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。

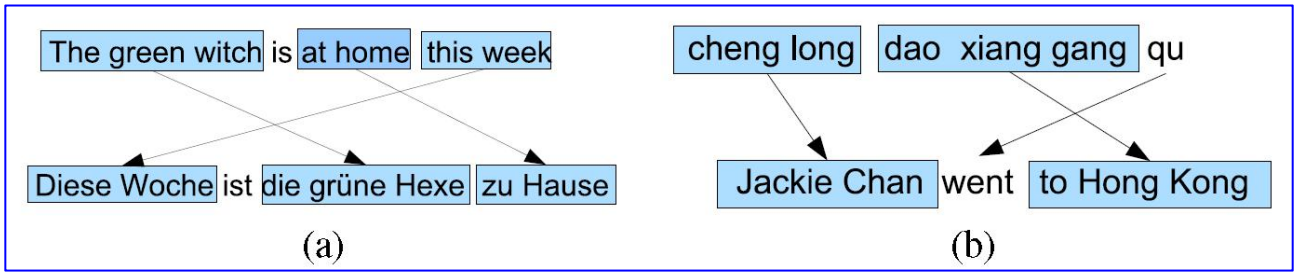

图 11-1：其他词序差异的例子

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。

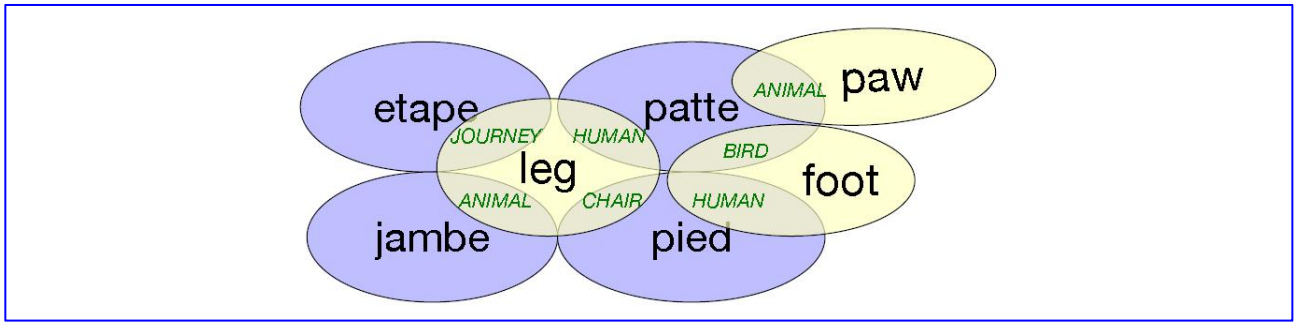

图 11-2：英语和法语翻译之间的复杂重叠

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：

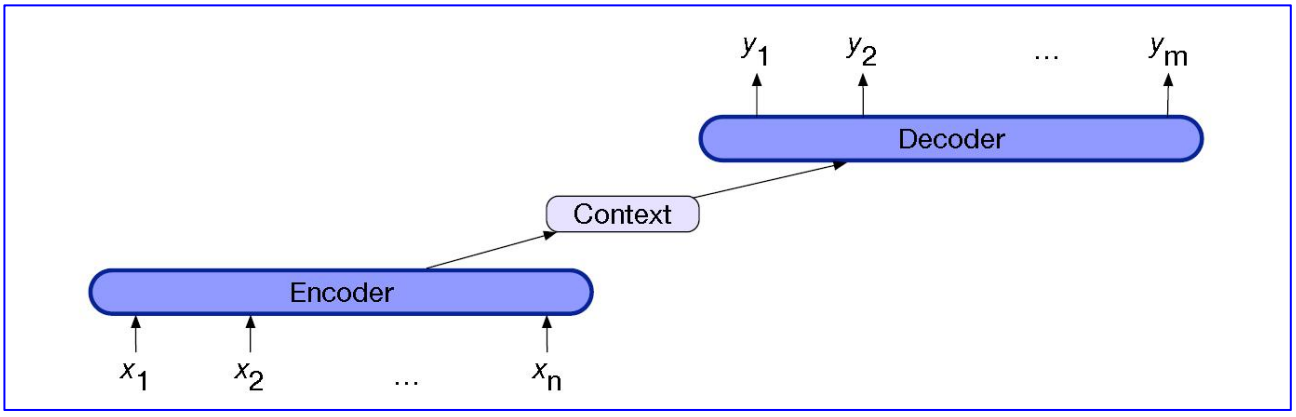

图 11-3：编码器-解码器架构

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。

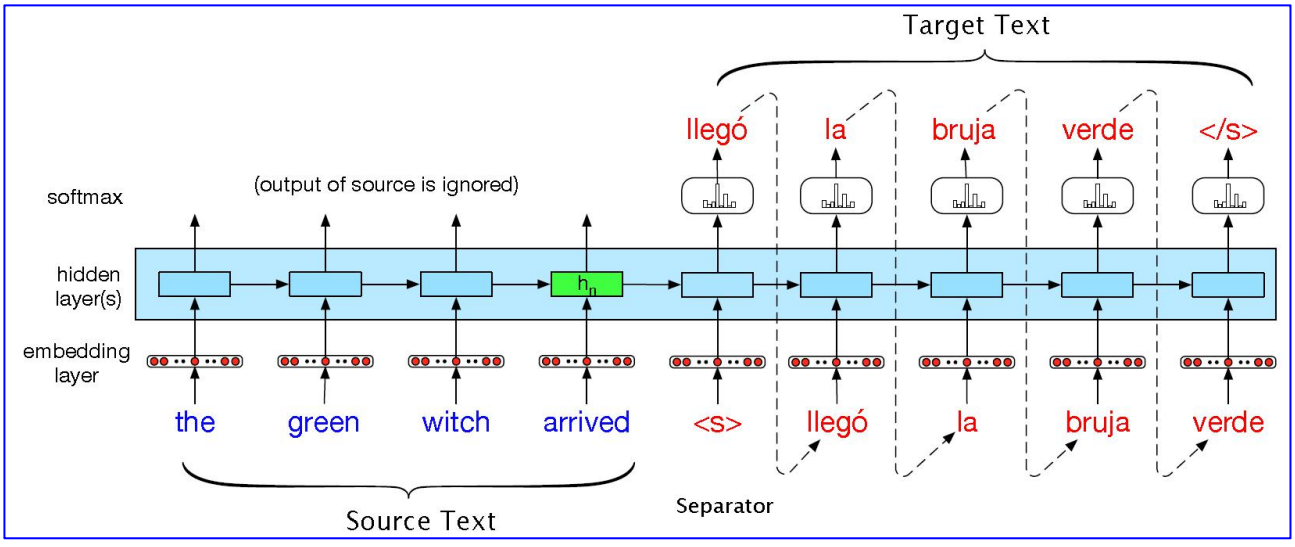

图 11-4：基本的 RNN 编码器-解码器的机器翻译

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。

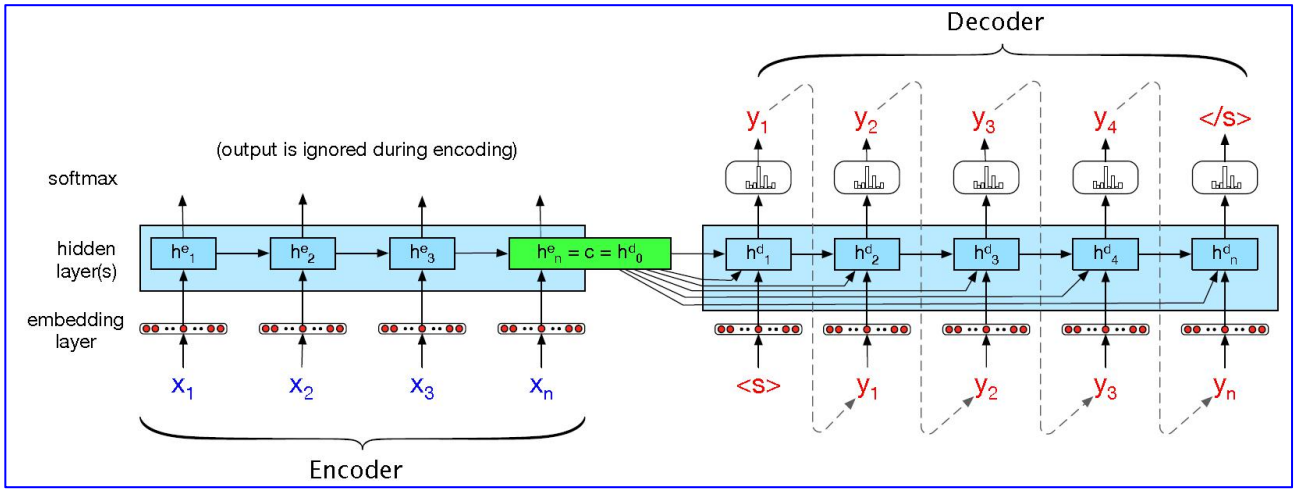

图 11-5：正式的 RNN 编解码器的机器翻译

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):

$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c) \tag{11.11}$$

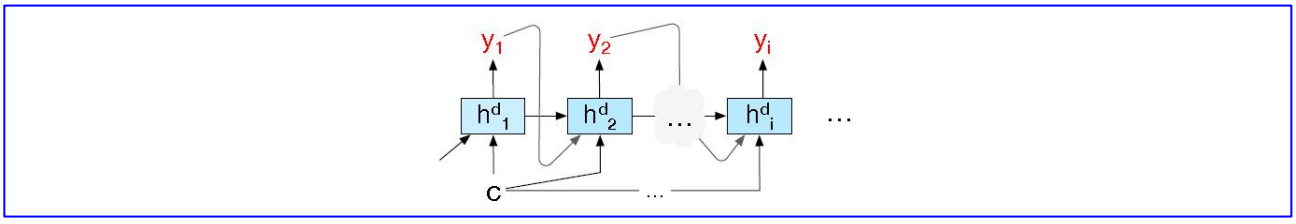

图 11-6：解码器的隐藏状态

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。

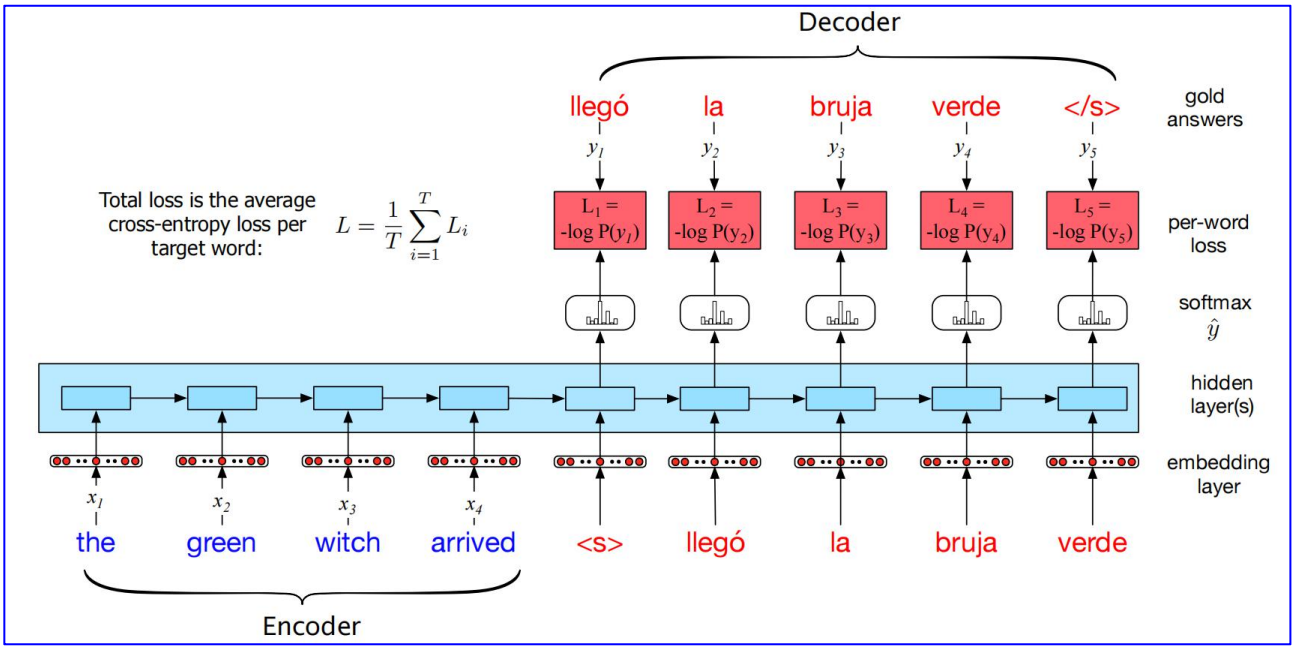

图 11-7：训练的 RNN 编解码器的机器翻译

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。

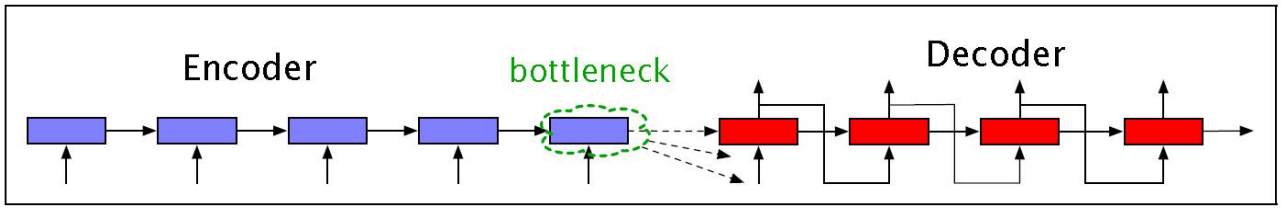

图 11-8：编码器的隐藏状态-瓶颈

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。

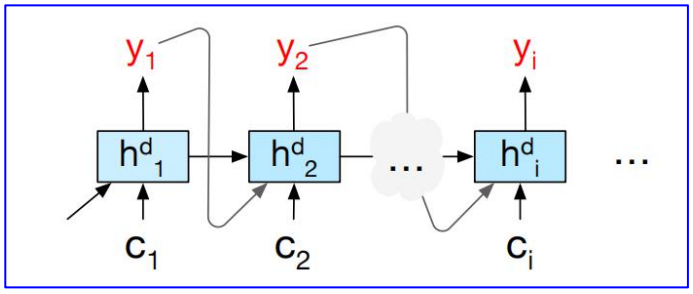

图 11-9：注意力机制的动态上下文

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。

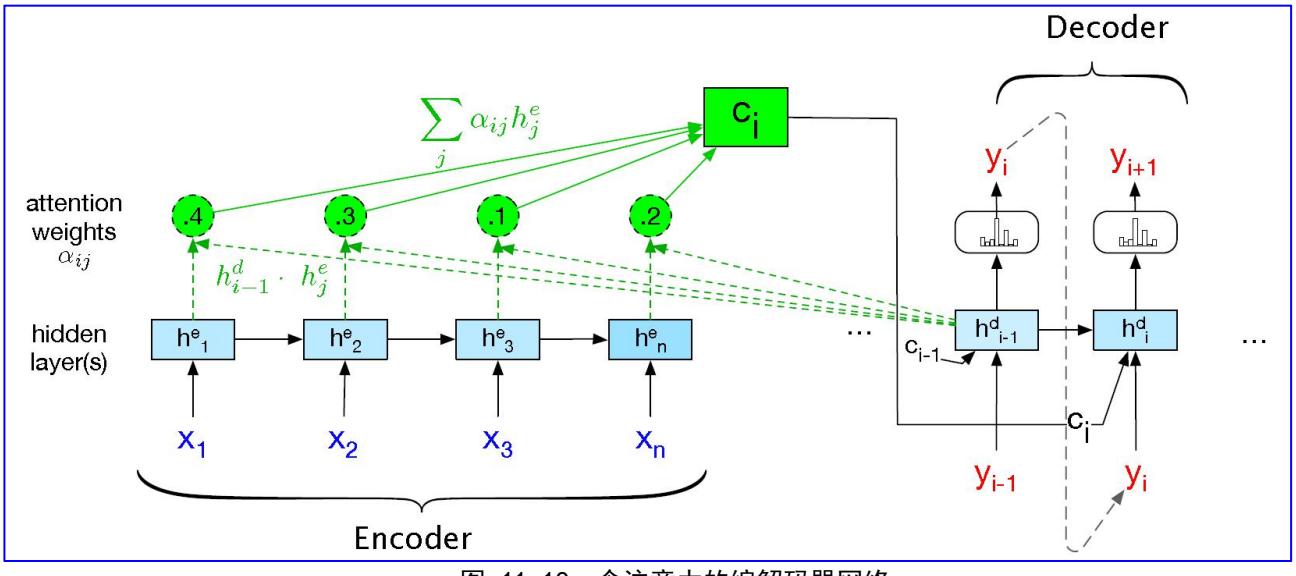

图 11-10：含注意力的编解码器网络

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。

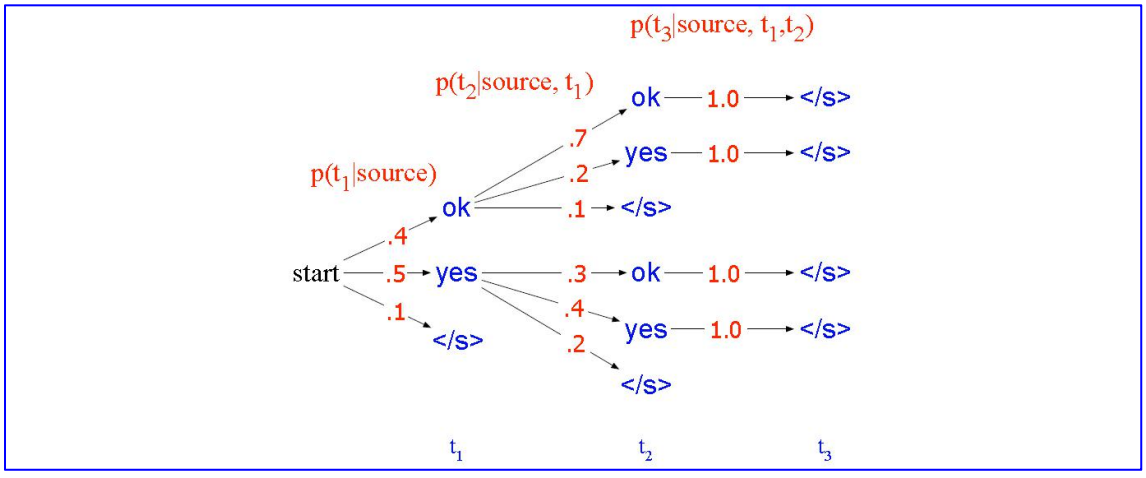

图 11-11：搜索树

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。

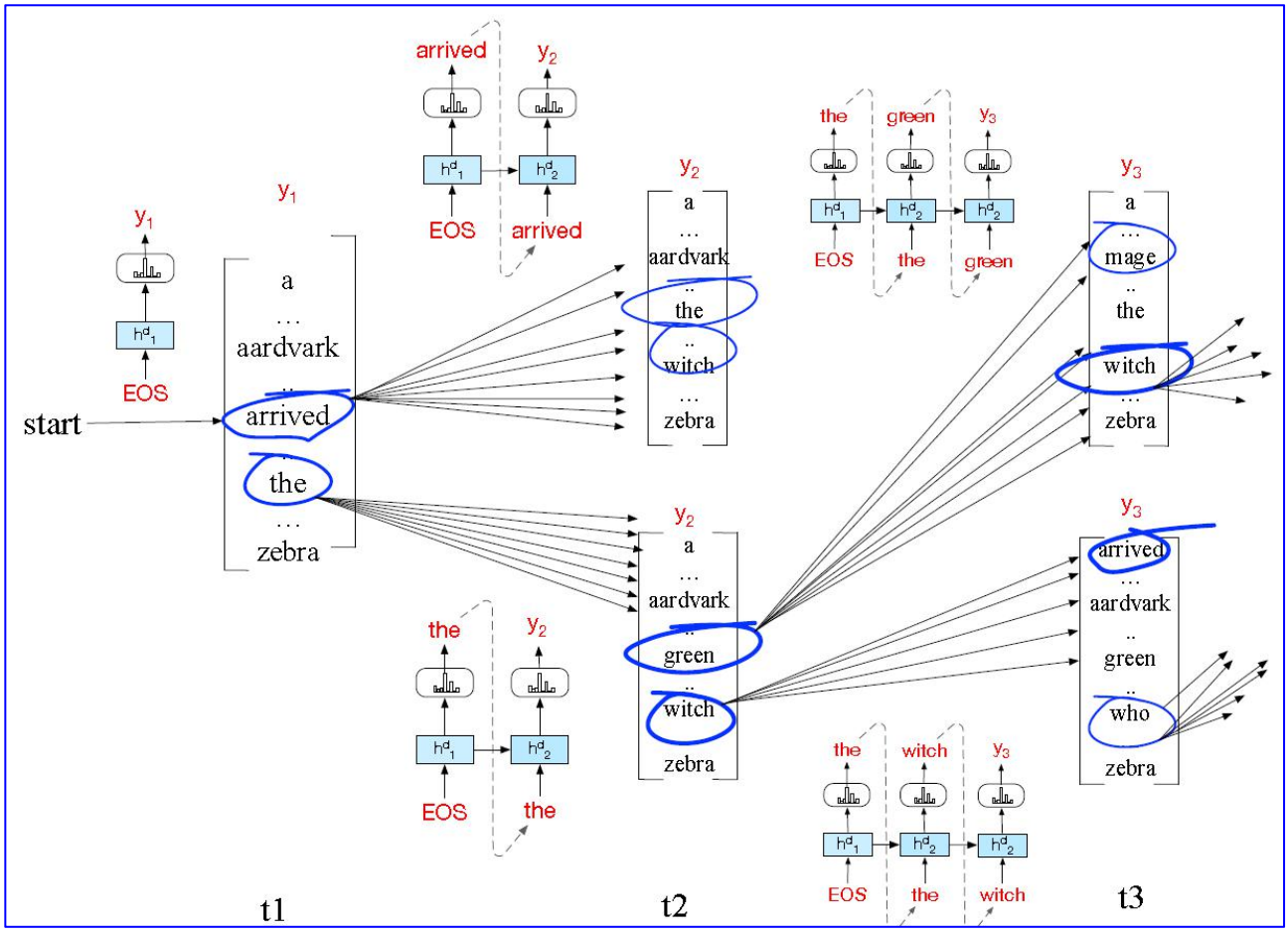

图 11-12:波束搜索解码的原理

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:

$$\begin{aligned} score(y) &= -\log P(y|x) \\ &= \frac{1}{t}\sum_{i=1}^{t} -\log P(y_i|y_1,...,y_{i-1},x) \end{aligned} \tag{11.21}$$

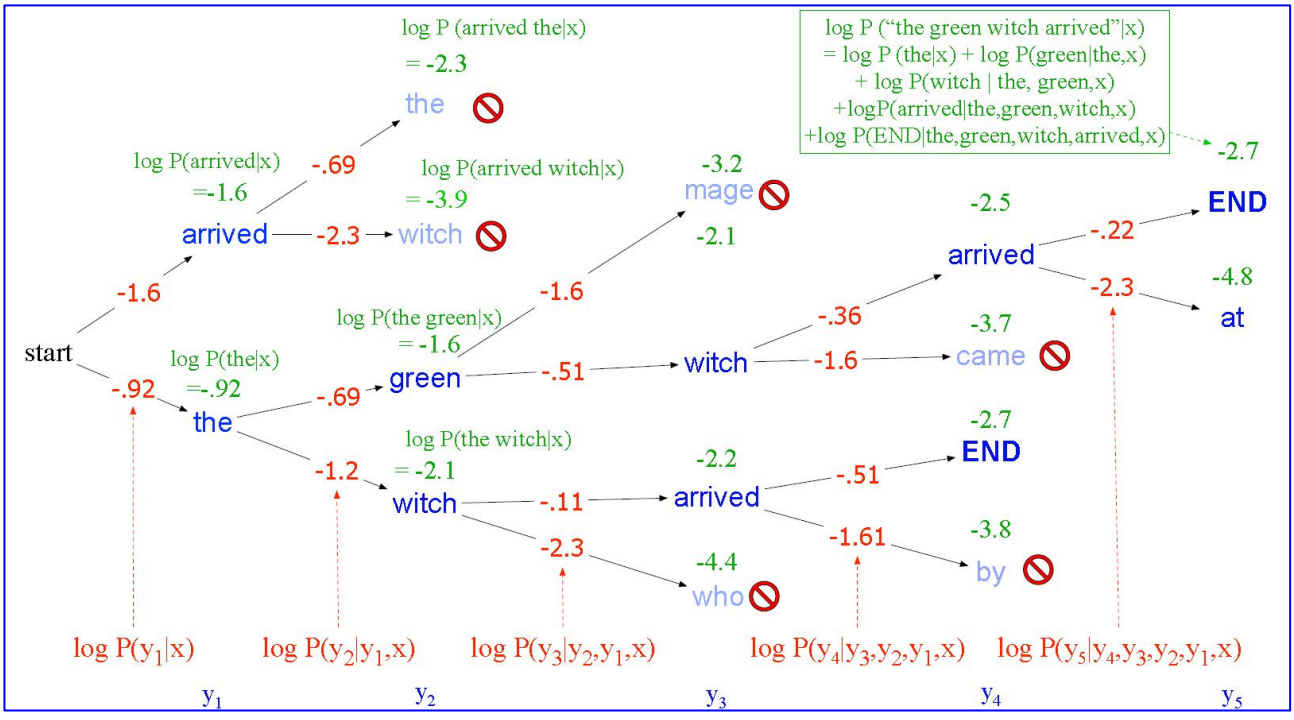

图 11-13：波束搜索解码评分

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!

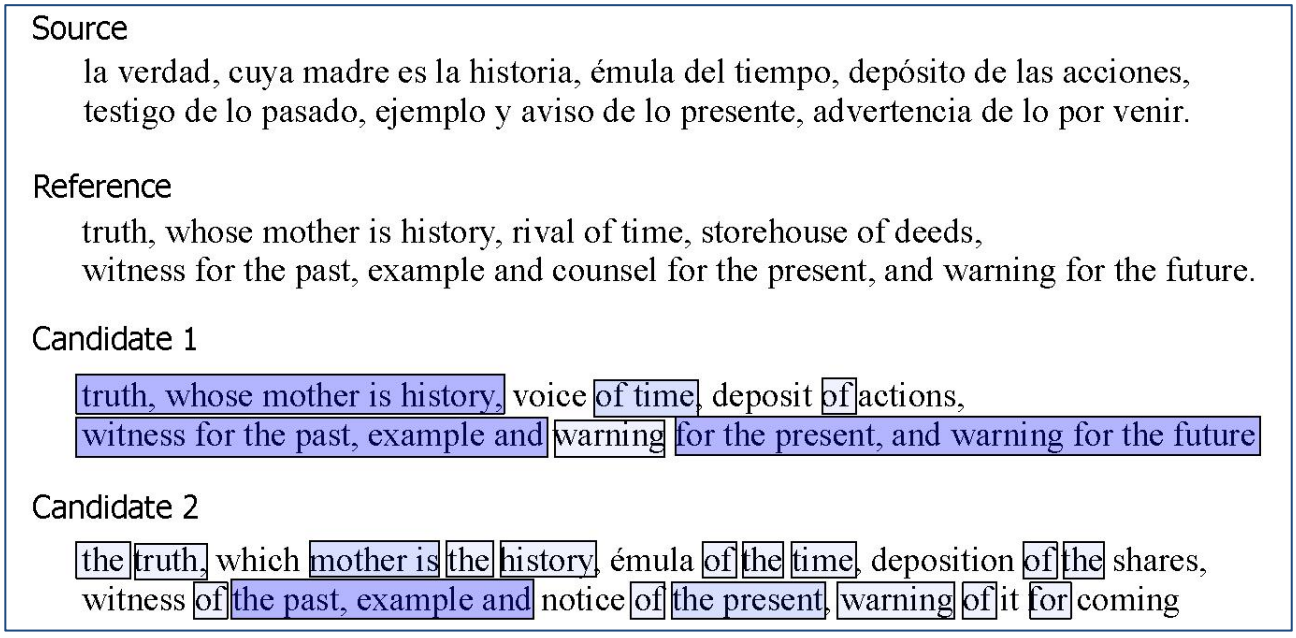

图 11-16：BLEU 的直觉

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。

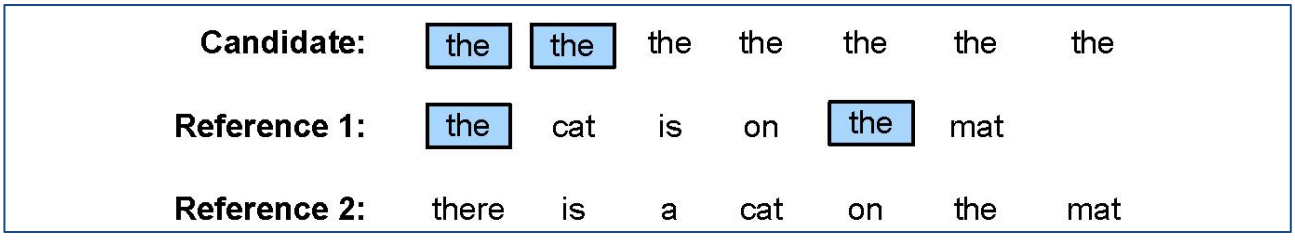

图 11-17：BLEU 使用的病态例句

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)：

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j \qquad P_{\text{BERT}} = \frac{1}{|\tilde{x}|}\sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j \tag{11.27}$$

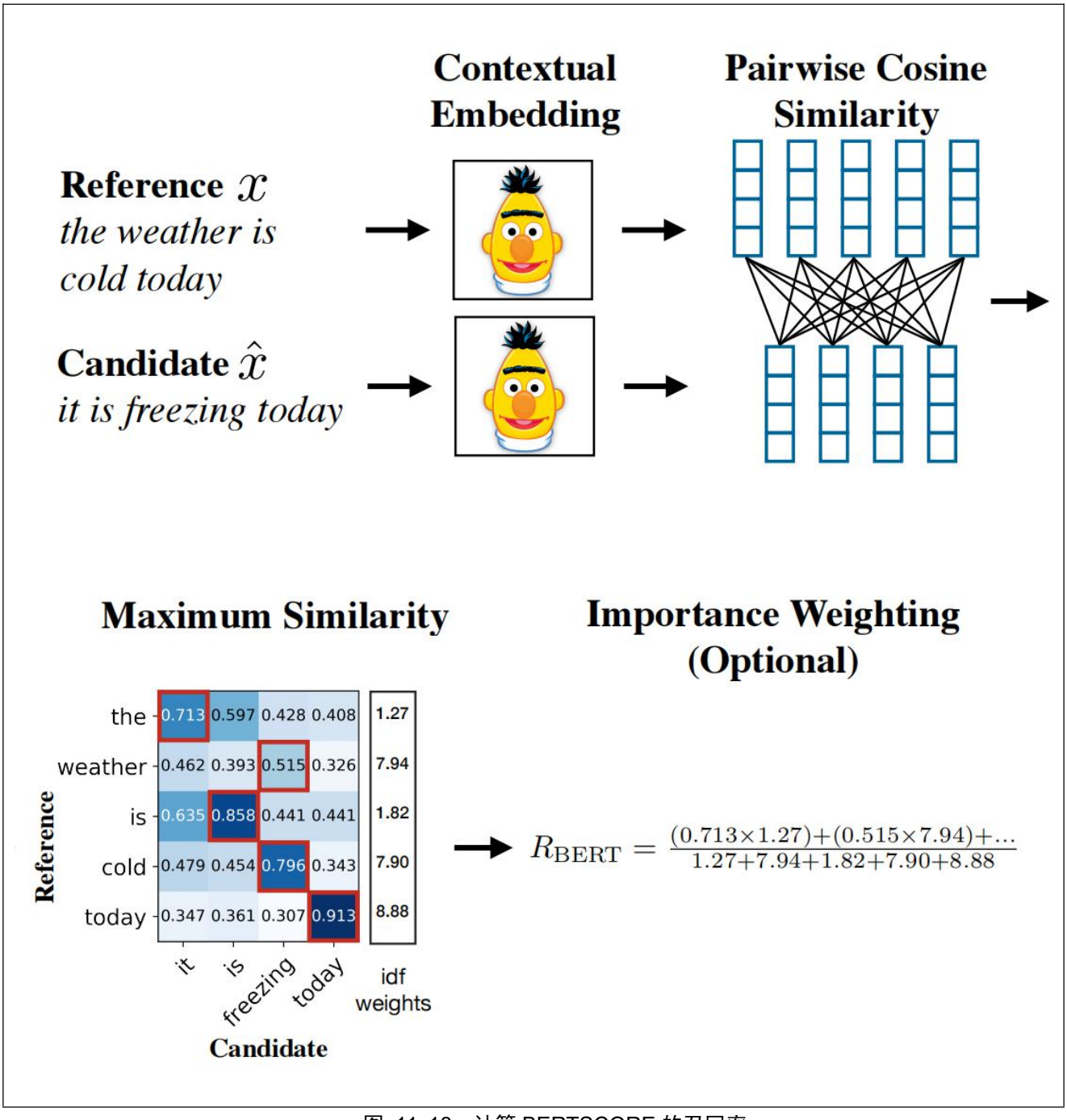

图 11-18：计算 BERTSCORE 的召回率

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。

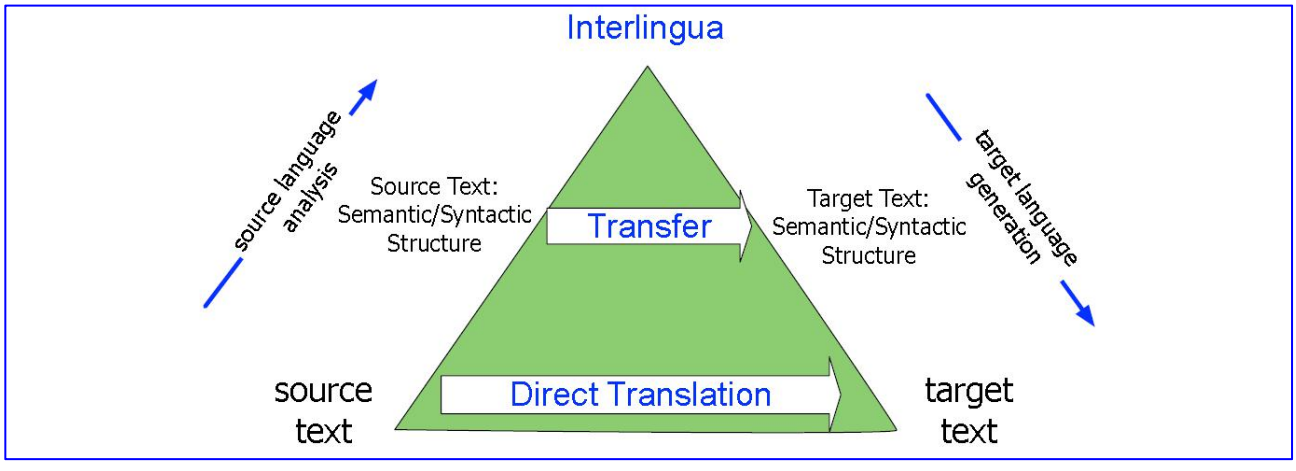

图 11-20：**沃夸**三角形

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)