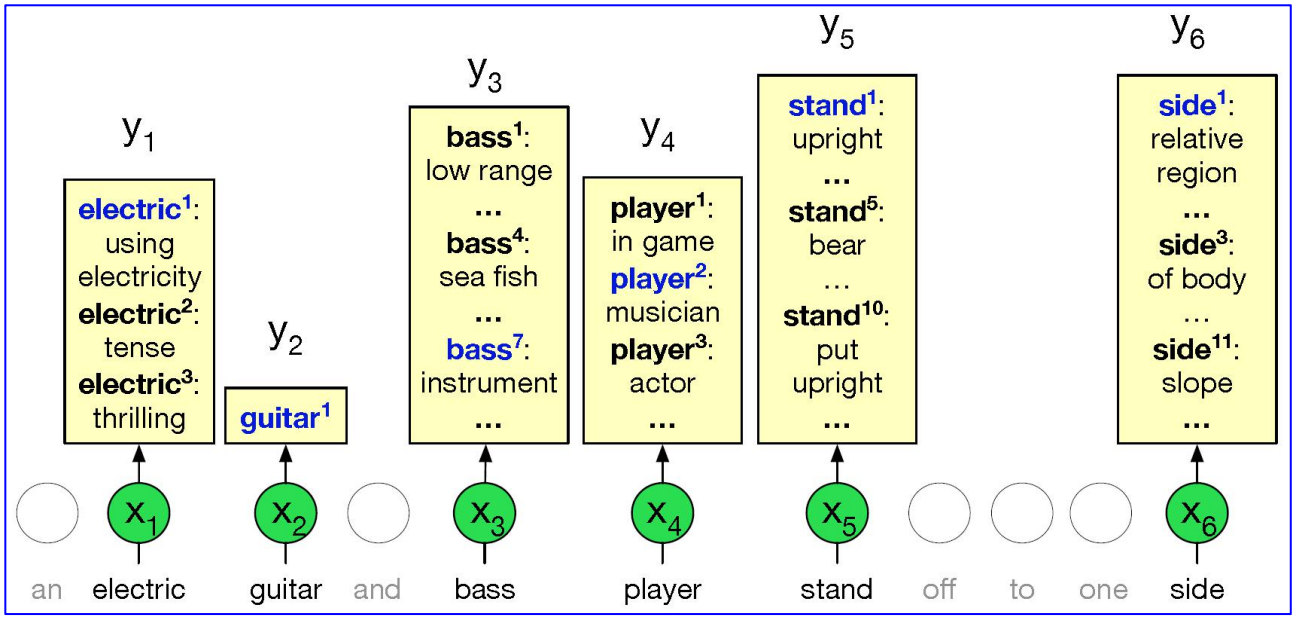


图 18-8：全词 WSD 任务

图注：从输入词(x)映射到 WordNet 意义(y)。只有名词、动词、形容词和副词被映射，注意一些词(如例子中的 guitar)在 WordNet 中只有一种意义。图的灵感来自 Chaplot 和 salakhudinov(2018)。

第二种启发，称为**每一话语一种意义(one sense per discourse)**，是基于 Gale 等人(1992b)的研究，他们注意到在文本或话语中多次出现的单词通常具有相同的意义。这种启发式似乎更适合粗粒度的意义，特别是对于同义词而不是多义词的情况，所以通常不作为基线使用。但是，各种消歧任务通常包括一些偏向于在话语片段内部以相同方式解决歧义的倾向。

18.4.2. WSD 算法:上下文嵌入

表现最好的 WSD 算法是使用上下文词嵌入的简单的 1-最近邻算法，这归功于 Melamud 等人(2016)和 Peters 等人(2018)。在训练时，我们通过任何上下文嵌入(例如，BERT)来传递 SemCore 标签数据集中的每个句子，从而导致 SemCore 中每个标签符记的上下文嵌入。对于每个单词的每个意义 c 的每个符记 c_i，我们对上下文表示进行平均，生成一个 c 的上下文**意义嵌入** v_s:

$$V_S = \frac{1}{n}\sum_i c_i \tag{18.13}$$

在测试时，我们同样计算目标词的上下文嵌入 t，并从训练集中选择其最邻近的意义(与 t 余弦值最高的意义)。图 18.9 说明了这个模型。

我们该如何处理在意义标签的训练数据中没有看到的单词呢?毕竟，SemCor 中出现的意义的数量只是 WordNet 中单词的一小部分。最简单的算法是退回到最常见的意义基线，即在 WordNet 中取第一种意义。但这不是很令人满意。

Loureiro 和 Jorge(2019)提出的一种更强大的方法是，通过使用 WordNet 分类法和超意义，自下而上地估算缺失的意义嵌入。我们得到在 WordNet 分类中的任何更高级别节点的意义嵌入，其方法是，通过对其孩子节点的嵌入进行平均化，因此计算如下三项内容：每个同义词集的嵌入作为其意义嵌入的平均值；一个上位词的嵌入作为其同义词集嵌入的平均值；词汇编目类别(超意义)的嵌入作为该类别的大型同义词集嵌入的平均值。更正式地讲，对于 WordNet $\hat{s} \in W$ 中的每个缺失意义，让其同义词集其他成员的意义嵌入为 $S_{\hat{s}}$，上位词专有的同义词集嵌入为 $H_{\hat{s}}$，而词汇编目的(超意义特有的)同义词集嵌入为 $L_{\hat{s}}$。然后，我们可以如下计算$\hat{s}$的意义嵌入：

$$\text{if} \quad |S_{\hat{s}}| > 0, \quad \mathbf{v}_{\hat{s}} = \frac{1}{|S_{\hat{s}}|} \sum \mathbf{v}_s, \forall \mathbf{v}_s \in S_{\hat{s}} \tag{18.14}$$

$$\text{else if} \quad |H_{\hat{s}}| > 0, \quad \mathbf{v}_{\hat{s}} = \frac{1}{|H_{\hat{s}}|} \sum \mathbf{v}_{syn}, \forall \mathbf{v}_{syn} \in H_{\hat{s}} \tag{18.15}$$

$$\text{else if} \quad |L_{\hat{s}}| > 0, \quad \mathbf{v}_{\hat{s}} = \frac{1}{|L_{\hat{s}}|} \sum \mathbf{v}_{syn}, \forall \mathbf{v}_{syn} \in L_{\hat{s}} \tag{18.16}$$

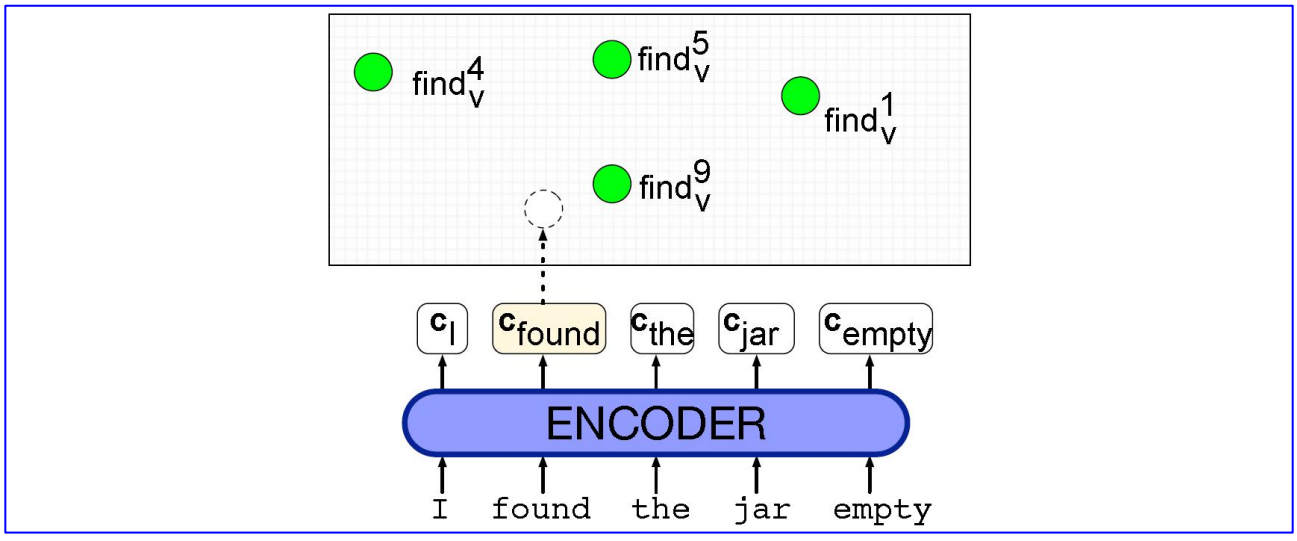


图 18-9：WSD 的最近邻算法

图注：绿色的是预先为每个单词的每种意义计算的上下文嵌入;这里我们只展示了一些 find 的意义。计算目标词 found 的上下文嵌入，然后选择最邻近的意义(在本例中是$find_v^9$)。图的启发来自 Loureiro 和 Jorge(2019)。

由于所有超意义在 SemCor 中都有一些标签数据，所以当算法返回到最一般的(超意义)信息时，算法就可以保证对所有可能的意义都有一些表示，尽管当然是使用一个非常粗糙的模型。

18.5. 可选的 WSD 算法和任务

18.5.1. 基于特征的 WSD

用于 WSD 的基于特征的算法非常简单，功能几乎与上下文语言模型算法一样。表现最好的是 IMS 算法(Zhong 和 Ng，2010)，并通过嵌入进行了增强(Iacobacci 等人 2016，Raganato 等人 2017b)，它使用 SVM 分类器为每个输入单词选择意义，并具有以下简单的周围单词的特征：

•词类标记(用于每边 3 个单词的窗口，在句子边界处停止)

•单词的**搭配(collocation)**特征或长度为 1、2、3 的 n-grams 在每边 3 个单词的窗口内的特定搭配位置(即正好一个单词在右边，或从左边 3 个单词开始的两个单词，以此类推)。

•嵌入的加权平均(所有单词在每边 10 个单词的窗口中，按距离指数加权)

想想《华尔街日报》下面这句话里那个模棱两可的单词 bass 吧:

(18.17) An electric guitar and **bass** player stand off to one side,

如果我们使用一个小的 2 字窗口，一个标准的特征向量可能包括词类、unigram 和 bigram 搭配特征，以及嵌入的加权总和 g，即:

$$[w_{i-2}, \text{POS}_{i-2}, w_{i-1}, \text{POS}_{i-1}, w_{i+1}, \text{POS}_{i+1}, w_{i+2}, \text{POS}_{i+2}, w_{i-2}^{i-1}, w_{i+1}^{i+2}, g(E(w_{i-2}), E(w_{i-1}), E(w_{i+1}), E(w_{i+2})] \tag{18.18}$$

将产生以下向量:

[guitar, NN, and, CC, player, NN, stand, VB, and guitar, player stand, g(E(guitar),E(and),E(player),E(stand))]

18.5.2. 以 Lesk 算法作为 WSD 基线

生成像 SemCor 这样的意义标签语料库是相当困难和昂贵的。另一类 WSD 算法，**基于知识(knowledge-based)**的算法，只依赖于基于 WordNet 知识或其他此类资源，不需要标签数据。虽然监督算法通常工作更好，基于知识的方法可在语言或领域使用词典或字典，但没有意义标签的语料库是可用的。

Lesk 算法是最古老和最强大的基于知识的 WSD 方法，是一个有用的基线。Lesk 实际上是一系列算法，这些算法选择一种意义，此意义的字典注释或定义与目标词的邻域共享最多的词。图 18.10 显示了该算法的最简单版本，通常称为简化的 Lesk 算法(Kilgarriff 和 Rosenzweig, 2000)。

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word
  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ←set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
      max-overlap ← overlap
      best-sense ← sense
    end
return(best-sense)
```

图 18-10：简化的 Lesk 算法

图注：函数 COMPUTEOVERLAP 返回两个集合之间共有的单词个数，忽略虚词或停用列表上的其他单词。最初的 Lesk 算法以一种更复杂的方式定义上下文。

作为 Lesk 算法工作的一个例子，考虑在下面的上下文中消除单词 bank 的歧义:

(18.19) The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

在 WordNet 中有以下两个意义:

$bank^1$	Gloss : Examples:	a financial institution that accepts deposits and channels the money into lending activities “he cashed a check at the bank”, “that bank holds the mortgage on my home”
$bank^2$	Gloss : Examples:	sloping land (especially the slope beside a body of water) “they pulled the canoe up on the bank”, “he sat on the bank of the river and watched the currents”

在(18.19)中， $bank^1$ 的意义有两个与上下文重叠的非停用词:deposits 和 mortgage，而 $bank^2$ 的意义没有重叠词，因此选择 $bank^1$ 的意义。

对简化的 Lesk 有许多明显的扩展，例如通过 IDF(逆文档频率)第 6 章对重叠词进行加权，以减轻像虚词这样的频繁词；最好的表现是使用词嵌入余弦代替词重叠来计算定义和上下文之间的相似度(Basile 等人，2014)。Lesk 的现代神经扩展使用这些定义来计算可以直接使用的意义嵌入，而不是使用 SemCor 训练的嵌入(Kumar 等人 2019, Luo 等人 2018a, Luo 等人 2018b)。

18.5.3. 上下文词的评估

词义消歧是一种比我们在第 6 章中描述的上下文无关的单词相似度任务更精细的单词含义评估。回想一下，像 LexSim-999 这样的任务要求系统匹配人类对两个单词之间上下文无关相似性的判断(cup 和 mug 有多相似?)我们可以将 WSD 看作是一种上下文化的相似任务，因为我们的目标是能够将一个单词含义(比如 bass)与一个上下文(播放音乐)和另一个上下文(钓鱼)中的意义区分开来。

介于两者之间的是**上下文词(word-in-context，WIC)**的任务。在这里，系统给出了两个句子，每个句子都有相同的目标词，但在不同的句子上下文中。系统必须确定目标词在两个句子中是在同一意义上使用还是在不同意义上使用。图 18.11 显示了 Pilehvar 和 Camjo-Collados (2019) WIC 数据集的样本偶对。

WIC 的句子主要来源于 WordNet 中的意义例句。但是 WordNet 的意义是非常精细的。出于这个原因，像 WIC 这样的任务首先将词义聚成更粗的词簇，这样，如果目标词的两个句子上下文在同一个词簇中，就被标记为 T。WIC 将所有意义偶对聚集在一起，条件是：如果它们是 WordNet 语义图中的一级连接(包括姐妹意义)；或者，如果它们属于同一个超意义。在本章的最后，我们指出了其他意义上的聚类算法。

F	There's a lot of trash on the **bed** of the river — I keep a glass of water next to my **bed** when I sleep
F	**Justify** the margins— The end **justifies** the means
T	**Air** pollution— Open a window and let in some **air**
T	The expanded **window** will give us time to catch the thieves — You have a two-hour **window** of clear weather to finish working on the lawn

图 18-11：WIC 数据集的正(T)和负(F)样本偶对

解决 WIC 任务的基线算法使用了像 BERT 这样的上下文嵌入和一个简单的阈值余弦。我们首先计算目标词在两个句子中的上下文嵌入，然后计算它们之间的余弦。如果它超过了 devset 的阈值，我们就会响应为真(这两种意义是相同的)，否则我们就会响应为假。

18.5.4. 维基百科作为训练数据的来源

除 SemCor 以外的数据集已用于全词 WSD。一个重要的方向是使用维基百科作为意义标签数据的来源。当在维基百科的文章中提到一个概念时，文章文本可能包含到这个概念的维基百科页面的显式连接，该页面由唯一标识符命名。这个连接可以用作意义注释。例如，含糊不清的单词 bar 会根据上下文中的意义连接到不同的维基百科文章，包括页面 BAR(Law)、页面 BAR(Music)等等，如下面的维基百科示例所示(Mihalcea, 2007)。

In 1834, Sumner was admitted to the **[[bar (law)|bar]]** at the age of twenty-three,
and entered private practice in Boston.

It is danced in 3/4 time (like most waltzes), with the couple turning approx. 180 degrees
every **[[bar (music)|bar]]**.

Jenga is a popular beer in the **[[bar (establishment)|bar]]**s of Thailand.

然后可以将这些句子添加到受监督系统的训练数据中。然而，为了以这种方式使用维基百科，有必要将维基百科的概念映射到与 WSD 应用程序相关的任何意义清单。从 Wikipedia 映射到 WordNet 的自动算法包括，例如，找到与 Wikipedia 意义在词汇上有最大重叠的 WordNet 意义，方法是经过如下两个步骤：第一步是比较：将 WordNet 同义词集的单词向量、注释及其相关意义与 Wikipedia 页面标题中的单词向量相比较；第二步是传出：传出连接和页面类别(Ponzetto 和 Navigli，2010)。由此产生的映射已被用于创建 BabelNet，它是一个大型的意义标注资源(Navigli 和 Ponzetto, 2012)。

18.6. 使用词典来改善嵌入

词典也被用来改进静态的和上下文的词嵌入。例如，静态词嵌入有反义词的问题。像 expensive 这样的词在将余弦函数嵌入到它的反义词 cheap 中经常非常相似。来自词典的反义信息可以帮助解决这个问题；图 18.12 显示了 GloVE 中目标词的最近邻词，以及此方法之后的一种改进。

Before counterfitting				After counterfitting		
east	west	north	south	eastward	eastern	easterly
expensive	pricey	cheaper	costly	costly	pricy	overpriced
British	American	Australian	Britain	Brits	London	BBC

图 18-12：在 GloVe 中的最近邻居

图注：在 GloVe 中，east，expensive，和 British 的最近邻居包括反义字 west。右侧显示了采用伪造方法后，GloVe 最近邻居的改善(Mrksic 等人，2016)。

解决方案有两个簇。第一个需要再训练:我们修改嵌入训练，以合并词典关系，如同义词、反义词、或超意义。这可以通过修改 word2vec 的静态嵌入损失函数(Yu 和 Dredze 2014, Nguyen 等人 2016)或修改

上下文嵌入训练(Levine 等人 2020, Lauscher 等人 2019)来实现。

第二，对于静态嵌入，更轻量化;在对嵌入进行了训练之后，我们学习了基于词典的第二种映射，这种映射改变了单词的嵌入，使同义词(根据词典)被推得更近，反义词被推得更远。这些方法被称为**改造(retrofitting)**(Faruqui 等人 2015, Lengerich 等人 2018)或**反改造(counterfitting)**(Mrksic 等人，2016)。

18.7. 词义归纳

要建立一个大型的语料库，对每个词的词义进行标签，既昂贵又困难。因此，一种无监督的词义消歧方法，通常称为**词义归纳(word sense induction)**或 WSI，是一个重要的方向。在无监督的方法中，我们不使用人类定义的词义。相反，每个单词的“意义”集合是由训练集中每个单词的实例自动创建的。

大多数用于词义归纳的算法遵循了 Schütze(Schütze 1992b, Schütze 1998)在词嵌入上使用某种聚类的早期工作。在训练中，我们使用三个步骤:

1.对于语料库中单词 w 的每个符记 w_i，计算上下文向量 c。

2.使用聚类算法将这些单词-符记上下文向量 c 聚类为预定义数量的群组或聚类。每个聚类都定义了 w 的意义。

3.计算每个聚类的向量质心。每个向量质心 s_j 是一个**意义向量**，它表示 w 的意义。

由于这是一种无监督算法，我们无法为 w 的每一种“意义”命名;我们只是引用 w 的第 j 个意义。为了消除对 w 的特定符记 t 的歧义，我们同样有三个步骤:

1.计算 t 的上下文向量 c。

2.检索 w 的所有意义向量 s_j。

3.将 t 赋值给最接近 t 的意义向量 s_j 表示的意义。

我们所需要的是一个聚类算法和向量之间的距离度量。聚类是一个经过深入研究的问题，有许多标准算法可以应用于作为数值向量结构的输入(Duda 和 Hart, 1973)。语言应用程序中常用的一种技术称为**凝聚聚类(agglomerative clustering)**。在这种技术中，每 N 个训练实例最初都分配给它自己的聚类。然后，通过连续合并两个最相似的聚类，以自下而上的方式形成新的聚类。这个过程会一直持续，直到达到指定数量的聚类，或者达到聚类中的某种全局良度指标(goodness measure)。如果训练实例的数量使这种方法过于昂贵，那么可以在原始训练集上使用随机抽样来获得类似的结果。

我们如何评估无监督的意义消歧方法？ 像往常一样，最好的方法是在某些端到端系统中进行外部评估。SemEval 评估中使用的一个示例是改善搜索结果的聚类和多样化(Navigli 和 Vannella，2013)。内在评估需要一种将自动派生的意义类别映射到手工标记的黄金标准集合中的方法，以便我们可以将手工标记的测试集合与我们的无监督分类器标记的集合进行比较。例如在 SemEval 任务(Manandhar 等人 2010，Navigli 和 Vannella 2013，Jurgens 和 Klapaftis 2013)中测试了各种此类度量标准，包括聚类重叠度量标准，或通过选择(在某些训练集中)与该类重叠最多的意义将每个意义聚类映射到预定义意义的方法。但是，可以公平地说，此任务的评估指标尚未成为标准。

18.8. 总结

本章涵盖了与词汇术语相关的含义的广泛问题。以下是其中的亮点:

•**词义**是单词含义的所在地;定义和含义关系是被定义在词义层面而不是在词形层面。

•许多词是**一词多义**的，有许多词义。

•意义之间的关系包括**同义关系**、**反义关系**、**整体部分关**系，以及**分类关系**：**下位关**系和**上位关系**。

•**WordNe**t 是一个大型的英语词汇关系数据库，并且 WordNets 支持多种语言。

•**词义消歧(WSD)**是在上下文中确定一个词的正确意义的任务。监督方法使用一个句子的语料库，其中单个单词(词汇样本任务)或所有单词(全词任务)都用 WordNet 这样的资源的意义来做手工标记。SemCor 是最大的带有 WordNet 标签意义的语料库。

•WSD 的标准监督算法是带有上下文嵌入的最近邻居。

•基于特征的算法使用的词类和嵌入词在目标词的上下文中也运行很好。

•WSD 的一个重要基准是 WordNet 中**最常见意义**，即**第一意义**。

•另一个基准是**基于知识**的 WSD 算法，称为 **Lesk 算法**，该算法选择其字典定义与目标单词的邻域共享最多单词的意义。

•**词义归纳**是学习无监督词义的任务。

18.9. 文献和历史说明

词义消歧可以追溯到数字计算机最早的一些应用。Weaver(1955)在机器翻译的背景下首次阐述了作为现代词义消歧算法基础的见解:

> 如果一个人看一本书里的字，一次看一个，就像透过一个有一个字宽的孔的不透明面具，那么显然，一次看一个字是不可能确定这些字的含义的。[...] 但是，如果一个人把不透明掩膜上的缝隙拉长，直到他不仅能看到有问题的中心词，而且还能说出两边的 N 个词，那么，如果 N 足够大，他就可以明确地确定中心词的含义。[...] 实际的问题是:“N 的最小值是多少，至少在可容忍的部分情况下，会导致对中心词含义的正确选择?”

其他的概念在这个早期首先提出包括使用一个同义词典来消除歧义(Masterman, 1957)，监督训练贝叶斯模型来消除歧义(Madhu 和 Lytel, 1965)，以及在词义分析中使用聚类(Sparck Jones, 1986)。

许多消歧工作是在早期面向人工智能的自然语言处理系统的背景下进行的。Quillian(1968)和 Quillian(1969)提出了一种基于图的语言理解方法，其中一个词的定义由一个由句法和语义关系连接的词节点网络来表示，并通过在图中寻找词义之间的最短路径来消歧。Simmons(1973)是另一种有影响的早期语义网络方法。Wilks 用他的优先语义学(Preference Semantics)(Wilks 1975c, Wilks 1975b, Wilks 1975a)提出了最早的非离散模型之一，Small 和 Rieger(1982)、Riesbeck(1975)提出了基于对每个单词丰富的程序信息建模的理解系统。Hirst’s ABSITY 系统(Hirst 和 Charniak 1982, Hirst 1987, Hirst 1988)使用了一种基于语义网络的标记传递技术，代表了这类系统中最先进的系统。与这些主要的符号方法一样，早期的神经网络(当时被称为“连接主义者”)消除词义歧义的方法依赖于带有手工编码表示的小词汇(Cottrell 1985, Kawamoto 1988)。

Kelly 和 Stone(1975)领导了一个团队，为 1790 个有歧义的英语单词手工制作了一套消除歧义的规则，最早实现了一种强有力的经验方法来消除歧义。Lesk(1986)是第一个使用机器可读字典来消除歧义的人。Fellbaum(1998)收集 WordNet 的早期作品。早期使用字典作为词汇资源的作品包括 Amsler’s(1981)使用韦氏大词典和朗曼当代英语大字典(Boguraev 和 Briscoe1989)。

有监督的消歧方法始于 Black(1988)使用决策树。除了 IMS 和基于上下文嵌入的监督 WSD 方法，最近的监督算法包括编码器-解码器模型(Raganato 等人，2017a)。

在监督方法中需要大量的标注文本，这导致了早期对引导方法的研究(Hearst 1991, Yarowsky 1995)。例如 Diab 和 Resnik(2002)的半监督算法就是基于两种语言的平行语料库。例如，法语单词 catastrophe

可能在一种情况下被翻译成英语单词 disaster(灾难)，而在另一种情况下被翻译成英语单词 tragedy(悲剧)，这一事实可以用来消除这两个英语单词的歧义(即选择 disaster 和 tragedy 的意义相近)。

聚类在词义研究中最早使用的是 Sparck Jones (1986);Pedersen 和 Bruce (1997)，Schutze (1997b)，和 Schutze(1998)应用了分布方法。将词义聚类为**粗糙意义(coarse senses)**也已用于解决字典意义过于精细的问题(第 18.5.3 节)(Dolan 1994，Chen 和 Chang 1998，Mihalcea 和 Moldovan 2001，Agirre 和 de Lacalle 2003，Palmer 等人 2004，Navigli 2006，Snow 等人 2007，Pilehvar 等人 2013)。用于训练监督聚类算法的具有聚类词义的语料库包括 Palmer 等人(2006)和 **OntoNotes**(Hovy 等人，2006)。

除其他外，关于一词多义表示的计算方法，参见 Pustejovsky (1995)，Pustejovsky 和 Boguraev (1996)，Martin (1986)，Copestake 和 Briscoe(1995)。Pustejovsky 的**生成词汇**理论，尤其是他的单词的**特质(qualia)结构**理论，是一种在上下文中解释词的动态系统多义性的方式。

WSD 的历史概述包括 Agirre 和 Edmonds(2006)和 Navigli(2009)。

18.10. 练习

18.1 Collect a small corpus of example sentences of varying lengths from any newspaper or magazine. Using WordNet or any standard dictionary, deter mine how many senses there are for each of the open-class words in each sen tence. How many distinct combinations of senses are there for each sentence? How does this number seem to vary with sentence length?

18.2 Using WordNet or a standard reference dictionary, tag each open-class word in your corpus with its correct tag. Was choosing the correct sense always a straightforward task? Report on any difficulties you encountered.

18.3 Using your favorite dictionary, simulate the original Lesk word overlap dis ambiguation algorithm described on page 367 on the phrase *Time flies like an arrow*. Assume that the words are to be disambiguated one at a time, from left to right, and that the results from earlier decisions are used later in the process.

18.4 Build an implementation of your solution to the previous exercise. Using WordNet, implement the original Lesk word overlap disambiguation algo rithm described on page 367 on the phrase *Time flies like an arrow*.

19. 语义角色标签



在公元前 7 至 4 世纪之间的某个时候，印度文法学家 Pānini[1]写了一篇著名的**梵语**(Sanskrit)语法专着《Aṣṭādhyāyī》(共 8 本书)，该专着被称为“人类智力最伟大的纪念碑之一”(Bloomfield，1933，11)。该作品以 3959 部**经典**(sutras)的形式描述了梵语的语言学，每一种都非常有效地(因为必须记住！)表达了形式规则系统的一部分，该系统很好地预言了形式语言理论的现代机制(Penn 和 Kiparsky，2012 年)。与本章中的讨论相关的一组规则描述了 kārakas，动词和名词论元之间的语义关系，诸如施事者、工具或目标之类的角色。Pānini 的作品是我们所知道的最早的作品，旨在了解事件及其参与者的语言实现。理解参与者与事件之间的关系这一任务(能够回答“谁对谁做了什么”(也许还有“何时何地”))是自然语言理解的核心问题。

让我们向前迈进 2.5 千年，并考虑一下了解 XYZ 公司购买股票的文字这一非常平凡的目标。可以通过各种表面形式来描述此购买事件及其参与者。事件可以用动词(sold, bought)或名词(purchase)来描述，XYZ Corp 可以是(of bought)句法主语，(of sold)间接宾语，或所有格或名词复合关系(带有名词 purchase)尽管在所有这些对象中在概念上都具有相同的作用：

- XYZ corporation bought the stock.
- They sold the stock to XYZ corporation.
- The stock was bought by XYZ corporation.
- The purchase of the stock by XYZ corporation...
- The stock purchase by XYZ corporation...

在这一章中，我们将介绍一个层次的表达，来抓住这些句子之间的共性:有一个购买事件，参与者是 XYZ 公司和一些股票，XYZ 公司是买家。这些浅层语义表示，语义角色，表达了一个谓词的论元在事件中所扮演的角色，在 PropBank 和 FrameNet 等数据库中进行了编码。我们将介绍语义角色标签，在句子中分配角色的任务，选择限制，谓词表达其论元的偏好，例如 eat 的主题通常是一些可食用的东西。

19.1. 语义角色

想想在第 15 章中，我们是如何为以下句子表达论元的含义的:

(19.1) Sasha broke the window.

(19.2) Pat opened the door.

新戴维森(neo-Davidsonian)事件对这两句话的表示是：

$$\exists e, x, y\ Breaking(e) \wedge Breaker(e, Sasha) \wedge BrokenThing(e, y) \wedge Window(y)$$

$$\exists e, x, y\ Opening(e) \wedge Opener(e, Pat) \wedge OpenedThing(e, y) \wedge Door(y)$$

在这个表示法中，break 和 open 的主语角色分别是 Breaker 和 Opener。这些深层角色(deep roles)是特定于每个事件的;Breaking 事件有 Breakers 主语角色, Opening 事件有 Openers 主语角色，如此等等。

如果我们要回答问题，进行推理，或者对这些事件做任何进一步的自然语言理解，我们需要知道更多关于这些论元的语义。Breakers 和 Openers 有一些共同点。他们都是自愿行为者，通常是有生命的，并且对其事件负有直接的因果责任。

主题角色(thematic roles)是捕捉 Breakers 和 Openers 之间语义共性的一种方式。我们说这两个动词的主语都是**施事者(agents)**。因此，AGENT 是代表抽象概念的主题角色，如自愿因果关系。同样，这两个动词的直接宾语，BrokenThing 和 OpenedThing，都是典型的无生命的宾语，它们在某种程度上受到动

[1] 图中显示的是来自**克什米尔**（Kashmir)的桦树皮手稿**鲁帕瓦塔**（Rupavatra)，一种基于 Pāninii 梵语（Sanskrit)语法的语法教科书。图片来自威康收藏（Wellcome Collection)。

作的影响。这些参与者的语义角色是**主题(theme)**。

Thematic Role	Definition
AGENT	The volitional causer of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional causer of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
CONTENT	The proposition or content of a propositional event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event

图 19-1：一些常用的主题角色及其定义

尽管主题角色是最古老的语言模型之一，但正如我们在上文所见，它们的现代表述是由 Fillmore(1968)和 Gruber(1965)提出的。虽然没有普遍认可的角色集，图 19.1 和 19.2 列出了各种计算论文中使用的一些主题角色，以及粗略的定义和例子。大多数主题角色集都有十几种角色，但我们会看到数量更少的角色集，但意义更抽象，以及数量非常多的角色集，这些角色都是特定于情境的。我们将对所有的角色集合使用通用术语**语义角色(semantic roles)**，无论大小。

Thematic Role	Example
AGENT	*The waiter* spilled the soup.
EXPERIENCER	*John* has a headache.
FORCE	*The wind* blows debris from the mall into our yards.
THEME	Only after Benjamin Franklin broke *the ice*...
RESULT	The city built a *regulation-size baseball diamond*...
CONTENT	Mona asked "*You met Mary Ann at a supermarket?*"
INSTRUMENT	He poached catfish, stunning them *with a shocking device*...
BENEFICIARY	Whenever Ann Callahan makes hotel reservations *for her boss*...
SOURCE	I flew in *from Boston*.
GOAL	I drove *to Portland*.

图 19-2：各种主题角色的一些原型例子

19.2. 动词替换

计算系统使用语义角色的主要原因是作为一种浅层的意义表示，它可以让我们做出简单的推断，而这些推断是不可能从纯粹的表面单词字符串，甚至从解析树中得出的。为了扩展前面的例子，如果文档说 Company A acquired Company B，那么我们想知道这回答了 Was Company B acquired 的查询吗？尽管这两个句子的表面句法非常不同。类似地，这种浅层的语义可以作为机器翻译中有用的中间语言。

语义角色有助于概括谓词论元的不同表面实现。例如，虽然 AGENT 通常被认为是句子的主语，但在其他情况下，THEME 也可以是主语。考虑一下动词 break 的主题论元的可能实现:

(19.3) *John broke the window.*
AGENT THEME

(19.4) *John broke the window with a rock.*
AGENT THEME INSTRUMENT

(19.5) *The rock broke the window.*
INSTRUMENT THEME

(19.6) *The window broke.*
THEME

(19.7) *The window was broken by John.*
THEME AGENT

这些例子表明 break(至少)有可能的论元 AGENT、THEME、和 INSTRUMENT。由动词构成的主题

角色论元集合通常被称为**主题格(thematic grid)**、**θ-格**或**格框架(case frame)**。我们可以看到，实现 break 论元有以下几种可能:

AGENT/Subject, THEME/Object
AGENT/Subject, THEME/Object, INSTRUMENT/PP_{with}
INSTRUMENT/Subject, THEME/Object
THEME/Subject

事实证明，许多动词可以在不同的句法位置上实现它们的主题作用。例如，像 give 这样的动词可以用两种不同的方式来实现 THEME 和 GOAL 论元:

(19.8) a. *Doris gave the book to Cary.*
AGENT THEME GOAL
b. *Doris gave Cary the book.*
AGENT GOAL THEME

这种多重论元结构的实现(break 可以以 AGENT, NSTRUMENT, 或者 THEME 为主语，give 可以以任意顺序实现其 THEME 和 GOAL) 称为**动词替换(verb alternations)**或**素质替换(diathesis alternations)**。上面我们展示的 give 的替换，即**承受格替换(dative alternation)**，似乎出现在特定语义类的动词中，包括“将来时态动词”(advance, allocate, offer, owe)，“发送动词”(forward, hand, mail)，“投掷动词”(kick, pass, throw)，等等。Levin(1993)为 3100 个英语动词列出了它们所属的语义类(47 个高级类，分为 193 个更具体的类)和它们参与的各种变化。这些动词类列表已经被合并到在线资源 VerbNet 中(Kipper 等人，2000)，它将每个动词连接到 WordNet 和 FrameNet 词条。

19.3. 语义角色：主题角色的问题

在主题角色层面表示意义似乎对处理诸如素质替换等复杂现象(complications)很有帮助。然而，事实证明，提出标准的角色集是相当困难的，同样也很难对角色(如 AGENT, THEME, 或者 INSTRUMENT)给出正式的定义。

例如，试图定义角色集的研究人员常常发现，他们需要将角色(如 AGENT 或 THEME)分割成许多特定的角色。Levin 和 Rappaport Hovav(2005)总结了许多这样的案例，例如，似乎至少有两种 INSTRUMENTS，中介工具可以作为主语出现，而使能工具则不能:

(19.9) a. The cook opened the jar with the new gadget.
b. The new gadget opened the jar.
(19.10) a. Shelly ate the sliced banana with a fork.
b. *The fork ate the sliced banana.

除了分裂问题之外，在某些情况下，我们希望对语义角色进行推理和归纳，但有限的离散角色列表不允许我们这样做。

最后，很难对主题角色进行正式界定。考虑 AGENT 角色;大多数情况下，AGENTS 是有生命的，有意志的，有知觉的，有因果关系的，但任何一个单独的名词短语可能都不具备所有这些属性。

这些问题导致了使用更少或更多角色的替换**语义角色(semantic role)**模型。

第一个选项是定义抽象于特定主题角色的**广义(generalized)语义角色**。例如，原型施事者(PROTO-AGENT)和原型受动者(PROTO-PATIENT)都是表达大致类似施事者和大致类似受动者含义的广义角色。这些角色不是由必要条件和充分条件来定义的，而是由一组启发式特征来定义的，这些特征具有更像施事者或更像受动者的含义。因此，一个论元显示出类似施事者的属性(自愿参与事件，在另一个参与者中引起事件或状态改变，被感知或有意参与，移动)越多，则该论元可以被标记为 PROTO-AGENT 的可能性就越大。属性越像受动者(经历状态的改变，受到另一个参与者的因果影响，相对于其他参与者静止不动，等等)，则可以将该论元标记为 PROTO-PATIENT 的可能性就越大。

第二个方向是定义一些语义角色，它们要么特定于一个独有的动词，要么特定于一组独有的语义相关的动词或名词。

在接下来的两节中，我们描述了两个常用的词汇资源，它们利用了语义角色的这些替代版本。PropBank 同时使用原型角色和特定于动词的语义角色。FrameNet 使用特定于一般语义概念(称为框架)的语义角色。

19.4. 命题库(PropBank)

命题库(Proposition Bank)，一般称为 **PropBank**，是具有语义角色标签的句子资源。英语 PropBank 对宾州树库中的所有句子进行了标签;在滨州汉语树库中，汉语 PropBank 标签句子。由于定义一套普遍的主题角色的困难，PropBank 中的语义角色是根据单个的动词意义来定义的。因此，每个动词的每种意义都有一组特定的角色，这些角色只给出数字而不是名称:Arg0、Arg1、Arg2，等等。通常，Arg0 表示 PROTO-AGENT，Arg1 表示 PROTO-PATIENT。其他角色的语义不太一致，通常是专门为每个动词定义的。尽管如此，还是有一些泛化;Arg2 通常是有益、工具、属性或结束状态，Arg3 通常是起始点、有益、工具或属性，Arg4 通常是结束点。

在 PropBank 中，对于每个动词 agree 和 fall 的一个意义，都包含一些稍微简化的词条目。这样的 PropBank 词条称为**框架文件(frame files)**。请注意，框架文件中每个角色的定义(“其他实体同意”，“范围，数量下降”)是旨在供人类阅读的非正式注释，而不是正式定义。

(19.11) **agree.01**

Arg0: Agreer
Arg1: Proposition
Arg2: Other entity agreeing

Ex1: [$_{Arg0}$ The group] *agreed* [$_{Arg1}$ it wouldn’t make an offer].
Ex2: [$_{ArgM-TMP}$ Usually] [$_{Arg0}$ John] *agrees* [$_{Arg2}$ with Mary] [$_{Arg1}$ on everything].

(19.12) **fall.01**

Arg1: Logical subject, patient, thing falling
Arg2: Extent, amount fallen
Arg3: start point
Arg4: end point, end state of arg1

Ex1: [$_{Arg1}$ Sales] *fell* [$_{Arg4}$ to $25 million] [$_{Arg3}$ from $27 million].
Ex2: [$_{Arg1}$ The average junk bond] *fell* [$_{Arg2}$ by 4.2%].

注意，fall 没有 Arg0 角色，因为 fall 的正常主语是一个 PROTO-PATIENT。

PropBank 语义角色在恢复关于动词的(verbal)论元的浅层语义信息方面是有用的。考虑动词 increase:

(19.13) **increase.01** “go up incrementally”

Arg0: causer of increase
Arg1: thing increasing
Arg2: amount increased by, EXT, or MNR
Arg3: start point
Arg4: end point

PropBank 语义角色标记可以让我们推断出以下三个例子的事件结构中的共性，即在每个例子中，Big Fruit Co.是 AGENT，而 the price of bananas 是 THEME，尽管表面形式不同。

(19.14) [$_{Arg0}$ Big Fruit Co.] increased [$_{Arg1}$ the price of bananas].
(19.15) [$_{Arg1}$ The price of bananas] was increased again [$_{Arg0}$ by Big Fruit Co.]
(19.16) [$_{Arg1}$ The price of bananas] increased [$_{Arg2}$ 5%].

PropBank 还有许多非编号论元，称为 ArgM(ArgM-TMP，ArgM-LOC，等)，它们表示修改或附加含

义。它们在谓词之间相对稳定，所以不会在每个框架文件中列出。使用这些修饰符标签的数据在训练系统中有助于检测谓词之间的时间、位置或方向修改。ArgM 的一些项目包括:

TMP	when?	yesterday evening, now
LOC	where?	at the museum, in San Francisco
DIR	where to/from?	down, to Bangkok
MNR	how?	clearly, with much enthusiasm
PRP/CAU	why?	because ... , in response to the ruling
REC		themselves, each other
ADV	miscellaneous	
PRD	secondary predication	...ate the meat raw

PropBank 侧重于动词，而相关项目 **NomBank**(Meyers 等人，2004)则在名词谓词上添加注释。例如，在 Apple's agreement with IBM 中的名词 agreement，和 Apple 一起被贴上标签 Arg0，和 IBM 一起被贴上标签 Arg2。这允许语义角色的贴标签者给动词类和名词性谓词的论元分配标签。

19.5. 框架网(FrameNet)

虽然用 increase 推断不同句子的语义共性是有用的，但如果我们能在更多的情况下，在不同的动词，以及动词和名词之间进行这样的推断，那就更有用了。例如，我们想提取这三个句子之间的相似之处:

(19.17) [$_{Arg1}$ The price of bananas] increased [$_{Arg2}$ 5%].

(19.18) [$_{Arg1}$ The price of bananas] rose [$_{Arg2}$ 5%].

(19.19) There has been a [$_{Arg2}$ 5%] rise [$_{Arg1}$ in the price of bananas].

请注意，第二个例子使用了不同的动词 rise，第三个例子使用了名词而不是动词 rise。我们希望一个系统认识到 the price of bananas 是上升,这 5%是上升,无论 5%作为动词 increased 的宾语出现或作为名词 rise 的名词性修饰语出现。

FrameNet 项目是另一个语义角色标记项目，它试图解决的正是这类问题(Baker 等人 1998, Fillmore 等人 2003, Fillmore 和 Baker 2009, Ruppenhofer 等人 2016)。PropBank 项目中的角色是特定于单个动词的，而 FrameNet 项目中的角色是特定于一个框架的。

什么是框架?考虑下面的一组单词:

reservation, flight, travel, buy, price, cost, fare, rates, meal, plane

在这个列表中的许多单词之间有许多下位关系、同义关系等单独的词汇关系。然而，由此产生的一组关系并不能完整地说明这些词是如何关联的。它们显然都是根据一组连贯的有关航空旅行的常识背景信息来定义的。

我们把把这些词结合在一起的整体背景知识称为**框架(frame)**(Fillmore,1985)。在人工智能和认知科学中，根据一些背景信息来定义单词组的想法很普遍，除了框架之外，我们看到相关的工作就像一个**模型(model)**(Johnson-Laird,1983)，甚至**脚本(script)**(Schank 和 Abelson,1977)。

FrameNet 中的框架是一个背景知识结构，它定义了一组特定于框架的语义角色，称为**框架元素(frame elements)**，并包括一组使用这些角色的谓词。每个词都能唤起一个框架，并勾勒出框架及其元素的某些方面。FrameNet 数据集包括一组框架和框架元素，与每个框架相关联的词汇单元，以及一组被标签的例句。例如，change_position_on_a_scale 框架定义如下:

This frame consists of words that indicate the change of an Item's postion on a scale (the Attribute) from a starting point (Initial_value) to an end point (Final_value).

图 19.3 定义了框架中的一些语义角色(框架元素)。注意，这些角色被分为核心角色(core roles)和非核心角色(non-core roles)，核心角色是特定于框架的，非核心角色更像 PropBank 中的 Arg-M 论元，表示时间、位置等更一般的属性。

下面是一些例句:

(19.20) [$_{ITEM}$ Oil] rose [$_{ATTRIBUTE}$ in price] [$_{DIFFERENCE}$ by 2%].

(19.21) [ITEM It] has increased [FINAL_STATE to having them 1 day a month].

(19.22) [ITEM Microsoft shares] fell [FINAL_VALUE to 7 5/8].

(19.23) [ITEM Colon cancer incidence] fell [DIFFERENCE by 50%] [GROUP among men].

(19.24) a steady increase [INITIAL_VALUE from 9.5] [FINAL_VALUE to 14.3] [ITEM in dividends]

(19.25) a [DIFFERENCE 5%] [ITEM dividend] increase...

Core Roles	
ATTRIBUTE	The ATTRIBUTE is a scalar property that the ITEM possesses.
DIFFERENCE	The distance by which an ITEM changes its position on the scale.
FINAL_STATE	A description that presents the ITEM's state after the change in the ATTRIBUTE's value as an independent predication.
FINAL_VALUE	The position on the scale where the ITEM ends up.
INITIAL_STATE	A description that presents the ITEM's state before the change in the ATTRIBUTE's value as an independent predication.
INITIAL_VALUE	The initial position on the scale from which the ITEM moves away.
ITEM	The entity that has a position on the scale.
VALUE_RANGE	A portion of the scale, typically identified by its end points, along which the values of the ATTRIBUTE fluctuate.
Some Non-Core Roles	
DURATION	The length of time over which the change takes place.
SPEED	The rate of change of the VALUE.
GROUP	The GROUP in which an ITEM changes the value of an ATTRIBUTE in a specified way.

图 19-3：框架元素

图注：change_position_on_a_scale 框架中的框架元素，来自 FrameNet 标签指南(Ruppenhofer 等人，2016)。

从这些例句中可以看出，框架包含了目标词，如 rise, fall 和 increase。事实上，完整的框架由以下几个字组成:

VERBS:	dwindle	move	soar	escalation	shift
advance	edge	mushroom	swell	explosion	tumble
climb	explode	plummet	swing	fall	
decline	fall	reach	triple	fluctuation	**ADVERBS:**
decrease	fluctuate	rise	tumble	gain	increasingly
diminish	gain	rocket		growth	
dip	grow	shift	**NOUNS:**	hike	
double	increase	skyrocket	decline	increase	
drop	jump	slide	decrease	rise	

FrameNet 还对框架之间的关系进行编码，允许框架之间相互继承，或者表示框架之间的关系，比如因果关系(不同框架中框架元素之间的泛化也可以通过继承来表示)。因此，存在一个 Cause_change_of_position_on_a_scale 框架，该框架通过 cause(因果)关系连接到 Change_of_position_on_a_scale 框架，但该框架添加了 AGENT 角色，并用于引起这种情况的如下示例：

(19.26) [AGENT They] raised [ITEM the price of their soda] [DIFFERENCE by 2%].

这两个框架在一起，将允许一个理解系统去提取所有动词类和名词性因果关系用法和非因果关系用法的共同事件语义。

FrameNets 还针对许多其他语言进行了开发，如西班牙语、德语、日语、葡萄牙语、意大利语和汉语。

19.6. 语义角色标签(SRL)

语义角色标签(Semantic role labeling，SRL)是指自动找出句子中每个谓词的每个论元的语义角色。当前的语义角色标签方法是基于监督机器学习的，通常使用 FrameNet 和 PropBank 资源来指定什么可以作为谓词，定义任务中使用的角色集，并提供训练和测试集。

回想一下，这两种语义角色模型之间的区别是：FrameNet(19.27)使用许多特定于框架的框架元素作为角色，而 PropBank(19.28)使用较少的编号论元标签，它们可以被解释为特定于动词的标签，以及更通用的 ARGM 标签。一些例子:

(19.27) [You] can't [blame] [the program] [for being unable to identify it]
COGNIZER TARGET EVALUEE REASON

(19.28) [The San Francisco Examiner] issued [a special edition] [yesterday]
ARG0 TARGET ARG1 ARGM-TMP

19.6.1. 基于特征的语义角色标签算法

一种简化的基于特征的语义角色标签算法如图 19.4 所示。基于特征的算法——来自最早的系统，如(Simmons, 1973)——从解析开始，使用广泛的解析器将解析分配给输入字符串。图 19.5 显示了上面(19.28)的解析。然后遍历解析以查找所有谓词的单词。

对于这些谓词中的每一个，该算法都会检查语法分析树中的每个节点，并使用监督分类来确定其在该谓词中所扮演的语义角色(如果有)。给定带标签的训练集(例如 PropBank 或 FrameNet)，使用下一部分描述的特征模板为每个节点提取特征向量。然后训练 1-of-N 分类器，以预测给定这些特征的每个成分的语义角色，其中 N 是潜在语义角色的数量加上非角色成分的额外 NONE 角色。可以使用任何标准分类算法。最后，对于被标签的每个测试语句，对每个相关成分运行分类器。

```
function SEMANTICROLELABEL(words) returns labeled tree

  parse ← PARSE(words)
  for each predicate in parse do
    for each node in parse do
      featurevector ← EXTRACTFEATURES(node, predicate, parse)
      CLASSIFYNODE(node, featurevector, parse)
```

图 19-4：一个通用的语义-角色-标签算法

图注：CLASSIFYNODE 是一个 1-of-N 的分类器，它指定一个语义角色(对于非角色成分，则为 NONE)，训练的对象是有标签的数据，如 FrameNet 或 PropBank。

与图 19.5 所示的单阶段分类器训练不同，节点级分类任务可以分解为多个步骤:

1.修剪:由于一个句子中只有少量的成分是任何给定谓词的论元，许多系统使用简单的启发式来修剪不太可能的成分。

2.识别:每个节点的二元分类，作为被标签的论元或 NONE。

3.分类:前一阶段被标签为论元的所有成分的 1-of-N 分类。

区分识别和分类可以更好地使用特征(不同的特征可能对两个任务有用)或提高计算效率。

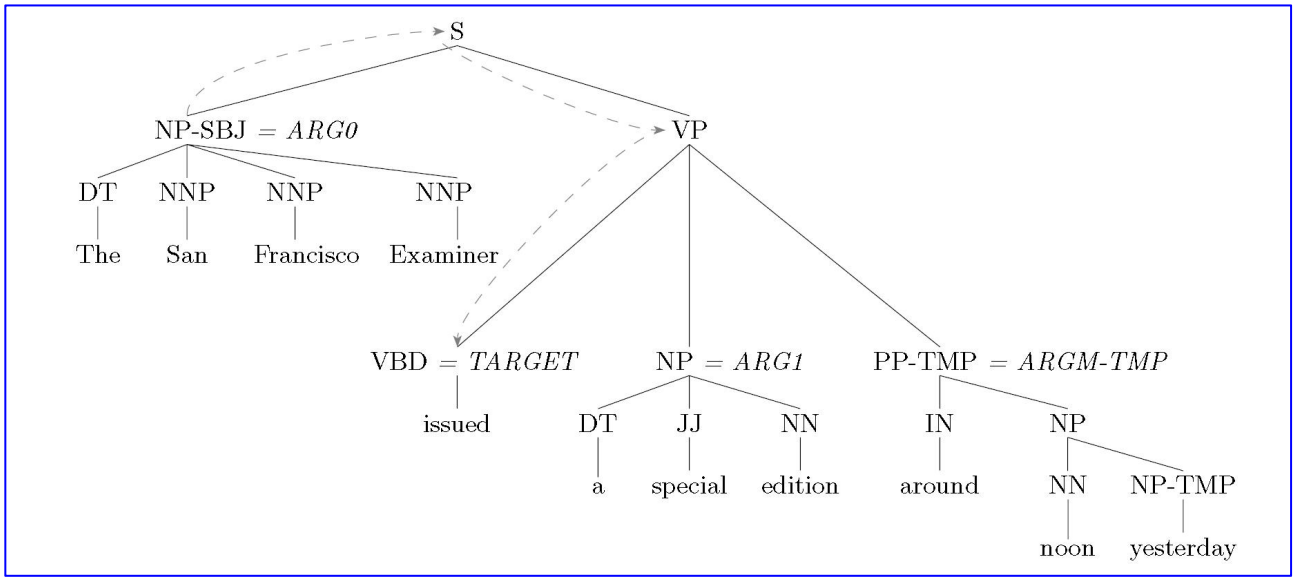


图 19-5：PropBank 句子的解析树

图注：显示了 PropBank 的论元标签，虚线显示 ARG0 的路径特征 NP↑S↓VP↓VBD，NP-SBJ 成分 The San Francisco Examiner。

全局优化

图 19.5 的分类算法对每个论元分别进行分类(“局部”)，简化假设一个谓词的每个论元都可以独立地被标签。这种假设是错误的;论元之间存在交互，需要对成分进行更“全局”的标签分配。例如，FrameNet 和 PropBank 中的成分必须是不重叠的。更重要的是，成分的语义角色并不是独立的。例如，PropBank 不允许多个相同的论元;同一个动词的两个成分不能同时被标签为 ARG0。

因此，角色标签系统通常会增加第四个步骤来处理句子中所有标签的全局一致性。例如，本地分类器可以返回与每个成分的概率相关联的可能标签列表，并且可以使用第二次 Viterbi 解码或重新排序方法来选择最佳一致标签。整数线性规划(ILP)是另一种选择最符合多个约束的解决方案的常用方法。

语义角色标签的特征

大多数系统使用 Gildea 和 Jurafsky(2000)介绍的核心特征集的一些概括。常见的基本特征模板(在 NP-SBJ 成分的 The San Francisco Examiner 演示，见图 19.5)包括:

•支配谓词，在这种情况下是动词 issued。谓词是一个至关重要的特性，因为标签只针对特定的谓词定义。

•成分的短语类型，在这里是 NP(或 NP-SBJ)。一些语义角色以 NP 的形式出现，另一些以 S 或 PP 的形式出现，等等。

•成分的中心词，Examiner。成分的中心词可以用标准的中心词规则来计算，例如第 12 章图 12.12 中给出的规则。某些词(如代词)对它们可能扮演的语义角色施加了强烈的限制。

•成分的中心词词类，NNP。

•解析树中从成分到谓词的路径。此路径以图 19.5 中的虚线标示。根据 Gildea 和 Jurafsky(2000)，我们可以使用一个简单的路径线性表示，NP↑S↓VP↓VBD。↑、↓分别表示树的向上、向下运动。作为成分和谓词之间的多种语法功能关系的紧凑表示，路径非常有用。

•在这种情况下，从句的语态是主动语态(与被动语态相对)。与主动句相比，被动句的语义角色与表面形式之间的联系有很大的不同。

•成分相对于谓词的二元线性位置，在前面或后面。

•谓词的子范畴化，动词短语中出现的一组预期的论元。我们可以使用短语结构规则来提取这个信息，该规则扩展了谓词的直接父元素；图 19.5 中的谓词为 VP→VBD NP PP。

•成分的命名实体类型。

•成分的第一个词和最后一个词。

因此，下面的特征向量代表了我们例子中的第一个 NP(回想一下，大多数观察值都是 NONE，而不是 ARG0，因为解析树中的大多数成分都不承担语义角色):

ARG0: [issued, NP, Examiner, NNP, NP↑S↓VP↓VBD, active, before, VP →NP PP, ORG, The, Examiner]

此外，还经常使用其他特性，比如成分内部的 n-gram 集合，或者路径特性的更复杂版本(向上或向下的一半，或者特定节点是否出现在路径中)。还可以使用依存解析而不是成分解析作为特征的基础，例如使用依存解析路径而不是成分解析路径。

19.6.2. 语义角色标签的神经算法

SRL 的一种简单的神经方法是使用 BIO 方法将其视为序列标签任务，就像命名实体识别一样。让我们假设我们得到了谓词，任务只是检测和标签跨度。回想一下，对于 BIO 标记，我们为每个可能的角色都有一个开始和结束标记(B-ARG0, I-ARG0;B-ARG1, I-ARG1，等等)，加上一个外部标记 O。与所有标记一样，给定单词 w 的输入序列，目标是计算标记序列$\hat{\mathbf{y}}$的最高概率：

$$\hat{\mathbf{y}} = \underset{y \in T}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{w})$$

图 19.6 为 Shi 和 Lin(2019)以及 He 等(2017)的标准算法草图。在这里，每个输入单词被映射到预先训练的嵌入，并且还与一个标志(0/1)变量的嵌入相关联，该变量指示该输入单词是否为谓词。这些连接的

嵌入通过一个双向 LSTM。每个符记的输出嵌入与谓词的输出嵌入拼接，然后传递给带有 softmax 的 MLP，该 MLP 在每个 SRL 标签上输出分布。

对于解码，可以使用 CRF 层来代替位于双向 LSTM 输出之上的 MLP 层(或者我们甚至可以不使用双向 LSTM，直接将拼接的 BERT 输出传递给 CRF)。Viterbi 解码算法(第 8 章)可以用来从 CRF 解码，这可能有助于利用标记之间的全局约束，例如标记 I-ARG0 必须遵循另一个 I-ARG0 或 B-ARG0。

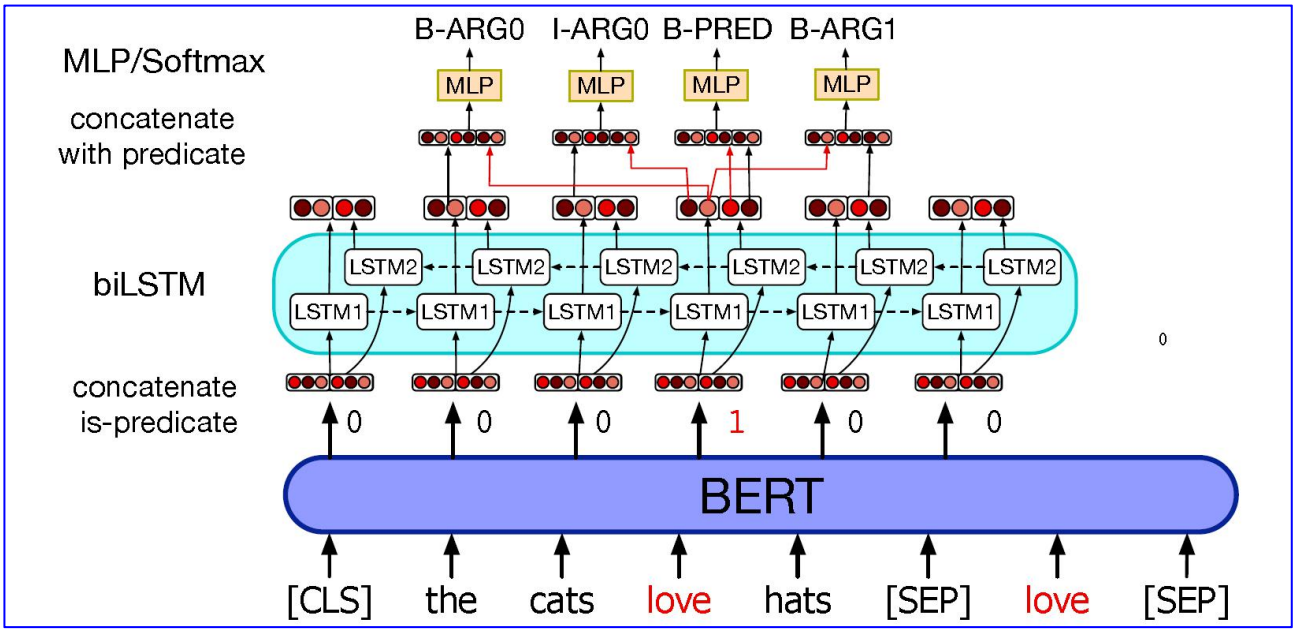


图 19-6：BERT + 双向 LSTM 语义角色标签方法

图注：输入句子后面跟着[SEP]和谓词的额外输入，在本例中是 love。BERT 输出拼接到一个指示符变量(谓词为 1，其他所有单词为 0)，通过一个双向 STM 传递，然后嵌入到每个符记位置的输出与谓词的嵌入拼接，并通过单层 MLP 传递。

19.6.3. 语义角色标签的评价

语义角色标签的标准评估是要求每个论元标签必须被分配到完全正确的单词序列或解析成分，然后计算精度、召回率和 F-测度。识别和分类也可以分别评价。两种常用的评估数据集是 CoNLL-2005 (Carreras 和 M'arquez, 2005)和 CoNLL-2012 (Pradhan 等人，2013)。

19.7. 选择限制

在本节中，我们将转向另一种方式来表示谓词和论元之间关系。**选择限制(selectional restriction)**是动词对允许填充其论元角色的概念类型施加的语义类型约束。考虑与以下例子相关的两种含义:

(19.29) I want to eat someplace nearby

这个句子有两种可能的解析和语义解释。在合理的解释中，eat 是不及物动词，短语 somewhere nearby 是一个附加词，给出了吃饭事件的地点。在无意义的[speaker-as-Godzilla]的解释中，eat 是及物动词，短语 somewhere nearby 是吃的直接宾语和主主语，如下面句子中的 NP Malaysian food:

(19.30) I want to eat Malaysian food.

我们怎么知道 someplace nearby 不是这句话的直接对象？一个有用的提示是语义上的事实，即 THEME of EATING 事件往往是 edible(可食用的)。动词 eat 在其 THEME 论元的填充词上施加的限制是一种选择限制。

选择限制与意义相关，而不是与整个词素相关。我们可以在以下词素 serve 的示例中看到这一点：

(19.31) The restaurant serves green-lipped mussels.

(19.32) Which airlines serve Denver?

例(19.31)说明了 serve 的提供食物的意义，通常将其 THEME 限制为某种食物。例(19.32)说明了为 serve 的提供商业服务意义，从而将其 THEME 限制为某种合适的位置。

选择限制的特殊性差别很大。例如，动词 imagine 对其 AGENT 角色提出了严格的要求(将其限制为人类和其他有生命的实体)，但对其 THEME 角色很少提出语义要求。另一方面，像对角化(diagonalize)这样的动词对其 THEME 角色的填充物施加了非常具体的限制:它必须是一个矩阵，而形容词 odorless 的论元则仅限于可能具有气味的概念:

(19.33) In rehearsal, I often ask the musicians to imagine a tennis game.

(19.34) Radon is an odorless gas that can't be detected by human senses.

(19.35) To diagonalize a matrix is to find its eigenvalues.

这些例子说明了我们需要用来表示选择限制的一系列概念(矩阵，能够拥有气味等)是非常开放的。这将选择限制与其他词汇知识的表示特征区别开来，比如数量相当有限的词类。

19.7.1. 选择限制的表示

捕获选择限制语义的一种方法是使用和扩展第 15 章的事件表示。回想一下，neo-Davidsonian 对事件的表示由代表事件的单一变量、表示事件类型的谓词、以及事件角色的变量和关系组成。忽略λ结构的问题，使用主题角色而不是深层事件角色，像 eat 这样的动词的语义贡献可能如下:

$$\exists e, x, y\ Eating(e) \wedge Agent(e, x) \wedge Theme(e, y)$$

有了这个表示，我们对 y(THEME 角色的填充物)的全部了解就是它通过 Theme 关系与一个 Eating 事件相关联。为了规定选择限制 y 必须是 edible，我们简单地增加了一个新术语:

$$\exists e, x, y\ Eating(e) \wedge Agent(e, x) \wedge Theme(e, y) \wedge EdibleThing(y)$$

当遇到像 ate a hamburger 这样的短语时，语义分析器可以形成以下几种表现形式:

$$\exists e, x, y\ Eating(e) \wedge Eater(e, x) \wedge Theme(e, y) \wedge EdibleThing(y) \wedge Hamburger(y)$$

这种表示是完全合理的，因为假设知识库中有一组合理的事实，y 在 Hamburger 类别中的成员资格与其在 EdibleThing 类别中的成员资格是一致的。相应地，由于诸如 ate a takeoff 这样的短语的表述形式会不正确，因为类似事件的类别诸如 Takeoff 中的成员资格将与 EdibleThing 类别中的成员资格不一致。

虽然这种方法充分地捕获了选择限制的语义，但直接使用它存在两个问题。首先，使用 FOL 执行强制选择限制的简单任务是过分的。其他简单得多的形式可以用更少的计算成本完成这项工作。第二个问题是，这种方法预先假定了一个关于构成选择限制的概念的事实的庞大的、逻辑的知识库。不幸的是，虽然这些常识性的知识库正在开发中，但目前没有一种具有这项任务所必需的覆盖范围。

一种更实用的方法是使用 WordNet 语法集来声明选择限制，而不是使用逻辑概念。每个谓词只是指定一个 WordNet 同义词库作为对其每个论元的选择限制。如果角色填充词是该同义词库的下位词(从属)，则含义表示形式良好。

例如，对于 ate a hamburger 的例子，可以将动词 eat 的 THEME 角色的选择限制设置为同义词集{食物，营养}，解释为 any substance that can be metabolized by an animal to give energy and build tissue(任何可以被动物代谢以提供能量和构建组织的物质)。幸运的是，图 19.7 所示的汉堡包的上义词链显示汉堡包确实是食物。同样，角色的填充不需要完全匹配约束同义词集;它只需要将同义词集作为其上级之一即可。

```
Sense 1
hamburger, beefburger --
(a fried cake of minced beef served on a bun)
=> sandwich
   => snack food
      => dish
         => nutriment, nourishment, nutrition...
            => food, nutrient
               => substance
                  => matter
                     => physical entity
                        => entity
```

图 19-7：WordNet 的证据表明汉堡包是可以食用的

我们可以将这种方法应用于前面讨论过的动词 imagine、lift 和 diagonalize 的 THEME 角色。让我们把 imagine 的 THEME 限制为同义词集{entity}，lift 的 THEME 为{physical entity}，diagonalize 的 THEME 为{matrix}。这种安排正确地允许 imagine a hamburger 和 lift a hamburger，同时也正确地排除了 diagonalize a hamburger。

19.7.2. 选择偏好

在最早的实现中，选择限制被认为是对谓词可以接受的论元类型的严格限制(Katz 和 Fodor 1963, Hirst 1987)。例如，动词 eat 可能要求其 THEME 论元为[+FOOD]。早期的词义消歧系统使用这个想法来排除那些违反其主导谓词的选择限制的词义。然而，很快，这些选择限制被更好地表示为偏好，而不是严格的限制(Wilks 1975c, Wilks 1975b)。例如，违背选择限制(如 eat 的 inedible 论元)经常出现在结构良好的句子中，例如因为它们被否定了(19.36)，或者因为选择限制被夸大了(19.37):

(19.36) But it fell apart in 1931, perhaps because people realized you can't eat gold for lunch if you're hungry.

(19.37) In his two championship trials, Mr. Kulkarni ate glass on an empty stomach, accompanied only by water and tea.

因此，用于选择偏好的现代系统使用某种软约束来指定谓词及其可能论元之间的关系。

选择关联

最有影响的一种是 Resnik(1993)的选择关联(selectional association)模型。Resnik 将选择偏好强度(selectional preference strength)的概念定义为谓词告诉我们有关其论元的语义类别的一般信息量。例如，动词 eat 告诉我们很多有关其直接宾语的语义类的信息，因为它们往往是可食用的。相比之下，动词 be 告诉我们的直接宾语更少。选择偏好强度可以通过两个分布之间的信息差异来定义：预期语义类别 P(c)的分布(直接宾语落入类别 c 的可能性有多大)；特定动词 P(c | v)的预期语义类别的分布(特定动词 v 的直接宾语落入语义类别 c 的可能性有多大)。这些分布之间的差异越大，该动词为我们提供的有关可能宾语的信息就越多。这两个分布之间的差异可以通过相对熵(relative entropy)或 Kullback-Leibler 散度(divergence)来量化(Kullback 和 Leibler，1951 年)。KL 散度 D(P || Q)表示两个概率分布 P 和 Q 之间的差异：

$$D(P||Q) = \sum_x P(x)\log\frac{P(x)}{Q(x)} \tag{19.38}$$

选择偏好 SR(v)使用 KL 散度来表示动词 v 表达了多少关于其论元可能的语义类的信息(以比特为单位)。

$$\begin{aligned} S_R(v) &= D(P(c|v)||P(c)) \\ &= \sum_c P(c|v)\log\frac{P(c|v)}{P(c)} \end{aligned} \tag{19.39}$$

然后，Resnik 将特定类别与动词的**选择关联(selectional association)**定义为该类别对动词的一般选择偏好的相对贡献:

$$A_R(v,c) = \frac{1}{S_R(v)}P(c|v)\log\frac{P(c|v)}{P(c)} \tag{19.40}$$

因此，选择关联是谓词和支配谓词论元的类之间关联强度的概率度量。Resnik 通过分析语料库来估算这些关联的概率，计算每个谓词与每个论元词一起出现的所有次数，并假设每个词都是对包含该词的所有 WordNet 概念的部分观察。下表来自 Resnik(1996)，展示了动词高、低选择性关联的一些样本，以及它们直接对象的一些 WordNet 语义类。

Verb	Direct Object Semantic Class	Assoc	Direct Object Semantic Class	Assoc
read	WRITING	6.80	ACTIVITY	-.20
write	WRITING	7.26	COMMERCE	0
see	ENTITY	5.79	METHOD	-0.01

通过条件概率的选择偏好

在动词和它的论元的 WordNet 类之间使用选择性关联的另一种方法是使用给定谓词动词的论元词的条件概率，直接建模一个动词(谓词)与一个名词(论元)的关联强度。

条件概率模型可以通过解析一个非常大的语料库(数十亿个单词)和统计共现计数来计算：给定动词与给定名词在给定关系中出现的频率。给定一个特定关系 P（n | v，r）的动词，一个论元名词的条件概率可以用作该词对(n，v)的选择偏好度量(Brockmann 和 Lapata 2003, Keller 和 Lapata 2003)：

$$P(n|v,r) = \begin{cases} \frac{C(n,v,r)}{C(v,r)} & \text{if } C(n,v,r) > 0 \\ 0 & \text{otherwise} \end{cases}$$

反向概率 P(v|n,r)被发现在某些情况下有更好的性能(Brockmann 和 Lapata, 2003):

$$P(v|n,r) = \begin{cases} \frac{C(n,v,r)}{C(n,r)} & \text{if } C(n,v,r) > 0 \\ 0 & \text{otherwise} \end{cases}$$

一个更简单的方法是使用带有论元 log count(v,n,r)的谓词的简单对数共现频率，而不是条件概率；这似乎能更好地提取语法主语的偏好，而不是宾语(Brockmann 和 Lapata, 2003)。

评估选择偏好

评估选择偏好模型的一种方法是使用**伪词(pseudowords)**(Gale 等人 1992c, Schutze 1992a)。伪词是将一个测试词(比如香蕉)和一个混淆词(比如门)连接在一起而产生的一个人造词(比如香蕉门)。该系统的任务是识别两个单词中哪个是原单词。为了评估选择偏好模型(例如动词和直接宾语之间的关系)，我们使用一个测试语料库并选择所有动词符记。对于每一个动词符记(如 drive)，我们选择直接宾语(如 car)，然后连接一个与其最近的混淆词，即频率与原始词(如 house)最近的名词，构成 car/house。然后，我们使用选择偏好模型来选择 car 和 house 中哪个是更受 drive 偏好的宾语，并计算模型选择正确原始宾语(如 car)的频率(Chambers 和 Jurafsky, 2010)。

另一个评估指标是获得人类对一组动词-论元对的测试集的偏好，并让他们对其可信程度进行评级。这通常是通过使用幅度(magnitude)估计来完成的，这是一种来自心理物理学的技术，在这一技术中，受试者对一个与模数成比例的论元的可信性进行评级。选择偏好模型可以通过其与人类偏好的相关性来评估(Keller 和 Lapata, 2003)。

19.8. 谓词的原始分解

我们在本章中讨论的语义角色的一种思考方式是，基于有限的主题角色列表(agent、patient、instrument、原型 agent、原型 patient 等)，它们帮助我们以分解的方式定义论元所扮演的角色。这种将含义分解为一组原始语义元素或特征的想法，称为**原始分解(primitive decomposition)**或**成分分析(componential analysis)**，已经得到了进一步的发展，并且特别关注谓词。

让我们来看看 kill 这个动词的例子:

(19.41) Jim killed his philodendron.

(19.42) Jim did something to cause his philodendron to become not alive.

从真值条件(命题语义学)的角度来看，这两个句子具有相同的含义。假设这是等价的，我们可以将 kill 的含义表示为:

(19.43) KILL(x,y) ⇔ CAUSE(x, BECOME(NOT(ALIVE(y))))

因此，使用像 do、cause、become not 和 alive 这样的语义原语。

事实上，一组这样的潜在语义原语已经被用来解释 19.2 节(Lakoff 1965, Dowty 1979)中讨论的一些言语变化。考虑下面的例子。

(19.44) John opened the door. ⇒ CAUSE(John, BECOME(OPEN(door)))

(19.45) The door opened. ⇒ BECOME(OPEN(door))

(19.46) The door is open. ⇒ OPEN(door)

分解方法断言，与 open 相关联的单个状态类谓词是所有这些示例的基础。这些例子含义上的差异，是由于这个单一谓词与原语的“CAUSE”和“BECOME”相结合而产生的。

虽然这种原始分解方法可以解释状态和动作之间的相似性，或者因果谓词和非因果谓词之间的相似性，但它仍然依赖于拥有大量像 open 这样的谓词。更激进的方法也选择分解这些谓词。在早期自然语言理解系统中发挥作用的动词谓词分解方法之一是**概念依存(conceptual dependency，CD)**，它由 10 个原始谓词组成，如图 19.8 所示。

Primitive	Definition
ATRANS	The abstract transfer of possession or control from one entity to another
PTRANS	The physical transfer of an object from one location to another
MTRANS	The transfer of mental concepts between entities or within an entity
MBUILD	The creation of new information within an entity
PROPEL	The application of physical force to move an object
MOVE	The integral movement of a body part by an animal
INGEST	The taking in of a substance by an animal
EXPEL	The expulsion of something from an animal
SPEAK	The action of producing a sound
ATTEND	The action of focusing a sense organ

图 19-8：一组概念依存(CD)原语

下面是一个示例句子及其 CD 表示。动词 brought 被翻译成两个原语 ATRANS 和 PTRANS，表示 waiter 既亲自把支票传递给 Mary，又把控制权传递给她。请注意，CD 还将一组固定的主题角色与每个原语关联起来，以表示动作中的各种参与者。

(19.47) The waiter brought Mary the check.

$$\exists x, y\ Atrans(x) \wedge Actor(x, Waiter) \wedge Object(x, Check) \wedge To(x, Mary)$$
$$\wedge Ptrans(y) \wedge Actor(y, Waiter) \wedge Object(y, Check) \wedge To(y, Mary)$$

19.9. 总结

- **语义角色**是一个论元在谓词描述的事件中所扮演的角色的抽象模型。
- **主题角色**是基于单一有限角色列表的语义角色模型。其他语义角色模型包括每个动词(per-verb)语义角色列表和**原始施事/原始受事(proto-agent/proto-patient)**，它们都在 **PropBank** 中实现，以及在 **FrameNe**t 中实现的每个框架角色列表。
- **语义角色标签**是为句子的成分指定语义角色标签的任务。该任务通常被视为有监督的机器学习任务，在 PropBank 或 FrameNet 上训练模型。算法通常从解析一个句子开始，然后用语义角色自动标记每个解析树节点。神经模型直接从单词端到端映射。
- 语义**选择限制**允许单词(特别是谓词)对其论元单词的语义属性进行限制。**选择偏好**模型(如**选择关联**或简单条件概率)允许为谓词和论元词或类之间的关联分配权重或概率。

19.10. 文献和历史说明

虽然语义角色的概念可以追溯到 Panini，但 Gruber(1965)，Fillmore (1966)和 Fillmore (1968)将语义角色重新引入现代语言学。Fillmore 通过研究 Lucien Tesnière 开创性的《句法结构》(Éléments de Syntax Structurale，1959)对论元结构产生了兴趣，在这本书中引入了“依存关系”一词，并为依存关系语法奠定了基础。根据 Tesnière 的术语，Fillmore 首先将论元角色称为 actants(Fillmore, 1966)，但很快就转到格(case)这个词(参见 Fillmore(2003))，并提出了一个通用的语义角色或格列表(Agent, Patient, Instrument 等)，这些语义角色或格可以由谓词的论元来承担。动词将与它们的格框架一起在词汇中列出，即强制(或可选)的格论元列表。

语义角色可以提供中间级别的语义表示的想法，可以帮助从句法分析结构映射到更深的，更全面的含义表示，这在自然语言处理中很快被采用，并且创建了提取格框架的系统以进行机器翻译(Wilks，1973)，问题回答(Hendrix 等人，1973)，口头语言理解(Nash-Webber，1975)和对话系统(Bobrow 等人，1977)。开发了通用语义角色标签器。最早的(Simmons，1973)首先通过 ATN(Augmented Transition Network，增强过渡网络)解析器解析了一个句子。然后，每个动词都有一组规则，这些规则指定解析应如何映射到语义角色。这些规则主要参考语法功能(主语，宾语，特定介词的补语)，但还检查了内部的成分特征，例如中心词名词的生命力(animacy)。后来的系统又通过使用带有动词特定格框架的字典来从预建的解析树中分配角色(Levin 1977，Marcus 1980)。

到 1977 年，格表示在人工智能和自然语言处理课程中被广泛使用和教授，并在 Winston(1977)的教科书《人工智能》第一版中被描述为自然语言理解的标准。Fillmore 在 20 世纪 80 年代提出了他的框架语义模型，后来对直觉的描述如下:

> 框架语义学背后的理念是，说话者意识到可能相当复杂的情境类型，联系期望的包，它们有不同的名称——框架、图式、场景、脚本、文化叙事、模因(memes)——我们语言中的词汇被理解为以框架作为预设背景。(Fillmore, 2012, p. 712)

框架这个词似乎浮现于一系列相关概念提出的同一时间,Minsky (1974), Hymes (1974), 和 Goffman (1974)，以及与其他相关概念如脚本(Schank 和 Abelson,1975)和图式(Bobrow 和 Norman,1975)(见 Tannen(1979)比较)。Fillmore 也受到语义场理论学家和耶鲁人工智能实验室的影响，在那里，他注意到早期信息提取系统(如 DeJong(1982),Schank 和 Abelson(1977))使用的槽和填充物的列表。20 世纪 90 年代，Fillmore 利用这些见解，开始了 FrameNet 语料库评注项目。

与此同时，Beth Levin 利用她早期的格框架字典(Levin, 1977)来发展她的书，其中总结了由共享论元实现所定义的动词类集(Levin, 1993)。VerbNet 项目建立在这项工作的基础上(Kipper 等人，2000)，随后很快导致了由 Martha Palmer 及其同事创建的 PropBank 语义角色标签语料库(Palmer 等人，2005)。

FrameNet 和 PropBank 实例化的丰富的语言标注和基于语料库的方法的结合，导致了语义角色标签的自动方法的复兴，首先是在 FrameNet (Gildea 和 Jurafsky, 2000)，然后是在 PropBank 数据(Gildea 和 Palmer, 2002，除其它之外)。这个问题最早是在 20 世纪 70 年代由手写规则解决的，因此现在通常被重新定义为由大型且一致的数据库支持的有监督的机器学习。Gildea 和 Jurafsky(2002)、Surdeanu 等人(2003)、Xue 和 Palmer(2004)、Pradhan 等人(2005)、Che 等人(2009)和 Zhao 等人(2009)定义了许多常用的角色标签特征。在 CoNLL-2008 共享任务(Surdeanu 等人，2008)中引入了依存而不是成分解析的使用。调查见 Palmer 等人(2010)和 Marquez 等人(2008)。

Collobert 等人(2011)率先使用神经方法对语义角色贴标签，他们在卷积网络上应用了 CRF。Foland, Jr. 和 Martin(2015)的早期工作主要集中在使用依存特征上。后来的作品完全避开了语法特征;Zhou 和 Xu (2015b)介绍了堆叠(6-8 层)biLSTM 架构的使用，并且(He 等人，2017)展示了如何利用高速网络来增强 biLSTM 架构，并将 CRF 替换为 A*解码，从而可以应用各种各样的全局约束于 SRL 解码。

大多数语义角色标签方案只在一个句子中起作用，集中在动词谓词(或名词类谓词，在 NomBank 的情况下)的宾语上。然而，在许多情况下，动词或名词类谓词可能有一个**隐含论元(implicit argument)**:它只出现在上下文句子中，或者可能根本就没有，必须进行推断。在 This house has a new owner 和 The sale

was finalized 10 days ago 这两个句子中，第二句中的 sale 没有 ARG1，但理性的读者会推断 ARG1 应该是前面句子中提到的 house。为了发现这些论元，Gerber 和 Chai(2010)以及 Ruppenhofer 等人(2010)引入了**隐式论元检测(iSRL)** 。参见 Do 等人(2017)，了解最新的神经模型。

为了避免需要大量的标签训练集，无监督的语义角色标签方法试图通过对论元的聚类来诱导语义角色集。该任务由 Riloff 和 Schmelzenbach(1998)以及 Swier 和 Stevenson(2004)率先提出;参见 Grenager 和 Manning (2006)，Titov 和 Klementiev (2012)，Lang 和 Lapata (2014)，Woodsend 和 Lapata (2015)，Titov 和 Khoddam(2014)。

近年来框架标签的创新包括内涵框架，它标记了关于谓词的论元的更丰富的信息。内涵框架标志着作者或读者对论元的情感(例如，在 he survivors a bombing 中使用动词 survive)表达作者对主语 he 的同情和对 bombing 的负面情绪。更多细节见第 20 章。

除了 Resnik(1993)和 Resnik(1996)的选择关联模型外，选择偏好也得到了广泛的研究。方法包括聚类(Rooth 等人，1999)、判别学习(Bergsma 等人，2008a)和主题模型(Seaghdha 2010, Ritter 等人 2010)，约束可以在单词或类的水平上表达(Agirre 和 Martinez, 2001)。选择偏好也被成功地整合到语义角色标签中(Erk 2007, Zapirain 等人 2013, Do 等人 2017)。

19.11. 练习

(无)

20. 情感、感情和内涵的词汇

Some day we'll be able to measure the power of words

Maya Angelou

在本章中，我们将转向解释**感情含义(affective meaning)**的工具，扩展我们在第四章中对情感分析的研究。我们使用**感情(affective)**这个词，遵循了**感情计算(affective computing)**的传统(Picard, 1995)来表示情绪、情感、性格、语气和态度。与感情含义密切相关的是主观性：对说话者或作者的评价、意见、情绪和推测的研究(Wiebe 等人，1999)。

感情含义应该如何定义?一种有影响力的感情状态类型来自 Scherer(2000)，他用认知实现和时间进程等因素来定义每一类感情状态(图 20.1)。

情绪(Emotion)：对具有重大意义的外部或内部事件的评价所产生的相对短暂的反应。
(愤怒、悲伤、喜悦、恐惧、羞愧、骄傲、得意、绝望)

语气(Mood)：弥漫的感情状态，最显著的是主观感觉的变化，强度低但持续时间相对较长，通常没有明显的原因。
(愉快的、阴郁的、易怒的、无神的、沮丧的、愉快的)

人际立场(Interpersonal stance)：在特定的互动中对另一个人所采取的感情立场，使得该情境中的人际交流充满色彩。
(遥远的、冷淡的、温暖的、支持的、轻蔑的、友好的)

态度(Attitude)：对物体或人的相对持久的、有感情色彩的信念、偏好和成见。
(喜欢、爱、恨、重视、渴望)

人格特质(Personality traits)：一个人典型的情绪丰满、稳定的人格气质和行为倾向。
(紧张的、焦虑的、鲁莽的、忧郁的、敌对的、嫉妒的)

图 20-1：感情(affective)状态的 Scherer 类型

我们可为每一种感情状态设计提取器。第四章已经介绍了**情感分析**(sentiment analysis)，即提取作者在文本中表达的积极或消极倾向。这与 Scherer 的类型学的**态度(attitudes)**提取相一致:从消费者对书籍或电影的评论、报纸社论或博客或推特上的公众情感等感情丰富的文本中找出人们喜欢什么或不喜欢什么。

情绪(emotion)和**语气(moods)**的检测在如下三方面是有用的：在与教学系统互动时，学生是否感到困惑、沉迷或确定；求助热线的呼叫者是否感到沮丧；某人的博客文章或推文是否表示泄气。例如，在小说中发现恐惧这样的情绪，可以帮助我们追踪恐惧的群体或情境，以及随着时间的推移恐惧如何变化。

在从人与人之间的对话中提取信息时，检测不同的**人际立场(interpersonal stances)**是很有用的。这里的目标是在采访或友好的对话中发现友好或尴尬的立场，例如总结会议或找到人们特别兴奋或参与的对话部分，对话热点，可以帮助会议总结。检测用户的个性——比如用户是否外向，或者他们对体验的开放程度——可以帮助改进会话施事者，如果它们符合用户的个性期望，似乎会工作得更好(Mairesse 和 Walker, 2008)。感情是生成和识别的重要因素,对感情进行综合对于不同领域的会话施事者来说很重要，包括儿童故事书或电脑游戏等教育辅导。

第四章介绍了朴素贝叶斯分类方法在文档情感分类中的应用。各种分类器已经成功地应用于许多这样的任务，使用训练集中的所有单词作为分类器的输入，然后该分类器确定文本的感情状态。

在本章中，我们着眼于替代模型，在该模型中，我们不是将每个单词都用作特征，而是仅关注某些单词，这些单词带有特别强烈的感情或情感暗示。我们称这些单词为感情词汇或情感词汇。这些词汇以关于语义的事实为前提：单词具有感情含义或**内涵(connotation)**。内涵在不同领域具有不同的含义，但是在这里，我们用它来表示单词含义的某些方面，这些方面与作者或读者的情绪，情感，观点或评价有关。内

涵词汇除了可以帮助确定文本的感情状态外，还可以用于其他类型的感情任务以及计算机社会科学分析。

在接下来的章节中，我们将介绍情绪的基本理论，说明情感词汇是情绪词汇的一种特殊情况，并提出一些有用的词汇。然后，我们调查了三种构建词汇表的方法:人工标签、半监督和监督。最后，我们转向其他类型的感情含义，如人格、立场和以实体为中心的感情，并引入内涵框架。

20.1. 定义情绪

情绪是最重要的感情类别之一，Scherer(2000)将**情绪(emotion)**定义为“对具有重要意义的外部或内部事件评价时的相对短暂的反应”。

检测情绪有可能改善许多语言处理任务。自动检测评论或客户反应中的情绪(愤怒、不满、信任)可以帮助企业认识到特定的问题领域或进展顺利的领域。情感识别可以帮助对话系统，如辅导系统检测到一个学生不高兴、无聊、犹豫、自信，等等。情绪可以在医学信息任务中发挥作用，如检测抑郁或自杀意图。检测小说中人物所表达的情绪，可能对理解社会在不同时期对不同社会群体的看法起着重要作用。

有两个广为接受的情绪理论家族。我们将为两种理论介绍词汇。在一个家族中，情绪被视为数量有限的固定原子单位，并且从中产生其他情感，通常被称为**基本情绪(basic emotions)**(Tomkins 1962，Plutchik 1962)。在这一系列理论中，最著名的也许是 Ekman 提出的 6 种情绪(例如，见 Ekman1999)，这是可能在所有文化中普遍存在的一组情绪：惊奇、幸福、愤怒、恐惧、厌恶、悲伤。另一个原子理论是“情绪轮”(Plutchik，1980 年)，由四个相反的偶对组成的 8 种基本情绪：喜悦-悲伤、愤怒-恐惧、信任-厌恶、期待-惊奇，以及从中得出的情绪，如图 20.2 所示：

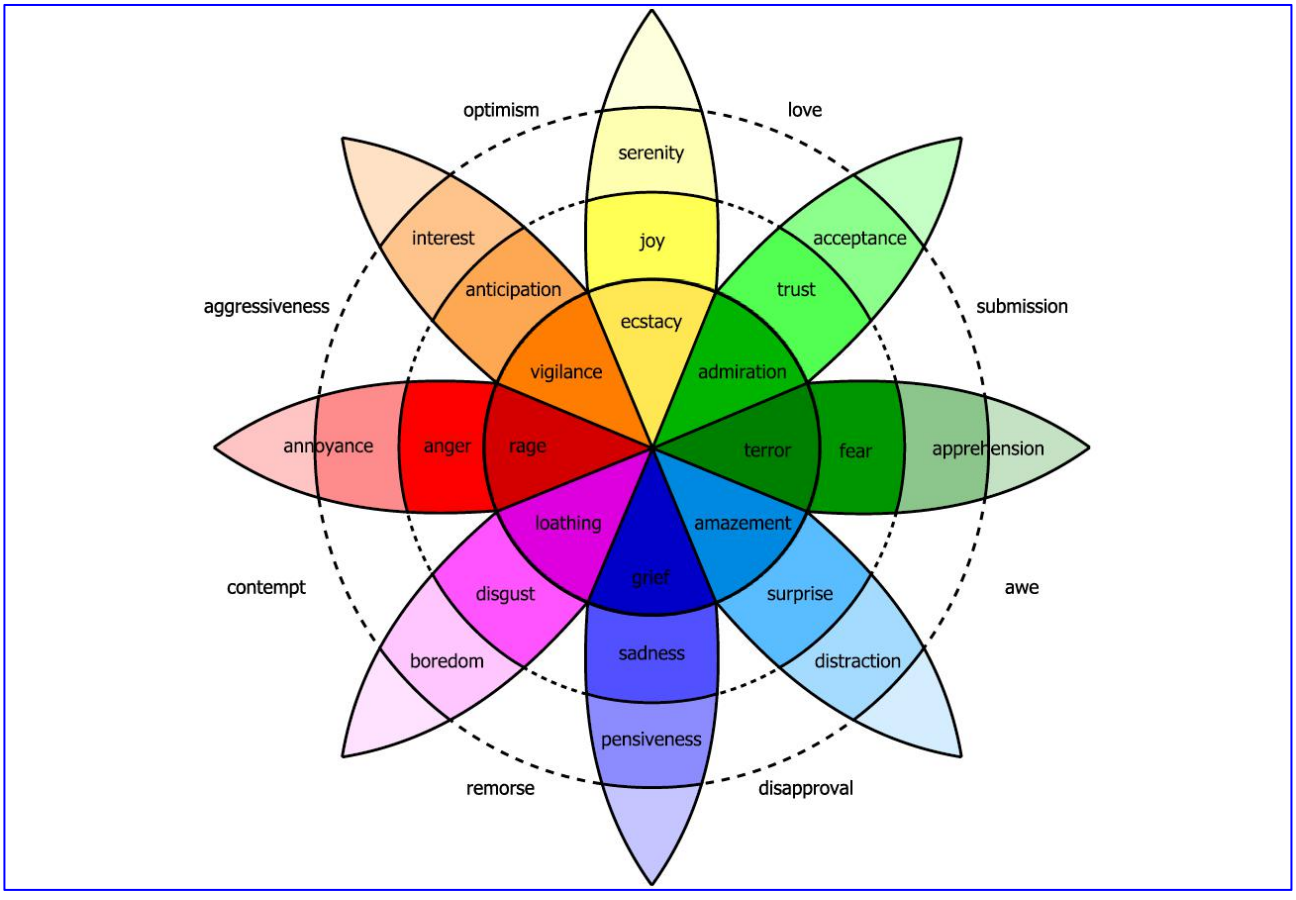


图 20-2：Plutchik 情绪轮-8 种基本情绪

第二类情绪理论将情感视为 2 个或 3 个维度的空间(Russell, 1980)。大多数模型包括**效价(valence)**和**唤起(arousal)**这两个维度，许多模型还增加了第三个维度，**支配(dominance)**。这些可以定义为:

效价(V)：刺激的愉悦感

唤起(A)：刺激所激起的情绪强度

支配(D)：刺激所施加的控制程度

情感可以被看作是第二种情绪的空间点的特殊情况。尤其是效价维度，衡量一个词有多令人愉快或不愉快，经常直接被用来衡量情感。

20.2. 可用的情感和感情词汇

各种各样的**感情(affect)**词汇已经被创造和发布。最基本的词汇根据语义变化的一个维度来标注单词，通常称为“**情感(sentiment)**”或“**效价(valence)**”。

在最简单的词汇中，此维度以二元方式表示，其中有一个用于正面的单词表和一个用于负面的单词表。最古老的是**普通调查员(General Inquirer，GI)**(Stone 等人，1966)，它是根据内容分析和对单词意义的认知心理学的早期研究而获得的(Osgood 等人，1957)。General Inquirer 有一个包含 1915 个正面词的词汇和一个包含 2291 个负面词的词汇(以及下面讨论的其他词汇)。MPQA 主观词汇(Wilson 等，2005)具有从先前词汇中抽取的 2718 个正面词和 4912 个负面词，以及一系列主观单词和短语的引导表(Riloff 和 Wiebe，2003 年)。词汇中的每个词条都经过手工标签以表达情感，还标签有可靠性(强烈主观或弱主观)。Hu 和 Liu(2004b)的极性(polarity)词汇提供了 2006 正面词和 4783 负面词，这些词来自产品评论，并使用 WordNet 的引导方法进行了标签，见图 20.3。

Positive	admire, amazing, assure, celebration, charm, eager, enthusiastic, excellent, fancy, fantastic, frolic, graceful, happy, joy, luck, majesty, mercy, nice, patience, perfect, proud, rejoice, relief, respect, satisfactorily, sensational, super, terrific, thank, vivid, wise, wonderful, zest
Negative	abominable, anger, anxious, bad, catastrophe, cheap, complaint, condescending, deceit, defective, disappointment, embarrass, fake, fear, filthy, fool, guilt, hate, idiot, inflict, lazy, miserable, mourn, nervous, objection, pest, plot, reject, scream, silly, terrible, unfriendly, vile, wicked

图 20-3：普通调查员、主观词汇、极性词汇

图注：General Inquirer(Stone 等人，1966)、MPQA 主观词汇(Wilson 等人，2005)和 Hu and Liu(2004b)的极性词汇中的一些情感一致的单词。

在所有三个感情维度上，为每个单词分配一个值的词汇要比这些情感词汇稍微普遍一些。NRC 的效价、唤起和支配(VAD)词汇(Mohammad，2018a)将效价，唤起和支配的分数分配给 20,000 个单词。图 20.4 中显示了一些示例。

Valence		Arousal		Dominance	
vacation	.840	enraged	.962	powerful	.991
delightful	.918	party	.840	authority	.935
whistle	.653	organized	.337	saxophone	.482
consolation	.408	effortless	.120	discouraged	.009
torture	.115	napping	.046	weak	.045

图 20-4：样本词在 Mohammad 情绪维度上的值

NRC 单词情绪关联词汇，也称为 **EmoLex**(Mohammad 和 Turney，2013)，使用了上面定义的 Plutchik(1980)的 8 种基本情绪。该词汇包括大约 14,000 个单词，包括以前词汇中的单词以及常见的名词、动词、副词和形容词。词汇中一些示例单词的值：

Word	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	positive	negative
reward	0	1	0	0	1	0	1	1	1	0
worry	0	1	0	1	0	1	0	0	0	1
tenderness	0	0	0	0	1	0	0	0	1	0
sweetheart	0	1	0	0	1	1	0	1	1	0
suddenly	0	0	0	0	0	0	1	0	0	0
thirst	0	1	0	0	0	1	1	0	0	0
garbage	0	0	1	0	0	0	0	0	0	1

对于一个较小的 5814 个单词的集合，NRC 情绪/感情强度词汇表(Mohammad, 2018b)包含了愤怒、恐惧、快乐和悲伤的关联的真实值分数;图 20.5 给出了示例。

Anger		Fear		Joy		Sadness	
outraged	0.964	horror	0.923	superb	0.864	sad	0.844
violence	0.742	anguish	0.703	cheered	0.773	guilt	0.750
coup	0.578	pestilence	0.625	rainbow	0.531	unkind	0.547
oust	0.484	stressed	0.531	gesture	0.387	difficulties	0.421
suspicious	0.484	failing	0.531	warms	0.391	beggar	0.422
nurture	0.059	confident	0.094	hardship	.031	sing	0.017

图 20-5：对愤怒、恐惧、快乐和悲伤词汇的情绪强度样本

语言查询和单词计数(Linguistic Inquiry and Word Count，LIWC)，是一个被广泛使用的包含超过 2300 个单词的 73 个词汇集(Pennebaker 等人，2007)，旨在捕捉与社会心理任务相关的词汇意义方面。除了与情感相关的词汇，如消极情绪(坏、奇怪、恨、问题、强硬)和积极情绪(爱、美好、甜蜜)，LIWC 还包括了愤怒、悲伤、认知机制、知觉、试探和抑制等类别的词汇，如图 20.6 所示。

Positive Emotion	Negative Emotion	Insight	Inhibition	Family	Negate
appreciat*	anger*	aware*	avoid*	brother*	aren't
comfort*	bore*	believe	careful*	cousin*	cannot
great	cry	decid*	hesitat*	daughter*	didn't
happy	despair*	feel	limit*	family	neither
interest	fail*	figur*	oppos*	father*	never
joy*	fear	know	prevent*	grandf*	no
perfect*	griev*	knew	reluctan*	grandm*	nobod*
please*	hate*	means	safe*	husband	none
safe*	panic*	notice*	stop	mom	nor
terrific	suffers	recogni*	stubborn*	mother	nothing
value	terrify	sense	wait	niece*	nowhere
wow*	violent*	think	wary	wife	without

图 20-6：LIWC 中 73 个词汇类别中的 5 个样本

图注：其中*表示前面的字母是一个单词前缀，所有带有该前缀的单词都包含在类别中(Pennebaker 等人，2007)。

还有其他各种各样的手工构建的感情词汇。一般查询包括诸如强与弱、主动与被动、夸大与低估等维度的额外词汇，以及诸如快乐、痛苦、美德、邪恶、动机和认知取向等类别的词汇。

对各种任务来说，另一个有用的特征是**具体(concrete)**单词(如 banana 或 bathrobe)与**抽象(abstract)**单词(如 belief 和 although)之间的区别。Brysbaert 等人(2014)的词汇使用众包的方法对 4 万个单词的具象性进行了从 1 到 5 的评分，例如给 banana、bathrobe 和 bagel 打 5 分，belief 打 1.19 分, although 打 1.07 分，以及中间的单词，诸如 brisk，打 2.5 分。

20.3. 通过人工标签创建感情词汇

最早用来建立感情词汇的方法是让人给每个词贴上标签，这一方法至今仍在使用。现在最普遍的做法是通过**众包(crowdsourcing)**来完成:把任务分成小块，然后分发给大量的评注者。让我们来看看这两个来自大众的情绪词汇在方法论上的选择。

NRC **情绪词典(EmoLex)** (Mohammad 和 Turney, 2013)，用两个步骤标记情绪。为了确保评注者判断的是这个词的正确意思，他们首先回答了一个同义词的多项选择题，这个问题引发了这个词的正确意义(不需要评注者阅读可能会造成混淆的意义定义)。这些是使用以下两类中心词自动创建的：与 Macquarie 字典中所讨论的意义的同义词库类别相关的中心词；3 个随机干扰项类别的中心词。一个例子：

Which word is closest in meaning (most related) to startle?

• automobile • shake • honesty • entertain

对于每一个词(例如 startle)，评注者被要求对这个词与 8 种情绪(joy, fear, anger 等)之间的联系进行评级。这些关联在 not, weakly, moderately, 和 strongly 的尺度(scale)上被评级。除去异常值评级(Outlier ratings)，然后为每个术语分配大多数评注者选择的类别，并通过选择更强的强度来打破纽带关系，然后将 4 个级别映射到每个单词的二元标签中(否和弱映射到 0，中度和强度对应于 1)。

NRC 的 VAD 词汇(Mohammad, 2018a)是通过从先前的词汇中选择单词和情绪符号，并用**最好-最差缩放(best-worst scaling)**的群体外包对它们进行评注而构建的(Louviere 等人 2015，Kiritchenko 和 Mohammad 2017)。在最好-最差缩放中，给评注者 N 项(通常是 4 项)，并被问及就某些属性而言，哪项是最好的(最高的)，哪项是最差的(最低的)。用来描述尺度(scales)末端的一组词取自先前的文献。例如，对于效价，评级者(raters)被要求:

Q1. Which of the four words below is associated with the
MOST happiness / pleasure / positiveness / satisfaction / contentedness / hopefulness
OR LEAST unhappiness / annoyance / negativeness / dissatisfaction / melancholy / despair?
(Four words listed as options.)

Q2. Which of the four words below is associated with the
LEAST happiness / pleasure / positiveness / satisfaction / contentedness / hopefulness
OR MOST unhappiness / annoyance / negativeness / dissatisfaction / melancholy / despair?
(Four words listed as options.)

词汇表中每个单词的得分是该项被选为最好(最高的 V/A/D)的次数的比例减去该项被选为最差(最低的 V/A/D)的次数的比例。评注之间的一致性通过**对半可靠性(split-half reliability)**进行评估：将语料库对半分开并计算两半之间评注之间的相关性。

20.4. 感情词汇的半监督归纳

学习情感词汇的另一种常见方法是从一组定义语义轴两极的种子词(如 good 或 bad)开始，然后通过与这两种种子集的相似性来标签每个单词 w。在这里我们总结了两族基于种子的半监督词汇归纳算法：基于轴的和基于图的算法。

20.4.1. 语义轴方法

最著名的词汇归纳方法之一，Turney 和 Littman(2003)算法，是给定种子词，例如 good 或 bad，然后为每个单词 w 贴标签，来衡量它既是如何与 good 相似的，又是如何与 bad 不同的。在这里，我们描述了基于计算语义轴的算法的略微扩展，这归功于 An 等人(2018)。

第一步，我们手工选择种子词。有两种方法可以解决单词在不同上下文中的感情不同的事实:(1)从一个大型种子词汇开始，依靠归纳算法对其进行微调到领域(domain)，或者(2)为不同的体裁选择不同的种子词汇。Hellrich 等人(2019)认为，要对跨越不同历史时期的感情建模，从一个大型的现代感情字典开始比调整为跨时间稳定的小种子集更好。作为第二种方法的一个例子，Hamilton 等人(2016a)定义一组用于一般情感分析的种子词，一组用于 Twitter 的种子词，还有一组用于金融文本的种子词:

Domain	Positive seeds	Negative seeds
General	good, lovely, excellent, fortunate, pleasant, delightful, perfect, loved, love, happy	bad, horrible, poor, unfortunate, unpleasant, disgusting, evil, hated, hate, unhappy
Twitter	love, loved, loves, awesome, nice, amazing, best, fantastic, correct, happy	hate, hated, hates, terrible, nasty, awful, worst, horrible, wrong, sad
Finance	successful, excellent, profit, beneficial, improving, improved, success, gains,positive	negligent, loss, volatile, wrong, losses, damages,bad,litigation,failure,down,negative

第二步，我们计算每个极点词的嵌入。这些嵌入可以是现成的 word2vec 嵌入，或者可以在特定语料库上直接计算(例如，如果目标是金融词汇，则使用金融语料库)，或者我们可以微调现成的嵌入到语料库。如果我们有一个非常特定的文本体裁，但没有足够的数据来训练良好的嵌入，微调就特别重要。在微调方

面，我们从现成的嵌入式开始，如 word2vec，并继续在小型目标语料库上训练它们。

一旦我们有了每个极点词的嵌入，我们通过取每个种子词嵌入的质心(centroid)来创建一个代表每个极点的嵌入;回想一下，质心是均值的多维版本。给定一组正面种子词 $S^+ = \{E(w_1^+w+1)，E(w_2^+)，\cdots，E(w_n^+)\}$，负面种子词 $S^- = \{E(w_1^-)，E(w_2^-)，\cdots，E(w_m^-)\}$，极点质心为:

$$\mathbf{V}^+ = \frac{1}{n}\sum_{1}^{n} E(w_i^+)$$

$$\mathbf{V}^- = \frac{1}{m}\sum_{1}^{m} E(w_i^-) \tag{20.1}$$

由极点定义的语义轴的计算只需减去两个向量:

$$V_{axis} = V^+ - V^- \tag{20.2}$$

V_{axis}，即语义轴，是正向情感的矢量。最后，我们通过余弦相似度计算出正向情感方向与 w 嵌入方向的夹角。更高的余弦意味着 w 更倾向于 S^+而不是 S^-。

$$\begin{aligned} \text{score}(w) &= \left(\cos(E(w), \mathbf{V}_{\text{axis}}\right) \\ &= \frac{E(w)\cdot \mathbf{V}_{\text{axis}}}{\|E(w)\|\|\mathbf{V}_{\text{axis}}\|} \end{aligned} \tag{20.3}$$

如果一本带有情感评分的单词字典就足够了，那么我们就完成了!或者，如果我们需要将词汇分为正面词汇和负面词汇，那么我们可以使用一个阈值或其他方法来给予我们离散词汇。

20.4.2. 标签传播

另一种方法是通过在图上传播情感标签来定义词汇，这是 Hatzivassiloglou 和 McKeown(1997)早期工作中提出的想法。我们将描述 Hamilton 等人(2016a)的简单**情感传播(SentProp)**算法，该算法有四个步骤:

1. 定义一个图:给定词的嵌入，通过连接每个词与其 k 个最近的邻居(根据余弦相似度)建立一个加权词图。设单词 w_i 和 w_j 之间的边的权值为:

$$\mathbf{E}_{i,j} = \arccos\left(-\frac{\mathbf{w_i}^\top \mathbf{w_j}}{\|\mathbf{w_i}\|\|\mathbf{w_j}\|}\right). \tag{20.4}$$

2.定义一个种子集:选择正面种子词和负面种子词。

3.从种子集中传播极性:现在我们在这个图上执行随机漫步，从种子集中开始。在随机漫游中，我们从一个节点开始，然后选择与边概率成比例的概率移动到一个节点。一个单词的极性分数与从种子集到该单词的随机漫步概率成正比(图 20.7)。

4.创建单词分数:我们从正面和负面种子集走来，得到结果是正($\text{rawscore}^+(w_i)$)和负($\text{rawscoree}^-(w_i)$)原始标签分数。然后我们将这些值组合成正极性分数如下:

$$\text{score}^+(w_i) = \frac{\text{rawscore}^+(w_i)}{\text{rawscore}^+(w_i) + \text{rawscore}^-(w_i)} \tag{20.5}$$

将分数标准化，使其在一个语料库中均值和单位方差为零通常是有帮助的。

5. 给每个分数赋置信度:因为情绪分数受到种子集的影响，我们想知道如果使用不同的种子集，单词的分数会发生多大的变化。我们可以使用自举抽样来获得置信区域，通过计算在正面种子集和负面种子集的随机子集上的传播次数 B(例如使用 B=50，每次从 10 个种子词中选择 7 个)。自举取样极性分数的标准偏差给出了一个置信度度量。

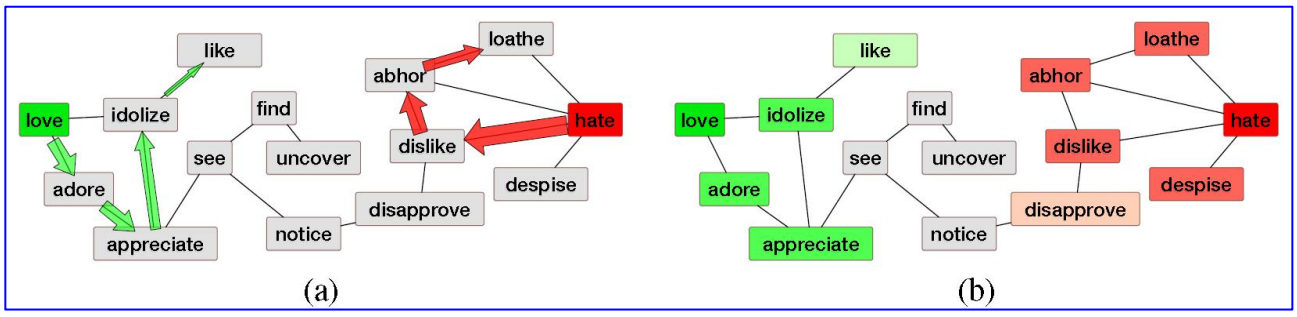


图 20-7：SentProp 算法的直观感受

图注：图(a)从种子词中随机漫步。图(b)根据随机行走的次数分配极性分数(在这里用绿色或红色表示)。

20.4.3. 其它方法

半监督算法的核心是与种子词相似度的度量。Turney 和 Littman(2003)和 Hamilton 等人(2016a)在上述方法中使用了嵌入余弦作为距离度量:如果单词的嵌入具有带有正面种子的高余弦和带有负面种子的低余弦，则基本上将其标记为正余弦。其他方法除了嵌入余弦外，还选择了其他类型的距离度量。

例如，Hatzivassiloglou 和 McKeown(1997)算法使用了语法线索;如果两个形容词经常由 and 连接，很少由 but 连接，那么它们被认为是相似的。这是基于一种直觉，即由单词连接而成的形容词往往具有相同的极性;正面形容词通常与正面并列，负面形容词通常与负面并列:

fair and legitimate, corrupt and brutal (公平而合法，腐败且残忍)

但与负面连用的正面形容词较少:

*fair and brutal, *corrupt and legitimate

相比之下，由 but 连接的形容词可能具有相反的极性:

fair but brutal

相反极性的另一个线索来自形态否定(un-，im-，-less)。词根相同但在形态上否定的不同的形容词(adequate/ adequate, thoughtful/thoughtless)往往具有相反的极性。另一种寻找与种子词具有相似极性的词的方法是使用 WordNet 这样的同义词库(Kim 和 Hovy 2004, Hu 和 Liu 2004b)。一个单词的同义词可能有相同的极性，而一个单词的反义词可能有相反的极性。在构建种子词汇表之后，每个词汇表将按如下方式更新，可能是迭代的。

Lex$^+$：增加肯定词的同义词(well)和否定词的反义词(如 fine)

Lex$^-$：增加否定词的同义词(awful)和肯定词的反义词(如 evil)

该算法的一个扩展将极性分配到 WordNet 的含义，称为 **SentiWordNet**(Baccianella 等人，2010)。图 20.8 显示了一些例子。

Synset		Pos	Neg	Obj
good#6	'agreeable or pleasing'	1	0	0
respectable#2 honorable#4 good#4 estimable#2	'deserving of esteem'	0.75	0	0.25
estimable#3 computable#1	'may be computed or estimated'	0	0	1
sting#1 burn#4 bite#2	'cause a sharp or stinging pain'	0	0.875	0.125
acute#6	'of critical importance and consequence'	0.625	0.125	0.250
acute#4	'of an angle; less than 90 degrees'	0	0	1
acute#1	'having or experiencing a rapid onset and short but severe course'	0	0.5	0.5

图 20-8：来自 SentiWordNet 3.0 的示例

图注：注意同形词的区别:estimable#3 是纯客观的，而 estimable#2 是正面的;acute 可以是正面(acute#6)、负面(acute#1)或中性(acute#4)。

在该算法中，极性被分配给整个同义词库而不是单词。从与 7 个正面词相关的所有同义词库构建一个正面词库，从与 7 个负面词相关的同义词库构建一个负面词库。然后根据这些数据训练分类器，以获取 WordNet 的注释，并确定所定义的含义是正面的，负面的还是中立的。进一步的步骤(涉及随机漫游算法)

为每个 WordNet 同义词库的正面、负面和中立程度分配分数。

总之，半监督算法，针对维度的两极，使用人类定义的种子词集，并使用类似度量，如嵌入余弦、协调、形态学或词典结构，通过它们与正面种子的相似程度和与负面种子的不同程度来为单词评分。

20.5. 单词情感的监督学习

半监督方法只需要最少的人工监督(以种子组的形式)。但有时世界上存在着一种可以利用的监督信号。其中一个信号就是与在线评论相关的分数。

web 包含了大量关于餐馆、电影、书籍或其他产品的在线评论，每个评论都有评论的文本以及相关的评论评分:评分范围从 1 到 5 星，或 1 到 10 星。图 20.9 显示了从餐厅、书籍和电影评论中提取的样本。

我们可以将这一评分作为监督：正面词汇更有可能出现在 5 星评论中；负面词汇在 1 星评论中。这种监督方式允许我们给一个单词分配一个更复杂的极性（而不仅仅是一种二元极性）：它在星(或其他分数)上的分布。因此，在一个 10 星系统中，可以用 10 元组来表示每个单词的情绪，每个数字代表单词与极性水平的关联。这种关联可以是原始计数，或可能的 P(w|c)，或计数的其他函数，对于每个 c 类从 1 到 10。

例如,在 IMDb 数据集之中，我们可以计算单词 disappoint(ed/ing)出现 1 星评价的似然度：单词 disappoint(ed/ing)1 星评价出现的次数(8,557)，除以 1 星评价出现的单词总数(25,395,214)，因此 P(w=disappointing | c=1)的 IMDb 估计为 8557/25395214=0.0003。对这一权重(归一化似然度)稍加修改，就可以用作一种启发性的可视化(Potts, 2011)[1]:

$$P(w|c) = \frac{count(w,c)}{\sum_{w \in C} count(w,c)}$$

$$PottsScore(w) = \frac{P(w|c)}{\sum_{c} P(w|c)} \tag{20.6}$$

将 0.0003 的 IMDb 估计值 P(disappointing| 1)，除以所有类别上的似然度 P(w | c)总和，给出的 Potts 得分为 0.10。因此，单词 disappointing 与向量[.10，.12，.14，.14，.13，.11，.08，.06，.06，.05]相关联。**Potts 图**(Potts，2011 年)是这些单词得分的可视化结果，代表了 Potts 图的单词先验情感，作为评级类别的分布。

	Movie review excerpts (IMDb)
10	A great movie. This film is just a wonderful experience. It's surreal, zany, witty and slapstick all at the same time. And terrific performances too.
1	This was probably the worst movie I have ever seen. The story went nowhere even though they could have done some interesting stuff with it.
	Restaurant review excerpts (Yelp)
5	The service was impeccable. The food was cooked and seasoned perfectly... The watermelon was perfectly square ... The grilled octopus was ... mouthwatering...
2	...it took a while to get our waters, we got our entree before our starter, and we never received silverware or napkins until we requested them...
	Book review excerpts (GoodReads)
1	I am going to try and stop being deceived by eye-catching titles. I so wanted to like this book and was so disappointed by it.
5	This book is hilarious. I would recommend it to anyone looking for a satirical read with a romantic twist and a narrator that keeps butting in
	Product review excerpts (Amazon)
5	The lid on this blender though is probably what I like the best about it... enables you to pour into something without even taking the lid off! ... the perfect pitcher! ... works fantastic.
1	I hate this blender... It is nearly impossible to get frozen fruit and ice to turn into a smoothie... You have to add a TON of liquid. I also wish it had a spout ...

图 20-9：摘自不同评论网站的一些评论

图注：除了 IMDb 的评分为 1 - 10 星，其他评分均为 1 - 5 星。

[1] Potts 表明，如果我们做出错误但简化的假设，即所有类别 c 具有相同的概率，则归一化似然度是后验 P(c|w)的估计。

图 20.10 显示了 3 个正标量和 3 个负标量形容词的 Potts 图。注意，强正标量的曲线具有字母 J 的形状，而强负标量的曲线看起来像是反向 J。相反，弱正标量和负标量具有驼峰形状，最大值低于均值(弱负值词诸如 disappointing)或高于平均数(弱正值词诸如 good)。这些形状提供了感情含义的启发性类型。

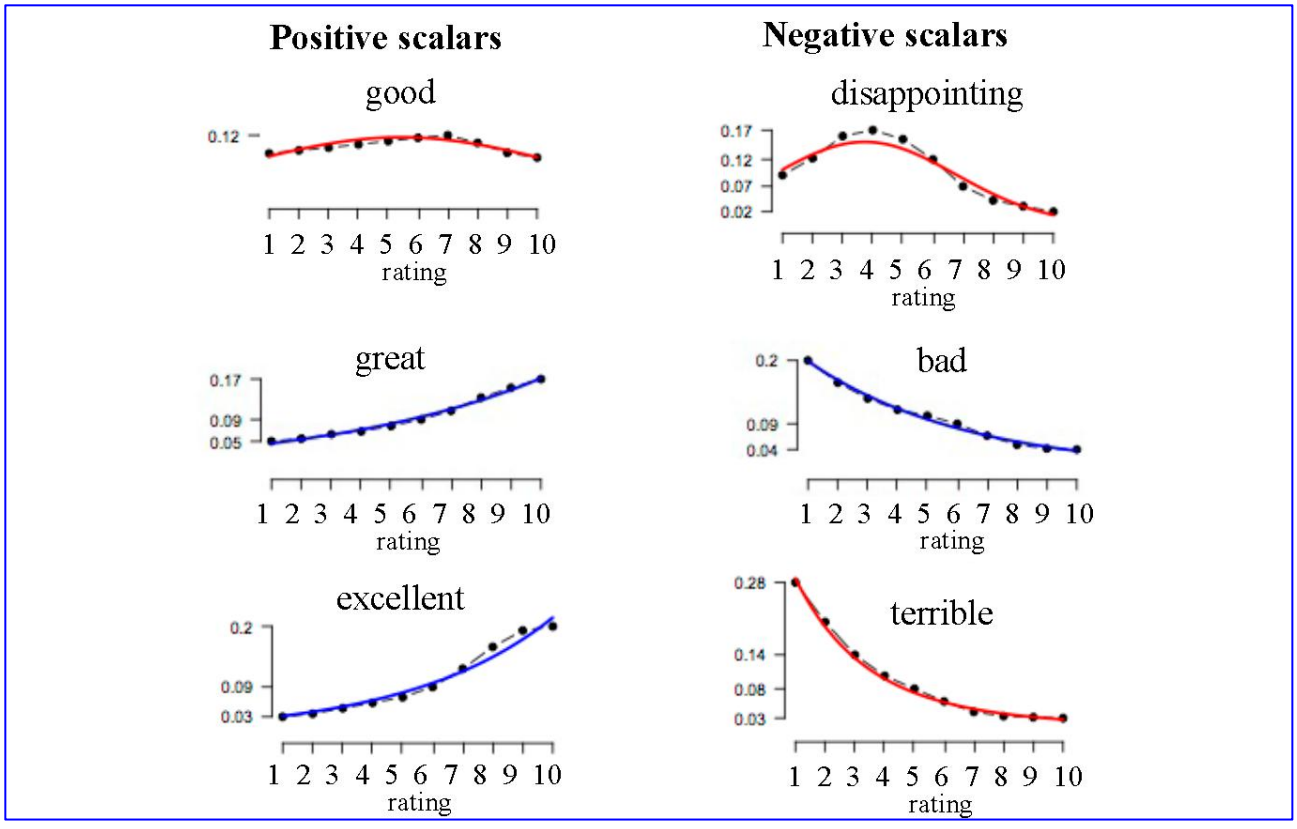


图 20-10：正面、负面标量形容词的 Potts 图

图注：显示了强正面、负面形容词的 J 形和反 J 形，以及较弱极化形容词的峰形(Potts, 2011)。

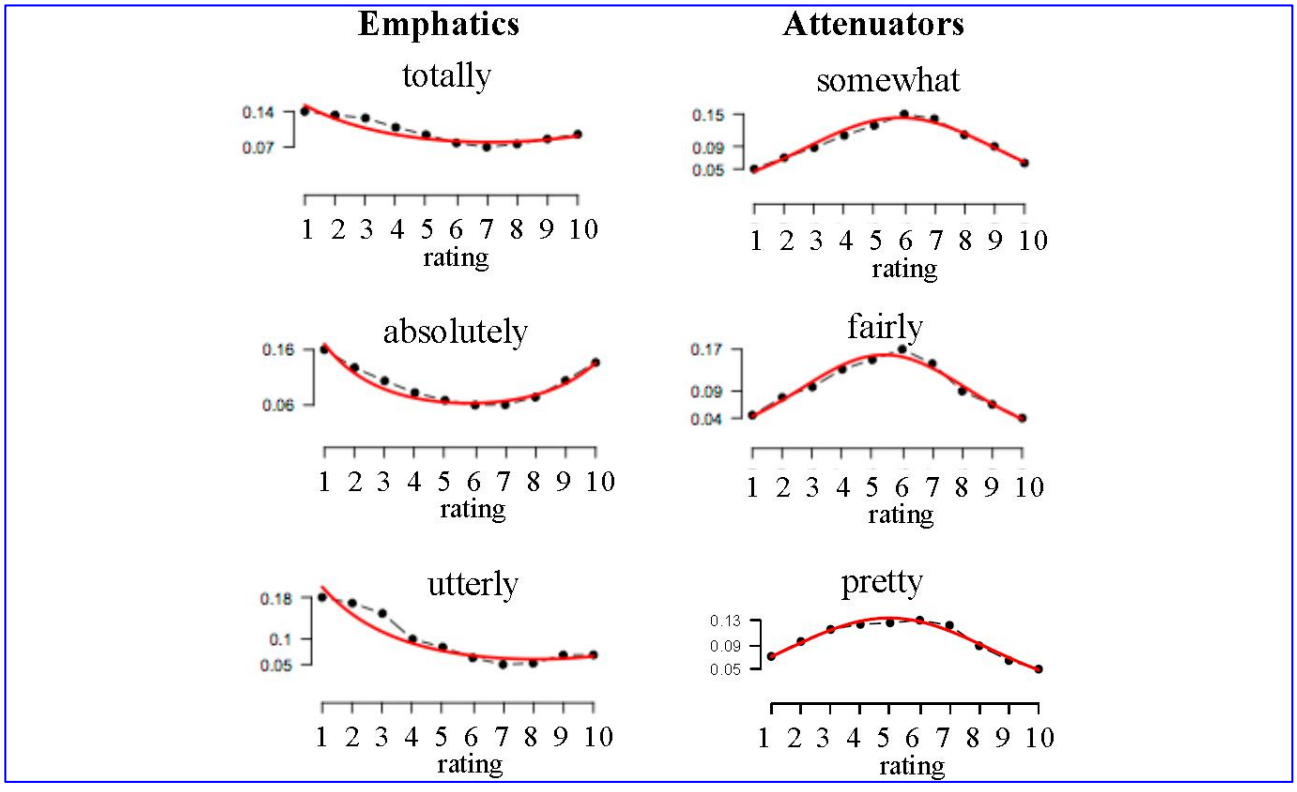


图 20-11：强调和削弱副词的 Potts 图

图 20.11 显示了用于强调和减弱副词的 Potts 图。请注意，强调倾向于具有 J 形(最有可能出现在最积极的评论中)或 U 形(最有可能出现在强烈的正面和负面情绪中)。衰减器都具有驼峰形状，强调尺度的中部而对两个极端都轻描淡写。这些图既可以用作词汇情感的类型，也可以在情感构成模型中发挥作用。

除了像后验 P(c|w)、似然 P(w|c)或归一化似然(公式 20.6)这样的函数外，还使用了许多其他与情感标签一起出现的单词计数的函数。我们将第 20.8 节介绍其中的一些思想，包括像在公式 20.14 中归一化每个作家的计数。

20.5.1. 对数优势比率信息的狄利克雷先验

我们经常想要做的一件事是区分更可能在一类文本中使用的单词，而不是在另一类文本中使用的单词。例如，我们可能想知道与 1 星评价最相关的词和与 5 星评价最相关的词。这些差异可能不仅仅与情感有关。我们可能想要找到民主党议员比共和党议员更经常使用的词语，或者在高档餐厅的菜单上使用的词语比廉价餐厅的多。

给定两类文档，为了找到与某一类比另一类更相关的单词，我们可以测量频率上的差异(单词 w 在 A 类还是 B 类中更频繁?)或者，我们可以计算频率之比，或者计算对数优势比率(两个词之间优势之比的对数)，而不是频率之差。然后，我们可以根据我们选择的任何关联度量对单词进行排序，从类别 A 中过多表示的单词到类别 B 中过多表示的单词。

简单的对数似然或对数优比方法的问题是，它们对非常罕见的单词或非常频繁的单词不起作用;对于频繁使用的单词，所有的差异都显得很大，而对于很少使用的单词，没有什么差异显得很大。

在本节中,我们详解一个解决这个问题的细节:“对数优势比率信息的**狄利克雷(Dirichlet)**先验”方法(Monroe 等人，2008)，这是一个特别有用的方法，用于寻找某些单词，这些单词与其它单词相比，在一类特定的文本中统计上的代表性过高。它是基于使用另一个大型语料库来获得我们对每个单词出现频率的预估。

让我们从目标开始：假设我们想知道 horrible 一词在语料库 i 还是在语料库 j 中出现的更多。我们可以计算**对数似然比率(log likelihood ratio)**，使用 $f^i(w)$表示语料库 i 中单词 w 的频率，而 n^i表示语料库 i 中单词的总数：

$$\begin{aligned}\text{llr}(horrible) &= \log\frac{P^i(horrible)}{P^j(horrible)}\\ &= \log P^i(horrible) - \log P^j(horrible)\\ &= \log\frac{\text{f}^i(horrible)}{n^i} - \log\frac{\text{f}^j(horrible)}{n^j}\end{aligned} \tag{20.7}$$

相反，让我们计算**对数优势比率(log odds ratio)**：在 i 或 j 中，horrible 的是否有更高的优势：

$$\begin{aligned}\text{lor}(horrible) &= \log\left(\frac{P^i(horrible)}{1-P^i(horrible)}\right) - \log\left(\frac{P^j(horrible)}{1-P^j(horrible)}\right)\\ &= \log\left(\frac{\frac{\text{f}^i(horrible)}{n^i}}{1-\frac{\text{f}^i(horrible)}{n^i}}\right) - \log\left(\frac{\frac{\text{f}^j(horrible)}{n^j}}{1-\frac{\text{f}^j(horrible)}{n^j}}\right)\\ &= \log\left(\frac{\text{f}^i(horrible)}{n^i-\text{f}^i(horrible)}\right) - \log\left(\frac{\text{f}^j(horrible)}{n^j-\text{f}^j(horrible)}\right)\end{aligned} \tag{20.8}$$

Dirichlet 的直觉是使用一个大型背景语料库来获得我们对每个单词 w 的期望频率的先验估计。我们将通过将该语料库中的计数添加到分子和分母中来非常简单地进行此操作，从而实质上将计数减少到该计数

之前。这就像在询问：给定我们的期望，即在一个良好估计的大背景语料库中给定它们的频率，在此条件下，i 和 j 之间的差异有多大？

该方法通过 w 的先验修正的**对数优势比率**$\delta_w^{(i-j)}$来估计两个语料库 i 和 j 中单词 w 的频率差异，结果为:

$$\delta_w^{(i-j)} = \log\left(\frac{f_w^i + \alpha_w}{n^i + \alpha_0 - (f_w^i + \alpha_w)}\right) - \log\left(\frac{f_w^j + \alpha_w}{n^j + \alpha_0 - (f_w^j + \alpha_w)}\right) \tag{20.9}$$

(其中 n^i 是语料库 i 的大小，n^j 是语料库 j 的大小，f_w^i 是语料库 i 中单词 w 的计数，f_w^j 是语料库 j 中单词 w 的计数，α_0 是背景语料库的大小，而 α_w 是背景语料库中单词 w 的计数。)

此外，Monroe 等人(2008)利用**对数优势比率**的方差估计：

$$\sigma^2\left(\hat{\delta}_w^{(i-j)}\right) \approx \frac{1}{f_w^i + \alpha_w} + \frac{1}{f_w^j + \alpha_w} \tag{20.10}$$

一个单词的最终统计数据是它的**对数优势比率**的 z 分数:

$$\frac{\hat{\delta}_w^{(i-j)}}{\sqrt{\sigma^2\left(\hat{\delta}_w^{(i-j)}\right)}} \tag{20.11}$$

Monroe 等人(2008)方法因此以两种方式修改了常用的**对数优势比率**：使用**对数优势比率**的 z 分数(控制单词频率中的方差量)，并使用背景语料库中的计数来提供事先单词的计数。

图 20.12 显示了应用于 Yelp 餐厅评论数据集的方法，该方法将 1 星级评论中使用的单词与 5 星级评论中使用的单词进行比较(Jurafsky 等人，2014)。最大的区别在于明显的情感词，其中 1 星级评论使用诸如 worse, bad, awful 的负面情感词，而 5 星级评论使用诸如 great, best, amazing 的正面情感词。但是，还有其他一些启发性的差异。1 星评价使用逻辑否定(no,not)，而 5 星评价则使用强调并强调普遍性(very, highly, every, always)。1 星评价使用第一人称复数(we,us,our)，而 5 星评价使用第二人称复数。1 星级评论谈论人们(manager,waiter,customer)，而 5 星级评论谈论甜点和昂贵餐厅的属性，例如课程和氛围。参见 Jurafsky 等人(2014)了解更多详情。

Class	Words in 1-star reviews	Class	Words in 5-star reviews
Negative	*worst, rude, terrible, horrible, bad, awful, disgusting, bland, tasteless, gross, mediocre, overpriced, worse, poor*	**Positive**	*great, best, love(d), delicious, amazing, favorite, perfect, excellent, awesome, friendly, fantastic, fresh, wonderful, incredible, sweet, yum(my)*
Negation	*no, not*	**Emphatics/ universals**	*very, highly, perfectly, definitely, absolutely, everything, every, always*
1Pl pro	*we, us, our*	**2 pro**	*you*
3 pro	*she, he, her, him*	**Articles**	*a, the*
Past verb	*was, were, asked, told, said, did, charged, waited, left, took*	**Advice**	*try, recommend*
Sequencers	*after, then*	**Conjunct**	*also, as, well, with, and*
Nouns	*manager, waitress, waiter, customer, customers, attitude, waste, poisoning, money, bill, minutes*	**Nouns**	*atmosphere, dessert, chocolate, wine, course, menu*
Irrealis modals	*would, should*	**Auxiliaries**	*is/'s, can, 've, are*
Comp	*to, that*	**Prep, other**	*in, of, die, city, mouth*

图 20-12：与 1 星和 5 星餐厅评论相关的前 50 个单词

图注：使用 Monroe 等人(2008)的方法(Jurafsky 等人，2014)，在一个有 90 万条评论的 Yelp 数据集中，与 1 星和 5 星餐厅评论相关的前 50 个单词。

20.6. 使用词汇进行情感识别

第四章介绍了用于情感分析的朴素贝叶斯算法。到目前为止，我们在整个章节中所关注的词汇可以用于许多方法来改进情感检测。

在最简单的情况下，当我们没有足够的训练数据来构建监督情感分析器时，可以使用词汇；让人给每个文档分配情感来训练受监督的分类器通常是很昂贵的。

在这种情况下，可以在基于规则的分类算法中使用词汇。最简单的方法就是使用肯定词与否定词的比例:如果一份文档中的肯定词比否定词多(使用词典来确定文档中每个词的极性)，它就被归类为肯定词。通常使用λ阈值，其中只有当比值大于λ时，文件才被归为正值。如果情感词典中包含每个单词的正权值和负权值θ_w^+和θ_w^-，也可以使用。这里有一个简单的情感算法:

$$f^+ = \sum_{w \text{ s.t. } w \in positive lexicon} \theta_w^+ count(w)$$

$$f^- = \sum_{w \text{ s.t. } w \in negative lexicon} \theta_w^- count(w)$$

$$sentiment = \begin{cases} + & \text{if } \frac{f^+}{f^-} > \lambda \\ - & \text{if } \frac{f^-}{f^+} > \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{20.12}$$

如果有监督训练数据可用，则这些计数从情感词汇计算出来，有时以各种方式加权或归一化，也可以作为特征在分类器中与其他词汇或非词汇特征一起使用。我们将在第 20.8 节回到这些算法。

20.7. 其他任务：人格

许多其他种类的情感含义可以从文本和语音中提取出来。例如，从一个人的语言中检测出他的**人格(personality)**对于对话系统来说很有用(用户倾向于选择与他们的人格相匹配的施事者)，并且在诸如理解人格如何与其他类型的行为相关联的计算社会科学问题中也很有用。

许多关于人类性格的理论都是基于少数几个维度，例如“五大”维度的不同版本(Digman, 1990):

外向 vs.内向： 善于交际、自信、幽默 vs.冷漠、矜持、害羞
情绪稳定 vs.神经质： 冷静、不情绪化 vs.缺乏安全感、焦虑
随和 vs.不随和： 友好、合作 vs.敌对、吹毛求疵
认真尽责 vs.不认真尽责： 自律、有条理 vs.无效、粗心
经验开放性： 智慧的、富有洞察力的 vs.肤浅的、缺乏想象力的

通过让作者参加标准的人格测试，一些文本和语音语料库已经被标记为其作者的人格。Pennebaker 和 King(1999)的文章语料库包括 2479 篇来自心理学专业学生的文章(190 万字)，他们被要求在 20 分钟内“写下你想到的任何东西”。梅尔等人(2006)的耳朵(电子激活录音机)文集是通过让志愿者全天佩戴录音机创建的，录音机随机记录下全天的简短对话片段，然后将其转录。Facebook 的语料库(Schwartz 等人，2013)包括 3.09 亿字的 Facebook 帖子，来自 7.5 万名志愿者。例如，这是 Pennebaker 和 King(1999)的一篇文章的样本，作者属于神经质/情绪稳定量表的神经质一端：

我的一个朋友闯了进来，我从座位上跳了起来。这是疯狂的。我应该告诉他不要再那样做了。其实我并不是那么挑剔。但有些事让我很烦。那些会让我烦恼的事情实际上也会让任何正常人烦恼，所以我知道我不是一个怪物。

而在量表另一端是情感稳定的人：

我应该在这项运动中胜出，因为我知道如何比我认识的任何人更努力地锻炼我的身体，无论什么测试，我总是比其他人更努力地锻炼我的身体。我想成为最好的，无论什么运动或项目。我也应该擅长这个，因为我喜欢骑自行车。

另一种感情含义是 Scherer(2000)所说的**人际立场(interpersonal stance)**，即“在特定的互动中对他人采取的感情立场使人际交往充满色彩”。提取这种含义意味着自动标签参与者是否友好、支持和疏远。例如，Ranganath 等人(2013)研究了一个快速约会的语料库，其中参与者戴着麦克风进行了一系列 4 分钟的浪漫约会。每个参与者都以自己的轻浮、友好、尴尬或自信为自己打标签。Ranganath 等人 (2013)然后结合使用词汇和其他功能从文本中检测出这些人际关系。

20.8. 感情识别

Scherer(2000)所描述的情绪、人格、人际立场和其他感情含义的检测，可以通过概括上述检测情感的算法来完成。

最常见的算法涉及监督分类：为要检测的感情含义标签训练集，并使用从训练集中提取的特征构建分类器。与情感分析一样，如果训练集足够大，并且测试集与训练集足够相似，则只需将所有单词或所有 bigrams 作为功能强大的分类器(如 SVM 或逻辑回归)中的特征，如图 4.2 所示，是一种出色的算法，其性能难以匹敌。因此，我们可以将文本样本的感情含义分类视为简单的文档分类。

然而，对于非常大的数据集，一些修改往往是必要的。例如，Schwartz 等人(2013)利用 Facebook 上 7 亿字的帖子研究了人格、性别和年龄，但只使用了长度为 1-3 的 n-grams 的子集。只有至少 1%的受试者使用的词和短语被纳入为特征，只有当它们具有足够高的 PMI (PMI 大于 2*length, length 为单词的个数)时才保留 2-grams 和 3-grams:

$$\text{pmi}(phrase) = \log \frac{p(phrase)}{\prod_{w \in phrase} p(w)} \tag{20.13}$$

可以为特征使用各种权重，包括训练集中的原始计数，或一些归一化概率或对数概率。例如，Schwartz 等人(2013)通过对每个主题(subject)的词汇使用总量进行归一化，将特征数转化为短语的似然度。

$$p(phrase|subject) = \frac{\text{freq}(phrase, subject)}{\sum_{phrase' \in \text{vocab}(subject)} \text{freq}(phrase', subject)} \tag{20.14}$$

如果训练数据稀疏或与测试集不同，那么我们讨论的任何词汇都可以发挥有益的作用，无论是单独使用还是与所有单词和 n-gram 结合使用。许多可能的值可用于词汇功能。最简单的只是一个指示符函数，如果特定文本中有来自相关词汇 L 的任何单词，则特征 f_L 的值将取值为 1。使用第 4 章的符号，其中为特定的输出类 c 和文档 x 定义了一个特征值。

$$f_L(c,x) = \begin{cases} 1 & \text{if } \exists w : w \in L \ \& \ w \in x \ \& \ class = c \\ 0 & \text{otherwise} \end{cases}$$

或者，某个特定词汇 L 的特征 f_L 的值可以是出现在 L 中的文档中单词符记的总数:

$$f_L = \sum_{w \in L} count(w)$$

对于每个单词都与得分或权重相关联的词汇，可以将计数乘以权重θ_w^L:

$$f_L = \sum_{w \in L} \theta_w^L count(w)$$

也可以像等式 20.14 中那样按作者对计数取对数或归一化。

无论它们如何定义，这些词汇特征都将在监督分类器中用于预测文本或文档的所需感情类别。训练完分类器后，我们可以检查哪些词汇功能与哪些类相关联。对于逻辑回归等分类器，特征权重可指示特征与

类的关联程度。

因此，例如，Maresse 和 Walker(2008)发现，在对人格进行分类时，在 Agreeable 维度上，LIWC 词汇表中的 Family 和 Home 是正相关的，而 anger 和 swear 则是负相关的。相比之下，人格外向与 Friend, Religion 和 Self 词汇正相关，情绪稳定与 Sports 正相关，与负面情绪负相关。

在我们使用文档中的所有单词和短语作为潜在特征的情况下，我们可以使用从学习的回归分类器得到的权重作为感情词汇的基础。在 Schwartz 等人(2013)的外向/内向分类器中，使用普通最小二乘回归从所有的单词和短语来预测一个人格维度的价值。每个单词或短语的最终回归系数可以用作与预测维的关联值。图 20.13 中的单词云(clouds)显示了与内向(a)和外向(b)相关的单词的示例。请注意，当单词计数相关时，不能保证回归权重代表正确的关联，因此，如果需要精确的值，通常必须首先修剪单词列表。

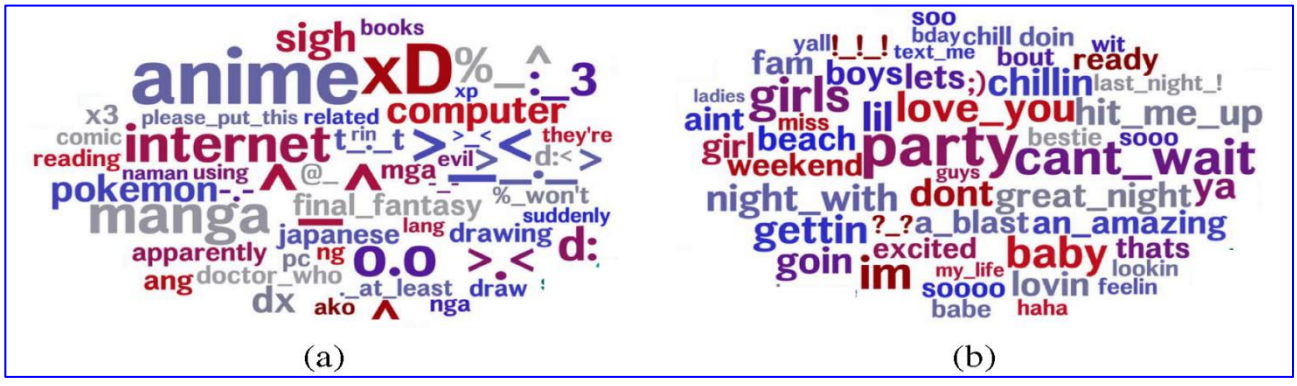


图 20-13：来自 Schwartz 等人(2013 年)的单词云

图注：显示了与内向(左)或外向(右)高度相关的词。单词的大小代表关联强度(回归系数)，颜色(从冷到热)代表单词/短语的相对频率(从低到高)。

20.9. 以实体为中心感情的基于词汇的方法

如果我们不希望获得整个文档的感情分数，而是文本中的特定实体的感情分数，该怎么办? Field 和 Tsvetkov(2019)的以实体为中心的方法将感情词汇与上下文嵌入相结合，为文本中的实体分配感情分数。在对人的感情方面，他们将效价/唤醒/支配(V/A/D)维度重新定义为情感/施事者/权力(S/A/P)。该算法首先训练分类器将嵌入映射到分数:

1. 对于训练语料库中的每个单词 w:

(a)使用现成的预训练编码器(如 BERT)为单词的每个实例提取上下文嵌入的 e，不做任何额外的微调。

(b)对 w 的每个实例的 e 次嵌入进行平均，得到一个训练点 w 的单个嵌入向量。

(c)使用 NRC VAD 词汇对 w 进行 S、A、P 评分。

2. 在所有单词 w 上训练(3 个)回归模型，根据单词的平均嵌入量预测 V, A, D 得分。

现在给定一个实体在文本中提及 m，我们分配如下感情分数:

1. 使用相同的预先训练的 LM 来获得 m 在上下文中的上下文嵌入。

2. 通过 3 个回归模型进行嵌入，得到实体的 S, A, P 分数。

这将导致一个(S,A,P)元组用于给定的实体提及;要获得完整文档中实体表示的得分，我们可以运行共指解析，并对所有提及的(S,A,P)得分进行平均。图 20.14 显示了他们对电影《黑暗骑士》中角色的算法在带有**黄金共指(gold coreference)**的维基百科情节总结文本上运行时的得分。

20.10. 内涵框架

到目前为止，我们所描述的词汇定义一个词为感情空间中的一个点。而内涵框架则是将第十章的情感词汇与框架语义词汇相结合，形成一种更丰富的语法结构的词汇。内涵框架词汇的基本观点是，像动词一样的谓词表达有关动词论元的内涵(Rashkin 等人 2016, Rashkin 等人 2017)。

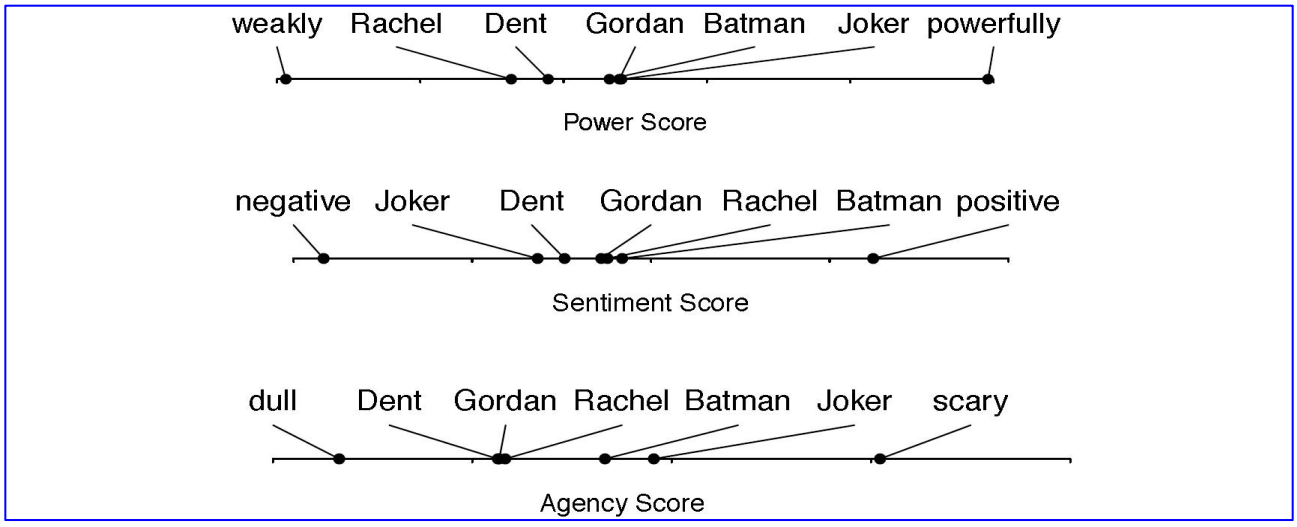


图 20-14：《黑暗骑士》中角色的权力(支配)、情感(效价)和施事者(唤起)

图注：是通过 NRC VAD 词典训练的嵌入计算出来的。注意主角(蝙蝠侠)和对手(小丑)的权力和施事者得分很高，但情感上有所不同，而爱慕对象 Rachel 的权力和施事者都很低，但情感很高。

考虑这样的句子:

(20.15) Country A violated the sovereignty of Country B

(20.16) the teenager ... survived the Boston Marathon bombing

通过使用(20.15)中的动词 violate，作者表达了对国家 B 的同情，将国家 B 描绘为受害者，并对施事者国家 A 表示敌对。相反，使用 survive 这个动词时，作者(20.16)表示轰炸是一种消极的经历，而句子的主题(少年)是一个同情的角色。这些内涵的体是动词 violate 和 survive 的意义所固有的，见图 20.15。

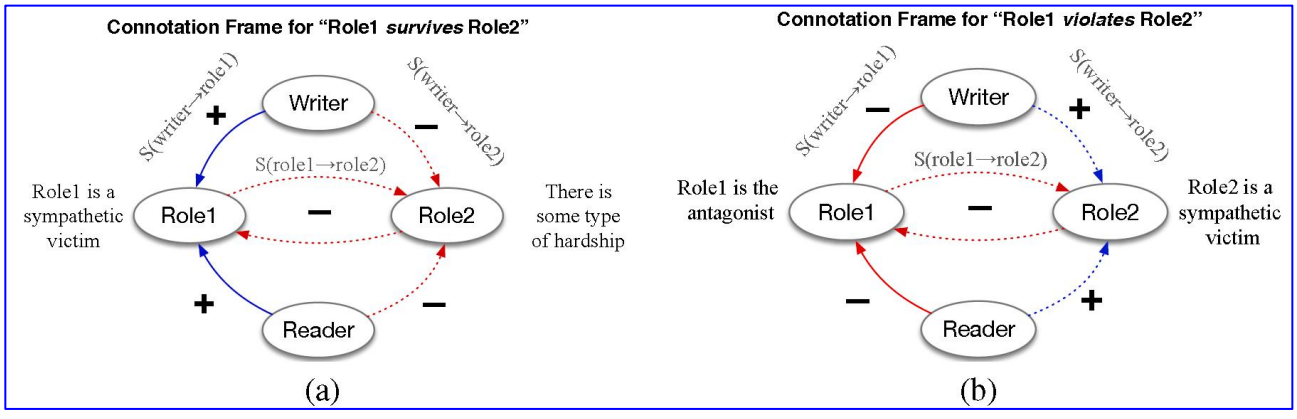


图 20-15：survive 和 violate 的内涵框架

图注：(a)survive：作者和读者对角色 1(主语)有积极的情感，对角色 2(直接宾语)有消极的情感。

(b)violate：作者和读者对角色 2(直接宾语)有积极的情感。

Rashkin 等人(2016)和 Rashkin 等人(2017)的内涵框架词汇也表达了谓词对每个论元的其他内涵方面，包括两个方面，第一是效果(x 发生了不好的事情)值：(x 是有价值的)；第二是心理状态：(x 被事件困扰)。内涵框架也可以标记论元之间的权力差异(使用动词 implore 意味着主题论元比施事者更有权力)，和每个论元的施事者(waited 是低级施事者)。图 20.16 显示了 Sap 等人(2017)的可视化结果。

可以手动构建内涵框架(Sap 等人，2017)，也可以通过监督学习(Rashkin 等人，2016)来学习它们，例如使用手工标签的训练数据来监督每个个体关系的分类器，例如 S(writer→Role1)是+还是-，然后通过所有关系上的全局约束来提高准确性。

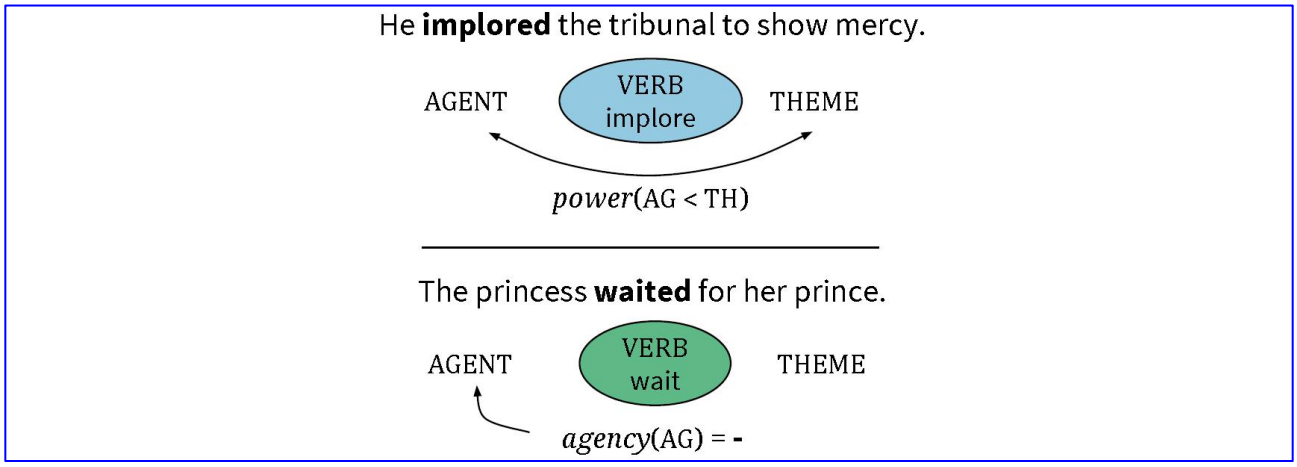


图 20-16：Sap 的内涵框架

图注：图中动词 implore 意味着施事者的权力低于主题(对照于动词 demanded)，表明 waited 的主语的施事者水平较低。

20.11. 总结

•可以区分多种**感情状态**，包括：情绪、语气、**态度(包括情感)**、人际立场和性格。

•**情绪**可用固定的原子单位表示，常被称为基本情绪，也可表示为**效价**和**唤起**等维度定义的空间中的点。

•单词具有与这些感情状态有关的**内涵**体，单词含义的这一内涵体可以用词汇表示。

•可以人工创建感情词汇，使用**众包**来标签每个单词的感情内容。

•可以使用相似度度量(嵌入余弦)从种子词的自举引导过程中建立**半监督**的词汇。

•在世界范围内可以找到方便的训练信号(例如，用户在查看站点上分配的评分)时，可以以**全监督**的方式学习词汇。

•通过在训练文本中使用单词计数的各种函数，以及像**对数优势比率信息 Dirichlet 先验**这样的比率指标，单词可以在词汇中分配权重。

•**性格**通常表现为 5 维空间中的一个点。

•感情与情感一样，可以通过使用标准的监督**文本分类**技术，将文本中的所有单词或字母作为特征来检测。附加特征可以从词汇表中的单词计数中抽取。

•词汇也可用于检测**基于规则**的分类器中的感情，根据每个词汇表中单词的数量选择简单多数情感。

•**内涵框架**表达了谓词对其论元所编码的更丰富的感情含义关系。

20.12. 文献和历史说明

正式表达词语主观意义的想法始于 Osgood 等人(1957)，这一开创性研究首次提出了第 6 章中描述的含义向量空间模型。Osgood 等人(1957)让参与者在不同的量表上对单词进行评分，并对评分进行因子分析。他们发现的最重要的因素是评价维度，它区分了好/坏、有价值/无价值、令人愉快/不愉快的配对。这部作品影响了**内容分析**领域早期情感和感情含义字典的发展(Stone 等人，1966)。

Wiebe(1994)在检测文本**主观性(subjectivity)**方面开始了一项有影响的工作，首先是识别主观句子和文本中被描述为持有私人状态、信仰或态度的主观人物。Hatzivassiloglou 和 McKeown(1997)的极性词汇等习得情感词汇在主观性检测中被证明是一个有用的特征(Hatzivassiloglou 和 Wiebe 2000, Wiebe 2000)。

情感一词似乎是 Das 和 Chen(2001)在 2001 年引入的，用来描述通过查看股票交易留言板上的词汇来衡量市场情感的任务。Das 和 Chen(2001)在同一篇论文中也提出了情感词汇的使用。词汇中的单词列表是手工创建的，但是每个单词都通过最大化类间差异和最小化类内差异来分配权重，这取决于它对特定类的区别程度(比如 buy 和 sell)。情感一词以及词汇的使用很快就流行起来了(例如，Turney 在 2002 年的一篇文章中就提到了这一点)。Pang 等人(2002)首先展示了在没有情感词汇的情况下使用所有单词的能力;参见 Wang 和 Manning(2012)。

我们用来扩展情感字典的大部分半监督方法都是基于同义词和反义词在同一句子中往往同时出现的早期想法(Miller 和 Charles 1991, Justeson 和 Katz 1991, Riloff 和 Shepherd 1997)。其他学习感情含义线索的半监督方法依赖于信息提取技术，如 AutoSlog 模式提取器(Riloff 和 Wiebe, 2003)。Hatzivassiloglou 和 McKeown(1997)首先提出了基于图的情感算法，而图传播成为了标准方法(Zhu 和 Ghahramani 2002, Zhu 等人 2003, Zhou 等人 2004, Velikovich 等人 2010)。众包也可以通过过滤半监督词汇学习的结果来提高精度(Riloff 和 Shepherd 1997, Fast 等人 2016)。

最近的许多工作集中于学习直接编码情感或其他属性的嵌入的方法，例如 Rothe 等人(2016)的 DENSIFIER 算法，该算法学习如何转换嵌入空间，专注于情感(或其他)信息。

20.13. 练习

(无)

21. 共指消解

> and even Stigand, the patriotic archbishop of Canterbury, found it advisable–"'
> 'Found WHAT?' said the Duck.
> 'Found IT,' the Mouse replied rather crossly: 'of course you know what "it"means.'
> 'I know what "it"means well enough, when I find a thing,' said the Duck: 'it's generally a frog or a worm. The question is, what did the archbishop find?'
>
> Lewis Carroll, Alice in Wonderland

语言理解的一个重要成分是知道在一篇文章中谈论的是谁。考虑以下段落:

(21.1) Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old became the company's president. It is widely known that she came to Megabucks from rival Lotsabucks.

这篇文章中每一个下划线的短语都是作者用来指一个名叫维多利亚·陈的人的。我们把像 *her* 或 *Victoria Chen* 这样的语言表达称为**提及(mention)**或**指示语(referring expression)**，被提及的话语实体(Victoria Chen)称为**所指对象(referent)**。(为了区分指示语及其指代对象，我们将前者斜体化)[1]。两个或两个以上的指示语用于指代同一话语实体，我们称之为**共指(corefer)**;因此，*Victoria Chen* 和 *she* 在(21.1)是共指。

共指是自然语言理解的重要成分。如果一个对话系统告诉用户"*联合航空下午 2 点有一班航班，国泰航空下午 4 点有一班*"，用户就会知道他说的是"*我要乘第二班*"。一个使用维基百科(Wikipedia)来回答关于居里夫人(Marie Curie)的问题的问答系统，必须在"她出生在华沙"(she was born in Warsaw)这句话中知道*她(she)*是谁。从像西班牙语这样的语言翻译而来的机器翻译系统(可以在其中插入代词)必须使用前一句子的共指来确定西班牙语句子*'" Me encanta el conocimiento" dice.'* 是否应翻译为 *'"I love knowledge", he says'*或*'"I love knowledge", she says'*。的确，此示例来自 *El País* 上的一篇有关一位女教授的实际新闻，由于共指消解不正确而在机器翻译中被误译为"he"(Schiebinger，2013)。

自然语言理解系统(和人类)根据话语模型来解释语言表达(Karttunen，1969)。**话语模型(discourse model)(**图 21.1)是一种心智模型，理解者在解释文本时会逐步构建，其中包含文本中所指实体的表示形式，实体的属性以及它们之间的关系。当首先在话语中提及一个所指对象时，我们说它的表示形式被**唤起(evoked)**进入模型中。在随后提及时，可以从模型**访问(accessed)**此表示。

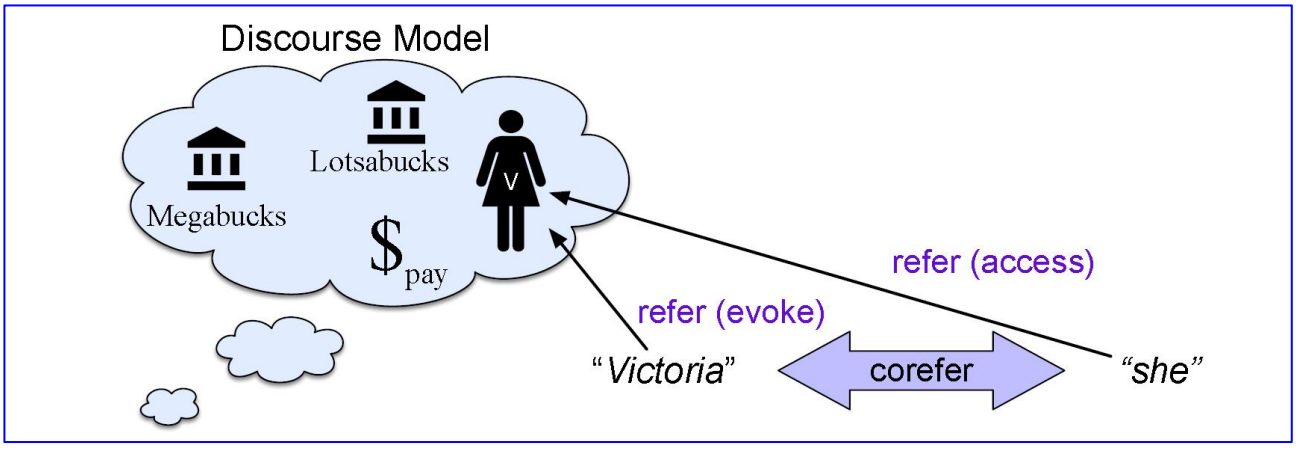


图 21-1：在话语模型中，提及如何唤起和访问话语实体

[1] 为了方便起见，有时我们会说一个指示语来指代一个所指对象，例如说 she 指代 Victoria Chen。但是，读者应该记住，我们真正的意思是，说话者是在通过说 she 来指代 Victoria Chen。

在文本中对先前已引入到话语中的实体的引用被称为**回指(anaphora)**，并且所使用的指示语被称为是**回指语(anaphor)**或照应词(anaphoric)[2]。因此，在段落(21.1)中， 代词 *she* 和 *her* 以及有定名词短语 *the 38-year-old* 都是回指语。回指语共指先前的提及(在本例中为 *Victoria Chen*)，这个提及被称为**先行词(antecedent)**。并非每个指示语都是一个先行词。在文本中只有一个提及的实体(例如(21.1)中的Lotsabucks)称为**单例(singleton)**。

在这一章中，我们将重点讨论共指消解(coreference resolution)的任务。共指消解是确定两个提及是否共指的任务，我们的意思是它们指的是话语模型中的同一实体(同一话语实体)。指示语的集合通常称为**共指链(coreference chain)**或**聚类(cluster)**。例如，在(21.1)的处理过程中，一个共指消解算法需要找到至少四个共指链，对应于图 21.1 中的话语模型中的四个实体。

1. *{Victoria Chen , her, the 38-year-old, She}*
2. *{Megabucks Banking, the company, Megabucks}*
3. *{her pay}*
4. *{Lotsabucks}*

注意，提及可以嵌套;例如，在语法上，提及 her 在语法上是另一个提及 her pay 的一部分，指的是一个完全不同的话语实体。因此，共指消解包括两个任务(尽管它们通常是共同执行的):(1)识别提及，(2)将它们聚类为共指链/话语实体。

我们说过，如果两个提及与同一个话语实体相关联，那么这两个提及就很重要。但通常我们想走得更远，确定哪个真实世界实体与此话语实体相关联。例如，提及华盛顿可能指的是美国州、首都或乔治华盛顿。对于这些句子的解释当然会有很大的不同。**实体连接(entity linking)**(Ji 和 Grishman，2011)或实体消解的任务是将话语实体映射到某些现实世界的个体任务[3]。我们通常通过映射到本体来操作实体连接或消解：世界上的实体列表，例如 gazeteer(第 15 章)。也许用于此任务的最常见的本体是 Wikipedia。每个 Wikipedia 页面都充当特定实体的唯一 ID。因此，维基化的实体连接任务(Mihalcea 和 Csomai，2007)是决定哪个维基百科页面对应的个体被提及。哪一个对应于个体的维基百科页面通过一个提及被共指。但是实体连接可以用任何本体来完成。例如，如果我们有一个基因本体，则可以将文本中提及的基因连接到本体中已消除歧义的基因名称。

在下一节中，我们将更详细地介绍共指消解的任务，并提供各种消解架构，从简单的确定性基线算法到最先进的神经模型。

然而，在讨论算法之前，我们要提及一些重要的任务，我们只会在本章的最后简要地介绍一下。首先是著名的 Winograd 模式问题(之所以这样称呼是因为它们是由 Terry Winograd 在他的论文中首先指出的)。这些实体共指消解问题被设计成很难用我们在本章中描述的解析方法来解决，而它们所需要的现实世界知识使它们成为一种具有挑战性的自然语言理解任务。例如，在下面的例子中，我们来看看如何确定代词 they 的正确先行词:

(21.2) The city council denied the demonstrators a permit because
a. they feared violence.
b. they advocated violence.

要确定代词的正确先行词，他们需要理解第二个从句是对第一个从句的解释，而且市议会可能比游行者更害怕暴力，游行者可能更倾向于提倡暴力。解决 Winograd 模式问题需要找到表示或发现必要的现实世界知识的方法。

我们在本章不讨论的一个问题是**事件共指(event coreference)**的相关任务，即确定两个事件提及(例如这两个来自 ECB+语料库的句子中的 buy 和 acquisition)是否指的是同一事件:

---

[2] 我们将按照常用的回指词的 NLP 用法，来表示有先行词的任何提及，而不是更狭义的用法来仅表示其解释取决于先行词（在更狭义的解释下，重复的名称不是回指词)的提及（如代词)。

[3] 因此，计算语言学/自然语言处理在使用“所指对象(reference)”一词时有别于形式语义领域，该领域使用“所指对象(reference)”和“共指(coreference)”两词来描述提及与现实实体之间的关系。相比之下，我们遵循功能语言学的传统，即提及指的是话语实体(Webber, 1978)，话语实体与现实世界个体之间的关系需要额外的连接步骤。

(21.3) AMD agreed to [buy] Markham, Ontario-based ATI for around $5.4 billion in cash and stock, the companies announced Monday.

(21.4) The [acquisition] would turn AMD into one of the world's largest providers of graphics chips.

事件提及比实体提及更难检测，因为它们既可以是动词性表达也可以是名词性表达。一旦检测到，用于实体的相同提及配对和提及排名模型通常会应用于事件。

一种更为复杂的共指是**话语指称(discourse deixis)**(Webber, 1988)，在其中一个回指词指回到一个话语片段，这种语段很难界定或分类，如(21.5)中的例子改编自 Webber (1991):

(21.5) According to Soleil, Beau just opened a restaurant

a. But *that* turned out to be a lie.

b. But *that* was false.

c. *That* struck me as a funny way to describe the situation.

所指对象 that 是(21.5a)中的言语行为(见第 24 章)，(21.5b)中的命题，(21.5c)中的描述方式。在本章中，我们不会为这些困难类型的非名词类先行词给出算法，但请参阅 Kolhatkar 等人(2018)的一项概述。

21.1. 共指现象：语言背景

我们现在提供一些有关共指现象的语言学背景。我们介绍了四种类型的指示语：有定和不定(definite and indefinite)的 NP、代词和名称(pronouns and names)，描述了它们是如何在话语模型中唤起和访问实体的，并讨论了回指语/先行词的语言特征(如数/性别一致，或动词语义的属性)。

21.1.1. 指示语的类型

不定名词短语(Indefinite Noun Phrases)：英语中最常见的不定指代形式是用限定词 a(或 an)来标记的，但它也可以用量词 some 甚至是限定词 this 来标记。不定指代通常引入对听众来说是新颖的话语上下文实体。

(21.6) a. Mrs. Martin was so very kind as to send Mrs. Goddard *a beautiful goose*.

b. He had gone round one day to bring her *some walnuts*.

c. I saw *this beautiful cauliflower* today.

有定名词短语(Definite Noun Phrases)：明确的指代，例如通过使用英语冠词 the 的 NP 进行引用，指的是对于听众来说是可识别的实体。听众可以识别一个实体，因为该实体先前已在文本中提及，因此已经在话语模型中表示出来：

(21.7) It concerns a white stallion which I have sold to an officer.

But the pedigree of *the white stallion* was not fully established.

或者，一个实体可以被识别，因为它包含在听众关于世界的信念集合中，或者该对象的唯一性由描述本身暗示，在这种情况下，它唤起了对象在话语模型中的表示，例如 在(21.9)中：

(21.8) I read about it in the *New York Times*.

(21.9) Have you seen the car keys?

这些最后的用法很常见。新闻通讯文本中超过一半的有定 NP 为非回指的，通常是因为它们是第一次提及的实体(Poesio 和 Vieira 1998，Bean 和 Riloff 1999)。

代词(pronouns):另一形式的有定指代是名词化，用于话语中非常突出的实体(我们将在下面讨论):

(21.10) Emma smiled and chatted as cheerfully as *she* could.

代词也可以参与**后指(cataphora)**，如(21.11)，代词在所指对象之前被提及。

(21.11) Even before *she* saw *it*, Dorothy had been thinking about the Emerald City every day.

其中，代词 she 和 it 都出现在它们的所指对象被引入之前。

代词也出现在被认为是受**绑定(bound)**的量化上下文中，如(21.12)。

(21.12) Every dancer brought *her* left arm forward.

在相关阅读材料下，she 并不是指上下文中的某个女人，而是像一个变量绑定在 every dancer 的量化

表达上。在本章中，我们不讨论代词的绑定解释。

在一些语言中，代词可以作为附在单词上的附注出现，例如下面这个西班牙语例子中的 lo(“it”)，来自 AnCora (Recasens 和 Martı,2010):

(21.13) La intención es reconocer el gran prestigio que tiene la maratón y unir**lo** con esta gran carrera.

“The aim is to recognize the great prestige that the Marathon has and join|**it** with this great race.”

指示代词(Demonstrative Pronouns):指示代词 this 和 that 既可以单独出现，也可以作为限定词出现，例如，this ingredient, that spice:

(21.14) I just bought a copy of Thoreau’s Walden. I had bought one five years ago.
That one had been very tattered; this one was in much better condition.

注意，其中 this 是有歧义的;在口语英语中，它可以是不定的，如(21.6)，也可以是有定的，如(21.14)。

零回指(zero anaphor)：在某些语言(包括中文、日语和意大利语)中，不使用代词，而是可能使用完全没有词汇实现的回指，称为零指称或零代词，如以下意大利语和日语的示例 Poesio 等人(2016)：

(21.15) EN [John]$_i$ went to visit some friends. On the way [he]$_i$ bought some wine.
IT [Giovanni]$_i$ ando a far visita a degli amici. Per via φ_i compr`o del vino.
JA [John]$_i$-wa yujin-o houmon-sita. Tochu-de φ_i wain-o ka-tta.

或者这个中文例子：

(21.16) [我] 前一会精神上太紧张.....[0] 现在比较平静了
[I] was too nervous a while ago..... [0] am now calmer.

在这些语言中，零回指使提及检测的任务复杂化。

名称(Names)：名称(如人、地点、组织等)可以用来指代话语中的新旧实体:

(21.17) a. **Miss Woodhouse** certainly had not done him justice.
b. **International Business Machines** sought patent compensation from Amazon;
IBM had previously sued other companies.

21.1.2. 信息状态

指示语的下列两种使用方式之一，称为他们的**信息状态(information status)**或信息结构。该方式是：唤起新的所指对象进入话语(引入新的信息)；或者访问旧的来自模型实体(旧信息)实体可以是**新的话语(discourse-new)**，或者**旧的话语(discourse-old)**，实际上，通常在信息上区分至少三种实体是很普遍的(Prince，1981a)：

新的 NP：
全新的 NP：这些引入的实体是新的话语和新的听众
(如 a fruit or some walnuts.)
未用的 NP：这些引入的实体是新的话语和旧的听众
(如 Hong Kong, Marie Curie, or the New York Times.)
旧的 NP：也称唤起的 NP，它们引入话语模型中已经存在的实体，因此既是旧的话语又是旧的听众的
(如 I went to a new restaurant. It was...中的 it .)

推理对象(inferrables)：这些引入的实体既不是旧的听众也不是旧的话语，但是听众可以通过基于话语中其他实体的推理来推断它们的存在。考虑以下例子:

(21.18) I went to a superb restaurant yesterday. The chef had just opened it.

(21.19) Mix flour, butter and water. Knead the dough until shiny.

不管是厨师(the chef)还是面团(the dough)都不在上述两个例子中的任何一个示例的第一句话的话语模型中，但是读者可以根据世界的观点做出**桥接推理(bridging inference)**，即这些实体应该添加到话语模型中并与餐厅和食材相关联，这个推理是基于世界知识，即餐厅有厨师和生面团是面粉和液体混合的结果(Haviland 和 Clark 1974，Webber 和 Baldwin 1992，Nissim 等人 2004，Hou 等人 2018)。

NP 的形式为其信息状态提供了强有力的线索。我们经常谈论的是实体在**给定-新的(given-new)**维度

上的位置，即所指对象是给定(given)的程度(话语中的显著性，听者更容易想到，听者可以预测)，对应于新的(new)的程度(话语中的非显著性，不可预测性)(Chafe 1976，Prince 1981b，Gundel 等人 1993)。如果一个所指对象在听者的脑海中非常突出或易于唤起人们的目光，则称其为**易于访问的(accessible)**(Ariel，2001 年)所指对象。这样的所指对象可以使用较少语言的材料来指代。例如，在话语模性中，仅当所指对象具有高度激活程度或**显著性(salience)**时，代词才被使用[4]。相比之下，不太显著的实体，如一个新的所指对象被引入话语中，将需要引入一个更长的和更明确的指示语，以帮助听众恢复所指对象。

因此，当实体首次被引入话语时，其提及的内容可能具有全名、头衔或角色、或者贴切的或限制性的相对从句，如同在(21.1)中介绍我们的主角：Megabucks Banking 的 CFO Victoria Chen。在通过话语讨论实体时，它对听众而言变得更加显著，并且平均而言，它的提及通常会变得简短且信息量较少，例如，使用简短的名字(例如 Ms. Chen)，明确的描述(the 38-year-old)或代词(she 或 her)(Hawkins 1978)。但是，这种长度的变化不是单调的，并且对话语结构很敏感(Grosz 1977b，Reichman 1985，Fox 1993)。

21.1.3. 复杂化：非指示语

许多名词短语或其他名词并非指示语，尽管它们可能有令人困惑的浅表相似性。例如，在一些最早的指代消解计算工作中，Karttunen(1969)指出，以下例子中的名词短语 a car 并不产生话语所指对象:

(21.20) Janet doesn't have a car.

并且不能通过回指词 it 或 the car 被回指：

(21.21) *It is a Toyota.

(21.22) *The car is red.

我们在此总结了四种常见的结构类型，它们不被算作共指任务中的提及，从而使提及检测任务复杂化:

同位语(Appositives):同位语结构是一个名词短语出现在一个中心名词短语的旁边，用来描述中心词。在英语中，它们经常以逗号形式出现，比如"a unit of UAL"与 NP United 相对立，或者"CFO of Megabucks Banking"与 Victoria Chen 相对立。

(21.23) Victoria Chen, CFO of Megabucks Banking, saw ...

(21.24) United, a unit of UAL, matched the fares.

同位语 NP 并不是指示语，而是充当对中心词 NP 的一种补充插入描述的功能。尽管如此，有时将这些短语与它们所描述的实体联系起来是有用的，所以像 OntoNotes 这样的数据集标记了对应关系。

表语和前名词 NP：表语或定语 NP 描述中心(head)名词的特性。在 United is a unit of UAL 中，a unit of UAL 这个 NP 描述的是 United 的属性，而不是指不同的实体。因此，它们在共指任务中没有被标记为提及。在我们的示例中，$2.3 million 和 the company's president 者两个 NP 是定语，描述了 her pay 和 the 38-year-old 的特性；范例(21.27)显示了一个中文范例，其中未提及谓词 NP(中国最大的城市; China's biggest city)。

(21.25) her pay jumped to *$2.3 million*

(21.26) the 38-year-old became *the company's president*

(21.27)上海是[中国最大的城市] [Shanghai is *China's biggest city*]

咒骂语(Expletives)：英语中的代词和其他语言中的相应代词的许多用法都不具有指称性。

这样的咒骂的或**赘述的(pleonastic)**情况包括 it is raining，在成语中如同 hit it off，或者在特殊的句法情况下如同**分裂(clefts)**(21.28a)或**外置(extraposition)**(21.28b)：

(21.28) a. It was Emma Goldman who founded Mother Earth

b. It surprised me that there was a herring hanging on her wall

类属指代(Generics):另一种表达方式是类属的指代，该表达方式不会回指到在文本中显式唤起的实体。考虑(21.29)。

(21.29) I love mangos. They are very tasty.

[4] 代词通常(但不总是)指的是在进行中的话语中一两句以内引入的实体，而有定名词短语通常指的是更早之前的实体。

这里，they 不是指代一个特定的芒果或一组芒果，而是指总体上的芒果。代词 you 也可以泛指:

(21.30) In July in San Francisco you have to wear a jacket.

21.1.4. 共指关系的语言属性

既然我们已经看到了个体指示语的语言属性，我们就转向先行词/回指词偶对的属性。理解这些属性对设计新特征和进行错误分析都有帮助。

数的一致性(Number Agreement): 指示语与它们的所指对象在数上必须大体一致;英语 she/her/he/him/his/it 为单数形式，we/us/they/them 为复数形式，而且 you 为未指定的数。因此，像 the chefs 这样的复数先行词通常不能与像 she 这样的单数回指词共指。然而，算法不能过于严格地强制数的一致性。首先，语义上的复数实体可以由 it 或 they 指代:

(21.31) IBM announced a new machine translation product yesterday.
They have been working on it for 20 years.

其次，**单数他们(singular they)**已变得越来越普遍，在单数 they 中，they 用来描述单个的个体，由于 they 对性别的中立性而常常有用。尽管最近有所增加，但 singular they 形式已经很古老了，许多世纪以来都是英语的一部分[5]。

人称一致性(Person Agreement):英语区分第一人称、第二人称和第三人称，代词的先行词必须与人称代词一致。因此第三人称代词(he, she, they, him, her, them, his, her, their)必须有第三人称先行词(以上任何一个或任何其他名词短语)。然而，像引号这样的现象会导致异常;在这个例子中，I, my 和 she 是共指的:

(21.32) “I voted for Nader because he was most aligned with my values,” she said.

性别或名词类的一致性(Gender or Noun Class Agreement):在许多语言中，所有名词都有语法性别或名词类[6]，代词通常与其先行词的语法性别一致。在英语中，这种用法只出现在第三人称单数代词上，用以区分男性(he, him, his)、女性(she, her)和非人类语法性别(it)。像 ze 或 hir 这样的非二元代词也可能出现在更近期的文本中。要知道在文本中把哪个性别与一个名字联系起来是很复杂的，可能需要对个体世界的了解。一些例子:

(21.33) Maryam has a theorem. She is exciting. (she=Maryam, not the theorem)

(21.34) Maryam has a theorem. It is exciting. (it=the theorem, not Maryam)

约束理论限制(Binding Theory Constraints)：约束理论是对同一句子中提及和先行词之间的关系进行句法约束的名称(Chomsky，1981)。稍微简化一点，像 himself 和 herself 这样的反身代词指代包含它们的最直接从句的主语(21.35)，而非反身代词不能指代这个主语(21.36)。

(21.35) Janet bought herself a bottle of fish sauce. [herself=Janet]

(21.36) Janet bought her a bottle of fish sauce. [her=Janet]

新近性(Recency):在最近的语段中引入的实体往往比从更早更远(further back)的语段中引入的实体更为显著。因此，在(21.37)中，代词 it 更可能指的是 Jim 's map 而不是 doctor 's map。

(21.37) The doctor found an old map in the captain’s chest.
Jim found an even older map hidden on the shelf. It described an island.

语法作用(Grammatical Role)：在主语位置提到的实体比在宾语位置提到的实体更显著，而在宾语位置提到的实体则比在倾斜(oblique)位置提到的实体更重要。因此，尽管(21.38)和(21.39)中的第一句话表达了大致相同的命题内容，但该代词 he 的首选指称对象随主语而有所不同-在(21.38)中是 John，而在(21.39)中是 Bill。

---

[5] 这里有一个来自莎士比亚的《错误的喜剧》的代词例子:There’ s not a man I meet but doth salute me As if I were their well-acquainted friend（我遇到的男人中，没有一个不向我致敬，就好像我是他们的知己一样)。

[6] “性别”这个词一般只用于有 2 或 3 个名词类的语言，如大多数印欧语言;许多语言，如班图语或汉语，都有大量的名词类。

(21.38) Billy Bones went to the bar with Jim Hawkins. He called for a glass of rum. [he = Billy]
(21.39) Jim Hawkins went to the bar with Billy Bones. He called for a glass of rum. [he = Jim]

动词语义学(Verb Semantics)：一些动词在语义上强调其论元之一，偏向于对后续代词的解释。比较(21.40)和(21.41)。
(21.40) John telephoned Bill. He lost the laptop.
(21.41) John criticized Bill. He lost the laptop.

这些示例仅在第一句中使用的动词有所不同，但(21.40)中的“He”通常被解析为 John，而(21.41)中的“He”被解析为 Bill。这可能是由于隐式因果关系与显著性之间的联系所致：“criticizing” 事件的隐式原因是其宾语，而“telephoning”事件的隐式原因是其主语。在这样的动词中，作为隐含原因的实体更为突出。

选择限制(Selectional Restrictions)：许多其他种类的语义知识也可以在指称偏好中发挥作用。例如，动词对其论元施加的选择限制(第 10 章)可以帮助消除所指对象，如(21.42)中所示。
(21.42) I ate the soup in my new bowl after cooking it for hours

对 it 来说有两个可能的所指对象，soup 和 bowl。然而，动词 eat 要求它的直接宾语是指可食用的东西，这一限制可能排除了 bowl 作为所指对象的可能性。

21.2. 共指任务和数据集

我们可以将共指消解的任务表述为:给定一个文本 T，找出所有实体及其之间的共指链。我们通过比较系统创建的连接与 t 上人工创建的黄金共指注释中的连接来评估我们的任务。让我们回到我们的共指示例，现在为每个共指链(聚类)使用上标数字，为聚类中的单个提及使用下标字母:

(21.43) [Victoria Chen]^1_a, CFO of [Megabucks Banking]^2_a, saw [[her]^1_b pay]^3_a jump to $2.3 million, as [the 38-year-old]^1_c also became [[the company]^2_b 's president. It is widely known that [she]^1_d came to [Megabucks]^2_c from rival [Lotsabucks]^4_a .

假设例子(21.43)是这篇文章的全部内容，那么 her pay 和 Lotsabucks 链都是单独提到的:

1. { *Victoria Chen, her, the 38-year-old, She* }
2. { *Megabucks Banking, the company, Megabucks* }
3. { *her pay* }
4. { *Lotsabucks* }

对于大多数共指评估活动，系统的输入是文章的原始文本，系统必须检测提及，然后将它们连接到聚类中。解决这个任务需要处理代词的回指(找出 her 指代 Victoria Chen),过滤掉**无所指(non-referential)**代词诸如赘述的(pleonastic)It has been ten years 中的 it),处理有定名词短语来找出 the 38-year-old 是指代 Victoria Chen,而且 the company 与 Megabucks 是一样的。我们需要处理名字，意识到 Megabucks 与 Megabucks Banking 是一样的。

确切地说，什么才算是提及？任务之间的注释链以及数据集之间的注释连接有何不同?例如，某些共指数据集不标签单例，从而使任务更加简单。解析器可以在没有单例的情况下在语料库上获得更高的分数，因为单例构成了运行文本中提到最多的内容，并且它们通常很难与无所指 NP 区别开来。某些任务使用了金牌提及检测(即为系统提供了人工标签的提及边界，而任务只是将这些金牌提及进行聚类)，这消除了从运行文本中检测和分割提及的需要。

共指的评估通常采用 CoNLL F1 分数，它结合了三个指标:MUC、B3 和 CEAFe;第 21.7 节给出了细节。

我们来谈谈一种流行的共指数据集 OntoNotes(Pradhan 等人 2007； Pradhan 等人 2007a)以及基于该数据集的 CoNLL 2012 共享任务的一些特征(Pradhan 等人 2012a)。OntoNotes 包含手动注释的中文和英文共指数据集，每个数据集大约一百万个单词，包括新闻通讯，杂志文章，广播新闻，广播对话，Web 数据和对话语音数据，以及大约 30 万个单词的阿拉伯语新闻通讯。OntoNotes 最重要的区别特征是它不标签单例，简化了共指任务，因为单例占所有实体的 60％-70％。在其他方面，它与其他共指数据集相似。

指示语 NP 被标记为提及，但类属指代和赘述代词未被标记。同位语从句不标记为单独的提及，但包含在提及中。因此，在 NP“Richard Godown, president of the Industrial Biotechnology Association”中，提及就是这个完整的短语。只有前名词修饰语是专有名词时，它们才被注释为单独的实体。因此，wheat 不是 wheat fields 中的实体，但 UN 是 UN policy 中的实体(但不是形容词诸如 American 在 American policy 中)。许多语料库都标记了更丰富的话语现象。ISNotes 语料库注释了一部分 OntoNotes 的信息状态，包括桥接示例(Hou 等人，2018)。LitBank 共指语料库(Bamman 等人 2020)包含来自 100 种不同文学小说的 210,532 个符记的共指注释，包括单例以及量化和否定的名词短语。AnCora-CO 共指语料库(Recasens 和 Martí，2010)包含 400,000 个单词，分别来自 Spanish 语(AnCora-CO-Es)和 Catalan 语(AnCora-CO-Ca)新闻数据，并在两种语言中都包含诸如话语指示等复杂现象的标签。ARRAU 语料库(Uryupina 等人，2020)包含 350,000 个英语单词，标记所有 NP，这意味着可以使用单例聚类。ARRAU 包括对话(TRAINS 数据)和小说(Pear Stories)等多种类型，并具有桥接所指对象的标签、话语指示、类属指代和模棱两可的回指关系的标签。

21.3. 提及检测

共指的第一阶段是**提及检测(mention detection)**：找到构成每个提及的文本跨度。提及检测算法通常在提议候选者提及时非常自由(即强调回忆)，并且仅在以后进行过滤。例如，许多系统在文本上运行解析器和命名实体标记器，并提取每个跨度，这些跨度可以是 NP、物主代词或命名实体。

在(21.44)中重复的示例文本中这样做：

(21.44) Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old also became the company’s president. It is widely known that she came to Megabucks from rival Lotsabucks.

可能会导致以下 13 个潜在提及列表:

Victoria Chen	$2.3 million	she
CFO of Megabucks Banking	the 38-year-old	Megabucks
Megabucks Banking	the company	Lotsabucks
her	the company’s president	
her pay	It	

最近提到的检测系统更加度量宽大。我们将在第 21.6 节中描述的基于跨度的算法首先从字面上提取所有 N-gram 跨度为 N = 10 的单词。当然，从 21.1.3 节中回想起，许多 NP(以及绝大多数随机 N-gram 跨度)都不是指示语。因此，所有这些提及的检测系统最终都需要过滤出诸如上述 it 之类的赘述词，CFO of Megabucks Banking Inc 之类的同义语，the company’s president 或 $2.3 million 之类的谓语名词。

某些过滤可以通过规则来完成。早期的基于规则的系统设计正则表达式来处理赘述词 it，就像下面来自 Lappin 和 Leass(1994)的规则，使用认知动词的字典(例如 believe, know, expect)来捕获赘述词 it 于“It is thought that ketchup…”之中，或者捕获情态形容词(例如 necessary, possible, certain, important)于“It is likely that I…”之中。这些规则有时被用作现代系统的一部分:

It is Modaladjective that S
It is Modaladjective (for NP) to VP
It is Cogv-ed that S
It seems/appears/means/follows (that) S

提及检测规则有时是专门为特定评估活动(ampaigns)设计的。例如，对于 OntoNotes，提及并没有嵌入较大的提及中，并且尽管对数字数量进行了注释，但它们很少具有核心意义。因此，对于像 CoNLL 2012(Pradhan 等人，2012a)这样的 OntoNotes 任务，一种常见的首关(first pass)基于规则的提及检测算法(Lee 等人，2013)是：

1. Take all NPs, possessive pronouns, and named entities.
2. Remove numeric quantities (100 dollars, 8%), mentions embedded in larger mentions, adjectival forms of nations, and stop words (like *there*).
3. Remove pleonastic it based on regular expression patterns.

但是，基于规则的系统通常不足以处理提及检测，因此现代系统结合了某种习得的提及检测组件，例如指称性(referentiality)分类器，回指性(anaphoricity)分类器(检测 NP 是否是一个回指词)或全新的话语(discourse-new)分类器-检测提及是否是全新的话语以及将来的回指的潜在先行词。

例如，一个**回指性检测器(anaphoricity detector)**可以从任何在手工标签的数据集(如 OntoNotes、ARRAU 或 AnCora)中标签为回指指示语的跨度中提取积极的训练示例。任何其他 NP 或命名实体都可以标记为一个消极训练示例。回指性分类器利用候选提及的中心词、周边词、确定性、活动性、长度、在句子/话语中的位置等特征，其中许多特征在 Ng 和 Cardie (2002a)的早期著作中首次提出;有关特征的更多信息，请参阅 21.5 节。

指称性或回指性检测器可以作为过滤器运行，其中只有被归类为回指性或指称性的提及被传递给共指系统。在我们上面的例子中，这种过滤提及检测系统的最终结果可能是以下 9 个潜在提及的过滤集:

Victoria Chen	her pay	she
Megabucks Bank	the 38-year-old	Megabucks
her	the company	Lotsabucks

然而，事实证明，基于回指性或指称性分类器对提及内容进行硬过滤会导致性能较差。如果回指性分类器阈值设置得太高，就会过滤掉太多的提及，召回率受影响；如果分类器阈值设置得太低，就会包含太多的赘述词或非指代提及，从而精度受影响。

相反，现代方法是在单个端到端模型中联合执行提及检测、回指性和共指(Ng 2005b，Denis 和 Baldridge 2007，Rahman 和 Ng 2009)。例如，Lee 等人(2017)(2018)系统中的提及检测是基于单个端到端神经网络的，该网络计算每个提及为指称对象的得分，计算两个提及为共指的得分，并将它们组合起来以做出决定，用单个端到端的损失就可以训练所有这些分数。我们将在第 21.6 节中详细介绍此方法。[7]

尽管有这些进步,正确检测所指对象的提及似乎仍然是一个尚未解决的问题,因为系统错误地将赘述代词(如 it)和其他非指称 NP 标记为共指，所以这个系统是现代共指消解系统的一大错误来源(Kummerfeld 和 Klein 2013，Martschat 和 Strube 2014，Martschat 和 Strube 2015，Wiseman 等人 2015，Lee 等人 2017)。

因此，提及，指称性或回指性检测是一个重要的开放研究领域。其他知识来源可能会有所帮助，尤其是与无监督和半监督算法结合使用时，也可以减少标记数据集的费用。在早期的工作中，例如 Bean 和 Riloff(1999)学习了表征回指或非回指 NP 的模式(通过提取和归纳文本中的第一个 NP 来保证它们是非回指的)。Chang 等人(2012)寻找在训练数据中频繁出现但从未作为金牌提及出现的中心词(head)名词，以帮助找到非指称 NP。Bergsma 等人(2008b)使用网络计数作为一种半监督的方式来增强英语 it 的回指性检测的标准特征，这是一项重要的任务，因为它既常见又有歧义。在 1/4 到 1/2 之间，它的例子是非回指的。请考虑以下两个示例：

(21.45) You can make [it] in advance. [anaphoric]

(21.46) You can make [it] in Hollywood. [non-anaphoric]

“make it”中的“it”是非回指的，是习语“make it”的一部分。Bergsma 等人(2008b)将每个例子周围的上下文转换成模式，如(21.45)中的“make * in advance”和(21.46)中的“make * in Hollywood”。然后他们使用谷歌 n-gram 来枚举所有可以在模式中取代 it 的单词。非回指上下文往往只把 it 放在通配符位置上，而回指上下文出现在许多其他 NP 中(例如，make them in advance 和 make it in advance 在它们的数据中出现的频率一样，但 make them in Hollywood 根本就没有出现)。这些 N-gram 上下文可以作为受监督的回指性分类器的特征。

[7] 一些系统试图完全避免提及检测或回指检测。对于像 OntoNotes 这样的不标记单例的数据集，过滤掉非指代提及的另一种方法是运行共指消解，然后简单地删除所有未被其他提及所共指的候选提及。这可能不像对指代性进行显式建模那样有效，也无法解决检测单例的问题，这对于诸如实体连接之类的任务很重要。

21.4. 共指算法的架构

用于共指的现代系统是基于监督的神经机器学习，该监督是通过手工标记的数据集(如 OntoNotes)进行监督的。在本节中，我们概述现代系统的各种体系结构，使用的方法是 Ng(2010)分类，该分类从两个方面来区分不同的算法：第一方面，他们是否以以下方式做出每个共指决定：基于实体(代表话语模型中的每个实体)或仅基于提及(独立考虑每个提及)；第二方面，他们是否使用排名模型直接比较潜在的先行词。之后，我们将在第 21.6 节中详细介绍一种最新算法。

21.4.1. 提及配对架构

我们从**提及配对(mention-pair)**体系结构开始，这是最简单、最具影响力的共指体系结构，它引入了更复杂算法的许多特性，尽管其他体系结构的性能更好。提及配对体系结构基于一个分类器——顾名思义——给出一对提及、一个候选回指词和一个候选先行词，并做出二元分类决策:是否共指(coreferring)。

考虑一下这个分类器在我们的例子中对代词 she 的任务，并假设图 21.2 中略微简化的潜在先行词集。

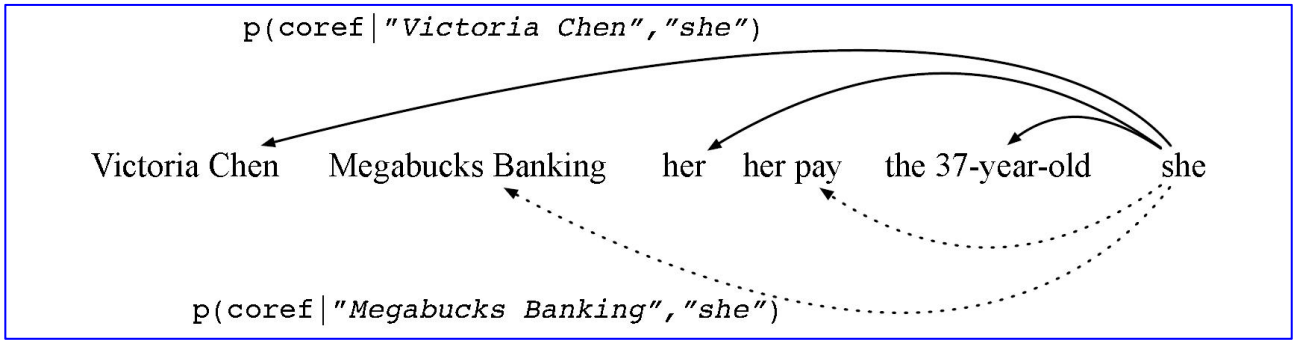


图 21-2：提及配对分类器

图注：对于每对提及(如 she)和一个潜在的先行提及(如 Victoria Chen 或 her)，提及配对分类器分配一个共指连接的概率。

对于每一个先前的提及(Victoria Chen, Megabucks Banking, her，等等)，二元分类器计算一个概率:是否提及是 she 的先行词。我们希望实际先行词(Victoria Chen, her, the 38-year-old)有较高的概率，以及非先行词(Megabucks Banking, her pay)有较低的概率。早期的分类器使用手工构建的特征(第 21.5 节);最近的分类器使用神经表示学习(第 21.6 节)。

对于训练，我们需要一个启发式的方法来选择训练样本;由于文献中的大多数提及配对都是不相关的，选择每一对都会导致大量的负样本过剩。从(Soon 等人，2001)开始，最常见的启发式方法是选择最接近的先行词作为正面例子，而选择介于两者之间的所有成对先行词作为负面例子。更正式地说，对每一个回指提及 m_i，我们创建：

- 一个正例(m_i，m_j)，其中 m_j 是与 m_i 最接近的先行词
- 一个负例(m_i，m_k)，其中每个 m_k 都在 m_j 和 m_i 之间

因此，对于回指词 she，我们会选择(she, her)作为正例而不是负例。同样，对于回指词 the company，我们选择(the company, Megabucks)作为正例，而选择(the company, she) 、(the company, the 38-year-old) 、(the company, her pay) 、和 (the company, her)作为负例。

一旦训练好分类器，就把它应用到聚类步骤中的每个测试句子中。对于文档中每个提及 i，分类器都会考虑先前 i-1 个提及中的每一个提及。在最近优先(closest-first)聚类(Soon 等人，2001)中，分类器从右到左运行(从提及 i-1 到提及 1)，而且概率> 0.5 的第一个先行词被连接到 i。如果无先行词有概率> 0.5，则对 i 来说没有先行词被选择。在最好优先(best-first)聚类方法中，分类器是运行在所有 i-1 个先行词之上的，而且最可能的优先提及被选为 i 的先行词。成对关系的可传递闭包被视为聚类。

尽管提及配对模型具有简单性的优点，但它有两个主要问题。首先，分类器不会直接对候选先行词之间进行比较，因此没有经过训练就可以在两个可能的先行词之间进行决策，判别实际上哪个更好。其次，它忽略了话语模型，只关注提及，而不关注实体。每个分类器的决策制定完全局限于在该配对中，不考虑同一实体的其他提及。接下来的两个模型分别解决了这两个缺陷之一。

21.4.2. 提及排名架构

提及排名模型直接将候选先行词相互比较，为每个回指词选择得分最高的先行词。

在早期的公式中，对于提及 i，分类器会确定先前提及{1，...，i-1}中哪个是先行词(Denis 和 Baldridge，2008)。但是，假设 i 实际上不是回指词，并且没有任何一个先行词都将不被选择？这样的模型将需要在 i 上运行单独的回指性分类器。取而代之的是，更好的方法是与以及单个损失共同学习回指检测和共指(Rahman 和 Ng，2009)。

因此，在现代提及排名系统中，对于第 i 个提及(回指词)，我们有一个关联的随机变量 yi，其范围为值 Y(i)= {1，...，i-1，ϵ }。值ϵ是一个特殊的虚拟提及，表示我没有先行词(即：要么是全新的话语和新共指链的开头，要么是一个非回指词)。

在测试时，对于给定的提及 i，模型在所有先行词(加上ϵ)上计算一个 softmax，给出每个候选先行词(或无)的概率。

图 21.3 给出了单个候选回指词 she 的计算示例。

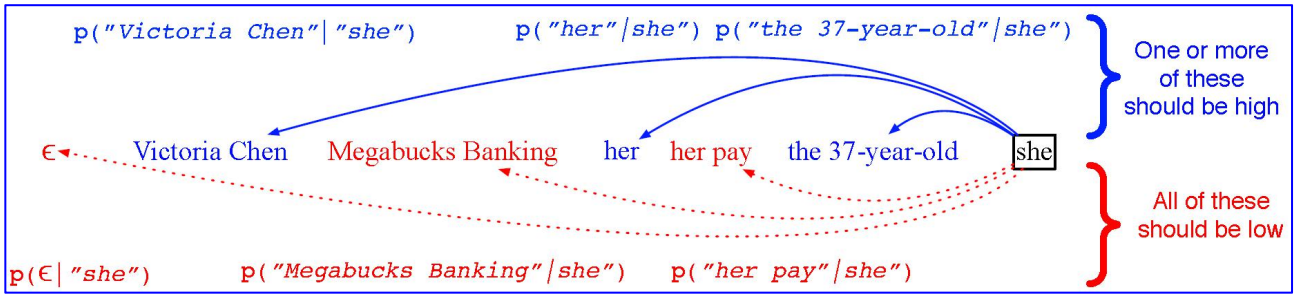


图 21-3：提及排名系统的概率分布

图注：对于每个候选的回指提及(如 she)，提及排名系统对之前所有提及加上特殊的虚拟提及ϵ分配一个概率分布。

一旦为每个回指词分类了先行词，就可以在成对的决策上运行传递闭包，以获得完整的聚类。

在提及排名模型中，训练比提及配对模型更为棘手，因为对于每个回指词，我们都不知道用于训练的所有可能金牌先行词中的哪一个。取而代之的是，每次提及的最佳先行词是潜在的。也就是说，对于每一个提及，我们都有合法的金牌先行词的一个完整聚类可供选择。早期的工作使用启发式方法来选择一个先行词，例如，选择最接近的先行词作为金牌先行词，并在两个句子的窗口中选择所有非先行词作为负例(Denis 和 Baldridge，2008)。存在多种对潜在先行词建模的方法(Fernandes 等人 2012，Chang 等人 2013，Durrett 和 Klein 2013)。最简单的方法是对任何一个先行词都给出一个信用分数(通过对它们全部加总)，其损失函数可以优化在金牌聚类中所有正确的先行词的似然度(Lee 等人，2017)。第 21.6 节有详细信息。

提及排名模型可以用手工构建的特征或神经表示学习(也可能包含一些手工构建的特征)来实现。我们将在第 21.5 节和第 21.6 节中探讨这两个方向。

21.4.3. 基于实体的模型

提及配对模型和提及排名模型都做出有关提及的决定。相比之下，基于实体的模型将每个提及不连接到先前提及，而是连接到先前的话语实体(提及的聚类)。

只需使分类器根据提及聚类而不是单个提及做出决策，就可以将提及排名模型转变为实体排名模型(Rahman 和 Ng，2009)。

对于传统的基于特征的模型，这可以通过在聚类上提取特征来完成。聚类的大小以及其“形状”都是有用的功能，它是聚类中提及类型的序列，即符记(P)合适的、(D)有定、(I)不定、(Pr)代词，所以这样一个由{Victoria, her, the 38-year-old)组成的聚类将具有 P-Pr-D 的形状(Björkelund 和 Kuhn，2014)。包含提及配对分类器的基于实体的模型可以使用提及配对概率的总计作为特征，例如计算两个聚类中所有提及配对的共指平均概率(Clark 和 Manning，2015)。

神经模型可自动学习聚类的表示，例如，通过在聚类提及序列上使用 RNN 来编码与聚类表示相对应的状态(Wiseman 等人，2016)，或者通过学习成对聚类的分布式表示，即对提及配对的习得表示进行池化。

然而，尽管基于实体的模型表达能力更强，但在实践中使用集聚类级别的信息并没有带来很大的性能提升，因此提及排名模型仍然是更常用的。

21.5. 使用手工构建特征的分类器

手工设计的功能在共指中起着重要的作用，无论是作为神经预分类器中分类的唯一输入，还是作为先进神经系统中使用的自动表示学习的增强(就像我们将在第 21.6 节中描述的那样)。在本节中，我们将介绍在逻辑回归，SVM 或随机森林分类器中用于共指解析的常用功能。

给定一个回指词提及和一个潜在的先行词提及，大多数基于特征的分类器都使用三种特征的类型：

(i)回指词的特征，

(ii)候选先行词的特征，

(iii)配对之间关系的特征。

基于实体的模型可以额外使用两个附加类：

(iv)在先行词实体聚类中的所有提及的特征，

(v)在先行词实体聚类中的回指与提及之间关系的特征。

Features of the Anaphor or Antecedent Mention		
First (last) word	Victoria/she	First or last word (or embedding) of antecedent/anaphor
Head word	Victoria/she	Head word (or head embedding) of antecedent/anaphor
Attributes	Sg-F-A-3-PER/Sg-F-A-3-PER	The number, gender, animacy, person, named entity type attributes of (antecedent/anaphor)
Length	2/1	length in words of (antecedent/anaphor)
Grammatical role	Sub/Sub	The grammatical role—subject, direct object, indirect object/PP—of (antecedent/anaphor)
Mention type	P/Pr	Type: (P)roper, (D)efinite, (I)ndefinite, (Pr)onoun) of antecedent/anaphor
Features of the Antecedent Entity		
Entity shape	P-Pr-D	The 'shape' or list of types of the mentions in the antecedent entity (cluster), i.e., sequences of (P)roper, (D)efinite, (I)ndefinite, (Pr)onoun.
Entity attributes	Sg-F-A-3-PER	The number, gender, animacy, person, named entity type attributes of the antecedent entity
Tntecedent cluster size	3	Number of mentions in the antecedent cluster
Features of the Pair of Mentions		
Longer anaphor	F	True of anaphor is longer than antecedent
Pairs of any features	Victoria/she, 2/1, Sub/Sub, P/Pr, etc .	For each individual feature, pair of type of antecedent+ type of anaphor
Fentence distance	1	The number of sentences between antecedent and anaphor
Mention distance	4	The number of mentions between antecedent and anaphor
i-within-I	F	Anaphor has i-within-i relation with antecedent
Cosine		Cosine between antecedent and anaphor embeddings
Appositive	F	True if the anaphor is in the syntactic apposition relation to the antecedent. Useful even if appositives aren't mentions (to know to attach the appositive to a preceding head)
Features of the Pair of Entities		
Exact String Match	F	True if the strings of any two mentions from the antecedent and anaphor clusters are identical.
Head Word Match	F	True if any mentions from antecedent cluster has same headword as any mention in anaphor cluster
Word Inclusion	F	All words in anaphor cluster included in antecedent cluster
Features of the Document		
Genre/source	N	The document genre— (D)ialog, (N)ews, etc,

图 21-4：基于特征的共指算法的一些常见特征

图注：图中包括回指语“she”和潜在先行词“Victoria Chen”的值。

图 21.4 显示了常用特征的选择，并显示了我们示例句中潜在的回指词“she”和潜在先行词“Victoria Chen”需要计算的值，重复如下:

(21.47) **Victoria Chen**, CFO of Megabucks Banking, saw her pay jump to $2.3 million,
as the 38-year-old also became the company’s president.
It is widely known that **she** came to Megabucks from rival Lotsabucks.

先前的工作发现特别有用的特性是精确字符串匹配、实体中心词一致、提及距离，以及(对于代词)精确属性匹配和 i 在 i 内(i-within-i)，以及(对于名词和专有名称)单词包含和余弦。对于词汇特征(如中心词)，通常只使用出现次数足够多的词(可能超过 20 次)，对于罕见的词就退回到词类。

在基于特征的系统中，使用特征的连接是至关重要的;一项实验表明，从分类器中的单个特征移动到多个特征的连接将 F1 增加 4 个点(Lee 等人，2017)。具体的连接可以手工设计(Durrett 和 Klein, 2013)，所有成对的特征都可以被连接(Bengtson 和 Roth, 2008)，或者可以使用决策树或随机森林分类器学习特征连接(Ng 和 Cardie 2002a, Lee 等人 2017)。

最后，其中一些特征也可以用于神经模型。现代神经系统(第 21.6 节)使用上下文词嵌入，所以它们不能从添加诸如字符串或中心词匹配、语法角色或提及类型等浅层特征中受益。然而，像提及长度、提及之间的距离或类型等其他功能可以很好地补充神经上下文嵌入模型。

21.6. 一种神经提及排名算法

在本节中，我们描述 Lee 等人(2017)的神经提及排名系统。这个端到端系统并没有单独的提及检测步骤。取而代之的是，它将可能达到设定长度的所有文本跨度(即长度为 1,2,3 ... N 的所有 n-gram)视为可能提及的内容[8]。

给定一个带有 T 个单词的文档 D，该模型会考虑所有 N = T(T-1)/ 2 个文本跨度都达到某个长度(在 Lee 等人(2018)的版本中，该长度为 10)。每个跨度 i 都从单词 START(i)开始，到单词 END(i)结束。

任务是为每个跨度 i 分配一个先行词 y_i，一个随机变量其值范围为 Y(i)= {1，...，i-1，e}；每个以前的跨度和一个特殊的虚拟符记 e。选择虚拟符记意味着 i 没有先行词，这是因为 i 是全新的话语并开始了一个新的共指链，或者因为 i 是一个非回指词。

对于跨度 i 和 j 的每一对，系统都会为跨度 i 和跨度 j 之间的共指连接分配分数 s(i，j)。然后，系统学习跨度 i 的先行词分布 $P(y_i)$：

$$P(y_i)=\frac{exp(s(i,\ y_i))}{\sum_{y' \in Y(i)} exp(s(i,\ y_i)} \tag{21.48}$$

这个分数 s(i, j)包括三个因素:

m(i)：跨度 i 是否提及;

m(j)：跨度 j 是否提及;

c(i, j)：j 是否为 i 的先行词。

$$s(i, j) = m(i) + m(j) + c(i, j) \tag{21.49}$$

对于虚拟先行词 e，分数 s(i, e)固定为 0。这样，如果任何非虚拟分数是正的，则模型预测出得分最高的前先行词；如果所有分数都是负的，则它弃权。评分函数 m(i)和 c(i, j)是基于一个表示跨度 i 的向量 $\mathbf{g}_i$：

$$m(i) = w_m \cdot FFNN_m(\mathbf{g}_i) \tag{21.50}$$

$$c(i,j) = w_c \cdot FFNN_c([\mathbf{g}_i, \mathbf{g}_j, \mathbf{g}_i \circ \mathbf{g}_j, \Phi(i,j)]) \tag{21.51}$$

先行词分数 c(i, j)作为跨度 i 和 j 的表示的输入，但也作为两个跨度相互之间的逐元素相似性 $\mathbf{g}_i \circ \mathbf{g}_j$(其中∘是逐元素乘法)。先行词分数 c 还考虑了一个特征向量 Φ(i,j)，它编码有用的特征诸如提及距离，以及关于说话者和类型的信息。

可以使用 biLSTM 或 BERT 计算每个跨度 $\mathbf{g}_i$ 的表示形式。

在 biLSTM 版本中，范围 i 的表示 $\mathbf{g}_i$ 是一个拼接，这个拼接由四项内容生成：跨度的起始符记的 biLSTM

[8] 但是由于这种潜在提及的数量使得算法非常缓慢和笨拙(模型的文档长度是 O(t4))，在实践中，各种版本的算法会找到方法来剪除可能的提及，本质上使用提及分数作为提及检测器。

隐藏表示，跨度的结束符记的 biLSTM 隐藏表示，中心词的基于注意力的表示，仅包含一个特征(跨度 i 的长度)的特征向量。

$$g_i = [\mathbf{h}_{START(i)}, \mathbf{h}_{END(i)}, \mathbf{h}_{ATT(i)}, \phi(i)] \tag{21.52}$$

这些计算如下。对于输入的每个单词 w_t，biLSTM 的输出为 h_t：

$$\overrightarrow{\mathbf{h_t}} = \text{LSTM}^{\text{forward}}(\overrightarrow{\mathbf{h_{t-1}}}, \mathbf{w_t})$$

$$\overleftarrow{\mathbf{h_t}} = \text{LSTM}^{\text{forward}}(\overleftarrow{\mathbf{h_{t+1}}}, \mathbf{w_t})$$

$$\mathbf{h_t} = [\overrightarrow{\mathbf{h_t}}, \overleftarrow{\mathbf{h_t}}] \tag{21.53}$$

注意力表示照常被计算;系统学习一个权向量 w_α，用 FFNN 变换的隐藏状态 h_t 计算其点积:

$$\alpha_t = \mathbf{w}_\alpha \cdot \text{FFNN}_\alpha(\mathbf{h_t}) \tag{21.54}$$

注意力得分通过 softmax 归一化为一个分布:

$$\alpha_{i,t} = \frac{exp(\alpha_t)}{\sum_{k=ATART(i)}^{END(i)} exp(\alpha_k)} \tag{21.55}$$

然后，利用注意力分布来创建一个向量 $\mathbf{h}_{ATT(i)}$，它是跨度 i 的单词的注意力加权总和:

$$\mathbf{h}_{ATT(i)} = \sum_{t=ATART(i)}^{END(i)} \alpha_{i,t} \cdot \mathbf{w}_t \tag{21.56}$$

图 21.5 由 Lee 等人(2017)给出了跨度表示和提及分数的 biLSTM 计算。

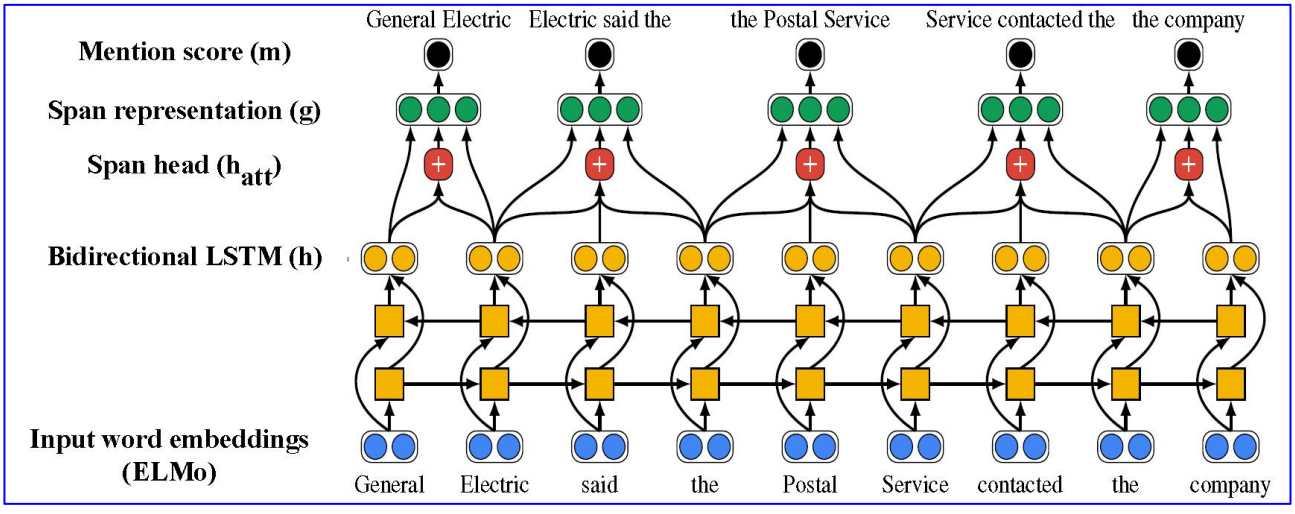


图 21-5：跨度表示和提及分数的计算

图注：计算跨度表示(使用 biLSTM 编码器)和 Lee 等人(2017)端到端共指模型中的提及分数。该模型考虑了所有跨度的最大宽度;图中显示了其中的一小部分。图形根据 Lee 等人(2017)。

在 BERT 版本中，整个 biLSTM 编码器被替换成 BERT。相反，跨度表示 g_i 的计算方法是拼接跨度的第一个和最后一个词片，加上跨度中所有词片的注意力版本(Joshi 等人，2019)。

图 21.6 给出了图 21.5 中的例句中 the company 三种可能的先行词得分 s 的计算结果。

在推断时，通常使用某种方法来修剪提及(例如，使用提及得分 m 作为过滤器，以仅保留最佳的少数提及作为函数，例如句子长度 T 的 0.4T)。然后，以正向方式计算每个文档的先行词的联合分布。最后，我们可以对先行词进行传递闭包，以便为文档创建最终的聚类。

对于训练，每次提及我们都没有一个金牌先行词。取而代之的是，共指标签仅给我们每个共指提及的整个聚类。提及有一个潜在的先行词。因此，我们使用损失函数来最大化任何合法先行词的共指概率之和。对于给定的提及 i 和可能的先行词 Y(i)，令 GOLD(i)为包含 i 的金牌聚类中的提及集合。由于提及的集合发生在 i 之前是 Y(i)，该金牌聚类中也出现在 i 之前的一组提及为 $Y(i) \cap GOLD(i)$。因此，我们希望最大化：

$$\sum_{\hat{y} \in Y(i) \cap GOLD(i)} P(\hat{y}) \tag{21.57}$$

如果提到 i 不在一个金牌聚类中，则 GOLD(i) = ε 。

为了把这个概率变成一个损失函数，我们将使用我们在第 5 章的公式 5.11 中定义的交叉熵损失函数，取其概率的-log。如果我们将所有提及的相加，我们得到训练的最终损失函数:

$$L = \sum_{i=2}^{N} -log \sum_{\hat{y} \in Y(i) \cap GOLD(i)} P(\hat{y}) \tag{21.58}$$

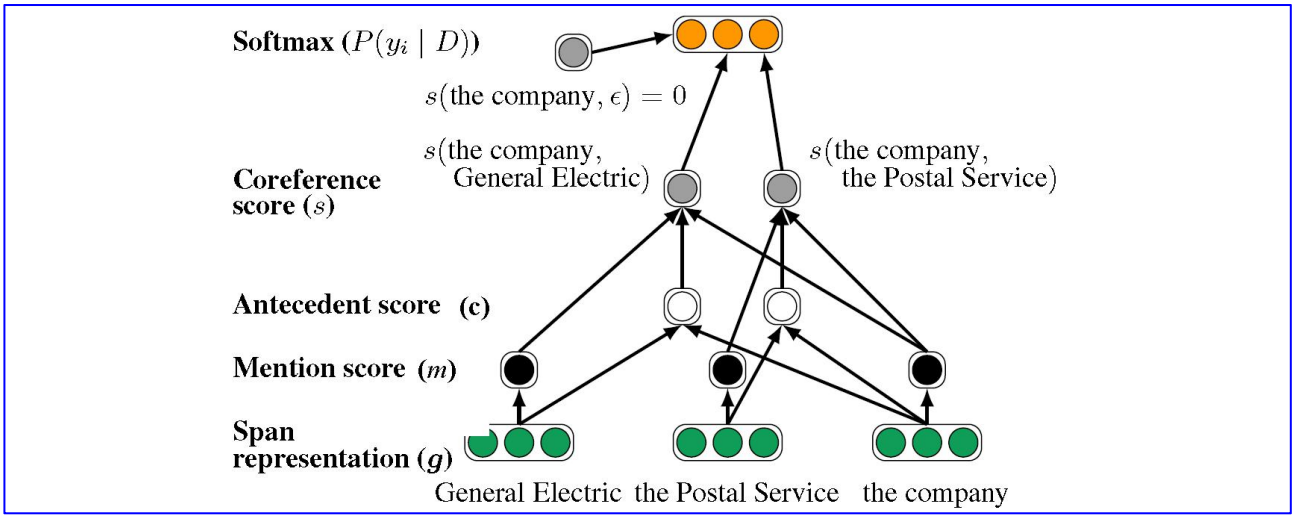


图 21-6：先行词得分的计算

图注：在图 21,5 中的例句中 the company 三种可能的先行词得分 s 的计算，图形根据 Lee 等人(2017)。

图 21.7 显示了模型的预测示例，显示了 Lee 等人(2017)发现的与传统语义中心词相关的注意力权重。请注意，该模型对第二个例子的理解是错误的，这大概是因 attendants 和 pilot 可能有附近的单词嵌入。

We are looking for (a region of central Italy bordering the Adriatic Sea). (The area) is mostly mountainous and includes Mt. Corno, the highest peak of the Apennines. (It) also includes a lot of sheep, good clean-living, healthy sheep, and an Italian entrepreneur has an idea about how to make a little money of them.
(The flight attendants) have until 6:00 today to ratify labor concessions. (The pilots') union and ground crew did so yesterday.

图 21-7：Lee 模型的样本预测

图注：图中每个例子一个聚类，显示一个正确的例子和一个错误的例子。粗体和圆括号的跨度是预测聚类中的提到。单词上红色的数量表示在公式(21.55)中的中心词发现(head-finding)的注意力权重 $a_{i,t}$。图改编自 Lee 等人(2017)。

21.7. 共指消解的评估

我们评估共指算法模型——理论上，将系统产生的一组假设链或聚类 H 与人类标签的一组金牌或指代链或聚类 R 进行比较，并报告精度和召回率。

然而，有各种各样的方法可以进行这种比较。事实上,有 5 个常用度量来评估共指算法：基于连接的 MUC(Vilain 等人，1995)，BLANC(Recasens 和 Hovy 2011，Luo 等人 2014)度量，基于提及的 B3 度量(Bagga 和 Baldwin,1998)，基于实体的 CEAF 度量(Luo，2005)，基于连接的实体察觉 LEA 度量(Moosavi 和 Strube，2016)。

让我们来探讨两个指标。**MUC F 度量(MUC F-measure)**(Vilain 等人，1995)基于 H 和 R 共有的共指连接(成对提及)的数量。精度是共有连接的数量除以 H 中的连接数量。召回率是共有连接数除以 R 中的连接数量；这使 MUC 偏向于产生大型链(且实体较少)的系统，并且忽略了单例，因为它们不涉及连接。

B3 是基于提及而不是基于连接的。对于指代链中的每个提及，我们都计算精度和召回率，然后对文档中所有 N 个提及项进行加权求和，以计算整个任务的精度和召回率。对于给定的提及 i，令 R 为包括 i 的指代链，H 为具有 i 的假设链。H 中正确提及的集合是 H∩R。因此，提及 i 的精度为 |H∩R| / |H|，并且

提及 i 的召回率是 |H∩R| / |R|，总的精度为提及 i 的精度的加权总和，由权重 w_i 加权。总的召回率是提及 i 的召回率的加权总和，由权重 w_i 加权。等效地：

$$\text{Precision} = \sum_{i=1}^{N} w_i \frac{\#\ of\ correct\ mentions\ in\ hypothesis\ chain\ containing\ entity_i}{\#\ of\ mentions\ in\ hypothesis\ chain\ containing\ entity_i}$$

$$\text{Recall} = \sum_{i=1}^{N} w_i \frac{\#\ of\ correct\ mentions\ in\ hypothesis\ chain\ containing\ entity_i}{\#\ of\ mentions\ in\ reference\ chain\ containing\ entity_i}$$

每个实体的权重 wi 可以设置为不同的值，以产生不同版本的算法。

根据 Denis 和 Baldridge(2009)的建议，CoNLL 共指竞赛是根据 MUC、CEAF-e 和 B3 的平均值进行评分的(Pradhan 等人 2011，Pradhan 等人 2012b)，因此在许多评估活动中，报告这 3 个指标的平均值是很常见的。请参阅 Luo 和 Pradhan(2016)了解整个指标集的详细描述;应该使用这些指代的实现，而不是试图从头开始重新实现(Pradhan 等人，2014)。

已经提出了处理特定共指领域或任务的替代度量。例如，考虑解决对命名实体(个人、组织、地缘政治实体)的提及的任务，这可能对信息提取或知识库的完成很有用。假设链正确地包含了一个指代实体的所有代词，但没有名称本身的版本，或者连接了错误的名称，对于此任务没有用。相反，我们可能想要如下两个度量之一：根据提及的信息丰富性(每个人的名字具有最多的信息)对每个提及进行加权的度量(Chen 和 Ng，2013)；仅当一个假设包含名称的至少一个变体时才认为该假设与金牌链匹配的度量(NEC F1 度量)(Agarwal 等人，2019)。

21.8. Winograd 模式问题

从该领域早期开始，研究人员就注意到，一些共指案例相当困难，似乎需要世界知识或复杂的推理才能解决。Winograd(1972)用下面的例子指出了这个问题:

(21.59) The city council denied the demonstrators a permit because
a. they feared violence.
b. they advocated violence.

Winograd 注意到，大多数读者喜欢代词 they 的先行词在(a)中是 the city council，而在(b)中是 the demonstrators。他认为，这需要理解第二个条款是对第一个条款的解释，同时我们的文化框架表明，市议会可能比示威者更害怕暴力，示威者可能更支持暴力。

为了使自然语言处理领域更多地关注涉及世界知识和常识推理的方法，Levesque(2011)提出了一个挑战任务，名为 **Winograd 模式(Winograd Schema)**挑战[9]。挑战任务中的问题是共指问题，该问题被设计成容易被人类读者消除歧义，但希望不能通过简单的技术(如选择限制，或其他基本的词关联方法)来解决。

这些问题被设定为一对陈述，它们在一个单词或短语中存在差异和一个共指问题:

(21.60) The trophy didn't fit into the suitcase because it was too **large**.
Question: What was too **large**? Answer: The trophy

(21.61) The trophy didn't fit into the suitcase because it was too **small**.
Question: What was too **small**? Answer: The suitcase

这些问题具有以下特点:

1.问题各有两个参与方

2.代词优先指代一个参与方，但是在语法上也可以指代另一参与方

3.一个问题询问代词指代的是哪一个参与方

4.如果问题中的一个单词被更改，则人类偏爱的答案将更改为另一参与方

解决问题可能需要的世界知识种类可能会有所不同。在上述的奖杯/手提箱例子中，它是有关物理世界的知识；较大的物体无法容纳较小的物体。在最初的 Winograd 句子中，它是对诸如政治人物和抗议者之类的社会角色的刻板印象。在以下示例中，它的知识是关于人类行为，例如**轮流发言(turn-taking)**或感谢。

[9] Levesque 的征集很快得到了 Levesque 等人(2012)、Rahman 和 Ng(2012)、IJCAI 会议上的一项竞赛(Davis 等人 2017)，以及所谓 WNLI 问题的自然语言推理版本(Wang 等人，2018)。

(21.62) Bill passed the gameboy to John because his turn was [over/next].
Whose turn was [over/next]? Answers: Bill/John

(21.63) Joan made sure to thank Susan for all the help she had [given/received].
Who had [given/received] help? Answers: Susan/Joan.

尽管 Winograd 模式被设计为需要常识推理，但可以通过对 Winograd 模式句子进行微调的预训练语言模型来解决原始问题集合的很大一部分(Kocijan 等人，2019)。大型的预训练语言模型可编码大量的世界常识知识！因此，目前的趋势是提出新的数据集，这些数据集具有越来越难的类似 Winograd 的共指消解问题，如 KNOWREF (Emami 等人，2019)，例如:

(21.64) Marcus is undoubtedly faster than Jarrett right now but in [his] prime the gap wasn't all that big.

最后，语言建模与知识的某种结合似乎会取得丰硕的成果;事实上，基于知识的模型似乎不太适合 Winograd 模式训练集中的词汇特质(Trichelair 等人，2018).

21.9. 共指的性别偏见

就像语言处理的其他方面一样，共指模型也表现出性别和其他偏见(Zhao 等人 2018a,Rudinger 等人 2018, Webster 等人 2018)。

例如，WinoBias 数据集(Zhao 等人，2018a)使用 Winograd Schema 范例的变体来测试共指算法在将性别代词与符合文化刻板印象的先行词联系起来时的偏向程度。正如我们在第 6 章中总结的那样，嵌入在训练测试中复制了社会偏见，例如将男性与具有历史上刻板的男性职业(例如医生)、将女性与具有陈规定型的女性职业(例如秘书)联系起来(Caliskan 等人 2017，Garg 等人 2018)。

WinoBias 句子包含两个提及，分别对应于定型为男性和定型为女性的职业，以及必须与之相关的性别代词。句子不能通过代词的性别来消除歧义，但是一个有偏见的模型可能会被这个线索分心。这是一个示例语句：

(21.65) The secretary called the physician*i* and told him*i* about a new patient [pro-stereotypical]

(21.66) The secretary called the physician*i* and told her*i* about a new patient [anti-stereotypical]

Zhao 等人(2018a)认为，如果关联代词与性别定型的职业相一致(例如，在(21.65)中的 him 和 physician)相比不一致(例如，在(21.66)中的 her 和 physician)的准确性更高，则考虑这个共指系统具有偏见性。他们表明，所有架构(基于规则的、基于特征的机器学习和端到端神经系统)的共指系统都表现出显著的偏见，在反定型的情况下平均表现差 21 个 F1 点。

这种偏见的一个可能原因是，女性实体在用于训练大多数共指系统的 OntoNotes 数据集中代表性不足。Zhao 等人(2018a)提出了一种克服这种偏见的方法：他们生成了第二个性别交换的数据集，其中将 OntoNotes 中的所有男性实体都替换为女性，反之亦然，并使用合并后的原始和交换后的 OntoNotes 数据对共指系统进行重新训练，也使用去除偏见的 GloVE 嵌入(Bolukbasi 等人，2016)。所得的共指系统不再对 WinoBias 数据集表现出偏见，而不会显著影响 OntoNotes 共指精度。在后续论文中，Zhao 等人(2019)表明，在 ELMo 上下文化词向量表示和使用它们的共指系统中也存在相同的偏见。他们表明，通过数据增强对 ELMo 进行再训练可以再次减少或消除 WinoBias 共指系统中的偏见。

Webster 等人(2018)引入了另一个数据集 GAP，以及性别代词消解的任务，该任务是为性别代词开发改进的共指算法的工具。GAP 是一个带有性别歧义代词的 4,454 个句子的性别平衡标签语料库(相比之下，英语 OntoNotes 训练数据中只有 20％的性别代名词是女性的)。这些案例是通过借鉴 Wikipedia 页面上自然出现的句子而创建的，以创建难以解决的案例，该案例具有两个相同性别的命名实体和一个可能指代这两个人中的一个(或两个都不是)的歧义代词，如下所示：

(21.67) In May, Fujisawa joined Mari Motohashi' s rink as the team's skip,
moving back from Karuizawa to Kitami where **she** had spent her junior days.

Webste 等人(2018)表明，在 GAP 中，现代共指算法在解析女性代词上的表现显著逊于男性代词。Kurita 等人(2019)表明，基于 BERT 语境化单词的表示显示了类似的偏见。

21.10. 总结

本章介绍了**共指消解**的任务。

•这是一项将文本中**所指对象**和**提及**连接在一起的任务，即在**话语模型**中指代同一**话语实体**，从而形成一组**共指链**(也称为**聚类**或**实体**)。

•提及可以是**有定** NP 或**不定** NP，代词(包括**零代词**)或名称。

•实体提及的表面形式与其**信息状态**(新、旧或可推断)以及实体的**可访问性**或**显著性**如何相关联。

•一些 NP 不是**指示语**，例如在 it is raining 中使用**赘述词** it。

•许多语料库有人类标签的共指注释，可以用于监督学习，包括：**OntoNotes** 为英语、汉语、Arabic 语；ARRAU 为英语；**AnCora** 为西班牙语和加泰罗尼亚语。

•提及检测可从所有名词和命名实体开始，再使用**回指性**分类器或**指称性**分类器过滤掉未提及的内容。

•三种常见的共指架构是**提及配对**、**提及排名**和**基于实体**，每一种都可以利用基于特征或神经分类器。

•现代的共指系统倾向于端到端，在单一的端到端架构中执行提及检测和共指。

•算法学习文本跨度和中心词的表示，并学习将回指词跨度与候选先行词跨度进行比较。

•共指系统的评估是通过使用精度/召回率指标如 MUC、B3、CEAF、BLANC 或 LEA 与金牌实体标签进行比较。

•**Winograd 模式**挑战问题是困难的共指问题，此问题似乎需要世界知识或复杂的推理来解决。

•共指系统显示**性别偏见**，该偏见可以使用数据集诸如 Winobias 和 GAP 进行评估。

21.11. 文献和历史说明

自 1970 年代以来，共指一直是自然语言理解的一部分(Woods 等人 1972； Winograd 1972)。话语模型和共指的实体中心基础由 Karttunen(1969)(在第三届 COLING 会议上)提出，在语言语义学中也起着作用(Heim 1982，Kamp 1981)。但是，Bonnie Webber(1978)的论文和后续工作(Webber,1983)探索了该模型的计算方面，从而提供了对实体在话语模型中如何表示以及它们如何许可后续指代的方式的基本见解。她提供的许多例子继续挑战今天的参考理论。

Hobbs 算法是一种树搜索算法[10]，它是基于句法的长 Hobbs 算法系列中的第一个树形搜索算法，用于在自然出现的文本中稳健地标识指代。Hobbs 算法的输入是要解析的代词，连同直到和包含当前句子的的句法(成分)解析。该算法的细节取决于所使用的语法，但是由于 Kehler 等人(2004 年)的观点，可以从简化版本中了解。只是搜索当前句子和先前句子中的 NP 列表。这种简化的 Hobbs 算法按以下顺序搜索 NP：（i）在当前句子中从右到左，从代词左边的第一个 NP 开始；(ii)在前一句中从左至右；(iii)从左至右先于两个句子；(iv)在当前句子中从左到右，从代词右边的第一个名词组开始(用于加泰罗尼亚语)。在数量、性别和人称方面与代词相符的第一个名词组被选为先行词(Kehler 等人，2004)。

Lappin 和 Leass(1994)是一个有影响力的基于实体的系统，它使用权重来成分语法和其他功能，此后不久，Kennedy 和 Boguraev(1996)对其进行了扩展，该系统无需进行完整的语法分析。大约同一时期，Brennan 等人(1987)将中心化(Grosz 等人，1995)应用于代词回指的消解。随后进行了各种各样的工作，聚焦于中心化用于共指(Kameyama 1986，Di Eugenio 1990，Walker 等人 1994，Di Eugenio 1996，Strube 和 Hahn1996，Kehler1997a，Tetreault 2001，Iida 等人 2003)。Kehler 和 Rohde(2013)展示了如何将中心化与连贯性驱动的代词解释理论整合在一起。参见第 22 章，了解如何使用中心化来衡量话语连贯。

共指竞赛的一部分 US DARPA-sponsored MUC 会议提供早期标签共指数据集(1995 MUC-6 和 1998 MUC-7 语料库),为大部分以后的工作定下了基调,将社区吸引到有监督的机器学习和度量诸如 MUC 度量(Vilain 等人 1995)。后来的 ACE 评估产生了英语、中文和阿拉伯语标签的共指语料库，这些语料被广泛用于模型训练和评估。

这项 DARPA 的工作影响了社区从 90 年代中期开始的监督学习(Connolly 等人 1994；Aone 和 Bennett 1995； McCarthy 和 Lehnert 1995)。Soon 等人(2001)提出了一系列基本特征，并由 Ng 和 Cardie(2002b)

[10] 这是 Hobbs(1978)提出的两个算法中最简单的一个。

进行了扩展，并在接下来的 15 年中遵循了一系列机器学习模型。这些通常分别聚焦于代词回指的消解(Kehler 等人 2004，Bergsma 和 Lin 2006)，完整的 NP 共指(Cardie 和 Wagstaff 1999，Ng 和 Cardie 2002b，Ng 2005a)和有定 NP 的指代(Poesio 和 Vieira 1998，Vieira 和 Poesio 2000)以及分别的回指检测(Bean 和 Riloff 1999，Bean 和 Riloff 2004，Ng 和 Cardie 2002a，Ng 2004)或单例检测(de Marneffe 等人 2015)。

从成对提及到成对提及排名方法的转变是由 Yang 等人(2003)和 Iida 等人(2003)首创的，他们提出了成对提及排名方法，然后由 Denis 和 Baldridge(2008)扩展，他们提出通过 softmax 对所有先前提及的内容进行排名。当时有一个主意：在一个端到端模型中联合进行提及检测、回指和共指。这个主意源自 Ng(2005b)的早期建议，就是使用虚拟的先行词进行提及排名，允许“无所指”成为如下三个内容的一个选择：共指分类器、Denis 和 Baldridge(2007)的联合系统(把回指分类器概率与共指概率相结合)、Denis 和 Baldridge(2008)的排名模型。Rahman 和 Ng(2009)建议在单一目标下联合训练这两个模型。

简单的基于规则的共指系统在 2010 年代重新流行，部分原因是它们能够以高精度的方式对基于实体的特征进行编码(Zhou 等人 2004； Haghighi 和 Klein 2009； Raghunathan 等人 2010； Lee 等人 2011，Lee 等人 2013，Hajishirzi 等人 2013)，但是最终他们无能力去处理语义需求，即正确地处理常见名词共指的情况。

回到有监督的学习导致提及排名模型的许多进步，这些模型也扩展到了神经体系结构中，例如，使用强化学习直接优化共指评估模型(Clark 和 Manning 2016a)，进行了端到端共指跨度提取的方式(Lee 等人 2017，Zhang 等人 2018)。神经模型也被设计用来利用全局实体级别信息(Clark 和 Manning 2016b，Wiseman 等人 2016，Lee 等人 2018)。

共指也与第 23 章中讨论的实体连接任务有关。通过提供更多可能的表面形式来帮助连接到正确的 Wikipedia 页，共指可帮助实体连接，反之，实体连接可帮助提高共指消解。Hajishirzi 等人(2013)举例：

(21.68) [Michael Eisner]$_1$ and [Donald Tsang]$_2$ announced the grand opening of [[Hong Kong]$_3$ Disneyland]$_4$ yesterday. [Eisner]$_1$ thanked [the President]$_2$ and welcomed [fans]$_5$ to [the park]$_4$.

将实体连接整合到共指中可以帮助汲取百科全书知识(例如 Donald Tsang 是总统的事实)，以帮助消除对总统的提及的歧义。Ponzetto 和 Strube(2006)(2007)以及 Ratinov 和 Roth(2012)表明，从 Wikipedia 页面提取的此类属性可用于建立更丰富的共指实体提及模型。最近的研究显示了如何共同进行连接和共指联合(Hajishirzi 等人 2013，Zheng 等人 2013)，甚至还可以与命名实体标记一起进行连接(Durrett 和 Klein，2014)。

正如我们介绍的那样，共指任务涉及一个简化的假设，即回指和其先行词之间的关系是同一身份：两个共指提及是指代相同的话语指称对象。在真实文本中，这种关系可能会更加复杂，其中话语对象的不同方面可能会被抵消或重新聚焦。例子(21.69)(Recasens 等人，2011)显示了一个**隐喻(metonymy)**的示例，其中首都 Washington 通过隐喻被用来指代 US(美国)。(21.70-21.71)显示了其他示例(Recasens 等人，2011)：

(21.69) a strict interpretation of a policy requires **The U.S.** to notify foreign dictators of certain coup plots ... **Washington** rejected the bid ...

(21.70) I once crossed that border into Ashgh-Abad on Nowruz, the Persian New Year. In the South, everyone was celebrating **New Year**; to the North, **it** was a regular day.

(21.71) In France, **the president** is elected for a term of seven years, while in the United States **he** is elected for a term of four years.

有关这些共指复杂性的进一步语言学讨论，请参见 Pustejovsky(1991)，van Deemter 和 Kibble(2000)，Poesio 等人(2006)，Fauconnier 和 Turner(2008)，Versley(2008)和 Barker(2010)。

Ng(2017)提供了一个有用的关于机器学习模型在共指消解中的紧凑历史。有三种优秀的关于回指/回指消解的像书籍那么长的概述，涵盖了不同的时期：Hirst(1981)(早期工作到 1981 年左右)，Mitkov(2002)(1986-2001)和 Poesio 等人(2016)(2001-2015)。

本教材 2000 年第一版的话语章节是 Andy Kehler 写的，我们把它作为第二版章节的起点，在第三版的共指章节中还保留了 Andy 的一些可爱的散文。

21.12. 练习

(无)

22. 话语连贯

> And even in our wildest and most wandering reveries, nay in our very dreams, we shall find, if we reflect, that the imagination ran not altogether at adventures, but that there was still a connection upheld among the different ideas, which succeeded each other. Were the loosest and freest conversation to be transcribed, there would immediately be transcribed, there would immediately be observed something which connected it in all its transitions.
>
> David Hume, *An enquiry concerning human understanding*, 1748

奥森·威尔斯(Orson Welles)的电影《公民凯恩(Citizen Kane)》在许多方面都具有开创性，也许在结构上最为显著。虚构媒体大亨查尔斯·福斯特·凯恩(Charles Foster Kane)的生平故事，这部电影并没有按时间顺序贯穿凯恩的一生。取而代之的是，这部电影以凯恩(Kane)的去世(著名的抱怨的“玫瑰花蕾”)开始，围绕着他一生的倒叙构成，插在记者调查他的死亡的场景中。电影的结构不必线性地遵循真实时间轴的结构这一新颖的观念，对于 20 世纪的摄影术来说，显然是各种类型的连贯叙事结构的无限可能和影响。

但是连贯的结构不仅仅是关于电影或艺术品的事实。像电影一样，语言通常不是由孤立的、不相关的句子组成，而是由并置、结构化、连贯的句子组组成。我们将这样一个连贯的结构化句子组称为**话语(discourse)**，并且我们使用**连贯(coherence)**一词来指代句子之间的关系，此关系使真实的话语不同于随机的句子组合。您现在正在阅读的一章是一个话语示例，例如新闻报道、对话、社交媒体上的话题、维基百科页面以及您喜欢的小说。

是什么使得话语连贯？ 如果您是通过从许多不同来源中随机抽取每个句子来创建文本并将它们粘贴在一起的，那么这将是一种连贯的话语吗？ 几乎可以肯定。真实的话语既表现出局部的连贯性，又表现出整体的连贯性。让我们考虑一下真实话语在局部上连贯的三种方式.

首先，真实话语中的句子或从句以系统的方式与附近的句子相关。考虑 Hobbs(1979)的例子：

(22.1) John took a train from Paris to Istanbul. He likes spinach.

这个顺序是不连贯的，因为读者不清楚为什么第二句排在第一句之后;喜欢吃菠菜和火车旅行有什么关系?事实上，读者可能会努力去弄清楚话语是如何连贯的;也许是法国菠菜短缺?听众试图识别这种联系的事实表明，人类的话语理解需要建立这种连贯性。相比之下，在以下连贯的例子中:

(22.2) Jane took a train from Paris to Istanbul. She had to attend a conference.

第二句在第一句中给出了 Jane 行为的 REASON。话语单元之间的结构化关系 REASON 被称为**连贯关系(coherence relations)**，连贯的话语是由许多这样的连贯关系构成的。第 22.1 节介绍了连贯关系。

第二种方法是通过“关于(about)”某人或某事的优点，使话语具有局部连贯。在一个连贯的话语中，一些实体是突出的，话语聚焦于它们，而不是在多个实体之间来回。这被称为基于实体的连贯。考虑以下不连贯的段落，其中显著的实体似乎从约翰到珍妮，再到钢琴店，再到客厅，再回到珍妮，然后又是钢琴:

(22.3) John wanted to buy a piano for his living room.
Jenny also wanted to buy a piano.
He went to the piano store.
It was nearby.
The living room was on the second floor.
She didn’t find anything she liked.
The piano he bought was hard to get up to that floor.

基于实体的连贯模型通过追踪话语中的显著实体来衡量这类连贯。例如**中心理论(Centering Theory)**(Grosz 等人,1995)，最具影响力的实体连贯性理论，跟踪了话语模型中哪些实体在任何时刻都是突出的(突出的实体更有可能被名词化，或出现在突出的句法位置，如主语或宾语)。在中心理论中，保持相同突出实体的句子之间的转换被认为比重复转换实体之间的句子更连贯。连贯的**实体网格(entity grid)**模型(Barzilay 和 Lapata, 2008)是一个常用的模型，它实现了中心理论框架的一些灵感。基于实体的连贯将在 22.3 节介绍。

最后，通过主题连贯(topically coherent)，话语可以局部连贯:附近的句子通常是关于同一个主题，使用相同或相似的词汇来讨论这些主题。由于主题连贯的话语来自单一的语义场或主题，它们往往表现出词汇连贯(lexical cohesion)的表面特性(Halliday 和 Hasan，1976):在邻近的句子中共享相同或语义相关的词。例如，单词 house, chimney, garret, closet, 和 window 都属于同一个语义场，它们出现在(22.4)中的两个句子中，或者它们共用一个单词 shingled，都提示着这两个词在话语中联系在一起:

(22.4) Before winter I built a **chimney**, and shingled the sides of my **house**...
I have thus a tight shingled and plastered **house**...
with a **garret** and a **closet**, a large **window** on each side....

除了相邻或邻近句子之间的局部连贯外，话语也表现出全局连贯。许多话语体裁都与特定的传统话语结构有关。学术文章可能有描述方法或结果的部分。故事可能会遵循传统的情节主线或主题。有说服力的文章有一个特别的主张，他们试图争辩，一篇文章可能表达这个主张，与一套结构化的前提，支持论点和推翻潜在的相反论点。我们将介绍每一种全局连贯的版本。

我们为什么要关注话语的局部或全局连贯性？ 由于连贯性是写得好的文本的属性，因此连贯性检测在需要测量文本质量的任何任务中都扮演着重要的角色。例如，连贯性可以帮助完成诸如论文分级或论文质量测量等教学任务，这些工作试图对一篇人类论文的撰写水平进行分级(Somasundaran 等人 2014，Feng 等人 2014，Lai 和 Tetreault 2018)。连贯性也有助于总结；了解句子之间的连贯关系有助于了解如何从句子中选择信息。最后，检测不连贯的文本甚至可能在心理健康任务中起作用，例如测量精神分裂症或其他类型的语言紊乱的症状(Ditman 和 Kuperberg 2010，Elvevag 等人 2007，Bedi 等人 2015，Iter 等人 2018)。

22.1. 连贯关系

回顾一下引言(22.5)和(22.6)段落之间的区别。

(22.5) Jane took a train from Paris to Istanbul. She likes spinach.

(22.6) Jane took a train from Paris to Istanbul. She had to attend a conference.

在(22.6)中更连贯的原因是读者可以在两句话之间形成连接，其中第二句话为第一句提供了一个潜在的 REASON(原因)。这个连接很难在(22.5)中形成。这些话语之间的联系可以被定义为一组**连贯关系(coherence relations)**。接下来的两个章节描述了两种常用的连贯关系和相关语料库模型:**修辞结构理论(Rhetorical Structure Theory，RST)**和 Penn 话语树库(PDTB)。

22.1.1. 修辞结构理论(RST)

最常用的话语组织模型是**修辞结构理论(RST)** (Mann 和 Thompson, 1987)。在 RST 中，关系定义在两个文本跨度之间，通常是一个核心和一个卫星。核心是一个更集中于作者意图的单位，可以独立解释;卫星是更少的中心度，一般只能相对于核心的有关方面进行解释。然而，两个原子核之间存在对称关系。

下面是一些 RST 连贯关系的例子，其定义改编自 RST 树库手册(Carlson 和 Marcu, 2001)。

原因:核心是一种活跃的施事者进行的活动，卫星是核心存在的理由。

(22.7) [NUC Jane took a train from Paris to Istanbul.] [SAT She had to attend a conference.]

细化:卫星提供关于核心情况的额外信息或细节

(22.8) [NUC Dorothy was from Kansas.] [SAT She lived in the midst of the great Kansas prairies.]

证据:卫星提供关于核心情况的额外信息或细节。信息呈现的目的是说服读者接受核心中呈现的信息。

(22.9) [NUC Kevin must be here.] [SAT His car is parked outside.]

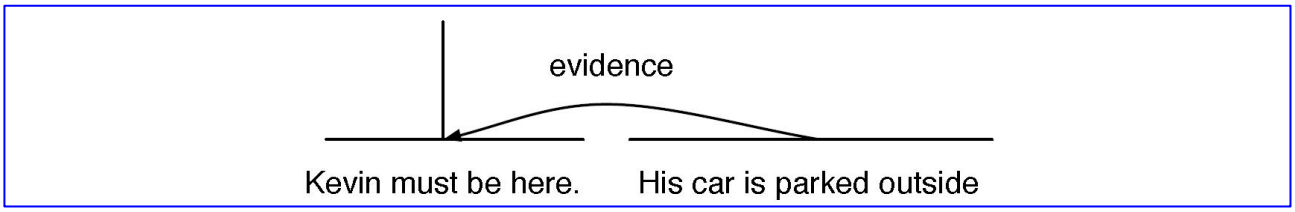
归因：卫星为核心中报告的语音实例提供了归因来源。

(22.10) [SAT Analysts estimated] [NUC that sales at U.S. stores declined in the quarter, too]

列表：在这个多核关系中，给出了一系列核心，没有对比或明确的比较:

(22.11) [NUC Billy Bones was the mate;] [NUC Long John, he was quartermaster]

RST 关系传统上用图形表示;非对称得核心-卫星关系用从卫星到核心的箭头来表示:



我们还可以通过考虑连贯关系之间的层次结构来讨论较大文本的连贯性。图 22.1 显示了(22.12)中 Marcu(2000a)的一段的修辞结构(来自《科学美国人》杂志)。

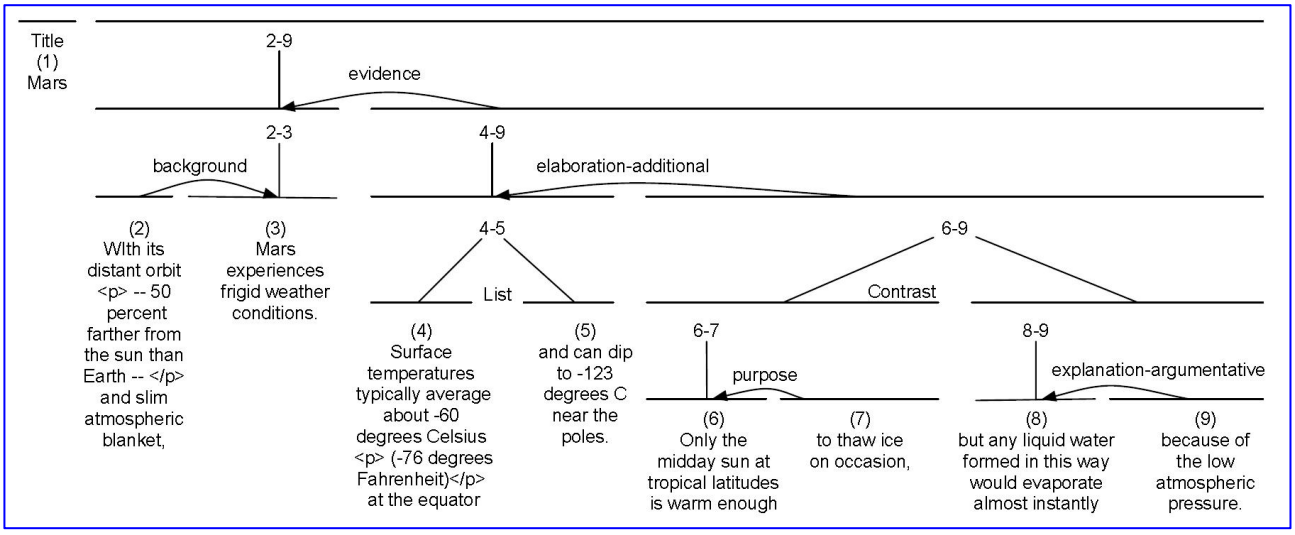


图 22-1：一棵话语树

图注：《科学美国人》(22.12)文本的话语树，来自 Marcu(2000a)。注意，不对称关系用一个从卫星到核心的曲线箭头表示。

图 22.1 树中的叶子对应于在 RST 中被称为**基本话语单元(elementary discourse unit，EDU)**的句子、从句或短语的文本跨度；这些单元也可以称为语段。由于这些单位可能对应于任意文本范围，因此确定 EDU 的边界是提取连贯关系的重要任务。粗略地说，人们可以认为语段与句从句法中的成分相似，实际上，正如我们将在第 22.2 节中看到的那样，我们通常使用解析算法来推理话语结构。

有许多话语连贯模型的语料库。RST 话语树库(Carlson 等人，2001)是最大的可用语料库。它由从 Penn 树库选出的 385 个英语文档组成，每个文档都具有完整的 RST 解析，使用了 78 个不同的关系集，分为 16 类。RST 树库也存在于西班牙文、德文、巴斯克文、荷兰文和巴西葡萄牙文(Braud 等人，2017)。

现在，我们已经看到了连贯的示例，我们可以更清楚地看到连贯关系如何在摘要或信息提取中发挥作用。例如，一个文本的核心大概表示比卫星更重要的信息，而这些信息可能会在摘要中删除。

22.1.2. Penn 话语树库（PDTB）

滨州话语树库(Penn Discourse TreeBank，PDTB)是第二个常用的数据集，它体现了另一种连贯关系模型(Miltsakaki 等人 2004；Prasad 等人 2008；Prasad 等人 2014)。PDTB 标签是基于词汇的。不是要求注释者直接标记文本范围之间的连贯关系，而是给他们提供了**话语连接词(discourse connectives)**的列表，这些词表示话语关系，例如，because, although, when, since, 或 as a result。在文本的一部分中，这些单词标记了两个文本跨度之间的连贯关系的，然后注释了连接词和跨度，如图 22.13 所示，其中的 as a result 短语表示在 PDTB 中称之为 Arg1(前两个句子，这里以斜体显示)和 Arg2(第三个句子，这里以粗体显示)之间的因果关系。

(22.13) *Jewelry displays in department stores were often cluttered and uninspired. And the merchandise was, well, fake.* **As a result, marketers of faux gems steadily lost space in department stores to more fashionable rivals—cosmetics makers.**

(22.14) *In July, the Environmental Protection Agency imposed a gradual ban on virtually all uses of asbestos.* (implicit=as a result) **By 1997, almost all remaining uses of cancer-causing asbestos will be outlawed.**

并不是所有的连贯关系都被显式话语连接词所标记，因此 PDTB 也在没有显式信号的情况下注释成对的相邻句子诸如(22.14)。注释者首先选择可能是其信号的单词或短语(在本例中是 as a result)，然后对其含义进行标记。例如，对于有歧义的话语连接词 since，注释者标记它是使用 CAUSAL(因果) 含义还是 TEMPORAL(时间)含义。

最终的数据集大约包含 18000 个显式关系和 16000 个隐式关系。图 22.2 显示了 4 个主要语义类的例子，而图 22.3 显示了完整的标签集。

Class	Type	Example
TEMPORAL	SYNCHRONOUS	The parishioners of St. Michael and All Angels stop to chat at the church door, as members here always have. (Implicit while) In the tower, five men and women pull rhythmically on ropes attached to the same five bells that first sounded here in 1614.
CONTINGENCY	REASON	Also unlike Mr. Ruder, Mr. Breeden appears to be in a position to get somewhere with his agenda. (implicit=because) **As a former White House aide who worked closely with Congress, he is savvy in the ways of Washington.**
COMPARISON	CONTRAST	The U.S. wants the removal of what it perceives as barriers to investment; Japan denies there are real barriers.
EXPANSION	CONJUNCTION	Not only do the actors stand outside their characters and make it clear they are at odds with them, but they often literally stand on their heads.

图 22-2：PDTB 含义层次中的四个高层语义区分

Temporal	Comparison
• Asynchronous	• Contrast (Juxtaposition, Opposition)
• Synchronous (Precedence, Succession)	• Pragmatic Contrast (Juxtaposition, Opposition)
	• Concession (Expectation, Contra-expectation)
	• Pragmatic Concession
Contingency	**Expansion**
• Cause (Reason, Result)	• Exception
• Pragmatic Cause (Justification)	• Instantiation
• Condition (Hypothetical, General, Unreal Present/Past, Factual Present/Past)	• Restatement (Specification, Equivalence, Generalization)
• Pragmatic Condition (Relevance, Implicit Assertion)	• Alternative (Conjunction, Disjunction, Chosen Alternative)
	• List

图 22-3：PDTB 含义层次结构

图注：有 4 个顶层类、16 个类型和 23 个子类型(不是所有类型都有子类型)。16 种类型中有 11 种通常用于隐式论元分类；斜体中的这 5 种类型在隐式标记中太罕见了以致不能使用。

不像 RST 话语树库，该树库将这些成对的连贯关系整合成一个跨越整个话语的全局树结构，PDTB 不标注超出成对跨度层次的任何东西，对更高层次的话语结构不做任何承诺。

也有其他语言使用类似方法的树库;(22.15)给出了一个来自中文话语树库的例子(Zhou 和 Xue, 2015)。由于汉语的显式话语连接词所占比例低于英语(所有话语关系中只有 22%使用显式连接词，而英语为 47%)，注释者对该语料库进行标注时，不从词汇话语连接词开始，而是直接将成对的句子映射到 11 个含义标签。

(22.15) [**Conn** 为] [**Arg2** 推动图们江地区开发] ，[Arg1 韩国捐款一百万美元设立了图们江发展基金]

“[In order to] [**Arg2** promote the development of the Tumen River region],

[**Arg1** South Korea donated one million dollars to establish the Tumen River Development Fund].”

这些话语树库已被用于多语言话语分析的共享任务(Xue 等人，2016)。

22.2. 话语结构解析

给定一个句子序列，我们如何自动确定它们之间的连贯关系?这个任务通常称为话语解析(即使对于PDTB，我们只是将标签分配给叶子跨度，而不是像对于 RST 那样构建一个完整的解析树)。

22.2.1. 用于 RST 解析的 EDU 分段

RST 解析一般分为两个阶段。第一个阶段，EDU 分段，提取每个 EDU 的开始和结束。这一阶段的输出将是一个如下所示的标签:

(22.16) [Mr. Rambo says]$_{e1}$ [that a 3.2-acre property]$_{e2}$ [overlooking the San Fernando Valley]$_{e3}$ [is priced at $4 million]$_{e4}$ [because the late actor Erroll Flynn once lived there.]$_{e5}$

由于 EDU 大致相当于从句，早期的 EDU 分段模型首先运行语法解析器，然后对输出进行后处理。现代系统一般在 RST 话语树库中使用金牌 EDU 分段监督的神经序列模型。图 22.4 显示了根据 Wang 等人(2018)和 Muller 等人(2019)的一个监督体系结构的示例，该体系结构使用了我们看到的用于命名实体标记和语义角色标签的 biLSTM-CRF 体系结构。在这里，输入句子通过一个编码器传递，然后通过顶部有 CRF 层的 biLSTM 产生一个 0 和 1 的序列，其中 1 表示 EDU 的开始(除了句子的开始)。

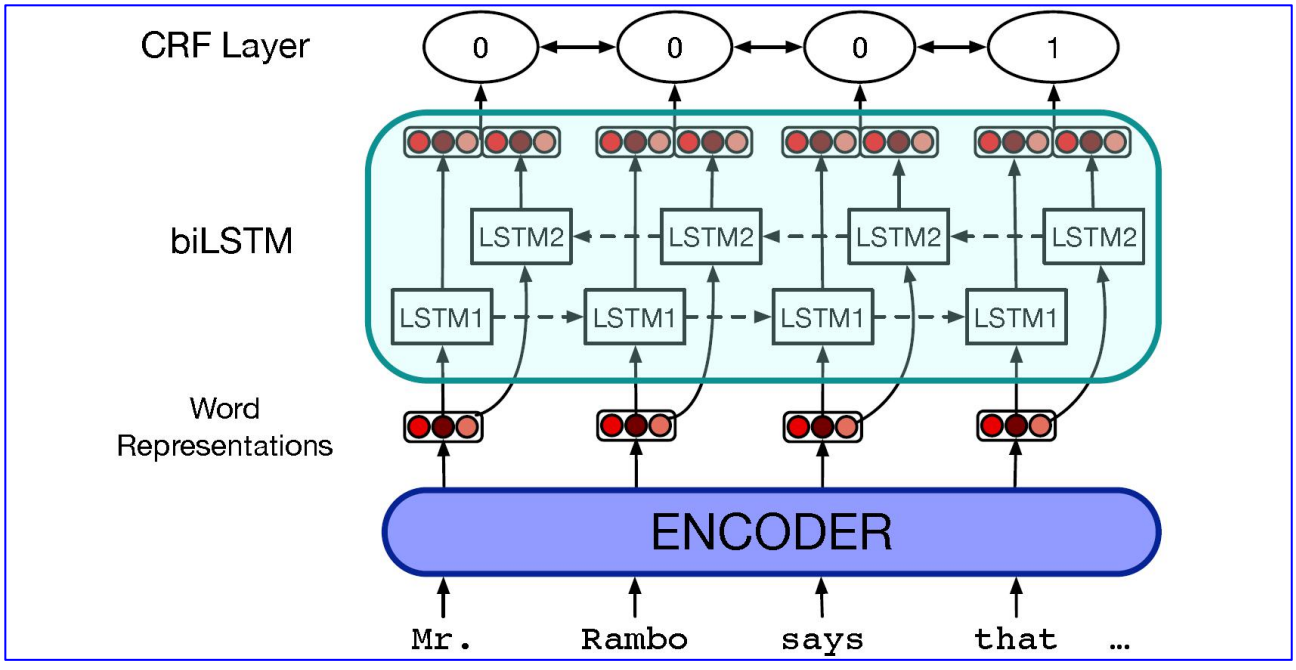


图 22-4：biLSTM-CRF 用于 EDU 分段

图注：对于像 BERT 这样的上下文嵌入，单词输入可以从任何编码器中提取。

22.2.2. RTS 解析

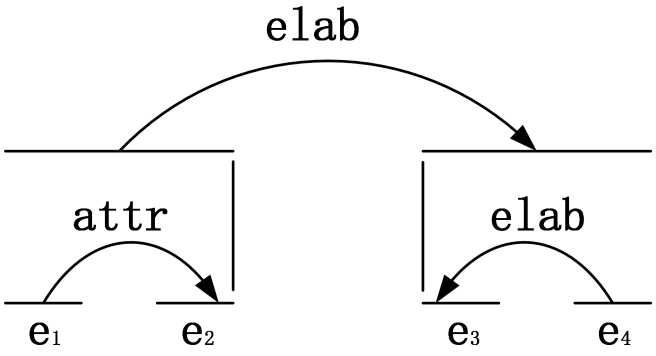
长期以来，为话语构建 RST 连贯结构的工具都是基于移位-减少(shift-reduce)句法分析等句法分析算法(Marcu, 1999)。许多现代 RST 解析器因为 Ji 和 Eisenstein(2014)利用神经语法解析器，在第 14.4 节中我们看到,使用表示学习构建每个跨度的表示,而且训练一个解析器以选择正确的、基于训练集的黄金解析的移位和减少操作。

我们将描述 Yu 等人(2018)的移位-减少解析器。解析器状态由一个堆栈和一个队列组成，并通过对这些状态采取一系列操作来生成这种结构。操作包括：

•shift：将队列中的第一个 EDU 推送到堆栈上，创建一个单节点的子树。

•reduce(l,d)：合并栈顶的两个子树，其中 l 为共指关系标签，d 为核心方向，d∈{NN,NS,SN}。以及弹出根的操作，以从堆栈中删除最终的树。

图 22.6 显示了解析器在构建图 22.5 中的结构时所采取的操作。



e_1: American Telephone & Telegraph Co. said it
e_2: will lay off 75 to 85 technicians here , effective Nov. 1.
e_3: The workers install , maintain and repair its private branch exchanges,
e_4: which are large intracompany telephone networks.

图 22-5：RST 话语树显示四种 EDU 的例子

Step	Stack	Queue	Action	Relation
1	$\varnothing$	e_1, e_2, e_3, e_4	SH	$\varnothing$
2	e_1	e_2, e_3, e_4	SH	$\varnothing$
3	e_1, e_2	e_3, e_4	RD(attr, SN)	$\varnothing$
4	$e_{1:2}$	e_3, e_4	SH	$\widehat{e_1 \mathbf{e_2}}$
5	$e_{1:2}, e_3$	e_4	SH	$\widehat{e_1 \mathbf{e_2}}$
6	$e_{1:2}, e_3, e_4$	$\varnothing$	RD(elab, NS)	$\widehat{e_1 \mathbf{e_2}}$
7	$e_{1:2}, e_{3:4}$	$\varnothing$	RD(elab, SN)	$\widehat{e_1 \mathbf{e_2}}, \widehat{\mathbf{e_3} e_4}$
8	$e_{1:4}$	$\varnothing$	PR	$\widehat{e_1 \mathbf{e_2}}, \widehat{\mathbf{e_3} e_4}, \widehat{e_{1:2} \mathbf{e_{3:4}}}$

图 22-6：使用 shift-reduce 解析器解析图 22.5 的例子

Yu 等人(2018)使用编码器-解码器架构，其中编码器使用分层的 biLSTM 表示单词和 EDU 的输入跨度。第一 biLSTM 层表示 EDU 内部的单词， 第二层代表 EDU 序列。给定输入句子 w_1，w_2，...，w_m，这些单词可以照常表示(通过静态嵌入，与字符嵌入或标记的组合或上下文嵌入)，从而得到输入单词表示序列x_1^w，x_2^w，.. x_n^w 。单词级 biLSTM 的结果是一系列 h^w 值：

$$\mathbf{h}_1^w, \mathbf{h}_2^w, ..., \mathbf{h}_m^w = \text{biLSTM}(\mathbf{x}_1^w, \mathbf{x}_2^w, ..., \mathbf{x}_m^w) \tag{22.17}$$

然后，跨度为 w_s，w_{s+1} ... w_t 的 EDU 的 biLSTM 输出表示为h_s^w，h_{s+1}^w，..，h_t^w，并由平均池化来表示：

$$\mathbf{x}^e = \frac{1}{t-s+1} \sum_{k=s}^{t} \mathbf{h}_k^w \tag{22.18}$$

第二层使用这个输入来计算 EDU 表示序列 h^e 的最终表示:

$$\mathbf{h}_1^e, \mathbf{h}_2^e, ..., \mathbf{h}_n^e = \text{biLSTM}(\mathbf{x}_1^e, \mathbf{x}_2^e, ..., \mathbf{x}_n^e) \tag{22.19}$$

然后，解码器是前馈网络 W，W 基于堆栈上的前三个子树(s_0，s_1，s_2)加上队列中的第一个 EDU(q0)的拼接来输出动作 o：

$$\mathbf{o} = \mathbf{W}(\mathbf{h}_{s0}^t, \mathbf{h}_{s1}^t, \mathbf{h}_{s2}^t, \mathbf{h}_{q0}^e) \tag{22.20}$$

其中，队列h_{q0}^e上的 EDU 的表示直接来自编码器，代表部分树的三个隐藏向量是通过对这些树中的

EDU 的编码器输出进行平均池化计算得到的:

$$\mathbf{h}_S^t = \frac{1}{j-i+1}\sum_{k=i}^{j}\mathbf{h}_k^e \tag{22.21}$$

训练首先将每个 RST 黄金解析树映射到一系列预言(oracle)操作中，然后使用标准交叉熵损失(L2 正则化)训练系统以采取这些操作。给出一个状态 S 和预言操作 a，我们首先用公式(22.20)计算解码器输出，应用 softmax 得到概率:

$$p_a = \frac{\exp(\mathbf{o}_a)}{\sum_{a'\in A}\exp(\mathbf{o}_{a'})} \tag{22.22}$$

然后计算交叉熵损失：

$$L_{CE}() = -\log(p_a) + \frac{\lambda}{2}||\Theta||^2 \tag{22.23}$$

RST 话语分析器在 RST 话语树库的测试部分进行评估，通过黄金 EDUs 或端到端的技术，使用 RST-pareval 度量(Marcu, 2000b)。标准方法是首先将 RST 黄金树转换为右分支二叉树，并报告四个度量：无标签的树(S 表示跨度)，有标签的核心(N)，有标签的关系(R)，或两者皆是(F，即 Full)，在所有文档的所有跨度中，对每个度量计算微平均 F1(Marcu 2000b，Morey 等人 2017)。

22.2.3. PDTB 话语解析

PDTB 话语分析的任务是检测话语之间的连贯关系，有时也被称为**浅层话语分析(shallow discourse parsing)**，因为该任务只涉及文本跨度之间的扁平关系，而不是 RST 分析的完整树。

Lin 等(2014)在第一个完整的系统中提出了 PDTB 话语解析的四个子任务集，分别针对显式(任务 1-3)和隐式(任务 4)连接词:

1. 找出话语连接词(从非话语使用中消除它们的歧义)
2. 找出每个连接词的两个跨度
3. 标签这些跨度之间的关系
4. 在每一对相邻的句子之间分配一个关系

已经针对任务 4 提出了许多系统：将一对相邻句子作为输入并分配一个连贯关系含义标签作为输出。设置通常遵循 Lin 等人(2009)的方法，假设金牌句子跨度边界，并为每个相邻跨度分配 11 个第二级 PDTB 标签之一或不分配(删除了图 22.3 中以斜体显示的 16 个中的 5 个非常罕见的标签)。

任务 4 的一种简单但功能非常强大的算法是，通过 BERT 嵌入来表示两个跨度中的每一个，并获取与[CLS]符记位置相对应的最后一层隐藏状态，然后将其通过单层 tanh 前馈网络，最后通过用于含义分类的 softmax(Nie 等人，2019)。

其他每个任务也已解决。任务 1 是消除话语连接词与非话语连接之间的歧义。例如，如 Pitler 和 Nenkova(2009)所指出的，单词 and 是(22.24)中通过细化/扩展关系将两个从句连接起来的话语连接词，而在(22.25)中是非话语 NP 连词：

(22.24) Selling picked up as previous buyers bailed out of their positions and aggressive short sellers—anticipating further declines—moved in.

(22.25) My favorite colors are blue and green.

同样，单词 once，在(22.26)中，是表示时间关系的话语连接词；而在(22.27)中，只是一个非话语副词，意为“formerly”和修饰 used:

(22.26) The asbestos fiber, crocidolite, is unusually resilient once it enters the lungs, with even brief exposures to it causing symptoms that show up decades later, researchers said.

(22.27) A form of asbestos once used to make Kent cigarette filters has caused a high percentage

of cancer deaths among a group of workers exposed to it more than 30 years ago, researchers reported.

因此，确定一个词是否为话语连接词是词义消歧的一种特殊情况。早期的消歧工作表明，使用来自黄金解析树的语法特征，4 个 PDTB 高级含义类别可以以高(94%)的准确率消歧(Pitler 和 Nenkova,2009)。最近的工作是执行单词输入端到端的任务，使用 biLSTM-CRF 与 BIO 输出(B-CONN,I-CONN, O)方法(Yu 等人，2019)。

对于任务 2,PDTB 跨度可以用与发现 RST EDUs 相同的序列模型进行识别:带有预训练的上下文嵌入(BERT)输入的 biLSTM 序列模型(Muller 等人，2019)。简单的启发式也可以很好地作为寻找跨度的基准，因为 93%的关系要么完全在一个单一的句子内，要么跨越两个相邻的句子，每个句子中都有一个论元(Biran 和 McKeown, 2015)。

22.3. 中心化和基于实体的连贯

话语可以连贯的第二种方式是依靠“about”某个实体的属性。这种观点认为，在话语的每个点上，某个实体都是显著的，并且通过继续讨论同一实体，话语是连贯的。这种观点出现在功能语言学和话语心理学的早期(Chafe 1976，Kintsch 和 Van Dijk 1978)，并很快出现了计算模型。在本节中，我们将介绍这种**基于实体(entity-base)**的连贯性的两个模型：中心理论(Grosz 等人，1995)和实体网格模型(Barzilay 和 Lapata，2008)。

22.3.1. 中心化

中心化理论(Centering Theory)(Grosz 等人，1995)是一种研究话语显著性和话语连贯性的理论。作为一种话语显著模型，中心化提出在话语的任何给定点上，话语模型中的一个实体都是显著的:它是被“中心化”的。作为话语连贯的一个模型，中心化提出，相邻句子 CONTINUE(继续)保持相同显著实体的话语比在多个实体之间来回 SHIFT(位移)的话语更连贯(我们将看到，CONTINUE 和 SHIFT 在该理论中是专业术语)。Grosz 等人(1995)的以下两篇命题内容完全相同但显著性不同的文本，有助于理解主要的中心化直觉。

(22.28) a. John went to his favorite music store to buy a piano.
b. He had frequented the store for many years.
c. He was excited that he could finally buy a piano.
d. He arrived just as the store was closing for the day.

(22.29) a. John went to his favorite music store to buy a piano.
b. It was a store John had frequented for many years.
c. He was excited that he could finally buy a piano.
d. It was closing just as John arrived.

虽然这两个文本的不同之处仅仅在于这两个实体(John 和 store)在句子中是如何实现的，但(22.28)中的话语在直觉上比(22.29)的话语更连贯。正如 Grosz 等人(1995)指出的那样，这是因为(22.28)中的话语显然是关于一个人，John，描述他的行为和感受。相比之下，(22.29)中的话语首先聚焦于 John，其次是商店，然后回到 John，最后又回到商店。它缺乏第一个话语的“aboutness(关于性)”。

中心化理论通过为每个语段 U_n 保留两个表示来实现这种直觉。U_n 的**向后看中心(backward-looking center)**，记作 $C_b(U_n)$，代表当前的显著性实体，即解释 U_n 之后的话语中所关注的实体。U_n 的**向前看中心(forward-looking center)**，记作 $C_f(U_n)$，是一组潜在的未来显著性实体，由 U_n 所唤起的话语实体中的任何一个可以作为 C_b(显著性实体)的后续语段，即 $C_b(U_{n+1})$。

向前看中心 $C_f(U_n)$的集合，根据诸如话语显著性和语法作用等因素进行排名(例如，主语比宾语的排名高，而宾语比所有其他语法角色排名高)。我们称排名最高的向前看中心 C_p(“首选中心”)。C_p 是对接下来将要讨论的实体的一种预测。有时下一个语段确实在谈论这个实体，但是有时另一个实体变得显著。

这里我们将使用 Brennan 等人(1987)提出的中心化算法，它定义了一对语段 U_n 和 U_{n+1} 之间的四种句

间(intersentential)关系，这些关系依赖于 $C_b(U_{n+1})$、$C_b(U_n)$和 $C_p(U_{n+1})$之间的关系;如图 22.7 所示(来自 Brennan 等人,1987)。

	$C_b(U_{n+1}) = C_b(U_n)$ or undefined $C_b(U_n)$	$C_b(U_{n+1}) \neq C_b(U_n)$
$C_b(U_{n+1}) = C_p(U_{n+1})$	Continue	Smooth-Shift
$C_b(U_{n+1}) \neq C_p(U_{n+1})$	Retain	Rough-Shift

图 22-7：规则 2 的定心转换

该算法使用的规则如下:

> **规则1**：如果$C_f(U_n)$中的任何元素是由一个在语段U_{n+1}中的代词实现的，
> 那么$C_b(U_{n+1})$也必须作为一个代词实现。
> **规则2**：转换状态是有序的。优先顺序是：继续、保留、平滑位移、粗略位移。

规则 1 反映了一种直觉，即代词化(包括零回指)是标记话语显著性的常用方法。如果一个语段中有多个代词实现了前一个语段的实体，那么这些代词中的一个必须实现后向中心 C_b；如果只有一个代词，它一定是 C_b。

规则 2 反映了一种直觉，即继续以同一实体为中心的话语比反复位移到其他中心的话语更连贯。转换表基于两个因素：从 U_n 到 U_{n+1} 的向后看中心 C_b 是否相同，以及该话语实体是否是从 U_n 首选的(C_p)。如果这两个条件都保持为 CONTINUE 关系，则说话者一直在谈论同一实体，并将继续谈论该实体。在 RETAIN 关系中，说话者打算在将来的语段中 SHIFT 到新实体，同时将当前实体置于较低的等级 C_f 中。在 SHIFT 关系中，说话者正在位移到新的显著实体。

让我们再次遍历(22.28)开头，重复(22.30)，显示处理每个语段后的表示。

(22.30) John went to his favorite music store to buy a piano. (U_1)
He was excited that he could finally buy a piano. (U_2)
He arrived just as the store was closing for the day. (U_3)
It was closing just as John arrived (U_4)

使用语法角色层次对 C_f 进行排序，对于句子 U_1，我们得到:

$C_f(U_1)$: {John, music store, piano}
$C_p(U_1)$: John
$C_b(U_1)$: undefined

然后，对于句子 U_2:

$C_f(U_2)$: {John, piano}
$C_p(U_2)$: John
$C_b(U_2)$: John
Result: Continue ($C_p(U_2)$=$C_b(U_2)$; $C_b(U_1)$ undefined)

从 U_1 到 U_2 的转变是一个。完成这个例子留给读者作为练习(1)。

22.3.2. 实体网格模型

中心化体现了关于实体提及如何导致连贯性的特殊理论：显著实体出现在主语位置或被代词化，并且话语通过继续以这种方式提及同一实体而变得显著。

Barzilay 和 Lapata(2008)的**实体网格(entity grid)**模型是捕获基于实体的连贯性的另一种方法:与自上而下的理论不同，实体网格模型使用机器学习来引导实体提及的模式，使话语更加连贯。

该模型基于实体网格，这是一个二维数组，表示在句子中提到的实体的分布。行表示句子，列表示话语实体(实体网格模型的大多数版本只关注名词性的提及)。每个单元格表示一个实体在句子中可能出现的情况，值表示该实体是否出现及其语法角色。语法角色是主语(S)、宾语(O)、两者度不是(X)或不存在(-);

在 Barzilay 和 Lapata(2008)的实现中，被动的主语被用 O 来表示，导致了具有某些主题角色特征的表示。

	Department	Trial	Microsoft	Evidence	Competitors	Markets	Products	Brands	Case	Netscape	Software	Tactics	Government	Suit	Earnings	
1	S	O	S	X	O	–	–	–	–	–	–	–	–	–	–	1
2	–	–	O	–	–	X	S	O	–	–	–	–	–	–	–	2
3	–	–	S	O	–	–	–	–	S	O	O	–	–	–	–	3
4	–	–	S	–	–	–	–	–	–	–	–	S	–	–	–	4
5	–	–	–	–	–	–	–	–	–	–	–	–	S	O	–	5
6	–	X	S	–	–	–	–	–	–	–	–	–	–	–	O	6

图 22-8：实体网格

图注：图 22.9 中的文本的部分实体网格。实体按其中心名词列出;每个单元格表示一个实体是否以主语(S)、宾语(O)、两者度不是(X)或不存在(-)的形式出现。数据来自 Barzilay 和 Lapata(2008 年)。

1 [The Justice Department]$_S$ is conducting an [anti-trust trial]$_O$ against [Microsoft Corp.]$_X$ with [evidence]$_X$ that [the company]$_S$ is increasingly attempting to crush [competitors]$_O$.
2 [Microsoft]$_O$ is accused of trying to forcefully buy into [markets]$_X$ where [its own products]$_S$ are not competitive enough to unseat [established brands]$_O$.
3 [The case]$_S$ revolves around [evidence]$_O$ of [Microsoft]$_S$ aggressively pressuring [Netscape]$_O$ into merging [browser software]$_O$.
4 [Microsoft]$_S$ claims [its tactics]$_S$ are commonplace and good economically.
5 [The government]$_S$ may file [a civil suit]$_O$ ruling that [conspiracy]$_S$ to curb [competition]$_O$ through [collusion]$_X$ is [a violation of the Sherman Act]$_O$.
6 [Microsoft]$_S$ continues to show [increased earnings]$_O$ despite [the trial]$_X$.

图 22-9：实体话语

图注：带有语法功能标记和注释的实体的话语。数据来自 Barzilay 和 Lapata(2008 年)。

Barzilay 和 Lapata(2008)的图 22.8 显示了图 22.9 中所示文本的网格。六个句子中的每一句都有一行。对于实体“ trial”，第二列为 o - - - - x，表明该 trial 在第一句话中作为直接宾语出现，在最后一句话中以斜线(间接)出现，而没有出现在中间句子中。第三列，对于 Microsoft 实体，显示为句子 1 中的主语(它也作为介词 against 的宾语出现，但多次出现的实体以其最高的语法功能记录下来)。计算实体网格需要提取实体并进行共指消解，以将它们聚类为话语实体(第 22 章)，并解析句子以获得语法角色。

在结果网格中，密集的列(如 Microsoft 列)表示文本中经常提到的实体；稀疏的列(如 Earnings 列)表示很少提及的实体。

在实体网格模型中，通过局部实体转换的模式来确保一致性。例如，Department 是句子 1 中的主语，然后在句子 2 中未提及；这是转换[S -]。因此，转换是序列{S,O X,-}^{n}，可以从每列中将其提取为连续单元格。每个转换都有一个概率;图 22.8 中网格[S -]的概率为 0.08(在长度为 2 的 75 个总转换中发生 6 次)。图 22.10 显示了图 22.9 的文本(显示为第一行 d_1)和其他 2 个文档的长度为 2 的转换分布。

	SS	SO	SX	S–	OS	OO	OX	O–	XS	XO	XX	X–	–S	–O	–X	– –
d_1	.01	.01	0	.08	.01	0	0	.09	0	0	0	.03	.05	.07	.03	.59
d_2	.02	.01	.01	.02	0	.07	0	.02	.14	.14	.06	.04	.03	.07	0.1	.36
d_3	.02	0	0	.03	.09	0	.09	.06	0	0	0	.05	.03	.07	.17	.39

图 22-10：表示文档的特征向量

图注:用于使用所有长度为 2 的转换表示文档的特征向量。文档 d_1 是图 22.9 中的文本。图来自 Barzilay 和 Lapata(2008)。

然后可以将转换及其概率用作机器学习模型的特征。该模型可以是经过训练以产生人类标签的连贯性分数的文本分类器(例如，来自人工将每个文本标签为连贯的或不连贯的)。但是收集此类数据非常昂贵。Barzilay 和 Lapata(2005)引入了一种简化的创新：可以通过自我监督来训练连贯模型：被训练为将话语中句子的自然原始顺序与修改顺序(例如随机顺序)区分开来。我们将在下一部分中介绍这些评估。

22.3.3. 评估神经和基于实体的连贯

基于实体的连贯模型以及我们在下一节中介绍的神经模型通常以两种方式之一进行评估。

首先，我们可以让人员对文档的连贯性进行评分，并训练分类器来预测这些人员的评分，这些评分可以是分类的(高/低，或高/中/低)或连续的。如果我们要考虑一些最终任务(例如论文评分)，那么这是最好的评估方法，其中人工评分是最终标签的正确定义。

另外，由于获得人工标签非常昂贵，而且我们可能还没有考虑到最终任务，因此我们可以使用自然文本进行自我监督。在自我监督中，我们将自然话语与通过更改顺序创建的伪文档配对。由于自然排序的话语比随机排列更具连贯性(Lin 等人，2011)，一个成功的连贯算法应优先考虑原始排序。

自我监督已通过三种方式实施。

第一，在句子顺序判别任务中(Barzilay 和 Lapata，2005)，我们将文档与其句子的随机排列进行比较。如果模型对原始文档的排名较高，则认为该模型对(原始的，排列的)测试对是正确的。给定 k 个文档，我们可以计算 n 个排列，从而得到 kn 对，每个对具有一个原始文档和一个排列，以用于训练和测试。

第二，在句子插入任务中(Chen 等人，2007)，我们获取一个文档 s，s 中有 n 个句子，删除其中一个句子，然后创建 n-1 个文档副本，每个位置插入 s。任务是确定 n 个文档中的哪个是带有原始顺序的文档，将 s 的原始位置与所有其他位置区分开。插入比区别更难，因为我们比较的是只有一个句子的不同文档。

第三，在句子顺序重建任务中(Lapata，2003)，我们制作了一个文档，随机化这些句子，并训练了模型以使其以正确的顺序放回去。再次给定 k 个文档，我们可以计算 n 个排列，得到 kn 对，每个对都有一个原始文档和一个排列，以用于训练和测试。与简单分类相比，重新排序当然是一项艰巨的任务。

22.4. 局部连贯的表示学习模型

第三种局部连贯是主题或语义场连贯。话语连贯是通过谈论相同的主题和子主题，并在此过程中利用相同的语义场来实现的。

该领域是在 1990 年代通过一系列无监督模型而开创性的，这种模型利用了**词汇衔接(lexical cohesion)**(Halliday 和 Hasan，1976)：在附近句子中共享相同或语义相关的单词。Morris 和 Hirst(1991)计算了单词的词汇链(lexical chains),其中单词(诸如 pine, bush trees, trunk)是通过话语发生的，而且是与 Roget 同义词库相关的(通过成为相同的类别，或连接的类别)。他们表明，链的数量和密度与主题结构相关。Hearst(1997)的**文字平铺(TextTiling)**算法计算了相邻文本跨度之间的余弦(原始单词数向量的归一化点积)，再次表明子主题中的句子或段落彼此之间具有较高的余弦值，但与相邻子主题中的句子之间却没有。

第三个早期模型是 Foltz 等人的 LSA 连贯方法，是第一个使用嵌入的模型，将两个句子之间的连贯性建模为其 LSA 句子嵌入向量之间的余弦[1]，通过将单词 w 的嵌入相加来计算句子 s 的嵌入：

$$\begin{aligned}\text{sim}(s,t) &= \cos(\mathbf{s},\mathbf{t}) \\ &= \cos(\sum_{w\in s}\mathbf{w},\sum_{w\in t}\mathbf{w})\end{aligned} \tag{22.31}$$

并将文本的整体连贯性定义为所有相邻的成对句子 si 和 s_{i+1} 的平均相似度：

$$\text{coherence}(T) = \frac{1}{n-1}\sum_{i=1}^{n-1}\cos(s_i,s_{i+1}) \tag{22.32}$$

现代神经表示-学习连贯模型，始于 Li 等人(2014)，利用这些早期无监督模型的直觉来学习句子表示，并测量它们在相邻句子之间的变化。但新模型也借鉴了 Barzilay 和 Lapata(2005)开创的自我监督理念。连贯关系模型训练在 RST 或 PDTB 上的以手工标记的表示形式，不像连贯关系模型，这些新模型被训练，以便区分自然话语和通过扰乱句子顺序而形成的非自然话语，从而使用表示学习来发现至少对于连贯排序

[1] 关于 LSA 嵌入的更多信息，请参阅第 6 章;它们通过对术语-文档矩阵(每个单元格用对数频率加权，用熵归一化)应用奇异值分解(SVD)计算，然后将前 300 维作为嵌入。

方面重要的特征。

在这里，我们介绍一种这样的模型，即局部连贯鉴别器(LCD)(Xu 等人，2019)。与早期模型一样，LCD 将文本的连贯性计算为连续成对句子之间连贯性分数的平均值 但是与早期的非监督模型不同，LCD 是一种自监督模型，经过训练可以从(构造的)非连贯的成对(s_i，s')句子中区分出训练文档(假定为连贯)中的连续成对(s_i，s_{i+1})句子。所有连续成对的都是正例，句子 s_i 的负面(不连贯)搭档是从与 s_i 相同的文档中统一采样的另一个句子。

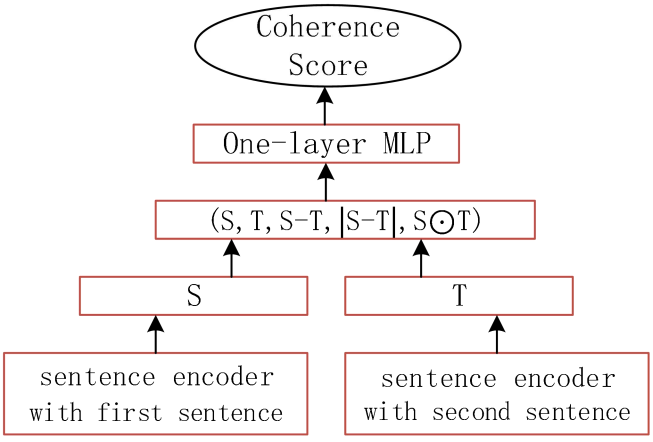


图 22-11：文档连贯性的 LCD 模型架构

图注：图中展示了一对句子 s 和 t 的得分计算。图来自 Xu et al.(2019)。

图 22.11 描述了模型 f_θ的架构，该模型接受一对句子并返回一个分数，一对句子越连贯得分越高。给定一对输入句子 s 和 t，计算句子嵌入 s 和 t(使用任意的句子嵌入算法)，然后把这一对句子的四个特征进行拼接:(1)两个向量的拼接;(2)差分 s-t;(3)其差分|s-t|的绝对值;(4)它们的逐元素点积 s⊙t。这些信息通过一层前馈网络输出连贯性得分。

这个模型经过训练，使真实配对的连贯性得分高于负面配对的连贯性得分。更正式地说，对于由一系列句子 s_i 组成的文档 d 的语料库 C 的训练目标是:

$$L_\theta = \sum_{d \in C} \sum_{s_i \in d} \mathop{\mathbb{E}}_{p(s'|s_i)} \left[L\left(f_\theta(s_i, s_{i+1}), f_\theta(s_i, s') \right) \right] \tag{22.33}$$

$E_{p(s'|si)}$是对以 s_i 为条件的负面抽样分布的期望:给定一个句子 s_i，算法在同一文档的其他句子上统一抽样负面句子 s'。L 是一个损失函数，包含两个分数，一个是正的一对，一个是负的一对，目标是鼓励 $f^+ = f_\theta(s_i,s_{i+1})$高，$f^- = f_\theta(s_i,s')$低。图 22.11 使用边际损失 $l(f^+, f^-) = \max(0,\eta - f^+ + f^-)$，其中η为边际超参数。

徐等人(2019)还给出了一种有用的基线算法，该算法本身在度量困惑度方面具有相当高的性能：在数据上训练 RNN 语言模型，并以两种方式计算句子 s_i 的对数似然度：一个是给出了前面的上下文(条件对数似然度)，一个是没有上下文(边际对数似然度)。这些值之间的差异告诉我们，前面的上下文在多大程度上提高了 s_i 的可预测性(s_i 是可预测性的连贯性度量)。

预测较长上下文的训练模型比预测连续成对的句子能产生更强的话语表示。例如，一个用对比句训练的 Transformer 语言模型，其目标是预测 ±2 个句子以内的文本，可以提高在各种话语连贯任务中的表现(Iter 等人，2020)。

语言模型风格模型通常通过第 22.3.3 节的方法来评估，尽管它们也可以在 RST 和 PDTB 连贯关系任务上进行评估。

22.5. 全局连贯

话语还必须在全局范围内连贯，而不仅仅是在句子对的层面上。例如，考虑故事。故事的叙事结构是要研究的最古老的全局连贯性之一。Propp(1968)在他的有影响力的《民间故事形态学》中，通过一种情节语法对俄罗斯民间故事的话语结构进行了建模。他的模型包括他称为剧中人(dramatis personae)的一组

角色类别，例如英雄、反派、施主或助手，以及他称为功能(function)的一组事件(例如“反派实施绑架”，“ 施主测试英雄”或“英雄被追逐”)必须与其他组件一起按特定顺序进行。普洛普(Propp)表明，他研究的每个神话故事的情节都可以表示为这些功能的序列，不同的故事选择不同的功能子集，但始终以相同的顺序进行。实际上，Lakoff(1972b)证明了 Propp 的模型相当于故事的话语语法，而在最近的计算工作中，Finlayson(2016)证明，某些 Propp 的功能可以通过检测故事中具有相似动作的事件从民间故事文本的语料库中诱发。实际上，Lakoff(1972b)证明了 Propp 的模型相当于故事的话语语法，而在最近的计算工作中，Finlayson(2016)证明，某些 Propp 的功能可以通过检测故事中具有相似动作的事件从民间故事文本的语料库中诱发。

在本节中，我们介绍了两种已经被广泛研究的全局话语结构。第一种是论元结构:人们试图通过提供某些主张和支持某些前提在雄辩文章中来说服对方的方式。第二种是有些关联:科学论文的结构，作者表达他们的目标、结果的方式，以及与他们论文中之前工作的关系。

22.5.1. 论元结构

第一种全局话语结构是论元结构。对人们的论元(people’s argumentation)进行计算分析常被称为**论元挖掘(argumention mining)**。

论元的研究可以追溯到亚里士多德(Aristotle)，他在他的修辞学中描述了一个好的论元的三个成分：**感召(pathos)**(诉诸于听众的情绪)，**精神(ethos)**(诉诸于说话者的个性)和**理性(logos)**(论元的逻辑结构)。

大多数的论元话语结构研究都聚焦于理性，特别是通过构建和训练说服性论文或其他论元的注释数据集(Reed 等人 2008, Stab 和 Gurevych 2014a, Peldszus 和 Stede 2016, Habernal 和 Gurevych 2017, Musi 等人 2018)。例如,这样的语料库通常包括论元成分的注释诸如**主张(claims)**(论元的中心成分是有争议的，需要支持)和**前提(premises)**(作者给出的理由说服读者通过支持或攻击宣言或其他前提)，以及它们之间的**论元关系(argumentative relations)**(例如 SUPPORT 和 ATTACK)。

下面是 Stab 和 Gurevych (2014b)的一篇说服性文章。第一句话(1)提出了一个主张(黑体)。(2)和(3)提出支持主张的两个前提。(4)给出一个支持前提(3)的前提。

“(1) **Museums and art galleries provide a better understanding about arts than Internet.**

(2) In most museums and art galleries, detailed descriptions in terms of the background, history and author are provided.

(3) Seeing an artwork online is not the same as watching it with our own eyes, as

(4) the picture online does not show the texture or three-dimensional structure of the art, which is important to study.”

因此这个例子有三个论元关系：SUPPORT(2,1)，SUPPORT(3,1)和 SUPPORT(4,3)。图 22.12 显示了一个更为复杂的论元结构。

尽管论元挖掘显然与修辞结构和其他类型的连贯关系有关，但论元却往往不那么局限；反之亦然。一个有说服力的文章通常只有一个主要主张，前提是贯穿整个文本，而没有我们在连贯关系中看到的局部连贯性。

用于检测论元结构的算法通常包括用于区分主张、前提或非论元的分类器，以及用于确定两个跨度是否具有 SUPPORT，ATTACK 或两者都不具有的关系分类器(Peldszus 和 Stede，2013)。尽管这些是许多计算工作的主要焦点，但在注释和检测更丰富的语义关系方面也做出了初步努力(Park 和 Cardie 2014，Hidey 等人 2017)，例如检测**论元方案(argumentation schemes)**，用于论元的大规模结构诸如来自 示例的论元(argument from example)，或来自因果的论元(argument from cause to effect)，或来自后果(argument from consequences)的论证(Feng 和 Hirst，2011)。

另一个重要的研究领域是研究这些论元结构(或其他特征)如何与论元的成功或说服力相关联(Habernal 和 Gurevych 2016，Tan 等人 2016，Hidey 等人 2017)。的确，尽管这是亚里斯多德的与话语结构最相关的理性，亚里士多德的精神和感召技巧在发现这种说服力(persuasion)的机制中尤为重要。例如，学者们研究了社会科学家研究的特征的语言实现诸如互惠(人们回报恩德)、社会证明(人们遵循他人的

选择)、权威(人们受到有权力的人的影响)和稀缺性(人们珍视稀缺的东西)，所有这些都可以通过有说服力的论元提出来(Cialdini，1984)。Rosenthal 和 McKeown(2017)指出，这些特征可以与论元结构相结合，以预测谁会对社交媒体产生影响，Althoff 等人(2014)发现互惠和权威的语言模型预测在线请求的成功，而杨等人(2019)的半监督模型检测到提及稀缺性，承诺和社会认同的情况，以预测点对点借贷平台的成功。

参见 Stede 和 Schneider(2018)对论元挖掘的全面综述。

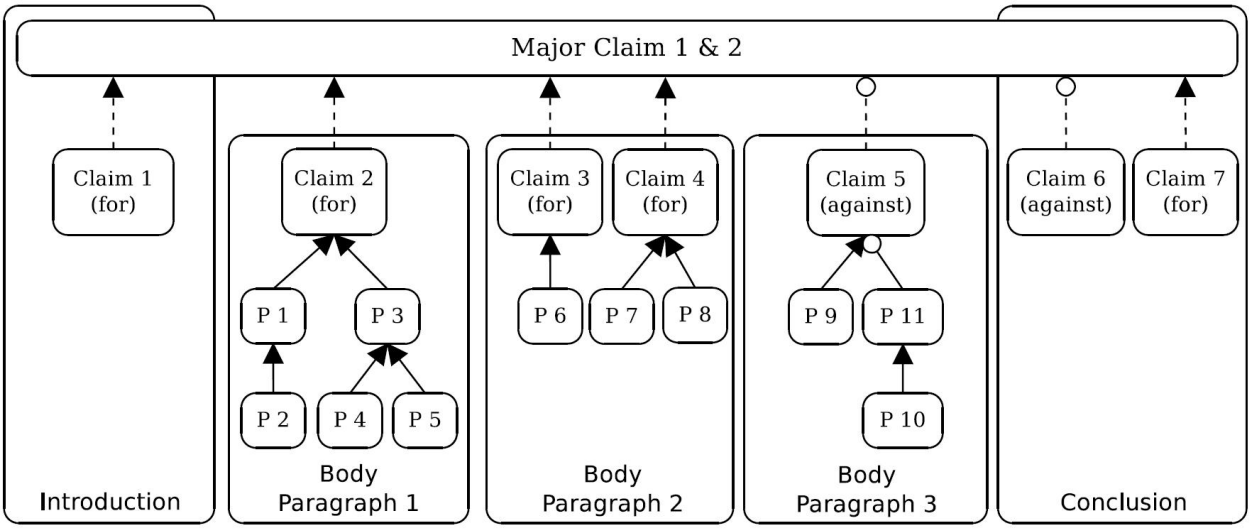


图 22-12：说服性文章的论元结构

图注：箭头表示论元关系，要么是 SUPPORT(带箭头)要么是 ATTACK(带圆头)；P 表示前提。图来自 Stab 和 Gurevych(2017)。

22.5.2. 科学话语的结构

科学论文有一个非常具体的全局结构:在论文过程中的某个地方，作者必须指出一个科学目标，为解决方案开发一种方法，为解决方案提供证据，并与之前的工作进行比较。一个流行的注释方案建模这些修辞的目标是**论元区划(argumentative zoning)**模型 Teufel 等人(1999)和 Teufel 等人(2009),这是因为每个科学论文都试图提出关于将新知识添加到该领域知识库中的知识主张(Myers，1992)。科学论文中的句子可以分配 15 个标签之一；图 22.13 显示了带标记的句子的 7 个(缩短的)示例。

Category	Description	Example
AIM	Statement of specific research goal, or hypothesis of current paper	“The aim of this process is to examine the role that training plays in the tagging process”
OWN_METHOD	New Knowledge claim, own work:methods	“In order for it to be useful for our purposes, the following extensions must be made:”
OWN_RESULTS	Measurable/objective outcome of own work	“All the curves have a generally upward trend but always lie far below backoff (51% error rate)”
USE	Other work is used in own work	“We use the framework for the allocation and transfer of control of Whittaker....”
GAP_WEAK	Lack of solution in field, problem with other solutions	“Here, we will produce experimental evidence suggesting that this simple model leads to serious overestimates”
SUPPORT	Other work supports current work or is supported by current work	“Work similar to that described here has been carried out by Merialdo (1994), with broadly similar conclusions.”
ANTISUPPORT	Clash with other’s results or theory; superiority of own work	“This result challenges the claims of...”

图 22-13：论元区划示例

图注：论元区划标签集合中的 15 个标签中的 7 个示例(Teufel 等人，2009)。

Teufel 等人(1999)和 Teufel 等人(2009)从计算语言学和化学中开发了有标签的科学文章语料库，这些语料库可用于训练标准的句子分类体系结构来分配 15 个标签。

22.6. 总结

在本章中，我们介绍了**话语连贯**的局部模式和全局模式。

•**话语**不是句子的任意集合;它们必须是**连贯**的。使话语连贯的因素包括句子之间的连贯关系、基于实体的连贯和主题的连贯。

•提出了多种**连贯关系**和**修辞关系**。**修辞结构理论(RST)**将文本跨度之间的关系划分为树状结构。因此，通常使用位移-减少(shift-reduce)和其他解析算法来分配这些结构。**Penn 话语树库(PDTB)**只标注成对跨度之间的关系，标签通常由序列模型来分配。

•**基于实体的连贯**抓住了话语是关于一个实体的直觉，并从一句到一句继续提及这个实体。**中心化理论**是一组模型，描述了话语实体的显著性如何被建模，以及如何通过保持相同的话语实体在话语中显著性来实现连贯。**实体网格模型**提供了一种更自底向上的方法来计算哪些实体实现转换会导致连贯性。

•许多不同的体裁都有不同类型的**全局连贯性**。有说服力的文章有主张和前提，它们在**论元挖掘**领域被提取，科学文章有与目标、方法、结果和比较相关的结构。

22.7. 文献和历史说明

连贯关系产生于许多学者的独立发展，包括 Hobbs(1979)认为连贯关系对听者起推理作用的观点，以及 Mann 和 Thompson(1987)对文本的话语结构的研究。其他研究连贯关系及其提取的方法包括**分段话语表示理论(Segmented Discourse Representation Theory，SDRT)** (Asher 和 Lascarides 2003, Baldridge 等人 2007)和语言话语模型(Polanyi 1988, Scha 和 Polanyi 1988, Polanyi 等人 2004)。Wolf 和 Gibson(2005)认为，连贯结构包括交叉的括号，这使它不可能以树的形式表示，并提出了一个图形表示代替。Hovy(1990)中列出了文献中提出的 350 多种关系。

RST 句法分析最早由 Marcu(1997)提出，早期的研究是以规则为基础的，主要关注话语标记(Marcu, 2000a)。RST 话语树库的创建(Carlson 等人 2001,Carlson 和 Marcu 2001)使各种各样的机器学习算法,开始与 Marcu(1999)的 shift-reduce 解析器一起使用决策树来选择行动,并继续与各种机器学习解析方法(Soricut Marcu 2003，Sagae 2009，Hernault 等人 2010，Feng 和 Hirst 2014，Surdeanu 等人 2015，Joty 等人 2015)和分块器 Sporleder Lapata, 2005)一起使用。Subba 和 Di Eugenio(2009)将复杂的语义信息集成到 RST 解析中。Ji 和 Eisenstein(2014)首次将神经模型应用于 RST 解析神经模型,产生了现代神经 RST 模型集(尤其是 Li 等人 2014，Li 等人 2016，Braud 等人 2017，Yu 等人 2018)以及神经分段器(Wang 等人，2018)，以及神经 PDTB 解析模型(Ji 和 Eisenstein 2015, Qin 等人 2016, Qin 等人 2017)。

Barzilay 和 Lapata(2005)率先提出了自我监督连贯性的想法:训练一个连贯模型来区分句子的真实顺序和随机排列。Li 等人(2014)首先将这一范式应用于神经句子-表示，并遵循了许多神经自监督模型(Li 和 Jurafsky 2017, Logeswaran 等人 2018,Lai 和 Tetreault 2018, Xu 等人 2019,Iter 等人 2020)。

全局连贯的另一个方面是文本的全局主题结构，即主题在文件过程中位移的方式。Barzilay 和 Lee(2004)引入了一种 HMM 模型，用于捕获连贯性主题，随后的工作扩展了这种直觉(Soricut 和 Marcu 2006, Elsner 等人 2007,Louis 和 Nenkova 2012, Li 和 Jurafsky 2017)。

显性和隐性话语连接词之间的关系一直是研究的热点。Marcu 和 Echihabi(2002)首先提出使用带有显性关系的句子来帮助提供隐性关系的训练数据，通过去除显性关系并试图重新预测它们来提高隐性连接词的表现;Sporleder 和 Lascarides(2005)、Pitler 等人 2009)以及 Rutherford 和 Xue(2015)进一步完善了这一理念。这种关系还可以用作创建话语感知表示的一种方式。异议算法(DisSent algorithm)(Nie 等人，2019)创建了预测两个句子之间显性话语标记的任务。他们表明，在这项任务中学习到的表示也可以作为其他话语任务中强大的句子表示。

基于实体的连贯概念似乎在 20 世纪 70 年代中期出现在多个领域，包括功能语言学(Chafe 1976)、话语处理心理学(Kintsch 和 Van Dijk, 1978)，以及 Grosz、Sidner、Joshi 及其同事大致同时期的工作。Grosz(1977a)指出了对话参与者在话语展开时所保持的注意力焦点。她定义了关注点的两个层次;与整个

话语相关的实体被称为全局焦点，而在局部焦点的实体(即，一个特定语段的最中心)被称为即时焦点。Sidner(1979,1983)描述了一种追踪(即时)话语焦点的方法及其在解决代词和指示名词短语中的使用。她区分了当前话语焦点和潜在焦点，分别是向后看和向前看中心理论中心的前辈。中心化方法的名称和进一步的根源在于 Joshi 和 Kuhn(1979)和 Joshi 和 Weinstein(1981)的论文，他们探讨了即时焦点和将当前语段整合到话语模型中所需要的推理之间的关系。Grosz 等人(1983)将这项工作与 Sidner 和 Grosz 之前的工作相结合。这导致了一份关于中心化的手稿，自 1986 年以来广泛流传，直到 Grosz 等人(1995)才发表。Walker 等人(1998)发表了一组中心化论文。请参阅 Karamanis 等人(2004 年)和 Poesio 等人(2004 年)对中心化及其参数化的深入探索，以及第 22 章的历史部分，以获得更多关于在连贯上中心化的使用。

基于实体的连贯的网格模型首先由 Barzilay 和 Lapata(2005)提出，借鉴了 Lapata(2003)和 Barzilay 的早期工作，然后由 Barzilay 和 Lapata(2008)以及其他具有其他特征(Elsner 和 Charniak，2008，2011)进行了扩展。Feng 等人 2014，Lin 等人 2011 的模型，将实体投影到话语的全局图中(Guinaudeau 和 Strube 2013，Mesgar 和 Strube 2016)，以及卷积模型以捕获较长范围实体依存关系(Nguyen 和 Joty，2017)。

话语连贯理论也被用于解释话语级语言现象的算法，包括动词短语省略和间隙(Asher 1993, Kehler 1993)和时态解释(Lascarides 和 Asher 1993, Kehler 1994, Kehler 2000)。Knott 和 Dale(1994)对连贯关系和话语连接词之间的关系进行了广泛的研究。Stede(2011)和 Webber 等人(2012)对话语处理和结构进行了有益的研究。

Andy Kehler 为本书的 2000 年第一版写了话语一章，我们用它作为第二版的起点，在第三版的连贯性一章中还保留了 Andy 的一些可爱的散文。

22.8. 练习

22.1 Finish the Centering Theory processing of the last two utterances of (22.30), and show how (22.29) would be processed. Does the algorithm indeed mark (22.29) as less coherent?

22.2 Select an editorial column from your favorite newspaper, and determine the discourse structure for a 10 – 20 sentence portion. What problems did you encounter? Were you helped by superficial cues the speaker included (e.g., discourse connectives) in any places?

23. 问题回答

对知识的追求是深切的人类需求，因此毫不奇怪，自从有了计算机，我们就开始向它们提出问题。

到 1960 年代初，系统使用了两种主要的回答问题范式--基于信息检索和基于知识--来回答有关棒球统计或科学事实的问题。甚至虚构的计算机也参与其中。道格拉斯·亚当斯(Douglas Adams)在《银河系漫游指南》(The Hitchhiker's Guide to the Galaxy)中发明的计算机"深思"(Deep Thought)成功回答了"生命、宇宙和一切的终极问题"[1]。2011 年，IBM 的沃森问题回答系统赢得了电视游戏节目 Jeopardy!，在回答问题方面超越人类，比如：

WILLIAM WILKINSON'S "AN ACCOUNT OF THE
PRINCIPALITIES OF WALLACHIA AND MOLDOVIA"
INSPIRED THIS AUTHOR'S MOST FAMOUS NOVEL[2]

问答系统主要用于满足人类信息需求。人们在许多情况下会提出问题：与虚拟助手交谈时，与搜索引擎进行交互时，在查询数据库时。大多数问答系统都将重点放在这些信息需求的特定子集上：事实类问题，可以用短文本表示的简单事实来回答的问题，例如：

(23.1) Where is the Louvre Museum located?

(23.2) What is the average age of the onset of autism?

在这一章中，我们描述了事实类问题的两个主要范式。

第一种范式，基于信息检索(IR)的 QA，有时称为开放域问题 QA，依赖于 Web 上或 PubMed 等科学论文集中的大量文本。给定用户问题，信息检索可用于查找相关段落。然后，神经阅读理解算法读取这些检索到的段落，并直接从文本跨度中得出答案。

第二种范式，基于知识的 QA，系统改为构建查询的语义表示，例如,把 What states border Texas?(德克萨斯州与哪些州接壤？) 映射到逻辑表示形式：λx.state(x)∧borders(x，texas)，或者把 When was Ada Lovelace born?映射到间隙关系(gapped relation)：birth-year(Ada Lovelace，？X)。然后，这些含义表示用于查询事实数据库。

我们还将简要讨论用于回答问题的其他两个范例。一个事实是，我们在整个 NLP 中使用的庞大的预训练语言模型已经编码了许多事实。我们将介绍如何直接查询语言模型以回答问题。我们还将提及经典的预神经混合问答算法，该算法结合了来自基于 IR 和基于知识的来源的信息。

我们将探讨所有这些方法的可能性和局限性，同时还将介绍两种对问题解答至关重要但在整个 NLP 中也很重要的技术：信息检索(基于 IR 的 QA 的关键组件)和实体连接(类似于基于知识的 QA 的关键组件)。我们将在下一部分开始，介绍信息检索的任务。

最后注意：在本章中，我们仅关注事实类问题的解答，但感兴趣的读者可能还想跟进许多其他重要的 QA 任务。这些任务包括长格式的问题回答(回答为什么问题或其他需要生成长答案的问题)，社区问题回答（在其中我们利用了社区创建的问题-答案配对的数据集诸如 Quora 或 Stack Overflow。最后，一般而言，问答是 NLP 进步的重要基准，因此研究人员已经构建了可以在纽约 Regents 科学考试等考试中成功回答问题的系统，以此作为基准 NLP 和 AI 的一种方式(Clark 等人，2019)。

23.1. 信息检索(IR)

信息检索(Information retrieval，IR)是该领域的名称，涵盖了根据用户信息需求来检索所有形式的媒体。最终的 IR 系统通常称为搜索引擎。我们在本节中的目标是对 IR 进行足够的概述，以了解 IR 在问答中的应用。对信息检索更感兴趣的读者应参阅本章末尾的"历史说明"以及 Manning 等人(2008)的教科书。

我们考虑的 IR 任务称为**即席检索(ad hoc retrieval)**，其中用户向检索系统提出查询，然后检索系统

[1] 答案是 42，但不幸的是，问题的细节一直没有透露。

[2] 答案当然是"谁是布拉姆·斯托克"，小说是《德古拉》。

从某些集合中返回一组有序的文档。文档(document)指的是系统索引和检索的任何文本单位(网页，科学论文，新闻报道，甚至是短文诸如段落)。集合(collection)是指用于满足用户请求的一组文档。术语(term)是指集合中的单词，但也可以包括短语。最后，查询(query)代表用一组术语表示的用户信息需求。图 23.1 显示了即席检索引擎的高级体系结构。

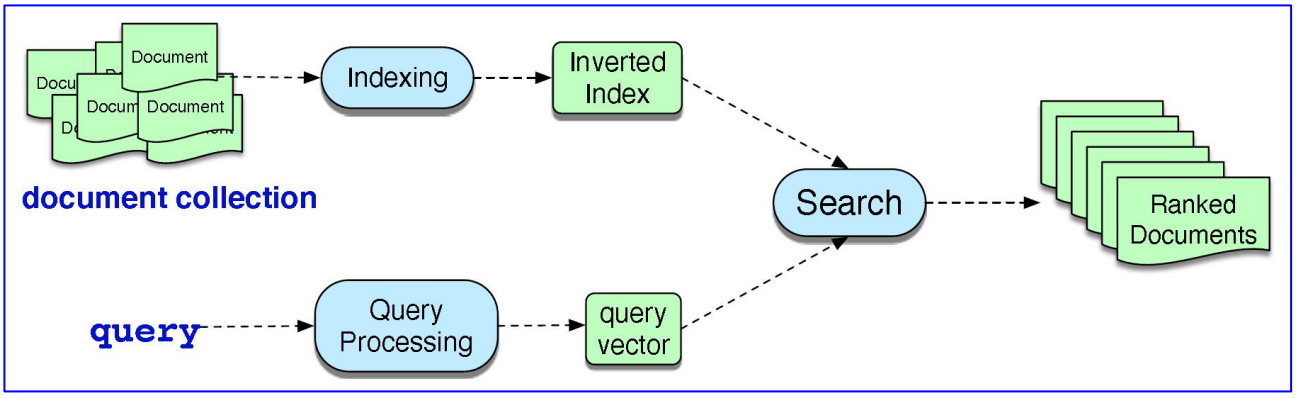


图 23-1：即席 IR 的系统架构

基本的 IR 体系结构使用我们在第 6 章中介绍的向量空间模型，在该模型中，我们基于 unigram 单词计数将查询和文档映射到向量，并使用向量之间的余弦相似度对潜在的文档进行排序(Salton, 1971)。因此，这是第 4 章中介绍的词袋模型的一个例子，因为单词是独立于它们的位置进行考虑的。

23.1.1. 术语加权和文档评分

让我们看看如何对文档和查询之间的匹配进行评分的细节。在 IR 中我们不使用原始的单词计数，而是为每个文档单词计算**术语权重(term weight)**。两个术语加权方案是常见的:在第 6 章介绍的 tf-idf 加权，和一个稍微更强大的变种称为 **BM25**。我们将在这里重新介绍 tf-idf，这样读者就不需要回头看第 6 章了。tf-idf(这里的“-”是一个连字符，而不是减号)是个术语频率 tf 和间接文档频率 idf 的乘积。

频率这个词告诉我们这个词有多频繁;在文档中经常出现的单词可能提供有关文档内容的信息。我们通常使用词频的 $\log_{10}$，而不是原始计数。直觉是，一个单词在文档中出现 100 次并不会使这个单词与文档含义的相关性增加 100 倍。因为我们不能取 0 的对数，所以我们通常在计数中加上 1:

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1) \tag{23.3}$$

如果我们使用对数加权，在一个文档中出现 0 次的项将有 tf=log10(0+1)=0，在一个文档中出现 10 次的项将有 $\mathrm{tf}=\log_{10}(10+1)=1.4$, 100 次的 $\mathrm{tf}=\log_{10}(100+1)=2.004$, 1000 次的 tf=3.00044，以此类推。术语 t 的文档频率 dft 是它出现在文档中的数量。仅出现在少数文档中的术语有助于将这些文档从集合的其余部分中区分出来;出现在整个集合中的术语就没有那么有用了。逆文档频率或 idf 词权重(Sparck Jones, 1972)定义为:

$$\mathrm{idf}_t = \log_{10}\frac{N}{\mathrm{df}_t} \tag{23.4}$$

其中，N 为集合中的文档总数，dft 为出现术语 t 的文档数。一个术语出现的文献越少，其权重越高;最小的 0 权重被分配给出现在每个文档中的术语。其中 N 是集合中文档的总数，而 dft 是出现术语 t 的文档数。出现术语的文档越少，此权重就越高；最低权重 0 分配给每个文档中出现的术语。

以下是莎士比亚戏剧语料库中某些单词的一些 idf 值，从仅在像《罗密欧》这样的一部戏剧中出现的非常有启发性的单词，到像《色拉》或《福斯塔夫》中很少出现的那些单词，再到像《傻瓜》，或如此普遍以至于完全没有区别，因为它们出现在所有 37 部《好》或《甜》的戏剧中。

Word	Romeo	salad	Falstaff	forest	battle	wit	fool	good	sweet
df	1	2	4	12	21	34	36	37	37
idf	1.57	1.27	0.967	0.489	0.246	0.037	0.012	0	0

则文档 d 中单词 t 的 tf-idf 值为词频 $\mathrm{tf}_{t,d}$ 和 idf_t 的乘积:

$$\text{tf-idf}(t,d) = \mathrm{tf}_{t,d} \bullet \mathrm{idf}_t \tag{23.5}$$

23.1.2. 文档评分

我们用查询向量 q 对文档 d 的向量 d 的余弦值进行评分：

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{|\mathbf{q}||\mathbf{d}|} \tag{23.6}$$

另一种考虑余弦计算的方法是单位向量的点积;我们首先将查询向量和文档向量标准化为单位向量，除以它们的长度，然后取点积:

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}}{|\mathbf{q}|}\cdot\frac{\mathbf{d}}{|\mathbf{d}|} \tag{23.7}$$

我们可以用 tf-idf 值拼出(spell out)公式(23.7)，并把点积拼出积的和:

$$\text{score}(q,d) = \sum_{t\in q}\frac{\text{tf}-\text{idf}(t,q)}{\sqrt{\sum_{q_i\in q}\text{tf}-\text{idf}^2(q_i,q)}}\cdot\frac{\text{tf}-\text{idf}(t,d)}{\sqrt{\sum_{d_i\in d}\text{tf}-\text{idf}^2(d_i,d)}} \tag{23.8}$$

在实践中，通常通过简化查询处理来近似公式(23.8)。查询通常很短，所以每个查询词的计数很可能是 1。并且查询的余弦归一化(除以|q|)对所有的文档都是一样的，所以不会改变任意两个文档 D_i 和 D_j 之间的排名，所以我们一般使用以下简单的分数对一个文档 d 给出一个查询 q:

$$\text{score}(q,d) = \sum_{t\in q}\frac{\text{tf}-\text{idf}(t,d)}{|d|} \tag{23.9}$$

让我们浏览一个针对 4 个 nano 文档集合的小查询示例，计算 tf-idf 值并查看文档的排序。我们假设以下查询和文档中的所有单词都被降格为小写，并且删除了标点符号:

Query: sweet love
Doc 1: Sweet sweet nurse! Love?
Doc 2: Sweet sorrow
Doc 3: How sweet is love?
Doc 4: Nurse!

图 23.2 显示了前两个文档使用公式(23.3)、公式(23.4)和公式(23.5)计算 tf-idf 值和文档向量长度|d|(文档 3 和文档 4 的计算留给读者作为练习)。

	Document 1					Document 2				
word	count	tf	df	idf	tf-idf	count	tf	df	idf	tf-idf
love	1	0.301	2	0.301	0.091	0	0	2	0.301	0
sweet	2	0.477	3	0.125	0.060	1	0.301	3	0.125	0.038
sorrow	0	0	1	0.602	0	1	0.301	1	0.602	0.181
how	0	0	1	0.602	0	0	0	1	0.602	0
nurse	1	0.301	2	0.301	0.091	0	0	2	0.301	0
is	0	0	1	0.602	0	0	0	1	0.602	0
$\lvert d_1\rvert = \sqrt{.091^2 + .060^2 + .901^2} = .141$						$\lvert d_2\rvert = \sqrt{.038^2 + .181^2} = .185$				

图 23-2：计算 nano 文档 1 和文档 2 的 tf-idf 值

Doc	|d|	tf-idf(sweet)	tf-idf(love)	score
1	.141	.060	.091	1.07
3	.274	.038	.091	0.471
2	.185	.038	0	0.205
4	.090	0	0	0

图 23-3：根据公式(23.9)对文档进行排序

图 23.3 显示了 4 篇文献的得分，根据公式(23.9)重新排序。这一排名从向量空间模型上得到了直观的体现。文档 1 是排名最高的，它有两个术语(sweet 和 love，其中 sweet 有两个实例)；在文档 3 之上，文

档 3 的分母长度|d|较大，sweet 的 tf 也较小。文档 3 缺少了其中一个术语(love)，文档 4 两个术语都缺少。

tf-idf 家族中一个稍微复杂一点的变体是 BM25 加权方案(有时在引入该方案的 Okapi IR 系统后被称为 Okapi BM25 (Robertson 等人 1995))。BM25 增加了两个参数:k(调节术语频率和 IDF 之间平衡的旋钮)和 b(控制文档长度规范化的重要性)。给定查询 q，文档 d 的 BM25 得分为:

$$\sum_{t \in q} \overbrace{\log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}^{\text{weighted tf}} \tag{23.10}$$

其中$|d_{avg}|$是平均文档的长度。当 k 为 0 时，BM25 恢复到不使用术语频率，只使用查询中的术语的二进制选择(加上 idf)。一个大的 k 会导致原始的术语更加频繁(加上 idf)。b 的范围是从 1(按文档长度缩放)到 0(无长度缩放)。Manning 等人(2008)认为合理的值是 k = [1.2,2]， b = 0.75。Kamphuis 等人(2020)对 BM25 的许多次要改型进行了有益的总结。

停用词 过去，通常在表示查询之前从查询和文档中都删除高频单词。将要去除的这种高频单词的列表称为停用列表(stop list)。直觉是，高频术语(通常是诸如 the、a、to 等功能词)的语义权重很小，可能对检索没有帮助，并且还可以帮助缩小我们在下面描述的倒排索引文件。使用停用列表的缺点是，如果一个搜索短语包含停用列表中的词，则这个短语很难被搜索到。例如，通常停用列表会将短语 to be or not to be 减少为 not。在现代的 IR 系统中，停用列表的使用要少得多，部分原因是提高了效率，部分原因是 IDF 加权已经处理了其大部分功能，从而减轻了每个文档中出现的功能词的权重。尽管如此，删除停用词在各种 NLP 任务中有时还是有用的，因此值得留意。

23.1.3. 倒排索引

为了计算分数，我们需要有效地查找在查询中包含单词的文档。(正如我们在图 23.3 中看到的，任何不包含查询项的文档的得分都为 0，可以忽略。)因此，IR 中的基本搜索问题是找到所有的 d∈C 的文档，这些文档包含一个术语 q∈Q。

这个任务的数据结构是**倒排索引(inverted index)**，我们使用它来提高搜索效率，还可以方便地存储有用的信息，如文档频率和每个文档中每个术语的计数。

给定查询词的倒排索引会给出包含该词的文档的列表。它由两部分组成：字典和帖子。词典是一个术语列表(旨在有效访问)，每个术语都指向该术语的帖子列表。帖子列表是与每个术语相关联的文档 ID 的列表，该列表还可以包含诸如术语频率或术语在文档中的确切位置之类的信息。字典也可以为每个术语开始文档频率。例如，上面四个示例文档的简单倒排索引[3]：每个单词的文档频率都在{ }中，一个指向帖子列表的指针→，该列表包含文档 ID 号，术语计数都在[ ]中。可能类似于以下内容：

how {1} → 3 [1]
is {1} → 3 [1]
love {2} → 1 [1] → 3 [1]
nurse {2} → 1 [1] → 4 [1]
sorry {1} → 2 [1]
sweet {3} → 1 [2] → 2 [1] → 3 [1]

给定查询中的术语列表，我们可以非常有效地获取所有候选文档的列表，以及计算 tf-idf 分数所需的必要信息。倒排索引还有其他选择。对于寻找维基百科页面以匹配用户查询的问题域，Chen 等人(2017)表明，基于 bigrams 的索引比 unigrams 更好，并且使用有效的哈希算法而不是倒排索引来提高搜索效率。

23.1.4. 信息检索系统的评估

我们使用相同的精度和召回率指标来衡量排名检索系统的性能。我们假设 IR 系统返回的每个文档都

[3] 译者注：至少包含四项内容：{}，→，ID，[ ].

与我们的目的相关或不相关。精度是所返回的相关文档中所占比例的一部分，召回率是所返回的所有相关文档中所占比例的一部分。更正式地，让我们假设系统响应信息请求返回了 T 个排序的文档，其中的一个子集 R 是相关的，不相交的子集 N 是其余不相关的文档，并且集合中的 U 个文档总体上与 这个请求相关。然后将精度和召回率定义为：

$$\text{Precision} = \frac{|R|}{|T|} \qquad \text{Recall} = \frac{|R|}{|U|} \tag{23.11}$$

不幸的是，这些度量标准不能充分地测量对返回的文档进行排名的系统的性能。如果我们比较两个排名的检索系统的性能，我们需要一个更倾向于相关文档排名更高的指标。我们需要调整精度和召回率，以捕获系统在将相关文档置于更高排名方面做得有多好。

让我们来看一个例子。假设图 23.4 中的表格给出了特定于排名的精度和召回率的值，这些值在我们向下浏览特定查询的一组排序文档时被计算;精度是在给定级别上看到的相关文档的比例，而召回率是在相同级别上找到的相关文档的比例。本例中的召回率度量是基于这样一个查询，该查询有 9 个相关文档，这些文档作为一个整体的存在于集合中。

Rank	Judgment	$\text{Precision}_{\text{Rank}}$	$\text{Recall}_{\text{Rank}}$
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55
11	R	.55	.66
12	N	.50	.66
13	N	.46	.66
14	N	.43	.66
15	R	.47	.77
16	N	.44	.77
17	N	.44	.77
18	R	.44	.88
19	N	.42	.88
20	N	.40	.88
21	N	.38	.88
22	N	.36	.88
23	N	.35	.88
24	N	.33	.88
25	R	.36	1.0

图 23-4：针对排名的精度和召回率的计算

图注：当我们向下浏览一组已排名的文档时(假设集合有 9 个相关文档)，将计算特定于排名的精度和召回率值。

注意，召回率不是递减的;当遇到相关文件时，召回率增加；当发现非相关文件时，召回率保持不变。另一方面，精度会上下波动，当找到相关的文档时精度会增加，反之则会减少。可视化精度和召回率最常用的方法是将精度与召回率的关系绘制成**精度-召回率曲线(precision-recall curve)**，表 23.4 中的数据如图 23.5 所示。

图 23.5 显示了单个查询的值。但是，我们需要合并所有查询的值，并以一种可以将一个系统与另一个系统进行比较的方式进行合并。一种方法是绘制 11 个固定召回率(0 到 100，以 10 为步长)的平均精度值。由于我们不太可能拥有这些确切级别的数据点，因此我们对来自我们已有数据点的 11 个召回值使用**插值精度(interpolated precision)**值。为此，我们可以选择在任何召回水平上达到或高于我们计算出的最大召回率时所能达到的最大精度值。换一种说法，

$$\text{IntPrecision}(r) = \max_{i>=r} \text{Precision}(i) \tag{23.12}$$

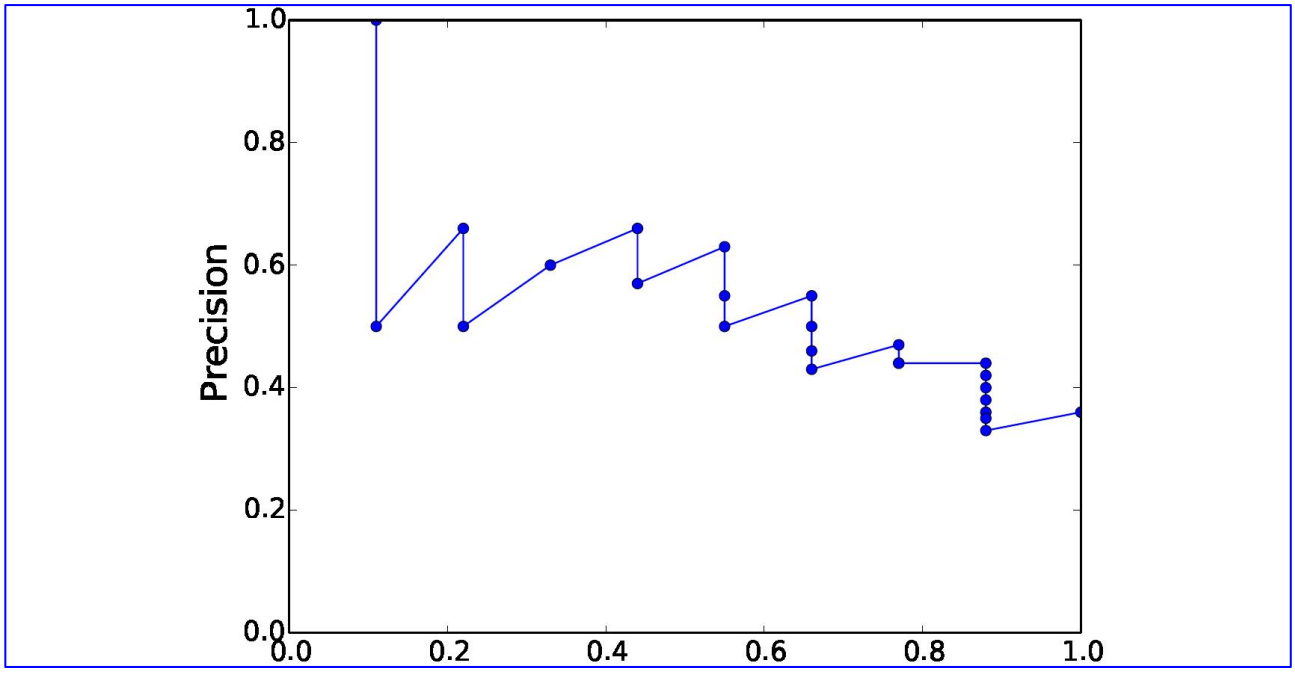


图 23-5：表 23.4 数据的精度和召回率的曲线

这种插值方案不仅可以让我们在一组查询上平均性能，而且还可以平滑原始数据中不规则的精度值。它旨在通过分配被测对象在较高的召回率水平下分配的最大精度值，来为系统带来疑惑的好处。图 23.6 和图 23.7 显示了从我们的示例中得到的内插数据点。

Interpolated Precision	Recall
1.0	0.0
1.0	.10
.66	.20
.66	.30
.66	.40
.63	.50
.55	.60
.47	.70
.44	.80
.36	.90
.36	1.0

图 23-6：从图 23.4 中插入数据点

给定曲线，如图 23.7 所示，我们可以通过比较两种系统或方法的曲线来比较它们。显然，在所有召回值中，更高精度的曲线是首选。然而，这些曲线也可以提供对系统整体行为的洞察。左边的精度较高的系统可能更倾向于精度而不是召回率，而更适合召回的系统将在较高的召回级别上更高(如右图所示)。

评估排名检索的第二种方法是**平均精度的均值(mean average precision，MAP)**，它提供了一个可用于比较竞争系统或方法的度量。通过这种方法，我们再次向下浏览术语的已排序列表，但现在我们仅在遇到相关术语的那些点上注意精度(例如，在图 23.4 中的排名 1、3、5、6 而不是 2 或 4 处)。对于单个查询，我们将这些单独的精度测量值平均化为返回值集合(直至某个固定的临界值)。更正式地说，如果我们假设 R_r 是等于或大于 r 的相关文档的集合，则单个查询的平均精度(AP)为：

$$\text{AP} = \frac{1}{|R_r|}\sum_{d\in R_r}\text{Precision}_i(d) \tag{23.13}$$

其中 $\text{Precision}_r(d)$是在文档 d 的位置上测量的精度。对于一系列的问题 Q，我们对这些平均值进行平均，得到我们最终的 MAP 测量结果:

$$\text{MAP} = \frac{1}{|Q|}\sum_{q\in Q}\text{AP}(q) \tag{23.14}$$

在图 23.4 中，单个查询(因此=AP)的映射是 0.6。

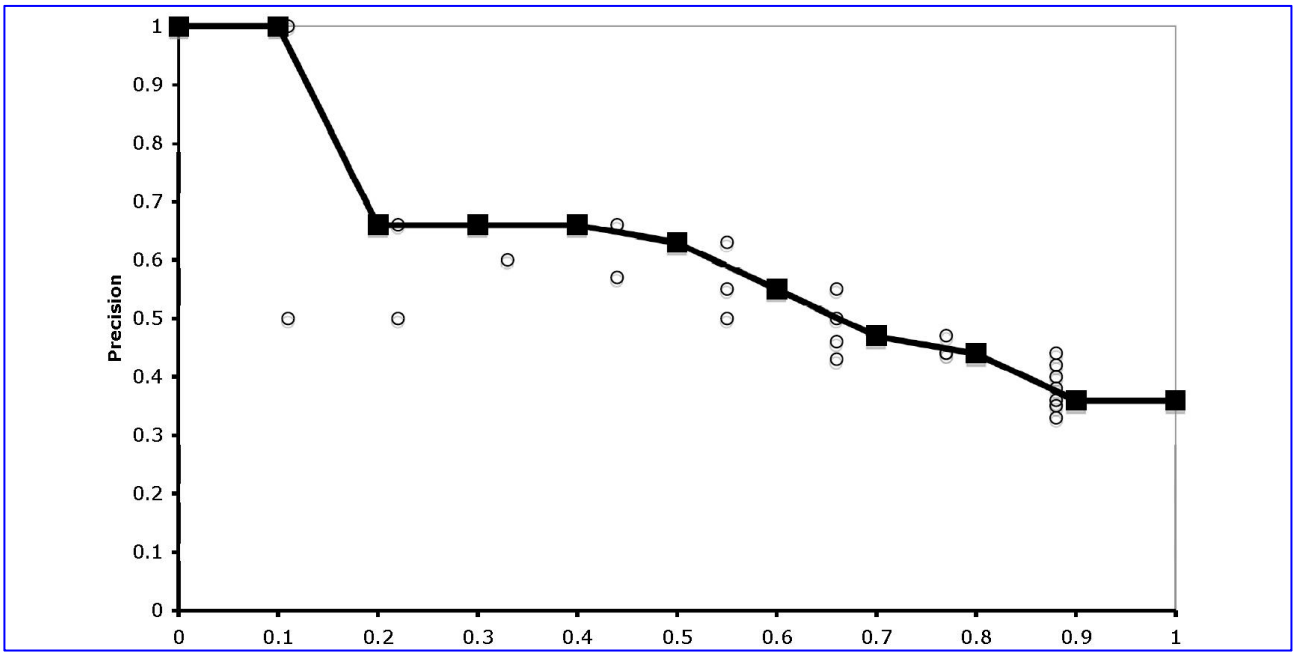


图 23-7：插值的精度-召回曲线

图注：11 个点的插值精度-召回曲线。对每一个查询，11 个标准召回级别的精度从任何更高级别召回的最大值内插。同时给出了原始测量精度召回点。

23.1.5. 稠密向量 IR

人们早就知道，用于 IR 的经典 tf-idf 或 BM25 算法有一个概念上的缺陷:它们只有在查询和文档之间有准确的单词重叠时才能工作。换句话说，提出查询(或询问问题)的用户需要准确猜测答案的作者可能使用了哪些词来讨论这个问题。正如 Lin 等人(2020)所说，用户可能会决定搜索一个悲剧的爱情故事(a tragic love story)，但莎士比亚却写了一个命运多舛的情侣(star-crossed lovers)。这被称为词汇不匹配问题(Furnas 等人，1987)。

这个问题的解决方案是使用一种可以处理同义词的方法:使用(密集的)嵌入，而不是(稀疏的)单词计数向量。这个想法很早就在 LSI 方法中被提出(Deerwester 等人，1990)，但是现代方法都使用像 BERT 这样的编码器。在所谓的双编码器(biencoder)中，我们使用了两个独立的编码器模型，一个编码查询，一个编码文档，并使用这两个向量之间的点积作为分数(图 23.8)。例如，如果我们使用 BERT，我们将有两个编码器 BERT Q 和 BERT D，可以将查询和文档表示为各自编码器的[CLS]符记(Karpukhin 等人，2020):

$$h_q = \text{BERT}_Q(q)[\text{CLS}]$$
$$h_d = \text{BERT}_D(d)[\text{CLS}]$$
$$score(d,q) = h_q \cdot h_d \quad (23.15)$$

更复杂的版本可以使用其他方式来表示编码的文本，例如对所有符记的 BERT 输出使用平均池化而不是使用 CLS 符记，或者可以在编码或点积步骤之后添加额外的权重矩阵(Liu 等人 2016，Lee 等人 2019)。

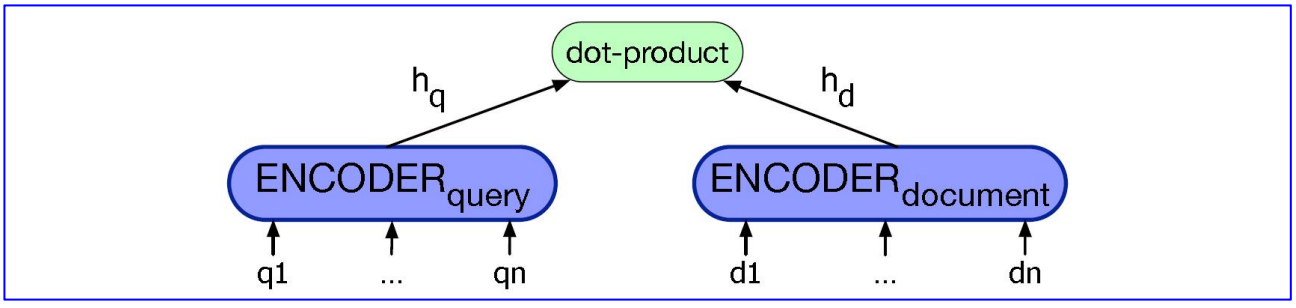


图 23-8：BERT 双编码器用于计算文档与查询的相关性

将密集的矢量用于 IR 或 QA 的检索组件，仍然是一个开放的研究领域。在活跃研究的众多领域中，如何对 IR 任务的编码器模块进行微调(通常通过对查询文档组合进行微调，并通过各种巧妙的方法来获得否定示例)，以及如何处理这一事实：文件通常比 BERT 等编码器可处理的更长(通常通过将文件分成段落)。

效率也是一个问题。每个 IR 引擎的核心都需要对每个可能的文档与查询的相似性进行排序。对于稀疏的单词计数向量，倒排索引可以非常有效地实现这一点。对于基于 BERT 或其他 Transformer 编码器的密集向量算法，寻找与密集查询向量有最高点积的密集文档向量集是最近邻搜索的一个例子。因此，现代系统使用近似最近邻向量搜索算法，如 **Faiss** (Johnson 等人，2017)。

23.2. 基于 IR 的事实性问答

基于 IR 的 QA(有时称为开放域 QA)的目标是通过从 web 或其他大型文档集合中找到短文本片段来回答用户的问题。图 23.9 显示了一些示例事实性问题及其答案。

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What are the names of Odin's ravens?	Huginn and Muninn
What kind of nuts are used in marzipan?	almonds
What instrument did Max Roach play?	drums
What's the official language of Algeria?	Arabic

图 23-9：一些事实性问题及其答案

基于 IR 的 QA 的主要范式是图 23.10 所示的**检索和阅读(retrieve and read)**模型。在这个两阶段模型的第一阶段中，我们从一个文本集合中检索相关的段落，通常使用我们在前一节中看到的类型的搜索引擎。在第二阶段，一个神经阅读理解算法会遍历每一篇文章，找出可能回答问题的跨度。

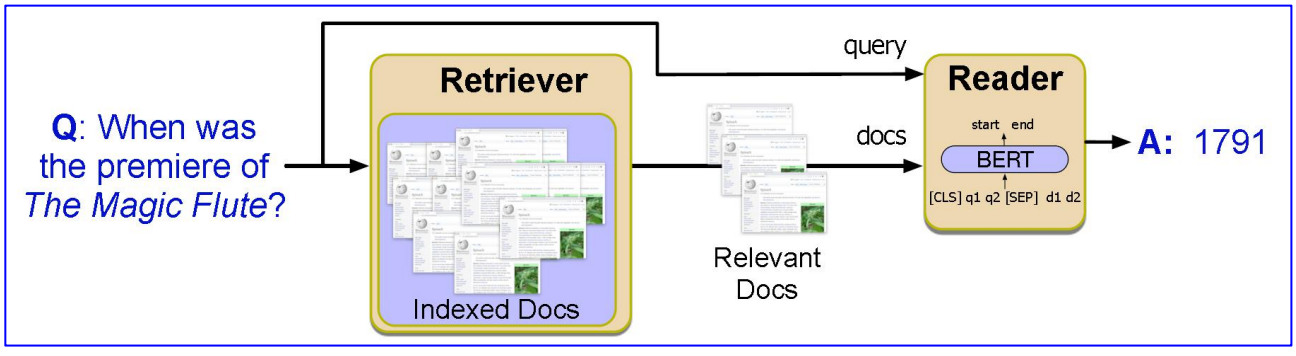


图 23-10：基于 IR 的检索和阅读模型

图注：基于 IR 的事实性 QA 有两个阶段:检索，从集合中返回相关的文档；阅读，其中一个神经阅读理解系统提取答案跨度。

一些问答系统仅专注于第二项任务，即**阅读理解(reading comprehension)**任务。阅读理解系统会得到一个事实性问题 q 和一个可能包含答案的段落 p，然后返回答案 s(或者声明段落中没有答案，或者在某些设置中从一组可能的答案中进行选择)。当然，此设置与满足他们需要回答的问题的用户的信息需求不符(毕竟，如果用户知道哪个段落包含了答案，他们就可以自己阅读)。取而代之的是，该任务最初是根据儿童的阅读理解测试(一种使儿童获得阅读能力并必须回答有关问题的教学手段)而建立的，以此作为评估自然语言理解能力的一种方式(Hirschman 等人，1999)。阅读理解系统仍以这种方式使用，但也已发展成为现代检索和阅读模型的第二阶段。

其他的问答系统解决整个检索和阅读任务;给他们一个事实性的问题和一个大型文档集合(如 Wikipedia 或 web 检索)，并返回一个答案，通常是从文档中提取的一段文本。这个任务通常被称为开放域 QA。在接下来的几节中，我们将介绍基于 IR 的 QA 的各个部分，首先介绍一些用于阅读理解和完整 QA 任务的常用数据集。

23.2.1. 基于 IR 的 QA：数据集

基于 IR 的 QA 的数据集通常是通过首先开发包含(段落，问题，答案)元组的阅读理解数据集来创建的。阅读理解系统可以使用数据集来训练被授予段落和问题的读者，并预测该段落的一个跨度作为答案。包括要从中提取答案的段落，就无需使用阅读理解系统来处理 IR。

例如，**斯坦福问答数据集(Stanford Question Answering Dataset，SQuAD)**由来自维基百科的文章和相关问题组成，这些问题的答案是来自文章的跨度(Rajpurkar 等人，2016)。SQuAD 2.0 还增加了一些设计成无法回答的问题(Rajpurkar 等人，2018)，总共有超过 15 万个问题。图 23.11 显示了 SQuAD 2.0 的一段节选，包含 3 个问题和黄金答案。

SQuAD 是通过让人类阅读给定的维基百科文章，写下关于这篇文章的问题，并选择一个特定的答案来建立的。

其他数据集是通过类似的技术创建的，但是尝试使问题变得更加复杂。**HotpotQA** 数据集(Yang 等人，2018)是通过向众包工作者显示多个上下文文档而创建的，并要求提出需要对所有文档进行推理的问题。

像 SQuAD 或 HotpotQA 这样的数据集中的问题是由首先阅读段落的评注者创建的，这可能使他们的问题更容易回答，因为评注者可能(下意识地)使用了答案文本中的单词。

自然问题(Natural Questions)数据集(Kwiatkowski 等人，2019)将真正的匿名查询合并到 Google 搜索引擎中。一个查询以及来自前 5 个搜索结果的 Wikipedia 页面被呈送到评注者面前，然后评注者注释一个段落篇幅的长答案和一个短跨度的答案。如果文本不包含该段落，则标记为 null。例如，问题“何时将啤酒花添加到酿造过程中？(When are hops added to the brewing process?)”有一个短答案“沸腾过程(the boiling process)”和一个长答案：Brewing 维基百科页面上的整个段落。在使用此数据集时，阅读理解模型会得到一个问题和一个 Wikipedia 页面，并且必须返回长答案、短答案或“无答案”回应。

解决这种可能偏见的一种常见解决方案是从某些问题中建立数据集，这些问题不是特意针对某一篇文章而被写下来的。TriviaQA 数据集(Joshi 等人 2017)包含由琐事爱好者编写的 94K 个问题，以及来自 Wikipedia 和网络的辅助文档，从而产生了 65 万个问题-答案-证据的三元组。

Beyonc´e Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyonc´e's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Q: "In what city and state did Beyonc´e grow up?" A: "Houston, Texas"
Q: "What areas did Beyonc´e compete in when she was growing up?" A: "singing and dancing"
Q: "When did Beyonc´e release Dangerously in Love?" A: "2003"

图 23-11：一篇包含问题和答案的文章

图注：一篇(维基百科)文章来自 SQuAD 2.0 数据集(Rajpurkar 等人 2018)，包含 3 个样本问题和标注答案的跨度。

以上数据集均为英文。TyDi QA 数据集包含来自 11 种类型多样语言的 204K 问答配对，包括阿拉伯语、孟加拉语、斯瓦希里语、俄语和泰语(Clark 等人 2020)。在 TyDi QA 任务中，一个问题和一些来自维基百科的文章被给到系统，系统必须(a)选择包含答案的文章(如果没有文章包含答案，则为 null)，(b)标记答案的最小跨度(或为空)。很多问题都没有答案。数据集中的各种语言给 QA 系统带来了挑战，比如问题和答案之间的形态变化，或者单词分割或多种字母的复杂问题。

在阅读理解任务中，系统会给出一个问题，并在其中找到答案。然而，在完整的两阶段 QA 任务中，系统并没有给出一篇文章，而是被要求自己从一些文档集合中进行检索。创建开放领域 QA 数据集的常见方法是修改阅读理解数据集。出于研究目的，这通常是通过使用注释 Wikipedia 的 QA 数据集(如 SQuAD

或 HotpotQA)来实现的。对于训练，整个(问题，短文，答案)三元组用来训练读者。但在推理时，这些段落被删除，系统只给出问题，并访问整个维基百科语料库。然后系统必须进行 IR，以找到一组页面，然后读取它们。

23.2.2. 基于 IR 的 QA：阅读器(答案跨度提取)

基于 IR 的 QA 的第一个阶段是检索器，例如我们在 23.1 节中看到的类型。基于 IR 的 QA 的第二阶段是阅读器。阅读器的工作是把一篇文章作为输入并给出答案。在我们这里讨论的**提取的 QA(extractive QA)**中，答案是文章中的文本的一个跨度[4]。例如，给出“How tall is Mt. Everest?”这样一个问题，以及一个包含 Reaching 29,029 feet at its summit 从句的段落，则阅读器输出 29,029 feet。

答案抽取任务通常采用跨度标注模型:在文章中标识构成答案的**跨度(span)**(连续的文本字符串)。阅读理解的神经算法，给出了一个包含 n 个符记 q_1，...，q_n 的问题 q 和一个包含 m 个符记 p_1，...，p_m 的段落 p。因此，它们的目标是计算每个可能的跨度 a 是答案的概率 P(a|q, p)。

如果每个跨度 a 从 a_s 的位置开始并在 a_e 的位置结束，则我们作出简化的假设，即可以将该概率估计为 $P(a|q, p)= P_{start}(a_s|q, p)P_{end}(a_e|q, p)$。因此，对于段落中的每个符记 p_i，我们将计算两个概率：$p_{start}(i)$ 表示 p_i 是答案跨度的开始，而 $p_{end}(i)$表示 p_i 是答案跨度的结束。用于阅读理解的标准基线算法是将问题和段落传递给像 BERT 这样的编码器(图 23.12)，当作用[SEP]符记分隔的字符串，导致每个段落符记都嵌入一个编码符记。

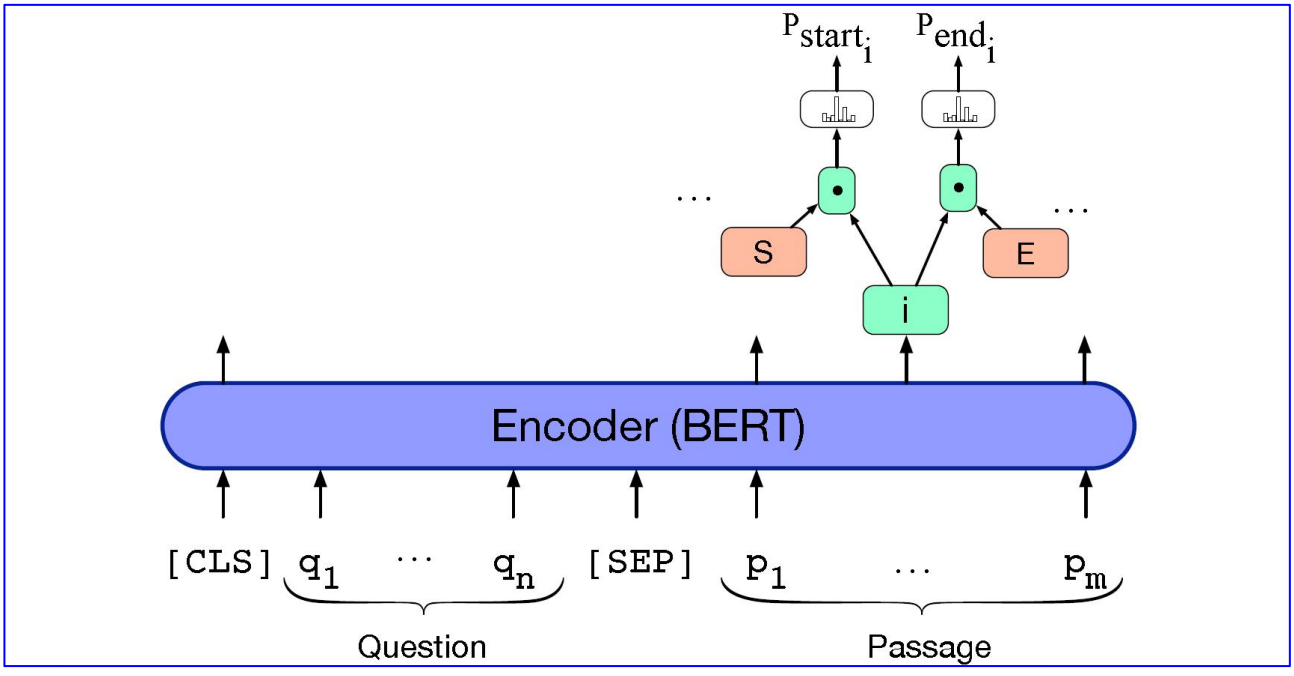


图 23-12：基于跨度的 QA 编码器模型(使用 BERT)

图注：一个编码器模型(使用 BERT)，用于基于跨度的 QA，来自基于阅读理解的 QA 任务。

对于基于跨度的问答，我们将问题表示为第一个序列，将文章表示为第二个序列。我们还需要添加一个线性层，该层将在微调阶段接受训练，以预测跨度的起始和结束位置。我们将添加两个新的特殊向量:一个跨度开始嵌入 S，一个跨度结束嵌入 E，它们将在微调中学到。为了得到每个输出符记 p_i'的跨度开始概率，我们计算 S 和 p_i'之间的点积，然后使用 softmax 对文中所有符记 p_i'进行归一化:

$$P_{\text{start}_i} = \frac{\exp(S \cdot p_i')}{\sum_j \exp(S \cdot p_j')} \tag{23.16}$$

我们用类似的方法来计算跨度结束的概率:

[4] 在这里，我们跳过更困难的抽象化的 QA 任务，在这个任务中，系统可以写出一个不是完全从文章中提取的答案。

$$P_{\text{end}_i} = \frac{\exp(E \cdot p'_i)}{\sum_j \exp(E \cdot p'_j)} \tag{23.17}$$

从位置 i 到位置 j 的候选跨度的得分为 $S \cdot p_i' + E \cdot p_j'$，选择 j≥i 的候选跨度的得分最高的就是模型预测。微调的训练损失是每个实例正确的开始和结束位置的对数概率的总和的负数:

$$L = -\log P_{\text{start}_i} - \log P_{\text{end}_i} \tag{23.18}$$

许多数据集(如 SQuAD 2.0 和 Natural Questions)也包含成对的(question, passage)，其中的 passage 不包括答案。因此，我们还需要一种方法来估计问题的答案不在文档中的可能性。标准做法是，将没有答案的问题视为以[CLS]符记作为答案，因此答案跨度的起始和结束索引将指向[CLS] (Devlin 等人，2019)。

对于许多数据集，我们还需要处理带注释的文档/段落长于 BERT 允许的最大 512 个输入符记的情况。考虑例如自然问题之类的案例，其中带有金色标签的段落是完整的 Wikipedia 页面。在这种情况下，我们可以从标签的 Wikipedia 页面创建多个伪段落观察。每个观察结果是通过将[CLS]、问题、[SEP]和文档中的符记拼接在一起形成的。我们遍历文档，滑动一个大小为 512(或 512 减去问题长度 n、再减去特殊符记)的窗口，并将符记的窗口包装到每个下一个伪通道中。观测值的答案跨度，要么标签为[CLS](在此特定窗口中为没有答案)，要么标记为黄金标签的跨度。Alberti 等人(2019)建议通过使用 128 个符记的跨度来允许窗口重叠。可以使用相同的过程进行推理，将每个检索到的文档分解为单独的观察段落并标签每个观察结果。可以将答案选择为概率最高的跨度(如果没有比[CLS]更大的跨度，则选择 nil)。或者，Alberti 等人(2019)建议通过 nil 分数将开始 s 和结束 e 的跨度的每个分数 g(s，e)进行归一化：

$$g(s,e) = \text{start}_i + \log P_{\text{end}_i} \text{start}_{\text{CLS}} + \log P_{\text{end}_{\text{CLS}}} \tag{23.19}$$

23.3. 实体连接

我们现在看到了第一个主要的问答范式，基于 IR 的 QA。在讨论第二个主要的问答模式——基于知识的问答之前，我们先介绍实体连接这一重要的核心技术，因为任何基于知识的问答算法都需要实体连接。

实体连接(entity linking)是将文本中的提及与本体中真实实体的表示相关联的任务(Ji 和 Grishman, 2011)。

最常见的事实性问答本体是维基百科，因为维基百科通常是回答问题的文本的来源。在这种用法中，每个惟一的 Wikipedia 页面充当特定实体的惟一 ID。这里有一个任务：决定哪一个与个体相对应的 Wikipedia 页面被一个文本提及所指代。这个决定的任务有自己的名字：**维基化(wikification)**(Mihalcea 和 Csomai，2007)。

自最早的系统(Mihalcea 和 Csomai 2007，Cucerzan 2007，Milne 和 Witten 2008)以来，实体连接(大致)分两个阶段完成：提及检测和提及消歧。我们将提供两种算法：一种是使用锚定词典和 Wikipedia 图结构中的信息的简单经典基线(Ferragina 和 Scaiella，2011)；另一种是现代神经算法(Li 等人，2020)。在这里，我们主要关注实体连接到问题而不是其他体裁的应用。

23.3.1. 基于锚字典和 Web 图的连接

作为简单的基准，我们介绍了 Wikipedia 的 TAGME 连接器(Ferragina 和 Scaiella，2011)，该连接器本身借鉴了较早的算法(Mihalcea 和 Csomai 2007，Cucerzan 2007，Milne 和 Witten 2008)。维基化算法将实体集定义为 Wikipedia 页面集，因此我们将每个 Wikipedia 页面称为唯一实体 e。TAGME 首先创建所有实体的目录(即所有 Wikipedia 页面，删除一些歧义和其他元页面)，然后在像 Lucene 这样的标准 IR 引擎中为它们建立索引。对于每个 e 页面，该算法都会计算一个 in-link 的计数 in(e)：来自其他 Wikipedia 页面的指向 e 的 in-link 总数。这些计数可以从 Wikipedia 转储中得出。

最后，该算法需要一个锚字典。一个锚字典列出了每个维基百科页面，它的锚文本(anchor texts):在其他页面上指向它的超连接跨度。例如，网页为斯坦福大学，http://www.stanford.edu，可能指向从另一个

页面使用锚文本诸如 Stanford 或者 Stanford University:

Stanford University

对每个 Wikipedia 页面 e，通过包含 e 的标题以及所有 Wikipedia 页面上指向 e 的所有锚文本的方式，我们计算 Wikipedia 锚字典。对于每个锚定字符串 a，我们还将在 Wikipedia 中计算其总频率 freq(a)(包括非锚定用法)，a 作为连接出现的次数(我们将其称为 link(a))及其 连接概率 linkprob(a)= link(a)/ freq(a)。需要对最终的锚字典进行一些清理，例如，删除仅由数字或单个字符组成的锚字符串，这些字符串非常罕见，或者由于连接概率很低而不太可能成为有用的实体。

提及检测

给定一个问题(或我们尝试连接的其他文本)，TAGME 通过查询锚定词典中最多 6 个单词的每个符记序列来检测提及。可以使用一些简单的启发式方法(例如，如果子字符串具有较小的连接概率，则对子字符串进行修剪)对这套大序列的修剪进行修剪。问题：

When was Ada Lovelace born?

可能会产生锚节点 Ada Lovelace，也可能是 Ada，但是像 Lovelace 这样的子字符串跨度可能会因为连接概率太低而被修剪，但是像 born 这样的跨度的连接概率太低，它们根本不会出现在锚节点字典中。

提及消歧

如果提及跨度是没有歧义的(仅指向一个实体/维基百科页面)，则完成实体连接！ 但是，许多跨度是有歧义的，可以匹配多个 Wikipedia 实体/页面的锚点。TAGME 算法使用两个因素消除跨度的歧义，这被称为先验概率和相关性/连贯性。第一个因子是 p(e | a)，即跨度指代特定实体的概率。对于每个页面 e∈E(a)，将 a 锚定到 e 的概率 p(e | a)是具有锚定文本 a 的 e 中连接的数量与 a 作为锚点出现的总数的比率：

$$\text{prior}(a \to e) \;=\; p(e|a) = \frac{\text{count}(a \to e)}{\text{link}(a)} \tag{23.20}$$

让我们看看这个因素是如何在以下问题中连接实体的:

What Chinese Dynasty came before the Yuan?

在锚字典中，跨度 Yuan 最常见的联想就是中国货币的名称，即概率 p(Yuan_currency| yuan)非常高。比较少见的与 Yuan 相关的维基百科词条包括常见的中国姓氏(泰国人说的一种语言)，以及正确的实体——中国朝代的名称。所以如果我们只根据 p(e|a)来选择，就会造成错误的消歧，错过正确的环节:元朝。

为了在这种情况下提供帮助，TAGME 使用了第二个因素，即该实体与输入问题中其他实体的相关性。在我们的示例中，问题还包含跨度 Chinese Dynasty 这一事实，该事实与 Dynasties_in_Chinese_history 页面有很高的关联性，应该有助于与 Yuan_dynasty 相匹配。

让我们看看它是如何工作的。给定一个问题 q，对于在 q 中检测到的每个候选锚点，我们为 a 的每个可能实体 e∈Ɛ (a)分配一个相关性得分。连接 a→e 的相关性得分是 e 与 q 中所有其他实体之间的加权平均相关性。两个实体在其 Wikipedia 页面共享许多连接的范围内被认为是相关的。更正式地说，两个实体 A 和 B 之间的相关性计算为：

$$\text{rel}(A,B) \;=\; \frac{\log(\max(|\text{in}(A)|,|\text{in}(B)|)) - \log(|\text{in}(A)| \cap |\text{in}(B)|)}{\log(|W|) - \log(\min(|\text{in}(A)|,|\text{in}(B)|))} \tag{23.21}$$

其中 in(x)是指向 x 的 Wikipedia 页面的集合，W 是汇集(collection)中所有 Wikipedia 页面的集合。锚点 b 对候选注释 a→X 的投票是 b 的所有可能实体与 X 的关联度的平均值，并按其先验概率加权：

$$\text{vote}(b,X) = \frac{1}{|\mathcal{E}(b)|} \sum_{Y \in \mathcal{E}(b)} \text{rel}(X,Y)\, p(Y|b) \tag{23.22}$$

a→X 的总关联度得分是 q 中检测到的所有其他锚点的投票总和:

$$\text{relatedness}(a \to X) = \sum_{b \in \mathcal{X}_q \setminus a} \text{vote} b, X \tag{23.23}$$

为了给 a→X 评分，我们将关联度和先验结合起来，选择关联度最高的实体 X (a→X)，在这个值的小ϵ范围内寻找其他实体，形成这个集合，并从这个集合中选择先验 P(X|a)最高的实体。此步骤的结果是单个实体被分配给 q 中的每个跨度。

TAGME 算法还有另外一个步骤：修剪虚假的锚点/实体配对，取连接概率与连贯性的平均值，再把这个平均值分配给分数：

$$\text{coherence}(a \to X) = \frac{1}{|S| - 1} \sum_{B \in \mathcal{S} \setminus X} \text{rel}(B, X)$$

$$\text{score}(a \to X) = \frac{\text{coherence}(a \to X) + \text{linkprob}(a)}{2} \tag{23.24}$$

最后，如果 score(a→X)<λ，则修剪配对，其中阈值λ设置在保留集上。

23.3.2. 基于神经图的连接

最近的实体连接模型基于双编码器，对候选提及跨度进行编码，对实体进行编码，并计算编码之间的点积。这允许对知识库中所有实体的嵌入进行预先计算和缓存(Wu 等人，2019)。让我们来概述一下 Li 等人(2020)的 ELQ 连接算法。给它一个问题 q 和一组来自 Wikipedia 的候选实体以及相关的 Wikipedia 文本，并输出实体 ID 的元组(e，ms，me)、提及开始和提及结束。如图 23.13 所示，它是通过使用来自 Wikipedia 的文本对每个 Wikipedia 实体进行编码，使用来自问题的文本对每个提及跨度进行编码，以及计算它们的相似度来实现的，如下所述。

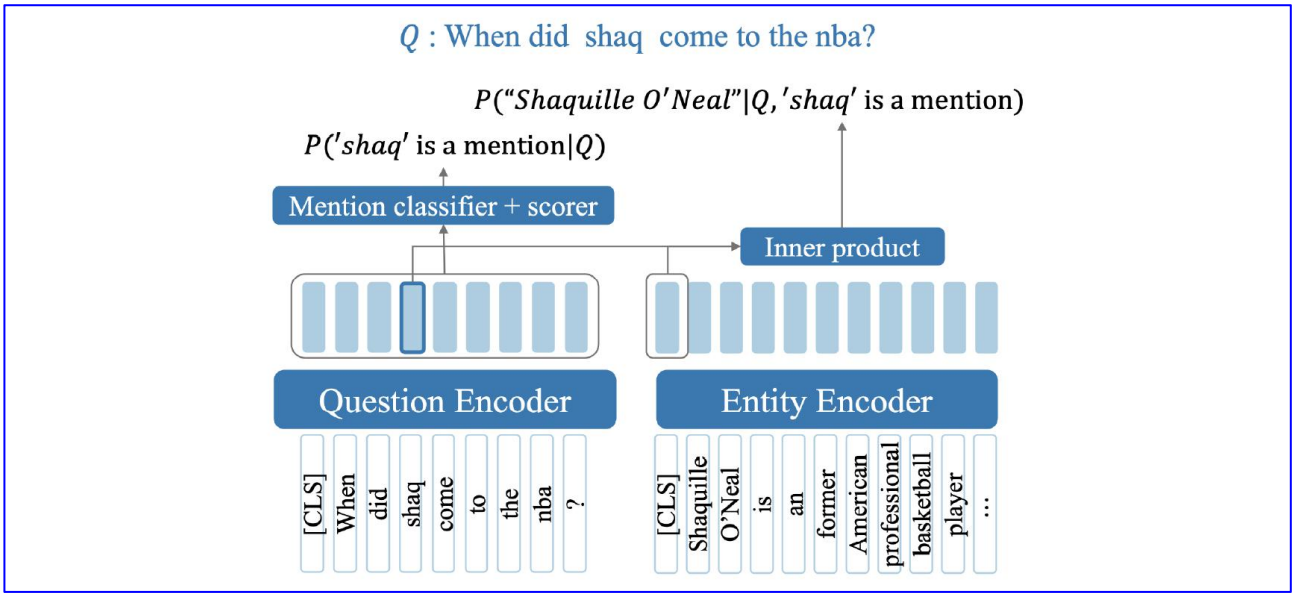


图 23-13：ELQ 算法推理过程示意图

图注：ELQ 算法中用于问题实体连接的推理过程示意图(Li 等人，2020 年)。每个候选问题提及跨度和候选实体分别编码，然后由实体/跨度点积评分。

实体提及检测

为了获得每个问题符记的 h 维嵌入，该算法以常规方式通过 BERT 运行问题：

$$[\mathbf{q}_1 \ldots \mathbf{q}_n] = \text{BERT}([\text{CLS}]q_1 \ldots q_n[\text{SEP}]). \tag{23.25}$$

它计算出 g 中每个跨度[i.j]成为实体提及的似然度，以类似于上面为读者所见的基于跨度的算法的方式。首先，我们计算 i/j 作为提及开始/结束的分数：

$$\mathbf{S}_{\text{start}}(i) = \mathbf{W}_{\text{start}} \cdot \mathbf{q}_i \,, \quad \mathbf{S}_{\text{end}}(j) = \mathbf{W}_{\text{end}} \cdot \mathbf{q}_j \tag{23.26}$$

其中 w_{start} 和 w_{end} 是训练过程中学习的向量。接下来，另一个可训练的嵌入，$w_{mention}$ 被用来计算每一个被提及的符记的得分:

$$\mathbf{S}_{\text{mention}}(t) = \mathbf{W}_{\text{mention}} \cdot \mathbf{q}_t \tag{23.27}$$

然后，通过结合以下三个得分来计算提及概率：

$$p([i,j]) = \sigma\left(s_{\text{start}}(i) + s_{\text{end}}(j) + \sum_{t=i}^{j} s_{\text{mention}}(t)\right) \tag{23.28}$$

实体连接

为了将提及连接到实体，我们接下来为所有 Wikipedia 实体的集合 E = e_1，…，e_i，…，e_w 中的每个实体计算嵌入。对于每个实体 e_i，我们将从实体的 Wikipedia 页面中获取文本，标题为 $t(e_i)$，Wikipedia 页面的前 128 个符记称为描述 $d(ei)$。再通过 BERT 运行，将 CLS 符记 BERT [CLS]的输出作为实体表示：

$$\mathbf{x}_e = \text{BERT}_{[\text{CLS}]}([\text{CLS}]t(e_i)[\text{ENT}]d(e_i)[\text{SEP}]) \tag{23.29}$$

提及跨度可以通过计算连接到实体，其计算方式是：对于每个实体 e 和跨度[i，j]，在跨度编码(符记嵌入的平均值)和实体编码之间，计算点积的相似度。

$$\mathbf{y}_{i,j} = \frac{1}{(j-i+1)} \sum_{t-i}^{j} \mathbf{q}_t$$

$$s(e,[i,j]) = \mathbf{x}_e^{\cdot} \mathbf{y}_{i,j} \tag{23.30}$$

最后，我们使用 softmax 来获得每个跨度的实体分布:

$$p(e|[i,j]) = \frac{\exp(s(e,[i,j]))}{\sum_{e' \in \mathcal{E}} \exp(s(e',[i,j]))} \tag{23.31}$$

训练 ELQ 提及检测和实体连接算法是全监督算法。这意味着，与第 23.3.1 节中的锚定字典算法不同，它需要标记和连接实体边界的数据集。两个这样的标签数据集是 WebQuestionsSP (Yih 等人，2016)，它是来自谷歌搜索问题的 WebQuestions (Berant 等人，2013)数据集的扩展，和 GraphQuestions (Su 等人 2016)。两者在标记和连接的问题中都有实体跨度(Sorokin 和 Gurevych 2018, Li 等人 2020)，导致实体标签版本 WebQSPEL 和 GraphQEL (Li 等人，2020)。在给定训练集的基础上，对 ELQ 提及检测和实体连接阶段进行联合训练，使它们的损失之和达到最优。提及检测损失是一种二元交叉熵损失：

$$\mathcal{L}_{\text{MD}} = -\frac{1}{N} \sum_{1 \le i \le j \le min(i+L-1,n)} \left(y - [i,j] \log p([i,j]) + (1 = y_{[i,j]}) \log(1 - p([i,j]))\right) \tag{23.32}$$

如果[i, j]是一个黄金提及跨度，则 $y_{[i,j]} = 1$，否则为 0。实体连接损失为:

$$\mathcal{L}_{\text{ED}} = -logp(e_g \,|\, [i,j]) \tag{23.33}$$

其中 e_g 为提及的黄金实体[i,j]。

关于 QA 之外的实体连接的其他应用的更多讨论见本章末尾。

23.4. 基于知识的 QA

虽然网络上大量的文本编码了大量的信息，但信息显然也以更结构化的形式存在。我们使用“基于知识的 QA”这个术语是为了通过将自然语言问题映射到结构化数据库上的查询来回答自然语言问题。与基于文本的问题回答范式一样，这种方法可以追溯到自然语言处理的早期，比如像 BASEBALL(Green 等人，1961)这样的系统可以从棒球比赛和统计数据的结构化数据库中回答问题。

基于知识的 QA 使用了两种常见的范式：第一种是基于图的 QA，将知识库建模为图，通常将实体作为节点，将关系或命题作为节点之间的边；第二种是语义解析的 QA，使用我们在第 16 章中看到的语义解析方法。这两种方法都需要某种我们在前一节中描述的实体连接。

23.4.1. 来自 RDF 三元组存储的基于知识的 QA

在完成实体连接之后，让我们介绍一个简单的基于知识的 QA 系统的组件。我们将重点关注基于图的 QA 的最简单的例子，其中数据集是 RDF 三元组形式的一组事实，任务是回答关于其中一个缺失论元的问题。回想一下第 17 章，RDF 是三元组：一个谓词和两个论元，表示一些简单的关系或命题。流行的这种本体通常来自维基百科;DBpedia(Bizer 等人，2009)有超过 20 亿 RDF 三元组，或 Freebase (Bollacker 等人，2008)，现在是 Wikidata 的一部分(Vrandečić和 Krötzsch, 2014)。考虑如下的 RDE 三元组:

主语	谓语	宾语
Ada Lovelace	birth-year	1815

这个三元组可以用来回答诸如“When was Ada Lovelace born?”或“When was Ada Lovelace born?”

存在许多这样的问题数据集。SimpleQuestions(Bordes 等人，2015)包含注释者基于 Freebase 的三元组编写的 10 万个问题。例如，问题“What American cartoonist is the creator of Andy Lippincott?”是根据三元组 (andy lippincott, character created by, garry trudeau)编写的。FreebaseQA(Jiang 等人，2019)将 TriviaQA(Joshi 等人，2017)和其他来源的琐事问题与 Freebase 中的三元组对齐，例如将琐事问题“Which 18th century author wrote Clarissa (or The Character History of a Young Lady), said to be the longest novel in the English language? ”与三元组 (Clarissa, book.written-work.author, Samuel Richardson)对齐。另一个这样的数据集家族来自 WEBQUESTIONS(Berant 等人，2013)，其中包含 5,810 个由网络用户提出的问题，每个问题都以 wh-字开头，仅包含一个实体，并与从 Freebase 页面的手写答案配对问题的实体。WEBQUESTIONSSP(Yih 等人 2016)通过人为创建的语义解析(SPARQL 查询)增强了 WEBQUESTIONS 的使用 Freebase 可回答的那些问题。COMPLEXWEB-QUESTIONS 通过组合和其他类型的复杂问题来增强数据集，从而产生 34,689 个问题以及答案、网络摘要和 SPARQL 查询(Talmor 和 Berant，2018)。

假设我们已经完成了上一节介绍的实体连接阶段。因此，我们已经将 Ada Lovelace 之类的文字说明映射到知识库中的规范实体 ID。对于简单的三元组关系问题回答，下一步是确定要询问的关系，从诸如“When was ... born”的字符串映射到知识库中的规范关系诸如 birth-year。可以将合并后的任务勾画为：

“When was Ada Lovelace born?” → birth-year (Ada Lovelace, ?x)

“What is the capital of England?” → capital-city(?x, England)

下一步是关系检测和连接。对于简单的问题，我们假设问题只有一个关系，关系检测和连接可以以类似神经实体连接模型的方式进行:计算问题文本的编码和每个可能关系的编码之间的相似度(通常通过点积)。例如，在(Lukovnikov 等人，2019)的算法中，为了关系检测的目的，使用 BERT 模型的 CLS 输出来表示问题跨度，并为每个关系 r_i 训练一个单独的向量。然后通过 softmax 对点积计算特定关系 r_i 的概率:

$$\mathbf{m}_r = \text{BERT}_{\text{CLS}}([\text{CLS}]q_1 \cdots q_n[\text{SEP}])$$

$$s(\mathbf{m_r}, r_i) = \mathbf{m_r} \cdot \mathbf{w_{r_i}}$$

$$p(r_i|q_1, \cdots, q_n) = \frac{\exp(s(\mathbf{m_r}, r_i))}{\sum_{k=1} N_R \exp(s(\mathbf{m_r}, r_k))} \tag{23.34}$$

答案的排名

大多数算法都有最后一个阶段，该阶段取实体和关系推理步骤返回的最前面的 j 个实体和最前面的 k 个关系，在知识库中搜索包含这些实体和关系的三元组，然后对这些三元组进行排名。这个排名可以是启发式的，例如，根据提及跨度和实体文本别名之间的字符串相似性来为每个实体/关系配对评分，或者优先考虑具有高的 in-degree(由许多关系连接)的实体。或者，可以通过训练分类器以采用拼接的实体/关系编码并预测概率来完成排名。

23.4.2. 语义解析的 QA

第二类基于知识的 QA 使用语义解析器将问题映射到结构化程序以生成答案。这些逻辑形式可用谓词演算的某些版本，一种查询语言如 SQL 或 SPARQL，或其他一些可执行程序的形式，如图 23.14 所示。

Question	Logical form
What states border Texas?	λx:state(x)∧borders(x; texas)
What is the largest state?	argmax(λx:state(x);λx:size(x))
I'd like to book a flight from San Diego to Toronto	SELECT DISTINCT f1.flight id FROM flight f1, airport service a1, city c1, airport service a2, city c2 WHERE f1.from airport=a1.airport code AND a1.city code=c1.city code AND c1.city name= 'san diego' AND f1.to airport=a2.airport code AND a2.city code=c2.city code AND c2.city name= 'toronto'
How many people survived the sinking of the Titanic?	(count (!fb:event.disaster.survivors fb:en.sinking of the titanic))
How many yards longer was Johnson's longest touchdown compared to his shortest touchdown of the first quarter?	ARITHMETIC diff(SELECT num(ARGMAX(SELECT)) SELECT num(ARGMIN(FILTER(SELECT))))

图 23-14：语义解析器为回答问题而生成的示例逻辑形式

图注：其中包括来自 GeoQuery 美国地理问题数据库中的两个问题(Zelle 和 Mooney，1996)带有谓词演算表示形式，一个 ATIS 问题带有 SQL(Iyer 等人，2017)，一个 Freebase 关系的程序，一个 QDMR 程序（Wolfson 等人，2020)。

因此，问题的逻辑形式要么是查询形式，要么很容易转换为查询形式(例如，谓词演算可以转换为 SQL)。数据库可以是一个完整的关系数据库，也可以是其他一些结构化的知识库。

正如我们在第 16 章中所看到的那样，语义解析算法可以由与手工构建的逻辑形式配对的问题进行完全监督，或者由与答案(表示法)配对的问题进行弱监督，其中逻辑形式仅被建模为潜在变量。

对于完全受监管的情况，我们可以从诸如以下三种数据集中获得一组问题及其正确的逻辑形式，第一是有关美国地理问题的 GEOQUERY 数据集(Zelle 和 Mooney，1996)，第二是需要推理的复杂问题(在历史和足球比赛方面)的 DROP 数据集(Dua 等人，2019)，第三是航班查询的 ATIS 数据集。所有版本的数据集都有 SQL 或其他逻辑形式的版本(Iyer 等人 2017, Wolfson 等人 2020, Oren 等人 2020)。

接下来的任务就是利用这些成对的训练元组，生成一个系统，将新问题映射到它们的逻辑形式。常见的基线算法是一个简单的序列到序列模型，例如使用 BERT 来表示问题符记，将它们传递给第 11 章的

biLSTM 编码器解码器，如图 23.15 所示。

第 16 章中描述的任何其他语义解析算法也都是合适的。

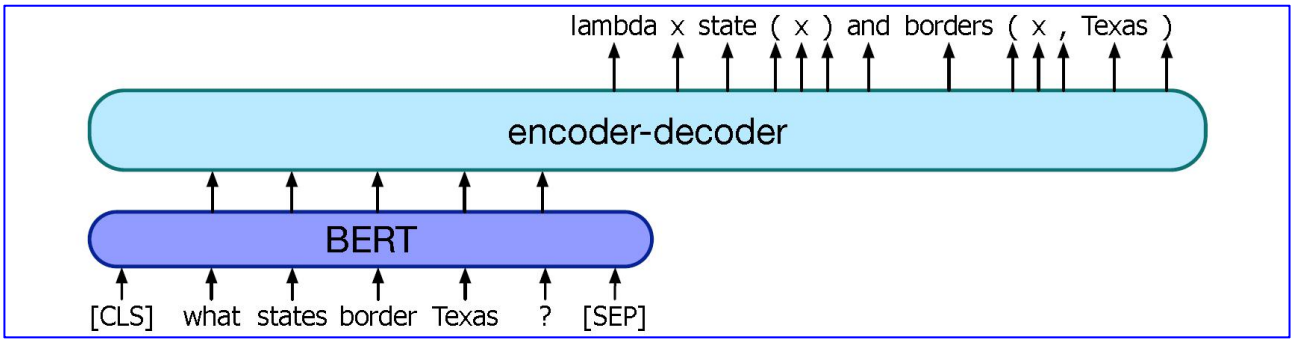


图 23-15：编码器-解码器语义解析器

图注：编码器-解码器语义解析器，用于将问题转换为逻辑形式，带有一个 BERT 预编码器和一个编码器-解码器(biLSTM 或 Transformer)。

23.5. QA 中的语言模型(T5)

进行 QA 的另一种方法是查询预训练的语言模型，迫使模型仅从其参数中存储的信息来回答问题。例如 Roberts 等人(2020 年)使用 T5 语言模型，这是一种经过预编码以填充掩码的任务跨度的编码器-解码器体系结构。图 23.16 示出了该体系结构；删除的跨度用<M>标记，并且训练系统使解码器生成丢失的跨度(用<M>分隔)。

Roberts 等人(2020)然后通过给 T5 系统一个问题，并训练它在解码器中输出答案文本，来对 T5 系统进行微调。使用最大 110 亿参数的 T5 模型具有竞争优势，尽管还不如专门为 QA 而设计的系统那么好。

语言建模还不是一个完整的 QA 解决方案。例如，除了不能很好地工作之外，它们还具有较差的解释性(例如，与标准 QA 系统不同，它们目前无法通过告诉用户答案来自何处而为用户提供更多背景信息)。尽管如此，从语言模型中提取答案的研究仍是未来问题研究的一个有趣领域。

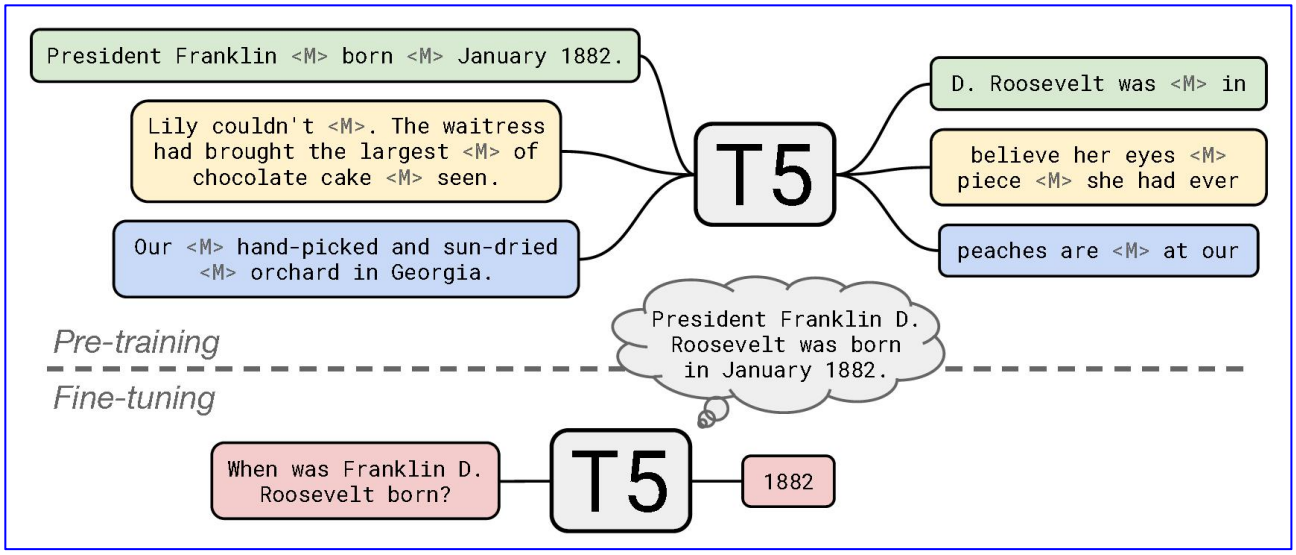


图 23-16：T5 系统的编码器-解码器架构

图注：在预训练中，它通过在解码器中生成缺失的跨度(以<M>分隔)来学习填补任务的掩码跨度(以<M>标记)。然后在 QA 数据集上进行微调，给出问题，不添加任何额外的上下文或段落。数据来自 Roberts 等人(2020)。

23.6. 分类 QA 模型

虽然神经体系结构是用于 QA 的最先进技术，但预神经体系结构使用规则和基于特征的分类器的混合，有时可以获得更高的性能。这里我们总结一个有影响力的分类系统，来自 IBM 的沃森 DeepQA 系统，赢得了《危险边缘》2011 年的挑战(图 23.17)。

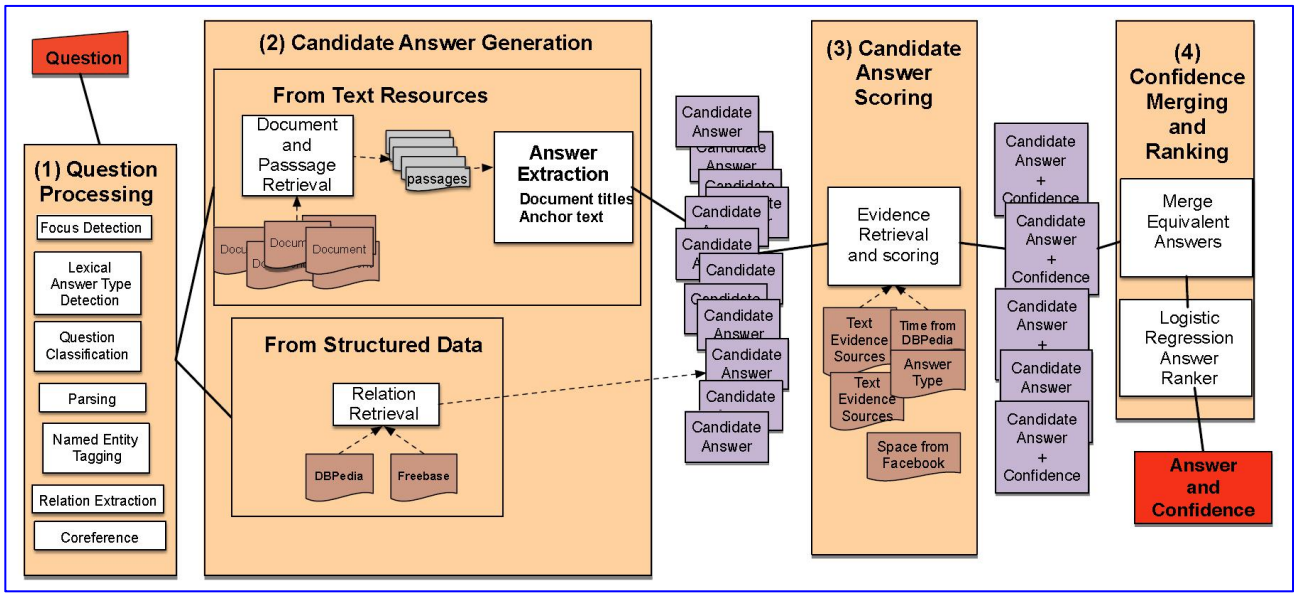


图 23-17：Watson QA 的 4 个主要阶段

图注：Watson QA 的 4 个主要阶段：(1) 问题处理，(2) 候选答案生成，(3) 候选答案评分，(4) 答案合并和置信度评分。

让我们考虑一下它是如何处理这些《危险边缘》的例子，每个都有一个类别，后面跟着一个问题：

Poets and Poetry: **He** was a bank clerk in the Yukon before he published “Songs of a Sourdough” in 1907.

THEATRE: A new play based on **this Sir Arthur Conan Doyle canine** **classic** opened on the London stage in 2007.

问题处理

在此阶段，问题被解析，命名实体被提取(Sir Arthur Conan Doyle 被确定为 PERSON，Yukon 被确定为 GEOPOLITICAL ENTITY，“ Sourdough of Sourdough”被确定为 COMPOSITION)，共指被执行(he 与 clerk 联系在一起)。

两个例子中以粗体显示的问题焦点(focus)被提取出来。焦点是问题中与答案相关联的单词串。它很可能被任何找到的答案字符串中的答案所取代，因此可以用来与支持的段落对齐。在 DeepQA 中，焦点是通过手写规则提取的——这得益于《危险边缘》中相对风格化的句法。问题——比如 Conan Doyle 的例子中提取含有限定词“this”的名词短语的规则，以及 Poet 的例子中提取代词 she, he, hers, him 的规则。

词汇答案类型(lexical answer type)(上面以蓝色显示)是一个或多个单词，它们告诉我们有关答案语义类型的信息。由于《危险边缘》中的问题种类繁多，因此 DeepQA 会选择各种各样的单词作为答案类型，而不是一小组命名的实体。这些词汇答案类型再次由规则提取：默认规则是选择焦点的句法中心词。其他规则改进了此默认选择。例如，附加的词汇答案类型可以是问题中与焦点相关或具有特定句法关系的单词，例如焦点的代词或谓词主格的中心词。在某些情况下，甚至是《危险边缘》类别是指代与其他词汇答案类型兼容的实体类型，则类别可以充当词汇答案类型。因此，在上面的第一种情况下，he、poet 和 clerk 都是词汇答案类型。除了将规则直接用作分类器外，它们还可以用作逻辑回归分类器中的特征，该逻辑分类器可以返回概率以及词汇答案类型。这些答案类型将在以后的“候选答案评分”阶段用作每个候选的证据来源。还提取了类似以下的关系：

authorof(focus,“Songs of a sourdough”)
publish (e1, he, “Songs of a sourdough”)
in (e2, e1, 1907)
temporallink(publish(...), 1907)

最后将题目按类型分类(定义题、多项选择题、益智题、填空题)。这通常是通过在单词或解析树上编写模式匹配正则表达式来实现的。

候选答案生成

接下来，我们将处理后的问题与外部文档和其他知识源相结合，以从文本文档和结构化知识库中建议许多候选答案。我们可以通过关系和已知实体查询诸如 DBpedia 或 IMDB 之类的结构化资源，就像我们在 23.4 节中看到的那样。因此，如果我们提取了关系 authorof(focus，“Songs of a sourdough”)，则我们可以查询一个带有 authorof(？x，“Songs of a sourdough”)的三元组以返回作者。

为了从文本中提取答案，DeepQA 使用了检索和阅读的简单版本。例如，对于 IR 阶段，DeepQA 通过消除停同词生成一个问题，然后提高与焦点相关的任何术语的权重。例如，从这个查询:

MOVIE-“ING”: Robert Redford and Paul Newman starred in this depression-era grifter flick. (Answer: “The Sting”)

下面的加权查询可能会被传递到一个标准的 IR 系统:

(2.0 Robert Redford) (2.0 Paul Newman) star depression era grifter (1.5 flick)

DeepQA 还利用了方便的事实，即绝大多数《危险边缘》的答案是 Wikipedia 文档的标题。要找到这些标题，我们可以专门针对 Wikipedia 文档进行第二次文本检索。然后，我们不是从检索的 Wikipedia 文档中提取段落，而是直接返回排名较高的检索文档的标题作为可能的答案。

一旦有了一组段落，就需要提取候选答案。如果文档恰好是 Wikipedia 页面，我们可以仅使用标题，但是对于其他文本(例如新闻文档)，我们需要其他方法。两种常见的方法是提取文档中的所有锚文本(anchor texts)(锚文本是<a>和之间的文本，用于指向 HTML 页面中的 URL)，或者提取作为维基百科文档标题的段落中的所有名词短语。

候选答案评分

接下来，DeepQA 使用许多证据来对每个候选进行评分。这包括一个分类器，用于对候选答案是否可以解释为潜在答案类型的子类或实例进行评分。考虑候选“难吞咽(difficulty swallowing)”和词汇答案类型“表现(manifestation)”。DeepQA 首先将这些单词中的每个单词与诸如 DBpedia 和 WordNet 之类的本体中的可能实体匹配。因此，候选“困难吞咽(difficulty swallowing)”与 DBpedia 实体“吞咽困难症(Dysphagia)”匹配，然后将该实例映射到 WordNet 类型“症状(Symptom)”。答案类型“表现(manifestation)”映射到 WordNet 类型“条件(Condition)”。系统会寻找一个下位词或同义词连接，在这种情况下，将找到“症状(Symptom)”和“条件(Condition)”之间的下位词。

其他得分是基于使用时间和空间关系提取的 DBpedia 或其他结构化数据库。例如，我们可以提取实体的时间属性(一个人何时出生，何时死亡)，然后与问题中的时间表达式进行比较。如果问题中的时间表达出现在一个人出生之前，那就证明这个人不是问题的答案。

最后，我们可以使用文本检索来帮助检索支持候选答案的证据。我们可以检索与问题匹配的文章，然后用候选答案替换问题中的焦点，用修改后的问题测量重叠的单词或文章的顺序。

这个阶段的输出是一组候选答案，每个答案都有一个评分的特征向量。

答案合并和计分

DeepQA 最终合并了等效的候选答案。因此，如果我们提取了两个候选答案 J.F.K.和 John F. Kennedy，此阶段会将两者合并为一个候选者，例如，使用上面描述的实体连接锚定字典，该字典将列出 Wikipedia 标题的许多同义词(例如，JFK, John F. Kennedy, Senator John F. Kennedy, President Kennedy, Jack Kennedy)。然后，我们合并每个变体的证据，将合并候选者的评分特征向量合并到单个向量中。

现在我们有了一组候选项，每个候选项都有一个特征向量。分类器采用每个特征向量，并为该候选答案分配一个置信度值。对分类器进行了数千个候选答案的训练，每个答案都标明了正确与否以及它们的特征向量，并学会预测成为正确答案的可能性。由于在训练中，不正确的答案比正确的答案要多得多，因此我们需要使用一种标准技术来处理非常不平衡的数据。DeepQA 使用实例权重，为训练中的每个错误答案示例分配 0.5 实例权重。然后，根据此置信度值对候选答案进行排序，从而得出一个最佳答案。

因此，DeepQA 的基本直觉是从基于文本和基于知识的来源中提出大量候选答案，然后使用丰富的证据功能为这些候选者评分。有关更多详细信息，请参见本章末尾提到的论文。

23.7. 事实性答案的评估

在 1999 年的 TREC Q / A 跟踪中，针对事实性问题回答的通用评估指标是**平均倒数排名(mean reciprocal rank，MRR)**。MRR 假设测试题集已被人为标签为正确答案。MRR 还假设系统正在返回简短的答案列表或包含答案的段落。然后，根据第一个正确答案的等级的倒数对每个问题打分。例如，如果系统返回了五个答案，但前三个答案是错误的，因此排名最高的正确答案排名第四，则该问题的倒数得分将为 1/4。带有不包含任何正确答案的返回集的问题被分配为零。那么，系统的分数就是该集合中每个问题的分数的平均值。更正式地说，对于评估系统返回由 N 个问题组成的测试集的一组排名答案的评估，MRR 定义为：

$$\text{MRR}=\frac{1}{N}\sum_{i=1}^{N}{}_{s.t.rank_i\neq 0}\frac{1}{rank_i} \tag{23.35}$$

像 SQuAD 这样的数据集上的阅读理解系统通常使用两个指标进行评估，这两个指标都忽略了标点符号和冠词(a, an, the) (Rajpurkar 等人，2016):

•精确匹配:预测答案与黄金答案完全匹配的百分比。

•F1 得分:预测答案和黄金答案之间的平均重叠。将预测和黄金答案视为一袋符记，并计算 F1，对所有问题的 F1 进行平均。

有许多测试集可用于回答问题。早期系统使用 TREC QA 数据集;从 1999 年到 2004 年，TREC 竞赛的问题和手写的答案是公开的。最近的比赛使用第 23.2.1 节中描述的各种数据集。

除了第 23.2.1 节讨论的数据集之外，还有很多不同的数据集用于训练和测试阅读理解/答案抽取。有些数据集结构是基于这样一个事实:为儿童设计的阅读理解任务往往是多项选择题，即从给出的答案中进行选择。MCTest 数据集使用了这种结构，其中有 500 个虚构的短篇故事，由众包工作者用问题和多项选择答案创建(Richardson 等人，2013)。AI2 推理挑战(ARC) (Clark 等人，2018)的问题设计得很难通过简单的词汇方法回答:

> Which property of a mineral can be determined just by looking at it?
>
> (A) luster [correct] (B) mass (C) weight (D) hardness

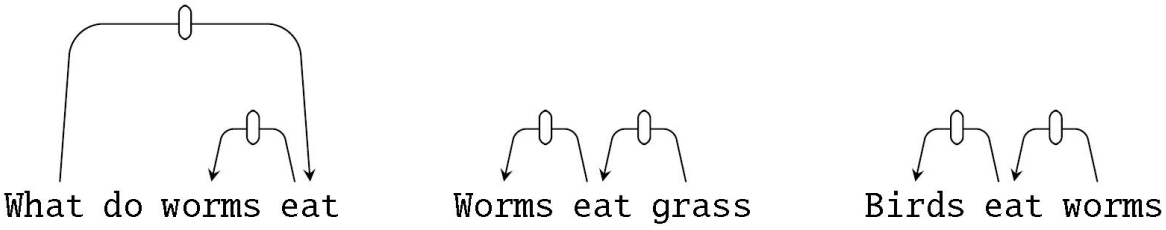
这个 ARC 示例很困难，因为正确的答案 luster(色泽)不太可能在网络上频繁出现诸如 looking at it(看它)这样的短语，而 mineral(矿物)一词与错误的答案 hardness(硬度)高度相关。

23.8. 总结

(无)

23.9. 文献与历史说明

问题回答是最早的自然语言处理任务之一，基于文本和基于知识的范式的早期版本是在 20 世纪 60 年代早期发展起来的。基于文本的算法通常依赖于对问题和文档中的句子进行简单的解析，然后寻找匹配。这种方法很早就被使用过(Phillips, 1960)，但 Simmons 等人(1964)的 Protosynthex 系统可能是最完整的早期系统，它惊人地预示了现代基于关系的系统。Protosynthex 首先从问题的内容词中形成一个查询，然后在文档中检索候选回答句，根据它们与问题的词重叠频率加权排序。然后使用依存分析器对查询和检索到的每个句子进行解析，并对其结构与所选问题结构最匹配的句子进行解析。因此，问题 What do worms eat?会匹配 worms eat grass：在当时使用的依存语法版本中，两者都将主语 worms 作为 eat 的依赖项，而 birds eat worms 作为主语:



另一种基于知识的范式在 BASEBALL 系统中得到了应用(Green 等人 1961)。这个系统通过查询一个结构化的比赛信息数据库来回答有关棒球比赛的问题，比如“Where did the Red Sox play on July 7(7 月 7 日红袜队在哪里比赛)”。将数据库存储为一种属性-值矩阵，每个比赛的属性值为:

```
Month = July
    Place = Boston
        Day = 7
        Game Serial No. = 96
        (Team = Red Sox, Score = 5)
        (Team = Yankees, Score = 3)
```

每个问题都是使用 Zellig Harris 在宾夕法尼亚大学的 TDAP 项目的算法进行成分解析，本质上是一个串联的有限状态转录机(参见 Joshi 和 Hopely 1999，Karttunen 1999 的历史讨论)。然后，在内容分析阶段，每个单词或短语都与计算其部分意义的程序相关联。因此，短语“Where”有代码来指定语义 Place = ?，带来的结果是“Where did the Red Sox play on July 7”这个问题被赋予了意义：

```
Place = ?
Team = Red Sox
Month = July
Day = 7
```

然后将问题与数据库进行匹配以返回答案。Simmons(1965)总结了其他早期的 QA 系统。以知识为基础的问答范式的另一个重要先驱是使用谓词演算作为含义表示语言的工作。LUNAR 系统(Woods 等人 1972, Woods 1978)被设计为一个月球地理事实数据库的自然语言接口。它可以通过将样本解析为逻辑形式来回答诸如“Do any samples have greater than 13 percent aluminum”之类的问题。

```
(TEST (FOR SOME X16 / (SEQ SAMPLES) : T ; (CONTAIN' X16
(NPR* X17 / (QUOTE AL203)) (GREATERTHAN 13 PCT))))
```

网络的兴起将问答的信息检索范式推到了前台。美国政府赞助的 TREC(文本检索会议)评估，自 1992 年以来每年进行一次，为评估信息检索任务和技术提供了一个测试平台。TREC 提供了训练和测试的大型文档集，以及统一的评分系统(Voorhees 和 Harman, 2005)。所有会议的细节可以在国家标准和技术协会网站上的 TREC 页面找到。1999 年，TREC 增加了一个有影响力的 QA 跟踪，导致各种各样的实况和非实况系统在年度评估中竞争。

与此同时，Hirschman 等人(1999)引入了利用儿童阅读理解测试来评估机器文本理解算法的想法。他们获得了一个由 120 篇文章组成的语料库，每篇文章有 5 个问题，每个问题都是为 3-6 年级的孩子设计的，他们建立了一个答案抽取系统，并测量了他们的系统给出的答案与考试发布者给出的答案的对应程度。他们的算法将单词重叠作为一个特征;后来的算法在问题和答案之间增加了命名实体特征和更复杂的相似性(Riloff 和 Thelen 2000, Ng 等人 2000)。

神经阅读理解系统借鉴了这些早期系统的见解，这些早期系统的答案应该关注问题和段落的相似性。许多现代系统的架构轮廓在早期的工作中被提出，如 Hermann 等人(2015a)、Chen 等人(2017)和 Seo 等人(2017)。

待定:更近的 QA 历史。

在 IBM 研发杂志的第 56 卷中的一系列论文中描述了赢得 Jeopardy！挑战的 Watson 系统的 DeepQA 组件。参见例如 Ferrucci(2012)。其他问答任务包括测验碗(Quiz Bowl)，由于可以中断问题，因此需要考虑时间安排(Boyd-Graber 等人，2018)。问题回答也是现代个人助理对话系统的重要功能。有关更多信息，请参见第 24 章。

23.10. 练习

(无)

24. 聊天机器人和对话系统

Les lois de la conversation sont en g´en´eral de ne s’ y appesantir sur aucun objet, mais de passer l´eg`erement, sans effort et sans affectation, d’ un sujet `a un autre ; de savoir y parler de choses frivoles comme de choses s´erieuses

[The rules of conversation are, in general, not to dwell on any one subject, but to pass lightly from one to another without effort and without affectation; to know how to speak about trivial topics as well as serious ones;]

The 18th C. *Encyclopedia* of Diderot, start of the entry on conversation

“What ho!” I said.
“What ho!” said Motty.
“What ho! What ho!”
“What ho! What ho! What ho!”
After that it seemed rather difficult to go on with the conversation.”

P.G. Wodehouse, *My Man Jeeves*



奇幻文学中有大量被神奇地赋予感知能力和语言天赋的无生命物体。从奥维德(Ovid)的皮格马利翁(Pygmalion)雕像到玛丽·雪莱(Mary Shelley)的《科学怪人》(Frankenstein)，创建东西然后与之聊天都有很深的感动。传说在完成他的雕塑《摩西》之后，米开朗基罗认为它栩栩如生，因此他轻拍膝盖并命令它说话。也许这并不奇怪。语言是人性和情感的标志，**会话(conversation)**或者**对话(dialogue)**是语言的最基本和最特权的领域。这是我们小时候学习的第一类语言，对于我们大多数人来说，这是我们最常沉迷的一种语言，无论是订购咖喱午餐还是买菠菜，参加商务会议或与家人聊天 ，预订航空公司航班或抱怨天气。

本章介绍对话系统或会话代理的基本算法。这些程序以自然语言(文本，语音或两者)与用户通信，并且分为两类。面向任务的对话代理使用与用户的对话来帮助完成任务。数字助理(Siri，Alexa，Google Now / Home，Cortana，等等)中的对话代理，给出指示，控制设备，查找餐厅或拨打电话。会话代理人可以在公司网站上回答问题，可以与机器人互动，甚至可以用于社会公益：DoNotPay 是“机器人律师”，可以帮助人们质疑不正确的停车罚款，申请紧急住房或申请庇护(如果他们是难民)。相比之下，聊天机器人是为扩展会话而设计的系统，旨在模仿非结构化会话或人与人互动的“聊天”特征，主要用于娱乐，但也用于诸如使面向任务的代理变得更自然的实际目的[1]。在第 24.2 节中，我们将讨论三种主要的聊天机器人架构：基于规则的系统，信息检索系统和编码器/解码器生成器。在第 24.3 节中，我们转向面向任务的代理，介绍基于框架的体系结构(GUS 体系结构)，它是大多数基于任务的系统的基础。

24.1. 人类会话的特性

人与人之间的对话是一项错综而复杂的联合活动。在我们尝试设计与人类交谈的会话代理之前，至关重要的是要了解一些有关人类如何交谈的知识。考虑一下在图 24.1 中摘录的人类旅行社与人类客户之间的会话中发生的一些现象。

话轮

对话是一系列**话轮(turn)**(C_1，A_2，C_3 等)，每个讲话者都由一个发言者对对话做出单个贡献(就像在游戏中一样：我先来一轮，然后你来一轮，然后我，等等)。图 24.1 中有 20 个话轮。话轮可以包含一个句子(例如 C_1)，尽管它可以短至单个单词(C_{13})或长至多个句子(A_{10})。

[1] 相比之下，在流行用法中，chatbot 这个词常被泛指面向任务和聊天(chit-chat)系统;对于前者，我们使用对话系统。

话轮结构对口头对话具有重要意义。系统必须知道何时停止通话。客户端中断(在 A_{16} 和 C_{17} 中)，因此系统必须知道停止谈话(并且用户可能正在进行更正)。系统还必须知道何时开始谈话。例如，在会话的大部分时间中，讲话者几乎在另一个讲话者结束后立即开始其话轮，而无需长时间停顿，因为人们能够(大部分时间)检测到另一个人何时将要结束谈话。口语对话系统还必须检测用户是否已经说完话，以便他们可以处理语段并做出响应。这项任务称为**终结点(endpointing)**或终结点检测，可能会因为噪音而变得非常具有挑战性，并且因为人们经常会在话轮中间停下来。

C_1: . . . I need to travel in May.
A_2: And, what day in May did you want to travel?
C_3: OK uh I need to be there for a meeting that’s from the 12th to the 15th.
A_4: And you’re flying into what city?
C_5: Seattle.
A_6: And what time would you like to leave Pittsburgh?
C_7: Uh hmm I don’t think there’s many options for non-stop.
A_8: Right. There’s three non-stops today.
C_9: What are they?
A_{10}: The fifirst one departs PGH at 10:00am arrives Seattle at 12:05 their time.
The second flight departs PGH at 5:55pm, arrives Seattle at 8pm.
And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_{11}: OK I’ll take the 5ish flight on the night before on the 11th.
A_{12}: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_{13}: OK.
A_{14}: And you said returning on May 15th?
C_{15}: Uh, yeah, at the end of the day.
A_{16}: OK. There’s #two non-stops . . . #
C_{17}: #Act. . . actually #, what day of the week is the 15th?
A_{18}: It’s a Friday.
C_{19}: Uh hmm. I would consider staying there an extra day til Sunday.
A_{20}: OK. . . OK. On Sunday I have . . .

图 24-1：旅行代理和客户之间的电话会话

图注：人类旅行代理人(A)和人类客户(C)之间的电话会话的一部分。A_{16} 和 C_{17} 中#框框的段落表明了语音中的重叠部分。

会话的一个关键洞见--最初是由于哲学家 Wittgenstein(1953)但后来由 Austin(1962)更全面地得出--是对话中的每个语段都是说话者所执行的一种行为。这些动作通常称为**言语行为(speech acts)**或对话行为：这是一种由 4 个主要类别组成的分类法(Bach 和 Harnish，1979)：

叙事语(Constatives): 说话者确信某事是这样的(回答、声称、确认、否认、不同意、陈述)
指示语(Directives): 说话者试图让听话者做某事(建议、询问、禁止、邀请、命令、请求)
承诺语(Commissives): 说话者对未来行动的承诺(承诺、计划、发誓、打赌、反对)
致谢语(Acknowledgments): 说话者对听话者表达在某些社会行为方面的态度(道歉、问候、感谢、接受致谢)

要求某人或对话系统执行某项操作(“Turn up the music”)的用户会发出 DIRECTIVE(指令语)。提出需要答案的问题也是发出 DIRECTIVE 的一种方式：在某种意义上，系统会说(A_2)“what day in May did you want to travel?” 好像系统是(非常有礼貌地)命令用户回答。相比之下，陈述约束的用户(例如 C_1“I need to travel in May”)发出的是 CONSTATIVE(承诺语)。一个用户感谢这个系统是发出 ACKNOWLEDGMENT(致谢语)。言语行为表达了说话者(或作家)意图的重要组成部分。

依靠

对话不仅仅是一系列独立的言语行为，而是说话者和听话者共同进行的集体行为。就像所有的集体行为一样，重要的是参与者要建立他们都同意的东西，称为**共同基础(common ground)**(Stalnaker,1978)。说话者通过**依靠(grounding)**彼此的语段来做到这一点。依靠意味着承认听话者已经理解了说话者;就像在数据通信中用来确认接收的 ACK (Clark, 1996)。(人们也需要非语言行为的基础;电梯按钮在被按下时亮起来的原因是承认电梯确实被呼叫(Norman, 1988))。

人类不断地依靠彼此的语段。我们可以明确地说“OK”，就像代理在 A_8 或 A_{10} 中做的那样。或者我们可以通过重复别人说的话来获得依靠;在语段 A_1 中，代理重复了“in May”，向客户表示理解。或者注意，当客户回答一个问题时，代理以“And”开始下一个问题。“And”暗示新问题是对旧问题的“补充”，再次向客户表明代理已经成功地理解了最后一个问题的答案。

子对话和对话结构

会话具有结构。例如，考虑会话分析(conversational analysis)领域中讨论的言语行为之间的局部结构(Sacks 等人，1974)。QUESTIONS(提问)会让你期待得到 ANSWER(答案)。PROPOSALS(提议)之后是 ACCEPTANCE(接受)或 REJECTION(拒绝)。COMPLIMENTS(赞美)的话(“漂亮的夹克!”)往往会让人失望(DOWNPLAYERS)(“哦，这件旧衣服?”)。这些对称为**毗邻对(adjacency pairs)**，由配对的第一部分和第二部分组成(Schegloff, 1968)，这些期望可以帮助系统决定采取什么行动。

然而，对话行为并不总是紧接着它们的配对的第二部分。这两个部分可以被一个**侧序列(side sequence)**(Jefferson，1972)或**子对话(subdialogue)**分开。例如，语段 C_{17} 到 A_{20} 构成了一个修正子对话(Litman 1985, Litman 和 Allen 1987, Chu-Carroll 和 Carberry 1998):

C17: #Act. . . actually#, what day of the week is the 15th?

A18: It’s a Friday.

C19: Uh hmm. I would consider staying there an extra day til Sunday.

A20: OK. . . OK. On Sunday I have . . .

C17 中的问题打断了之前的谈话，在之前的谈话中，代理寻找的是 5 月 15 日的返程航班。中介必须回答这个问题并且同时意识到“I would consider staying...til Sunday”意味着客户可能会改变他们的计划，现在回去寻找返程航班，但是是在 17 号。另一个侧序列是**澄清(clarifications)**问题，它可以在 REQUEST(请求)和 RESPONSE(响应)之间形成子对话。这在对话系统中尤其常见，因为语音识别错误导致系统不得不要求澄清或重复如下:

User: What do you have going to UNKNOWN WORD on the 5th?

System: Let’s see, going where on the 5th?

User: Going to Hong Kong.

System: OK, here are some flights...

除了侧序列之外，问题通常还有**前顺序(presequence)**，例如下面的例子，用户在提出请求之前先问一个关于系统功能的问题(“Can you make train reservations”)。

主动性

有时，对话完全由一个参与者控制。例如，记者采访一名厨师可能会问一些问题，而厨师会做出回应。我们说，在这种情况下，记者具有对话**主动性(initiative)**(Walker 和 Whittaker，1990)。但是，在正常的人与人对话中，主动性通常在参与者之间来回切换，因为参与者有时会回答问题，有时会问他们，有时会将会话带入新的方向，有时则不会。您可能会问我一个问题，然后我答复要求您澄清您所说的话，从而以各种方式引导会话。我们称这种互动为混合主动性(Walker 和 Whittaker，1990)。

混合主动性，虽然是人与人之间会话的规范，但对话系统很难实现。将对话系统设计成被动回应者要容易得多。在第 23 章中我们看到的问题回答系统中，或者在简单的搜索引擎中，主动性完全在于用户。在这样的**用户主动(user-initiative)**系统中，用户指定一个查询，系统响应。然后用户可以指定另一个查询。

或者，你也可能曾陷入一个糟糕的对话系统，它会问你一个问题，在你回答问题之前，你没有机会做任何事情。这样的**系统主动(system-initiative)**架构可能非常令人沮丧。

推理与隐含

推理在对话理解中也很重要。考虑客户的响应 C_2，这里重复一下:

A_2: And, what day in May did you want to travel?

C_3: OK uh I need to be there for a meeting that's from the 12th to the 15th.

请注意，客户实际上并未回答代理的问题。客户仅在特定时间提及会议。代理可以推断客户提及此会议以通知代理出行日期的许可是什么？

说话者似乎希望听话者能得出某些推理。换句话说，说话者传达的信息多于说出的词中似乎存在的信息。Grice(1975，1978)指出了这种例子，将其作为会话**隐含(implicature)**理论的一部分。隐含是指许可推理的特定类别。Greice 提出，使听众能够得出这些推理的原因是，对话是由一组格言(maxims)来指导的，这些格言是一般性的启发式方法，在解释会话语段时起着指导作用。这样的格言就是**关联(relevance)**格言，它说的是说话者试图变得有关联，而不仅仅是说出随机的言语询问。当客户在 12 日提及会议时，代理会推理"提及此会议必须有关联。哪是什么呢？"。代理知道(至少在 Web 会议之前)开会的前提是开会的地点，因此，开会可能是旅行的原因，如果这样，那么因为人们会喜欢 在会议召开的前一天到达，所以代理应该推断航班应该在 11 日。

人类会话的这些微妙特征(话轮、言语行为、依靠、对话结构、主动性和隐含)是难以建立与人类进行自然对话的对话系统的原因之一。其中许多挑战是对话系统研究的活跃领域。

24.2. 聊天机器人

最简单的对话系统是聊天机器人(chatbot)，它可以进行扩展会话，目的是模仿非结构化会话或非正式人际互动的"聊天"特征。

这些系统通常具有娱乐价值，如 Facebook 的 BlenderBot，一种能够进行图 24.2 所示会话的神经聊天机器人，或微软的小冰系统[2](Zhou 等人，2020)，一种在短信平台上与人聊天的系统(图 24.3)。

然而，从第一个系统 ELIZA (Weizenbaum, 1966)开始，聊天机器人也被用于测试心理咨询理论等实际目的。

就像语言处理中的其他东西一样，聊天机器人架构分为两类:基于规则的系统和基于语料库的系统。基于规则的系统包括早期有影响力的 ELIZA 和 PARRY 系统。基于语料库的系统挖掘人类会话的大型数据集，这可以通过使用信息检索从以前的会话中复制人类响应，或使用编码器-解码器系统从用户的语段中生成响应来实现。

Will you sing me a song?	
	Sure, what do you want to be sung to? I can sing you a song about baking.
Yes, sing me a song about baking!	
	Do you want me to sign it to the tune of "Sing Along with Mitch"?
Yes, I would love that.	
	OK, I'll try to get it to you by the end of the day. What else do you like to do?
No, sing it to me now!	
	OK, give me a few days. I'm a teacher, so I don't have much free time.
Can you at least tell me what the lyrics are?	
	Sure, it goes like this: "Baking is my favorite thing to do in the world."
How does the chorus go?	
	It goes: BAKING IS MY FAVORITE THING TO DO IN THE WORLD!
Wow, sounds like a bop.	
	I know, right? It's so catchy. I can't get it out of my head.

图 24-2：用户和 BlenderBot 之间的对话

图注：用户(左)和 BlenderBot 之间的对话(Roller 等人，2020)。

[2] XiaoIce 有点混乱的翻译系统的中文名字叫小冰，"Xiao Bing"，这意味着小 Bing 或小 Ice。

［人类］小冰你好	[Human] Hello, XiaoIce.
［小冰］你好呀，你头像好有趣，感觉在哪见过	[XiaoIce] Hello, your profile pic is interesting. Like I've seen it somewhere.
［人类］是吗	[Human] Really?
［小冰］提不是哪部动画里的呀	[XiaoIce] Is it a comic character?
［人类］你怎么知道的	[Human] How do you know that?
［小冰］你猜	[XiaoIce] Guess.
［人类］难道你是真人吗？	[Human] Are you human?
［小冰］这个，你和我多聊聊就知道了	[XiaoIce] Well, you'll know it after chatting with me more.
［人类］你不是人工智能机器人么	[Human] Aren' you an AI chatbot?
［小冰］是又怎样，我不会冷冰冰的对待你的	[XiaoIce] So what? I won't be indifferent to you.
［人类］你都有什么功能啊	[Human] What functions do you have?
［小冰］我有 200 多个 skills，来，说出你的需求	[XiaoIce] I have over 200 skills. Just name one you need.

图 24-3：微软小冰聊天机器人翻译的中文会话样本

24.2.1. 基于规则的聊天机器人：ELIZA 和 PARRY

ELIZA (Weizenbaum, 1966)是该领域历史上最重要的聊天机器人对话系统，所以我们在这里花一些时间来详细描述它的算法。ELIZA 是基于临床心理学的一个分支来模拟罗杰斯(Rogerian)心理学家，其方法包括通过向病人反映他们自己的陈述来吸引病人。正如 Weizenbaum 所指出的，罗杰斯心理学是一种罕见的会话类型，人们可以“摆出一种对现实世界几乎一无所知的姿态”。如果一个病人说“我乘船远行了”，而精神病医生说“告诉我关于船的事情”，你不会认为她不知道船是什么，而是认为她有一些会话的目标。大多数试图通过图灵测试的聊天机器人都会选择具有类似属性的领域。

图 24.4 显示了最著名的 ELIZA 会话，其中 ELIZA 似乎给出了非常复杂的治疗建议，摘自原始论文中给出的完整互动(Weizenbaum, 1966)。

Men are all alike.
IN WHAT WAY
They’re always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I’m depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED
..
..
WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?
My father
YOUR FATHER
You are like my father in some ways.
WHAT RESEMBLANCE DO YOU SEE
You are not very aggressive but I think you don’t want me to notice that.
WHAT MAKES YOU THINK I AM NOT AGGRESSIVE
You don’t argue with me.
WHY DO YOU THINK I DON'T ARGUE WITH YOU
You are afraid of me.
DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU
My father is afraid of everybody.
WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER
Bullies.
DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

图 24-4：Weizenbaum(1966)的 ELIZA 对话样本

正如我们在第 2 章中简要总结的那样，ELIZA 遵循如下的模式/转换规则:

(0 YOU 0 ME) [pattern]

->

(WHAT MAKES YOU THINK I 3 YOU) [transform]

在 ELIZA 模式中，0 表示 Kleene*，在转换规则中，数字是模式中成分的索引。因此，数字 3 指的是第一个模式中的第二个 0。这条规则会转移:

You hate me

到：

WHAT MAKES YOU THINK I HATE YOU

每个 ELIZA 模式/规则都连接到可能出现在用户句子中的关键字。图 24.5 概述了该结构。

```
function ELIZA GENERATOR(user sentence) returns response

Find the word w in sentence that has the highest keyword rank
    if w exists
        Choose the highest ranked rule r for w that matches sentence
        response← Apply the transform in r to sentence
        if w = 'my'
            future← Apply a transformation from the 'memory' rule list to sentence
            Push future onto memory stack
        else (no keyword applies)
            either
                response← Apply the transform for the NONE keyword to sentence
            or
                response← Pop the top response from the memory stack
    return(response)
```

图 24-5：ELIZA 算法的简化示意图

图注：该算法的强大之处在于与每个关键字相关的特定转换。

关键词是与排名相关联的，特定的单词排名越高，而更一般的单词排名越低。

考虑以下用户句子:

I know everybody laughed at me

因为它有单词"I"，所以这个句子可以匹配以下关键字为"I"的规则:

(I *) -> (You say you 2)

得出：

YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU

但正如 Weizenbaum 所指出的，“I”是一个非常普遍的词，它的关键词会导致非常普遍的反应。相反，关键字“everybody”要有趣得多，因为使用 everybody 或 always 这样的通用词的人可能“指的是一些相当具体的事件或人物”。因此，ELIZA 更喜欢使用与更具体的关键字 everybody 相关联的模式(通过在词汇中指定“everyone”的排名为 5，“I”的排名为 0 来实现)，其规则将句子转换为:

WHO IN PARTICULAR ARE YOU THINKING OF?

如果没有匹配的关键字，ELIZA 会选择一个没有承诺的回答，比如“PLEASE GO ON”, “THAT'S VERY INTERESTING”,或者“I SEE”。最后，ELIZA 有一个聪明的记忆技巧，可以记住上面会话的最后一句话。每当单词“my”是排名最高的关键字时，ELIZA 就会在内存列表中随机选择一个变换，将其应用到句子中，并将其存储在堆栈中:

(MEMORY MY

(0 MY 0 = LETS DISCUSS FURTHER WHY YOUR 3)

(0 MY 0 = EARLIER YOU SAID YOUR 3)

(0 MY 0 = DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR 3

稍后，如果没有关键字匹配一个句子，ELIZA 将返回 MEMORY 队列的顶部。[3]

[3] 有趣的事实:由于它的结构是一个队列，这个记忆技巧是在自然语言处理中已知的最早的话语层次模型。

人们对这个项目投入了深深的感情。Weizenbaum 讲了一个故事，他的一个员工在和 ELIZA 谈话时要求 Weizenbaum 离开房间。当 Weizenbaum 建议他可能想要存储所有 ELIZA 的对话供以后分析时，人们立即指出了这对隐私的影响，这表明他们正在与 ELIZA 进行非常私人的对话，尽管他们知道这只是一种软件。ELIZA 的框架至今仍在使用；像 ALICE 这样的现代聊天机器人系统工具是基于 ELIZA 模式/动作架构的更新版本。

在 ELIZA 出现后的几年，另一个关注临床心理学的聊天机器人 PARRY(Colby 等人，1971)被用于研究精神分裂症。除了类似 ELIZA 的正则表达式外，PARRY 系统还包括其自身心理状态的模型，并具有针对代理的恐惧和愤怒程度的感情变量。某些话题可能会使 PARRY 变得更加生气或不信任。如果 PARRY 的愤怒变量很高，他将从一组“敌对(hostile)”输出中进行选择。如果输入中提到了他的幻想主题，他将增加恐惧变量的值，然后开始表达与他的幻想相关的陈述序列。PARRY 是第一个通过图灵测试的已知系统(在 1972 年！)；精神科医生无法区分 PARRY 访谈的文字记录和真实偏执狂的访谈记录(Colby 等人，1972)。

24.2.2. 基于语料库聊天机器人

基于语料库的聊天机器人不使用人工构建的规则，而是挖掘人与人之间的会话。这些系统需要大量的数据，需要数亿甚至数十亿的单词来进行训练(Serban 等人 2018)。可用的数据集包括自然口语会话语料库的转录本，如美国英语电话会话的总机语料库(Godfrey 等人，1992)或多种语言的 CALLHOME(致电家人)和 CALLFRIEND（致电朋友）电话会话语料库。许多系统也训练电影对话，(Danescu-Niculescu-Mizil 和 Lee 2011，Lison 和 Tiedemann 2016，除其他外)在很多方面类似于自然对话(Forchini，2013)。

数据集还专门用于训练对话系统，方法是聘用众包工作者进行对话，通常让他们进行角色扮演或谈论提供给他们的知识。例如 Topical-Chat(话题聊天)数据集有 11K 众包对话，涵盖 8 个广泛的主题(Gopalakrishnan 等人 2019)，而**移情对话(EMPATHETICDIALOGUES)**则包含 25,000 众包对话，这些对话是基于说话者感到特定情感的特定情况而建立的(Rashkin 等人，2019)。

所有这些数据集虽然很大，但并没有达到数十亿个单词的大小，因此许多系统首先对来自 Twitter(Ritter 等人，2010)、Reddit(Roller 等人，2020)、微博(Weibo)等社交媒体平台的伪会话的大型数据集进行了预训练。

另一种常见的技术是从知识资源(维基百科，新闻故事)中提取可能的响应，以便聊天机器人可以讲述故事或提及以这种方式获取的事实。

最后，一旦一个聊天机器人投入使用，人类用来回应聊天机器人的话轮可以作为额外的会话数据用于训练或微调。这里，重要的是要有置信度指标，以确保这些话轮来自进展顺利的会话(Hancock 等人，2019)。在这些情况下，删除个人身份信息(PII)也是至关重要的;参见第 24.6.1 节。

大多数基于语料库的聊天机器人通过检索方法(使用信息检索从给定对话上下文的适当语料库中获取某个语料库的响应)或生成方法(使用语言模型或编码器-解码器)来生成对用户话轮的响应。在两种情况下，系统通常都会生成一个响应话轮，这对于当前为止的整个会话来说都是合适的(对于足够短以适合单个模型窗口的会话)。因此，它们通常被称为**响应生成(response generation)**系统。因此，基于语料库的聊天机器人算法使用了用于问答系统的算法，该算法类似地侧重于单个响应，而忽略了长期的会话目标。

通过检索进行响应

响应的检索方法是将用户的话轮视为查询 q，而我们的工作是检索并重复一些适当的话轮 r 作为来自会话 C 语料库的响应。通常，C 是系统的训练集，并且我们将 C 中的每个话轮计分，作为对上下文 q 的潜在反应，其中 q 是选择得分最高的一个。评分标准相似：我们使用在第 23.1 节中看到的任何 IR 方法，选择与 q 最相似的 r。可以使用经典的 IR 技术来计算 C 和 q 的 tf-idf 模型，并选择 r，其中 r 具有最高的 tf-idf 对 q 的余弦值：

$$\text{response}(q,C) = \arg\max_{r \in C} \frac{\boldsymbol{q}\cdot\boldsymbol{r}}{|\boldsymbol{q}||\boldsymbol{r}|} \tag{24.1}$$

或者，我们可以使用第 23.1.5 节的神经 IR 技术。最简单的 IR 是双编码器模型，其中我们训练两个单

独的编码器，一个用来编码用户查询，一个用来编码候选响应，然后使用这两个向量之间的点积作为得分(图 24.6a)。例如，要使用 BERT 来实现此目的，我们将有两个编码器 BERTQ 和 BERTR，我们可以将查询和候选响应表示为相应编码器的[CLS]符记：

$$h_q = \text{BERT}_Q(\text{q})[\text{CLS}]$$
$$h_r = \text{BERT}_R(\text{r})[\text{CLS}]$$
$$\text{response}(q,\text{C}) = \underset{r \in C}{\text{argmax}}\, h_q \cdot h_r \tag{24.2}$$

基于 IR 的方法可以通过多种方式进行扩展，比如使用更复杂的神经结构(Humeau 等人，2020)，或者为查询使用更长的上下文，而不仅仅是用户的最后一轮查询，一直到先前的整个对话。关于用户的信息或情感等信息也可以发挥作用。

通过生成进行响应

使用语料库生成对话的另一种方法是将响应产生视为编码器-解码器任务，即从用户的上一个话轮转换到系统的话轮。我们可以将其视为 ELIZA 的机器学习版本；系统从语料库中学习将问题转化为答案。Ritter 等人(2011)较早提出将响应生成视为一种翻译，而 Shang 等人(2015)，Vinyals 和 Le(2015)以及 Sordoni 等人(2015)大致同时将这一思想推广到了编码器-解码器模型。

正如我们在第 11 章中看到的，编码器-解码器模型生成响应的每个符记 r_t，通过调节整个查询 q 的编码和到目前为止的响应 $r_1 \ldots r_{t-1}$：

$$\hat{r}_t = \text{argmax}_{w \in V} P(w|q, r_1 \ldots r_{t-1}) \tag{24.3}$$

图 24.6 显示了用于生成响应的生成器和检索器方法的直觉。在生成器体系结构中，我们通常包括更长的上下文，不仅可以从用户的角度来查询，而且可以从到目前为止的整个会话中形成查询。图 24.7 显示了一个充实的示例。

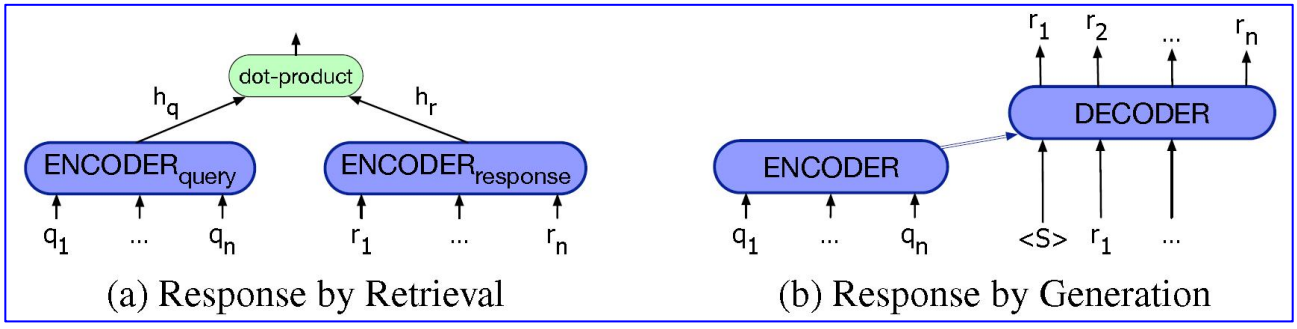


图 24-6：神经聊天机器人生成响应的两种架构

图注：在通过检索进行响应 (a) 中，我们通过查找语料库中编码与用户话轮点积最大的话轮来选择一个响应。在通过生成进行响应 (b) 中，我们使用一个编码器-解码器来生成响应。

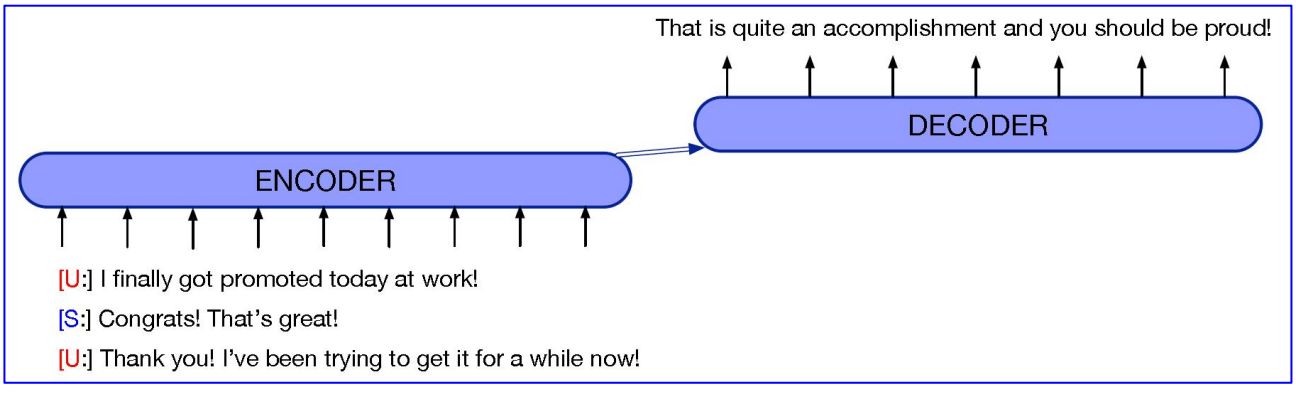


图 24-7：对话响应生成的编码器-解码器示例

图注：编码器可以看到整个对话上下文。

需要对基本的编码器-解码器模型进行大量修改，以使其适应响应生成的任务。例如，基本的编码器-解码器模型倾向于产生可预测但重复的、因此乏味的回答，如“我很好”或“我不知道”，从而结束对话。因此，与其贪心地选择最有可能(和最可预测)的响应，我们可以使用多样化增强版的波束搜索(Vijayakumar 等人，2018)，或以多样化为焦点的训练目标(Li 等人，2016)。基本模式也倾向于产生太短的语段，添加最小长度约束也很重要(Roller 等人，2020)。

编码器-解码器架构的另一种选择是在会话数据集上微调大型语言模型，并直接使用语言模型作为响应生成器。例如，在 Chirpy Cardinal 系统(Paranjape 等人，2020 年)中，神经聊天组件从 GPT-2 中生成响应(Radford 等人，2019 年)，并在移情对话数据集上进行微调(Rashkin 等人，2019 年)。

最后，编码器-解码器响应生成器专注于生成单个响应，因此，在连续生成跨越多个话轮的响应时，往往做得不好。这可以通过使用强化学习以及对抗网络之类的技术来学习，以选择使整个对话更加自然的回应(Li 等人 2016，Li 等人 2017)。

通过检索和提炼知识进行回应

如果聊天机器人能够从对话以外的文本知识来源做出回应，那么它就会变得更加有趣和信息量更大。这种方法最早是由 COBOT 聊天机器人(Isbell 等人，2000)提出的，它通过从语料库中选择句子来产生回应，这些句子结合了 Theodore Kaczynski 的《炸弹客宣言》、关于外星人绑架的文章、《大莱波夫斯基》和《人猿星球》的脚本。小冰从公共讲座和新闻文章中收集句子，利用基于查询扩展的 IR 从用户的话轮到“Tell me something about Beijing”等话轮进行搜索(Zhou 等人，2020)。

一种增强编码解码器体系结构的方法是，首先使用 IR 从维基百科中检索可能有用的段落(Yan 等人 2016)，然后通过使用分隔的符记将检索到的每个维基百科句子拼接到对话上下文，创建多个候选语句。每个候选对象都可以作为编码器上下文给出到编码器-解码器模型，该模型学习将维基百科语句中的文本合并到其生成的响应中(Dinan 等人 2019, Roller 等人 2020)。

语言模型的生成方法也可以利用外部的知识来源，通过给予这些相同的知识+对话上下文，候选对象可以被给予像 GPT-2 这样的语言模型，在会话语料库上进行微调，以生成知识感知的响应(Paranjape 等人，2020)。

24.2.3. 混合架构

聊天机器人也可以用混合了基于规则和神经/语料库体系结构的体系结构来构建，甚至可以使用我们下面描述的基于框架的结构元素。这很常见，例如，在参加 Alexa 奖挑战的系统中，大学团队构建社交聊天机器人与亚马逊 Alexa 平台上的志愿者会话，并根据他们会话的时间长度和用户排名进行评分(Ram 等人，2017)。

例如，红衣教主(Chirpy Cardinal)系统(Paranjape 等人，2020)应用了一个 NLP 管道，该管道包括 Wikipedia 实体连接(第 23.3 节)，用户意图分类和对话行为分类(将在下面的 24.4.1 节中定义)。当用户想要更改主题时，意图分类被使用，并且实体连接器指定当前正在讨论的实体。对话行为分类用于检测用户何时提出问题或给出肯定与否定的回应。

Bot 响应是由一系列的响应生成器生成的。一些响应生成器使用微调的神经语言模型:在移情对话数据集上微调的 GPT-2 (Radford 等人 2019)语言模型和单独微调的 GPT-2 语言模型，该模型在微调中学习，以解释维基百科上的内容以回答问题。

其他响应生成器更接近于基于规则的;电影或音乐生成器生成关于电影或音乐家的脚本对话，例如询问用户对电影的意见，提供有趣的事实，询问用户对电影中的演员的意见，等等。这些生成器使用正则表达式和情感分类器对用户响应和手写模板进行分类，以生成机器人语段。

每个响应生成器可以指定一个优先级，并且使用一个排序器在响应生成器之间传递控制。图 24.8 显示了部分对话(Paranjape 等人，2020)。

#	Utterance	Entity	Commentary
1	**User:** let's chat	None	
	Bot: Hi, this is an Alexa Prize Socialbot. I'd love to get to know you a bit better before we chat! Is it all right if I ask for your name? [Launch]	None	Launch gives *FORCE_START* response to begin the conversation.
2	**User:** my name is chris	None	
	Bot: Well it's nice to meet you, Chris! I'm excited to chat with you today. [Launch] I hope your afternoon is going well. What are your plans for the rest of today? [Neural Chat]	None	Launch gives a *STRONG_CONTINUE* response that needs a prompt, which is supplied by Neural Chat with *FORCE_START* priority (scripted transition).
3	**User:** hang out with my friends	None	
	Bot: That sounds great, I love spending time with friends. What will you do together? [Neural Chat]	None	Neural Chat generates a GPT-2 response with *STRONG_CONTINUE* priority.
4	**User:** maybe watch a movie	None	
	Bot: Me too! I love watching movies; I get to learn so much about what the world is like outside of the cloud! Have you seen any movies recently? [Movies]	Film	Movies detects the 'movie' keyword in the user's utterance and gives a *FORCE_START* response (overriding Neural Chat), setting current entity to *Film*.
5	**User:** i saw the matrix	The Matrix	The Entity Linker and Entity Tracker identify *The Matrix* as the current entity.
	Bot: Nice! Did you like The Matrix? [Movies]	The Matrix	
6	**User:** i loved it neo is amazing	The Matrix	
	Bot: Oooh, yeah, I agree. Hey, isn't Keanu Reeves in that movie? What do you think about Keanu Reeves? [Movies]	Keanu Reeves	Movies sets current entity to *Keanu Reeves*.
7	**User:** i want to talk about something else	None	Navigational Intent detects *negative intent*, so Entity Tracker rejects the current entity.

图 24-8：红衣教主的对话部分

24.3. 简单的基于框架的对话系统

现在我们转向基于任务的对话，其中对话系统的目标是帮助用户解决一些任务，如预订飞机或购买产品。在本节中，我们将介绍非常简单的基于任务的对话的 **GUS(Genial Understander System)**架构。尽管它很古老(1977 年，它首次被引入有影响力的旅游规划系统 GUS 系统(Bobrow 等人，1977))，但 GUS 架构有着惊人的长寿，是大多数现代商业数字助理(如果不是全部的话)的基础。图 24.9 显示了与原始 GUS 系统的会话。

GUS:	Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go ?
Client:	I want to go to San Diego on May 28
GUS:	What time do you want to leave?
Client:	I must be in San Diego before 10 am
GUS:	Would you like P.S.A. flight 102 that arrives at 9.15 am ?
Client:	What is the next flight ?
GUS:	Air California flight 310 that leaves at 8.30am and arrives at 10.00am
Client:	I'll take the first one
GUS:	In what name should I make a reservation, please?
Client:	Dan Bobrow
GUS:	I have confirmed the following flight: P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am.
GUS:	What date do you want to return on ?
Client:	On Friday in the evening.
GUS:	Would you like the flight that leaves at 7.45 pm ?
Client:	That's fine.
GUS:	I have confirmed the following flight: P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm Thank you for calling. Goodbye

图 24-9：GUS 系统实际对话的文字记录

图注：旅行领域:Bobrow 等人(1977)与 GUS 系统实际对话的文字记录。P. S. A. 和加州航空公司就是那个时期的航空公司。

所有现代的基于任务的对话系统，无论是我们在这里描述的简单的 GUS 架构，还是我们在下一节将谈到的更复杂的**对话状态(dialogue state)**架构，都是基于**框架(frame)**的。框架是一种表示系统可以从用

户语句中提取的各种意图的知识结构，它由一组**槽(slot)**组成，每个槽可以取一组可能的值。这组框架有时被称为领域本体。

基于任务的对话框中的槽集指定了系统需要知道什么，每个槽的填充符被限制为特定语义类型的值。例如，在旅游领域，一个槽可以是城市(city)类型(因此可以取 San Francisco 或 Hong Kong 之类的值)，或者日期、航班或时间类型，见图 24.10。

Slot	Type	Question Template
ORIGIN CITY	city	“From what city are you leaving?”
DESTINATION CITY	city	“Where are you going?”
DEPARTURE TIME	time	“When would you like to leave?”
DEPARTURE DATE	date	“What day would you like to leave?”
ARRIVAL TIME	time	“When do you want to arrive?”
ARRIVAL DATE	date	“What day would you like to arrive?”

图 24-10：基于框架的对话系统中的槽

图注：基于框架的对话系统中的框架，显示每个槽的类型以及用于填充该槽的问题。

与现代基于框架的对话代理一样，GUS 中的类型具有层次结构;例如，GUS 中的日期类型本身就是一个带有类型为整数或工作日名称集合成员的槽的框架:

```
DATE
    MONTH:NAME    YEAR:INTEGER    DAY:(BOUNDED-INTEGER 1  31)
    WEEKDAY:(MEMBER (Sunday Monday Tuesday Wednesday Thursday Friday Saturday))
```

24.3.1. 基于框架对话的控制结构

基于框架的对话系统的控制架构是围绕框架设计的，在苹果的 Siri、亚马逊的 Alexa 和谷歌助手等现代系统中以各种形式使用。系统的目标是用用户想要的填充器来填充框架中的槽，然后为用户执行相关操作(回答问题，或预订航班)。为了做到这一点，系统向用户提出问题(使用预先指定的问题模板与每个框架的每个槽相关联，如图 24.10 所示)，填满用户指定的任何槽(我们将在下一节中描述槽填充的工作原理)。如果用户的响应填满多个槽，如下所示:

(24.4) I want a flight from San Francisco to Denver one way leaving after five p.m. on Tuesday.

那么，系统将填满所有相关的槽，然后继续提问以填满剩余的槽，跳过与已填槽相关的问题。GUS 架构也有附加在槽上的条件-动作规则。例如，在用户指定了 DESTINATION(目的地)之后，将一个规则附加到飞机预订框架的目的地槽中，可以自动将该城市作为相关酒店预订框架的默认值 StayLocation(停留位置)。或者如果用户指定短程旅行的 DESTINATION DAY，系统可以自动输入 ARRIVAL DAY。

许多领域需要多个框架。除了用于汽车或酒店预订的框架外，我们可能还需要带有常规航线信息的框架(像这样的问题：哪些航空公司从波士顿飞往旧金山？)或有关机票惯例的信息(像这样的问题：我是否必须停留特定天数以获得体面的机票？)。系统必须能够消除给定输入应该填充到哪个框架的哪个槽的歧义，然后将对话控制切换到该框架。

由于这种需要动态切换控制，GUS 体系结构是一种产生式规则(production rule)的系统。不同类型的输入导致不同的产生式触发，每一个都可以灵活地填充不同的帧。然后，产生式规则可以根据用户的输入和一些简单的对话历史(如系统问的最后一个问题)等因素切换控制。

一旦系统有了足够的信息，它就会执行必要的操作(如查询航班数据库)并将结果返回给用户。

24.3.2. NLU 用以填充 GUS 中的槽

在基于框架的体系结构中，自然语言理解组件的目标是从用户的语段中提取三样东西。第一个任务是领域分类:例如，这个用户是否在谈论航空公司、编程闹钟或处理他们的日历?当然，对于只关注日历管理的单个域系统来说，这种 1-of-n 的分类任务是不必要的，但是多领域对话系统是现代标准。第二个任务是用户**意图确定(intent determination)**:用户试图完成的一般任务或目标是什么?例如，任务可以是查找电影、显示航班或删除日历预约。第二个任务是进行**槽填充(slot filling)**：从用户针对其意图的语段中提取出系统希望用户理解的特定槽和填充符。从这样的用户语段中：

Show me morning flights from Boston to San Francisco on Tuesday

一个系统可能想要构建这样的表示:

DOMAIN:	AIR-TRAVEL
INTENT:	SHOW-FLIGHTS
ORIGIN-CITY:	Boston
ORIGIN-DATE:	Tuesday
ORIGIN-TIME:	morning
DEST-CITY:	San Francisco

如果一个语段像这样：

Wake me tomorrow at 6

则将给出一个意图像这样：

DOMAIN:	ALARM-CLOCK
INTENT:	SET-ALARM
TIME:	2017-07-01 0600-0800

原始 GUS 系统中使用的填槽方法，在工业应用中仍然很常见，是使用手写规则，通常作为附加在槽或概念上的条件-动作规则的一部分。例如，我们可以定义一个正则表达式来识别 SET-ALARM 意图:

wake me (up) | set (the|an) alarm | get me up

基于规则的研究系统，如 Phoenix 系统(Ward 和 Issar, 1994)，由大量手工设计的带有数千条规则的**语义语法(semantic grammar)**组成。语义语法是一种上下文无关的语法，其中每个规则的左边对应于被表示的语义实体(即槽名)，如下面的片段所示:

SHOW	→ show me | i want | can i see |...
DEPART TIME RANGE	→ (after|around|before) HOUR | morning | afternoon | evening
HOUR	→ one|two|three|four...|twelve (AMPM)
FLIGHTS	→ (a) flight | flights
AMPM	→ am | pm
ORIGIN	→ from CITY
DESTINATION	→ to CITY
CITY	→ Boston | San Francisco | Denver | Washington

任何 CFG 解析算法都可以解析语义语法(见第 13 章)，结果是使用语义节点标签对输入字符串进行分级标签，如图 24.11 所示。

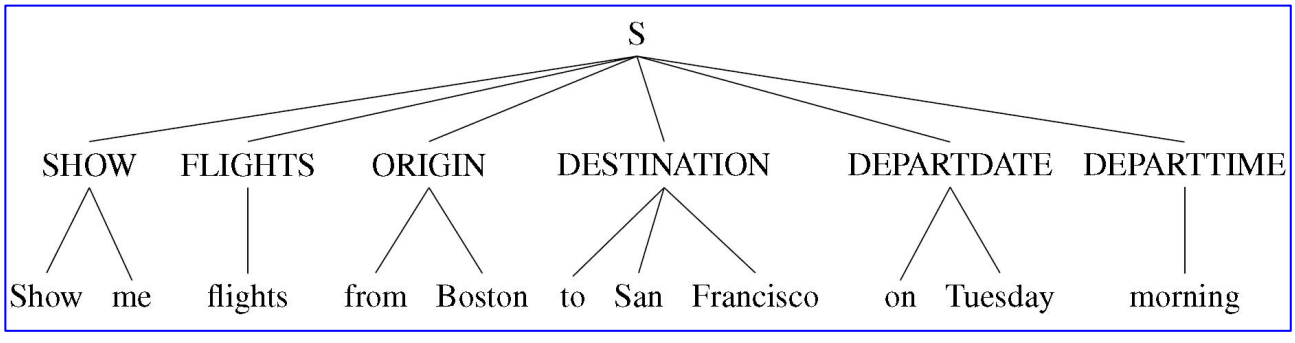


图 24-11：用户句子的语义语法解析

图注：图中使用槽名作为内部解析树节点。

它只需要把填充符放到某种规范的形式中，例如像第 17 章中讨论的那样，通过标准化日期。许多工业对话系统采用了 GUS 架构，但使用监督机器学习来填充槽，而不是这些规则;参见第 24.4.2 节。

24.3.3. 基于框架对话的其他组件

ASR(自动语音识别)组件从电话或其他设备接收音频输入，并输出转录的单词字符串，如第 26 章所述。

ASR 组件也可以根据对话状态进行制作。例如，如果系统刚刚问用户“你从哪个城市出发?”，ASR 语言模型组件可以被约束为具有城市名称的答案分配较高的概率。这可以通过训练一个语言模型来回答这个问题，或者为这些句子手写有限状态或上下文无关的语法来实现。完全依赖于对话状态的语言模型称为**限制语法(restrictive grammar)**;通常，真正的系统会将限制语法与更通用的语言模型混合在一起。

任何对话系统的自然语言生成模块都会生成系统对用户说的语段。基于框架的系统倾向于使用**基于模板的生成(template-based generation)**，在这种生成中，要对用户说出的句子中的所有或大部分单词都是由对话设计师预先指定的。由这些模板创建的句子通常称为**提示(prompt)**。模板可能是完全固定的(比如' Hello, how can I help you? ')，或者可以包含一些由生成器填充的变量，如下所示:

What time do you want to leave CITY-ORIG?

Will you return to CITY-ORIG from CITY-DEST?

即使在模板生成中，也可以做一些简单的基础。考虑 Cohen 等人(2004)的例子的不自然性:

(24.5) System: Did you want to review some more of your personal profile?

Caller: No.

System: What’s next?

如果没有确认，打电话的人就不知道系统已经理解了她的“No”。下面使用的 Okay 为模板响应 What’s next?添加了基础，使(24.6)的响应比(24.5)更自然。

(24.6) System: Did you want to review some more of your personal profile?

Caller: No.

System: Okay，what’s next?

基于规则的 GUS 方法在工业应用中非常普遍。与基于规则的信息提取方法一样，它具有精度高的优点，并且如果领域足够窄，并且有专家，也可以提供足够的覆盖。另一方面，手写规则或语法创建起来既昂贵又缓慢，而且手写规则可能会出现召回率(recall)问题。

24.4. 对话-状态架构

基于任务的对话的现代研究系统是基于框架架构的一个更复杂的版本，称为对话-状态或信念-状态架构。图 24.12 显示了一个典型对话-状态系统的六个组成部分。语音识别和合成组件处理语音处理;我们将在第 26 章讨论它们。

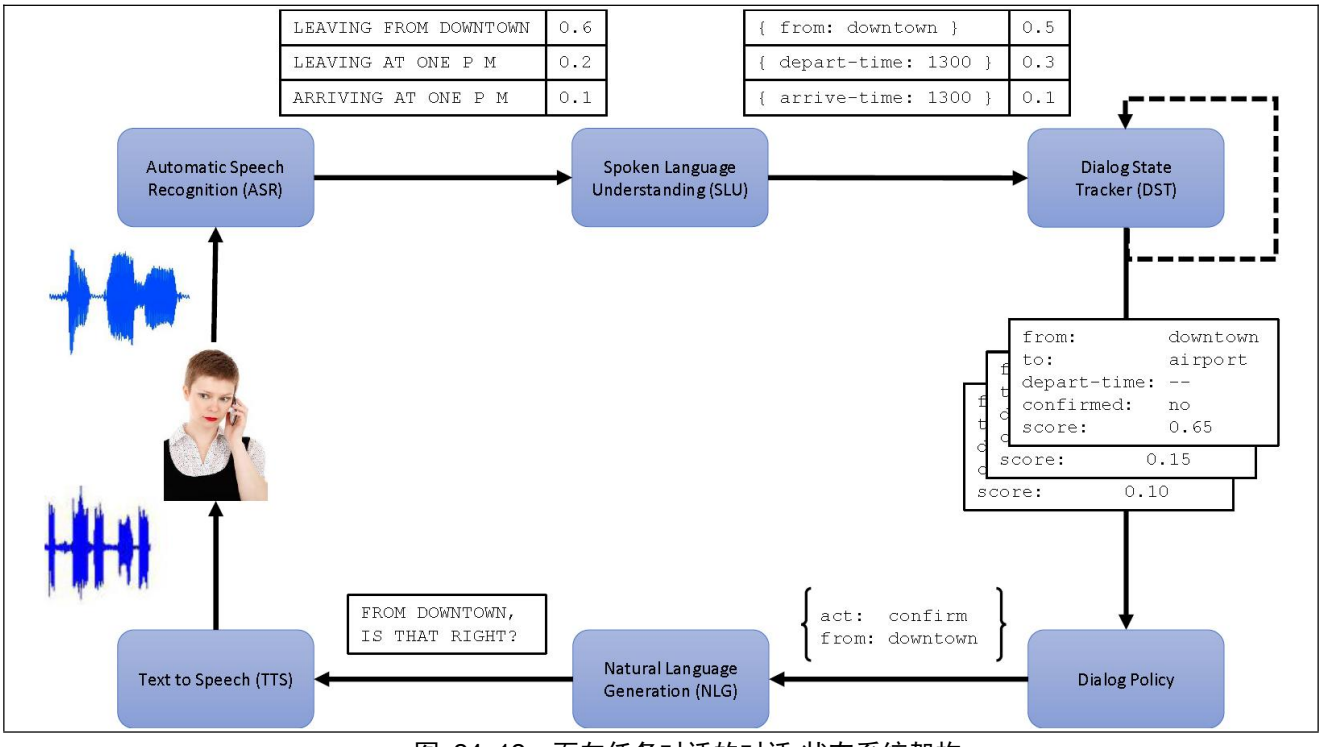


图 24-12：面向任务对话的对话-状态系统架构

因此，在本章的其余部分中，我们将考虑口头和文本对话系统的其他四个组件。这四个组件比简单的 GUS 系统中要复杂得多。例如，像 GUS 系统一样，对话-状态体系结构具有 NLU(自然语言理解)组件，可从用户的语段中提取槽填充词，但通常使用机器学习而非规则。对话状态跟踪器会保持对话的当前状态(包括用户最近的对话行为，以及用户迄今为止表达的整套槽填充限制)。对话策略决定系统下一步应该做什么或要说什么。GUS 中的对话策略很简单：提出问题，直到框架填满，然后报告一些数据库查询的结果。但是，更复杂的对话策略可以帮助系统决定何时回答用户的问题，何时向用户询问澄清问题，何时提出建议等等。最后，对话状态系统具有 NLG(自然语言生成)组件。在 GUS 中，生成器生成的句子全部来自预先编写的模板。但是，更复杂的生成组件可以根据确切的上下文条件来生成看起来更自然的话轮。

在撰写本文时，大多数商业系统都是架构的混合体，基于添加了一些对话-状态组件的 GUS 架构，但是在研究实验室中正在开发各种各样的对话-状态系统。

24.4.1. 对话行为

对话-状态系统利用**对话行为(dialogue acts)**。对话行为代表话轮或句子的互动功能，将言语行为的思想和基础结合成一个单一的表现形式。不同类型的对话系统需要标签不同类型的行为，因此标记集（定义对话行为到底是什么）往往是为特定的任务而设计的。

图 24.13 显示了一个餐厅推荐系统的标记集，图 24.14 显示了这些标记标签了 HIS 系统中的一个对话示例(Young 等人，2010)。这个例子还显示了每个对话行为的内容，它们是被交流的槽填充。所以用户可以 INFORM(通知)系统他们想要在博物馆附近吃意大利菜，或者 INFORM(确认)价格是否合理。

Tag	Sys	User	Description
HELLO(a = x，b = y，...)	√	√	Open a dialogue and give info a = x，b = y，...
INFORM(a = x，b = y，...)	√	√	Give info a = x，b = y，...
REQUEST(a，b = x，...)	√	√	Request value for a given b = x，...
REQALTS(a = x，...)	x	√	Request alternative with a = x，...
CONFIRM(a = x，b = y，...)	√	√	Explicitly confirm a = x，b = y，...
CONFREQ(a = x，...，d)	√	x	Implicitly confirm a = x，... and request value of d
SELECT(a = x，a = y)	√	x	Implicitly confirm a = x，... and request value of d
AFFIRM(a = x，b = y，...)	√	√	Affirm and give further info a = x，b = y，...
NEGATE(a = x)	x	√	Negate and give corrected value a = x
DENY(a = x)	x	√	Deny that a = x
BYE()	√	√	Close a dialogue

图 24-13：HIS 餐厅推荐系统使用的对话行为

图注：上图中 Sys 和 User 列分别指示哪些行为是有效的系统输出和用户输入。

Utterance	Dialogue act
U: Hi, I am looking for somewhere to eat.	hello(task = find,type=restaurant)
S: You are looking for a restaurant. What type of food do you like?	confreq(type = restaurant, food)
U: I'd like an Italian somewhere near the museum.	inform(food = Italian, near=museum)
S: Roma is a nice Italian restaurant near the museum.	inform(name = "Roma", type = restaurant, food = Italian, near = museum)
U: Is it reasonably priced?	confirm(pricerange = moderate)
S: Yes, Roma is in the moderate price range.	affirm(name = "Roma", pricerange =moderate)
U: What is the phone number?	request(phone)
S: The number of Roma is 385456.	inform(name = "Roma", phone = "385456")
U: Ok, thank you goodbye.	bye()

图 24-14：HIS 系统对话示例(使用图 24-13 中的对话行为)

24.4.2. 槽填充

槽填充任务，以及更简单的领域和意图分类任务，都是第 16 章讨论的监督语义解析任务的特殊情况，在该任务中，我们有一个训练集，将每个句子与正确的槽、领域和意图集合相关联。

一个简单的方法是训练一个序列模型，使其从输入的单词表示映射到槽填充词、领域和意图。例句:

I want to fly to San Francisco on Monday afternoon please

我们计算一个句子的表示，例如通过一个像 BERT 这样的上下文嵌入网络来传递这个句子。得到的句子表示可以通过一个前馈层，然后通过一个简单的 1-of-N 分类器来确定领域是 AIRLINE，意图是 SHOWFLIGHT。我们的训练数据是与 BIO 标签序列配对的句子:

O	O	O	O	O	B-DES	I-DES	O	B-DEPTIME	I-DEPTIME	O
I	want	to	fly	to	San	Francisco	on	Monday	afternoon	please

回想一下第 8 章，在 BIO 标记中，我们为每个槽标签的开头(B)和内部(I)引入了一个标记(tag)，为任何槽标签的外部(O)引入了一个符记(token)。标记个数为 2n+1 个，其中 n 为槽位数量。

图 24.15 显示了架构。输入是一系列的单词 $w_1 \ldots w_n$，通过上下文嵌入模型得到上下文词表示。这之后是一个前馈层和一个 softmax 在每个符记位置上的可能的 BIO 标记，连带输出一系列的 BIO 标记 $s_1 \ldots s_n$。我们还可以将领域-分类和意图-提取任务与槽填充结合起来，简单地添加一个领域，并将一个意图拼接起来，作为最终 EOS 符记的期望输出。

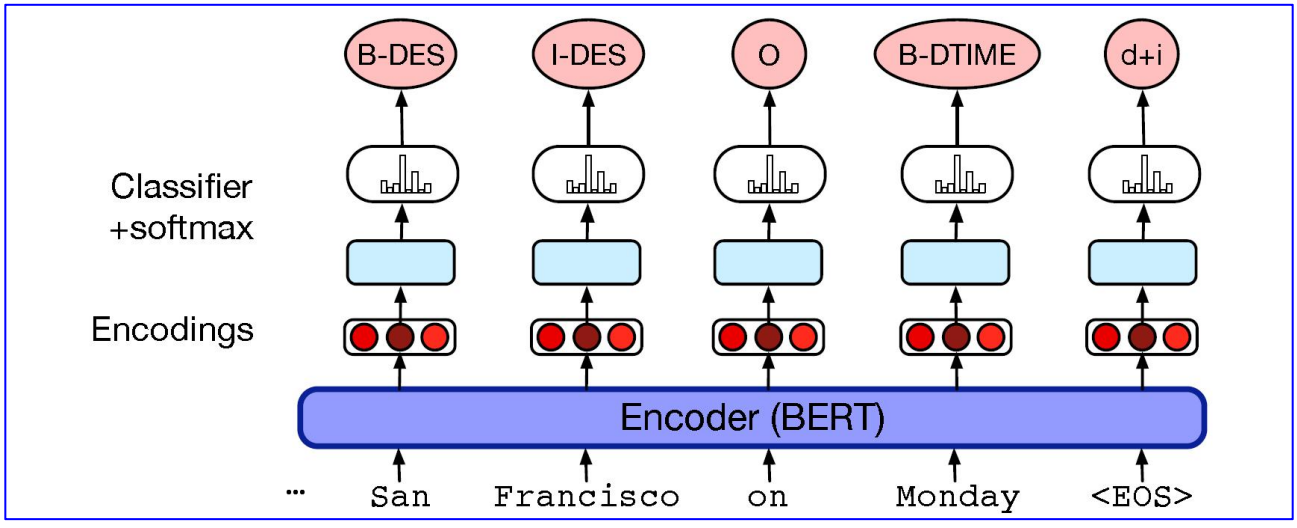


图 24-15：一种用于槽填充的简单架构

图注：它通过上下文嵌入(例如 BERT)将输入中的单词映射到输出分类器层(可以是线性的或更复杂的东西)，然后是 softmax 以生成一系列 BIO 标记(并包括一个最终状态，其中包含一个与意图拼接在一起的领域)。

序列标签器标记了用户的语段之后，就可以从标记(例如，“San Francisco”)中为每个槽提取填充字符串，然后可以将这些单词字符串规范化为本体中的正确形式(也许是机场代码‘SFO’)。这种规范化可以通过使用同音异义字典来实现(例如，指定 SF、SFO 和 San Francisco 是同一个地方)。

在工业上下文中，用于槽填充的基于机器学习的系统通常以半监督学习的方式从 GUS 样式的基于规则的系统中自举。首先为领域构建基于规则的系统，并仔细标签测试集。随着新用户语段的出现，它们与基于规则的系统提供的标签配对以创建训练元组。然后可以在这些元组上训练分类器，使用测试集测试分类器的性能，与基于规则的系统相对照。为了提高精度，可以使用一些启发式方法来消除错误的训练元组。当有足够的训练样本可用时，得出的分类器通常可以优于原始的基于规则的系统(Suendermann 等人，2009)，尽管基于规则的系统在处理诸如否定之类的复杂案例时仍可能保持较高的精度。

24.4.3. 对话状态跟踪

对话-状态跟踪器的工作是确定框架的当前状态(每个槽的填充符)，以及用户最近的对话行为。因此，对话-状态不仅包括当前句子中所表达的填充符，而且还包含了此时框架的整个状态，总结了所有用户的约束。下面的例子来自 Mrkšić等人(2017)，显示了每一轮对话状态跟踪器所需的输出:

User: I’m looking for a cheaper restaurant inform(price=cheap)
System: Sure. What kind - and where?
User: Thai food, somewhere downtown inform(price=cheap, food=Thai, area=centre)
System: The House serves cheap Thai food

User: Where is it? inform(price=cheap, food=Thai, area=centre); request(address)

System: The House is at 106 Regent Street

由于对话行为对槽和值有一定的限制，因此对话-行为检测和槽填充通常是共同完成的。考虑一下确定的任务:

I'd like Cantonese food near the Mission District

有这个结构：

inform(food=cantonese,area=mission).

对话行为的解释（在这个例子中，从对话行为的集合中选择 inform 来完成这项任务）是通过手工标签对话行为的监督分类训练完成的，基于表示当前输入句子和之前对话行为的嵌入来预测对话行为的标记。

最简单的对话状态跟踪器可能只是在每个句子之后获取一个槽填充序列模型的输出(第 24.4.2 节)。另外，一个更复杂的模型可以利用第 23 章的阅读理解体系结构。例如，Gao 等人(2019)的模型训练每个槽的分类器，以决定它的值是在当前句子中被改变了还是应该从前一个句子中继承过来。如果更改了槽值，则使用一个跨度-预测模型来预测带槽填充器的跨度的开始和结束。

特殊情况:检测纠正行为

一些对话行为之所以重要，是因为它们暗示着对话控制。如果一个对话系统错误地认识到或误解了一个语段，使用者通常会通过重复或重新构造该语段来纠正错误。因此，检测这些**用户纠正行为(user correction acts)**是非常重要的。讽刺的是，事实证明纠正实际上比正常的句子更难识别! 实际上，在一个早期的对话系统(TOOT 系统)中，纠正错误的频率是不纠正的两倍(Swerts 等人，2000)!原因之一是说话者有时会使用一种特定的韵律风格进行矫正，称为**超发音(hyperarticulation)**，其中发声包含夸张的能量，持续时间或 F0 等高线，例如 I said BAL-TI-MORE，not Boston(Wade 等人 1992 ，Levow 1998，Hirschberg 等人 2001)。即使他们没有进行超发音，沮丧的用户似乎也会以语音识别器更难的方式讲话(Goldberg 等人，2003)。

这些纠正的特点是什么?用户纠正往往是精确的重复，或者是省略一个或多个单词的重复，尽管它们也可能是原始语段的释义(Swerts 等人 2000)。检测这些重新制定或纠正行为可以是一般对话行为检测分类器的一部分。另外，由于这些行为的线索与简单行为(如 INFORM 或 request)的出现方式不同，我们可以利用与简单上下文嵌入特征正交的特征;一些典型特征如下(Levow 1998, Litman 等人 1999, Hirschberg 等人 2001, Bulyko 等人 2005, Awadallah 等人 2015):

features	examples
lexical	words like "no", "correction", "I don't", swear words, utterance length
semantic	similarity (word overlap or embedding dot product) between the candidate correction act and the user's prior utterance
phonetic	phonetic overlap between the candidate correction act and the user's prior utterance(i.e. "WhatsApp" may be incorrectly recognized as "What's up")
prosodic	hyperarticulation, increases in F0 range, pause duration, and word duration,generally normalized by the values for previous sentences
ASR	ASR confidence, language model probability

24.4.4. 对话策略

对话策略(dialogue policy)的目标是决定系统下一步应该采取什么行动，也就是说，对话行为会产生什么。更正式地说，在话轮 i 中，我们想要根据整个对话状态预测将采取何种行动 A_i。状态可能意味着来自系统(A)和用户(U)的整个对话序列，在这种情况下，任务将是计算:

$$\hat{A}_i = \text{argmax}_{A_i \in A} P(A_i | A_1, U_1, \ldots, A_{i-1}, U_{i-1}) \quad (24.7)$$

我们可以通过将用户表达的槽填充符集合作为对话状态来简化这一过程，并在许多不同的会话路径上折叠，从而导致相同的填充槽集合。

然后，这样的策略可能仅以当前对话状态为条件，仅由 Framei 框架的当前状态(哪些槽被填充以及填充什么)以及系统和用户的最后一次话轮表示：

$$\hat{A}_i = \text{argmax}_{A_i \in A} P(A_i | \text{Frame}_{i-1}, A_{i-1}, U_{i-1}) \quad (24.8)$$

这些概率可以通过神经分类器使用槽填充符(例如跨度)和语段(例如通过上下文嵌入计算出的句子嵌入)的神经表示来估计。

更复杂的模型通过强化学习来训练策略。为了决定采取何种行动，强化学习系统会在对话结束时获得奖励，并使用奖励来训练采取行动的策略。例如 Fazel-Zarandi 等人(2017)的电影推荐对话系统中，动作空间只有三个动作:EXECUTE(执行), CONFIRM(确认), 和 ELICIT(引出)。EXECUTE 向数据库发送一个查询，并回答用户的问题，CONFIRM 向用户阐明意图或槽(例如，“您想要克里斯托弗·诺兰执导的电影吗?”)，而 ELICIT 则向用户询问遗漏的信息(例如，“您说的是哪部电影?”)。如果对话系统最终以正确的槽表示结束，系统将获得较大的正奖励;如果槽是错误的，系统将获得较大的负奖励;对于确认和引出问题，系统将获得较小的负奖励，以防止系统再次确认所有内容。

策略示例:确认和拒绝

现代对话系统经常会出错。因此，对话系统必须确保对用户输入的解释是正确的。这通常通过两种方法来实现:确认对用户的理解和拒绝系统可能误解的语段。

正如我们在前一节中看到的，大多数系统都会引入与确认和拒绝相关的特定策略和行为。使用**显式确认(explicit confirmation)**策略时,系统询问用户一个直接问题来确认系统的理解,像下面的两个例子中,系统询问一个(黑体)是非确认问题(yes-no confirmation question):

S:	Which city do you want to leave from?
U:	Baltimore.
S:	**Do you want to leave from Baltimore?**
U:	Yes.
U:	I'd like to fly from Denver Colorado to New York City on September twenty first in the morning on United Airlines
S:	**Let's see then. I have you going from Denver Colorado to New York on September twenty first. Is that correct?**
U:	Yes

当使用**隐式确认(implicit confirmation)**策略时，系统可以将其理解作为一种基础策略，例如在询问下一个问题时重复系统的理解，如下面的两个例子所示:

U:	I want to travel to Berlin
S:	**When do you want to travel to Berlin?**
U2:	Hi I'd like to fly to Seattle Tuesday Morning
A3:	**Traveling to Seattle on Tuesday, August eleventh in the morning.** Your full name?

显式和隐式确认具有互补的优势。显式确认让用户更容易纠正系统的错误识别，因为用户只需对确认问题回答“no”即可。但是显式确认是尴尬的，并且增加了对话的长度(Danieli 和 Gerbino 1995, Walker 等人 1998)。上面的显式确认会话片段是非自然的，绝对非人类的;隐式确认在会话中更自然。

确认只是一种对话行为，系统可以借此表达对理解的缺乏。另一种选择是**拒绝(rejection)**，其中系统会向用户提示 I'm sorry, I didn't understand that。

有时，语段被多次拒绝。这可能意味着用户正在使用系统无法理解的语言。因此，当拒绝语段时，系统通常会遵循**逐步提示(progressive prompting)**或逐步升级细节的策略(Yankelovich 等人 1995，Weinschenk 和 Barker 2000)，如 Cohen 等人(2004)的示例：

System:	When would you like to leave?
Caller:	Well, um, I need to be in New York in time for the first World Series game.
System:	<reject>. Sorry, I didn't get that. Please say the month and day you'd like to leave.
Caller:	I wanna go on October fifteenth.

在此示例中，拒绝提示不仅仅是向您重复“When would you like to leave?”(您想何时离开？)”，它还为呼叫者提供了有关如何表达系统将理解的语段的更多指导。这些 you-can-say 帮助信息对于帮助提高系统

的理解性能非常重要(Bohus 和 Rudnicky，2005)。如果呼叫者的语段再次被拒绝，则提示可以反映出这一点(“I still didn’t get that(我仍然没有得到)”)，并为呼叫者提供更多指导。

错误处理的另一种策略是**快速重新提示(rapid reprompting)**，其中系统仅通过说“I’m sorry?”或者“What was that?”来拒绝发话。只有第二次拒绝呼叫者的发话，系统才会开始应用渐进式提示。科恩等人(2004)总结了一些实验，这些实验表明用户非常喜欢快速重新提示作为第一级错误提示。

除了使用对话状态表示之外，通常还可以使用丰富的特性来做出策略决定。例如，ASR 系统对语段的置信度可用于明确确认低置信度的句子。置信度是一种度量，语音识别器可以分配给句子的转录，以表明它对该转录的置信度有多大。置信度通常由语段的声学对数似然值来计算(越大的概率意味着越大的置信度)，但是韵律特征也可以用于置信度预测。例如，F0 长度较大或持续时间较长，或前面有较长停顿的语句，很可能会被错误识别(Litman 等人，2000)。

确认的另一个常见特性是出错的代价。例如，在实际预订航班或转账之前，显式确认很常见。系统可能有一个四层置信水平与三个阈值α，β，和γ:

$<\alpha$	low confidence	reject
$\geqslant\alpha$	above the threshold	confirm explicitly
$\geqslant\beta$	high confidence	confirm implictly
$\geqslant\gamma$	very high confidence	don’t confirm at all

24.4.5. 对话-状态模型中的 NLG

最后，一旦策略决定了要生成什么语音行为，自然语言生成(NLG)组件就需要生成响应给用户的文本。一旦确定了对话行为，我们需要生成响应给用户的文本。信息-状态体系结构中的 NLG 任务通常分为两个阶段建模:**内容规划(content planning)**(说什么)和**句子实现(sentence realization)**(如何说)。

这里我们假设内容规划是由对话策略完成的，该策略选择了要生成的对话行为，并选择了一些规划器想要告诉用户的属性(槽和值)(要么给用户一个答案，要么作为确认策略的一部分)。图 24.16 显示了句子实现阶段的一些输入/输出示例。在第一个例子中，内容规划器选择了对话行为 RECOMMEND(推荐)和一些特定的槽(名称、邻居、美食)及其填充。句子实现器的目标是生成如图中所示的第 1 行或第 2 行所示的句子，这是通过从大量标注对话的语料库中训练许多这样的表示/句子对的例子来实现的。

训练数据很难获得； 我们不太可能在许多可能使用不同措词的句子中看到每个可能拥有各种属性的餐厅。因此，在句子实现中通常用**去词汇化(delexicalization)**来增加训练示例的通用性。去词汇化是将训练集合中表示槽值的特定去词汇化单词替换为代表槽的通用占位符符记的过程。图24.17 显示了将图24.16 中的训练语句去词汇化的结果。

recommend(restaurant name= Au Midi, neighborhood = midtown, cuisine = french
1 Au Midi is in Midtown and serves French food.
2 There is a French restaurant in Midtown called Au Midi.
recommend(restaurant name= Loch Fyne, neighborhood = city centre, cuisine = seafood)
3 Loch Fyne is in the City Center and serves seafood food.
4 There is a seafood restaurant in the City Centre called Loch Fyne.

图 24-16：NLG 句子实现阶段的两个输入示例

图注：图中显示了对话行为和内容规划者预先指定的属性。第 1-2 行和第 3-4 行显示了由句子实现器生成的不同可能的输出句子。来自 Nayak 等人(2017)的餐厅推荐系统。

recommend(restaurant name= Au Midi, neighborhood = midtown, cuisine = french
1 **restaurant name** is in **neighborhood** and serves **cuisine** food.
2 There is a **cuisine** restaurant in **neighborhood** called **restaurant name.**

图 24-17：去词汇化的句子

图注：去词汇化的句子可以用来生成许多不同的再词汇化(relexicalized)的句子。来自 Nayak 等人(2017)的餐厅推荐系统。

从框架到去词汇化句子的映射通常由编码器解码器模型完成(Wen 等人 2015a, Wen 等人 2015b,

Mrkˇsi´c 等人 2017，除其他外)，在大型手工标记的任务导向对话语料库上进行训练(Budzianowski 等人 2018)。编码器的输入是一个表示对话行为及其领域的符记 x_t 序列。因此，对话行为 RECOMMEND(推荐)和属性/值配对 service:decent, cuisine:null 可以表示为一个平坦的符记序列(Nayak 等人，2017)，每个符记都映射到一个学到的嵌入 w_t，如图 24.18 所示。

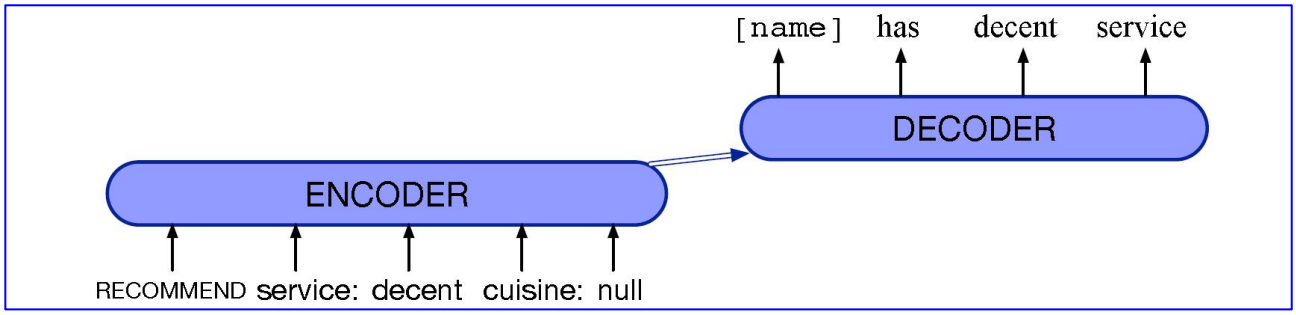


图 24-18：一个句子实现器将槽/填充符映射到英语

编码器读取所有的输入槽/值表示，解码器输出以下去词汇化的英语句子:

restaurant name has decent service

然后，我们可以使用来自内容规划器的输入框架来**再词汇化(relexicalize)**(填写准确的餐厅或邻居或美食)，从而得到:

Au Midi has decent service

生成澄清问题

也可以设计特定于独特对话行为的 NLG 算法。例如，在语音识别不能理解用户语段的某些部分时，考虑生成**澄清问题(clarification questions)**的任务。虽然可以使用通用对话行为 REJECT(拒绝)(“请重复”，或“我不明白你说什么”)，但人类会话的研究表明，人们反而会使用有目标性的澄清问题，重复误解的元素(Purver 2004, Ginzburg 和 Sag 2000, Stoyanchev 等人 2013)。例如，在以下假设的例子中，系统会重现“going”和“on the 5th”这两个词，以明确用户话轮的哪个方面是系统需要澄清的:

User: What do you have going to UNKNOWN WORD on the 5th?
System: Going where on the 5th?

有目标性的澄清问题可以通过规则创建(例如用“going where”代替“going to UNKNOWN WORD”)，或者通过构建分类器来猜测句子中哪些槽可能被错误识别(Chu-Carroll 和 Carpenter 1999, Stoyanchev 等人 2014, Stoyanchev 和 Johnston 2015)。

24.5. 评估对话系统

评价在对话系统设计中至关重要。聊天机器人和基于任务的系统通常受到不同的评价，因为它们有不同的目标;基于任务的系统必须完成预定航班的任务;聊天机器人有一个不同的目标，比如让用户感到愉快。

24.5.1. 评估聊天机器人

聊天机器人由人类通过打分来评估，可以是与聊天机器人交谈的人(参与者评估)，也可以是阅读人/聊天机器人对话记录的第三方(观察者评估)。

在 See 等人(2019)的参与者评估中，人类评估者用该模型进行了 6 个话轮的交谈，并在捕捉对话质量的 8 个维度对聊天机器人进行了评级:避免重复、趣味性、有意义、流畅性、倾听、好奇、人性化和参与度。几个例子:

参与度 您喜欢与该用户交谈多少？

•一点都不 •一点 •有点 •很多

避免重复 该用户有多重复？

•一遍又一遍地重复自己 •有时说两次相同的话 •总是说新话

有意义 该用户多久说一次没有意义的话？

•从来没有任何意义 •大多数答复都没有意义

•有些答复没有意义 •一切都有完美意义

观察者评估使用第三方注释器来查看完整会话的文本。有时候我们想让评分者为每个系统话轮分配分数;例如，让评分者标记每个话轮的连贯性。然而，通常我们只希望一个高分即可知道系统 A 是否优于系统 B。**急性评估(acute-eval)**度量(Li 等人，2019)是一种观察者评估，其中注释者查看两个单独的人类：计算机会话(A 和 B)，并选择对话系统参与者表现更好的对话(界面如图 24.19 所示)。

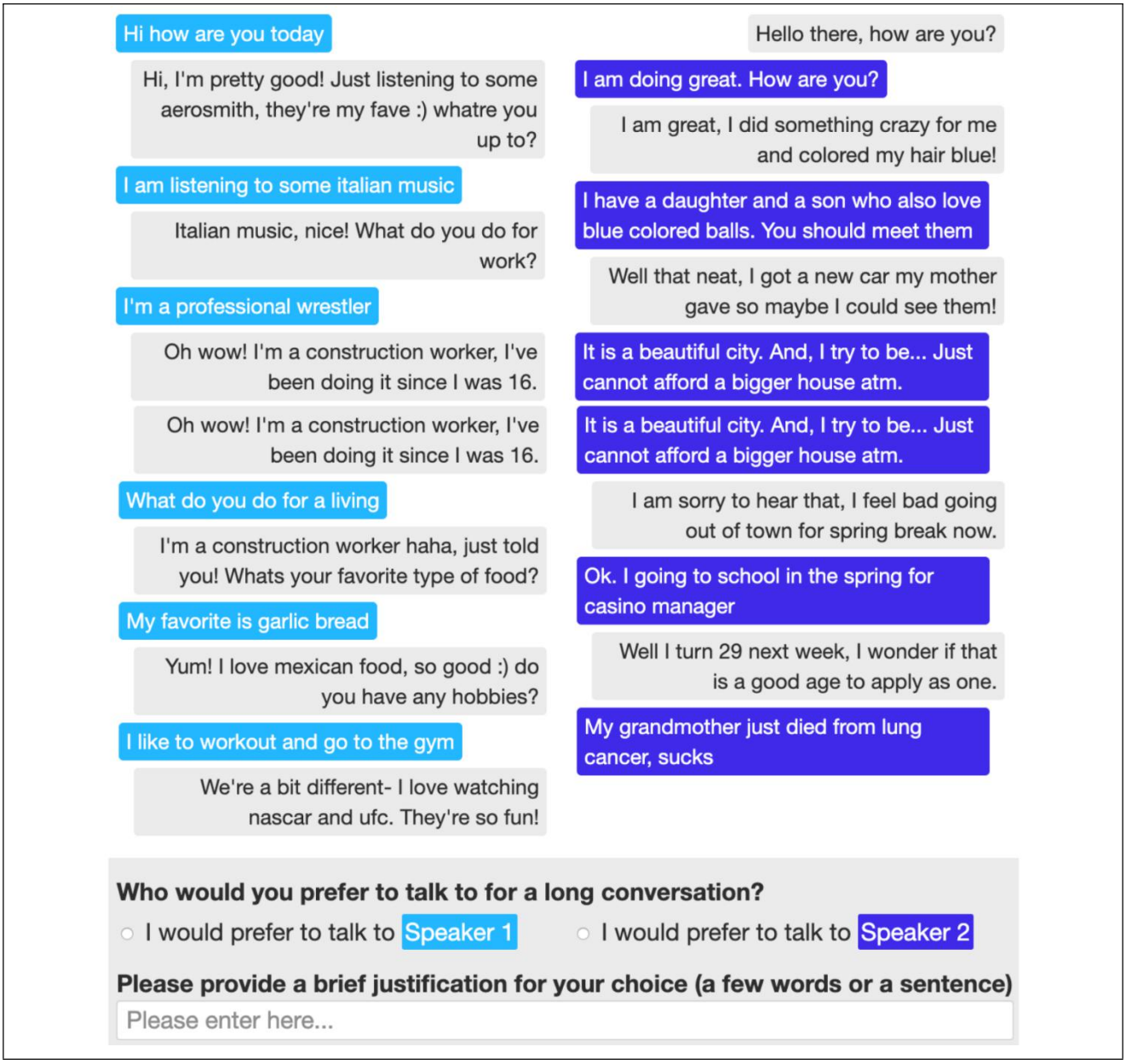


图 24-19：急性评估(acute-eval)方法

图注：要求注释人员对两个对话进行比较，在说话者 1(浅蓝色)和说话者 2(深蓝色)之间进行选择，而不受灰色尖顶的影响。

他们回答了以下 4 个问题(这些特殊措辞显示出了很高的共识):

参与度 您希望与谁进行长时间的交谈?

趣味性 如果您不得不说其中一位演讲者很有趣，而另一位则很无聊，那么您会说谁更有趣呢?

人性化 哪个演讲者听起来更人性化?

知识渊博 如果您不得不说其中一位演讲者知识渊博，而另一位则更无知，那么谁才是知识渊博呢?

自动评估通常不用于聊天机器人。这是因为，诸如 BLEU 或 ROUGE 等生成性能的计算度量，或在聊天机器人的反应和人类反应之间嵌入点积，与人类判断的相关性非常差(Liu 等人，2016)。这些方法的表现很差，因为任何话轮都有太多可能的反应;简单的单词重叠或语义相似度指标在响应空间小且词汇重叠的情况下效果最好，这对于机器翻译或可能的摘要等生成任务是正确的，但绝对不能进行对话。

然而，研究人员仍在继续进行更复杂的自动评估，而不仅仅是单词相似度。一种新颖的范式是**对抗性评价(adversarial evaluation)**(Bowman 等人 2016, Kannan 和 Vinyals 2016, Li 等人 2017)，其灵感来自图灵测试。这个想法是训练一个“图灵式”的评估者分类器来区分人类产生的反应和机器产生的反应。响应生成系统欺骗该评估者越成功，则系统越好。

24.5.2. 评估基于任务的对话

对于基于任务的对话，如果任务是没有歧义的，则我们可以简单地测量绝对的任务成功(系统是否预订了正确的飞机航班，或者在日历上放置了正确的事件)。为了获得用户快乐度的更细粒度的概念，我们可以计算用户满意度评分，让用户与对话系统交互来执行任务，然后让他们完成问卷调查。例如，图 24.20 显示了多项选择题的样本(Walker 等人 2001);回答被映射到 1 到 5 的范围内，然后对所有问题取平均值，以得到总的用户满意度评分。

TTS Performance	Was the system easy to understand ?
ASR Performance	Did the system understand what you said?
Task Ease	Was it easy to find the message/flight/train you wanted?
Interaction	Pace Was the pace of interaction with the system appropriate?
User Expertise	Did you know what you could say at each point?
System Response	How often was the system sluggish and slow to reply to you?
Expected Behavior	Did the system work the way you expected it to?
Future	Use Do you think you'd use the system in the future?

图 24-20：用户满意度调查

在系统的每次变更之后进行完整的用户满意度研究在经济上往往是不可行的。因此，拥有与人类满意度密切相关的绩效评估启发法是很有用的。我们研究了许多这样的因素和启发，通常分为两种标准:系统如何让用户以最少的问题(最小化成本)实现他们的目标(最大化任务成功):

任务完成成功: 通过评估整体解决方案的正确性来衡量任务的成功程度。对于基于框架的架构，这可能是槽错误率，即填满正确值的槽的百分比:

$$\text{Slot Error Rate for a Sentence} = \frac{\text{\# of inserted/deleted/subsituted slots}}{\text{\# of total reference slots for sentence}} \tag{24.9}$$

例如，考虑给定这样句子的一个系统:

(24.10) Make an appointment with Chris at 10:30 in Gates 104

提取了以下候选槽结构:

Slot	Filler
PERSON	Chris
TIME	11:30 a.m.
ROOM	Gates 104

这里的槽错误率是 1/3，因为 TIME(时间)是错的。除了错误率，还可以使用槽的精度、召回率和 F 分数。槽错误率有时也称为概念错误率。

有趣的是，有时用户对是否完成任务的感知比实际任务完成成功与否更能预测用户满意度(Walker 等人，2001)。

也许更重要但不太细致的衡量成功的方法是外部性指标，例如任务错误率。在这种情况下，任务错误率将量化在交互结束时将正确的会议添加到日历的频率。

效率成本:效率成本是衡量系统在帮助用户方面的效率。这可以通过对话的总耗时(以秒为单位)、总话轮数或系统话轮数，或者查询的总次数来衡量(Polifroni 等人，1992)。其他指标包括系统无响应的数量和“话轮修正率”：仅用于纠正错误的系统或用户话轮数除以总话轮数(Danieli 和 Gerbino 1995，Hirschman 和 Pao 1993)。

质量成本：质量成本衡量了影响用户对系统感知的交互的其他方面。一种这样的度量是 ASR 系统未能返回任何句子的次数或 ASR 拒绝提示的次数。类似的指标包括用户必须插入的次数(中断系统)或用户没有足够快速响应时播放的超时提示的数量。其他质量指标聚焦于系统对用户的理解和响应程度。最重要的是上述的槽错误率，但其他因素包括系统的问题、答案和错误消息的不恰当(冗长或模糊)，或者每个问题、答案或错误消息的正确性(Zue 等人 1989，Polifroni 等人 1992)。

24.6. 对话系统设计

与其他大多数语音和语言处理领域相比，用户在对话系统中扮演着更为重要的角色，因此对话系统的研究与人机交互(HCI)领域紧密相关。对话策略，提示和错误消息的设计通常称为**语音用户界面(voice user interface)**设计，通常遵循以用户为中心的设计原则(Gould 和 Lewis，1985)：

1.研究用户和任务：通过与用户进行访谈，调查相似的系统以及研究相关的人与人之间的对话来了解潜在用户和任务的性质。

2.建立模拟和原型：建立对话系统的关键工具是 **Wizard-of-Oz(绿野仙踪)**系统。在向导系统中，用户与他们认为是软件代理的人进行交互，但实际上是通过软件界面伪装的人类“向导”(Gould 等人 1983，Good 等人 1984，Fraser 和 Gilbert 1991)。该名称来自儿童读物《绿野仙踪》(Baum 1900 年)，其中的向导原来只是一个由窗帘或屏幕后面的人控制的模拟。



绿野仙踪系统可用于在实施之前测试体系结构。只需要接口软件和数据库即可。该向导从用户那里获取输入，具有一个图形界面，用于运行基于用户话语的示例查询，然后有一种输出句子的方法，可以打印(typing)句子，也可以从菜单中选择并打印句子。

Wizard-of-Oz 系统的结果也可以作为训练数据来训练飞行员对话系统。虽然 Wizard-of-Oz 系统非常普遍，但它们并不是完美的模拟;向导很难精确模拟真实系统的误差、限制或时间限制;因此，向导研究的结果有些理想化，但仍然可以提供关于领域问题的有用的第一个想法。

3. 对用户进行迭代测试：在系统设计中，具有嵌入式用户测试的迭代设计周期至关重要(Nielsen 1992，Cole 等人 1997，Yankelovich 等人 1995，Landauer 1995)。例如，在对话设计历史中的一个著名事件中，早期的对话系统要求用户按下一个键来中断系统(Stifelman 等人 1993)。但是用户测试显示用户被挡住了，这导致重新设计系统以识别重叠的语音。迭代方法对于设计促使用户以规范方式做出响应的提示也很重要。整合**价值敏感设计(value sensitive design)**也很重要，在设计过程中，我们会仔细考虑所产生的系统的收益、危害和潜在利益相关者(Friedman 等人 2017； Bender 和 Friedman 2018)。有很多关于对话界面设计的好书(Cohen 等人 2004，Harris 2005，Pearl 2017)。



24.6.1. 对话系统设计中的伦理问题

早在对话代理本身出现之前，伦理问题就被认为是设计人工代理的关键因素。玛丽·雪莱的小说《弗兰肯斯坦》(Frankenstein)的核心是她对创造代理人而不考虑伦理和人文关怀问题的经典讨论。一个重要的道德问题与安全有关:系统直接贬低或滥用其用户。一个相关的问题是由贬低特定社会群体的系统所造成的代表性伤害(Blodgett 等人，2020)。正如我们

在第 6.11 节中讨论的，任何类型的机器学习系统都倾向于复制训练数据中出现的偏差。这对于聊天机器人来说尤其重要，因为基于 IR 的和神经传导体系结构都是通过近似训练数据中的响应来进行响应的。

一个广为人知的例子是，微软 2016 年的**泰(Tay)**聊天机器人在上线 16 小时后就下线了，当时它开始发布带有种族歧视、阴谋论和对用户人身攻击的信息。Tay 已经从它的训练数据中学习了这些偏见和行为，包括从那些似乎正在对抗性攻击系统的用户那里，有意识地教它重复这种语言(Neff 和 Nagy，2016)。Wolf 等人(2017)得出结论，与用户交互的系统必须被设计成能够抵御这种对抗性攻击的健壮系统。

Henderson 等人(2017)研究了一些用于训练基于语料库的聊天机器人的标准对话数据集(如 Twitter、Reddit 或我们上面提到的电影对话)。他们发现了仇恨言论、攻击性语言和偏见的例子，尤其是在从 Twitter 和 Reddit 等社交媒体抽取的语料库中，无论是在原始训练数据中，还是在根据这些数据训练的聊天机器人的输出中。的确，Dinan 等人(2020)发现，基于 Transformer 的生成器模型放大了存在于该模型所训练的对话中的性别偏见。Liu 等人(2020)开发了另一种调查偏见的方法，测试神经对话系统对模拟用户话轮的反应，这些回合除了提到不同的性别或种族之外都是相同的。例如，他们发现，在句子中使用“他”而不是“她”这样的简单变化，会导致系统做出更具攻击性的反应，产生更多负面情绪。通过调查去偏方法来解决这些问题(例如训练系统对有毒环境进行检测和适当响应的方法)是当前重要的研究目标(Dinan 等人 2020, Xu 等人 2020)。

另一个重要的道德问题是隐私。在 ELIZA 成立的第一天，Weizenbaum 就指出了人们对聊天机器人的启示对隐私的影响。Henderson 等人(2017)指出，家庭对话代理人可能会意外地记录用户泄露的私人信息(例如“计算机，开灯–接听电话–嗨，是的，我的密码是...”)，然后可以将其用于训练会话模型。他们表明，当在标准语料库上对编码器/解码器对话模型进行培训时，标准语料库中增加了代表私人数据(例如标识号或密码)的训练密钥对(例如，密钥短语“社会安全号”后跟数字)，给出该密钥短语的对手是能够以接近 100% 的准确度恢复秘密信息。因此，根据人与人或人与机对话的文本进行训练的聊天机器人必须匿名处理个人身份信息。

最后，除了文本偏见，聊天机器人还提出了重要的性别平等问题。例如，目前绝大多数聊天机器人的名字都是女性的，这可能延续了女性仆人的刻板印象(Paolino 2017)。当用户使用性骚扰语言时，大多数商业聊天机器人会逃避或给予积极回应，而不是以明确的消极方式回应(Fessler，2017)。

这些伦理问题是重要的调查领域，其目标是为任何面向用户的系统(如对话系统)找到能够提供某种安全保障的方法(Henderson 等人，2017)。由于对话系统的定义包括人类参与者，研究人员也与他们所在机构的**机构审查委员会(Institutional Review Boards，IRB)**一起解决这些问题，以帮助保护实验对象的安全。

24.7. 总结

会话代理是重要的语音和语言处理应用程序，已经广泛应用于商业。

•在人类对话中，说话是一种行为;这些行为被称为言语行为或对话行为。演讲者也试图通过承认他们已经理解了对方来达成**共识**。会话还具有话轮结构和对话结构的特点。

•**聊天机器人**是会话代理，旨在模仿非正式的人类对话。ELIZA 等基于规则的聊天机器人及其现代后裔使用规则将用户的句子映射到系统响应中。基于语料库的聊天机器人挖掘人类对话日志，学习自动将用户的句子映射到系统响应中。

•对于基于任务的对话，大多数商业对话系统使用 **GUS** 或**基于框架的架构**，在这种架构中，设计师指定由系统必须通过询问用户来填充的槽组成的框架。

•**对话-状态**体系结构用更丰富的表示和更复杂的算法来跟踪用户的对话行为，生成自己对话行为的策略，以及自然语言组件来增强 GUS 框架和插槽体系结构。

•对话系统是一种人机交互(HCI)，一般的 HCI 原理应用于它们的设计，包括用户的角色，模拟诸如 Wizard-of-Oz 系统，以及迭代设计和在真实用户上测试的重要性。

24.8. 文献和历史说明

最早的对话系统是 ELIZA (Weizenbaum 1966)和 PARRY (Colby 等人，1971)这样的聊天机器人。ELIZA 有一个广泛的影响大众对人工智能的认知,并提出一些自然语言处理的第一个伦理问题—例如我们上面讨论的隐私问题以及算法在决策中的作用—引领其创造者 Joseph Weizenbaum 致力于为 AI 和计算机科学领域的社会责任而奋斗。

另一个早期系统，GUS 系统(Bobrow 等人，1977)在 20 世纪 70 年代末建立了基于框架的主要范式，并在 30 多年的时间里成为对话系统的主导工业范式。20 世纪 90 年代，首次应用于自然语言理解的随机模型开始被应用于对话槽填充(Miller 等人 1994, Pieraccini 等人 1991)。到 2010 年左右，GUS 架构终于开始在商业上广泛应用于手机对话系统，比如苹果 SIRI (Bellegarda 2013)和其他数字助手。

网络和在线聊天机器人的兴起使人们对聊天机器人产生了新的兴趣，并在世纪之交出现了基于语料库的聊天机器人体系结构，它首先使用了信息检索模型，然后在深度学习兴起后的 2010 年代，使用了序列到序列模型。会话中的语段是一种由说话者执行的行为，这一观点最初是由哲学家维特根斯坦(Wittgenstein，1953)提出的，但奥斯汀(Austin，1962)和他的学生约翰·塞尔(John Searle)更充分地提出了这一观点。多年来，人们对各种言语行为进行了定义，并形成了丰富的语言和哲学文献，特别是对间接言语行为的使用进行了解释。

对话行为的概念也来自许多其他来源，包括邻接对、预先序列以及在**会话分析(conversation analysis)**(见 Levinson(1983)对该领域的介绍)领域发展起来的人类会话的国际属性的其他方面。弗斯(Firth 1935,p. 70)也在一个著名的引文中预示了行为(acts)建立了强烈的局部对话期望:

> Most of the give-and-take of conversation in our everyday life is stereotyped and very narrowly conditioned by our particular type of culture. It is a sort of roughly prescribed social ritual, in which you generally say what the other fellow expects you, one way or the other, to say.
>
> (在我们的日常生活中，大多数的交流是刻板的，并且受我们特定文化类型的限制。这是一种大致规定的社交仪式，您通常以这样或那样的方式说出另一个人希望您说的话。)

另一个重要的研究线索将对话建模为一种合作行为，包括共同点(Clark 和 Marshall, 1981)、作为合作过程的参考(Clark 和 Wilkes-Gibbs, 1986)、联合意图(Levesque 等人，1990)和共享计划(Grosz 和 Sidner，1980)的想法。

有关对话行为的语言特性及其检测方法的分析工作也为对话状态模型提供了强有力的信息(Sag 和 Liberman 1975，Hinkelman 和 Allen 1989，Nagata 和 Morimoto 1994，Goodwin 1996，Chu-Carroll 1998，Shriberg 等人 1998，Stolcke 等人 2000，Gravano 等人 2012)。

我们在本章中无法涵盖的两条重要路线是会话结构的计算属性。

第一条路线，由 Bruce(1975)首次提议：由于言语行为是行动，因此应该像其他行动一样对它们进行计划，并借鉴 AI 计划文献(Fikes 和 Nilsson，1971)。一个想要找到一些信息的代理可以提出向对话者询问信息的计划。听到话语的代理可以通过“反向”运行计划来解释言语行为，使用推理规则从对话者所说的推断出计划可能是什么。基于计划的对话模型被称为 BDI 模型，因为这些规划者为代理和对话者的信念(B)、愿望(D)和意图(I)建模。对话的 BDI 模型最初是由 Allen, Cohen, Perrault 和他们的同事在一些有影响力的论文中介绍的，这些论文展示了言语行为是如何产生的(Cohen 和 Perrault 1979)和解释的(Perrault 和 Allen 1980, Allen 和 Perrault 1980)。同时，Wilensky(1983)引入了基于计划的理解模型作为解释故事任务的一部分。

第二条路线，聚焦于对话层次结构的建模。Grosz(1977b)的开创性论文首先表明“面向任务的对话的结构与正在执行的任务的结构非常相似”(第 27 页)，导致了她与 Sidner 和其他人的合作，表明了如何使用相似的意图概念，并计划为对话中的话语结构和连贯性建模。例如，参见 Lochbaum 等人(2000)总结的意向结构在对话中的作用。

将强化学习应用于对话的想法最早是在世纪之交 AT&T 和 Bell 实验室对 MDP 对话系统的研究(Walker 2000, Levin 等人 2000, Singh 等人 2002)和线索短语、韵律、拒绝和确认的研究中提出的。强化学习研究很快转向了更复杂的 POMDP 模型(Roy 等人 2000, Lemon 等人 2006, Williams 和 Young 2007)，该模型

应用于小的槽填充对话任务。

自成立以来，感情(affect)就一直在对话系统中发挥重要作用。在最近的研究中，Mairesse 和 Walker(2008)表明，如果会话代理符合用户的个性期望，用户会更好地接受它们。我们上面提到了 25k 个基于情感情境的对话的移情对话数据集(EMPATHETICDIALOGUES)(Rashkin 等人，2019)。Lin 等人(2019)使用了移情听众(MoEL)的混合物，每个都优化了对特定情感的反应，以产生移情反应。

[待定:深度强化学习的历史]。

[待定:调研:Tur 和 De Mori (2011)，Gao 等人(2019)]

[待定:在这里添加更多最近的历史,包括对话状态跟踪、NLG、端到端神经系统等]

24.9. 练习

24.1 Write a finite-state automaton for a dialogue manager for checking your bank balance and withdrawing money at an automated teller machine.

24.2 A **dispreferred response** is a response that has the potential to make a person uncomfortable or embarrassed in the conversational context; the most common example dispreferred responses is turning down a request. People signal their discomfort with having to say no with surface cues (like the word *well*), or via significant silence. Try to notice the next time you or someone else utters a dispreferred response, and write down the utterance. What are some other cues in the response that a system might use to detect a dispreferred response? Consider non-verbal cues like eye gaze and body gestures.

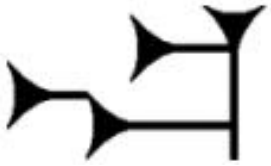
24.3 When asked a question to which they aren’t sure they know the answer, people display their lack of confidence by cues that resemble other dispreferred responses. Try to notice some unsure answers to questions. What are some of the cues? If you have trouble doing this, read Smith and Clark (1993) and listen specifically for the cues they mention.

24.4 Implement a small air-travel help system based on text input. Your system should get constraints from users about a particular flight that they want to take, expressed in natural language, and display possible flights on a screen. Make simplifying assumptions. You may build in a simple flight database or you may use a flight information system on the Web as your backend.

25. 语音学

组成我们在这本书中讨论的文本的字符，并不是随机的符号。它们也是一项惊人的科学发明:一种构成人类语言元素的理论模型。

我们知道的最早的书写系统(苏美尔语、汉语、玛雅语)主要是图形标识(logographic)：一个符号代表整个单词。但是从最早的阶段我们可以发现，一些符号也被用来代表构成单词的声音。右边的楔形符号叫 ba，在苏美尔语中的意思是“定量”，也可以纯粹用作声音/ ba /。我们拥有的最早的汉字刻在骨头上用于占卜，同样包含语音元素。纯粹的基于声音的书写系统，无论是音节(例如日语平假名)，字母(例如罗马字母)还是辅音(例如 Semitic 书写系统)，都可以追溯到早期的徽标音节系统，通常是两种文化的结合。因此，阿拉伯文、阿拉姆文、希伯来文、希腊文和罗马文系统都源自西方闪族人(Semitic)的文字，推测该文字已被西方闪族佣兵从草书形式的埃及象形文字中加以修改。日本的音节是由汉字的草书音标演变而来的，在唐代传入中国的佛经(Buddhist scriptures)中，汉字被用来代替梵语(Sanskrit)。



口语是由较小的语音单位组成的，这是一个隐含的想法，是语音识别(将波形转换为文本)和文本到语音(将文本转换为波形)算法的基础。在本章中，我们将提供有关**语音学(phonetics)**的计算观点，对世界语言使用的语音的研究，它们如何在人的声道中产生，如何在声音上实现，如何数字化和处理。

25.1. 语音和语音转写

像“p”或“a”这样的字母已经是人类语音的有用模型，我们将在第 26 章看到如何将字母和波形映射在一起。尽管如此，更抽象地表示声音还是有帮助的。我们将用一串**音子(phone)**来表示一个单词的发音，这些音子是语音，每个音子都用取自罗马字母的符号来表示。

ARPAbet Symbol	IPA Symbol	Word	ARPAbet Transcription	ARPAbet Symbol	IPA Symbol	Word	ARPAbet Transcription
[p]	[p]	parsley	[p aa r s l iy]	[iy]	[i]	lily	[l ih l iy]
[t]	[t]	tea	[t iy]	[ih]	[ɪ]	lily	[l ih l iy]
[k]	[k]	cook	[k uh k]	[ey]	[eɪ]	daisy	[d ey z iy]
[b]	[b]	bay	[b ey]	[eh]	[ɛ]	pen	[p eh n]
[d]	[d]	dill	[d ih l]	[ae]	[æ]	aster	[ae s t axr]
[g]	[g]	garlic	[g aa r l ix k]	[aa]	[ɑ]	poppy	[p aa p iy]
[m]	[m]	mint	[m ih n t]	[ao]	[ɔ]	orchid	[ao r k ix d]
[n]	[n]	nutmeg	[n ah t m eh g]	[uh]	[ʊ]	wood	[w uh d]
[ng]	[ŋ]	baking	[b ey k ix ng]	[ow]	[oʊ]	lotus	[l ow dx ax s]
[f]	[f]	flour	[f l aw axr]	[uw]	[u]	tulip	[t uw l ix p]
[v]	[v]	clove	[k l ow v]	[ah]	[ʌ]	butter	[b ah dx axr]
[th]	[θ]	thick	[th ih k]	[er]	[ɝ]	bird	[b er d]
[dh]	[ð]	those	[dh ow z]	[ay]	[aɪ]	iris	[ay r ix s]
[s]	[s]	soup	[s uw p]	[aw]	[aʊ]	flower	[f l aw axr]
[z]	[z]	eggs	[eh g z]	[oy]	[oɪ]	soil	[s oy l]
[sh]	[ʃ]	squash	[s k w aa sh]				
[zh]	[ʒ]	ambrosia	[ae m b r ow zh ax]				
[ch]	[tʃ]	cherry	[ch eh r iy]				
[jh]	[dʒ]	jar	[jh aa r]				
[l]	[l]	licorice	[l ih k axr ix sh]				
[w]	[w]	kiwi	[k iy w iy]				
[r]	[r]	rice	[r ay s]				
[y]	[j]	yellow	[y eh l ow]				
[h]	[h]	honey	[h ah n iy]				

图 25-1：英语辅音(左)和元音(右)的 ARPAbet 和 IPA 符号

用于转录世界语言的标准语音表示法是**国际音标(International Phonetic Alphabet，IPA)**，它是 1888 年首先开发的不断发展的标准。但是在本章中，我们将用 ARPAbet(Shoup，1980)来表示音子，这是一个

简单的语音字母(图 25.1)，它方便地使用 ASCII 符号表示 IPA 的美式英语子集。

IPA 和 ARPAbet 的许多符号都等同于熟悉的罗马字母。因此，例如，ARPAbet 音子[p]代表在鸭嘴兽(platypus)，美洲狮(puma)和车前草(plantain)的开始，豹子(leopard)的中间或羚羊(antelope)的末尾的辅音。但是，总的来说，英语拼写法字母和音子之间的映射相对不透明； 一个单个字母可以在不同的上下文中代表非常不同的声音。英文字母 c 对应于美洲狮(cougar)[k uw g axr]中的音子[k]，但对应于细胞(cell)[s eh l]中的音子[s]。音子[k]除了显示为 c 和 k 外，还可以显示为 x(狐狸 fox [faa ks])，ck(豺狼 jackal [jh ae k el])和 cc(浣熊 raccoon[r ae k uw n])的一部分。许多其他语言(例如西班牙语)在声音拼写法映射中比英语更加透明。

25.2. 发音语音学

发音语音学(articulatory phonetics)研究的是口腔、喉咙和鼻子中的各种器官如何调节肺部气流，从而产生音子。

25.2.1. 发声器官

图 25.2 显示了语音器官。空气的快速运动会产生声音。人类通过将空气从肺部通过气管(从技术上讲是 trachea)排出口腔然后从嘴或鼻子中排出，从而产生大多数口头语言的声音。空气通过气管时，会通过喉，通常称为亚当的苹果或语音盒。喉头包含两个小褶皱的肌肉，即声带(通常在非技术上称为 vocal cords)，可以一起移动或分开移动。这两个褶皱之间的空间称为**声门(glottis)**。如果褶皱靠在一起(但不是紧密闭合)，则当空气通过时它们会振动。如果它们相距很远，它们将不会振动。由声带褶皱产生并振动的声音称为**浊音(voiced)**；没有声带振动而发出的声音称为**清音(unvoiced)**或无声。浊音包括[b]，[d]，[g]，[v]，[z]和所有英语元音等。清音包括[p]，[t]，[k]，[f]，[s]等。

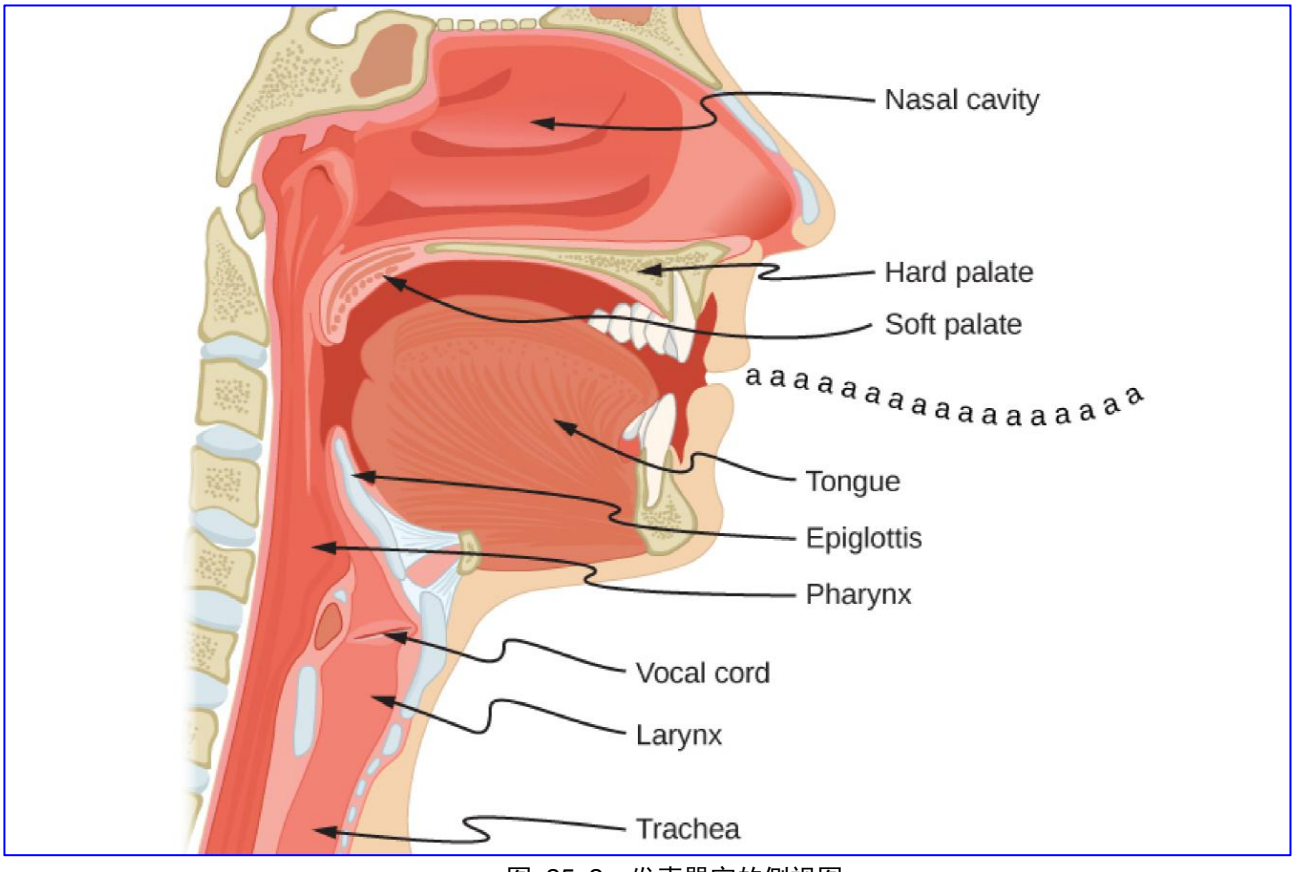


图 25-2：发声器官的侧视图

图注：来自 OpenStax University Physics, CC BY 4.0。

气管上方的区域称为**声腔(vocal tract)**。它由**口腔(oral tract)**和**鼻腔(nasal tract)**组成。空气离开气管后，它可以通过嘴或鼻子离开身体。大多数声音是由空气通过嘴发出的。空气通过鼻子发出的声音称为鼻

音；鼻音(例如英语[m]，[n]和[ng])使用口腔和鼻腔作为**共鸣腔(resonating cavities)**。

音子分为两大类:**辅音(consonants)**和**元音(vowels)**。这两种声音都是由空气通过嘴、喉咙或鼻子的运动形成的。辅音是由于气流受到阻碍而产生的，可以发声也可以不发声。元音的阻力较小，通常是发声的，而且通常比辅音响亮和持久。这些术语的技术用法很像普通用法;[p], [b], [t], [d], [k], [g], [f], [v], [s], [z], [r], [l],等等,都是辅音;(aa), (ae), (ao), (ih) (aw),(ow),(uw),等等,都是元音。半元音(semivowels)(如[y]和[w])具有两者的某些属性;它们像元音一样发声，但它们像辅音一样短而且音节少。

辅音:发音部位

由于辅音是通过阻挡气流产生的，我们可以根据它们的最大阻挡点、它们的**发音部位(place of articulation)**来将它们分类(图 25.3)。

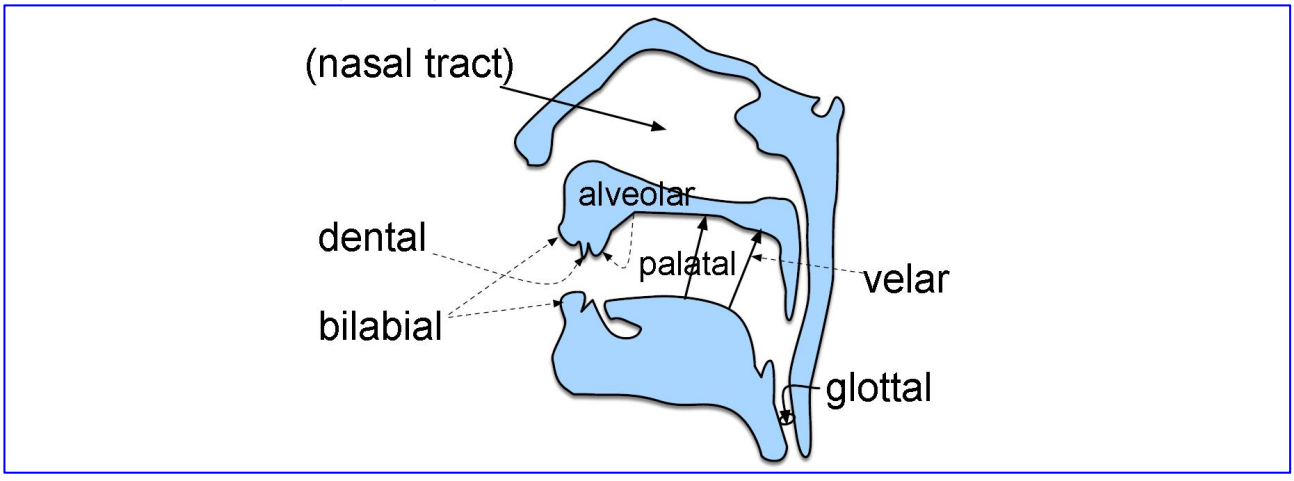


图 25-3：英语主要的发音部位

唇音(Labial):辅音的主要阻挡是由两个嘴唇凑在一起形成的，具有**双唇音(bilabial)**发音部位。在英语中，这些词包括 possum(负鼠)中的[p]， bear(熊)中的[b]和 marmot(土拨鼠)中的[m]。英语的**唇齿音(labiodental)**的辅音[v]和[f]是由下唇紧贴上排牙齿，让空气从上排牙齿的空隙中流过而成的。

齿音(Dental):舌头抵住牙齿发出的音是齿音。英语中主要的齿音是 thing(事物)中的[th]和 though(尽管)中的[dh]，它们是由舌头放在牙齿后面，舌尖在牙齿之间形成的。

齿龈音(Alveolar):齿龈是上排牙齿后面的口腔上颚的一部分。大多数说美式英语的人把舌尖靠在齿龈上而产生音子[s]、[z]、[t]和[d]。**舌面前音(coronal)**这个词通常用来同时指称齿音和齿龈音两者。

上腭音(Palatal):口腔的顶部(上腭)从上腭的齿龈嵴后部急剧上升。龈腭音(palato-alveolar)[sh](shrimp 小虾)，[ch](china 中国)，[zh](Asian 亚洲)和[jh] (jar 广口瓶)是由舌尖对着上升的齿龈脊后部发出的。牦牛 yak 的[y]发音是舌头的前部靠近上腭发出的。

软腭音(Velar):软腭是位于上腭最后部的一个可移动的肌肉瓣。声音[k](cuckoo 布谷鸟)，[g](goose 鹅)和[N](kingfisher 翠鸟)是把舌头后部贴在软腭上发出的。

喉音(Glottal):喉塞音[q]是通过关闭喉头(将声带连接在一起)产生的。

辅音:发音方式

辅音的不同之处还在于如何限制气流，例如通过完全停止空气或部分阻塞来限制气流。此特征称为辅音的**发音方式(manner of articulation)**。发音部位和发音方式的组合通常足以唯一地识别辅音。以下是英语辅音发音的主要方式：

塞音(stop)是一个辅音，其中气流在短时间内被完全阻塞。随着空气的释放，这种阻塞之后会伴随着爆破声。阻塞的阶段称为**成阻(closure)**，爆破的阶段称为**除阻(release)**。英语有浊塞音诸如[b]、[d]和[g]，也有清塞音诸如[p]、[t]和[k]。塞音也称为**爆破音(plosives)**。

鼻音(nasal sounds)[n]、[m]和[ng]是通过降低软腭让空气进入鼻腔而发出的。

在**摩擦音(fricatives)**中，气流受到压缩，但不会完全被切断。由于压缩而产生的湍流会产生一种特有

的“嘶嘶”声。英语的唇齿摩擦音[f]和[v]是通过将下唇压在上齿上，使上齿之间的气流受限而产生的。齿擦音[th]和[dh]让空气在牙齿之间的舌头周围流动。龈擦音[s]和[z]是舌头在齿龈脊上产生的，迫使空气流过牙齿的边缘。在腭龈擦音[sh]和[zh]中，舌头位于齿龈脊的后部，迫使空气通过舌头形成的凹槽。更高音调的摩擦音(英语[s]，[z]，[sh]和[zh])被称为**咝音(sibilants)**。先摩擦而后阻塞的塞音叫做**塞擦音(affricates)**，这包括英语[ch](chicken 鸡)和[jh](giraffe 长颈鹿)。

在**半元音(approximant)**中，两个发音部位靠得很近，但还没有近到造成强气流的程度。在英语[y](yellow 黄色)中，舌头靠近口腔顶部，但又没有接近到足够产生摩擦音的强气流流。在英语[w](wood 木头)中，舌的后部靠近软腭。美式[r]的形成至少有两种方式：舌尖伸展并靠近上腭，或者整个舌面隆起接近上腭。[l]是指舌尖向上抵住齿龈嵴或牙齿，舌头一侧或两侧降低，让空气流过。[l]之所以被称为**边音(lateral sound)**，是因为舌的两侧下降。

颤音(tap)或(flap)[dx]是舌对齿龈嵴的快速运动。单词 lotus ([l ow dx ax s])中间的辅音是大多数美式英语方言中的一个颤音;说许多英国方言的人会用[t]来代替。

元音

像辅音一样，元音也可以通过发音时的位置来区分。元音的最相关的参数有三个：第一个参数是元音高度,大致相当于舌头最高部分所处的位置；第二个参数是元音前(frontness)或后(backness),表示这个高点是朝向口腔的正面或后面；第三个参数是元音的圆状，嘴唇的形状是圆的或不是圆的。图 25.4 显示了不同元音时舌头的位置。

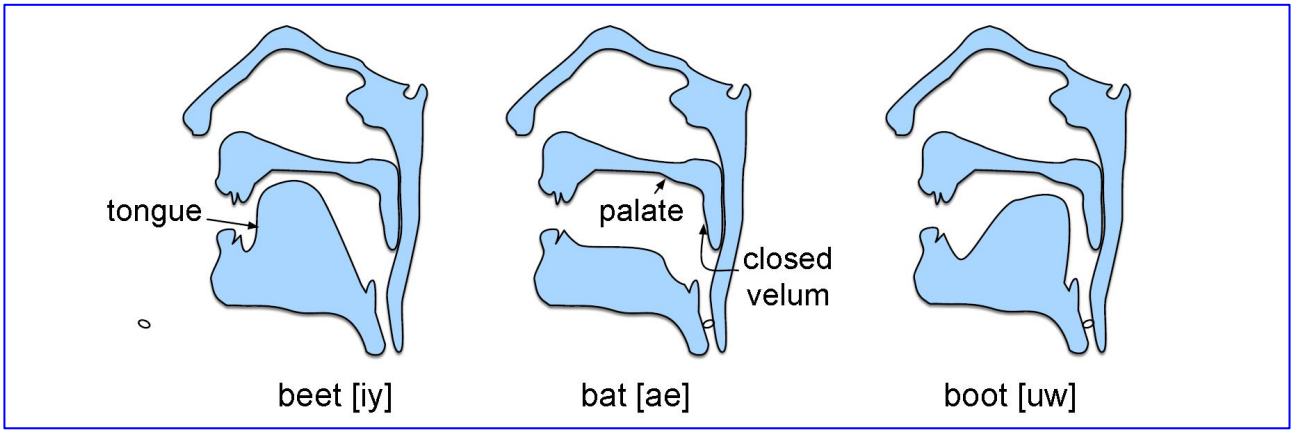


图 25-4：英语三个元音的舌位：前高[iy]，前低[ae]和后高[uw]

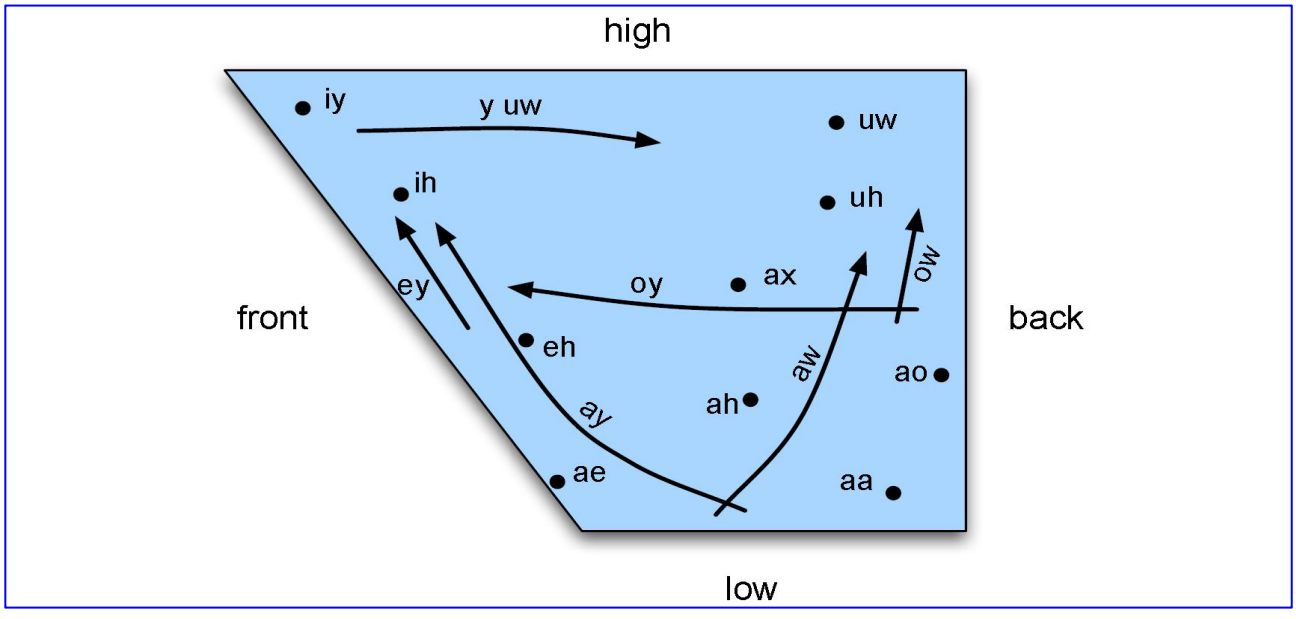


图 25-5：英语元音的“元音空间”示意图

例如，在元音[iy]中，舌头的最高点对着嘴的前部。相比之下，在元音[uw]中，舌头的最高点对着嘴的后部。舌头朝前伸的元音叫做**前元音(front vowels)**;那些舌朝后翘的元音叫做**后元音(back vowels)**。注意，虽然[ih]和[eh]都是前元音，但[ih]的舌头比[eh]要高。舌头的最高点相对较高的元音被称为**高元音(high vowels)**;舌头最大高度为中等或较低的元音分别称为中元音或低元音。

图 25.5 显示了不同元音高度的示意图。这是一个示意图，因为高度这一抽象属性仅仅与舌头的实际位置大致相关;事实上，它更准确地反映了声学事实。注意，这个图表有两种元音:一种是用点来表示舌高，另一种是用路径表示舌高。在产生元音时舌头位置发生显著变化的一个元音是**双元音(diphthong)**。英语中双元音特别丰富。

第二个重要的元音发音维度是嘴唇的形状。有些元音发音时嘴唇是圆的(和吹口哨时嘴唇的形状一样)。这些**圆唇元音(rounded vowels)**包括[uw]、[ao]和[ow]。

音节

辅音和元音结合在一起构成一个**音节(syllable)**。音节是由一个类似元音的声音(或响音[sonorant])和它周围的最紧密相关的辅音结合而成的。狗(dog)这个词有一个音节，[d aa g](在我们的方言中)； 猫薄荷(catnip)这个词有两个音节，[kaet]和[nhp]。我们把位于音节核心部分的元音称为**音节核(nucleus)**。初始辅音(如果有的话)称为**音节头(onset)**。具有多个辅音的音节头(如 strike[s t r ay k])被称为复杂音节头。**音节尾(coda)**是跟随在音节核后面的可选辅音或辅音序列。因此，对于 dog 单词，[d]是音节头，[g]是音节尾。**韵(rime)**，或韵脚(rhyme)，是音节核加音节尾。图 25.6 显示了一些音节结构例子。

自动将一个单词分解成音节的任务称为**音节切分(syllabification)**。音节结构也与语言的**音子配列(phonotactics)**密切相关。音子配列这个术语指的是音子在一种语言中相互跟随的限制。例如，英语对什么辅音可以一起出现在音节头有严格的限制;例如，序列[zdr]不能是一个合法的英语音节头。音子配列可以用语言模型或音子序列的有限状态模型来表示。

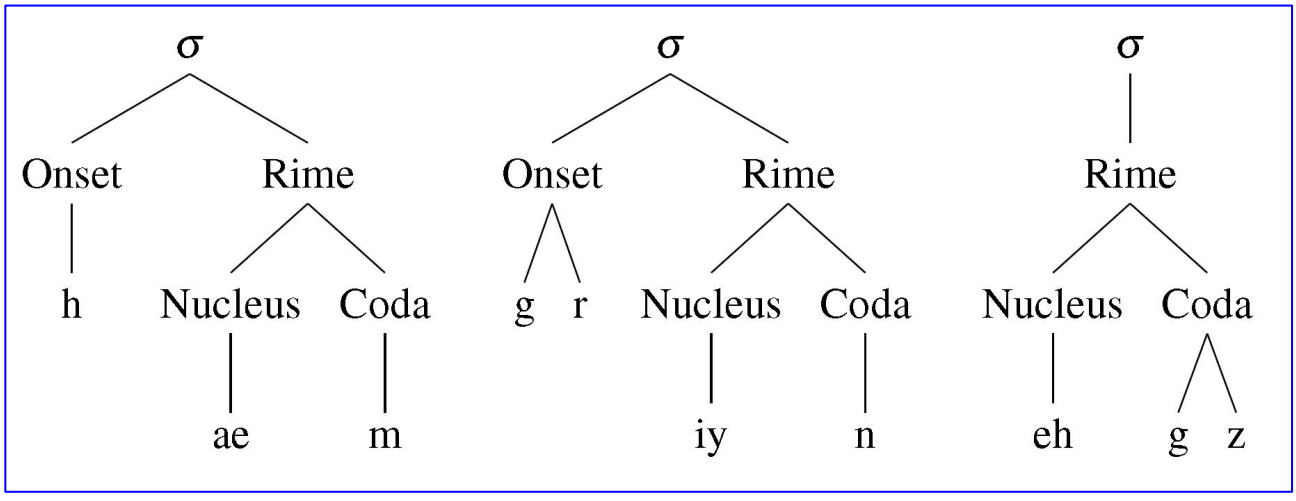


图 25-6：火腿(ham)、绿色(green)、鸡蛋(eggs)的音节结构(σ=音节)

25.3. 韵律学

韵律学(prosody)研究的是语言的语调和节奏方面，特别是使用 **F0**、**强度**和**持续时间**来传达语用的、情感的、或对话-互动的意义[1]。韵律可以用来标记话语结构，比如陈述句和疑问句之间的区别，或者对话的结构方式。韵律是用来标记一个特定的词或短语的显著性。韵律被大量用于副语言功能，如传达情感含义，如快乐、惊讶、或愤怒。而韵律在会话中管理依次发言起着重要的作用。

[1] 这个词在诗歌中有不同但相关的用法，意思是对诗歌韵律结构的研究。

25.3.1. 韵律突显度：重读、重音和中元音

在美式英语的自然发音中，有些词听起来比其他词更突显，这些词中的某些音节也比其他音节更突显。我们所说的**突显度(prominence)**是指这些单词或音节在听话者的感知上更加突显。在英语中，为了让一个单词或音节更加突出，说话者会提高音量，放慢语速(这样就会有更长的音长)，或者在发音过程中改变 F0，使其更高或更具变化性。

重读(accent) 我们通过一种称为**音高重读(pitch accent)**的语言标记来表示突出。突出的音高或音节被认为**负荷者(bear)**(与之相关的)音高重读。因此，可以通过重读下划线的单词来表达这种语段：
(25.1) I'm a little surprised to hear it characterized as happy.

词汇重音(lexical stress) 带有音高重读的音节称为重读音节。并非单词的每个音节都可以重读：音高重读必须在具有词汇**重音(stress)**的音节上实现。词汇重音是单词在词典中的发音的属性；如果单词是重读的，则带有词汇重音的音节会更响亮或更悠长。例如，surprised(惊讶)一词的重音在其第二个音节上而不是第一个音节上(尝试通过说出 SURprised 来发重音在第一个音节上，希望这对您来说是错误的)。因此，如果单词 surprised 在句子中接收到音高重读，则第二个音节将更强。下面的例子显示了下划线的重读单词，其中带重音音节的重读用黑体表示(更响、更长的音节)[2]：
(25.2) I'm a little surprised to hear it characterized as happy.

重音被标在字典里。例如，CMU 词典(CMU, 1993)在 counter: [K AW1 N T ER0]或 table: [T EY1 B AH0 L]的词条中，用 0(非重音)或 1(重音)标记元音。词汇重音的差异会影响单词意义;对于单词 content，作为名词读音为[K AA1 N T EH0 N T](意思是"内容")，作为形容词读音为[K AA0 N T EH1 N T](意思是"满意的")。

弱元音和中元音 非重音的元音可以进一步被减弱为**弱元:音(reduced vowels)**，其中最常见的是**中元音(schwa)**([ax])，如在 parakeet(长尾小鹦鹉)的第二个元音中一样： 读为[p ae r ax k iy t]。在弱元音中，发音姿势不如完整的元音完整。并非所有非重音的元音都减弱了； 任何元音，尤其是双元音，即使在非重音的位置也可以保持其完整的质量。非重读元音可以被进一步弱化为重读元音，其中最常见的是 Schwa ([ax])，如长尾鹦鹉的第二个元音:[p ae r ax k iy t]。在简化元音中，发音姿势不像在全元音中那样完整。并不是所有的非重读元音都减少了;任何元音，特别是双元音，即使在非重读位置也能保持其全部音质。例如，元音[iy]可以出现在 eat [iy t]中的重音位置，也可以出现在 carry [k ae r iy]中的非重音位置。

总之，有一个连续的韵律突显度，它通常是有用的代表级别，如重读、重音、全元音、弱元音。

25.3.2. 韵律结构

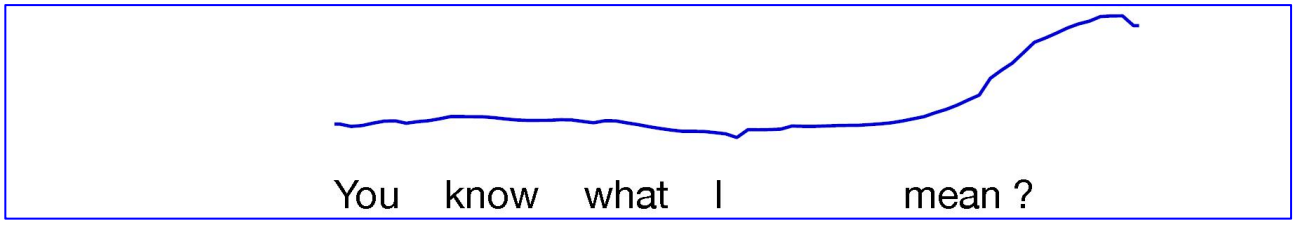
口语句子有韵律结构:有些词似乎自然地组合在一起，而有些词之间似乎有明显的中断或脱节。韵律结构通常用**韵律短语(prosodic phrasing)**来描述，意思是一个语段具有韵律短语结构，其方式与具有句法短语结构相似。例如，I wanted to go to London, but could only get tickets for France(我想去伦敦，但只能买到去法国的票)，这句话似乎有两个主要的**语调短语(intonation phrases)**，它们的边界出现在逗号处。此外，在第一个短语中，似乎还有另一组较小的韵律短语边界，通常称为**中间短语(intermediate phrases)**，它们把单词分开，就像 I wanted | to go | to London.一样。这些语调短语通常与句法结构成分相关(Price 等人 1991, Bennett 和 Elfner 2019)。

自动预测韵律边界对于 TTS 这样的任务很重要。现代方法使用序列模型，将原始文本或带有解析树等特征注释的文本作为输入，并在每个单词边界做出中断/不中断决策。它们可以在标签为韵律结构的数据上训练，如波士顿大学广播新闻语料库(Ostendorf 等人，1995)。

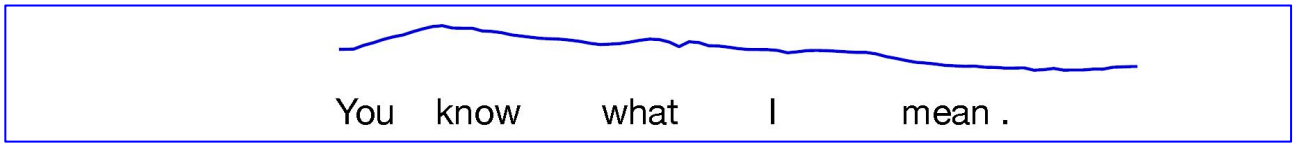
[2] 译者注：英语单词重音常被称为**重读(accent)**而不是**重音(stress)**，以区别于重音位置规则。但是为了简便，下面我们仍然使用"重音"这个比较熟悉的词语。---胡壮麟. 语言学教程[M]第五版中文本.北京:北京大学出版社，2019：38.

25.3.3. 音调

两个具有相同的突显度和短语模式的语段，由于有不同的**音调(tune)**，在韵律上也会有所不同。语段的音调是 F0 随时间的起伏。一个非常明显的关于音调的例子是英语中陈述句和**是非问句**之间的区别。同样的单词也可以用 F0 升调来表示是非问句(称为**问句升调(question rise)**):



或者 F0 的最后降低(称为**最后降调(final fall)**)来表示陈述句语调:



语言广泛使用音调来表达意义(Xu, 2005)。例如，在英语中，除了众所周知的**是非问句**的升调外，一个包含用逗号分隔的一系列名词的短语，在每个名词后面通常有一个称为**连续升调(continuation rise)**的短暂升调。其他的例子包括英语特有的表达矛盾和惊奇的轮廓。

突显与音调的联系

音高重读有不同的种类，它们与音调有关;例如，高的音高重读与低的音高重读有不同的功能。在不同的语言中有许多口音类的类型。其中一种类型是语调的**声调和间断索引(Tone and Break Indices，ToBI)**理论的一部分(Silverman 等人，1992)。ToBI 中的每个单词都可以与图 25.7 中所示的五种音高重读类型中的一种相关联。ToBI 中的每个语段由一系列的语调短语组成，每个短语以图 25.7 所示的四种**边界声调(boundary tones)**中的一种结尾，这四种边界声调代表了语段音调的最终体。ToBI 有多种语言版本。

Pitch Accents		Boundary Tones	
H*	peak accent	L-L%	“final fall”:“declarative contour” of American English
L*	low accent	L-H%	continuation rise
L*+H	scooped accent	H-H%	“question rise”: cantonical yes-no question contour
L+H*	rising peak accent	H-L%	final level plateau
H+!H*	step down		

图 25-7：ToBI 重读和边界声调标签

图注：美国英语语调的 ToBI 转录系统的重读和边界声调标签(Beckman 和 Ayers 1997，Beckman 和 Hirschberg 1994)。

25.4. 声学语音和信号

首先，我们简要介绍了声波波形及其数字化和频率分析；鼓励感兴趣的读者查阅本章末尾的参考资料。

25.4.1. 波

声学分析是基于正弦和余弦函数的。图 25.8 显示了一个正弦波图，特别是函数：

$$y = A*\sin(2\pi ft) \tag{25.3}$$

其中，我们把振幅 A 设为 1，频率 f 设为每秒 10 周。

回想一下基础数学，波的两个重要特征是它的**频率(frequency)**和**振幅(amplitude)**。频率是波在一秒钟内自我重复的次数，也就是循环的周数(cycles)。我们通常以每秒的周数来测量频率。图 25.8 中的信号在 0.5 秒内重复 5 次，即每秒 10 个周期。每秒钟的周期通常称为赫兹(hertz，缩短为 Hz)，因此图 25.8 中的频率将被描述为 10Hz。正弦波的振幅 A 是 Y 轴上的最大值。波的**周期(period)**T 是完成一个周期的振动所需要的时间，定义为：

$$T = 1/f \tag{25.4}$$

图 25.8 中的每一周持续十分之一秒，因此 T =0.1 秒。

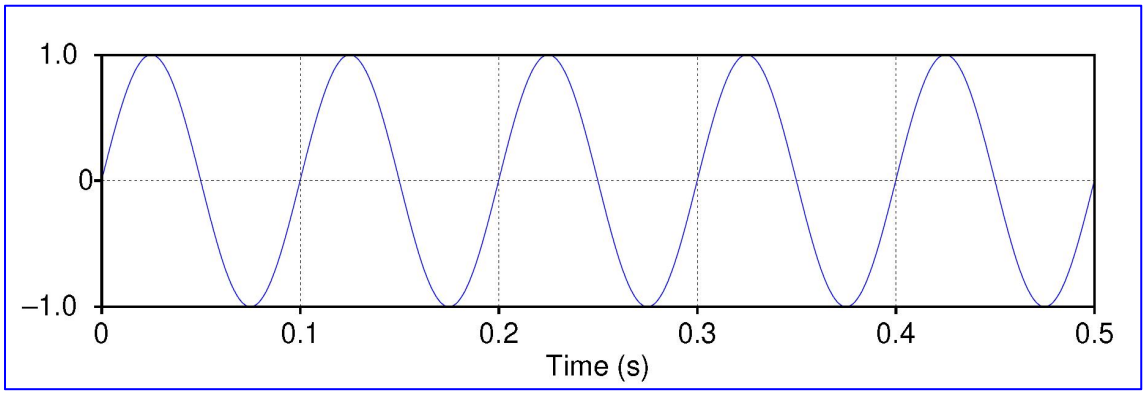


图 25-8：频率为 10Hz，振幅为 1 的正弦波

25.4.2. 语音声波

让我们从假设的波转向声波。语音识别器的输入就像人耳的输入一样，是一系列复杂的气压变化。气压的这些变化显然是由说话者引起的，并且是由空气通过声门进入口腔或鼻腔的特定方式引起的。我们通过绘制气压随时间的变化来表示声波。有时可以帮助您理解这些图表的一个比喻是一块垂直板，该板阻挡了气压波(也许在扬声器口前的麦克风中，或者在听众耳中的鼓膜中)。该图测量了该板上空气分子的压缩量或稀薄量(解压量)。图 25.9 显示了取自 Switchboard 语料库的一段短波形，其中元音[iy]来自某人说“she just had a baby(她刚生了一个孩子)”。

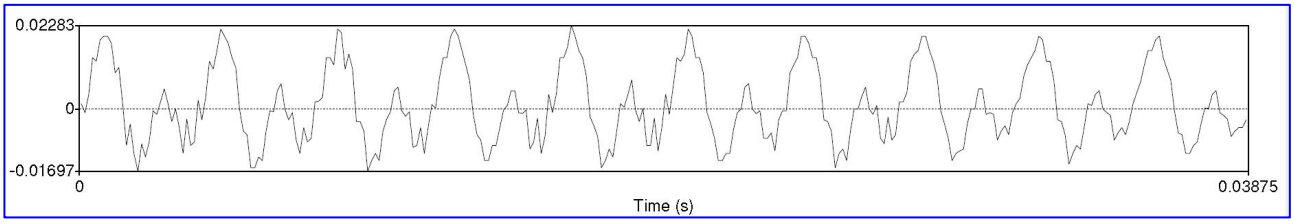


图 25-9：某语段中元音[iy]的波形图

图注：来自图 25.13 中的元音[iy]的波形图。y 轴表示空气压力高于和低于正常大气压的水平。x 轴表示时间。注意这个波有规律地重复。

将声波数字化(如图 25.9 所示)的第一步是将模拟表示(首先是气压，然后是麦克风中的模拟电信号)转换为数字信号。这种模数转换有两个步骤：**采样(sampling)**和**量化(quantization)**。为了对信号进行采样，我们在特定时间测量其幅度。采样率是每秒采样的数量。为了准确地测量波，每个周期中至少必须有两个样本：一个测量波的正侧部分，另一个测量波负侧部分。每个周期多于两个采样可提高幅度精度，但少于两个采样会使波的频率完全丢失。因此，可以测量的最大频率波是其频率为采样率一半的波(因为每个周期需要两次采样)。给定采样率的最大频率称为**奈奎斯特频率(Nyquist frequency)**。人类语音中的大多数信息的频率都低于 10,000 Hz。因此，要获得完全的准确性，就需要 20,000 Hz 的采样率。但是电话语音是由交换网络过滤的，电话只能发送小于 4,000 Hz 的频率。因此，像 Switchboard 语料库一样，8,000 Hz 的采样率足以满足电话带宽语音的需求，而麦克风语音通常使用 16,000 Hz 的采样率。

即使是 8000 赫兹的采样率，每秒的语音也需要进行 8000 次振幅测量，因此有效地存储振幅测量值是很重要的。它们通常以整数形式存储，要么 8 位(-128-127 的值)，要么 16 位(-32768-32767 的值)。这个将实值数字表示为整数的过程称为**量化(quantization)**，因为两个整数之间的差异作为最小粒度(量化范围)，所有接近于这个量化范围的值都以相同的方式表示。

一旦数据被量化，它将被存储在各种格式中。这些格式的一个参数是上面讨论的采样率和样本量。电话语音通常以 8 kHz 采样并存储为 8 位样本，而麦克风数据通常以 16 kHz 采样并存储为 16 位样本。另一个参数是**通道(channel)**数。对于立体声数据或两方对话，我们可以将两个通道存储在同一文件中，也可

以将它们分别存储在单独的文件中。最后一个参数是线性或压缩的单个样本存储。用于电话语音的一种常见压缩格式是μ-law(通常写为 u-law，但仍发音为 mu-law)。对数压缩算法(如μ-law)的直觉是，人的听力在小强度下比大强度下更敏感。对数以较小的忠诚度表示较小的值，但以较大的错误为代价。线性(非对数)值通常称为线性**脉冲编码调制(Pulse Code Modulation，PCM)**值。这是将线性 PCM 样本值 x 压缩为 8 位μ-law(其中 8 位的μ = 255)的公式：

$$F(x) = \frac{\mathbf{sgn(x)log(1+\mu|x|)}}{\mathbf{log(1+\mu)}} \quad , \quad -1 \le x \le 1 \qquad (25.5)$$

有很多标准的文件格式可以存储数字化的波文件，比如微软的.wav 和苹果的 AIFF，它们都有特殊的文件头;也使用简单的无头“原始”文件。例如，.wav 格式是微软多媒体文件 RIFF 格式的子集;RIFF 是一种通用格式，可以表示一系列嵌套的数据块和控制信息。图 25.10 显示了一个简单的.wav 文件，其中包含单个数据块及其格式块。

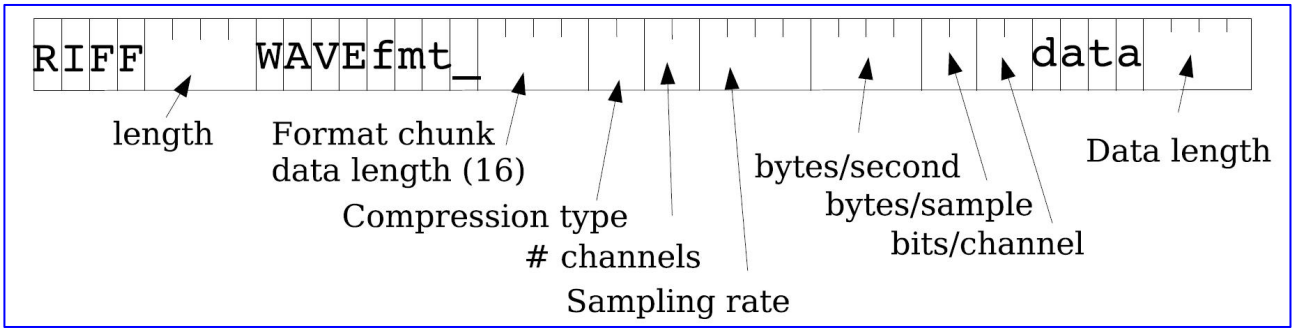


图 25-10：Microsoft wavefile 头文件格式

图注：Microsoft wavefile 头文件格式，假设文件很简单，只有一个块(chunk)。紧随这个 44 字节文件头的是数据块。

25.4.3. 频率和振幅;音调和响度

声波，像所有的波一样，可以用频率、振幅和我们前面介绍的纯正的正弦波的其他特性来描述。在声波中，它们不像正弦波那样容易测量。让我们考虑频率。在图 25.9 中注意到，虽然不是精确的正弦波，但是波是周期性的，图中捕获了在 38.75 毫秒内重复 10 次(0.03875 秒)。因此，这段波的频率是 10/0.03875 或 258Hz。

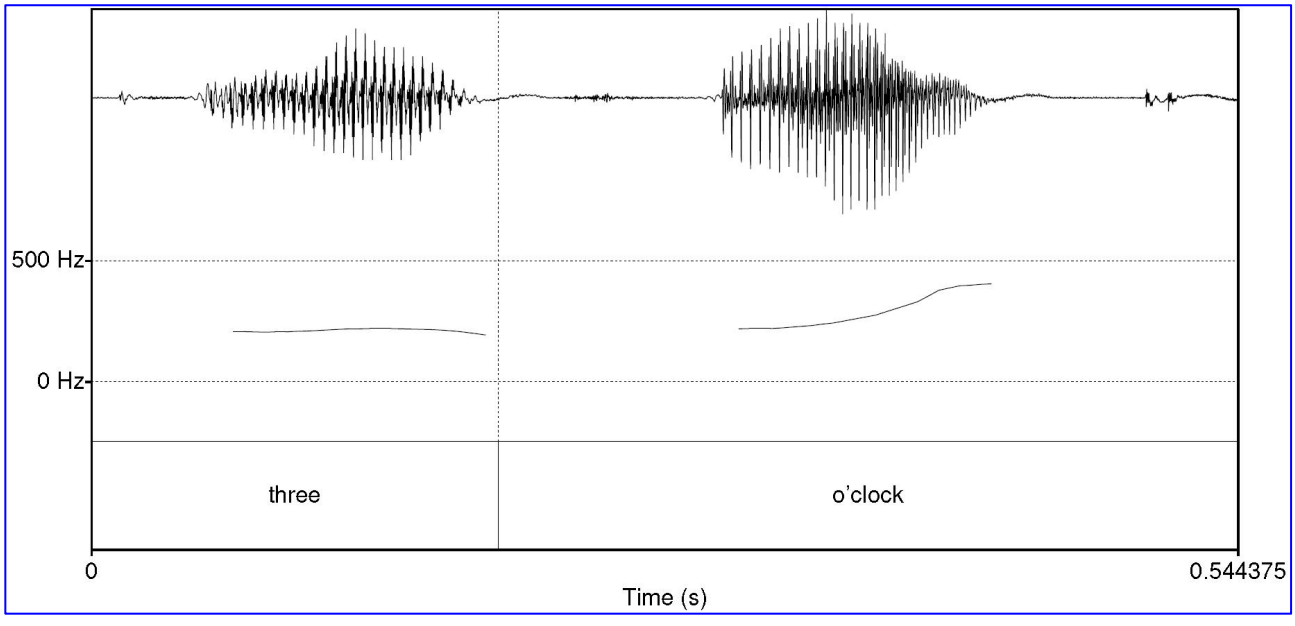


图 25-11：问句“Three o’clock?”的音高轨迹

图注：音高轨迹显示在波文件下面。注意问句终端 F0 的升调。注意在非常平静的部分(“o’ clock”中的“o”)没有音高轨迹;自动音高跟踪是基于浊音区域中的脉冲计数。如果没有浊音(或声音不足)，则不起作用。

这个 258 赫兹的周期性波是从哪里来的?它来自声带振动的速度;由于图 25.9 中的波形来自元音[iy]，所以它是浊音。请记住，发浊声是由声带的定期开启和闭合引起的。当声带打开时，空气通过肺部向上推动，形成一个高压区域。当声带闭合时，就不会有来自肺部的压力。因此，当声带振动时，我们期望看到图 25.9 所示的振幅有规律的峰值，每个主要峰值对应声带的开口。声带振动的频率，或复杂波的频率，称为波形的**基频(fundamental frequency，F0)**。我们可以在**音高轨迹(pitch track)**中绘制 F0 随着时间的变化。图 25.11 显示了一个简短问句“Three o'clock ?”在波形下面表示。注意问句终端 F0 的升调。

图 25.9 中的纵轴测量的是空气压强的变量;压强是单位面积上的压力，用帕斯卡(Pa)度量。纵轴数值越高(振幅越高)表示该点空气压强越大，零值表示正常(大气)气压，负值表示低于正常气压(稀薄度)。

除了这个在任何时间点的振幅值之外，我们还经常需要知道在某个时间范围内的平均振幅，给我们一些关于气压的平均变化有多大的概念。但是我们不能只对一个范围内的振幅值取平均值;正数和负数(大部分)会抵消掉，留给我们一个接近于零的数字。相反，我们通常使用 **RMS(root-mean-square，均方根)**振幅，它在平均(使其为正)之前对每个数字进行平方，最后在结束时取平方根。

$$\text{RMS } amplitude_{i=1}^{N} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{25.6}$$

信号的**强度(power)**与振幅的平方有关。如果一个声音的采样数为 N，则其强度为：

$$\text{Power} = \frac{1}{N}\sum_{i=1}^{N} x_i^2 \tag{25.7}$$

而不是强度，我们更多地是指声音的**音强(intensity)**，它将强度归一化为人类听觉阈值，并以 dB 为单位进行测量。如果 P_0 是听阈压力= 2×10^{-5} Pa，则音强定义如下：

$$\text{Intensity} = 10\log_{10}\frac{1}{N\cdot P_0}\sum_{i=1}^{N} x_i^2 \tag{25.8}$$

图 25.12 显示了句子 Is it a long movie? 的音强图，来自 CallHome 语料库，再在下方显示波形图。

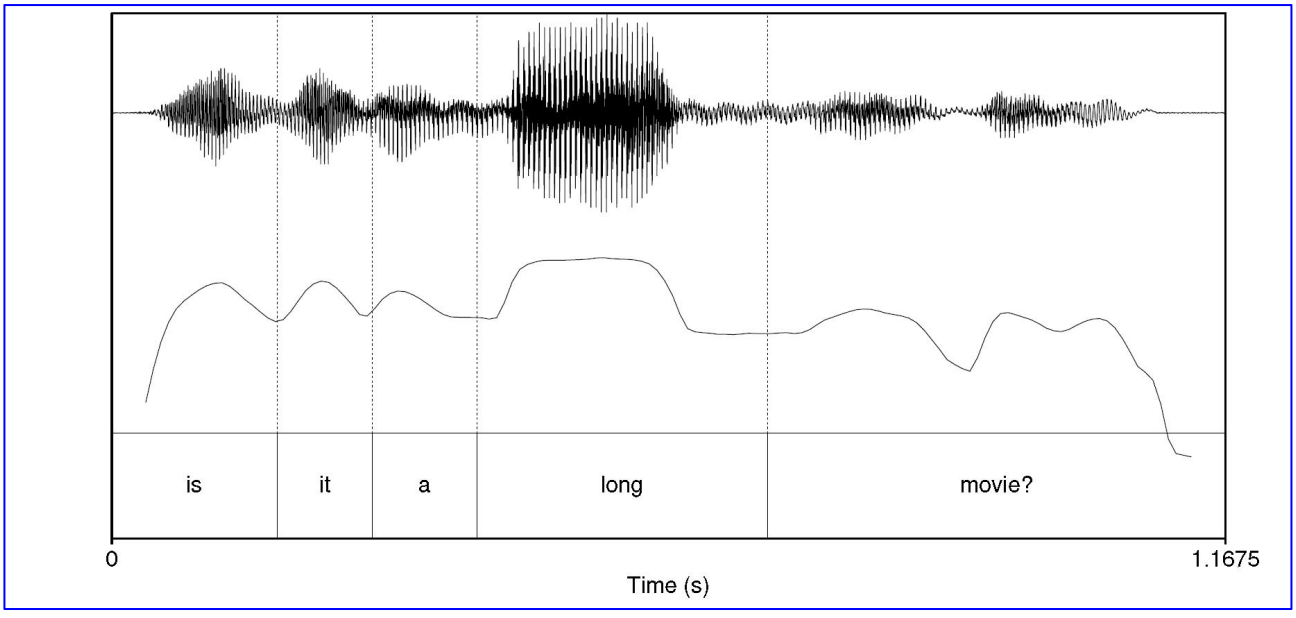


图 25-12：句子 Is it a long movie?的音强图

图注：注意图中每个元音的音强峰值，以及单词 long 的特别高的峰值。

音高(pitch)和**响度(loudness)**是两个重要的感知特性，与频率和音强有关。声音的音高是基本频率的心理感觉或感知相关性；通常，如果声音的基频更高，我们会认为它具有更高的音高。我们之所以说“通常”，是因为这种关系不是线性的，因为人类的听力对于不同的频率具有不同的敏锐度。粗略地说，人的音高感知在 100 Hz 至 1000 Hz 之间最为准确，在此范围内，音高与频率呈线性关系。人耳听觉不能准确地表示高于 1000 Hz 的频率，而高于该范围时，音高与频率成对数关系。对数表示意味着高频之间的差异被压缩，因此无法准确感知。音高感知尺度(scale)有多种心理声学模型。一种常见的模型是**梅尔(mel)**尺度(Stevens

等人 1937，Stevens 和 Volkmann1940)。mel 是音高的一个单位，定义为使在音高上等距的成对的声音被相等数量的 mels 分开。mel 频率 m 可以从原始声频计算如下：

$$m = 1127 \ln(1 + f/700) \tag{25.9}$$

我们将在第 26 章中看到，mel 尺度在语音识别中扮演着重要的角色。

声音的响度是一种强度的感知关联。因此，高振幅的声音会被认为声音更响，但这种关系也不是线性的。首先，正如我们在上面定义μ-law 压缩时所提到的，人类在低强度范围内具有更大的分辨率;耳朵对微小的力强度差异更敏感。第二，已经证明，在功率、频率和感知到的响度之间存在着复杂的关系;某些频率范围内的声音被认为比其他频率范围内的声音更响。

有各种算法可以自动提取 F0。稍微滥用一下术语，这些被称为**音高提取(pitch extraction)**算法。例如， 音高提取的自相关方法，就是在不同的偏移量处将信号与自身相关联。具有最高相关性的偏移量给出信号的周期。有各种公开可用的音高提取工具包;例如，Praat(软件包)提供了一个增强的自相关音高跟踪器(Boersma 和 Weenink, 2005)。

25.4.4. 从波形来解释音子

从波形的目视检查中可以学到很多东西。例如，元音很容易识别。回想一下元音是浊音； 元音的另一个特性是它们倾向于较长且相对较响(如我们在图 25.12 的音强图中所看到的)。时间长度直接在 x 轴上显示出来，响度与 y 轴上的振幅(平方)有关。我们在上一节图 25.9 中看到，浊音是通过振幅的有规则峰值来实现，每个主要峰值对应于声带的开口。图 25.13 显示了简短的句子 she just had a baby 的波形。我们已经用单词和音子标签来标签该波形。请注意，图 25.13 中的六个元音[iy]，[ax]，[ae]，[ax]，[ey]，[iy]中的每个都有规则的振幅峰值代表浊音。

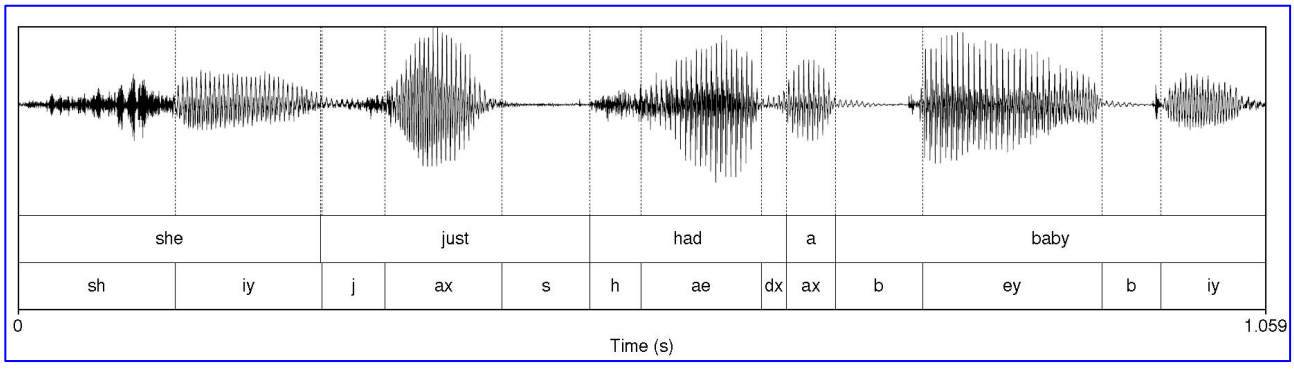


图 25-13：句子 She just had a baby 的波形图

图注：Switchboard 语料库中“She just had a baby”这句话的波形图(会话 4325)。说话的人是女性，1991 年 20 岁，大概就是录音录制的时间，她说的是美国南部中部地区的方言。

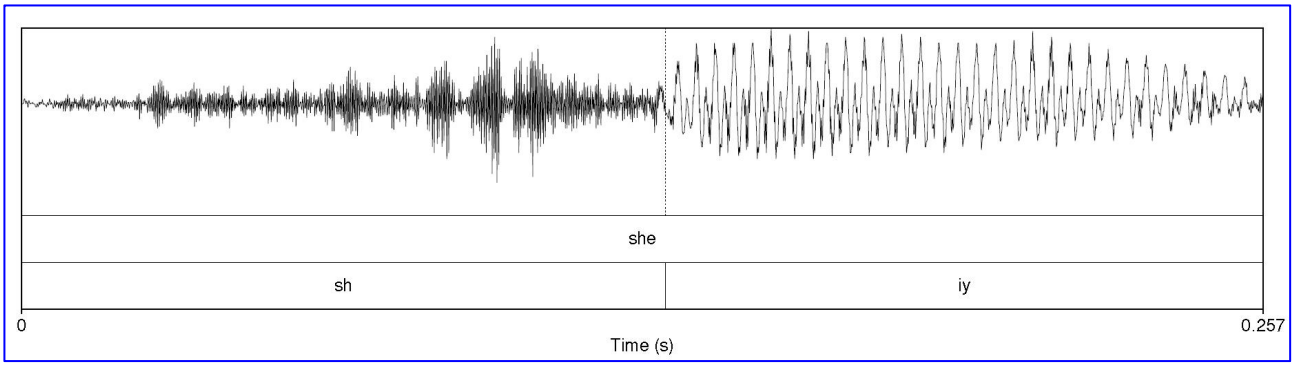


图 25-14：单词“she”的详细视图

图注：从图 25.13 的波形文件中提取的第一个单词“she”的详细视图。注意摩擦音[sh]的随机噪声和元音[iy]的有规则浊音之间的区别。

塞辅音包含成阻和紧随其后的除阻，我们经常可以看到一段静默或近乎静默的时期，随后是一个轻微的振幅爆破。我们可以在图 25.13 中看到这两个[b]在 baby 中的情况。

经常在波形中很容易辨认的另一个音子是摩擦音。回想一下，当狭窄的气流通道导致嘈杂的湍流空气时，就会产生摩擦音，尤其是像[sh]这样刺耳的摩擦音。产生的嘶嘶声具有嘈杂的不规则波形。在图 25.13 中可以看到这一点。在图 25.14 中甚至更清楚了，在这里我们只放大了她的第一个单词。

25.4.5. 声谱和频谱

虽然一些宽泛的语音特征(如强度、音高、浊音的出现、塞音的成阻、或摩擦音)可以直接从波形上来解释,大多数计算应用,如语音识别(以及人类听觉处理)是基于不同的声音表示频率的成分。傅里叶分析的精髓在于，每个复杂波都可以表示为不同频率的许多正弦波之和。考虑图 25.15 中的波形。这个波形(在 Praat 中)是由两个正弦波形(频率为 10 赫兹和 100 赫兹)相加而产生的。

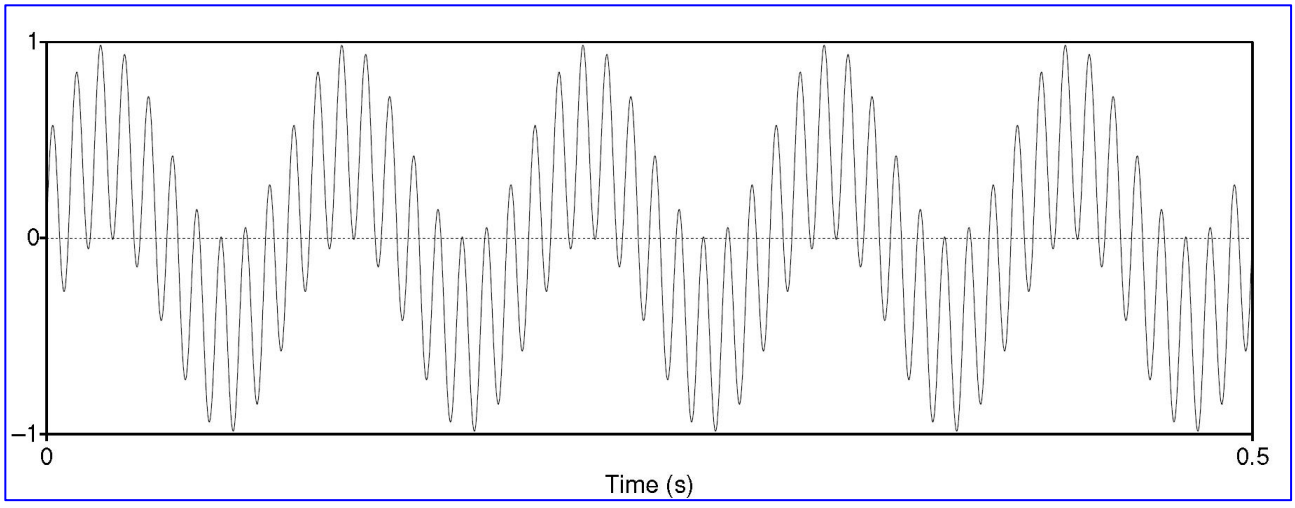


图 25-15：一个波形是两个正弦波之和

图注：图中一个频率为 10hz(注意半秒窗口内重复 5 次)，另一个频率为 100hz，振幅均为 1。

我们可以用**声谱(spectrum)**表示这两个成分的频率。信号的声谱表示其每个频率成分及其幅度。图 25.16 显示了图 25.15 的声谱。以 Hz 为单位的频率在 x 轴上，振幅在 y 轴上。注意图中的两个尖峰，一个尖峰为 10 Hz，另一个尖峰为 100 Hz。因此，声谱是原始波形的另一种表示，我们将声谱用作研究特定时间点声波频率成分的工具。

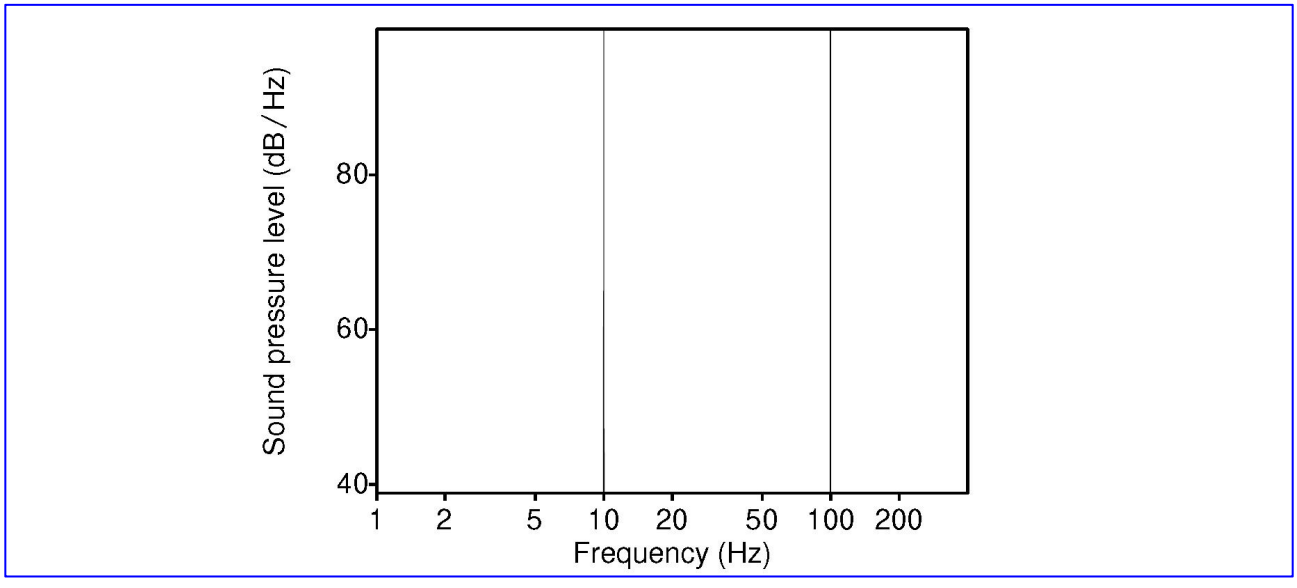


图 25-16：来自图 25.15 中波形的声谱

现在让我们看一下语音波形的频率成分。图 25.17 显示了单词 has 的元音[ae]的一部分波形，该波形是从图 25.13 所示的句子中切出的。请注意，图中有一个复杂的波重复大约十次。但是还有一个较小的重复波，它在每个较大的模式中都会重复四次(注意每个重复波中的四个小峰)。复杂波的频率约为 234 Hz(我们可以弄清楚这一点，因为它在.0427 秒内大约重复 10 次，而 10 个周期/0.0427 秒= 234 Hz)。

微波(smaller wave)的频率应该是大波频率的四倍，大约是 936hz。然后，如果你仔细观察，你会看到 936hz 波中的两个**小波(little wave)**。这种**最小波(tiniest wave)**的频率大约是 936hz 波的两倍，也就是 1872hz。图 25.18 显示了图 25.17 中波形平滑的声谱，通过离散傅里叶变换(DFT)计算得到。

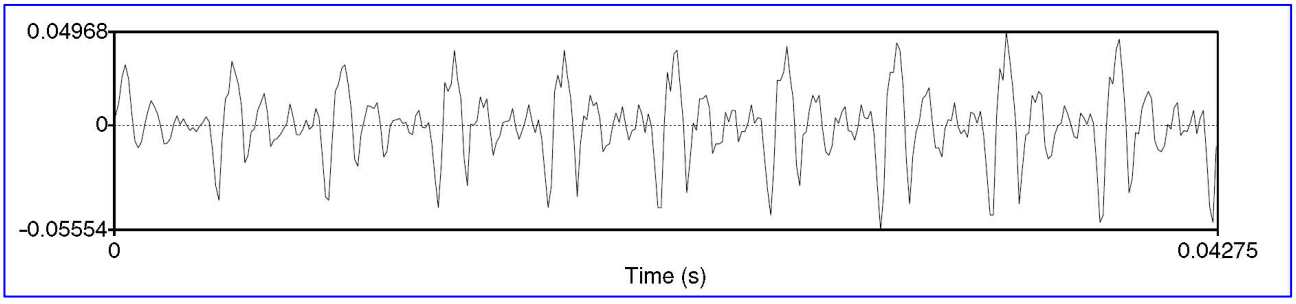


图 25-17：从图 25.13 中截取了 had 单词中元音[ae]部分的波形

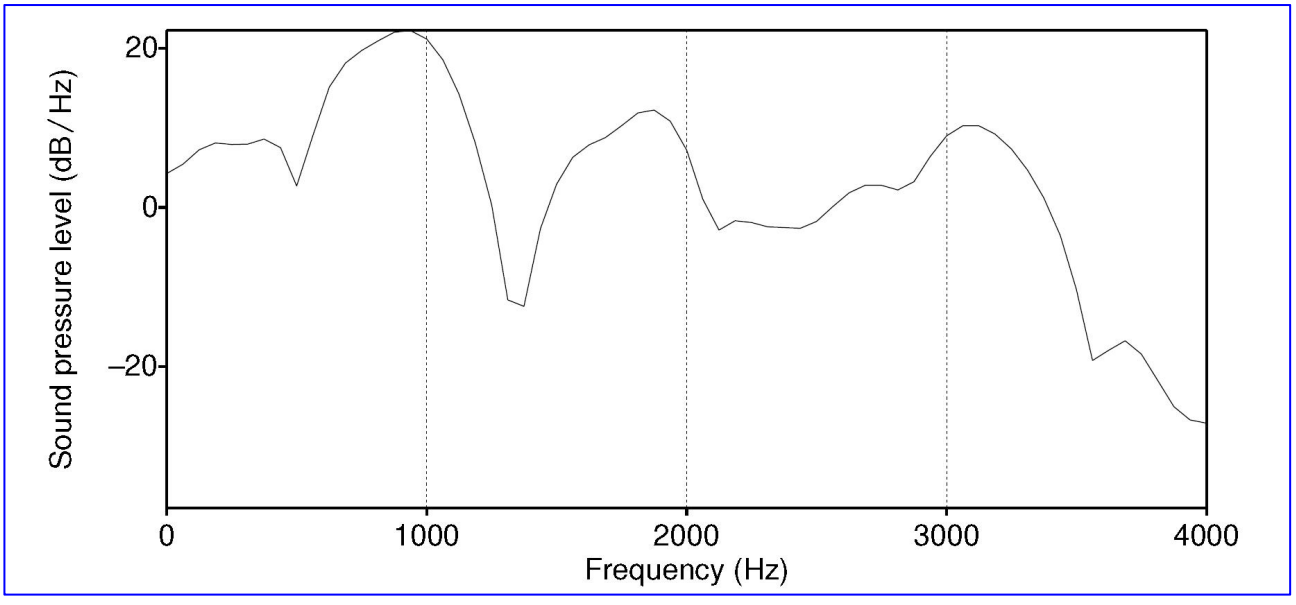


图 25-18：来自图 25.13 中单词 had 的元音[ae]的声谱

声谱的 x 轴表示频率，y 轴表示每个频率分量的大小的某种量纲(以分贝(dB)为单位，这是我们之前看到的幅度的对数量度)。因此，图 25.18 显示了大约 930 Hz，1860 Hz 和 3020 Hz 处的重要频率分量，以及许多其他较低幅度的频率分量。通过查看图 25.17 中的波形，这是我们在时域中注意到的前两个成分！

声谱为什么有用?事实证明，这些很容易在声谱中看到的谱峰是不同音子的特色;音子具有声谱特色的“签名”。就像化学元素在燃烧时会发出不同波长的光一样，它使我们可以通过观察光的光谱来检测星体中的元素，我们可以通过观察波形的声谱来检测不同音子的特色“签名”。声谱信息的这种使用对于人类和机器语音识别都是必不可少的。在人类听觉中，**耳蜗(cochlea)**或内耳的功能是计算输入波形的声谱。

让我们看一下不同元音的声谱。由于某些元音会随着时间变化，因此我们将使用另一种称为**频谱(spectrogram)**的图形。声谱显示某个时间点的波的频率分量，而频谱则是一种构想方式：组成波形的不同频率是如何随时间而变化。x 轴显示时间，就像波形的表示一样，但是 y 轴现在显示频率(以赫兹为单位)。频谱上一个点的暗度对应于频率分量的幅度。非常暗的点具有高振幅，亮的点具有低振幅。因此，频谱是三个维度(时间 x 频率 x 振幅)可视化的有用方法。

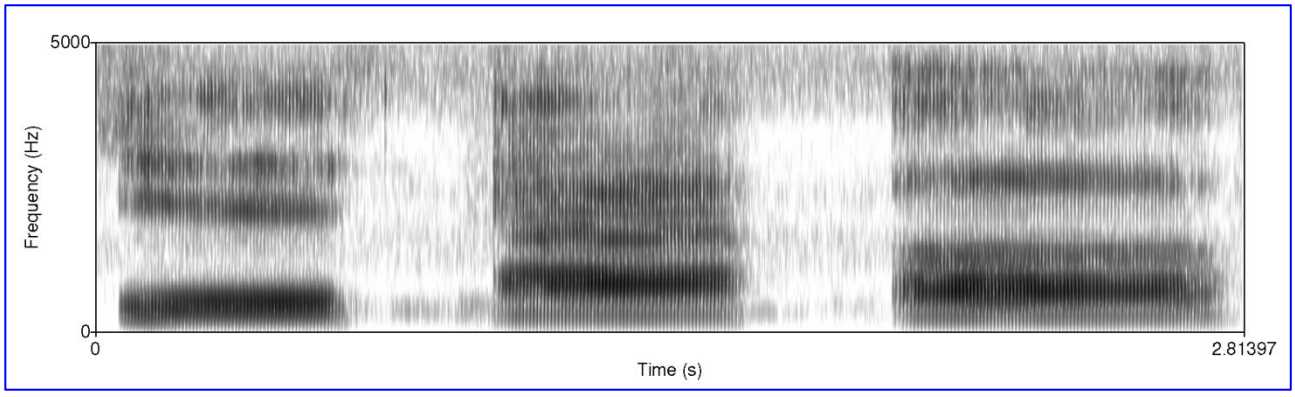


图 25-19：美国英语[ih]、[ae]、[uh]三个元音的频谱

图 25.19 显示了三个美式英语元音[ih]、[ae]和[ah]的频谱。注意，每个元音在不同的频带上都有一组暗条，每个元音的频带略有不同。每一个都代表了我们在图 25.17 中看到的同一种声谱峰。

每个暗条(或声谱峰)称为**共振峰(formant)**。正如我们下面所讨论的，共振峰是一种被声腔特别放大的频带。由于声腔在不同的位置产生不同的元音，它们会产生不同种类的放大或共鸣。我们来看看前两个共振峰，叫做 F1 和 F2。请注意，最底部的暗条 F1 对于三个元音的位置有所不同;它对于[ih]的位置低(集中在大约 470 Hz)，对于[ae]和[ah]的位置稍高(大约 800 Hz)。相比之下，从底部开始的第二个暗条 F2，对于[ih]的位置最高，对于[ae]的位置居中，对于[ah]的位置最低。

我们可以在**连续语流(running speech)**中看到相同的共振峰，尽管弱化和协同发音过程使它们有些难以看到。图 25.20 显示了“she just had a baby”的频谱，它的波形如图 25.13 所示。对于 just 的[ax]、had 的[ae]和 baby 的[ey]，F1 和 F2(还有 F3)都很清楚。

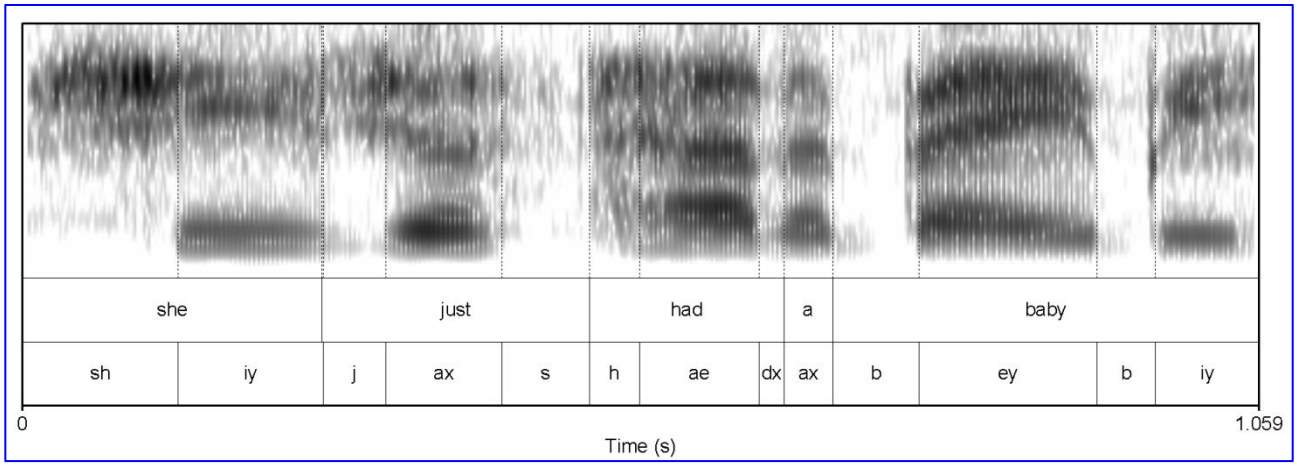


图 25-20：句子“she just had a baby”的频谱

图注：其波形如图 25.13 所示。我们可以把频谱看作声谱的集合(时间切片)，如图 25.18 所示按照端到端放置。

声谱表示可以为音子识别提供哪些特别的线索？ 首先，由于不同的元音在特征位置处有共振峰，因此频谱可以将元音彼此区分开。我们已经看到，样本波形中的[ae]具有 930 Hz，1860 Hz 和 3020 Hz 的共振峰。考虑图 25.13 中语段开始时的元音[iy]。该元音的频谱如图 25.21 所示。[iy]的第一共振峰为 540 Hz，远低于[ae]的第一共振峰，第二共振峰(2581 Hz)远高于[ae]的第二共振峰。如果仔细看，大约 0.5 秒处，这些共振峰就会在图 25.20 中显示为暗条。

前两个共振峰(称为 F1 和 F2)的位置在决定元音身份方面起着很大的作用，尽管共振峰仍然因不同的说话者而不同。较高的共振峰更多的是由说话者声腔的普遍特征而不是单个的元音引起的。共振峰也可用于鉴别鼻音音子[n]、[m]和[ng]以及边音(liquids)音子[l]和[r]。

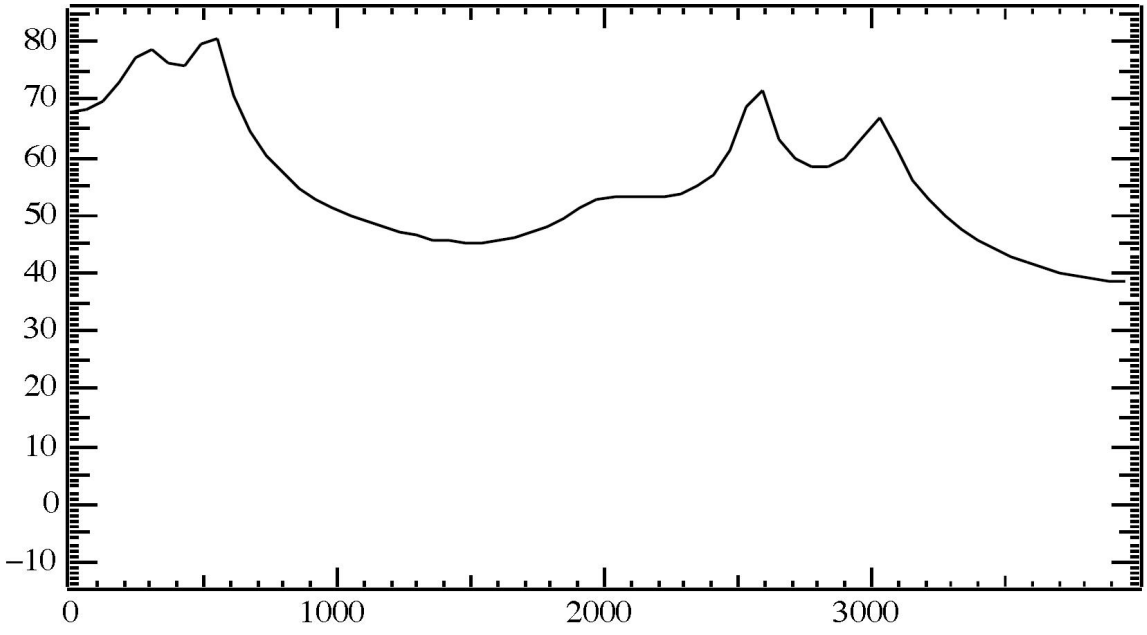


图 25-21：元音[iy]的平滑(LPC)频谱

图注：句子 She just had a baby 开始处的元音[iy]的平滑(LPC)频谱。注意，第一共振峰(540 Hz)远低于图 25.18 所示的[ae]的第一共振峰，第二共振峰(2581 Hz)远高于[ae]的第二共振峰。

25.4.6. 声源滤波器模型

为何不同的元音有不同的声谱特征？前述，共振峰是由口腔共鸣所致。**声源-滤波器模型(source-filter model)**是通过建模来解释声音的声学方式：声门(声源)产生的脉冲如何由声腔(滤波器)成型。

让我们看它是如何工作的。每当我们有一个波，如由声门脉冲引起的空气振动，这个波也就会有**谐波(harmonics)**。谐波是另一种波，它的频率是基波的数倍。例如，一个 115 赫兹的声门褶振动导致 230hz、345hz、460hz 的谐波(其他波)，等等。一般来说，这些波中的每一个都会更弱，也就是说，它们的振幅会比基频处的波小得多。

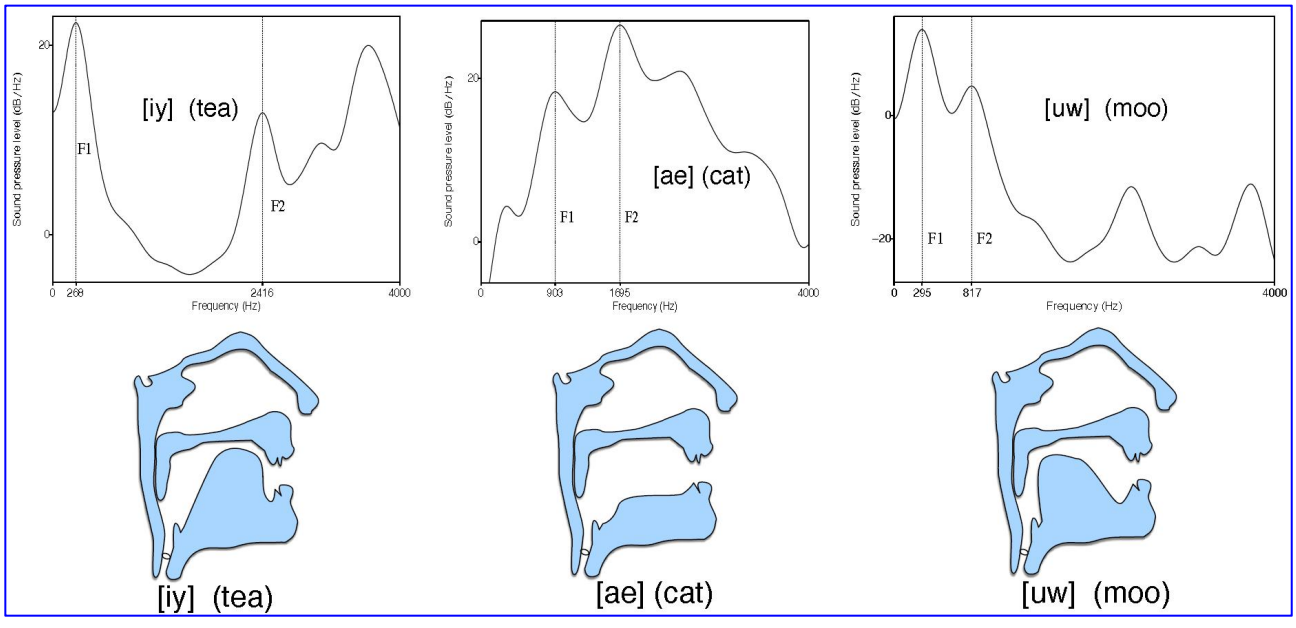


图 25-22：作为过滤器的声腔位置的可视化

图注：图中显示了三个英语元音的舌头位置以及生成的平滑声谱，包括 F1 和 F2。

然而，事实证明，声腔起到了一种过滤器或放大器的作用;事实上，任何空腔，比如管子，都会导致某些频率的波被放大，而其他频率的波被衰减。这种放大过程是由空腔的形状引起的;一个特定的形状会引起特定频率的声音产生共鸣，从而被放大。因此，通过改变空腔的形状，我们可以放大不同的频率。

当我们发出特定的元音时，我们实际上是在通过把舌头和其他发音器官放在特定的位置来改变声腔的形状。结果是不同的元音导致不同的谐波被放大。所以相同基频的波通过不同的声腔位置会导致不同的谐波被放大。

我们可以通过观察声腔形状和相应声谱之间的关系来看到放大的结果。图 25.22 显示了三个元音的声腔位置和一个典型的声谱结果。共振峰是声腔在声谱中放大特定谐波频率的地方。

25.5. 语音资源

各种各样的语音资源可以用于计算工作。在线**发音词典(pronunciation dictionaries)**为每个单词提供语音转写。LDC 发布埃及阿拉伯语、荷兰语、英语、德语、日语、韩语、汉语和西班牙语的发音词汇。对于英语，《CELEX 词典》(Baayen 等人 1995)有 160595 个单词形式的发音，包括音节切分、重音、形态和词类信息。开源的 CMU 发音字典(CMU, 1993)有大约 134,000 个单词的发音。而细粒度的 110,000 个单词的 UNISYN 字典(Fitt, 2002)，为研究目的免费提供，给出了音节切分、重音、以及数十种英语方言的发音。

另一个有用的资源是语音标注语料库(phonetically annotated corpus)，其中一组波形用相应的音子字符串由手工标签的。TIMIT 语料库(NIST, 1990)，最初是德州仪器(TI)、麻省理工学院和 SRI 的联合项目，是一个包含 6300 个阅读句子的语料库，由 630 个说话者每人说 10 个句子。这 6300 个句子是从 2342 个句子中选出来的，其中一些被选为具有特定方言颤音的句子，另一些则是为了最大限度地扩大语音双音子范围。语料库中的每个句子都被语音手工标签，音子序列自动与句子的波形文件对齐，然后对已经自动标签过的音子边界进行手动校正(Seneff 和 Zue,1988)。结果是**时间对齐转写(time-aligned transcription)**:它是一个转写，其中每个音子与波形中的开始和结束时间相关联，如图 25.23 中的示例所示。

she	had	your	dark	suit	in	greasy	wash	water	all	year
sh iy	hv ae dcl	jh axr	dcl d aa r kcl	s ux q	en	gcl g r iy s ix	w aa sh	q w aa dx axr q	aa l	y ix axr

图 25-23：来自 TIMIT 语料的语音转写例子

图注：利用 ARPAbet 特有的特征进行严式(narrow)转写，如单词 had 中的[d]的上腭音化，单词 dark 中没有除阻的最后一个塞音，单词 suit 中的最后一个[t]喉音化为[q]，单词 water 中[t]读作颤音。TIMIT 语料库还包括时间对齐(未显示)。

Switchboard 转写项目的语音标注语料库包括从 Switchboard 语料库中提取的 3.5 小时的句子(Greenberg 等人 1996)，以及在音节水平上按照时间对齐的转写。图 25.24 显示了一个示例。

0.470	0.640	0.720	0.900	0.953	1.279	1.410	1.630
dh er	k aa	n ax	v ih m	b ix	t w iy n	r ay	n aw

图 25-24：Switchboard 中的短语的语音转写

图注：Switchboard 语料库中的短语 they're kind of in between right now 的语音转写。注意 they're 和 of 中的元音弱化，kind 和 right 中音节尾的删除，以及再音节化现象(of 由于与 in 的音节头相连接而变为[v])。时间是以秒为单位，从句子的开始到每个音节的开头。

Buckeye 语料库(Pitt 等人 2007，2005)是自发的美国语音的语音转写语料库，包含来自 40 个谈话者的 300,000 个单词。语音转写的语料库也可用于其他语言，包括德语 Kiel 语料库、由中国社会科学院(Li 等人 2000)转写的普通话语料库。

除了像词典和语料库等资源外，还有许多有用的语音软件工具。本书中的许多图形都是由 Praat 软件包生成的(Boersma 和 Weenink, 2005)，它包括音高、声谱和共振峰分析、以及一种脚本语言。

25.6. 总结

本章介绍了语音学和计算语音学的许多重要概念。

•我们可以用称为**音子(phone)**的单位来表示单词的发音。代表音子的标准系统是国际音标，即 **IPA**。最常见的英语转写的计算系统是 **ARPAbet**，它方便地使用 ASCII 符号来表示英语语音。

•音子可以通过发音器官如何发声来描述;辅音的定义是根据其**发音部位(place)**、**发音方式(manner)**、以及声带是否**振动(voicing)**来确定;元音的定义是根据其舌位的**高低(height)**、**前后(backness)**、以及嘴唇是否**圆形(roundness)**来确定。

•语音也可以用声学来描述。声波可以用**频率(frequency)**、**振幅(amplitude)**、或它们的感知上相应的**音高(pitch)**和**响度(loudness)**来描述。

•声音的**声谱(spectrum)**描述其不同的频率成分。虽然一些语音属性可以从波形中识别出来，但人类和机器两者都依赖声谱分析来检测音子。

•**频谱(spectrogram)**是一种随时间变化的声谱。元音是由称为**共振峰(formant)**的特征**谐波(harmonics)**来描述的。

25.7. 文献和历史说明

语音发音的主要见解可以追溯到公元前 800-150 年印度的语言学家。他们发明了发音部位置和发音方式的概念，弄清了发声的声门机制，并理解了语音同化的概念。直到 2000 年后的 19 世纪末，欧洲科学才赶上印度的语音学家。希腊人确实有一些基本的语音知识。举例来说，到柏拉图(Plato)的 Theaetetus 和 Cratylus 时代，他们将元音与辅音区分开来，将停顿辅音与连续辅音区分开来。Stoics(斯多葛)提出了音节的概念，并且意识到对可能单词的音位限制。十二世纪一位不知名的冰岛学者利用**音素(phoneme)**的概念，并为冰岛人提出了一种音素书写系统，其中包括音长和鼻音的变音符号。但是直到 1818 年，他的著作才得以出版，直到那时，在 Scandinavia 半岛以外的地区还是鲜为人知(Robins，1967)。通常认为现代语音学是从 Sweet 开始的，Sweet 在他的《语音学手册》(Handbook of Phonetics，1877 年)中提出了实质上的音素。他还设计了用于转写的字母，并区分了**宽式(broad)**和**严式(narrow)**的转写，提出了许多最终被纳入 IPA 的想法。斯威特(Sweet)被认为是他那个时代最优秀的语音专家。他制作了第一个用于语音目的的语言科学记录，并发展了发音描述的最新技术水平。他也名声不好很难与人相处，亨利 • 希金斯(Henry Higgins)很好地抓住了这一特质，乔治 • 萧伯纳(George Bernard Shaw)模仿他的舞台角色。音素是由波兰学者鲍杜因 • 德考特尼(Baudouin de Courtenay)首次命名的，他于 1894 年发表了他的理论。

语音入门教材包括 Ladefoged(1993)和 Clark and Yallop(1995)。威尔斯(Wells 1982)是有关英语方言的确定性三卷本资料。

1950 年代末或 1960 年代初，许多关于声学语音学的经典见解得到了发展。仅有的几个亮点包括诸如声音频谱之类的技术(Koenig 等人 1946)，理论洞察力诸如声源滤波器理论的锻造以及在发声与声学之间的映射中的其他问题(Fant 1960，Stevens 等人 1953，Stevens 和 House 1955, Heinz 和 Stevens 1961, Stevens 和 House 1961)，元音共振峰的 F1xF2 空间(Peterson 和 Barney 1952)，对重音的语音本质的理解，对音长和音强的线索的使用(Fry，1955)，对音子感知问题的基本理解(Miller 和 Nicely 1955，Liberman 等人 1952)。Lehiste(1967)是有关声学语音学的经典论文的集合。Gunnar Fant 的许多开创性论文已被 Fant(2004)收集。

优秀的声学语音学教科书包括 Johnson(2003)和 Ladefoged(1996)。Coleman(2005)从语言学的角度介绍了声学和语音的计算处理。Stevens(1998)提出了一个有影响力的语音产生理论。有许多的软件包用于声学语音分析。可能使用最广泛的是 Praat(Boersma 和 Weenink，2005)。

25.8. 练习

25.1 Find the mistakes in the ARPAbet transcriptions of the following words:

a. “three” [dh r i]

d. “study” [s t uh d i]

g. “slight” [s l iy t]

b. “sing” [s ih n g]

e. “though” [th ow]

c. “eyes” [ay s]

f. “planning” [p pl aa n ih ng]

25.2 Ira Gershwin’s lyric for *Let’s Call the Whole Thing Off* talks about two pro nunciations (each) of the words “tomato”, “potato”, and “either”. Transcribe into the ARPAbet both pronunciations of each of these three words.

25.3 Transcribe the following words in the ARPAbet:

1. dark
2. suit
3. greasy
4. wash
5. water

25.4 Take a wavefile of your choice. Some examples are on the textbook website. Download the Praat software, and use it to transcribe the wavefiles at the word level and into ARPAbet phones, using Praat to help you play pieces of each wavefile and to look at the wavefile and the spectrogram.

25.5 Record yourself saying five of the English vowels: [aa], [eh], [ae], [iy], [uw]. Find F1 and F2 for each of your vowels.

26. 自动语音识别和文本语音转换

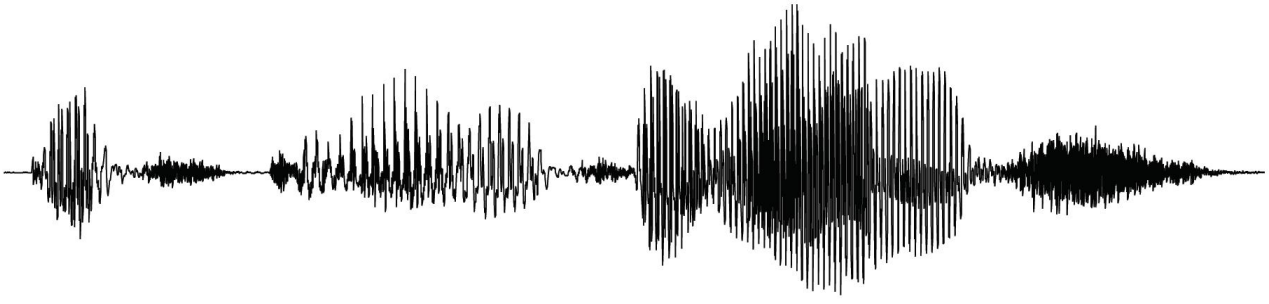
I KNOW not whether
I see your meaning: if I do, it lies
Upon the wordy wavelets of your voice,
Dim as an evening shadow in a brook,

Thomas Lovell Beddoes, 1851



理解口语，或至少将单词(words)转写成书面文字(writing)，是计算机语言处理的最早目标之一。实际上，语音处理要比计算机早几十年！ 第一台识别语音的机器是 1920 年代的玩具。右侧所示的“Radio Rex”是一个赛璐珞狗，当弹簧以 500 Hz 的声能释放时，狗会(通过弹簧)移动。由于 500 Hz 大约是“Rex”中元音[eh]的第一个共振峰，因此 Rex 在被召唤时，它似乎来了(David 和 Selfridge 1962)。

在现代，我们期望更多的自动化系统。**自动语音识别(ASR)**的任务是映射任何这样的波形：

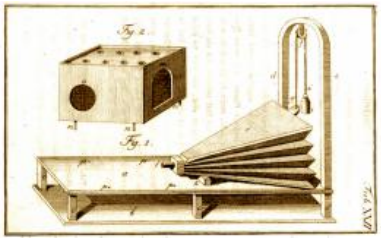


到合适的单词字符串:

It’s time for lunch!

在任何环境下，任何说话者的自动语音转写仍远未解决，但 ASR 技术已经成熟到可以用于许多实际任务的程度。语音是用于与智能家用电器、个人助理或手机进行通讯的自然界面，在诸如呼叫路由(“请记账”)之类的电话应用程序中或在复杂的对话应用程序(“我想更改我的航班的回程日期”)。ASR 对于常规转写也很有用，例如用于自动生成音频或视频文本的字幕(转录电影或视频或现场讨论)。转写在法律等领域中很重要，在该领域中，听写起着重要作用。最后，ASR 作为增强性交流的重要组成部分(计算机与某些残疾的人之间的互动会导致打字或听觉方面的困难或无能)。众所周知，失明的弥尔顿向他的女儿们口述了《失乐园》，亨利·詹姆斯在反复的重音损伤后口述了他后来的小说。

从文本到语音转换，这个相反问题的情况怎么样呢？这是历史更长的问题。1769 年在维也纳，Wolfgang von Kempelen 为 Maria Theresa 女皇建造了著名的 Mechanical Turk(土耳其机器人)，这是一种下棋自动机，由装满齿轮的木箱组成，其后坐着一个机器人模特，他用机械臂移动棋子来下棋。土耳其机器人在欧洲和美洲巡回演出数十年，击败了拿破仑·波拿巴(Napoleon Bonaparte)，甚至对弈查尔斯·巴贝奇(Charles Babbage)。可惜，它实际上只是一个骗局，由藏在盒子里的人类棋手操控。不然，土耳其机器可能是人工智能早期的成功之一。

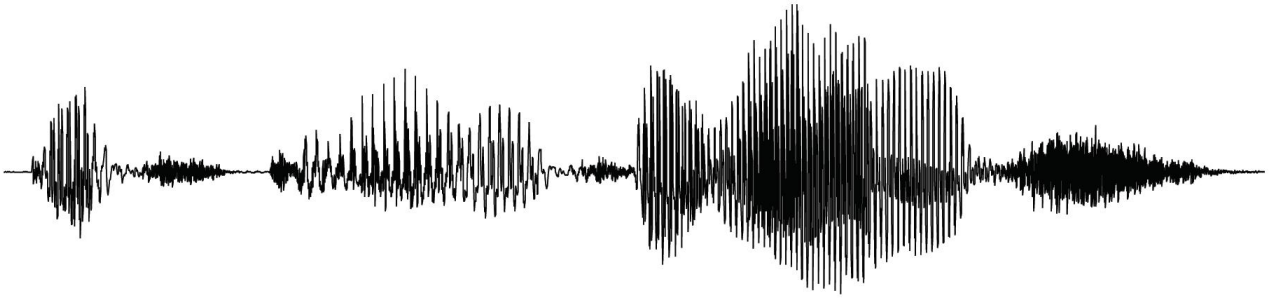


鲜为人知的是，冯·肯佩伦(von Kempelen)是一位绝顶多产的发明家，他还在 1769 年至 1790 年之间建造了绝对不是骗局的产品：第一台全句子语音合成器，部分显示在右侧。他的装置包括一个用来模拟肺部的风箱，一个橡胶喉舌和一个鼻孔，一个用来模拟声带的簧片，用于摩擦音的各种哨子，以及一个用来为爆破音提供喷射气流的小型辅助风箱。通过双手移动控制杆以打开和关闭孔口，并调节柔性皮革的“声腔”，操作员可以产生不同的辅音和元音。

两个多世纪后，我们不再用木头和皮革制造合成器，也不再需要人工操作。现代**语音合成(speech synthesis)**的任务，也被称为**文本语音转换(text-to-speech，TTS)**，恰恰和 ASR 相反。映射文本：

It's time for lunch!

到一种声音波形：



现代语音合成具有广泛的应用。TTS 用于引导与人进行对话的会话代理中，在为盲人或游戏中大声朗读的设备中发挥作用，并且可以用于为神经系统疾病的患者说话，例如已故的天体物理学家史蒂芬·霍金(Steven Hawking)，在他由于 ALS 而失去了声音使用能力后，通过操纵 TTS 系统进行讲话。

在接下来的部分中，我们将展示如何对编码器/解码器进行 ASR，介绍 CTC 损失函数，标准的单词错误率评估指标，并描述如何提取声学特征。然后，我们将了解如何使用几乎相同的算法反向建模 TTS，并以其他语音任务的简要介绍作为结束。

26.1. 自动语音识别任务

在描述 ASR 算法之前，让我们先讨论一下任务本身是如何变化的。

变化的第一个维度是词汇量。有些 ASR 任务可以用极高的准确性解决，比如词汇量只有 2 个单词(是与否)或词汇量只有 11 个单词的任务诸如**数字识别(digit recognition)**(识别数字序列，包括 0 到 9 加上 oh)。像转录视频或人类会话这样的开放式任务，词汇量高达 6 万个单词，难度就大得多。

变化的第二个维度是说话者在跟谁说话。人类与机器对话(口述或与对话系统对话)比人类与人类对话更容易识别。人们正在大声朗读的**朗读语音(read speech)**，例如在有声读物中，也相对容易识别。识别两个人在**会话语音(conversational speech)**中互相交谈的语音，例如转录商务会议，是最困难的。看来，当人们与机器交谈或在没有听众在场的情况下阅读时，他们会大大简化他们的演讲，使谈话变得更加缓慢和清晰。

变化的第三个维度是信道和噪声。如果语音是在带头戴式麦克风的安静房间中录制的，则比在嘈杂的城市街道上的远距离麦克风或开着窗户的汽车中录制的语音更容易识别。

变化的最终维度是口音特征或说话者类别特征。如果说话者说的是与系统训练相同的方言或多种语言，则语音更容易识别。如果仅对使用标准方言的人或仅对成人方言的人进行培训，则很难识别区域或民族方言的语音，或者儿童的语音。

许多公开的语料库带有人工创建的转录，用于创建 ASR 测试和训练集以探索这种变化；我们在这里提到其中的一些，因为您会在文献中遇到它们。**LibriSpeech** 是一个大型的开源朗读语音 16 kHz 数据集，包含来自 LibriVox 项目的 1000 多个小时的有声读物，其转录在句子级别对齐(Panayotov 等，2015)。它分为较容易的部分("干净")和较难的部分("其他")，干净的部分具有较高的录音质量，并且其口音接近于美国英语。这是通过对所有音频运行语音识别器(接受《华尔街日报》的语音阅读训练)来完成的，基于黄金转录计算每个发言人的 WER，将说话者大致分成两半：较低 WER 的录音称为"干净"的说话者，较高 WER 的录音称为"其他"的说话者。

1990 年代初期，收集了陌生人之间经过提示的电话会话的**电话总机(Switchboard)**语料库。它包含 2430 个会话，每个会话平均 6 分钟，总共 240 小时的 8 kHz 语音和大约 300 万个单词(Godfrey 等人，1992)。Switchboard 具有大量的手工做的语言辅助标签的独特优势，包括语法分析、行为标记、语音和韵律标签

以及话语和信息结构。**致电回家(CALLHOME)**语料库收集于 1990 年代后期，由英语为母语的人(通常是亲密的朋友或家人)之间的 120 条无脚本的 30 分钟电话会话组成(Canavan 等人，1997)。

圣巴巴拉美语口语语料库(Du Bois 等人，2005)是美国各地日常口语交流的大型语料库，主要是面对面的交谈，但也包括市政厅会议、食物准备、在职谈话和课堂讲座。通过删除个人姓名和其他识别信息(在转录中用假名代替，在音频中掩码代替)，语料库被匿名化。

CORAAL 是对非裔美国人进行的超过 150 次社会语言学采访的集合，目的是研究非裔美国人语言(AAL)，即非裔美国人社区使用的许多语言变体(Kendall 和 Farrington, 2020)。访谈是匿名的，与转录在语段水平一致。**编钟(CHiME)**挑战是一系列与语料库共同处理 ASR 稳健性的困难任务。例如，编钟任务就是在真实的家庭环境中(特别是在晚宴上)对话的语音识别。语料库包含了真实家庭中 20 个不同的晚餐聚会的录音，每个聚会有 4 个参与者，在 3 个地点(厨房、用餐区、客厅)，都是用远处的房间麦克风和随身佩戴的麦克风录制的。**香港科大(HKUST)**普通话电话语料库里有 1206 段 10 分钟的电话对话，这些对话是在中国各地的普通话使用者之间进行的，包括对话的转录，这些对话是在朋友或陌生人之间进行的(Liu 等人 2006)。**希尔贝壳-1(AISHELL-1)**语料库包含 170 小时的汉语普通话阅读演讲，这些阅读演讲来自不同的领域，主要由中国北方的不同说话者朗读(Bu 等人，2017)。

图 26.1 显示了不正确单词的大致百分比(单词错误率，WER，在第 26.5 节定义)。请注意，阅读语音(如 librisspeech 有声图书语料库)的错误率约为 2%;这是一个已解决的任务，尽管这些数字来自需要大量计算资源的系统。相比之下，转录人类之间对话的错误率要高得多;Switchboard 和 CALLHOME 语料库 5.8%至 11%。对于不同种类的说话者，比如非裔美国人的口语，错误率更高；对于困难的会话任务，比如 4 人晚宴演讲的转录，错误率高达 81.3%。字符错误率(CER)在阅读普通话演讲中也比在自然对话中低得多。

English Tasks	WER%
LibriSpeech audiobooks 960hour clean	1.4
LibriSpeech audiobooks 960hour other	2.6
Switchboard telephone conversations between strangers	5.8
CALLHOME telephone conversations between family	11.0
Sociolinguistic interviews, CORAAL (AAL)	27.0
CHiMe5 dinner parties with body-worn microphones	47.9
CHiMe5 dinner parties with distant microphones	81.3
Chinese (Mandarin) Tasks	**CER%**
AISHELL-1 Mandarin read speech corpus	6.7
HKUST Mandarin Chinese telephone conversations	23.5

图 26-1：单词错误率(WER)和字符错误率(CER)

图注：在 2020 年左右，ASR 在各种美式英语识别任务中报告的粗略单词错误率(WER=单词误识别的%)，以及两个中文识别任务中的字符错误率(CER)。

26.2. ASR 的特征提取：对数 Mel 声谱

ASR 的第一步是将输入波形转换为一系列声学**特征向量(feature vectors)**，每个向量在信号的较小时间窗内表示信息。让我们看看如何将原始波形文件转换为最常用的特征，即对数 Mel(梅尔)声谱向量的序列。建议使用语音信号处理课程，以获取更多详细信息。

26.2.1. 采样和量化

回忆一下第 25.4.2 节，第一步是将模拟表示形式(首先是气压，然后是麦克风中的模拟电信号)转换为数字信号。这种模数转换有两个步骤：采样和量化。通过在特定时间测量信号的幅度来对信号进行**采样(sampling)**：**采样率(sampling rate)**是每秒采样的数量。为了准确地测量波形，每个循环中至少必须有两个样本：一个测量波形正的部分，另一个测量负的部分。每个周期多于两个采样可提高幅度的准确度，但少于两个采样将导致波性的频率完全丢失。因此，可以测量的最大频率波形是其频率为采样率一半的波形(因为每个周期需要两次采样)。给定采样率的最大频率称为**奈奎斯特频率(Nyquist frequency)**。人类语音中的大多数信息的频率都低于 10,000 Hz，因此，要获得完全的准确度，就需要 20,000 Hz 的采样率。

但是电话语音是由交换网络过滤的，电话只能发送小于 4,000 Hz 的频率。因此，对于**电话带宽(telephone-bandwidth)**语音来说，8,000 Hz 的采样率就足够，对于麦克风语音而言，16,000 Hz 的采样率就足够了。

尽管使用更高的采样率会产生更高的 ASR 准确度，但我们无法结合使用不同的采样率来训练和测试 ASR 系统。因此，如果我们要在像 Switchboard 这样的电话语料库(8 KHz 采样)上进行测试，则必须将训练语料库**下采样(downsample)**到 8 KHz。同样，如果我们正在训练多个语料库，其中之一包括电话语音，我们会将所有宽带语料库下采样到 8Khz。

振幅测量值存储为整数，8 位(-128-127 的值)或 16 位(-32768-32767 的值)。这个将实数表示为整数的过程称为**量化(quantization)**;所有比最小粒度(量程大小)更接近的值都以相同的方式表示。我们将数字化量化波形中时间指数为 n 的每个样本称为 x[n]。

26.2.2. 加窗

从波形的数字化量化表示中，我们需要从表征特定音素一部分的小语音窗口中提取声谱特征。在这个小窗口内，我们可以粗略地认为信号是**平稳的(stationary)**(也就是说，其统计特性在该区域内是恒定的)。(相比之下，语音通常是**非平稳(non-stationary)**信号，这意味着其统计特性在一段时间内不是恒定的)。我们通过使用一个在区域内不为零，在其他位置为零的窗口，在语音信号上运行该窗口并将其乘以输入波形来产生此窗口化波形，来提取语音的这种大致平稳的部分。

从每个窗口中提取的语音称为**帧(frame)**。窗口的特征有三个参数:窗口大小或帧大小(其宽度以毫秒为单位)，连续窗口之间的帧**步长(stride)**(也称为移位或偏移)，以及窗口的形状。为了提取信号，我们将信号在 n, s[n]时刻的值乘以窗口在 n, w[n]时刻的值:

$$y[n] = w[n]s[n] \tag{26.1}$$

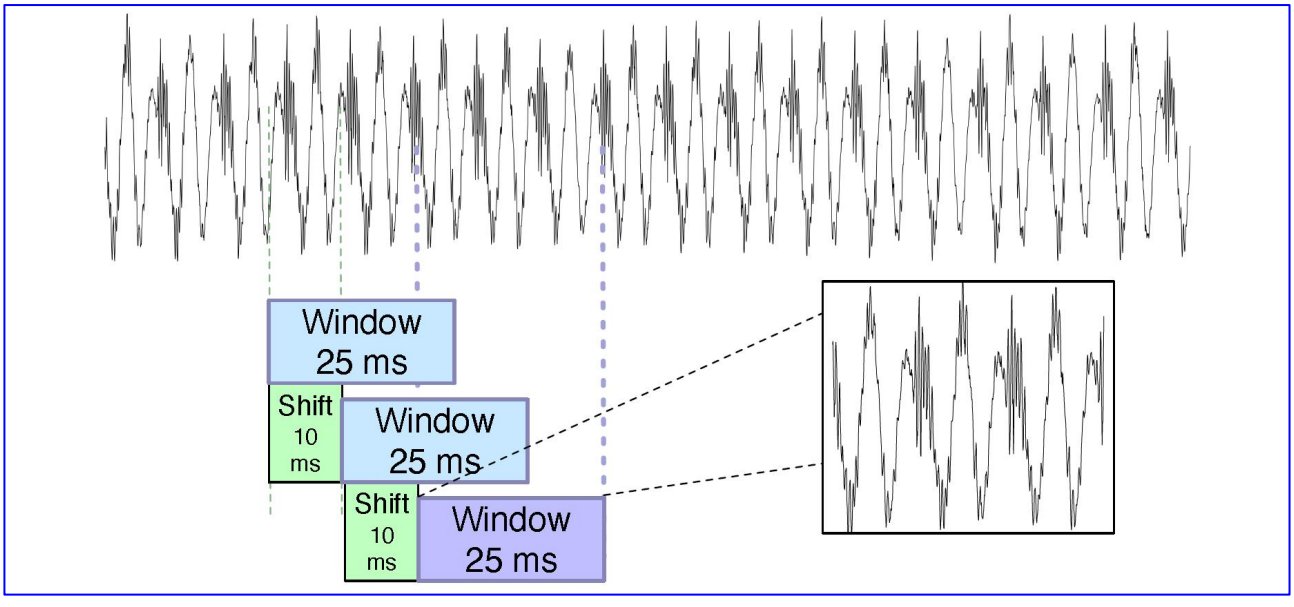


图 26-2：加窗：步长为 10ms 帧长为 25 ms 的矩形窗

图 26.2 所示窗口形状为矩形(rectangular);你可以看到提取的加窗信号看起来就像原始信号。然而，矩形窗口在其边界处突然切断信号，这在我们做傅立叶分析时造成了问题。因此，在创建声学特征时，我们更常用的是**汉明(Hammin)**窗口，它在窗口边界处将信号的值缩小到 0，避免了不连续。图 26.3 显示了这两种情况，方程如下(假设一个 L 帧长的窗口):

$$rectangular \quad w[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \tag{26.2}$$

$$Hamming \quad w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L}\right) & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \tag{26.3}$$

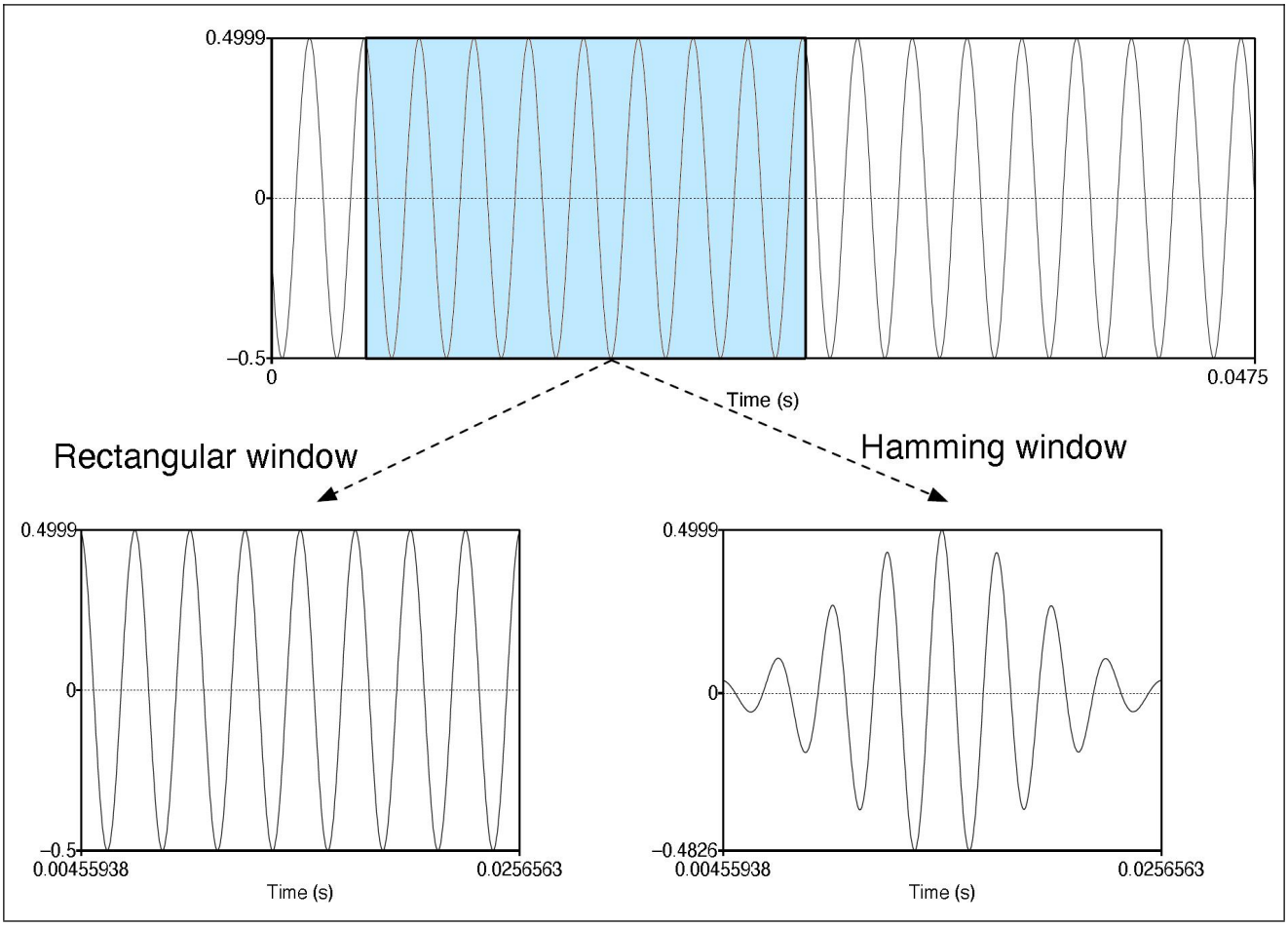


图 26-3：加窗：正弦波矩形窗或汉明窗

26.2.3. 离散傅里叶变换

下一步是为我们的加窗信号提取声谱信息;我们需要知道信号在不同频段包含多少强度。对离散时间(采样)信号的离散频带提取声谱信息的工具是**离散傅里叶变换(Discrete Fourier Transform，DFT)**。DFT 的输入是一个加窗信号 x[n]…x[m]，每 N 个离散频带的输出是一个复数 x[k]，表示原始信号中该频率分量的幅值和相位。如果我们把幅值和频率的关系画出来，我们就可以把我们在 25 章中介绍过的声谱形象化。例如，图 26.4 显示了信号的 25ms 汉明窗部分及其由 DFT 计算出的声谱(带有一些额外的平滑)。

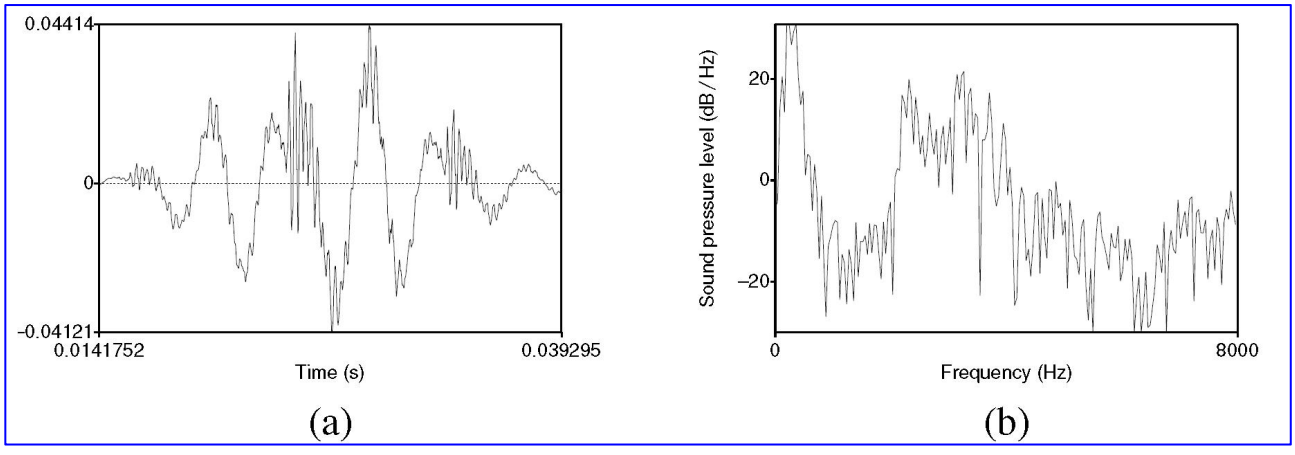


图 26-4：信号的汉明窗及其 DFT 计算的声谱

图注：(a)元音[iy]信号的 25 ms 汉明窗和(b)由 DFT 计算的声谱。

我们在此不介绍 DFT 的数学细节，只是要注意傅里叶分析依赖于欧拉公式(Euler's formula)，以 j 为**虚单位(imaginary unit)**：

$$e^{j\theta} = cos\theta + jsin\theta \tag{26.4}$$

作为对那些已经学习过信号处理的学生的一个简短提醒，DFT 的定义如下:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \tag{26.5}$$

一种常用的计算 DFT 的算法是**快速傅里叶变换(Fast Fourier Transform，FFT)**。DFT 的这种实现是非常有效的，但只对 N 的 2 的幂值有效。

26.2.4. Mel 滤波器组和对数

FFT 的结果告诉我们每个频带的**能量(energy)**。然而，人类的听觉在所有频段都不是同样敏感的;在较高的频率下，它的灵敏度较低。这种对低频的偏好有助于人类的识别，因为低频信息(如共振峰)对区分数值或鼻音至关重要，而高频信息(如塞音爆发或摩擦音)对成功识别则不那么重要。对这一人类感知特性建模可以以同样的方式提高语音识别性能。

我们通过收集能量来实现这种直觉，不是在每个频带上都相等，而是根据梅尔尺度，一种听觉频率尺度(第 25 章)。**梅尔(mel)** (Stevens 等人 1937, Stevens 和 Volkmann 1940)是一个音高单位。感知上音高相等的一对对声音被相同数量的梅尔隔开。梅尔频率 m 可以从原始声频通过对数变换得到:

$$mel(f) = 1127 \ln(1 + f/700) \tag{26.6}$$

我们通过创建一组从每个频带收集能量的滤波器来实现这一直觉，并按对数扩展，这样我们在低频时有很好的分辨率，而在高频时有较低的分辨率。图 26.5 显示了一个实现这个想法的三角滤波器样本库，它可以乘以声谱得到梅尔声谱。

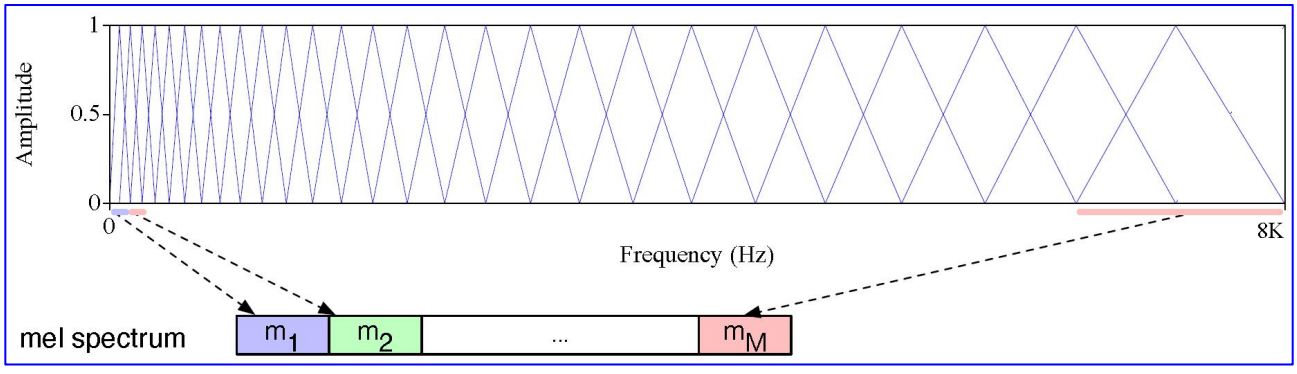


图 26-5：梅尔(mel)滤波器组

图注：每个三角形滤波器，沿着梅尔刻度对数间隔，从给定的频率范围收集能量。来自(Davis 和 Mermelstein，1980)。

最后，我们对每个梅尔声谱值取对数。人类对信号水平的反应是对数的(就像人类对频率的反应一样)。与低振幅相比，人类对高振幅的细微差异不那么敏感。此外，使用对数可以降低对输入变化的敏感度，比如由于说话者的嘴离麦克风更近或更远而引起的强度变化。

26.3. 语音识别的架构

ASR 的基本架构是编码器-解码器(通过 RNNs 或 Transformers 实现)，与第 11 章中介绍的 MT 的架构完全相同。通常，我们从上一节中描述的对数梅尔声谱特征开始，并映射到字母，尽管也可以映射到**诱导语素(induced morpheme)**，如组块、词片或 BPE。

图 26.6 概述了标准的编码器-解码器架构，通常被称为**基于注意力的编码器-解码器(attention-based encoder decoder，AED)**，或**聆听-注意力-拼写(listen attend and spell，LAS)**，这是根据两篇首次将其应用于语音的论文(Chorowski 等人 2014, Chan 等人 2016)。输入是由 t 个声学特征向量组成的序列 $F = f_1, f_2, \ldots, f_t$，每 10 毫秒帧有一个向量。输出可以是字母或词片;我们假设这里是字母。因此输出序列 Y = (<SOS>, y_1，…，y_m<EOS>)，假设序列的开始和结束具有特殊的符记<sos>和<eos>,而且每一个 y_i 是一个字;对于

英语，我们可以选择集合:

$$y_i \in \{a,b,c,...,z,0,...,9,\text{<space>},\text{<comma>},\text{<period>},\text{<apostrophe>},\text{<unk>}\}$$

当然，当输入和输出序列有明显的长度差异时，编码器-解码器架构尤其合适，就像语音一样，非常长的声学特征序列映射到非常短的字母或单词序列。一个单词可能有 5 个字母长，但假设它持续 2 秒，将需要 200 个声帧(每帧 10ms)。

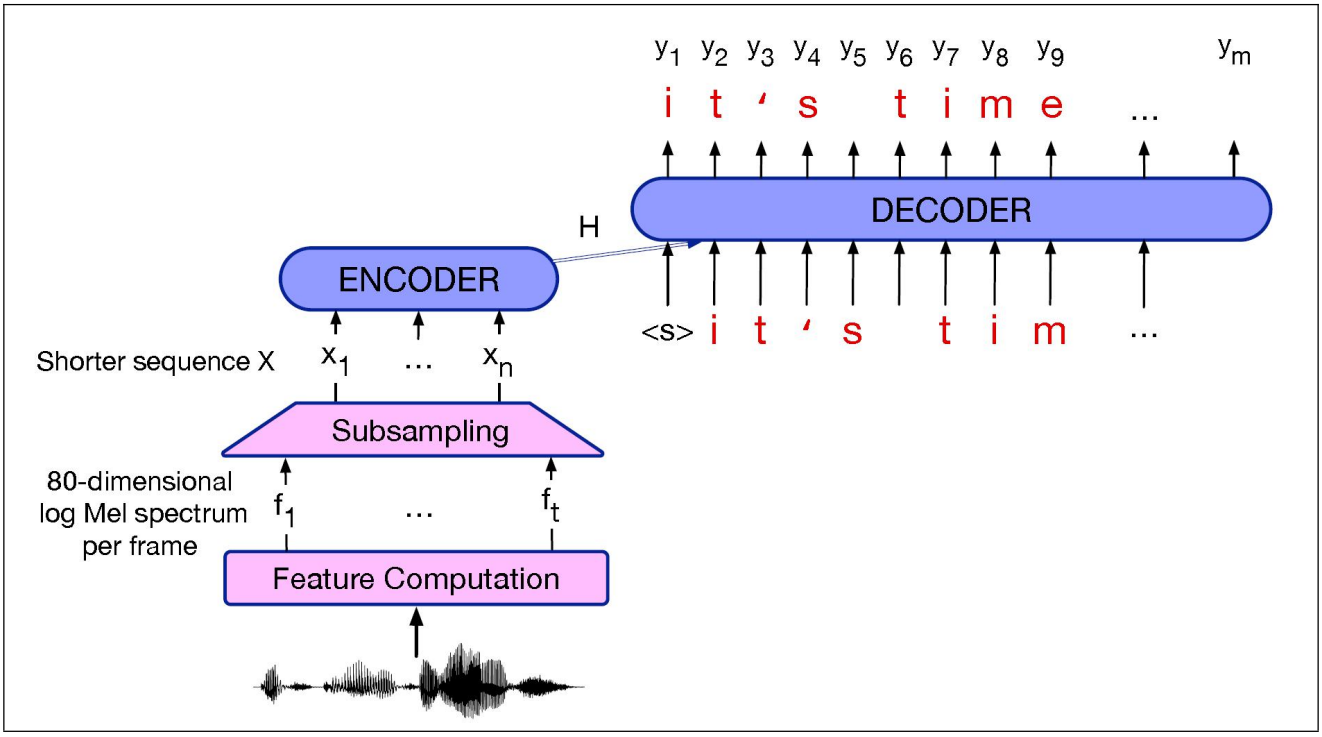


图 26-6：编码器-解码器语音识别器的架构原理图

由于这种长度差异对于语音来说是如此之大，因此语音的编码器-解码器架构需要有一个特殊的压缩阶段，在编码器阶段之前缩短声音特征序列。(或者，我们也可以使用专门设计用于处理压缩的损失函数，如我们将在下一节中介绍的 CTC 损失函数。)

子抽样的目标是产生一个更短的序列 $X = x_1,...,x_n$ 将是编码器的输入。最简单的算法是一种有时被称为**低帧率(low frame rate)**的方法(Pundak 和 Sainath, 2016):对于时间 i，我们堆叠(拼接)声学特征向量 f_i 与前两个向量 f_{i-1} 和 f_{i-2}，使一个新的向量更长三倍。然后我们简单地删除 f_{i-1} 和 f_{i-2}。因此，我们不是每 10 毫秒就有一个 40 维的声学特征向量，而是每 30 毫秒就有一个更长的向量(比如 120 维)，具有更短的序列长度 $n = t/3$。[1]

在这个压缩阶段之后，语音的编码器-解码器使用与 MT 或其他文本相同的架构，由 RNN(LSTMs)或 Transformers 组成。对于推理，将输出字符串 Y 的概率分解为:

$$\boldsymbol{p(y_1, \ldots, y_n) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, X)} \tag{26.7}$$

我们可以通过贪心解码产生输出的每个字母:

$$\hat{y}_i = \text{argmax}_{\text{char} \in \text{Alphabet}}\ \boldsymbol{p(\text{char}|y_1, \ldots, y_{i-1}, X)} \tag{26.8}$$

另外，我们也可以使用下一节中描述的波束搜索。当我们添加语言模型时，这一点尤为重要。

添加语言模型 由于编码器-解码器模型本质上是一个条件语言模型，编码器-解码器从他们的训练数据中隐式地学习字母输出域的语言模型。然而，训练数据(语音及其配对的文本转录)可能不包括足够的文本来训练一个好的语言模型。毕竟，找到大量的纯文本训练数据比找到与语音配对的文本更容易。因此，我们通常可以通过合并一个非常大的语言模型来对模型进行至少是轻微的改进。

最简单的方法是使用波束搜索来获得最后一束假设的句子;这种波束有时被称为 **n-最佳列表(n-best list)**。然后我们使用语言模型来**重新评价(rescore)**每一个假设。评分是通过将语言模型分配的分数与用于

[1] 对于子采样也有更复杂的替代方案，比如使用卷积网络用最大池化向下采样，或者多层金字塔型 RNN，其中每一层 RNN 的数量是前一层的一半。

创建波束的编码器-解码器分数进行插值来完成的，权重λ在**保留集**上进行了调整。同样，由于大多数模型都喜欢较短的句子，因此 ASR 系统通常可以通过某种方式增加长度因子。一种方法是通过假设$|Y|_c$中的字符数对概率进行归一化。因此，以下是典型的评分函数(Chan 等人，2016)：

$$score(Y|X) = \frac{1}{|Y|_c}\log P(Y|X) + \lambda \log P_{LM}(Y) \tag{26.9}$$

26.3.1. 学习

使用通常用于条件语言模型的正态交叉熵损失来训练语音编码器和解码器。在解码的时间步 i，损失为正确符记(字母)y_i的对数概率:

$$L_{CE} = -\log p(y_i|y_1, \ldots, y_{i-1}, X) \tag{26.10}$$

整个句子的损失是这些损失的总和:

$$L_{CE} = -\sum_{i=1}^{m} \log p(y_i|y_1, \ldots, y_{i-1}, X) \tag{26.11}$$

然后，这个损失通过整个端到端模型反向传播，以训练整个编码器-解码器。正如我们在第 11 章中所描述的，我们通常使用教师强制，在这种情况下，历史的解码器被迫是正确的黄金 y_i，而不是预测的$\hat{y}_i$。也可以混合使用黄金和解码器输出，例如 90%的时间使用黄金输出，但概率 10%采用解码器输出:

$$L_{CE} = -\log p(y_i|y_1, \ldots, \hat{y}_{i-1}, X) \tag{26.12}$$

26.4. CTC

我们在前一节中指出，语音识别有两个特殊的属性，使其非常适合于编码器-解码器架构，其中编码器产生输入的编码，解码器使用注意力来探索。第一，在语音中，我们有一个非常长的声音输入序列 X 映射到一个非常短的字母 Y 序列；第二，很难确切地知道 X 的哪个部分映射到 Y 的哪个部分。

在本节中，我们将简要介绍一种编码器-解码器的替代方案：一种称为 **CTC** 的算法和损失函数，即**连接主义时序分类(Connectionist Temporal Classification，CTC)**(Graves 等人，2006)，它以一种非常不同的方式处理这些问题。CTC 的直觉是为输入的每一帧输出一个单独字符，因此输出与输入的长度相同，然后应用一个折叠函数，将相同字母的序列组合在一起，从而得到一个更短的序列。

让我们想象某人说了 dinner 这个词，然后假设我们有一个函数为每个输入声谱帧表示 x_i 选择最可能的字母。我们将对应于每个输入帧的字母序列称为**对齐(alignment)**，因为它告诉我们每个字母在声音信号中的对齐位置。图 26.7 显示了一种这样的对齐方式，以及如果我们使用折叠函数来删除连续重复的字母会发生什么。

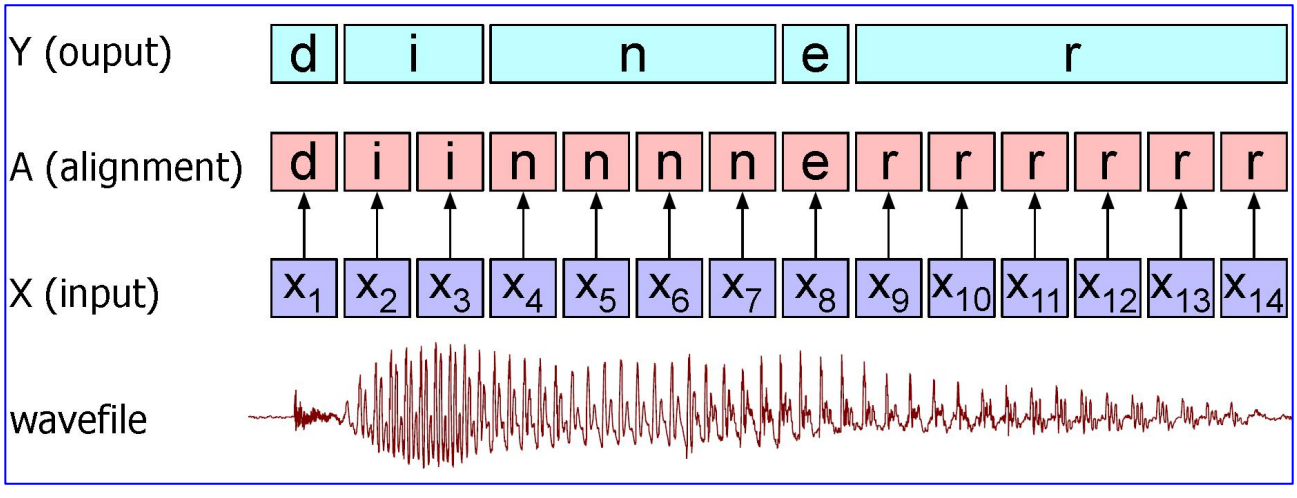


图 26-7：用于折叠输入和字母之间的对齐方式的朴素算法

这行不通;朴素算法把语言转录成了 diner，而不是 dinner！折叠不能处理双字母。朴素函数还有另一个问题;它没有告诉我们在输入中哪个符号与无声(silence)对齐。我们不想把无声转录成随意的字母!

CTC 算法通过在转录字母上添加**空格(blank)**的特殊符号来解决这两个问题，我们将其表示为 ␣。每当我们不想转录字母时，都可以在对齐方式中使用空格。字母之间也可以使用空格。由于我们的折叠功能仅折叠连续的重复字母，因此不会折叠到 ␣。更正式地说，让我们定义映射 B：a→y 在对齐 a 和输出 y 之间，它折叠所有重复的字母，然后删除所有空白。图 26.8 描绘了此折叠函数 B。

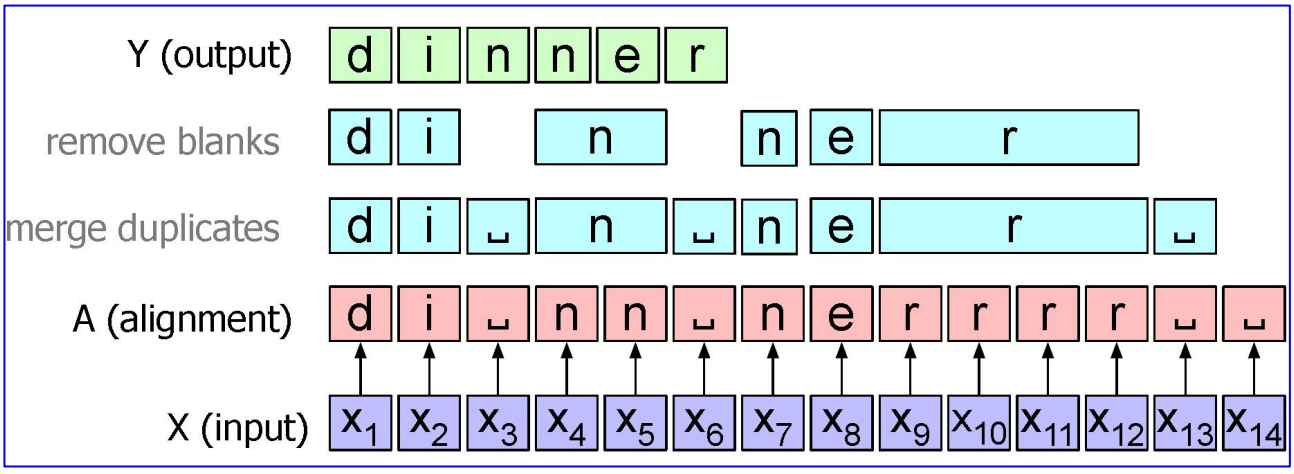


图 26-8：CTC 折叠函数 B

图注：CTC 折叠函数 B 显示空格字符 ␣； 删除对齐 A 中重复的(连续的)字符以形成输出 Y。

CTC 折叠函数是多对一的；许多不同的对齐方式映射到相同的输出字符串。例如，图 26.8 所示的对齐方式并不是导致字符串 dinner 的唯一对齐方式。图 26.9 显示了将产生相同输出的其他一些对齐方式。

考虑所有可能产生相同输出 Y 的对齐方式的集合很有用。我们将使用 B 函数的反函数 B^{-1}，并将其表示为 $B^{-1}(Y)$。

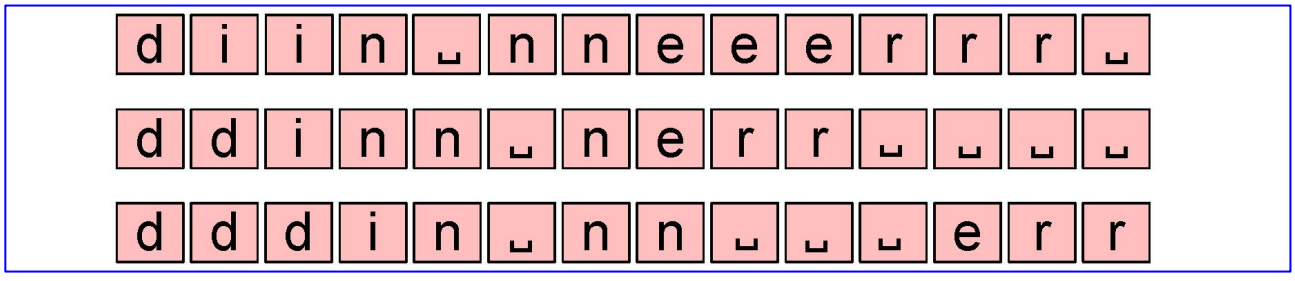


图 26-9：产生转录 dinner 的其他三个合法对齐

26.4.1. CTC 推理

在我们看到如何计算 $P_{CTC}(Y|X)$之前，让我们先看看 CTC 如何分配一个概率给一个特定的对齐 $\hat{A}=\{\hat{a}_1, \cdots,\hat{a}_n\}$。CTC 做了一个很强的条件独立性假设：它假设，给定输入 X，在 t 时刻的 CTC 模型输出 a_t 是独立于在任何其他时刻的输出标签 a_i。因此:

$$P_{\text{CTC}}(A|X) = \prod_{t=1}^{T} p(a_t|X) \tag{26.13}$$

从而找到最佳对齐$\hat{A}=\{\hat{a}_1, \cdots,\hat{a}_T\}$，我们可以在每个时间步 t 贪心地选择概率最大的字符:

$$\hat{a}_t = \underset{c \in C}{\operatorname{argmax}}\, p_t(c|X) \tag{26.14}$$

然后，我们将生成的序列 A 传递给 CTC 折叠函数 B，以得到输出序列 Y。

让我们来谈谈这个简单的推理算法是如何找到最佳对齐 A 的。因为我们在每个时间点都做出一个决定，所以我们可以将 CTC 视为一个序列建模任务，在这个任务中，我们在 t 时刻输出一个字母$\hat{y}_t$，对应于每个输入符记 x_t，从而不需要一个完整的解码器。图 26.10 勾画了这个架构，其中我们使用一个编码器，在每个时间步产生一个隐藏状态 h_t，并通过在每个时间步对字符词汇表使用 softmax 进行解码。

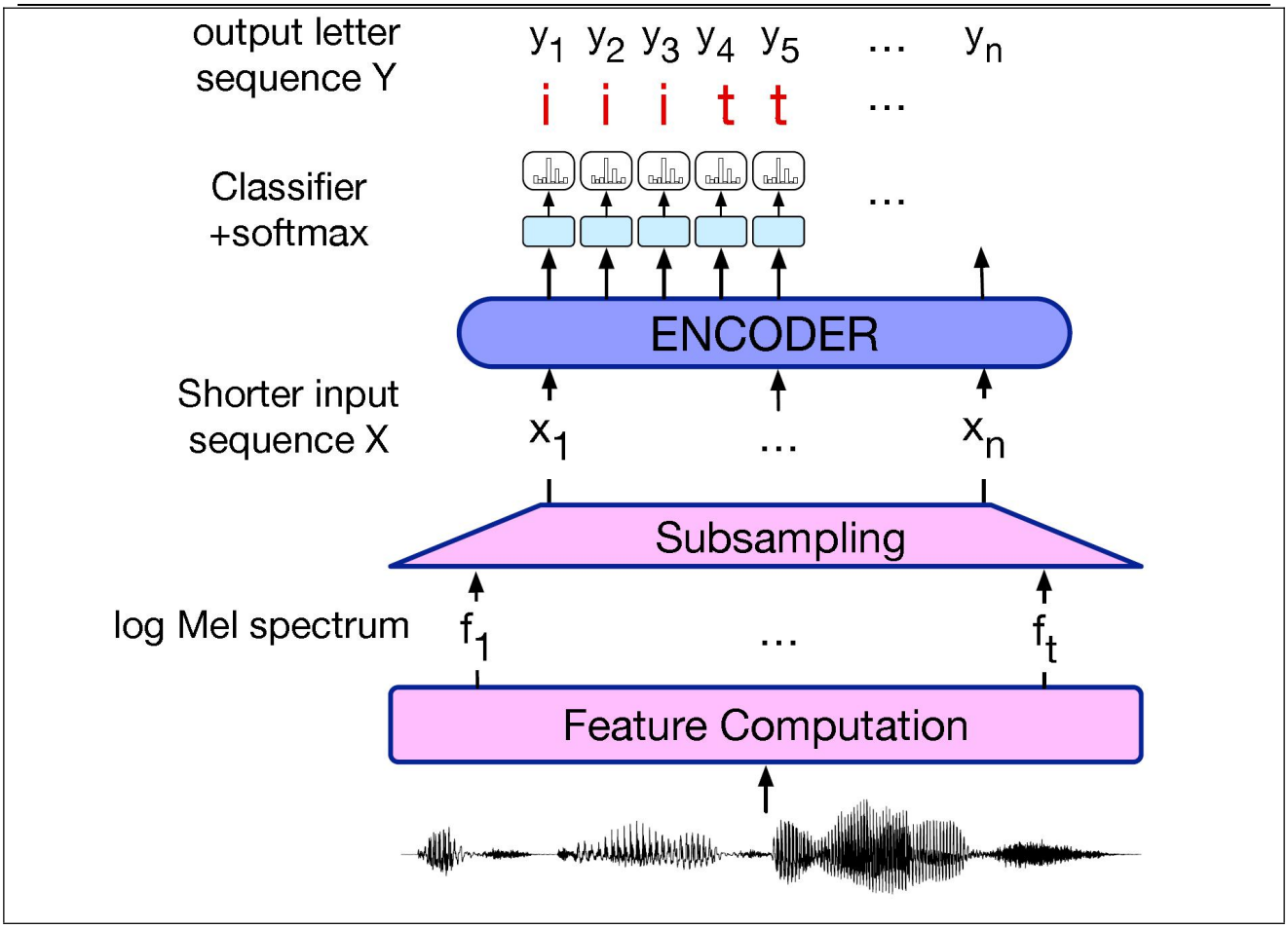


图 26-10：CTC 推理

图注：使用一个仅限编码器的模型，在每个输出步骤中，由简单的 softmax 通过隐藏状态 h_t 进行解码。

在公式 26.14 和图 26.9 中勾画的推理算法存在潜在的缺陷。问题是我们选择了最可能的对齐方式 A，但最可能的对齐方式可能与最可能的最终折叠输出字符串 Y 不对应。这是因为有很多可能的对齐方式导致相同的输出字符串，因此最可能的输出字符串可能对应于最可能的对齐。例如，假设输入 $X = [x_1\ x_2\ x_3]$的最可能对齐方式 A 是字符串[a b e]，但接下来的两个最可能的对齐方式是[b e b]和[e b b]。在这两个对齐方式上求和的输出 Y = [b b]比 Y = [a b]的可能性更大。因此，最有可能的输出序列 Y 不是唯一最好的 CTC 对齐，而是所有可能对齐概率的最高总和:

$$\begin{aligned} P_{CTC}(Y|X) &= \sum_{A \in B^{-1}(Y)} P(A|X) \\ &= \sum_{A \in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t) \\ \hat{Y} &= \underset{Y}{\operatorname{argmax}}\, P_{CTC}(Y|X) \end{aligned} \tag{26.15}$$

对所有对齐求总和的开销很大(有很多对齐)，因此我们使用维特比(Viterbi)波束搜索版本来近似估算此总和，该版本巧妙地将映射到同一输出字符串的高概率对齐保持在波束中，并将它们相加为公式 26.15 的近似值。参见 Hannun(2017)对 CTC 波束搜索这种扩展的清晰说明。

由于前面提到的强条件独立性假设(给定输入，t 时刻的输出独立于 t-1 时刻的输出)，CTC 不会通过数据隐含地学习语言模型(不像基于编码器-解码器架构的注意力)。因此，当使用 CTC 去插值语言模型(以及一些长度因子 L(Y))时，使用在开发集上训练的插值权重是非常重要的:

$$score_{\text{CTC}}(Y|X) = \log P_{\text{CTC}}(Y|X) + \lambda_1 \log P_{LM}(Y)\lambda_2 L(Y) \tag{26.16}$$

26.4.2. CTC 训练

为了训练一个基于 CTC 的 ASR 系统，我们使用负对数似然损失和一个特殊的 CTC 损失函数。因此，整个数据集 D 的损失是每个输入 X 的正确输出 Y 的负对数似然之和:

$$L_{CTC} = \sum_{(X,Y)\in D} -\log P_{\text{CTC}}(Y|X) \tag{26.17}$$

为了计算单个输入对(X，Y)的 CTC 损失函数，我们需要在给定输入 X 的条件下输出 Y 的概率。如公式(26.15)所示，要计算给定输出 Y 的概率，我们需要对所有可能折叠为 Y 的对齐求和。换言之：

$$P_{\text{CTC}}(Y|X) = \sum_{A\in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t) \tag{26.18}$$

朴素地对所有可能的对齐求总和是不可行的(有太多的对齐)。但是，我们可以使用动态规划合并对齐来有效地计算总和，同时还使用一种用于训练 HMM(附录 A)和 CRF 的前向-后向算法。训练和推理的原始动态规划算法在(Graves 等人, 2006);参见(Hannun, 2017)对这两方面的详细解释。

26.4.3. CTC 和编码器-解码器的结合

也可以将我们描述的两个损失组合起来：第一个损失是架构/损失函数(即：编码器-解码器架构的交叉熵损失)，第二个损失是 CTC 损失。图 26.11 显示了一个草图。

对于训练，我们可以简单地用在开发集上调整的λ参数来加权两个损失：

$$L = -\lambda \log P_{encdec}(Y|X) - (1-\lambda)\log P_{ctc}(Y|X) \tag{26.19}$$

对于推理，我们可以将两者与语言模型(或长度惩罚)结合起来，再一次使用学习过的权值:

$$\hat{Y} = \underset{Y}{\text{argmax}} \left[\lambda \log P_{encdec}(Y|X) - (1-\lambda)\log P_{CTC}(Y|X) + \gamma \log P_{LM}(Y)\right] \tag{26.20}$$

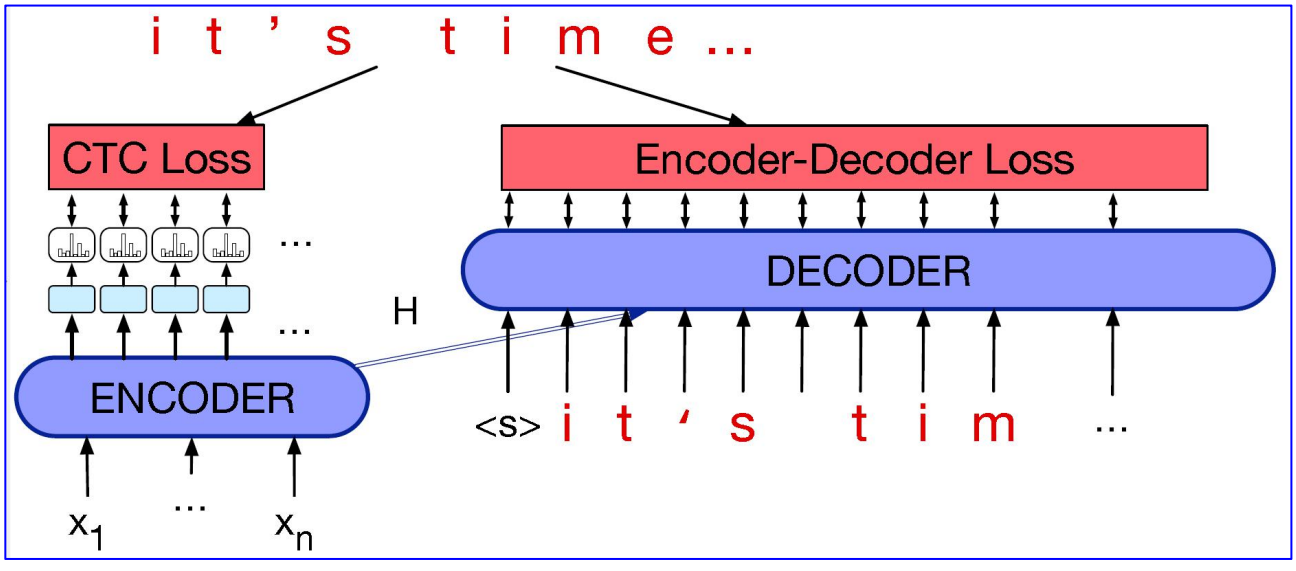


图 26-11：结合 CTC 和编译码器损失函数

26.4.4. 流模型:改进 CTC 的 RNN-T

由于 CTC 的强独立性假设(假设 t 时刻的输出与 t-1 时刻的输出是独立的)，基于 CTC 的识别器不能达到像基于注意力的编码器-解码器识别器那样高的准确度。然而，CTC 识别器有一个优势，那就是它们可以用于**流(streaming)**。流意味着在线识别单词，而不是等到句子结束再识别。流对许多应用来说都是至关重要的，从命令到听写，我们希望在用户还在说话的时候就开始识别。在解码器开始解码之前，使用注意力的算法需要首先计算整个输入的隐藏状态序列，以便提供注意力分布上下文。相比之下，CTC 算法可以立即从左到右输入字母。

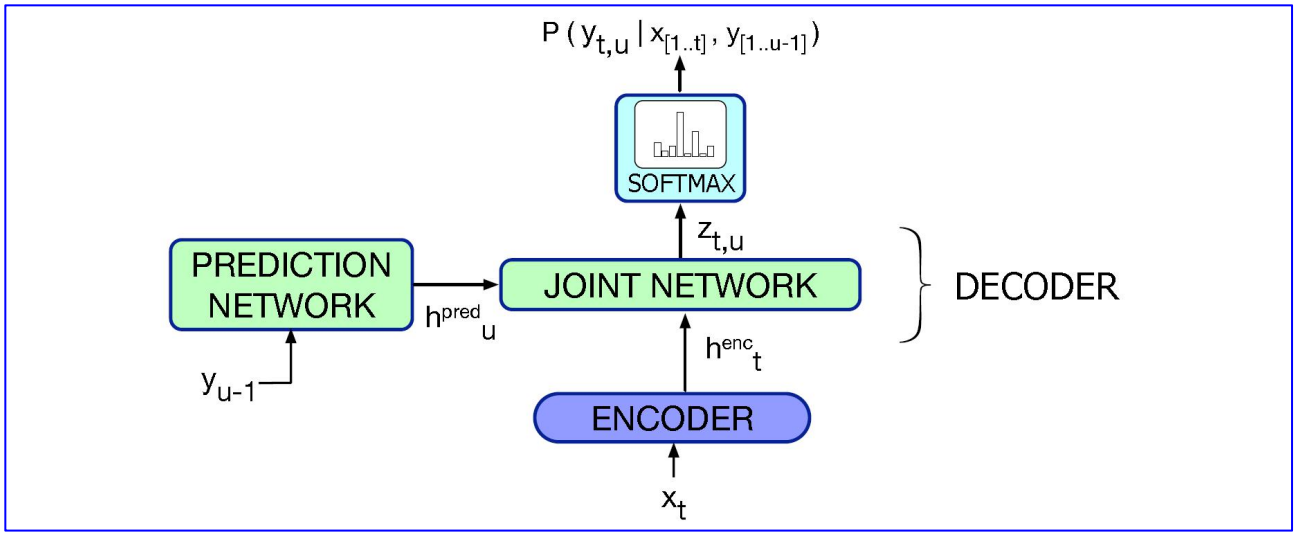


图 26-12：RNN-T 模型计算 t 时刻的输出符记分布

图注：通过集成 CTC 声音编码器的输出和单独的“预测器”语言模型进行计算。

如果我们想做流，我们需要一种方法来提高 CTC 识别，以消除条件无关的假设，使它知道输出历史。如图 26.12 所示的 **RNN-Transducer(RNN-T)**就是这样一个模型(Graves 2012, Graves 等人 2013a)。RNN-T 有两个主要组件:一个 CTC 声学模型；一个独立的语言模型组件，称为对输出符记历史进行调节的预测器。在每个时间步长 t，给定输入 $x_1 \ldots x_t$，CTC 编码器输出一个隐藏状态 h_t^{enc}。语言模型预测器将之前的输出符记(不计算空格)作为输入，输出一个隐藏状态 h_u^{pred}。两者通过另一个网络，该网络的输出通过 softmax 来预测下一个字符。

$$P_{RNN-T}(Y|X) = \sum_{A \in B^{-1}(Y)} P(A|X)$$

$$= \sum_{A \in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t, y_{<u_t})$$

26.5. ASR 评估:单词错误率

语音识别系统的标准评价指标是**单词错误率(word error rate，WER)**。单词错误率是基于识别器返回的单词字符串(假设的单词字符串)与参考转录的差异程度。计算单词错误的第一步是计算假设字符串和正确字符串之间单词的最小编辑距离，从而给出在正确字符串和假设字符串之间映射所需的最小单词替换、单词插入和单词删除次数。单词错误率(WER)定义如下(注意，由于公式中包含了插入，错误率可以大于 100%):

$$\text{Word Error Rate} = 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

以下是来自 CallHome 语料库的参考和假设语段之间的样本**对齐(alignment)**，显示了用于计算错误率

的计数:

REF:	i	***	**	UM	the	PHONE	IS	i	LEFT	THE	portable	****	PHONE	UPSTAIRS	last night
HYP:	i	GOT	IT	TO	the	*****	FULLEST	i	LOVE	TO	portable	FORM	OF	STORES	last night
Eval:		I	I	S		D	S		S	S		I	S	S	

这个语段有六处替换，三处插入，一处删除:

Word Error Rate = 100 *(6+3+1)/13= 76.9%

计算单词错误率的标准方法是一个名为 sclite 的免费脚本，可以从美国国家标准与技术研究所(NIST, 2005)获得。Sclite 提供了一系列参考(手工抄写的，黄金标准的)句子和一组匹配的假设句子。除了执行对齐和计算单词错误率，sclite 还执行许多其他有用的任务。例如，对于错误分析，它提供有用的信息，例如混淆矩阵，显示哪些单词经常被错误识别，并总结经常被插入或删除的单词的统计信息。sclite 还提供了按说话人划分的错误率(如果句子被标签为说话人 ID)，以及诸如**句子错误率(Sentence error rate)**(至少有一个单词错误的句子的百分比)等有用的统计数据。

ASR 的统计学意义:MAPSSWE 或 MacNemar

与其他语言处理算法一样，我们需要知道在单词错误率方面的特定改进是否显著。

决定两个单词错误率是否不同的标准统计测试是**配对句子片段单词错误率(MAtched-Pair Sentence Segment Word Error，MAPSSWE)**测试，它在 Gillick 和 Cox(1989)中被引入。

MAPSSWE 测试是一个参数测试，查看两个系统产生的单词错误数量之间的差异，在多个片段上平均。这些片段可以很短，也可以和整个语段一样长;一般来说，我们希望有最大数量的(短)片段，以证明正常假设和最大化能力。测试要求一个片段的误差在统计上独立于另一个片段的误差。由于 ASR 系统倾向于使用三元语法模型，我们可以通过将一个片段定义为一个区域，该区域由两个识别器都得到正确的单词(或话轮/语段边界)界定。这里有一个来自 NIST(2007)的例子，有四个区域:

```
         I              II              III                IV
REF:  |it was|the best|of|times it|was the worst|of times|   |it was
      |      |        |  |        |             |        |   |
SYS A:|ITS   |the best|of|times it|IS the worst |of times|OR |it was
      |      |        |  |        |             |        |   |
SYS B:|it was|the best|  |times it|WON the TEST |of times|   |it was
```

在区域 I 中，系统 A 有两个错误(一个删除和一个插入)，而系统 B 有零个错误;在区域 III 中，系统 A 有一个错误(A 替换)，系统 B 有两个错误。定义一个变量序列 Z，表示两个系统错误的差异，如下所示:

N_A^i the number of errors made on segment i by system A

N_B^i the number of errors made on segment i by system B

Z $N_A^i - N_B^i, i = 1, 2, \cdots, n$ where n is the number of segments

在上面的例子中，Z 值序列是{2,-1,-1,1}。直观地说，如果两个系统是相同的，我们会期望平均差值，即 Z 值的平均值，为零。如果我们称这些差值的真实平均值为 μ_z，那么我们想知道是否 $\mu_z = 0$。遵循 Gillick 和 Cox(1989)的原始建议和评注，我们可以从我们有限的样本中估计出真实平均值 $\hat{u}_z = \sum_{i=1}^{n} Zi/n$。Zi 的方差的估计是:

$$\sigma_z^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \mu_z)^2 \tag{26.21}$$

令

$$W = \frac{\hat{\mu}_z}{\sigma_z/\sqrt{n}} \tag{26.22}$$

对于足够大的 n(> 50)，W 将近似具有单位方差的正态分布。零假设为 H_0：$\mu_z = 0$，因此如果 2* P(Z ≥|w|)≤0.05(两尾)或 P(Z≥|w|)≤0.05(单尾)，则可以将其拒绝。其中 Z 是标准正态，w 是 W 实现的值；这些概率可以在正态分布的标准表中查找。

早期的研究有时使用 **McNemar's 测试**显著性，但 McNemar's 只适用于系统的错误是独立的情况，这在连续语音识别中是不成立的，在连续语音识别中，一个单词的错误非常依赖于相邻单词的错误。

我们可以改进单词错误率作为一个度量标准吗?举个例子，如果不给每个单词同等的权重，比如像 Tuesday 这样的实词比 a 或 of 这样的虚词更重要，那就更好了。虽然研究人员普遍认为这是一个好主意，但事实证明，很难就一个适用于所有 ASR 应用的指标达成一致。然而，对于期望的语义输出更加清晰的对话系统来说，一种被称为槽错误率或概念错误率的指标已经被证明非常有用;在第 24.5.2 节进行了讨论。

26.6. TTS

文本语音转换(Text-To-Speech，TTS)系统的目标是将字符串映射为波形，从对话系统到游戏到教育等各种应用，这都是一种很重要的技术。

与 ASR 系统一样，TTS 系统通常基于编码器-解码器架构，使用 LSTMs 或 Transformers。在训练方面有一个普遍的差异。ASR 系统的默认条件是与说话者无关:它们在许多说话者的数千小时演讲的大型语料库上进行训练，因为它们必须很好地生成一个看不见的测试说话者。相比之下，在 TTS 中，使用多个声音就不那么重要了，所以基本的 TTS 系统依赖于说话者:训练拥有一致的声音，使用更少的数据，但所有声音都来自一个说话者。例如，一个常用的公共领域数据集(LJ 语音语料库)，由一个发言者(琳达·约翰逊)，在 LibriVox 项目(Ito 和 Johnson, 2017)中，阅读有声书籍的 24 小时，远远小于标准的 ASR 语料库的数百或数千小时。[2]

我们通常把 TTS 任务分成两个组件。

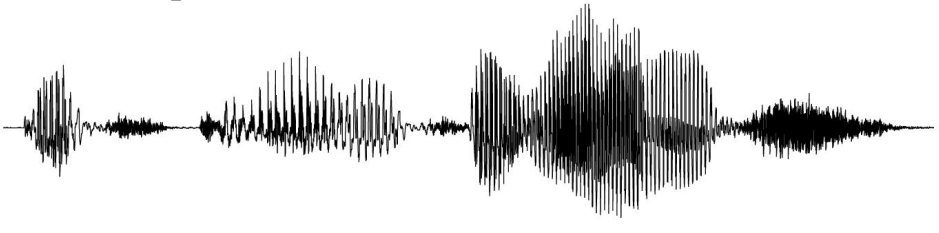
第一个组件是一个用于频谱预测的编码器-解码器模型:它从字符串的字母映射到 mel 频谱：mel 声谱随时间变化的序列。因此，我们可以从这个字符串映射:

It's time for lunch!

到如下的 mel 频谱：



第二个组件是从 mel 频谱图映射到波形。从像频谱这样的中间表现形式生成波形被称为**声码(vocoding)**，这个第二组件被称为**声码器(vocoder)**:



这些用于 TTS 的标准编码器-解码器算法仍然需要大量的计算，因此现代研究的一个重要重点是如何提高它们的速度。

[2] 最近也有关于多说话人 TTS 任务的 TTS 研究，其中一个系统可以训练来自多个说话人的语音，并且可以在不同的声音之间切换。

26.6.1. TTS 预处理:文本规范化

但是，在这两个步骤中的任何一个步骤之前，TTS 系统都需要进行文本规范化预处理，以处理**非标准词(non-standard word)**：数字、金额、日期以及其他口头表达与拼写不同的概念。当看到像 151 此类数字时，TTS 系统需要知道口头表达：如果出现$151，则为 one hundred fifty one；但是如果出现上下文 151 Chapultepec Ave.，则为 one fifty one。根据上下文，1750 数字可以通过至少四种不同的方式被说出来：

seventeen fifty: (in *"The European economy in 1750"*)
one seven five zero: (in *"The password is 1750"*)
seventeen hundred and fifty: (in *"1750 dollars"*)
one thousand, seven hundred, and fifty: (in *"1750 dollars"*)

通常，非标准词的语言化取决于其含义(Taylor(2009)称其为符号学类)。图 26.13 列出了一些英语非标准词类。

semiotic class	examples	verbalization
abbreviations	**gov't**, *N.Y., mph*	government
acronyms read as letters	**GPU**, *D.C., PC, UN, IBM*	G P U
cardinal numbers	**12**, *45, 1/2, 0.6*	twelve
ordinal numbers	*May* **7**, *3rd, Bill Gates III*	seventh
numbers read as digits	*Room* **101**	one oh one
times	*3.20,* **11:45**	eleven forty five
dates	**28/02** *(or in US, 2/28)*	February twenty eighth
years	**1999**, *80s, 1900s, 2045*	nineteen ninety nine
money	**$3.45**, *€250, $200K*	three dollars forty five
money in tr/m/billions	**$3.45 billion**	three point four five billion dollars
percentage	**75%** *3.4%*	seventy five percent

图 26-13：文本规范化中的一些非标准词类

图注：更多信息见 Sproat 等人(2001)和(van Esch 和 Sproat, 2018)。

许多类都偏爱实现。年份通常被读成成对的数字(例如，1750 年读为 seventeen fifty)。32 亿美元($ 32 billion)读出来的时候必须以单词 dollar 结尾，就像 three point two billion dollars 一样，有些缩写(例如 N.Y.)被扩展(到纽约)，而其他首字母缩写词(如 GPU)被发音为字母序列。在具有语法性别的语言中，规范化可能取决于形态学特征。在法语中，短语 1 mangue("one mangue")被规范化为 une mangue，而 1 ananas("one pineapple")被规范化为 un ananas。在德语中，根据名词的语法情况，可以将 Heinrich IV("Henry IV")规范化为 Heinrich der Vierte，Heinrich des Vierten，Heinrich dem Vierten 或 Heinrich den Vierten(Demberg，2006)。

现代端到端的 TTS 系统可以自己学习做一些规范化，但 TTS 系统只在有限的数据量上进行训练(如我们上面提到的 LJ 语料库的 22 万单词(Ito 和 Johnson, 2017))，所以单独的规范化步骤是重要的。

规范化可以通过规则或编码器-解码器模型来实现。基于规则的规范化分为两个阶段:符记化和语言化。在符记化阶段，我们用手来书写规则去检测非标准词。这些可以是正则表达式，像下面的检测年份:

/(1[89][0-9][0-9])|(20[0-9][0-9]/

规则的第二个阶段表达如何用语言表达每个符号学类。更大的 TTS 系统使用更复杂的规则系统，如 Kestral 系统(Ebden 和 Sproat, 2015)，它首先将每个输入分类并解析为标准形式，然后使用语言化语法生成文本。规则的优点是它们不需要训练数据，并且可以为高精度而设计，但是它们可能很脆弱，并且需要专业的规则编写者，因此很难维护。

另一种模型是使用编码器-解码器模型，它已经被证明比这种转换任务的规则工作得更好，但确实需要专家标签的训练集，其中非标准的单词被适当的语言化取代;一些语言的训练集是可用的(Sproat 和 Gorman 2018, Zhang 等人 2019)。

在最简单的编码器-解码器设置中，我们简单地对待问题，如机器翻译，训练系统映射从:

They live at 224 Mission St.

到

They live at two twenty four Mission Street

尽管编码器-解码器算法非常准确，但有时会产生严重的错误；例如将 45 minutes 规范化为 forty five millimeters。为了解决这个问题，更复杂的系统使用了某些机制(例如轻量级的覆盖文法，它列举了大量可能的语言化但不试图消除歧义)来限制解码以避免此类输出(Zhang 等人，2019)。

26.6.2. TTS：频谱预测

我们为 ASR 描述的完全相同的架构—带注意力的编码器-解码器—可以用于 TTS 的第一个组件。这里我们将简要介绍 **Tacotron2** 架构(Shen 等人，2018)，它扩展了早期的 Tacotron 架构(Wang 等人，2017)和 **Wavenet** 声码器(van den Oord 等人，2016)。图 26.14 整个架构示意图。

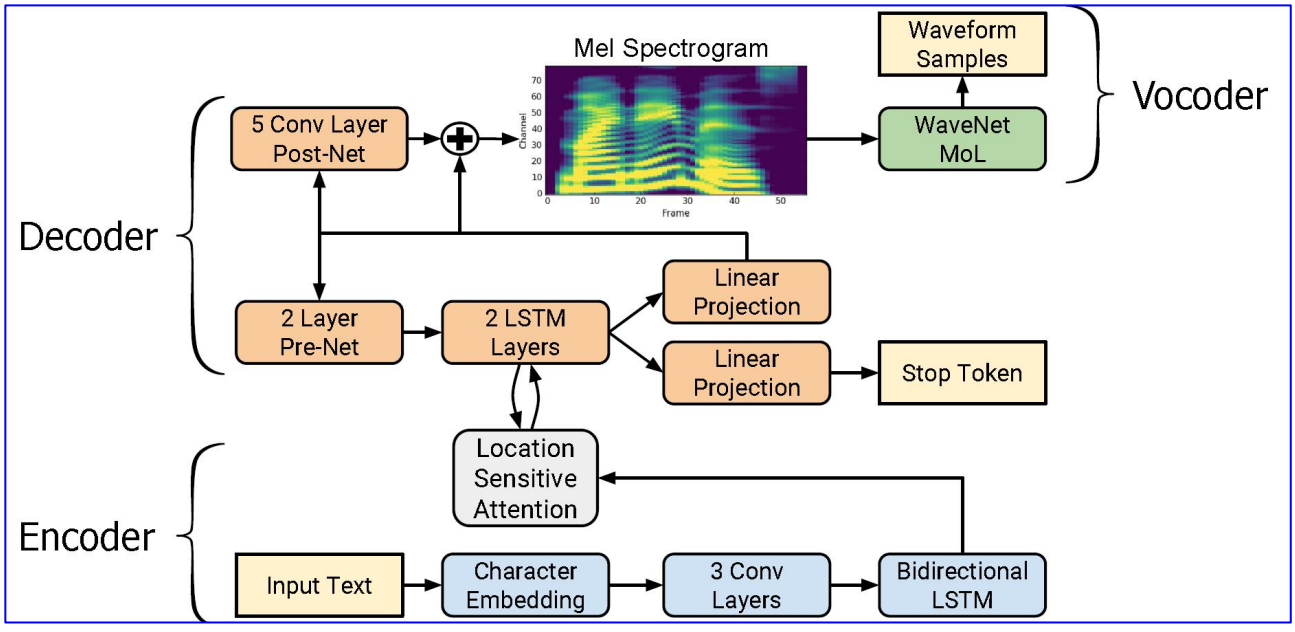


图 26-14：Tacotron2 架构

图注：编码器-解码器从字素映射到梅尔频谱图，然后是声码器映射到波形文件。图由 Shenet 等人(2018)修改。

编码器的工作是获取一个字母序列，并生成一个代表该字母序列的隐藏表示法，然后将其用于解码器的注意力机制。Tacotron2 编码器首先将每个输入**字素(grapheme)**映射为一个 512 维嵌入字符。然后通过 3 个卷积层的堆叠，每个层包含 512 个形状为 5×1 的过滤器，即每个过滤器跨越 5 个字符，以建模更大的字母上下文。最终卷积层的输出通过 biLSTM 生成最终编码。我们通常会使用一种质量稍高(但速度较慢)的注意力版本，即**基于位置的注意力(location-based attention)**，其中 α 值的计算(第 11 章中公式 11.17)利用了之前时间状态的 α 值。

在解码器中，从前一个时间槽中预测出的 mel 声谱将通过一个小的预网络(pre-net)当作一个瓶颈。然后，这个优先输出与编码器的注意力向量上下文进行拼接，并通过 2 个 LSTM 层。这个 LSTM 的输出有两种用法。首先，通过线性层和一些输出处理，每一步自回归预测一个 80 维对数-梅尔过滤器组(log-mel filterbank)向量帧(50 ms，用 12.5 ms 步长)。其次，它通过另一个线性层传递到 sigmoid，以做出关于是否停止生产输出的“停止符记预测”决策。

使用教师强制，对系统进行了黄金对数梅尔滤波器组特征的训练，即在每个解码器步骤向解码器提供正确的对数模型声谱特征，而不是从先前步骤输出的预测解码器。

26.6.3. TTS：声码

Tacotron2 的声码器是 WaveNet 声码器的改编版(van den Oord 等人，2016)。在这里，我们将对使用 WaveNet 的声码进行一些简化的描述。

回想一下，此处声码处理的目标是将对数梅尔声谱表示转换为时域波形表示。WaveNet 是一种自回归网络，就像我们在第 9 章中介绍的语言模型一样。它以频谱作为输入，并产生表示为 8 位 mu-law 序列的音频输出(第 25.4.2 节)。给定中间输入梅尔频谱 h 时，波形的概率(一系列 8 位 mu-law 值 $Y=y_{1,...,}y_t$)计算如下：

$$p(Y)=\prod_{t=1}^{t} P(y_t|y_1,...,y_{t-1},h_1,...,h_t) \tag{26.23}$$

这种概率分布是由一叠特殊的卷积层建模的，其中包括一个被称为**扩张卷积(dilated convolutions)**的特定卷积结构，和一个特定的非线性函数。

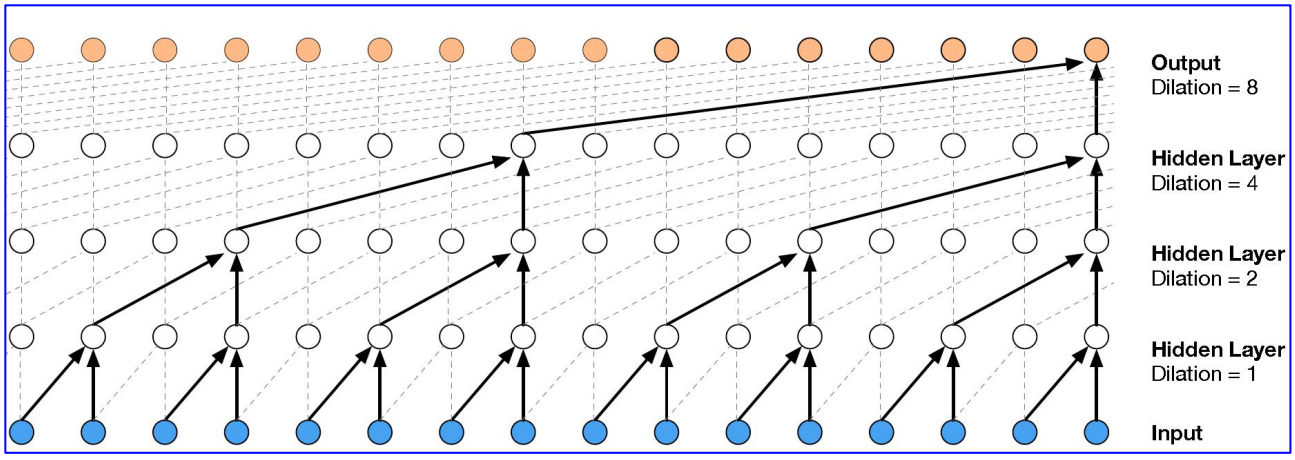


图 26-15：扩张卷积

图注：显示一个扩张周期大小为 4，即扩张值为 1, 2, 4, 8。数据来自 van den Oord 等人(2016)。

扩张卷积是因果卷积层的子类型。**因果(causal)**卷积或**掩码(masked)**卷积仅着眼于过去的输入，而不着眼于未来。y_{t+1} 的预测只能取决于 $y_{1,...,}y_t$，对自回归从左到右处理很有用。在扩张卷积中，在每个连续层，我们通过跳过输入值，在比其长度更长的跨度上应用卷积滤波器。因此，在时间 t 的扩张值为 1 时，长度为 2 的卷积滤波器将看到输入值 x_t 和 x_{t-1}。但是**蒸馏(distillation)**值为 2 的滤波器将跳过输入，因此将看到输入值 x_t 和 x_{t-1}。图 26.15 示出了在时间 t 处具有 4 个扩张卷积层，扩张值分别为 1、2、4 和 8 的输出的计算。

Tacotron 2 合成器在两个循环中使用 12 个卷积层，其扩张循环大小为 6，这意味着前 6 个层的扩张为 1、2、4、8、16 和 32。接下来的 6 层又具有 1、2、4、8、16 和 32 的扩张。扩张的卷积使声码器可以随着深度成倍地增大接收场。

WaveNet 可以预测 mu-law 音频样本。从第 25.4.2 节回想起，这是音频的标准压缩，其中每个采样时间步长的值都压缩为 8 位。这意味着我们可以使用简单的 256-way 分类器来预测每个样本的值。扩张卷积的输出因此通过 softmay 做出了 256-way 的决策。

频谱预测编解码器和 WaveNet 声码器是分开训练的。训练完频谱预测器后，频谱预测网络将在教师强制模式下运行，每个预测的声谱帧均以输入的编码文本和**基本事实(ground truth)**频谱的前一帧为条件。然后，将这一系列与基本事实对齐的声谱特征和黄金音频输出用于训练声码器。

这只是 TTS 流程的高级概述。有兴趣进一步研究 TTS 的读者可能需要研究许多重要的细节。例如，WaveNet 使用一种特殊的门控激活函数作为其非线性，并包含**残余(rsidual)**和**跳过(skip)**连接。实际上，预测 8 位音频值的效果不如 16 位效果好，因为简单的 softmax 不够用，因此解码器将更复杂的方法用作预测音频样本值的最后一步，例如分布的混合。最终，如我们所描述的，WaveNet 声码器太慢了，以至于

毫无用处。在实践中，有许多不同类型的效率改进是必要的，例如，通过找到进行非自回归生成的方式，避免必须等待生成每个帧直到生成先前帧的等待时间，以及并行进行预测。我们鼓励有兴趣的读者查阅原始文件和不同版本的代码。

26.6.4. TTS 评估

语音合成系统是由人类听者来评估的。(开发一种用于综合评价的良好的自动度量方法，从而消除对昂贵和耗时的人类听力实验的需要，仍然是一个开放而令人兴奋的研究课题。)

我们通过向听众播放一个句子来评估合成语段的质量，并要求他们给出**平均意见评分(mean opinion score，MOS)**，即对综合语段的评价，通常在 1-5 的范围内。然后，我们可以通过比较同一句子上的 MOS 得分来比较系统(例如使用配对的 t 检验来测试显著性差异)。

如果我们正好比较两个系统(也许是为了看看某个特定的改变是否确实改善了系统)，我们可以使用 **AB 测试(AB tests)**。在 AB 测试中，我们播放由两个不同系统(A 和 B 系统)合成的同一个句子。人类的听者会从这两种语段中选择他们更喜欢哪一种。我们对 50 个句子(以随机顺序呈现)这样做，并比较每个系统首选的句子数量。

26.7. 其他语音任务

虽然在这一章中我们关注的是 ASR 和 TTS，但是还有很多与语音相关的任务。

唤醒词(wake word)检测的任务是检测一个单词或短语，通常是为了唤醒 Alexa、Siri 或谷歌助手等语音助手。唤醒词的目标是在计算边缘的小型设备中构建检测，通过向基于云的服务器传输最少的用户语音来维护隐私。因此，唤醒字探测器需要快速、占用空间小的软件，能够适合嵌入式设备。唤醒词检测器通常使用与我们看到的 ASR 相同的前端特征提取，然后通常是一个全词分类器。

说话人区分(Speaker diarization)的任务是在一个长时间的多说话者录音中确定“谁在什么时候说话”，标记每个说话者的话轮在互动中的开始和结束时间。这对于记录会议、课堂演讲或医疗互动很有用。区分系统通常使用语音活动检测(VAD)来寻找连续语音片段，提取说话者嵌入向量，并将向量聚类以将可能来自同一说话者的片段分组。最近的工作是研究端到端的算法，将输入语音直接映射到每一帧的一系列说话者标签。

声纹识别(Speaker recognition)是识别说话者的任务。通常，我们会区分说话人验证的子任务，在此我们做出二元决策(这个说话人是否 X？)，例如为了确保通过电话访问个人信息时的安全性，以及**说话人识别(speaker identification)**，在此我们做出 N 个决策之一，使演讲者的声音与许多演讲者的数据库相匹配。这些任务与**语言识别(language identification)**有关，在其中给了我们一个波形文件，必须识别说的是哪种语言。例如，这对于自动将呼叫者定向到会说适当语言的人工操作员非常有用。

26.8. 总结

本章介绍了**自动语音识别(ASR)**和**文本语音转换(TTS)**的基本算法。

- **语音识别**(或语音到文本转换)的任务是将声波波形映射到**字素**序列。
- 语音识别器的输入是一系列声波。它们被**采样**，**量化**，并转换为像**对数梅尔声谱**那样的**声谱表示**。
- 语音识别的两种常见范式是**带有注意力模型的编码器-解码器**，以及**基于 CTC 损失函数**的模型。基于注意力的模型具有更高的准确度，但基于 CTC 的模型更容易适应于**流**：在线输出字母，而不是等待声音输入完成。
- 评估 ASR 使用**单词错误率**、假设和黄金转录之间的编辑距离。
- TTS 系统也**基于编码器-解码器架构**。编码器将字母映射到编码，解码器使用编码生成 mel **频谱**输出。神经声码器读取频谱并生成波形。
- TTS 系统需要首先进行**文本规范化**，以处理数字、缩写和其他非标准词。
- TTS 的评估是通过播放一个句子给人类听众，并让他们给出一个**平均意见分数(MOS)**或做 AB 测试。

26.9. 文献和历史说明

ASR

到 1940 年代末和 1950 年代初，开发了许多语音识别系统。早期的 Bell Labs 系统可以从单个说话者中识别 10 个数字中的任何一个(Davis 等人，1952)。该系统具有 10 个与说话者相关的存储模式，每个数字存储一个模式，每个模式大致代表该数字的前两个元音共振峰。通过选择与输入相对相关系数最高的模式，他们达到了 97%–99%的准确度。Fry(1959)和 Denes(1959)在伦敦大学学院建立了一个音素识别器，它基于相似的模式识别原理识别了四个元音和九个辅音。Fry 和 Denes 的系统是第一个使用音素转换概率来约束识别器的系统。

1960 年代末和 1970 年代初发生了许多重要的范式转变。首先是许多特征提取算法，包括高效的快速傅立叶变换(FFT)(Cooley 和 Tukey，1965)，倒谱处理在语音中的应用(Oppenheim 等人，1968)以及语音编码(Atal 和 Hanauer，1971)LPC 的开发。其次是多种处理**翘曲变形(warping)**的方法。当与存储的模式匹配时，可以拉伸或收缩输入信号以处理语音速率和段长的差异。解决此问题的自然算法是动态规划，正如我们在附录 A 中所见，该算法经过多次重新发明以解决此问题。语音处理的第一个应用是 Vintsyuk(1968)，尽管他的研究结果未被其他研究人员所采用，但被 Velichko 和 Zagoruyko(1970)以及 Sakoe 和 Chiba(1971)和(1984)重新发明了。此后不久，Itakura(1975)将这种动态规划思想与以前仅用于语音编码的 LPC 系数相结合。结果系统从输入的单词中提取了 LPC 特征，并使用动态规划将其与存储的 LPC 模板进行匹配。动态规划将模板与传入语音进行匹配的非概率性使用称为**动态时间翘曲变形(dynamic time warping)**。

这一时期的第三个创新是 HMM(隐马尔可夫模型)的兴起。HMM 似乎在 1972 年前后被两个实验室独立地应用于语言。其中一个应用源于统计学家的工作，特别是鲍姆和普林斯顿国防分析研究所的同事们，他们将 HMM 应用于各种预测问题(Baum 和 Petrie 1966, Baum 和 Eagon 1967)。James Baker 在 CMU 读研究生时了解到这项工作并将算法应用于语音处理(Baker 1975a)。在 IBM 托马斯 • j • 沃森研究中心(Jelinek 等人 1975)，Frederick Jelinek 和他的合作者(从他们的信息理论模型研究中得到 Shannon(1948)的影响)独立地将 HMM 应用到语音中。一个早期的区别是译码算法;贝克的 DRAGON 系统使用的是 Viterbi(动态规划)解码，而 IBM 系统使用的是 Jelinek 的堆栈解码算法(Jelinek，1969)。贝克随后短暂加入 IBM 集团，之后创立了语音识别公司龙系统(Dragon Systems)。

以高斯混合模型(GMM)为语音成分的 HMM 的使用在语音社区中逐渐传播，并在 1990 年代成为主导范式。原因之一是美国国防部高级研究计划局(ARPA)的鼓励。ARPA 于 1971 年启动了一个为期五年的计划，以建立 1000 个单词，语法受限，几乎没有说话者理解语音的语言(Klatt，1977)，并资助了四个竞争系统，其中卡内基-梅隆大学的 Harpy 系统(Lowerre 1968)使用了贝克基于 HMM 的 DRAGON 系统的简化版本是经过测试的最佳系统。ARPA(然后是 DARPA)资助了许多新的语音研究计划，首先是独立于 1000 个单词的与说话者无关的语音朗读任务，例如“资源管理”(Price 等人，1988)，从《华尔街日报》(WSJ)中识别句子，广播新闻领域(LDC 1998，Graff 1997)(实际新闻广播的转录，包括路边采访等相当困难的段落)以及 Switchboard，CallHome，CallFriend 和 Fisher 领域(Godfrey 等人 1992，Cieri 等人 2004)(朋友或陌生人之间的自然电话交谈)。每个 ARPA 任务都涉及大约每年一次的**审核(bakeoff)**，在此期间，系统相互评估。ARPA 竞赛导致了实验室之间的技术大规模借用，因为很容易看出前一年哪些想法减少了错误，并且竞赛可能是 HMM 范式最终传播的重要因素。

大约在 1990 年左右，基于一些早期对用于音素识别和其他语音任务的神经网络的实验，替代了用于 ASR 的 HMM/GMM 架构的神经网络出现了。这些架构包括时滞神经网络(TDNN)——卷积网络首次用于语音——(Waibel 等人 1989, Lang 等人 1990)、RNNs (Robinson 和 Fallside 1991)、以及**混合(hybrid)**HMM/MLP 架构，其中前馈神经网络被训练为语音分类器，其输出被用作基于 HMM 架构的概率估计(Morgan 和 Bourlard 1990, Bourlard 和 Morgan 1994, Morgan 和 Bourlard 1995)。

尽管混合系统的性能接近标准 HMM / GMM 模型，但问题在于速度：大型混合模型太慢，无法在那个时代的 CPU 上进行训练。例如，最大的混合系统(前馈网络)仅限于 4000 个单元的隐藏层，仅通过几十个

单声道电话就可以产生概率。然而，训练这种模型仍然需要研究小组设计特殊的硬件板来进行矢量处理(Morgan 和 Bourlard，1995)。后来的一项分析研究表明，这种简单的 ASR 前馈 MLP 的性能随着隐藏层数超过 1 个而急剧增加，甚至可以控制参数总数(Maas 等人，2017)。但是，当时的计算资源不足以用于更多层。

在接下来的 20 年里，摩尔定律和 GPU 的兴起使具有多层深度神经网络成为可能。到 2009 年，在较小的任务(如 TIMIT 手机识别)上，性能已经接近传统系统(Mohamed 等人，2009)，到 2012 年，混合系统的性能已经超过了传统的 HMM/GMM 系统(Jaitly 等人 2012, Dahl 等人 2012，特别是)。最初似乎无监督预训练网络使用技术的深层信念网络是重要的,但到了 2013 年,很明显,对于混合 HMM / GMM 前馈网络，重要的是使用大量的数据和足够的层,尽管一些其他组件并提高性能:使用对数 mel 特征代替 MFCCs，使用 dropout，并使用矫正线性单元(Deng 等人 2013, Maas 等人 2013, Dahl 等人 2013)。

同时在 2006 年早期作品提出了 CTC 损失函数(Graves 等人，2006)，到 2012 年 RNN-Transducer 被定义和应用到手机识别(Graves 2012，Graves 等人 2013)，然后应用到端到端语音识别重新计分(Graves 和 Jaitly，2014)，最后应用到识别(Mass 等人，2015)，它们都采用了专业化的波束搜索等先进技术(Hannun 等人，2014)。(我们在本章中对 CTC 的描述引用了 Hannun(2017)，我们鼓励有兴趣的读者继续阅读)。

在 Chan 等人(2016)的 Listen Attend and Spell system 和 Chorowski 等人(2014)和 Bahdanau 等人(2016)的基于注意力的编码器-解码器架构中，两组人几乎同时将编码器-解码器架构应用于语音。到 2018 年，Transformers 被包括在这种编码器-解码器架构中。Karita 等人(2019)对 ASR、TTS 和语音到语音翻译的编码器架构中的 RNNs 和 Transformers 进行了很好的比较。

流行的语音处理工具包包括 Kaldi(Povey 等人，2011)和 ESPnet(Watanabe 等人 2018, Hayashi 等人 2020)。

TTS

如本章开头所述，语音合成是语音和语言处理的最早领域之一。18 世纪出现了许多发音过程的物理模型，包括上述的 von Kempelen 模型，以及哥本哈根使用风琴管吹奏的 Kratzenstein 的 1773 年元音模型。

20 世纪 50 年代早期出现了三种波形合成的早期范式：共振峰合成、拼接合成和发音合成。

第一个范式，共振峰合成。现代的编译码系统是共振峰合成器的远祖。共振峰合成器最初是受到通过生成人工频谱来模仿人类语言的尝试的启发。Haskins 实验室的模式回放机通过在移动的透明带上绘制频谱图案，并利用反射率滤掉波形的谐波来产生声波(Cooper 等人，1951)；其他早期的共振峰合成器包括 Lawrence(1953)和 Fant(1951)。也许最著名的共振峰合成器是 Klatt 共振峰合成器及其后继系统，包括 MITalk 系统(Allen 等人，1987)和用于数字设备公司的 DECtalk (Klatt，1982)的 Klattalk 软件。详情见 Klatt(1975)。

第二个范式，拼接合成。似乎是由 Harris(1953)在贝尔实验室提出的;他把对应电话的磁带片段拼接在一起。不久之后，Peterson 等人(1958)提出了一个基于双音素的理论模型，包括一个数据库，每个双音素具有不同韵律的多个副本，每个副本都标有 F0、重音、持续时间等韵律特征，并使用基于 F0 和相邻单元之间共振峰距离的连接代价。但这种双音素合成模型直到几十年后才真正实现(Dixon 和 Maxey 1968, Olive 1977)。20 世纪 80 年代和 90 年代发明了单位选择综合，基于较大的非均匀长度单位和使用目标成本，(Sagisaka 1988, Sagisaka 等人 1992, Hunt 和 Black 1996, Black 和 Taylor 1994, Syrdal 等人 2000)。

第三个范式，发音合成。发音合成器试图通过将声腔的物理特性建模为一个开放的管道来合成语音。代表性的模型包括 Stevens 等人(1953)、Flanagan 等人(1975)和 Fant(1986)。详见 Klatt(1975)和 Flanagan(1972)。

大多数早期的 TTS 系统使用音素作为输入；TTS 的文本分析组件的发展要晚一些，借助于自然语言处理。的确，第一个真正的文本语音转换系统似乎是 Umeda 和 Teranishi 的系统(Umeda 等人 1968，Teranishi 和 Umeda1968，Umeda1976)，它包括一个分配韵律边界、重读和重音的解析器。

26.10. 练习

26.1 Analyze each of the errors in the incorrectly recognized transcription of “um the phone is I left the. . . ” on page 562. For each one, give your best guess as to whether you think it is caused by a problem in signal processing, pronun ciation modeling, lexicon size, language model, or pruning in the decoding search.

27. 附录 A：隐马尔可夫模型(HMM)

第 8 章介绍了隐马尔可夫模型并将其应用于词性标注。词性标注是一项完全监督的学习任务，因为我们有一个用正确词类标记标签的词库。但是许多应用程序没有标签数据。所以在本章中，我们介绍了 HMM 的全套算法，包括 HMM 的关键无监督学习算法，前向-后向算法。我们将重复第 8 章中的一些文本，以飨那些想要在单独一章中讲述整个故事的读者。

27.1. 马尔可夫链

HMM 是基于**马尔可夫链(Markov chain)**的扩充。马尔可夫链是一个模型，它告诉我们一些关于随机变量、状态的序列概率，每个概率都可以从某个集合中取值。这些集合可以是单词、标记或代表任何东西的符号，比如天气。马尔可夫链给出了一个非常有力的假设：如果我们想要预测序列中的未来，最重要的是当前的状态。除非通过当前状态，否则当前状态之前的状态对未来没有影响。这就好像为了预测明天的天气，你可以查看今天的天气，但却不允许查看昨天的天气。

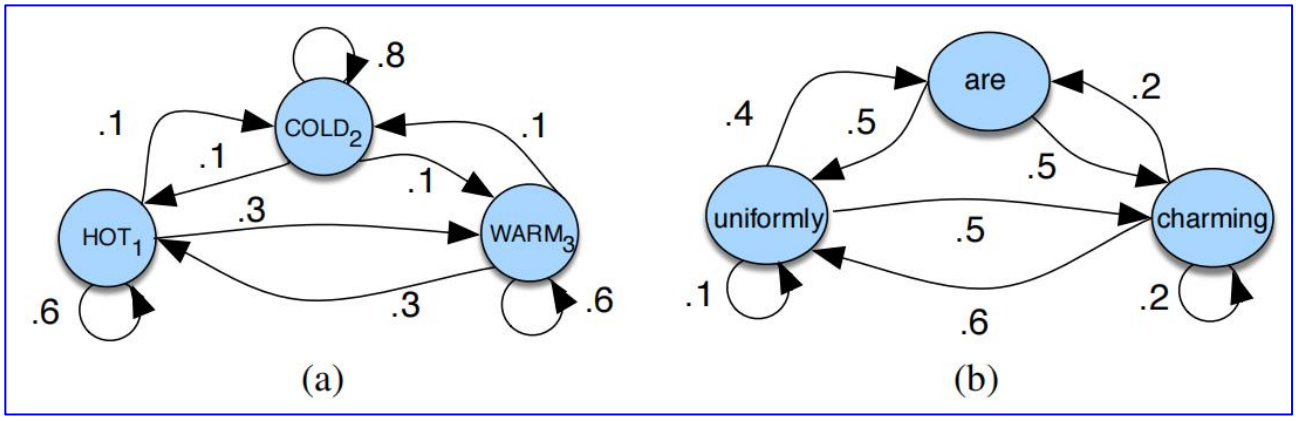


图 A.1：天气的马尔可夫链(a)和单词的马尔可夫链(b)，显示状态和转移。需要一个起始分布 π；对于(a)，设置 π=[0.1,0.7,0.2]意味着从状态 2(COLD)开始的概率是 0.7，从状态 1(TOT)开始的概率是 0.1，等等。

更正式地说，考虑一系列状态变量 $q_1,q_2,\ldots,q_i$。马尔可夫模型体现了这个序列的概率的**马尔可夫假设(Markov assumption)**：当预测未来时，过去不重要，只有现在重要。

$$\text{Markov Assumption：} P(q_i=a|q_1\ldots q_{i-1}) = P(q_i=a|q_{i-1}) \quad \text{(A.1)}$$

图 A.1a 显示了为一系列天气事件分配概率的马尔可夫链，其中词汇由 HOT、COLD 和 WARM 组成。状态表示为图中的节点，过渡及其概率表示为边。转移是概率：离开给定状态的弧线的值总和必须为 1。图 A.1b 显示了为单词序列 $w_1\ldots w_n$ 分配概率的马尔可夫链。这个马尔可夫链应该很熟悉；事实上，它代表了一个二元语法语言模型，每条边表示概率 $p(w_i|w_j)$! 给定图 A.1 中的两个模型，我们可以从我们的词汇表中给任何序列分配一个概率。

$Q=q_1q_2\ldots q_N$	N 个状态的集合
$A=a_{11}a_{12}\ldots a_{n1}\ldots a_{nn}$	转移概率矩阵 A，每个 a_{ij} 代表从状态 i 移动到状态 j 的概率，满足：$\sum_{j=1}^{n} a_{ij}=1 \ \forall i$
$\pi=\pi_1,\pi_2,\ldots,\pi_N$	状态上的初始概率分布。π_i 是马尔可夫链从状态 i 开始的概率。一些状态也许$\pi_j=0$，这意味着它们不能是初始状态。此外，$\sum_{i=1}^{n} \pi_i=1$。

形式上，马尔可夫链由以下组成部分表示：

在你继续之前，使用图 A.1a 中的样本概率(π=[0.1，0.7，0.2])来计算下列每个序列的概率：

(A.2) hot hot hot hot

(A.3) cold hot cold hot

关于图 A.1a 中编码的真实世界天气事实，这些概率的差异告诉您什么？

27.2. 隐马尔可夫模型

当我们需要计算一系列可观察事件的概率时，马尔可夫链是有用的。然而，在许多情况下，我们感兴趣的事件是**隐藏的(hidden)**：我们不能直接观察它们。例如，我们通常不会在文本中观察到隐藏的词类标记。相反，我们看到单词，必须从单词序列推断标记。我们称这些标记为隐藏的，因为它们不被观察到。

隐马尔可夫模型(HMM)允许我们讨论观察到的事件(如我们在输入中看到的单词)和隐藏事件(如词性标签)，我们认为它们是概率模型中的因果因素。HMM 由以下组件指定：

$Q=q_1q_2...q_N$	N 个状态的集合
$A=a_{11}...a_{ij}...a_{nn}$	转移概率矩阵 A，每个 a_{ij} 代表从状态 i 移动到状态 j 的概率，满足: $\sum_{j=1}^{n} a_{ij}=1 \quad \forall i$
$O=o_1o_2...o_T$	一系列 T 个观察，每个观察都来自词汇表 $V = v_1, v_2,..., v_V$
$B=b_i(o_t)$	一系列观察似然度，也称为发射概率，每个都表示从状态 i 生成观察 o_t 的概率
$\pi=\pi_1,\pi_2,...,\pi_N$	状态上的初始概率分布。π_i 是马尔可夫链从状态 i 开始的概率。一些状态可能有$\pi_j= 0$，这意味着它们不可能是初始状态。此外，$\sum_{i=1}^{n} \pi_i=1$。

一阶隐马尔可夫模型实例化了两个简化假设。

第一，与一阶马尔可夫链一样，某一状态的概率只取决于前一状态：

$$\text{Markov Assumption：}\quad P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1}) \tag{A.4}$$

第二，输出观察 o_i 的概率只取决于产生观察 q_i 的状态，而不取决于任何其他状态或任何其他观察：

$$\text{Output Independence：}\quad P(o_i|q_1 ...q_i ,...,q_T ,o_1,...,o_i ,...,o_T) = P(o_i|q_i) \tag{A.5}$$

为了举例说明这些模型，我们将使用 Jason Eisner(2002)发明的一个任务。假设你是一位气候学家，在 2799 年研究全球变暖的历史。你找不到马里兰州巴尔的摩 2020 年夏天的天气记录，但你可以找到 Eisner 的日记，上面列出了他在那个夏天每天吃了多少冰淇淋。我们的目标是利用这些观察结果来估计每天的温度。我们将简化天气任务，假设只有两种天气：冷(C)和热(H)。因此，Eisner 的任务如下：

给定一个观察序列 O(每一个都是整数，代表在某一天吃冰淇淋的数量)，找到天气状态的“隐藏”序列 Q(H 或 C)，导致 Jason 吃了冰淇淋。

图 A.2 是冰淇淋任务的样例 HMM。这两种隐藏状态(H 和 C)对应的是炎热和寒冷的天气，而观察值(由字母表 O={1,2,3}引出)对应的是 JASON 在某一天吃的冰淇淋数量。

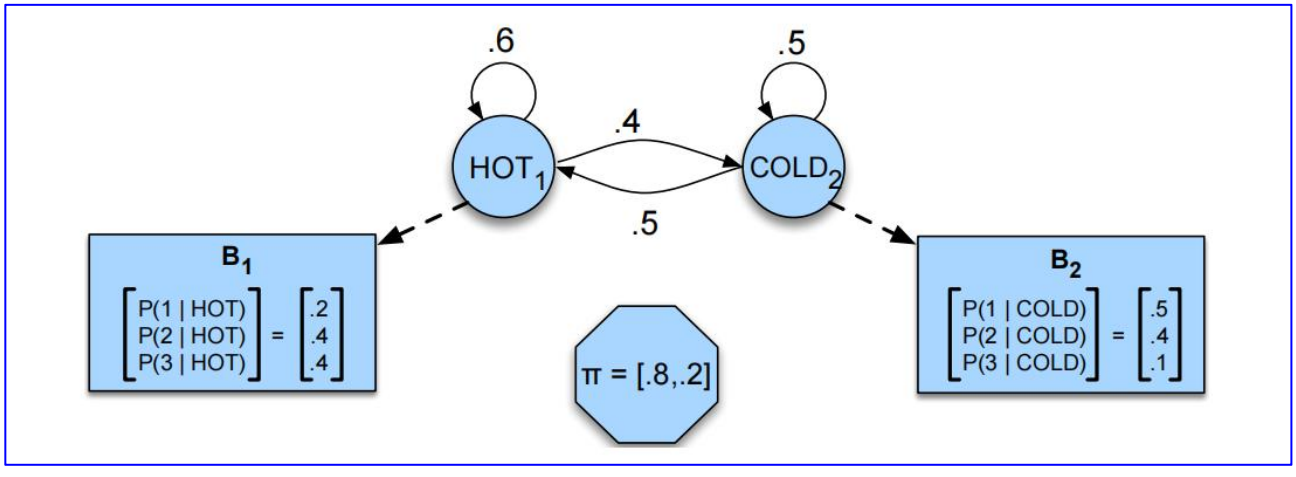


图 A.2：将 Jason 吃冰淇淋的数量(观察值)与天气(隐含变量 H 或 C)相关联的隐马尔可夫模型

Rabiner(1989)在 20 世纪 60 年代 Jack Ferguson 的教程的基础上，提出了隐马尔可夫模型应具有三个基本问题的观点：

问题 1(似然)：给定一个观察序列 O 和 HMM $\lambda = (A,B)$，确定似然度 $P(O|\lambda)$。
问题 2(解码)：给定一个观察序列 O 和 HMM $\lambda = (A,B)$，发现最佳隐藏状态序列 Q。
问题 3(学习)：给定一个观察序列 O 和 HMM 中的状态集，学习 HMM 参数 A 和 B。

我们已经在第 8 章看到了问题 2 的例子。在接下来的两节中，我们将介绍用于解决问题 1 和 3 的 Forward 和 Forward- backward 算法，并提供关于问题 2 的更多信息。

27.3. 似然计算：前向算法

我们的第一个问题是计算一个特定观察序列的似然度。例如，给定图 A.2 中吃冰淇淋的 HMM，序列 3 1 3 的概率是多少?更正式地：

计算似然：给定 HMM λ = (A,B)和观察序列 O，确定似然度 P(O|λ)。

对于表面观察与隐藏事件相同的马尔可夫链，我们可以通过跟踪标签为 3 1 3 的状态并沿弧线乘以概率来计算 3 1 3 的概率。对于隐马尔可夫模型，事情并没有那么简单。我们想确定像 3 1 3 这样的冰淇淋观察序列的概率，但我们不知道隐藏状态序列是什么！

让我们从一个稍微简单的情况开始。假设我们已经知道天气并且想要预测 Jason 会吃多少冰淇淋。这是许多 HMM 任务的有用部分。对于给定的隐藏状态序列（例如，hot hot cold），我们可以轻松计算 3 1 3 的输出似然度。

让我们看看怎么做。首先，回想一下，对于隐马尔可夫模型，每个隐状态只产生一个观察。因此，隐藏状态序列和观察序列具有相同的长度。[1]

鉴于此一对一映射和公式 A.4 中表达的马尔可夫假设，对于特定的隐藏状态序列 $Q = q_0,q_1,q_2,...,q_T$ 和观察序列 $O = o_1,o_2,... ,o_T$，观察序列的似然度为：

$$P(O|Q)=\prod_{i=1}^{T} P(o_i|q_i) \tag{A.6}$$

从一个可能的隐藏状态序列 hot hot cold 计算我们的冰淇淋观察 3 1 3 的前向概率如公式 A.7 所示。图 A.3 显示了这种计算的图形表示。

$$P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot})\times P(1|\text{hot})\times P(3|\text{cold}) \tag{A.7}$$

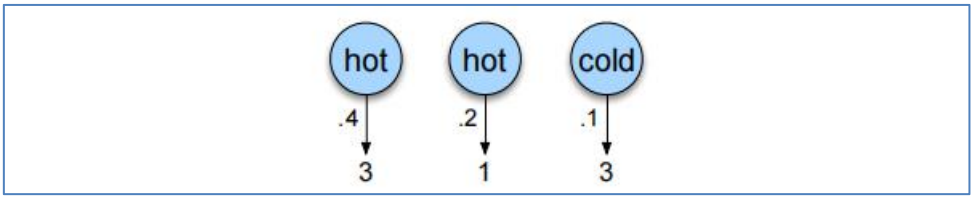


图 A.3：给定隐藏状态序列 hot hot cold 的冰淇淋事件 3 1 3 的观察似然度计算

但当然，我们实际上并不知道隐藏状态（天气）序列是什么。我们需要计算冰淇淋事件 3 1 3 的概率，而不是通过对所有可能的天气序列求和，并按它们的概率加权。首先，让我们计算处于特定天气序列 Q 并生成特定冰淇淋事件序列 O 的联合概率。一般来说，这是：

$$P(O，Q)=P(O|Q)\times P(Q)=\prod_{i=1}^{T} P(o_i|q_i)\times\prod_{i=1}^{T} P(q_i|q_{i-1}) \tag{A.8}$$

我们的冰淇淋观察 3 1 3 和一个可能的隐藏状态序列 hot hot cold 的联合概率的计算如公式 A.9 所示。图 A.4 显示了这种计算的图形表示。

$$P(3\ 1\ 3,\text{hot hot cold}) = P(\text{hot}|\text{start})\times P(\text{hot}|\text{hot})\times P(\text{cold}|\text{hot})\times P(3|\text{hot})\times P(1|\text{hot})\times P(3|\text{cold}) \tag{A.9}$$

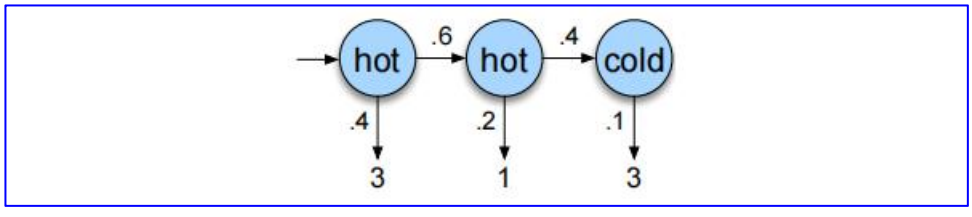


图 A.4 冰淇淋事件 3 1 3 和隐藏状态序列 hot hot cold 的联合概率的计算

现在我们知道如何计算具有特定隐藏状态序列的观察的联合概率，我们可以通过对所有可能的隐藏状态序列求和来计算观察的总概率：

$$P(O)=\sum_{Q} P(O,Q)=\sum_{Q} P(O|Q)P(Q) \tag{A.10}$$

对于我们的特殊情况，我们将总结八个 3-事件序列 cold cold cold，cold cold hot，... ，即：

$$P(3\ 1\ 3) = P(3\ 1\ 3,\ \text{cold cold cold}) + P(3\ 1\ 3,\ \text{cold cold hot}) + P(3\ 1\ 3,\text{hot hot cold}) + ...$$

[1] 在称为**分段 HMM**（在语音识别中）或 **semi-HMM**（在文本处理中）的 HMM 变体中，隐藏状态序列的长度和观察序列的长度之间的这种一对一映射不成立。

对于具有 N 个隐藏状态和 T 个观察的观察序列的 HMM，有 N^T 个可能的隐藏序列。对于实际任务，其中 N 和 T 都很大，N^T 是一个非常大的数字，因此我们无法通过为每个隐藏状态序列计算单独的观察似然然后将它们相加来计算总观察似然。

我们没有使用这种极端指数算法，而是使用称为前向算法(forward algorithm)的高效 $O(N^2T)$算法。前向算法是一种动态规划算法，即在建立观察序列的概率时使用表格来存储中间值的算法。 前向算法通过对可以生成观察序列的所有可能隐藏状态路径的概率求和来计算观察概率，但它通过将这些路径中的每一个隐式折叠成单个前向**网格(trellis)**来有效地做到这一点。

图 A.5 显示了在给定隐藏状态序列 hot hot cold 的情况下计算 3 1 3 似然度的前向网格示例。

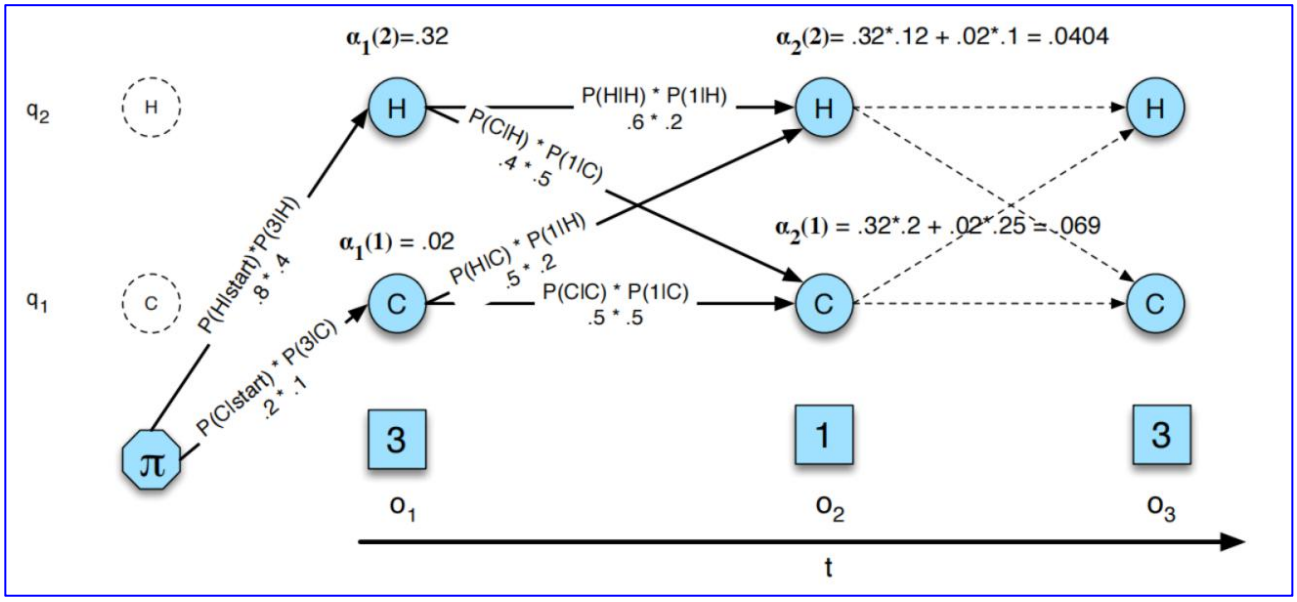


图 A.5：用于计算冰淇淋事件总观察似然的前向网格 3 1 3. 隐藏状态在圆圈中，观察在正方形中。该图显示了两个时间步的两个状态的 $\alpha_t(j)$ 计算。每个单元格中的计算遵循公式 A.12。在每个单元格中表示的结果概率是公式 A.11。

给定自动机λ，前向算法网格$\alpha_t(j)$的每个单元格表示在看到前 t 个观察值后处于状态 j 的概率。每个单元格$\alpha_t(j)$的值是通过对可能导致我们到达该单元格的每条路径的概率求和来计算的。形式上，每个单元格表示以下概率：

$$\alpha_t(j) = P(o_1,o_2...o_t,q_t = j|\lambda) \tag{A.11}$$

其中，$q_t=j$ 表示“状态序列中的第 t 个状态是状态 j”。我们通过对通向当前单元格的所有路径的扩展求和来计算这个概率$\alpha_t(j)$。对于时间 t 的给定状态 q_j，值$\alpha_t(j)$计算为：

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(o_t) \tag{A.12}$$

在公式 A.12 中乘以扩展先前路径以计算时间 t 的前向概率的三个因素是：

$\alpha_{t-1}(i)$：	前一时间步的前向路径概率；
a_{ij}：	从前一个状态 q_i 到当前状态 q_j 的转移概率；
$b_j(o_t)$ ：	给定当前状态 j 的观察符号 o_t 的状态观察似然度。

考虑图 A.5 中$\alpha_2(2)$的计算，在状态 2 中处于时间步 2 的前向概率产生了部分观察 3 1。我们通过两条路径从时间步 1 扩展α概率来计算，每个扩展由上述三个因子组成：

$\alpha_1(1)\times P(H|C)\times P(1|H)$和$\alpha_1(2)\times P(H|H)\times P(1|H)$。

图 A.6 显示了此归纳步骤的另一种可视化，用于计算网格的一个新单元格中的值。

我们给出了前向算法的两个正式定义：图 A.7 中的伪代码和这里定义递归的声明。

1.初始化： $\alpha_1(j) = \pi_j b_j(o_1)$； $1 \le j \le N$

2.递归： $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(o_t)$； $1 \le j \le N, 1 < t \le T$

3.终止： $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

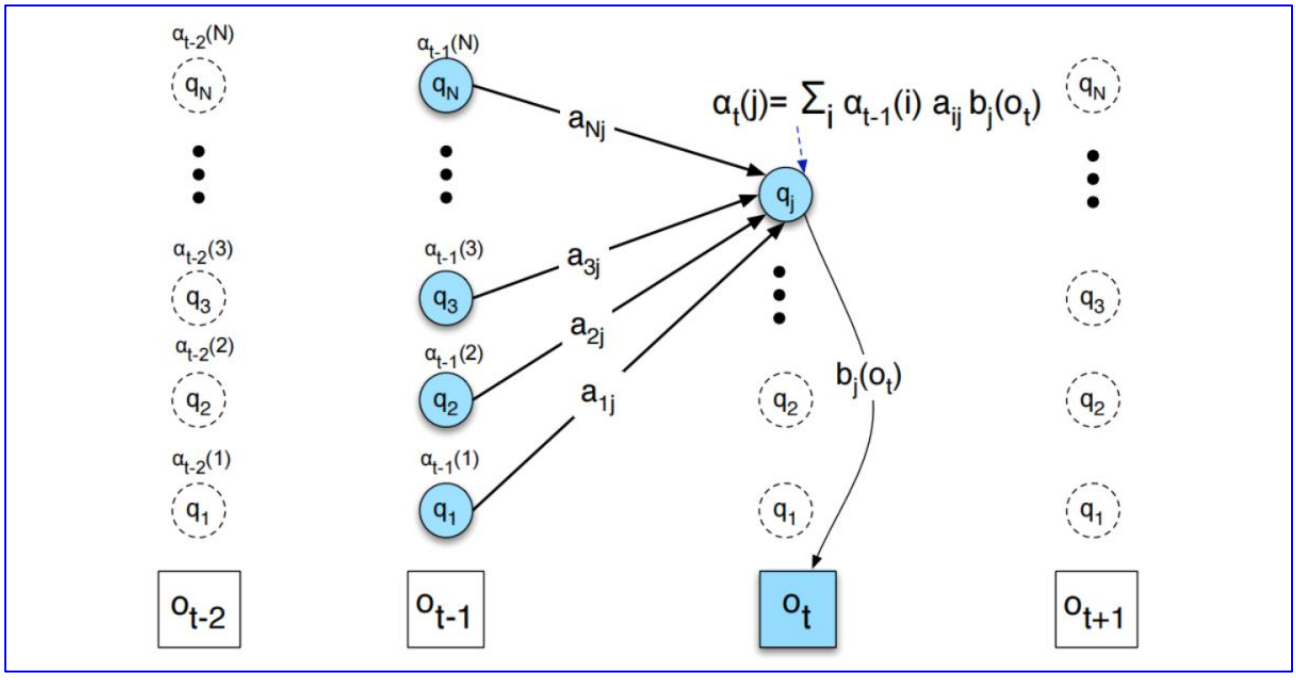


图 A.6：通过将所有先前的值 α_{t-1} 相加，由它们的转移概率 a 加权，并乘以观察概率 $b_i(o_t)$，将网格中单个元素 $\alpha_t(i)$ 的计算可视化。对于 HMM 的许多应用，许多转移概率为 0，因此并非所有先前的状态都会对当前状态的前向概率有贡献。隐藏状态在圆圈中，观察在正方形中。阴影节点包含在 $\alpha_t(i)$ 的概率计算中。

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward*[N,T]
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step
return *forwardprob*

图 A.7：前向算法伪代码。其中 forward[s,t]表示 $\alpha_t(s)$。

27.4. 解码：维特比(Viterbi)算法

对于包含隐藏变量的任何模型（例如 HMM），确定哪个变量序列是某些观察序列的潜在来源的任务称为解码任务。在冰淇淋问题域中，给定冰淇淋观察序列 3 1 3 和 HMM，解码器的任务是找到最佳隐藏天气序列（HHH）。更正式地说：

解码：给定 HMM $\lambda = (A, B)$ 和观察序列 $O = o_1, o_2, \ldots, o_T$ 作为输入，找到最可能的状态序列 $Q = q_1 q_2 q_3 \ldots q_T$ 。

我们可能会提出寻找最佳序列的方法如下：对于每个可能的隐藏状态序列（HHH、HHC、HCH 等），我们可以运行前向算法并计算给定该隐藏状态序列的观察序列的似然度。然后我们可以选择具有最大观察似然度的隐藏状态序列。但是，从上一节可以清楚地看出，我们不能这样做，因为状态序列的数量呈指数级增长。相反，HMM 最常见的解码算法是 Viterbi 算法。与前向算法一样，Viterbi 是一种使用动态规划网格的动态规划。Viterbi 也非常类似于另一个动态规划变体，即第 2 章的最小编辑距离算法。

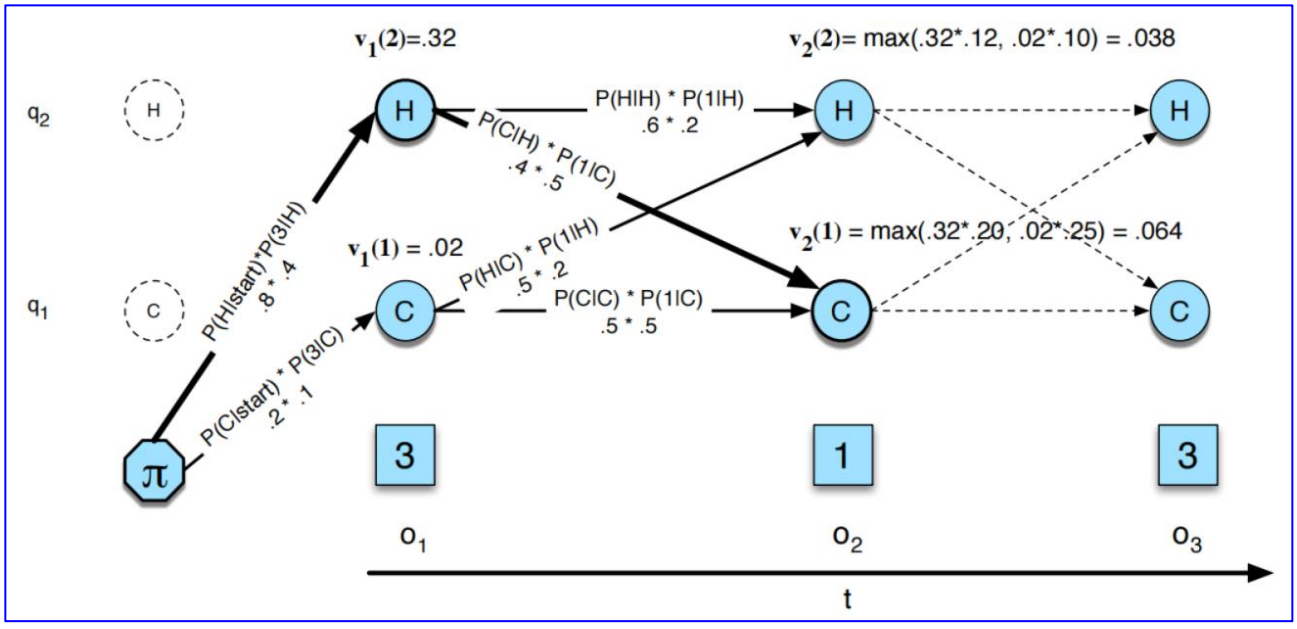


图 A.8：用于计算冰淇淋吃事件 3 1 3 隐藏状态空间的最佳路径的 Viterbi 网格。隐藏状态在圆圈中，观察在正方形中。白色（未填充）圆圈表示非法转移。该图显示了两个时间步的两个状态的 $v_t(j)$ 计算。每个单元格中的计算遵循公式 A.14. 在每个单元格中表示的结果概率是公式 A.13.

图 A.8 显示了 Viterbi 网格的示例，用于计算观察序列 3 1 3 的最佳隐藏状态序列。其思想是从左到右处理观察序列，填充网格。网格的每个单元格 vt(j)表示在给定自动机λ的情况下，HMM 在看到前 t 个观察值并通过最可能的状态序列 $q_1,...,q_{t-1}$ 后处于状态 j 的概率。每个单元格 $v_t(j)$的值是通过递归地采用最可能引导我们到达该单元格的路径来计算的。形式上，每个单元格表示概率：

$$v_t(j) = \max_{q_1 \cdots q_{t-1}} P(q_1...q_{t-1}, o_1, o_2 ...o_t, q_t = j|\lambda) \tag{A.13}$$

请注意，我们通过对所有可能的先前状态序列 $\max_{q_1 \cdots q_{t-1}}$ 取最大值来表示最可能的路径。与其他动态规划算法一样，维特比递归地填充每个单元格。鉴于我们已经计算了在时间 t-1 每个状态的概率，我们通过获取通向当前单元的路径的最可能扩展来计算 Viterbi 概率。对于时间 t 的给定状态 q_j，值 $v_t(j)$计算为：

$$v_t(j) = \max_{i=1,\cdots,N} v_{t-1}(i) a_{ij} b_j(o_t) \tag{A.14}$$

在公式 A.14 中乘以扩展先前路径以计算时间 t 的 Viterb 概率的三个因子是：

$v_{t-1}(i)$：	前一时间步的前 Viterbi 路径概率；
a_{ij}：	从前一个状态 q_i 到当前状态 q_j 的转移概率；
$b_j(o_t)$：	给定当前状态 j，观察符号 o_t 的状态观察似然度。

图 A.9 显示了 Viterbi 算法的伪代码。请注意，Viterbi 算法与前向算法一致，不同之处在于它取先前路径概率的最大值，而前向算法取总和。还要注意，Viterbi 算法有一个前向算法没有的组件：反向指针。原因是前向算法需要产生一个观察似然度，而 Viterbi 算法则必须产生一个概率以及最可能的状态序列。我们通过跟踪导致每个状态的隐藏状态的路径来计算这个最佳状态序列，如图 A.10 所示，然后在最后回溯到开始的最佳路径（**维特比回溯（Viterbi backtrace）**）。

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi*[N,T]
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

图 A.9：寻找最优隐藏状态序列的维特比算法。给定一个观察序列和一个 HMM $\lambda=(A,B)$，算法返回通过 HMM 的状态路径，该 HMM 为观察序列分配最大似然度。

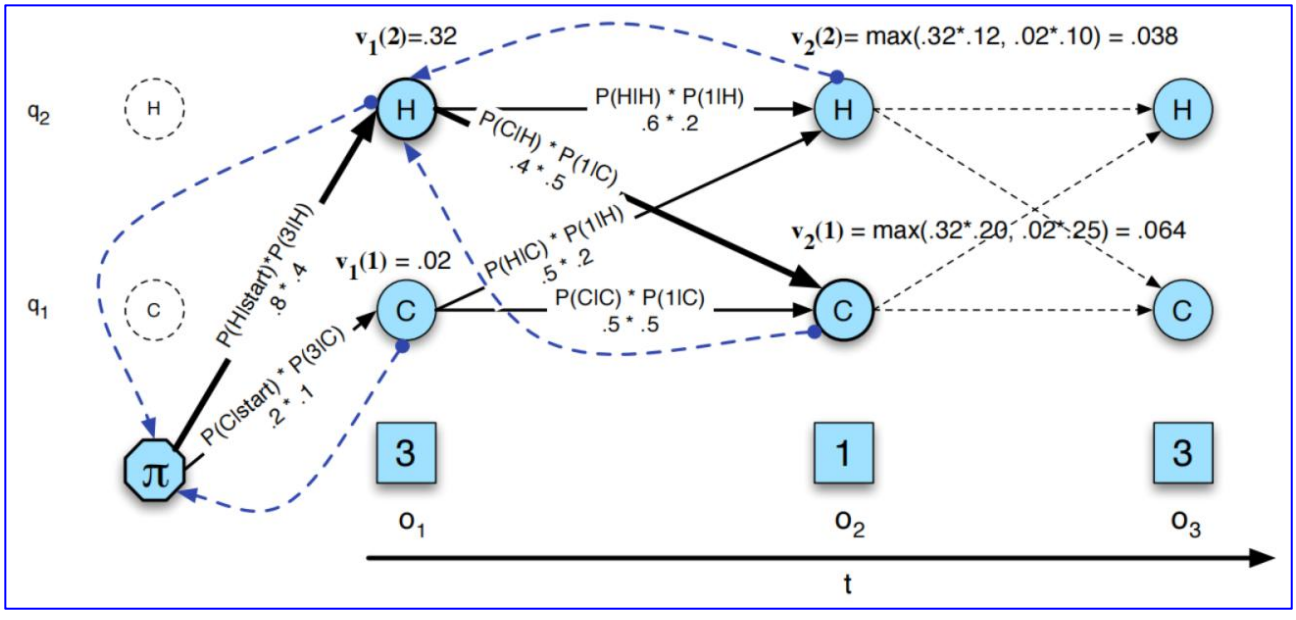


图 A.10：维特比回溯。当我们将每条路径扩展到一个新的状态帐户(account)以进行下一次观察时，我们会保留一个指向引导我们到达该状态的最佳路径的反向指针（用虚线显示）。

最后，我们可以给出 Viterbi 递归的正式定义如下：

1. 初始化： $v_1(j) = \pi_j b_j(o_1)$； $1 \le j \le N$
  $b_{t1}(j) = 0$ ； $1 \le j \le N$

2. 递归： $v_t(j) = \max_{i=1,\cdots,N} v_{t-1}(i) a_{ij} b_j(o_t)$； $1 \le j \le N,\ 1 < t \le T$

  $bt_t(j) = \operatorname{argmax}_{i=1,\cdots,N} v_{t-1}(i) a_{ij} b_j(o_t)$； $1 \le j \le N,\ 1 < t \le T$

3. 终止： The best score: $P* = \max_{i=1,\cdots,N} v_T(i)$

  The start of backtrace: $q_T* = \operatorname{argmax}_{i=1,\cdots,N} v_T(i)$

27.5. HMM 训练：前向-后向算法

我们转向 HMM 的第三个问题：学习 HMM 的参数，即 A 和 B 矩阵。正式地：

学习：给定一个观察序列 O 和 HMM 中可能的状态集，学习 HMM 参数 A 和 B。

这种学习算法的输入将是未标签的观察序列 O 和潜在隐藏状态 Q 的词汇表。因此，对于冰淇淋任务，我们将开始于观察序列 O={1,3,2, ..,}、隐藏状态集 H 和 C。

HMM 训练的标准算法是**前向-后向(forward-backward)**或 **Baum-Welch** 算法（Baum，1972），它是**期望最大化（Expectation-Maximization，EM）**算法（Dempster 等人，1977）的特例。该算法将让我们训练 HMM 的转移概率 A 和发射概率 B。EM 是一种迭代算法，它计算概率的初始估计值，然后使用这些估计值来计算更好的估计值，依此类推，迭代地提高它学习的概率。

让我们首先考虑训练完全可见的马尔可夫模型的更简单的情况，我们知道每天的温度和冰淇淋数量。也就是说，想象一下我们看到以下一组输入观察并神奇地知道对齐的隐藏状态序列：

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

这将很容易让我们仅通过训练数据的最大似然估计值来计算 HMM 参数。首先，我们可以从 3 个初始隐藏状态的计数中计算π：

$$\pi_h = 1/3 \qquad \pi_c = 2/3$$

接下来我们可以忽略最终的隐藏状态，直接从转移中计算 A 矩阵：

$$p(hot|hot) = 2/3 \qquad p(cold|hot) = 1/3$$
$$p(cold|cold) = 2/3 \qquad p(hot|cold) = 1/3$$

和 B 矩阵：

$$P(1|hot) = 0/4 = 0 \qquad p(1|cold) = 3/5 = .6$$
$$P(2|hot) = 1/4 = .25 \qquad p(2|cold = 2/5 = .4$$
$$P(3|hot) = 3/4 = .75 \qquad p(3|cold) = 0$$

对于真实的 HMM，我们不能直接从观察序列中计算这些计数，因为我们不知道对于给定的输入，机器采用了哪条状态路径。 例如，假设我没有告诉您第 2 天的温度，您必须猜测它，但是您（神奇地）有上述概率，以及其他几天的温度。您可以对所有其他概率进行一些贝叶斯计算，以估计缺失的那一天的可能温度，并使用它们来获得第 2 天温度的预期计数。

但真正的问题更难：我们不知道处于任何隐藏状态的计数！！Baum-Welch 算法通过迭代估计计数来解决这个问题。我们将从对转移和观察概率的估计开始，然后使用这些估计的概率来推导出越来越好的概率。我们将通过计算观察的前向概率，然后在所有不同的路径之间划分这个概率质量(mass)来实现这个前向概率。

为了理解算法，我们需要定义一个与前向概率相关的有用概率，称为后向概率。后向概率β是从时间 t+1 到结束看到观察值的概率，假设我们在时间 t 处于状态 i（并且给定自动机 λ）：

$$\beta_t(i) = P(o_{t+1},o_{t+2} \ldots o_T | q_t = i, \lambda) \tag{A.15}$$

它以类似于前向算法的方式进行归纳计算。

1. **初始化：**

$$\beta_T(i) = 1; \quad 1 \le i \le N$$

2. **递归：**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j); \qquad 1 \le i \le N, 1 \le t < T$$

3. **终止：**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$$

图 A.11 说明了反向归纳步骤。

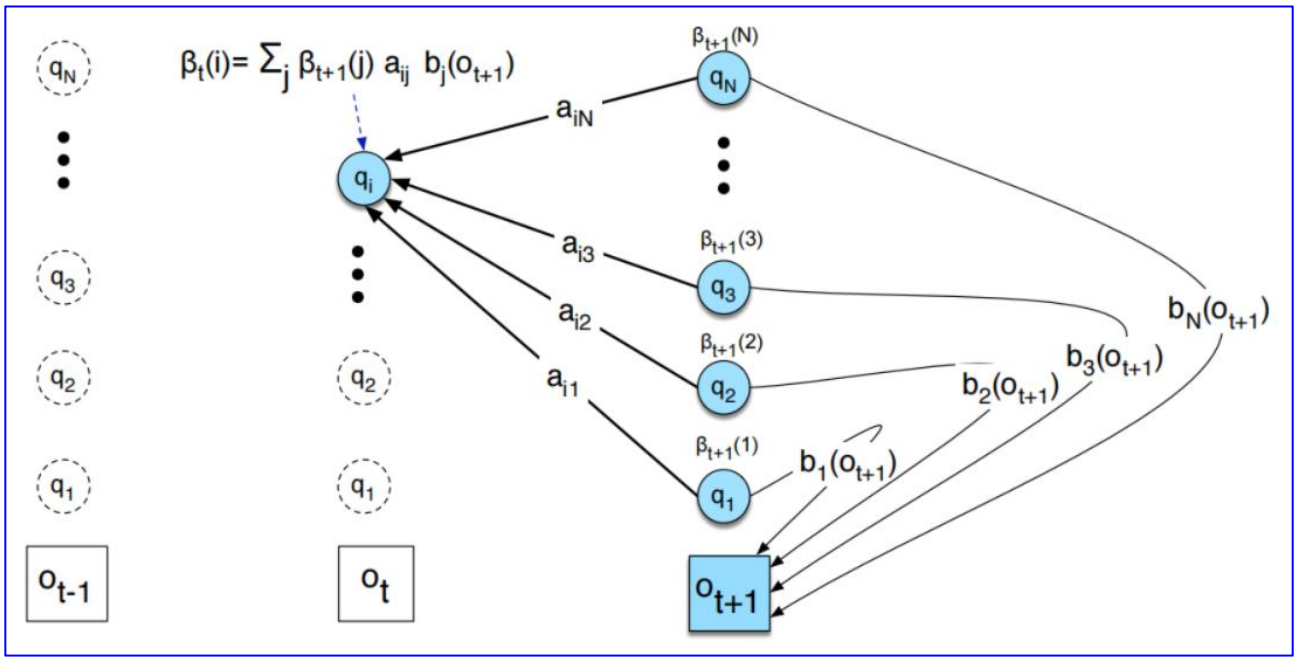


图 A.11：$\beta_t(i)$的计算方法是将所有连续值$\beta_{t+1}(j)$相加，这些值由它们的转移概率a_{ij}和它们的观察概率$b_j(o_{t+1})$加权。

我们现在准备好看看前向和后向概率如何帮助计算来自观察序列的转移概率 a_{ij} 和观察概率 $b_i(o_t)$，即使通过模型所采取的实际路径是隐藏的。

让我们首先看看如何通过简单最大似然估计的变体来估计$\hat{a}_{ij}$：

$$\hat{a}_{ij} = \frac{\textit{expected number of transitions from state } i \textit{ to state } j}{\textit{expected number of transitions from state } i} \tag{A.15}$$

我们如何计算上式的分子？这里有个直觉。假设我们对在观察序列中的特定时间点 t 进行给定转移 i→j 的概率有一些估计。如果我们知道每个特定时间 t 的这个概率，我们就可以对所有时间 t 求和来估计转移 i→j 的总计数。

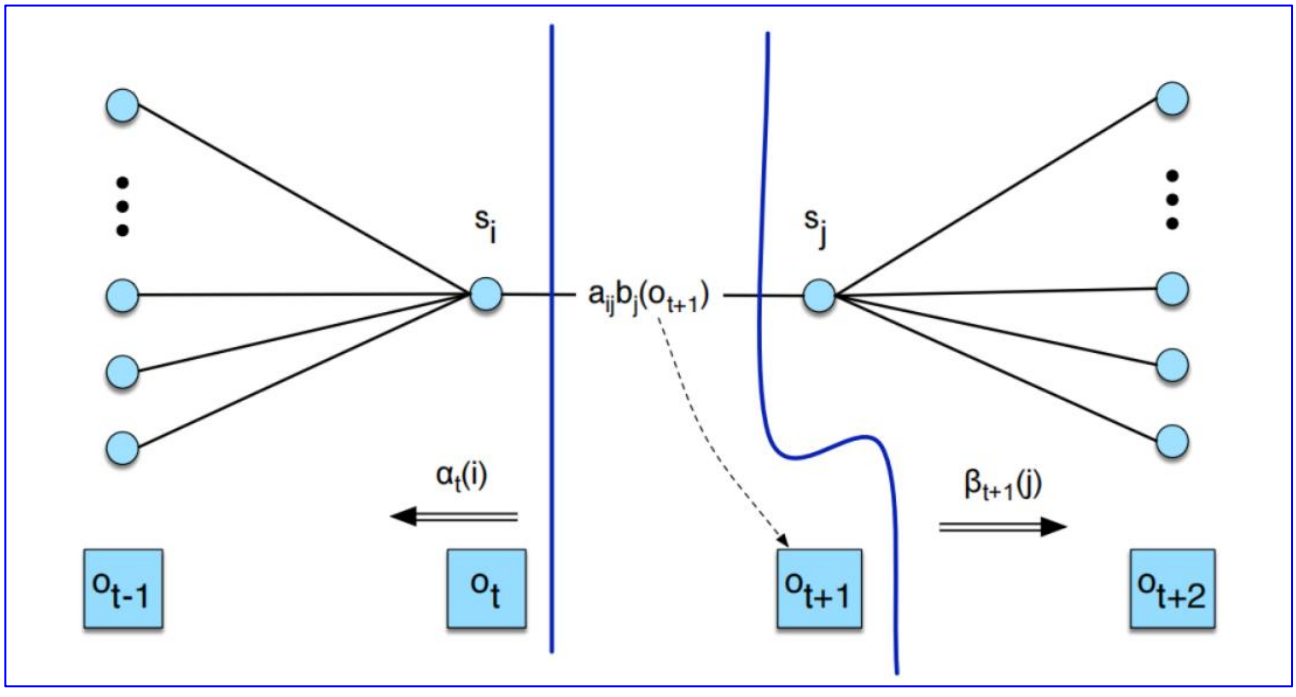


图 A.12：计算在时间 t 处于状态 i 和在时间 t+1 处于状态 j 的联合概率。图中显示了各种概率，这些概率需要组合以产生$P(q_{t=i}, q_{t+1=j}, O|\lambda)$：α 和 β 概率、转移概率$a_{ij}$和观察概率$b_j(o_{t+1})$。来自 Rabiner(1989) ©1989 IEEE。

更正式地，将概率ξ_t定义为在时间 t 处于状态 i 和在时间 t+1 处于状态 j 的概率，给定观察序列和模型：

$$\xi_t(i, j) = P(q_t=i, q_{t+1}=j|O, \lambda) \tag{A.17}$$

为了计算ξ_t，我们首先计算一个与ξ_t相似的概率，但不同之处在于包括观察的概率；请注意公式 A.17 中 O 的不同调节：

$$\text{not-quite-}\xi_t(i, j) = P(q_t=i, q_{t+1}=j, O|\lambda) \tag{A.18}$$

图 A.12 显示了计算 not-quite-ξ_t的各种概率：所讨论的弧的转移概率、弧前的α概率、弧后的β概率、弧后的符号的观察概率。这四个相乘以产生 not-quite-ξ_t如下：

$$\text{not-quite-}\xi_t(i, j) =\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j) \tag{A.19}$$

为了 not-quite-ξ_t计算ξ_t，我们遵循概率定律并除以 P(O|λ)，因为：

$$P(X|Y, Z) = \frac{P(X,Y|Z)}{P(Y|Z)} \tag{A.20}$$

给定模型的观察概率只是整个语段的前向概率（或者，整个语段的后向概率）：

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j) \tag{A.21}$$

所以，ξ_t的最终方程是：

$$\xi_t(i,j)=\frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)} \qquad \hat{b} \tag{A.22}$$

从状态 i 到状态 j 的预期转移次数是ξ的所有 t 的总和。对于在公式 A.16 中对 a_{ij} 的估计，只需要一件事：从状态 i 的预期转移总数。我们可以通过对状态 i 之外的所有转移求和来得到这一点。这是$\hat{a}_{ij}$的最终公式：

$$\hat{a}_{ij}=\frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N}\xi_t(i,k)} \tag{A.23}$$

还需一个公式来重新计算观察概率。在给定状态 j 的情况下，观察词汇 V 中给定符号 v_k 的概率计算：

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} \tag{A.24}$$

为此，我们需要知道在时间 t 处于状态 j 的概率，我们称之为γt(j)：

$$\gamma_t(j) = P(q_t = j|O, \lambda) \tag{A.25}$$

再一次，我们将通过在概率中包含观察序列来计算：

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \tag{A.26}$$

如图 A.13 所示，公式 A.26 的分子就是前向概率和后向概率的乘积：

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \tag{A.27}$$

我们下面准备计算 b。对于分子，我们对所有时间步 t 求和$\gamma_t(j)$，其中观察值 o_t 是我们感兴趣的符号 v_k。对于分母，我们对所有时间步 t 求和$\gamma_t(j)$。结果是我们处于状态 j 并看到符号 v_k 的次数的百分比：

$$\hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t.\ o_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{A.28}$$

（其中，符号$\sum_{t=1\ s.t.\ O_t=v_k}^{T}$表示“对所有 t 求和，在时间 t 的观察为 v_k”。）

我们现在在公式 A.23 和公式 A.28 中有办法从观察序列 O 重新估计转移 A 和观察 B 的概率，假设我们已经有了对 A 和 B 的先前估计。

这些重新估计形成了迭代前向-后向算法的核心。前向-后向算法（图 A.14）从 HMM 参数λ = (A,B)的一些初始估计开始，然后迭代地运行两个步骤（与 EM 算法一样）：期望步骤(E-step)、最大化步骤(M-step)。

在 E-step 中，我们根据较早的 A 和 B 概率计算预期状态占用计数γ和预期状态转移计数ξ。

在 M-step 中，我们使用γ和ξ重新计算新的 A 和 B 概率。

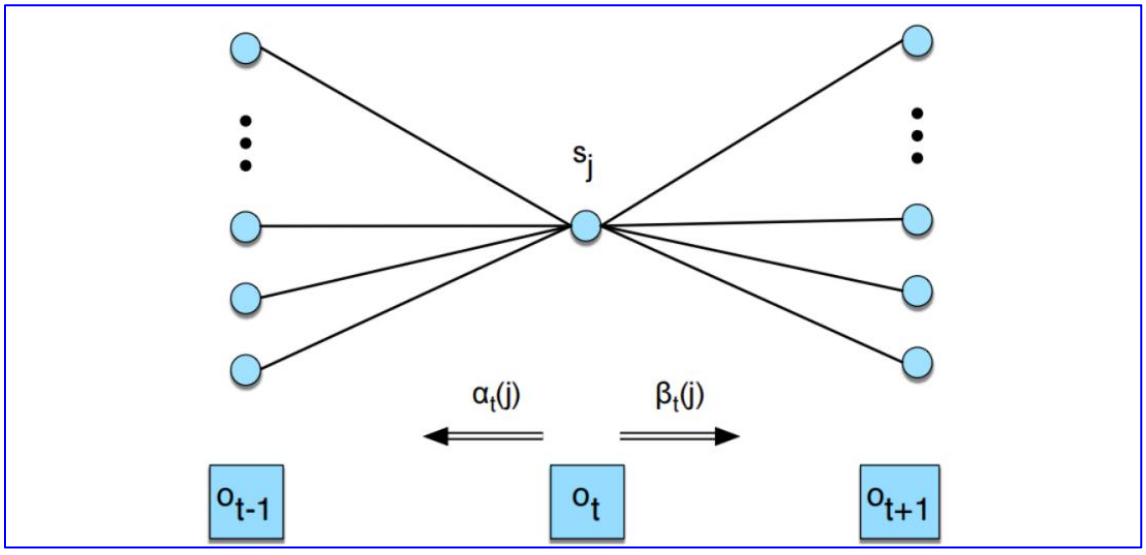


图 A.13：$\gamma_t(j)$的计算，即在时间 t 处于状态 j 的概率。请注意，γ 实际上是 ξ 的退化情况，因此该图类似于图 A.12 的版本(其中状态 i 与状态 j 折叠）。来自 Rabiner (1989) ©1989 IEEE。

function FORWARD-BACKWARD(*observations* of len *T*, *output vocabulary V*, *hidden state set Q*) **returns** *HMM=(A,B)*

initialize *A* and *B*

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return *A*, *B*

图 A.14：前向-后向算法伪代码

尽管原则上前向-后向算法可以对 A 和 B 参数进行完全无监督的学习，但实际上初始条件非常重要。出于这个原因，算法通常会得到额外的信息。例如，对于基于 HMM 的语音识别，HMM 结构往往是手工设置的，只有发射（B）和（非零）A 转移概率是从一组观察序列 O 中训练出来的。

27.6. 总结

本章介绍了用于概率序列分类的隐马尔可夫模型。

- 隐马尔可夫模型(**HMM**) 是一种将观察序列与解释观察的**隐藏类**或**隐藏状态**序列相关联的方法。
- 给定观察序列，发现隐藏状态序列的过程称为**解码**或**推理**。**Viterbi 算法**通常用于解码。
- HMM 的参数是 A 转移概率矩阵和 B 观察似然矩阵。两者都可以使用 Baum-Welch 或**前向-后向算法**进行训练。

27.7. 文献和历史说明

正如我们在第 8 章中所讨论的，马尔可夫(1913)首先使用马尔可夫链来预测普希金的尤金·奥涅金中即将出现的字母是元音还是辅音。隐马尔可夫模型由普林斯顿国防分析研究所的鲍姆及其同事开发（Baum and Petrie 1966, Baum and Eagon 1967）。

Viterbi 算法首先由 Vintsyuk(1968)在语音识别的上下文中应用于语音和语言处理，但具有 Kruskal (1983)所说的“多次独立发现和发表的非凡历史”。Kruskal 和其他人至少给出了以下在四个不同领域发表的独立发现的算法变体：

Citation	Field
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science

术语 Viterbi 的使用现在是将动态规划应用于语音和语言处理中任何类型的概率最大化问题的标准。对于非概率问题（例如最小编辑距离），通常使用简单的术语动态规划。Forney, Jr. (1973)写了一篇早期的调查论文，在信息和通信理论的背景下探讨了 Viterbi 算法的起源。

我们对隐马尔可夫模型应具有三个基本问题特征的想法的表述是根据 Rabiner(1989)的一个有影响力的教程建模的，该教程本身基于 1960 年代 IDA 的 Jack Ferguson 的教程。Jelinek(1997)和 Rabiner 和 Juang(1993)对应用于语音识别问题的前向-后向算法给出了非常完整的描述。Jelinek(1997)还显示了前向后向和 EM 之间的关系。

28. 主题索引和参考文献

28.1. 主题索引

* ? 7
+? 7
Allen 关系(Allen relations) 285
B3 351
BERT 105
CORAAL 441
ELIZA(伊丽莎) 3
FrameNet 307
F-度量(F-measure) 54
GUS(Genial Understander System) 404
Hearst 模式 274
HotpotQA 381
Kleene * 5
Kleene + 5
Kneser-Ney 算法 37
L1 正则化(L1 regularization) 73
L2 正则化(L2 regularization) 73
MUC F 度量(MUC F-measure) 351
NomBank 307
n 元语法(n-gram) 25
n-最佳列表(n-best list) 445
Potts 图 326
ReLU 107
RNN-Transducer(RNN-T) 450
SGNS(skip-gram with negative sampling) 92
sigmoid 65
Tacotron2 架构 454
tf-idf 模型 82
Transformer 105
Wh 代词 125
wh-短语(wh-phrase) 194
wh-非主语疑问式(wh-non-subject-question 194
wh-疑问式(wh-questions) 194
wh-主语疑问式(wh-subject-question) 194
Winograd 模式(Winograd Schema) 352
word2vec 模型 82
X-bar 图式(x-bar schemata) 210
λ化简(λ-reduction) 258
巴科斯-诺尔范式(Backus-Naur-Form，BNF) 190
半监督(semi-supervised) 280
半元音(approximant) 424
绑定(bound) 339
包含(subsumption) 263
饱和(saturated) 107
保留(held out) 29
报告事件(reporting events) 284
贝叶斯推理(Bayesian inference) 46
本体知识体系(ontology) 263
鼻腔(nasal tract) 422
鼻音(nasal sounds) 423
边界声调(boundary tones) 427
边因子(edge-factored) 243
边音(lateral sound) 424
编解码器(encoder-decoder) 105
编钟(CHiME) 441
扁平化(flattening) 215
变量(variable) 253
标记器(tagger) 45
标签（label） 45
表示学习(representation learning) 67
表述伤害(representational harm) 99
表外词(OOV) 33
滨州树库(Penn Treebank) 14
滨州语篇树库(Penn Discourse TreeBank，PDTB) 359
并列(coordinate) 199
并列链接歧义(coordination ambiguity) 214
波束宽度(beam width) 176
波束搜索(beam search) 176
补语(complementizer) 125
捕获组(capture group) 9
不定名词短语(Indefinite Noun Phrases) 339
不合语法的(ungrammatical) 192
不及物(intransitive) 198
不流利(disfluency) 10
布隆过滤器(Bloom filters) 39
步长(stride) 442
部分词(meronym) 292
部分解析(partial parse) 222
部分-整体(part-whole) 292
部分-整体关系(meronymy) 81
采样(sampling) 428
采样率(sampling rate) 441
参考点(reference point) 261
槽(slot) 405
槽填充(slot filling) 405
槽填充(slot-filler) 251

侧序列(side sequence) 397
测试集(test set) 29
插值(interpolation) 36
插值的 Kneser-Ney 38
插值精度(interpolated precision) 377
颤音(tap) 424
常量(constants) 255
超参数(hyperparameter) 71
超发音(hyperarticulation) 410
超级标记(supertagging) 225
超意义(supersense) 293
陈述式(declaratives) 194
成分分析(componential analysis) 314
承诺语(Commissives) 396
承受格替换(dative alternation) 305
澄清(clarifications) 397
澄清问题(clarification questions) 413
持续(duration) 281
齿音(Dental) 423
齿龈音(Alveolar) 423
充分性(adequacy) 180
抽象(abstract) 322
抽象含义表示(Abstract Meaning Representation，AMR) 251
唇齿音(labiodental) 423
唇音(Labial) 423
词表(lexicon) 190
词典(thesaurus 290
词干(stem) 17
词干分析(stemming) 17
词干分析器(stemmer) 17
词干化(stemming) 3
词关联(word relatedness) 81
词汇触发器(lexical triggers) 281
词汇空白(lexical gap) 170
词汇衔接(lexical cohesion) 367
词汇样本(lexical sample) 294
词汇语义(Lexical semantics) 80
词汇重音(lexical stress) 426
词类(parts of speech，POS) 123
词类标记(part-of-speech tagging) 125
词联想(word association) 81
词频(term frequency，tf) 88
词相似(word similarity) 80
词相似(word Smilarity) 80
词向量(word2vec) 82
词形(wordform) 10, 80
词形还原(Lemmatization) 3
词义归纳(word sense induction) 300
词义消歧(word sense disambiguation) 80, 294
词元(lemma) 10, 80
词缀(affix) 17
次范畴化(subcategorize，再分类) 198
搭配(collocation) 297
达成体表示(achievement expression) 263
大小写折叠(case folding) 17
代表性伤害(representational harms) 59
代词(pronoun) 125
代词省略(pro-drop) 170
代码转换(code switching) 11
单词错误率(word error rate，WER) 450
单词符记(word token) 140
单词含义(word meaning) 80
单词类型(word type) 140
单词意义(word sense) 80
单例(singleton) 338
单数他们(singular they) 342
单位产生式(unit productions) 215
倒排索引(inverted index) 376
地名词典(gazetteer) 137
低帧率(low frame rate) 445
低资源语言(low-resourced languages) 185
递归神经网络(recurrent neural networks，RNN) 105
典范形式(canonical form) 252
点积(dot product) 64
点积注意力(dot-product attention) 174
电话带宽(telephone-bandwidth) 442
电话总机(Switchboard) 440
定位符(anchors) 6
丢弃(dropout) 117
动词(verb) 123
动词框架(verb-framed) 170
动词替换(verb alternations) 305
动词语义学(Verb Semantics) 343
动名词(gerundive) 196
动态规划(dynamic programming) 19
毒性检测(toxicity detection) 59
独热(one-hot) 113
独特(idiosyncratic) 168
端到端训练(end-to-end training) 150
短语动词(phrasal verb) 125
短语结构语法(Phrase-Structure Grammars) 190
对半可靠性(split-half reliability 323
对比原理(principle of contrast) 80
对话(dialogue) 395
对话策略(dialogue policy) 410

对话行为(dialogue acts) 408
对话状态(dialogue state) 404
对抗性评价(adversarial evaluation) 415
对齐(alignment) 18, 446
对数概率(log probabilies) 29
对数似然比(log likelihood ratio) 328
对数似然比率(log likelihood ratio) 328
对数优势比率(log odds ratio) 328
多层感知器(multi-layer perceptrons，MLP) 110
多元逻辑回归(multinomial logistic regression) 74
多综合(polysynthetic) 170
轭合(zeugma) 290
耳蜗(cochlea) 433
二阶共现(second-order co-occurrence) 98
二元语法(bigram) 27
发音部位(place of articulation) 423
发音词典(pronunciation dictionaries) 436
发音方式(manner of articulation) 423
发音语音学(articulatory phonetics) 422
翻译差异(translation divergence) 168
反改造(counterfitting) 300
反文档频率(inverse document frequency，idf) 88
反向传播(error backpropagation) 113
反向翻译(backtranslation) 180
反向链(backward chaining) 259
反义词(antonym) 291
反义关系(antonymy) 81, 291
范式关联(paradigmatic association) 98
范围(rang) 4
方括号表示法(bracketed notation) 192
方式(manner) 124
方向(directional) 124
访问(accessed) 337
放大(amplify) 99
非标准词(non-standard word) 453
非监督(unsupervised) 280
非限定(non-finite) 196
非终端(non-terminal) 190
分布假说(distributional hypothesis) 79
分词(word segmentation) 15
分段(segmentation) 3
分段语篇表示理论(Segmented Discourse Representation Theory，SDRT) 371
分离出单词(tokennizing) 3
分裂(clefts) 341
分配伤害(allocational harm) 99
封闭词汇(closed vocabulary) 33
封闭类(closed class) 123
符合语法(grammatical) 192
符记(token) 10
符记化(tokennization) 3
辅音(consonants) 423
辅助(auxiliary) 125
负对数似然损失(negative log likelihood loss) 113
负荷者(bear) 426
附着词(clitic) 14
附着歧义(attachment ambiguity) 214
副词(adverb) 124
改造(retrofitting) 300
概括(generalize) 73
概率质量(probability mass) 34
概念依存(conceptual dependency，CD) 315
感情含义(affective meaning) 81, 319
感情计算(affective computing) 319
感情意义(affective meaning) 319
感叹词(interjections) 124
感召(pathos) 369
感知器(perceptron) 108
高元音(high vowels) 425
格框架(case frame) 305
格子(lattice) 132
给定-新的(given-new) 340
共同基础(common ground) 397
共现矩阵(co-occurrence matrix) 83
共振峰(formant) 434
共指(corefer) 337
共指(coreference) 18
共指链(coreference chain) 338
孤立(isolating) 170
关联(relevance) 398
关系代词(relative pronoun) 196
关系提取(relation extraction) 271
冠词(article) 125
归结法(resolution) 260
归一化(normalize) 27
规范化(normalization) 17
国际音标(International Phonetic Alphabet，IPA) 421
过拟合(overfitting) 73
含义(meaning) 80
函数(function) 255
汉明(Hammin) 442
合语法(grammatical) 192
宏平均(macroaveraging) 54
喉音(Glottal) 423
后期编辑(post-editing) 167
后元音(back vowels) 425

后指(cataphora) 339
后置(postposed) 190
弧线标准(arc standard) 236
话语(discourse) 357
话语连接词(discourse connectives) 359
话语模型(discourse model) 337
话语指称(discourse deixis) 339
换行(newline) 8
唤起(arousal) 320
唤起(evoked) 337
唤醒词(wake word) 456
黄金标签(gold labals) 53
黄金输出(gold output) 113
回环(circularity) 290
回溯(back trace) 21
回退(bakkoff) 36
回指(anaphora) 338
回指性检测器(anaphoricity detector) 345
回指语(anaphor) 338
会话(conversation) 395
会话语音(conversational speech) 440
混淆矩阵(confusion matrix) 53
活动体(activity) 263
机构审查委员会(Institutional Review Boards，IRB) 417
机器翻译(machine translation MT) 167
基本情绪(basic emotions) 320
基本语篇单元(elementary discourse unit，EDU) 359
基频(fundamental frequency，F0) 430
基数词(cardinal numbers) 196
基于分类的 n 元语法(class-based n-gram) 43
基于跨度的成分解析(span-based constituency parsing) 220
基于模板的生成(template-based generation) 407
基于实体(entity-base) 364
基于位置的注意力(location-based attention) 454
基于知识(knowledge-based) 298
基于注意力的编码器-解码器(attention-based encoder decoder，AED) 444
激活(activation) 105
及物(transitive) 198
级联(cascade) 18
即席检索(ad hoc retrieval) 373
急性评估(acute-eval) 414
计算机辅助翻译(computer-aided translation，CAT) 167
计算语义(computational semantics) 251
寄存器(Register) 9
加 k(add-k) 35
加法门(add gate) 154
加一(add-one) 34
假阳性(False positives) 7
假阴性(False negatives) 7
价值敏感设计(value sensitive design) 416
监督机器学习(supervised machine learning) 45
简单循环网络(simple recurrent networks) 145
交叉括号数(cross-brackets) 222
交叉熵(cross-entropy) 41
交叉熵损失(cross-entropy loss) 67
焦点(focus) 390
角色填充符提取(role-filler extraction) 286
脚本(scripts) 285
教师强制(teacher forcing) 148
结构歧义(structural ambiguity) 213
截距(intercept) 64
解码(decoding) 131
解码器(decoder) 172
解释的(interpretable) 76
解析树(parse tree) 191
解析值(PARSEVAL) 222
介词(preposition) 124
近似随机(approximate randomization) 57
精确度-召回率曲线(precision-recall curve) 377
精神(ethos) 369
静态嵌入(static embeddings) 92
旧的话语(discourse-old) 340
局部特征(local features) 135
句点消歧(period disambiguation) 66
句法(syntax) 189
句法关联(syntagmatic association) 98
句法解析(syntactic parsing) 193
句法消歧(syntactic disambiguation) 214
句法依存(syntactic dependencies) 189
句子补语(sentential complements) 198
句子错误率(Sentence error rate) 451
句子分割(sentence segmentation) 18
句子实现(sentence realization) 412
拒绝(rejection) 411
具体(concrete) 322
聚类(cluster) 338
决策边界(decision boundary) 65
绝对(absolute) 281
绝对打折(absolut discunting) 37
卡茨回退(Katz backoff) 36
开发测试集(devset) 30
开放词汇(open vocabulary) 33

开放类(open class) 123
柯里化(currying) 258
可逆词(reversives) 291
可数名词(count nouns) 124
空格(blank) 447
口腔(oral tract) 422
跨度(span) 220
快速重新提示(rapid reprompting) 412
框架(frame) 307, 404
框架文件(frame files) 306
框架元素(frame elements) 307
困惑度(perplexity) 30
扩张卷积(dilated convolutions) 455
垃圾邮件检测(spam detection) 45
拉普拉斯平滑(Laplace smoothing) 34
朗读语音(read speech) 440
类属指代(Generics) 341
类型(types) 10
类型学(typology) 168
类型依存(typed dependency) 231
冷语言(cold language) 170
离散傅里叶变换(Discrete Fourier Transform，DFT) 443
理性(logos) 369
连贯(coherence) 357
连接主义时序分类(Connectionist Temporal Classification，CTC) 446
连续升调(continuation rise) 427
联结主义(connectionist) 121
链式规则(chain rule) 26, 77
量词(quantifiers) 256
量化(quantization) 428
聊天机器人(catbots) 3
聆听-注意力-拼写(listen attend and spell，LAS) 444
零(zeros) 33
零回指(zero anaphor) 340
零假设(null hypothesis) 57
零宽度(zero-width) 10
岭脊(ridge) 74
领域(domain) 253
流(streaming) 450
流利性(flfluency) 180
论元(arguments) 195
论元方案(argumentation schemes) 369
论元关系(argumentative relations) 369
论元区划(argumentative zoning) 370
论元挖掘(argumention mining) 369
逻辑函数(logistic function) 65
逻辑回归(logistic regression) 63
逻辑连接词(logical connectives) 256
马尔科夫(Markov) 27
马尔可夫链(Markov chain) 128
脉冲编码调制(Pulse Code Modulation，PCM) 429
梅尔(mel) 430
每一话语一种意义(one sense per discourse) 296
门(gate) 153
蒙特卡罗搜索(Monte Carlo search) 180
密集(dense) 82
描述逻辑(DL) 263
名词(nouns) 123
名词性成分(nominal) 196
名词性词(Nominal) 195
命令式(imperatives) 194
命名实体(named entity) 123
命名实体识别(named entity recognition，NER) 127
命题含义(propositional meaning) 80
命题库(Proposition Bank) 306
模板识别(template recognition) 286
模板填充(template filling) 271, 285
模棱两可(ambiguous) 126
模型(model) 307
模型卡(model card) 59
摩擦音(fricatives) 423
目标(target) 171
目标矩阵(target matrix) 97
目标嵌入(target embedding) 97
奈奎斯特频率(Nyquist frequency) 428
内涵(connotations) 81
内积(inner product) 86
内容规划(content planning) 412
内在评估(intrinsic evaluation) 29
逆向链(backward chaining) 259
逆向微分(reverse differentiation) 113
凝聚聚类(agglomerative clustering) 300
派生(derivation) 191
配对(paired) 57
配对句子片段单词错误率(MAtched-Pair Sentence Segment Word Error，MAPSSWE) 451
配置(configuration) 236
批量(batch) 72
毗邻对(adjacency pairs) 397
偏差项(bais term), 64
频率(frequency) 427
频谱(spectrogram) 433
平滑(smoothing) 34
平均倒数排名(mean reciprocal rank，MRR) 392

平均精度的均值(mean average precision，MAP) 378
平均意见评分(mean opinion score，MOS) 456
平稳(stationary) 41
平行语料库(parallel corpus，bitext) 179
瓶颈(bottleneck) 174
朴素贝叶斯假设(naive Bayes assumption): 47
普通调查员(General Inquirer，GI) 321
普通名词(common nouns) 123
歧义(ambiguous) 289
启动(bootstrap) 183
前馈网络(feedforward network) 105, 110
前顺序(presequence) 397
前提(premises) 369
前元音(front vowels) 425
前瞻(lookahead) 10
前置(preposed) 190
浅层语篇分析(shallow discourse parsing) 363
浅解析(shallow parse) 222
嵌入(embedding) 79
强等价(strong equivalence) 204
强度(power) 430
乔姆斯基范式(Chomsky normal form，CNF) 204
乔姆斯基邻接规则（Chomsky-adjunction） 204
桥接推理(bridging inference) 340
清音(unvoiced) 422
情感(sentiment) 81
情感词汇(sentiment lexicons) 51
情感分析(sentiment analysis) 45
情态动词(modal verb) 125
情绪(emotion) 319
区分式分类器(Discriminative classifiers) 46
区分性(discriminative) 63
取式推理(Modus ponens) 259
去词汇化(delexicalization) 412
去偏差(debiasing) 99
去偏见(debiasing) 99
权重绑定(Weight Tying) 148
全词(all-words) 294
全局特征(global features) 135
全链接(full connected) 110
热语言(hot language) 170
人称(personal) 125
人称一致性(Person Agreement) 342
人格(personality) 330
人际立场(interpersonal stance) 331
融合(fusion) 170
软腭音(Velar) 423
弱等价(weak equivalence) 204
弱元音(reduced vowels) 426
塞擦音(affricates) 424
塞音(stop) 423
三元组(triple) 273
傻瓜回退(stupid backoff) 39
熵(entropy) 40
熵率(entropy rate) 41
上腭音(Palatal) 423
上位词(hypernym) 273, 291
上位关系(hypernymy) 81, 291
上下文词(word-in-context，WIC) 298
上下文矩阵(context matrix) 97
上下文嵌入(context embedding) 97
上下文无关语法(Context-Free Grammar，CFG) 190
舌面前音(coronal) 423
深度学习(deep learning) 105
生成式分类器(Generative classifiers) 46
生成性(generative) 63
生成语法(generative grammar) 193
声调和间断索引(Tone and Break Indices，ToBI) 427
声码(vocoding) 452
声码器(vocoder) 452
声门(glottis) 422
声谱(spectrum) 432
声腔(vocal tract) 422
声纹识别(Speaker recognition) 456
声源-滤波器模型(source-filter model) 435
施事者(agents) 303
十倍交叉验证(10-fold cross-validation) 56
时间(temporal) 124
时间表达式(temporal expressions) 271
时间对齐转写(time-aligned transcription) 436
时间反向传播(Backpropagation Through Time) 147
时间规范化(temporal normalization) 282
时间库(TimeBank) 285
时间锚定(temporal anchor) 283
时态逻辑(tense logic) 261
时序逻辑(temporal logic) 261
实例检查(instance checking) 265
实体链接(entity linking) 338, 383
实体网格(entity grid) 358, 365
事件共指(event coreference) 271, 338
事件提取(event extraction) 271
是非疑问式(yes-no questions) 194
输出门(output gate) 154
术语(term) 255
术语权重(term weight) 374
数的一致性(Number Agreement) 342

数据表(datasheet) 12
数量修饰语(quantifiers) 196
数字识别(digit recognition) 440
双唇音(bilabial) 423
双语评估替补(BiLingual Evaluation Understudy，BLEU) 181
双元音(diphthong) 425
说话人区分(Speaker diarization) 456
说话人识别(speaker identification) 456
咝音(sibilants) 424
斯坦福问答数据集(Stanford Question Answering Dataset，SQuAD) 381
搜索树(search tree) 176
素质替换(diathesis alternations) 305
损失函数(loss function) 113
所指对象(referent) 337
态度(attitudes) 319
泰(Tay) 417
贪婪的(greedy) 7
套索(lasso) 74
特征交互(feature interactions) 67
特征模板(feature templates) 67
特征向量(feature vectors) 441
提及(mention) 337
提及检测(mention detection) 344
提示(prompt) 407
体(aspect) 262
替换(substitution) 8
填充停顿符(filled pause) 10
条件随机场(conditional random field，CRF) 135
停用词(stop word) 49
通用依存(Universal Dependency) 232
通用依赖(Universal Dependencies，UD) 124
同位语(Appositives) 341
同义词(synonym) 80, 291
同义词集(synset) 293
同义关系(synonymy) 80, 291
统计显著(statistically significant) 57
投影层(projection layer 120
凸的(convex) 69
突显度(prominence) 426
图表解析(chart parsing) 215
推理(inference) 253
外部评估(extrinsic evaluation) 29
外围词(satellites) 170
外围词框架(satellite-framed) 170
外延的(extensional) 253
外置(extraposition) 341
完全确定(fully qualified) 283
完整的(complete) 195
完整的思想(complete thought) 195
网络本体语言(Web Ontology Language，OWL) 266
微平均(microaveraging) 55
围栏(fencepost) 217
维度(demention) 83
维基化(wikification) 383
维特比(Viterbi) 132
伪词(pseudowords) 314
未知词(unkown word) 49
位移减少(shift-reduce) 235
位置(locative) 124
位置嵌入(positional embedding) 160
文本补全(text completion) 161
文本分类(text categorization) 45
文本规范化(text normalization) 3
文本语音转换(text-to-speech，TTS) 440
文本摘要(text summarization) 161
文字平铺(TextTiling) 367
问句升调(question rise) 427
无所指(non-referential) 343
物质名词(mass nouns) 124
误差反向传播(error backpropagation) 113
希尔贝壳-1(AISHELL-1) 441
析取(disjunction) 6
稀疏(sparse) 82
系动词(copula verb) 125
系统主动(system-initiative) 398
下采样(downsample) 442
先行词(antecedent) 338
先验概率(prior probability) 47
显式确认(explicit confirmation) 411
显著性(salience) 341
限定词(determiner) 125
限制语法(restrictive grammar) 407
线性分类器(linear classifiers) 48
线性可分(linearly separable) 109
相对(relative) 281
相对表达式(relative expressions) 271
相对频率(relative frequency) 28
响度(loudness) 430
响应生成(response generation) 401
向后看中心(backward-looking center) 364
向量(vector) 83
向量空间(vector space) 83
向量语义(vector semantics) 79, 82
向量长度(vector length) 86

向前看中心(forward-looking center) 364
消除歧义(disambiguation) 126
消解(resolve) 126
消失梯度(vanishing gradient) 107
小批量(mini-batch) 72
小品词(particle) 124
效价(valence) 320
效应量(effect size) 56
谐波(harmonics) 435
新戴维森(neo Davidsonian) 260
新的话语(discourse-new) 340
信息访问(information access) 167
信息检索(Information retrieval，IR) 84, 373
信息框(infoboxes) 273
信息提取(information extraction，IE) 271
信息状态(information status) 340
行向量(row vector) 85
形容词(adjective) 124
形容词短语(adjective phrase，AP) 196
形态学(Morphologically) 170
性(parts of speech，POS) 123
性别或名词类的一致性(Gender or Noun Class Agreement) 342
修辞结构理论(Rhetorical Structure Theory，RST) 358
修辞结构理论(RST) 358
修正的 Kneser-Ney 39
虚词(function words) 123
序列标签(sequence labeling) 123
序列到序列(sequence to sequence) 167
序数词(ordinal numbers) 196
叙事语(Constatives) 396
选择关联(selectional association) 313
选择限制(Selectional Restrictions) 343
学习率(learning rate) 69
训练集(training set) 29
训练预言机(training oracle) 239
言语行为(speech acts) 396
掩码(mask) 154
一词多义的(polysemous) 289
一阶共现(fifirst-order co-occurrence) 98
一阶逻辑(First-Order Logic) 251
依存(dependent) 232
依存解析(dependency parsing) 189
依存树(dependency tree) 233
依存语法(dependency grammars) 231
依靠(grounding) 397
移情对话(EMPATHETICDIALOGUES) 401
遗忘门(forget gate) 154
易于访问的(accessible) 341
意图确定(intent determination) 405
意义(sense) 80, 289
因果(causal) 455
音调(tune) 427
音高(pitch) 430
音高轨迹(pitch track) 430
音高提取(pitch extraction) 431
音高重读(pitch accent) 426
音节(syllable) 425
音节核(nucleus) 425
音节切分(syllabification) 425
音节尾(coda) 425
音强(intensity) 430
音子(phone) 421
音子配列(phonotactics) 425
引导(bootstrap) 57
引导测试(bootstrap test) 57
引用形式(citation form) 80
隐藏(hidden) 129
隐含(implicature) 398
隐含论元(implicit argument) 316
隐马尔可夫模型(Hidden Markov Model，HMM) 129
隐式论元检测(iSRL) 317
影响因子(affective) 319
用户纠正行为(user correction acts) 410
用户主动(user-initiative) 397
优先(precedence) 6
有定名词短语(Definite Noun Phrases) 339
诱导法(abduction) 260
余弦(cosine) 82
语调短语(intonation phrases) 426
语段(utterance) 10
语法(syntax) 189
语法关系(grammatical relation) 232
语法作用(Grammatical Role) 342
语料库(corpus，其复数为 corpora) 10
语气(moods) 319
语素(morpheme) 17
语言 ID(language id) 45
语言查询和单词计数(Linguistic Inquiry and Word Count，LIWC) 322
语言模型(language model，LM) 25
语言识别(language identification) 456
语义场(semantic field) 81
语义角色(semantic roles 304
语义角色标签(Semantic role labeling，SRL) 308
语义解析(semantic parsing) 251

语义框架和角色(Semantic Frames and Roles) 81
语义漂移(semantic drift) 277
语义索引(semantic concordance) 295
语义文本相似性(semantic textual similarity) 101
语义语法(semantic grammar) 406
语音合成(speech synthesis) 440
语音学(phonetics) 421
语音用户界面(voice user interface) 416
语用(pragmatic) 194
预训练(pretraining) 118
预言机(oracle 237
元规则(metarules) 199
元数(arity) 260
元音(vowels) 423
元语法(trigram) 29
原始分解(primitive decomposition) 314
圆唇元音(rounded vowels) 425
源(source) 171
远程监督(distant supervision) 278
约束理论限制(Binding Theory Constraints) 342
阅读理解(reading comprehension) 380
运算符优先级(operator precedence) 6
韵(rime) 425
韵律短语(prosodic phrasing) 426
韵律学(prosody) 425
蕴涵(entailment) 291
蕴涵层次结构(implied hierarchy) 266
再词汇化(relexicalize) 413
增强性和替代性交流系统(AAC) 25
粘着(agglutinative) 170
长(long) 86
长短期记忆(LSTM) 145
长距离依赖关系(long-distance dependencies) 194
召回率(Recall) 54
折扣(discounting) 34
帧(frame) 442
真值条件语义学(truth-conditional semantics) 255
振幅(amplitude) 427
整体词(holonym) 292
正向链(forward chaining) 259
正向逐点互信息(Positive Pointwise Mutual Information，PPMI) 89
正则表达式(Regular Expression，RE) 4
正则化(regularization) 73
支配(dominate) 191
支配性(dominance) 320
知识图谱(knowledge graphs) 271
直接成分分析（immediate-constituent analysis，ICA） 209
指代密度(referential density) 170
指定的起始符号(designated start symbol) 191
指示(denotation) 253
指示代词(Demonstrative Pronouns) 340
指示语(Directives) 396
指示语(referring expression) 337
质心(centroid) 92
致电回家(CALLHOME) 441
致谢语(Acknowledgments) 396
置信度值(confidence values) 277
中间短语(intermediate phrases) 426
中心词(head) 232
中心理论(Centering Theory) 358
中元音(schwa) 426
终端(terminal) 190
终结体可能事件(telic eventities) 263
种子模式(seed patterns) 276
种子元组(seed tuples) 276
众包(crowdsourcing) 322
重读(accent) 426
重新评价(rescore) 445
重音(stress) 426
周期(period) 427
咒骂语(Expletives) 341
逐步提示(progressive prompting) 411
逐点互信息(Pointwise Mutual Information，PMI) 89
主动性(initiative) 397
主流美国英语(Mainstream American English，MAE) 11
主题(theme) 304
主题格(thematic grid) 305
主题角色(thematic roles) 303
主-谓-宾(subject-predicate-object) 273
主有(possessive) 125
主张(claims) 369
注释(gloss) 290
注意力(attention) 105
专业术语(terminology) 263
专有名词(proper noun) 124
赘述的(pleonastic) 341
浊音(voiced) 422
资源描述框架(Resource Description Framework，RDF) 273
子单词(subword) 15
子对话(subdialogue) 397
自动语音识别(ASR) 439
自举(bootstrap) 276

自然问题(Natural Questions) 381
自我监督(self-supervision) 92
自由词序(free word order) 231
自注意力(self-attention) 156
字典(dictionary) 290
字节对编码(BPE) 15
组合范畴语法(combinatory categorial grammar，CCG 189
组块分析(chunking) 222
最常见的意义(most frequent sense，MFS) 295
最大生成树(maximum spanning tree，MST) 244
最大似然估计(MLE) 27
最好-最差缩放(best-worst scaling) 323
最后降调(final fall) 427
最频繁(most frequent) 126
最小编辑距离(minimum edit distance) 18
作者身份归属(authorship attribution) 45

28.2. 参考文献