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。

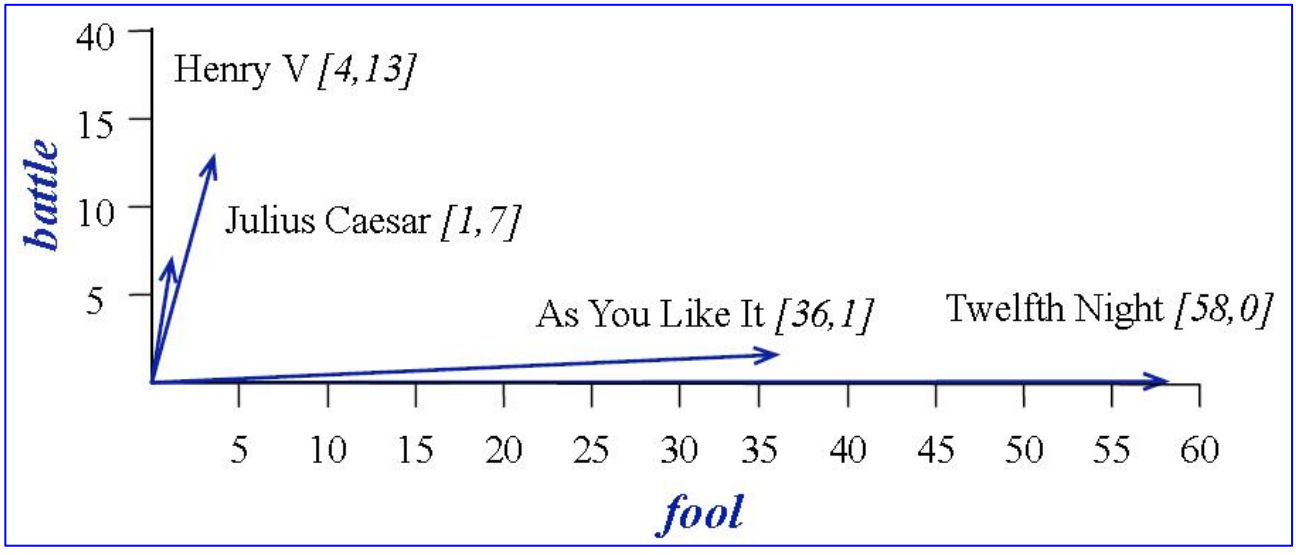


图 6-4：文档向量的空间可视化

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。

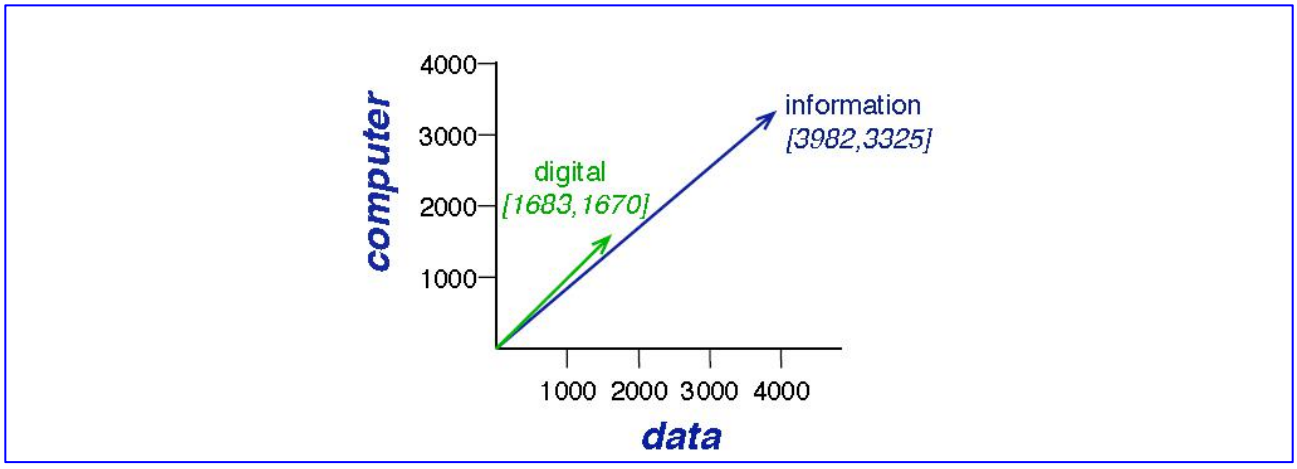


图 6-7：词向量的空间可视化

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。

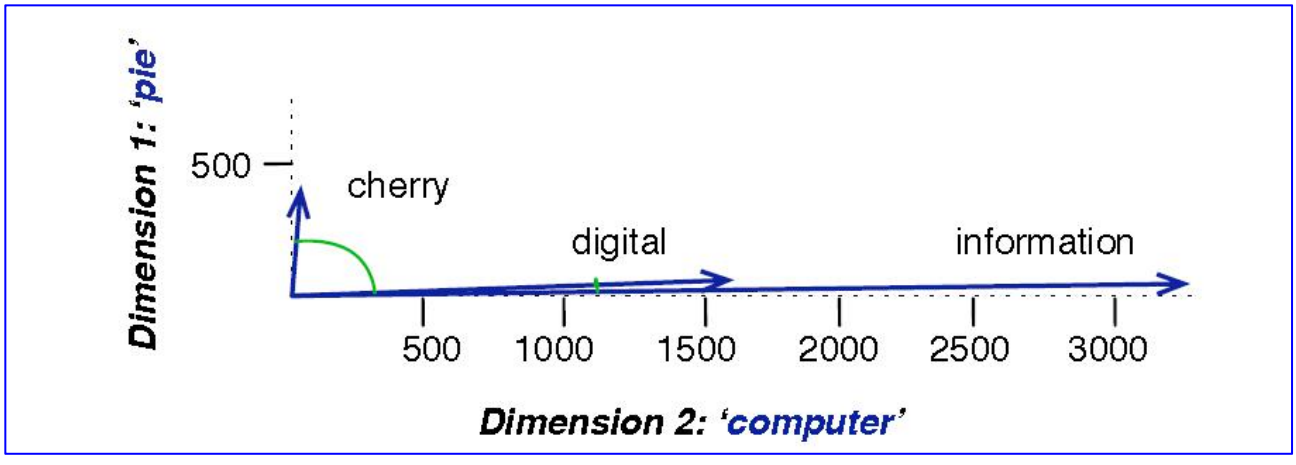


图 6-8：余弦相似度的(粗略)图形演示

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)。

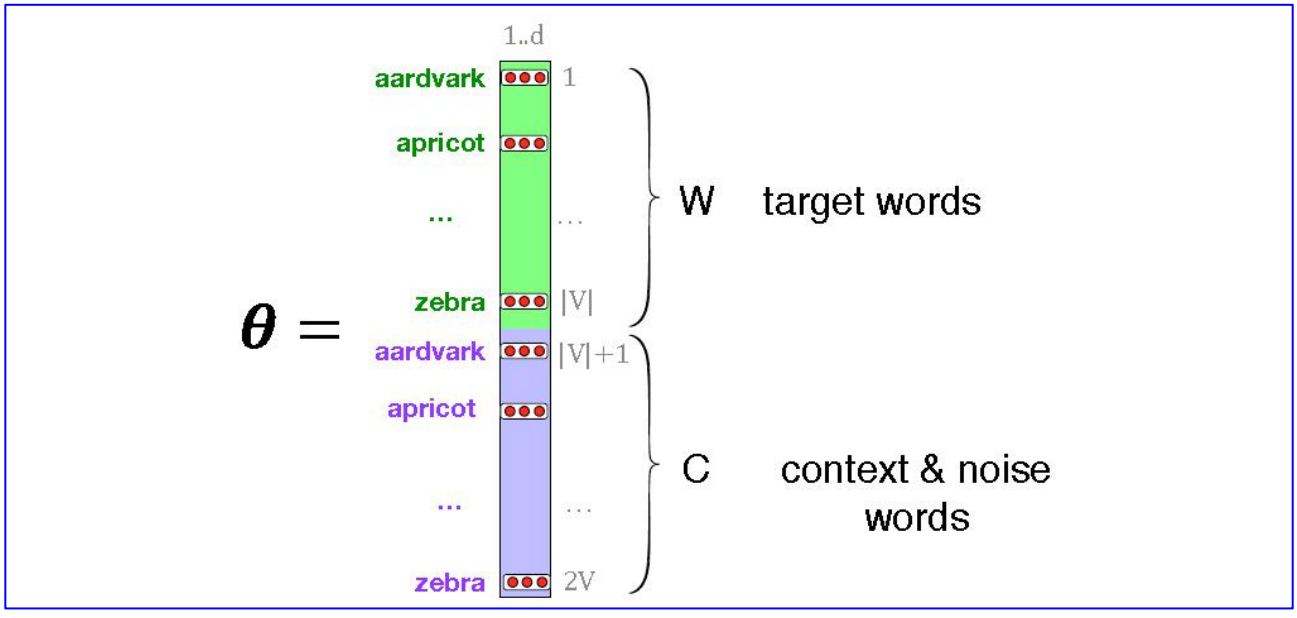


图 6-13：skip-gram 模型学习到的嵌入

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。

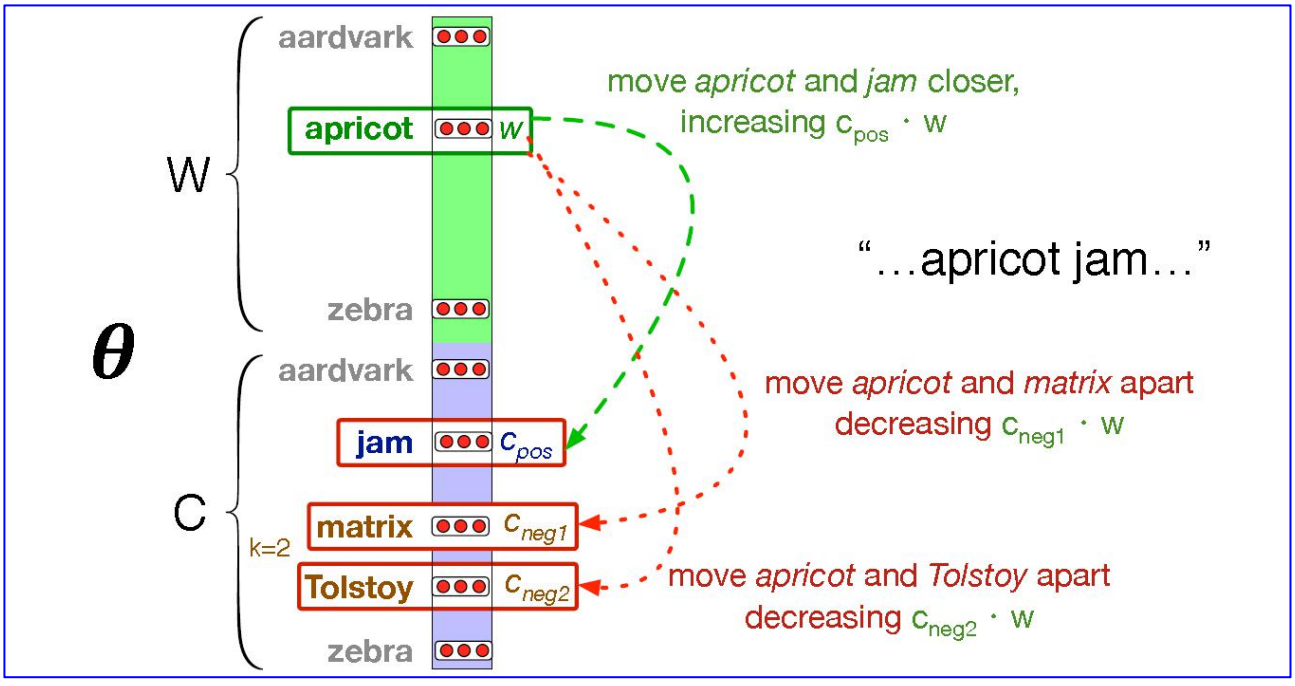


图 6-14：梯度下降法的直观感受

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>

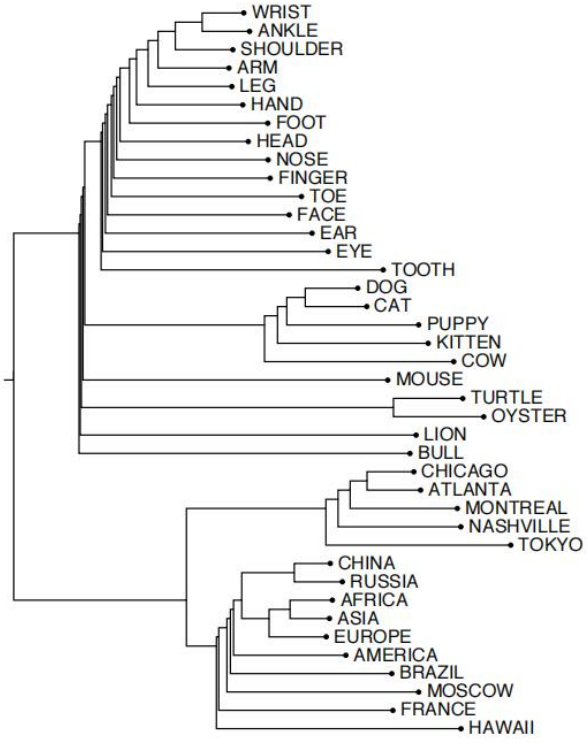
然后，为每个 n-gram 成分学习一个 skip-gram 嵌入，单词 where 由其 n-gram 成分的所有嵌入之和表示。快速文本开源库，包括针对 157 种语言的预训练嵌入，可从 https://fasttext.cc 获得。

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。

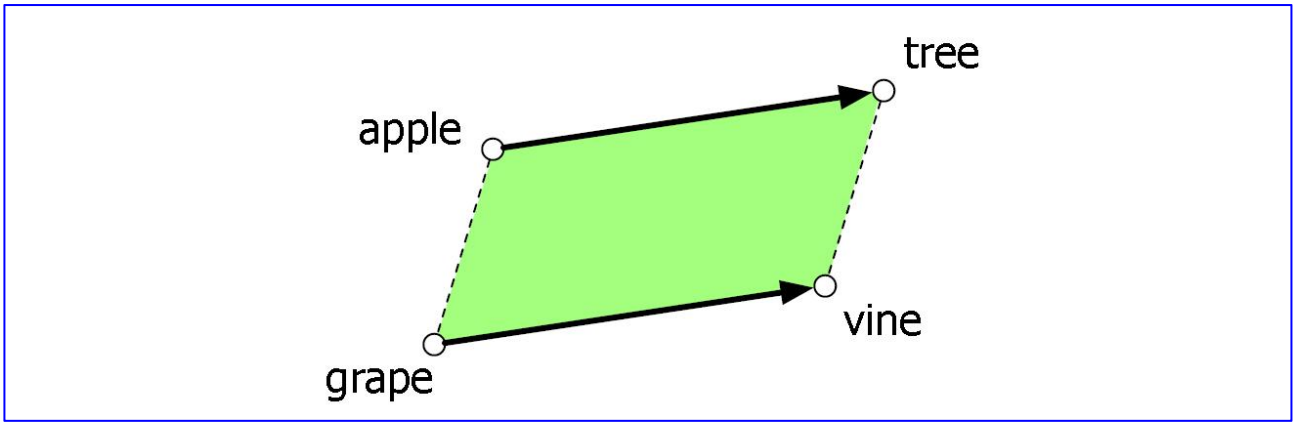


图 6-15：类比问题的平行四边形模型

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

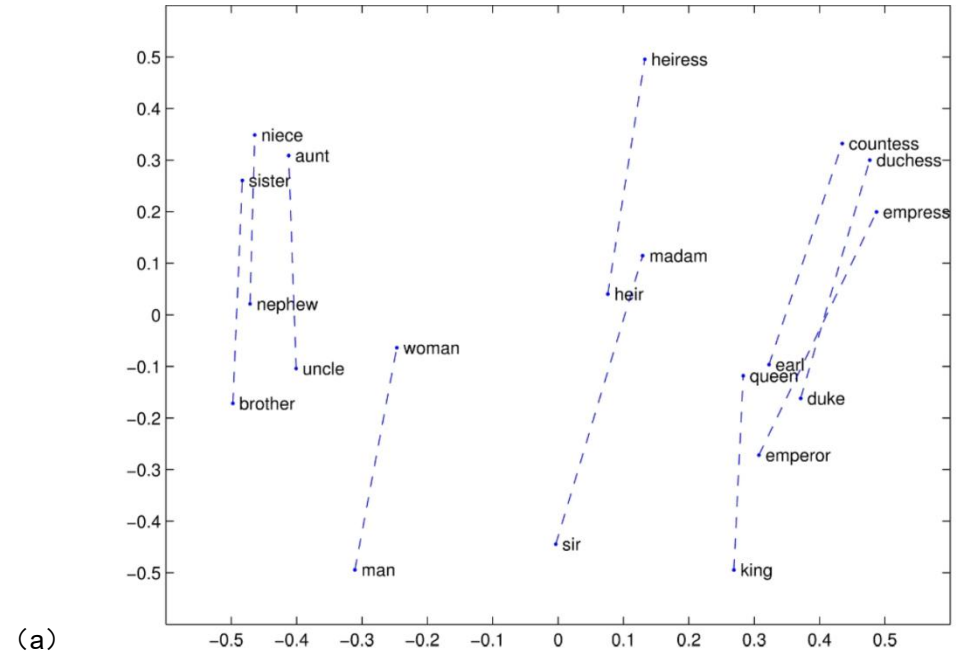
除了他们从文本中学习单词含义的能力外，嵌入，唉，也复制了潜在的文本中的隐性偏见和刻板印象。如前一节所示，嵌入可以大致模拟关系相似性:'queen'作为最接近'king'-'man'+'woman' 暗示了类似的含义:man:woman::king:queen。但是这些同样的内嵌类比也表现出性别刻板印象。例如，Bolukbasi 等人(2016)发现，在新闻文本上训练的 word2vec 嵌入中，与"男人"-"计算机程序员"+"女人"最接近的职业是"家庭主妇"，嵌入同样表明，"父亲"是"医生"，就像"母亲"是"护士"。这可能导致 Crawford(2017)和 Blodgett 等人(2020)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)。

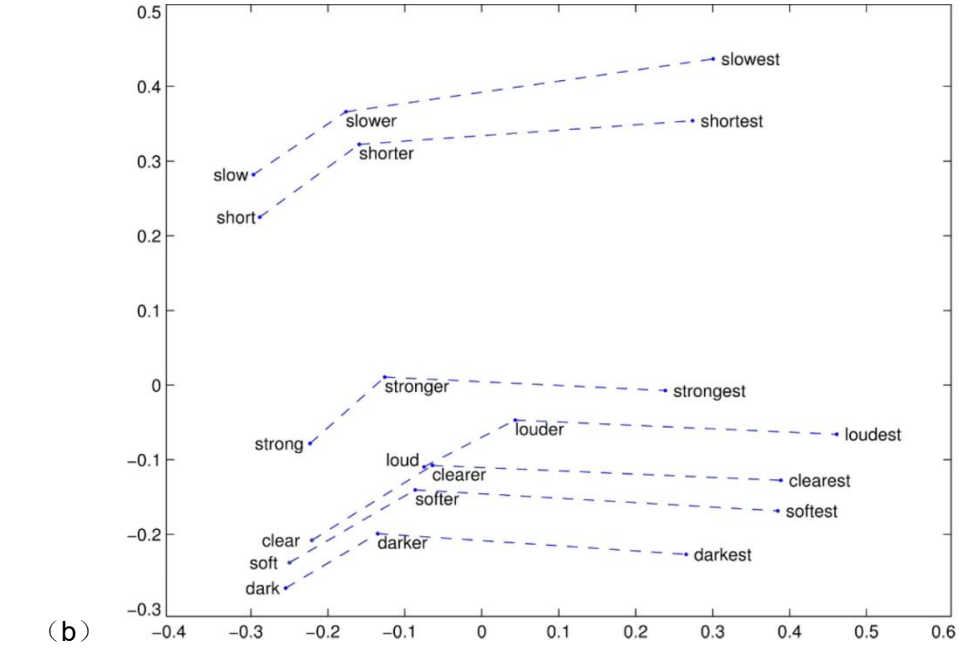
嵌入也编码了内隐联想，这是人类推理的一种属性。内隐联想测试(Greenwald 等人，1998)通过测量人们在不同类别中标记词语的潜伏期的差异，来衡量人们在概念(如"花"或"昆虫")和属性(如"愉快"和"不愉快")之间的联系[8]。使用这种方法，已证明美国民众将非裔美国人的名字与不愉快的单词联系在一起(比欧裔美国人的名字更多)，男性名字与数学联系在一起，女性名字与艺术联系在一起，而老年人的名字则与讨厌的单词联系在一起(Greenwald 等人 1998；Nosek 等人 2002a，Nosek 等人 2002b)。Caliskan 等人(2017)使用 GloVe 向量和余弦相似度而不是人类潜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。

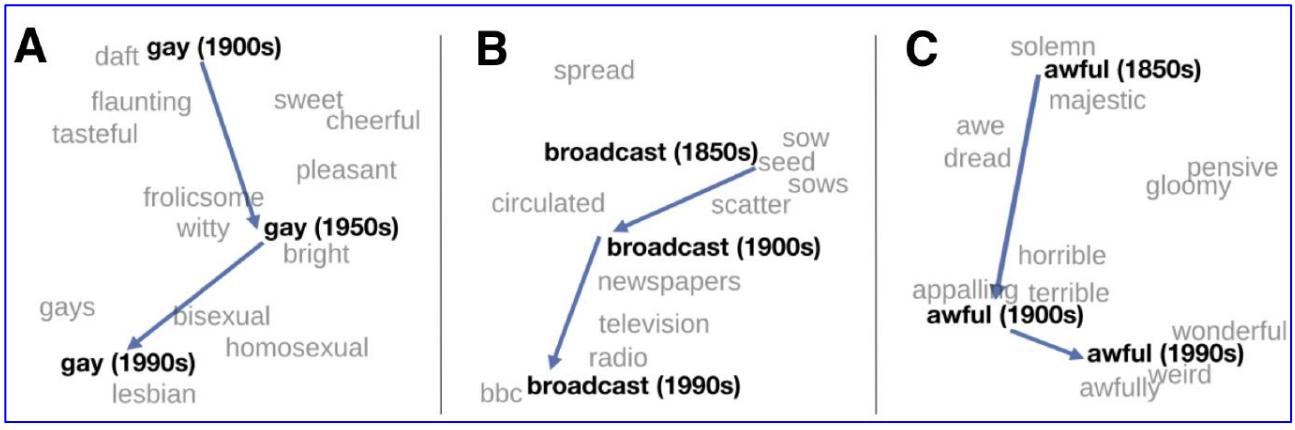


图 6-17：使用 word2vec 进行 t-SNE 可视化

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)