自然语言处理导论

张奇 桂韬 黄萱菁

2022 年 12 月 26 日

.

数学符号

数与数组

α	标量
$\boldsymbol{\alpha}$	向量
$\boldsymbol{A}$	矩阵
$\mathbf{A}$	张量
$\boldsymbol{I}_n$	n 行 n 列单位矩阵
$\boldsymbol{v}_w$	单词 w 的分布式向量表示
$\boldsymbol{e}_w$	单词 w 的独热向量表示：[0,0,...,1,0,...0]，w 下标处元素为 1

索引

α_i	向量 $\boldsymbol{\alpha}$ 中索引 i 处的元素
α_{-i}	向量 $\boldsymbol{\alpha}$ 中除索引 i 之外的元素
$w_{i:j}$	序列 w 中从第 i 个元素到第 j 个元素组成的片段或子序列
A_{ij}	矩阵 $\boldsymbol{A}$ 中第 i 行、第 j 列处的元素
$\boldsymbol{A}_{i:}$	矩阵 $\boldsymbol{A}$ 中第 i 行
$\boldsymbol{A}_{:j}$	矩阵 $\boldsymbol{A}$ 中第 j 列
$\boldsymbol{A}_{ijk}$	三维张量 $\mathbf{A}$ 中索引为 (i,j,k) 处元素
$\mathbf{A}_{::i}$	三维张量 $\mathbf{A}$ 中的一个二维切片

集合

$\mathbb{A}$	集合
$\mathbb{R}$	实数集合
$0, 1$	含 0 和 1 的二值集合
$0, 1, ..., n$	含 0 和 n 的正整数的集合
$[a, b]$	a 到 b 的实数闭区间
$(a, b]$	a 到 b 的实数左开右闭区间

线性代数

$\boldsymbol{A}^\top$	矩阵 $\boldsymbol{A}$ 的转置
$\boldsymbol{A} \odot \boldsymbol{B}$	矩阵 $\boldsymbol{A}$ 与矩阵 $\boldsymbol{B}$ 的 Hardamard 乘积
det$\boldsymbol{A}^\top$	矩阵 $\boldsymbol{A}$ 的行列式
$[\boldsymbol{x}; \boldsymbol{y}]$	向量 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的拼接
$[\boldsymbol{U}; \boldsymbol{V}]$	矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{V}$ 沿行向量拼接
$\boldsymbol{x} \cdot \boldsymbol{y}$ 或 $\boldsymbol{x}^\top \boldsymbol{y}$	向量 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的点积

微积分

$\frac{\mathrm{d}y}{\mathrm{d}x}$	y 对 x 的导数
$\frac{\partial y}{\partial x}$	y 对 x 的偏导数
$\bigtriangledown_{\boldsymbol{x}} y$	y 对向量 $\boldsymbol{x}$ 的梯度
$\bigtriangledown_{\boldsymbol{X}} y$	y 对矩阵 $\boldsymbol{X}$ 的梯度
$\bigtriangledown_{\mathbf{X}} y$	y 对张量 $\mathbf{X}$ 的梯度

概率与信息论

$a \perp b$	随机变量 a 与 b 独立
$a \perp b \mid c$	随机变量 a 与 b 关于 c 条件独立
$P(a)$	离散变量概率分布
$p(a)$	连续变量概率分布
$a \sim P$	随机变量 a 服从分布 P
$\mathbb{E}_{x \sim P}[f(x)]$ 或 $\mathbb{E}[f(x)]$	$f(x)$ 在分布 $P(x)$ 下的期望
Var$(f(x))$	$f(x)$ 在分布 $P(x)$ 下的方差
Cov$(f(x), g(x))$	$f(x)$ 与 $g(x)$ 在分布 $P(x)$ 下的协方差
$H(f(x))$	随机变量 x 的信息熵
$D_{KL}(P \parallel Q)$	概率分布 P 与 Q 的 KL 散度
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	均值为 $\boldsymbol{\mu}$、协方差为 $\boldsymbol{\Sigma}$ 的高斯分布

数据与概率分布

$\mathbb{X}$	数据集
$\boldsymbol{x}^{(i)}$	数据集中第 i 个样本（输入）
$\boldsymbol{y}^{(i)}$ 或 $y^{(i)}$	第 i 个样本 $\boldsymbol{x}^{(i)}$ 的标签（输出）

函数

$f : \mathcal{A} \longrightarrow \mathcal{B}$	由定义域 $\mathcal{A}$ 到值域 $\mathcal{B}$ 的函数（映射）f
$f \circ g$	f 与 g 的复合函数
$f(\boldsymbol{x}; \boldsymbol{\theta})$	由参数 $\boldsymbol{\theta}$ 定义的关于 $\boldsymbol{x}$ 的函数（也可以直接写作 $f(\boldsymbol{x})$，省略 $\boldsymbol{\theta}$）
$\log x$	x 的自然对数函数
$\sigma(x)$	Sigmoid 函数 $\dfrac{1}{1+exp(-x)}$
$\|\|\boldsymbol{x}\|\|_p$	$\boldsymbol{x}$ 的 L^p 范数
$\|\|\boldsymbol{x}\|\|$	$\boldsymbol{x}$ 的 L^2 范数
$\mathbf{1}^{\text{condition}}$	条件指示函数：如果 condition 为真，则值为 1；否则值为 0

本书中常用写法

- 给定词表 $\mathbb{V}$，其大小为 $|\mathbb{V}|$
- 序列 $x = x_1, x_2, ..., x_n$ 中第 i 个单词 x_i 的词向量 $\boldsymbol{v}_{x_i}$
- 损失函数 $\mathcal{L}$ 为负对数似然函数：$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{(x,y)} \log P(y|x_1...x_n)$
- 算法的空间复杂度为 $\mathcal{O}(mn)$

目 录

8. 机器翻译

根据联合国统计，目前世界上正在使用的语言约有 6000 种，教育系统和公共领域中使用到语言也有数百种之多。我们不可能掌握如此之多的语言，即便是熟练使用联合国规定的包括阿拉伯文、中文、英文、法文、俄文和西班牙文在内的六种正式语言都非常困难。因此，1947 年当在第一台通用计算机 ENIAC 才刚刚面世一年，Warren Weaver 就提出了利用计算机翻译人类语言的可能。**机器翻译**（Machine Translation）是指利用计算机将一种语言（源语言）自动翻译为另外一种语言（目标语言）的过程。机器翻译是自然语言处理中最重要的任务之一。

本章首先介绍机器翻译的基本概念和常见任务，在此基础上介绍基于统计的机器翻译方法和基于神经网络的机器翻译方法，最后介绍机器翻译的评测方法和机器翻译常见的语料库和评测集合。

8.1 机器翻译概述

机器翻译（Machine Translation，简称 MT）这一概念拥有很长的历史，领域相关的研究最早可以被追溯到 17 世纪。1629 年，Descartes 等人就提出使用统一符号表达不同语言中的同一概念的语义。现代机器翻译的研究始于上世纪五十年代，Bar-Hillel 等先驱在 1951 开始了对机器翻译的研究并在 1952 年组织了第一届国际机器翻译会议（International Conference on Machine Translation）。机器翻译的任务定义相对简单，通过计算机将源语言（Source Language）翻译为目标语言（Target Language）。如图8.1所示，机器翻译系统将中文翻译为英文，在这个例子中中文为源语言，英文为目标语言。

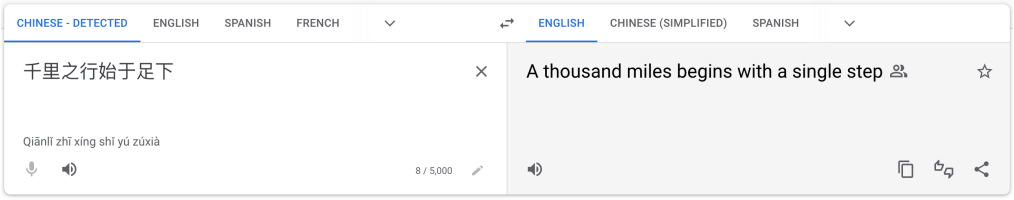


图 8.1　机器翻译系统样例

8.1.1 机器翻译发展历程

机器翻译的发展历程基本代表了自然语言处理领域的发展过程，迄今为止，机器翻译的研究与发展大体上经历了三次主要的浪潮，也即基于规则的机器翻译[1]，基于统计的机器翻译[2] 和基于神经网络的机器翻译[3]。

基于规则的机器翻译：基于规则的机器翻译是机器翻译问题的第一套解决方案，它基于“每一种语义在不同的语言当中都存在与其相对应的符号”这一假设。这一假设是具备一定的合理性的，因为对于某种语言中的大多数单词通常都能够在另一种语言当中找到表达相同含义的对应的单词。在这类方法当中，翻译过程通常被看作一个源语言的词替换过程。之所以被称为“基于规则的方法”，是因为同一种语义在不同的语言当中通常会以不同的词序去表达，词替换过程相对应地需要两种语言的句法规则作为指导，源语言中的每一个单词需要被放置在目标语言中相对应的位置。基于规则的机器翻译方法的理论非常简洁清晰，但在实践中的性能却不尽如人意。这是由于选择与给定源语言相适配的句法规则在计算上非常低效。同时，为了应对多样的语言现象，语言学家们设计了规模庞大的句法规则，这些规则很难被有效地组织，甚至会出现不同规则相互矛盾的情况。基于规则的方法最严重的缺陷在于其完全地忽略了翻译过程中对上下文信息的建模，这使得基于规则的翻译模型的鲁棒性不佳。Marvin Minsky 在 1966 年给出了非常著名的一个例子来阐述这一问题：

“*The pen is in the box*”

“*The box is in the pen*”

上述两个句子具有相同的句法结构，第一个句子很容易理解。但第二个句子非常令人困惑，由于在英文中“pen”本身是一个多义词，除了“笔”之外它还有“栅栏”的意思。但对于计算机来说很难直接将“pen”对应到“栅栏”的含义上，进而产生错误的翻译结果。

基于统计的机器翻译：在过去的 20 年以来，统计机器翻译（Statistical Machine Translation，SMT）已经成为机器翻译领域的主流方法，并在工业界得到了广泛地实际应用。和基于规则的机器翻译方法不同，统计机器翻译完全从数据驱动的角度建模机器翻译任务。具体来说，通过对双语语料库的统计找到表达相同含义的单词或短语。给定一个源语言句子，统计机器翻译首先将其分割成若干个子句，接下来每个部分可以被目标语言的单词或短语替代。统计机器翻译中最主流的方法是基于短语的统计机器翻译（Phrase-based SMT），总体包含预处理、句子对齐、词对齐、短语抽取、短语特征准备、语言模型训练等步骤。基于短语的统计机器翻译方法的最重要的部件是基于短语的词典，它将源语言和目标语言中的短语配对。这一词典通过双语语料库训练数据构建得到。在这一过程中，翻译模型能够利用短语内部的上下文信息，因而优于简单的单词到单词的翻译方法。

基于神经网络的机器翻译：神经网络方法在机器翻译任务上的应用可以被追溯到上世纪八九十年代[4, 5]。但受限于当时的计算资源和数据规模的限制，神经网络方法的性能差强人意，故而其发展停滞了很多年。近年来，伴随着深度学习技术的广泛应用，越来越多的自然语言处理任务实现了极大地性能提升。基于深度神经网络的机器翻译模型也逐渐受到了许多关注。神经网络方法

在机器翻译任务上的第一次成功应用是 Kalchbrenner 和 Blunsom 等人提出的基于递归神经网络的方法，这在当时的机器翻译领域是一种全新的概念。相比于其他模型，神经机器翻译模型在对语言学知识的依赖更少的前提下达到与之前方法相媲美的性能。从这之后，神经机器翻译方法正式走上了历史舞台，大量的研究者开始在这类方法上进一步研究和改进。到今天为止，神经机器翻译被广泛地应用在不同的工业场景下[6]。

8.1.2 机器翻译现状与挑战

机器翻译在经历了几十年的发展后，特别是深度神经网络有效应用于机器翻译，使得模型机器翻译的效果有了很大的提高，在特定条件下机器翻译的效果已经能够达到非常好的效果，甚至可以接近人工翻译效果。然而，在开放环境中，翻译效果还远没有达到直接使用的程度。根据机器翻译权威评测 WMT21[7] 给出的人工评测结果，在新闻领域最好的中文到英文翻译系统评分也仅有 75 分左右（满分 100 分）。机器翻译完全代替人工翻译还有很长的道路。以王佐良先生对 Samuel Ullman 所著的《Youth》译文为例：

原文： Youth is not a time of life; it is a state of mind; it is not a matter of rosy cheeks, red lips and supple knees; it is a matter of the will, a quality of the imagination, a vigor of the emotions; it is the freshness of the deep springs of life.

机器翻译结果： 青春不是生命的时光；这是一种心态；这不是红润的脸颊、红润的嘴唇和柔软的膝盖；这是意志的问题，是想象力的质量，是情感的活力；它是生命深泉的清新。

王佐良译文： 青春不是年华，而是心境；青春不是桃面、丹唇、柔膝，而是深沉的意志，恢宏的想象，炙热的感情；青春是生命的深泉在涌流。

可以看到虽然机器翻译的结果从词语到语法都已经达到了很好的效果，甚至一些相对不常见的句式结构也能较好的翻译。但是整个翻译的效果距离人工翻译“信达雅”的要求还是有很大的差距。上例中，虽然每个翻译后的句子都符合语法，但是句子之间的意义连贯性以及词语的搭配还是会让人难以理解。当然，这里给出的例子是相对困难的，绝大部分人工翻译也很难能够达到这种程度。对于新闻的翻译，相较于上述例子来说就简单很多。与人工翻译相比，机器翻译面对互联网上每天数以亿计的新闻和短消息的处理就体现出了巨大的优势。

机器翻译虽然经过很多年的发展，目前在特定应用场景下已经能够有很好的效果，但是仍然面向如下挑战：

(1) 自然语言复杂度高。自然语言具有高度的复杂性、概括性以及多变性，并且是在不断发展过程中。虽然目前已经有深度神经网络模型参数量达到了 1.75 万亿，但是相比于自然语言的复杂度来说还是相差很多。在第 6 章语言模型中介绍过，按照《现代汉语词典（第七版）》包含 7 万词条，句子长度按照 20 个词计算，参数量达到 7.9792×10^{96} 的天文数字。更不要说，语言还是在动态发展中的，在 2018 年以前 Bert 代表的是《芝麻街》中的卡通布偶“伯特”，而现在计算机领域多

是指代预训练语言模型 BERT。在数据驱动的统计机器翻译和神经机器翻译模型中，如何才能让模型具备超大规模参数空间的建模能力和持续学习能力都是十分巨大的挑战。

(2) 翻译结果不可解释。目前机器翻译算法多采用数据驱动的方法，所采用的模型通常不具备可解释性。这就造成了机器翻译算法虽然给出了翻译结果，并且效果可能还很好，但是其对语言的理解和翻译过程与人的理解和翻译过程完全不同。机器翻译算法的目标仅是根据人工定义的目标函数进行优化，使得人们无法理解机器翻译算法的过程，不能够对算法进行解释。这就带来一系列的严重问题，包括我们不知道机器翻译算法在什么时候会出错，会出现什么样的错误，为什么会出错，如何才能修正和改进等等。这也使得我们很难在关键需求中完全依赖机器翻译。试想一下，对于一个我们完全不认识的语言“БЕЖАТЬ ВПЕРЕД”（俄语），机器翻译给出的结果是“向前跑”，我们无法确认模型结果正确与否。因此，如何构建具备可解释性的机器翻译模型也是需要解决的难点。

(3) 翻译结果评测困难。语言有很大的灵活性和多样性，同样一句话可以有非常多种的翻译方法。对机器翻译性能进行评测可以采用人工评测和半自动评测方法。人工评测虽然是相对准确的一种方式，但是其成本高昂，根据艾伦人工智能研究院（AI2）GENIE 人工评测榜单给出的数据，针对 800 条机器翻译效果进行评测需要花费约 80 美元[8]。如果采用半自动评测方法，利用人工给定的标注翻译结果和评测函数可以快速高效的给出评测结果，但是目前半自动评测结果与人工评测的一致性还亟待提升。对于用词差别很大，但是语义相同的句子的判断本身也是自然语言处理领域的难题。如果有效的评测翻译结果是面临的挑战。

8.2 基于统计的机器翻译方法

机器翻译任务到今天已经经历了长达数十年的发展历史。在很长一段时间之内，基于统计的机器翻译方法在学术界受到了许多关注并在工业界得到了广泛地应用。尽管这些模型在当下已经不再是人们关注的焦点，回顾这些方法仍然对我们完整地了解机器翻译的过去以及展望未来的研究具有十分重要的意义。本节将以在统计机器翻译中具有里程碑式意义的 IBM 模型为例，简要阐述统计机器翻译方法的基础概念、基本假设和流程。

8.2.1 任务定义与基本问题

IBM 机器翻译模型构建在噪声信道模型（Noise Channel Model）的基础之上，其模型基本框架如图8.2所示。源语言句子 $\boldsymbol{s}$ 和候选的目标语言句子 $\boldsymbol{t}_i$ 通过一个噪声信道相连，在已知 $\boldsymbol{s}$ 和噪音信道性质的前提之下，就能够得到信源，也即目标语言的分布 $P(\boldsymbol{t}|\boldsymbol{s})$。机器翻译的过程本质上就是在给定源语言句子 $\boldsymbol{s}$ 的前提下，从分布 $P(\boldsymbol{t}|\boldsymbol{s})$ 找到最有可能的目标语言 $\boldsymbol{t}$ 作为输出，这一搜索过程也被称为解码。

$$\hat{t} = \arg\max_{t} P(\boldsymbol{t}|\boldsymbol{s}). \tag{8.1}$$

在上述噪声信道基础框架下，统计机器翻译任务需要解决如下三个核心的基本问题：

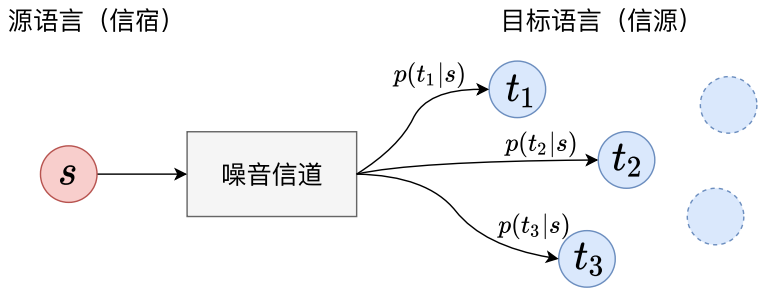


图 8.2 噪声信道模型

问题 1：建模 如何通过易于被计算机处理的数学模型对 $P(\boldsymbol{t}|\boldsymbol{s})$ 进行合理地建模以刻画源语言和目标语言之间的关系。

问题 2：训练 如何从给定的平行语料库（即源语言-目标语言对组成的语料集合）中获得最优的模型参数。

问题 3：解码 如何从模型 $P(\boldsymbol{t}|\boldsymbol{s})$ 中搜索出最优的目标语言序列 $\boldsymbol{t}$。

接下来，将依次介绍 IBM 模型是如何建模并解决上述三个问题的。首先，统计机器翻译模型的核心在于对 $P(\boldsymbol{t}|\boldsymbol{s})$ 的定义，这一定义决定了模型性能的上限并且也是后续训练和解码的基础。IBM 模型通过贝叶斯公式对这一翻译概率做如下变换：

$$P(\boldsymbol{t}|\boldsymbol{s}) = \frac{P(\boldsymbol{s},\boldsymbol{t})}{P(\boldsymbol{s})} = \frac{P(\boldsymbol{s}|\boldsymbol{t})P(\boldsymbol{t})}{P(\boldsymbol{s})} \tag{8.2}$$

通过上述变换，翻译模型 $P(\boldsymbol{t}|\boldsymbol{s})$ 被分解为了三个部分：(1) 从目标语言指向源语言的翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$；(2) 目标语言的语言模型 $P(\boldsymbol{t})$；(3) 源语言序列语言模型 $P(\boldsymbol{s})$。需要注意的是，通过贝叶斯变换 $P(\boldsymbol{t}|\boldsymbol{s})$ 和 $P(\boldsymbol{s}|\boldsymbol{t})$ 只是翻译的方向不同，建模难度并没有下降。其核心是为了引入目标语言的语言模型。这是由于 IBM 模型本质上是一种基于词的统计机器翻译模型，仅通过翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 很难有效地建模目标语言单词之间的相对位置关系，也即目标语言序列的流畅程度。相对应地，引入语言模型 $P(\boldsymbol{t})$ 可以有效地缓解上述问题。此外，$P(\boldsymbol{s})$ 是一个不变量，因此它不会影响到 $\frac{P(\boldsymbol{s}|\boldsymbol{t})P(\boldsymbol{t})}{P(\boldsymbol{s})}$ 的相对大小，也就不会影响到最终的解码过程，在建模的过程当中 $P(\boldsymbol{s})$ 通常不需要被计算，可以省略：

$$\hat{t} = \arg\max_t P(\boldsymbol{t}|\boldsymbol{s}) = \arg\max_t \frac{P(\boldsymbol{s}|\boldsymbol{t})P(\boldsymbol{t})}{P(\boldsymbol{s})} = \arg\max_t P(\boldsymbol{s}|\boldsymbol{t})P(\boldsymbol{t}) \tag{8.3}$$

基于上述分析，IBM 模型的建模问题转换为如何建模翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 以及语言模型 $P(\boldsymbol{t})$。翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 主要用于衡量源语言和目标语言之间的匹配程度。然而，自然语言拥有极其庞大的潜在的组合方式。假设某种语言对应的词表大小为 10000，那么一个简单的长度为 10 的句子就对

应着 $10000^{10} = 10^{40}$ 种不同的组合方式。基于任何已有的平行语料库直接在句子层级对上述翻译概率进行估计都会面临严重的数据稀缺问题。因此，IBM 模型将句子层级的翻译概率进一步拆解为单词级别的对应关系的组合，从而缓解上述数据稀疏的问题，这一拆解过程又被称为**词对齐**。

词对齐作为 IBM 模型构建的重要基础之一，描述了目标语言和源语言之间单词级别的对应关系。以图8.3中的对齐实例 1 为例，给定源语言文本“机器 翻译”，对应的目标语言翻译为“Machine Translation”。其中，“机器”一词对应“Machine”而“翻译”一词对应“Translation”。使用记号 $\boldsymbol{a} = \{a_1, ..., a_m\}$ 表示这种对应关系，其中 a_j 表示源语言中的单词 s_j 和目标语言中的单词 t_{a_j} 存在对应关系。举例来说，在对齐实例 1 中，$a_1 = 1, a_2 = 2$。

为了建模方便，IBM 模型对词对齐做了如下两个限制：

- 对于每一个源语言单词，至多只能对齐到一个目标语言单词上。图8.3的对齐实例 2 中，源语言单词“机器”同时对应到了两个目标语言单词“Machine”和“Translation”，这就违反了上述 IBM 模型假设。而其余的对齐实例均满足这一假设。
- 存在一些源语言单词，它们可以对齐到一个额外增设的虚拟目标语言单词“Null”上，也即对空。图8.3的对齐实例 4 中的“机器”一词就对应到了目标语言的“Null”上。对空情况的额外考虑并不是没有意义的。事实上，对空的现象在翻译的过程当中频繁出现，如虚词的翻译。

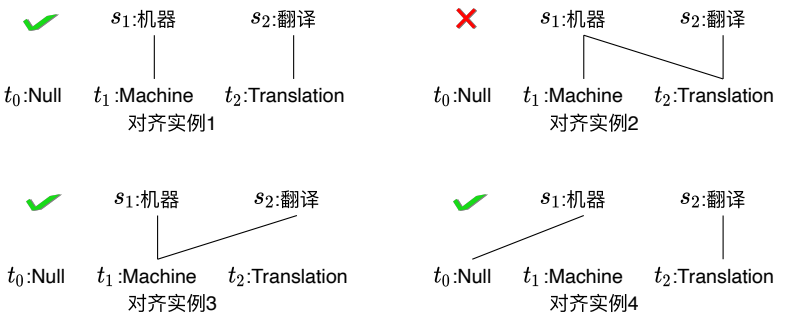


图 8.3 IBM 词对齐实例

IBM 模型认为句子级别的翻译概率可以通过单词级别的翻译概率组合而成，并将词对齐作为一种隐变量整合到翻译概率的建模过程中。这样，原本较为困难的句子级别的建模问题就被分解为一个分步学习的问题：

$$P(\boldsymbol{s}|\boldsymbol{t}) = \sum_{\boldsymbol{a}} P(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{t}), \tag{8.4}$$

此处 $\boldsymbol{s} = \{s_1, s_2, ..., s_m\}$ 表示一个长度为 m 的源语言序列,$\boldsymbol{t} = \{t_1, t_2, ...t_l\}$ 表示一个长度为 l 的目标语言序列，$\boldsymbol{a} = \{a_1, a_2, ..., a_m\}$ 表示源语言中每一个单词 s_j 对应的目标语言单词序号 a_j。直接建模 $P(\boldsymbol{s}|\boldsymbol{t})$ 仍然非常复杂，为了解决这个问题，IBM 模型对上述概率通过链式法则做了进一步展

开并为后续的简化做了准备。

$$P(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{t}) = P(m|\boldsymbol{t}) \prod_{j=1}^{m} P(a_j|\boldsymbol{a}_1^{j-1},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t}) P(s_j|\boldsymbol{a}_1^{j},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t}), \tag{8.5}$$

其中 $\boldsymbol{a}_1^{j-1}$ 表示源语言序列中前 $j-1$ 个单词的词对齐，$\boldsymbol{s}_1^{j-1}$ 表示源语言序列中的前 $j-1$ 个单词。这一展开看似较为复杂，实际上每个部分都具有较为清晰的物理含义。给定一个目标语言序列 $\boldsymbol{t}$，我们首先通过概率 $P(m|\boldsymbol{t})$ 估计源语言序列的长度 m。接下来，我们通过 m 次循环从左向右依次生成源语言序列和它们的词对齐。在第 j 次循环当中，我们首先通过目标语言序列 $\boldsymbol{t}$，前 $j-1$ 次循环中生成的词对齐序列 $\boldsymbol{a}_1^{j-1}$ 以及源语言序列 $\boldsymbol{s}_1^{j-1}$ 产生当前位置的词对齐 a_j，也即 $P(a_j|\boldsymbol{a}_1^{j-1},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t})$ 这一部分。接下来结合 a_j 进一步生成当前位置的源语言单词 s_j，也即 $P(s_j|\boldsymbol{a}_1^{j},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t})$ 这一部分。至此，翻译概率的建模实际上就被转换为源语言文本和词对齐的生成问题。但是，任然还存在两个迫切需要解决的问题：

- 为了最终实现对翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 的建模，在公式8.4中需要对所有可能的词对齐进行求和。然而，可能的词对齐的数量随着源语言序列的长度呈指数级别增长，如何计算这一求和式是第一个需要被解决的为题。
- 公式8.5通过链式分解为建模 $P(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{t})$ 提供了一种可行的方向，然而如何通过目标语言序列估计源语言序列的长度，也即 $P(m|\boldsymbol{t})$，如何建模源语言 $P(s_j|\boldsymbol{a}_1^{j},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t})$ 以及词对齐的生成过程 $P(a_j|\boldsymbol{a}_1^{j-1},\boldsymbol{s}_1^{j-1},m,\boldsymbol{t})$ 尚待解决。

对于上述两个问题的解决实际上对应着 5 个不同的 IBM 模型，我们将在后续的章节中详细阐述它们是如何简化并解决上述两个问题以及如何如何基于构建的翻译模型从平行语料库当中学习最优的参数。除了翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 之外，语言模型 $P(\boldsymbol{t})$ 是 IBM 模型的另外一个重要的组成部分，对于语言模型的详细介绍可以参考本书的第六章。在这里，我们仅给出一个简略的回顾。

为了衡量生成译文的流畅程度，IBM 模型引入了 n-gram 语言模型。它使用概率化的方法描述了句子的生成过程。以 2-gram 语言模型为例，一个目标语言序列的生成概率可以按照下式评估：

$$\begin{aligned} P_{\text{lm}}(\boldsymbol{t}) &= P_{\text{lm}}(t_1,...,t_l) \\ &= P(t_1) \times P(t_2|t_1) \times P(t_3|t_2) \times ... \times P(t_l|t_{l-1}) \end{aligned} \tag{8.6}$$

上述的每一个 $P(t_i|t_{i-1})$ 均对应着语言模型的不同参数。这些参数的估计可以通过对语料库的极大似然估计完成。

$$P(t_i|t_{i-1}) = \frac{\text{单词对} < t_{i-1}, t_i > \text{出现的次数}}{\text{单词 } t_{i-1} \text{ 出现的次数}} \tag{8.7}$$

在完成了对翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 以及语言模型 $P(\boldsymbol{t})$ 的建模与优化之后，下一个需要被解决的问题就是解码，也即 $\arg\max_{\boldsymbol{t}} P(\boldsymbol{s}|\boldsymbol{t}) \cdot P(\boldsymbol{t})$ 的问题。在这里，给出一种简单的贪婪解码算法作为例子，算法8.1给出了具体的过程。

代码 8.1: 贪婪解码算法

输入: 源语言句子序列 $\boldsymbol{s} = \{s_1, s_2, ..., s_m\}$
输出: 解码出的目标语言序列 $\boldsymbol{t} = \{t_1, t_2, ..., t_l\}$
初始化一个空集 $\boldsymbol{t}_{best} = \emptyset$ 记录解码出的最优目标语言序列;
初始化一个布尔数组 u，记录每个源语言单词是否已经被解码;
对于每个源语言单词 s_i，获取其对应单词级别目标语言候选 π;
for $i = 1$ **to** m **do**
    初始化一个空集 $\boldsymbol{h} = \emptyset$ 记录当前最优解码序列的扩展;
    for $j = 1$ **to** m **do**
        if $u[j]$ 为假，也即当前单词没有被解码过 **then**
            将当前单词的候选翻译 $\pi[j]$ 添加到 $\boldsymbol{h}$ 当中;
        end
    end
    从当前的最优翻译扩展候选 $\boldsymbol{h}$ 中找到最优的翻译结果，并记为 $\boldsymbol{t}_{best}$;
    这一最优结果所对应的源语言单词被记录为已解码 $u[j] = True$;
end
return $\boldsymbol{t}_{best}$

8.2.2 IBM 模型 I

上一节介绍了IBM模型如何将翻译问题转化为一个机器学习问题。然而,在推导出的公式8.5中,仍然存在两个问题需要解决：(1) 如何高效地计算对不同的词对齐序列求和的问题；(2) 如何合理地简化并建模公式8.5右侧的若干概率。接下来，我们详细阐述 IBM 模型 I 是如何解决这两个问题的。

从公式8.5右侧三个概率的化简及建模入手，首先是如何基于目标语言序列估计源语言序列的长度 $P(m|\boldsymbol{t})$。IBM 模型 I 假定源语言句子序列长度的生成概率服从均匀分布，即:

$$P(m|\boldsymbol{t}) \equiv \epsilon \tag{8.8}$$

其中 ϵ 是一个常量,在实际的建模和优化过程中,它通常被取特定的值来保证概率的归一化。源语言中的每一个单词被认为是等可能地和目标语言中的所有单词对齐,也即对齐概率 $P(a_j|\boldsymbol{a}_1^{j-1}, \boldsymbol{s}_1^{j-1}, m, \boldsymbol{t})$ 按照如下方式进行建模:

$$P(a_j|\boldsymbol{a}_1^{j-1}, \boldsymbol{s}_1^{j-1}, m, \boldsymbol{t}) \equiv \frac{1}{l+1} \tag{8.9}$$

其中 $l+1$ 表示目标语言序列长度加上一个保留的虚拟单词 “Null”。当对齐关系明确之后，IBM 模

型 I 假设当前时刻源语言单词 s_j 的生成只依赖于和它对齐的目标语言单词 t_{a_j}：

$$P(s_j|\boldsymbol{a}_1^j, \boldsymbol{s}_1^{j-1}, m, \boldsymbol{t}) \equiv f(s_j|t_{a_j}) \tag{8.10}$$

上式中的 $f(s_j|t_{a_j})$ 表示给定目标语言单词 t_{a_j} 之后，生成源语言单词 s_j 的概率，这一概率将作为 IBM 模型 I 的参数用于后续的优化过程。经过上述三个部分的化简，翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 可以按照下面的方式得到：

$$\begin{aligned}
P(\boldsymbol{s}|\boldsymbol{t}) &= \sum_{\boldsymbol{a}} P(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{t}) \\
&= P(m|\boldsymbol{t}) \prod_{j=1}^{m} P(a_j|\boldsymbol{a}_1^{j-1}, \boldsymbol{s}_1^{j-1}, m, \boldsymbol{t}) P(s_j|\boldsymbol{a}_1^{j}, \boldsymbol{s}_1^{j-1}, m, \boldsymbol{t}) \\
&= \sum_{\boldsymbol{a}} \epsilon \prod_{j=1}^{m} \frac{1}{l+1} f(s_j|t_{a_j}) \\
&= \sum_{\boldsymbol{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j}),
\end{aligned} \tag{8.11}$$

观察 IBM 模型 I 最终的建模结果可以发现，翻译概率 $P(\boldsymbol{s}|\boldsymbol{t})$ 最终变成了在所有可能的词对齐的基础上，对单词对翻译概率的连乘。因此，可以使用一种十分简单的方式建模原本复杂的公式8.5右侧的形式。

另一个需要解决的问题是对齐序列 $\boldsymbol{a}$ 的求和问题。一个长度为 m 的源语言序列的每一个单词有可能对齐到长度为 $l+1$ 的目标语言的任何一个位置上。因此，对齐序列 $\sum_{\boldsymbol{a}}(\cdot)$ 的求和一共要遍历 $(l+1)^m$ 项。每一项都是一个 m 个单词级别翻译概率的乘积。具体如下所示：

$$P(\boldsymbol{s}|\boldsymbol{t}) = \sum_{\boldsymbol{a}} P(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{t}) = \underbrace{\sum_{\boldsymbol{a}} \frac{\epsilon}{(l+1)^m}}_{(l+1)^m\text{次循环}} \underbrace{\prod_{j=1}^{m} f(s_j|t_{a_j})}_{m\text{次循环}}, \tag{8.12}$$

从上述分析我们能够看出，公式8.11的计算复杂度是 $\mathcal{O}((l+1)^m \cdot m)$，这在源语言序列长度 m 稍

大的情况下几乎是不可能的。因此，在实际的计算过程中，IBM 模型采用如下的计算技巧：

$$
\begin{aligned}
P(\boldsymbol{s}|\boldsymbol{t}) &= \sum_{\boldsymbol{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j}) \\
&= \frac{\epsilon}{(l+1)^m} \sum_{\boldsymbol{a}} \prod_{j=1}^{m} f(s_j|t_{a_j}) \\
&= \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=1}^{l} f(s_j|t_i),
\end{aligned}
\tag{8.13}
$$

此处的计算技巧通过将若干个连乘结果的加和转换为若干加和结果的连乘。计算复杂度由原本的 $\mathcal{O}((l+1)^m \cdot m)$ 降低为 $\mathcal{O}((l+1) \cdot m)$。计算复杂度的下降带来了更高的运行效率，可以通过图8.4对上述计算技巧为什么成立有一个直观的理解。

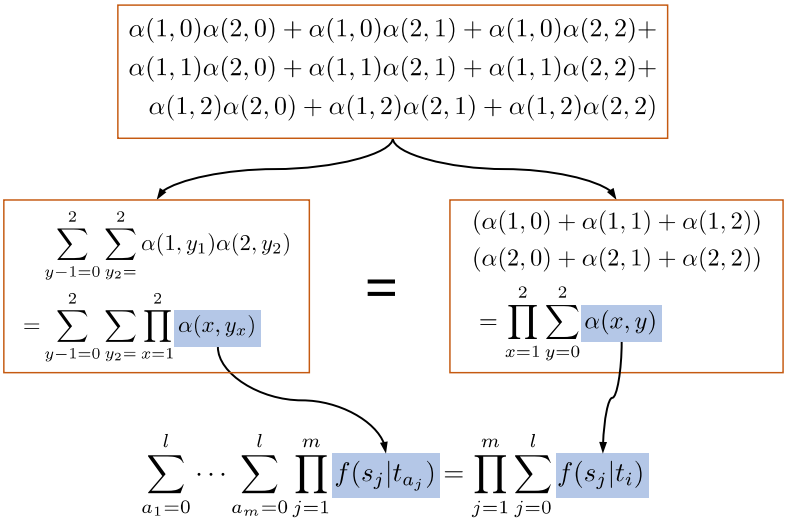


图 8.4　对齐序列计算技巧实例

IBM 模型 I 的优化过程本质上基于极大似然估计的思想，也即找到一组参数 $\hat{\boldsymbol{\theta}} = \{f(s_x|t_y)\}$，

使得模型 $P_{\boldsymbol{\theta}}(\boldsymbol{s}|\boldsymbol{t})$ 能够对训练集中的句对 $(\boldsymbol{s},\boldsymbol{t})$ 输出尽可能大的概率，形式化的描述如下：

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\boldsymbol{s}|\boldsymbol{t}) \\ &= \arg\max_{\boldsymbol{\theta}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i), \\ &\text{s.t. } \forall\, t_y, \sum_{s_x} f(s_x|t_y) = 1,\end{aligned} \tag{8.14}$$

这里的约束表示任意的目标语言单词 t_y 翻译到不同源语言单词的概率求和为 1，也即概率的归一化约束。我们知道，拉格朗日乘子法可以将带有 n 个变量和 m 个约束的优化问题转换为带有 $m+n$ 个变量的无约束优化问题。因此，将这一方法应用到上述有约束的优化目标上，我们能够得到如下无约束的优化目标：

$$L(f,\lambda) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i) - \sum_{t_y} \lambda_{t_y}(\sum_{s_x} f(s_x|t_y) - 1) \tag{8.15}$$

接下来，通过计算函数 $L(f,\lambda)$ 对参数 $f(s_x|t_y)$ 导数为 0 的位置得到其极值点：

$$\begin{aligned}\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)} &= \frac{\partial \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i)}{\partial f(s_u|t_v)} - \frac{\partial \sum_{t_y}\lambda_{t_y}(\sum_{s_x} f(s_x|t_y)-1)}{\partial f(s_u|t_v)} \\ &= \frac{\epsilon}{(l+1)^m}\cdot\frac{\prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i)}{\partial f(s_u|t_v)} - \lambda_{t_v} \\ &= \frac{\epsilon}{(l+1)^m}\cdot\frac{\sum_{j=1}^{m}\delta(s_j,s_u)\cdot\sum_{i=1}^{l}\delta(t_i,t_v)}{\sum_{i=1}^{l} f(s_u|t_i)}\prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i) - \lambda_{t_v}\end{aligned} \tag{8.16}$$

此处的 $\delta(x,y)$ 为指示函数，当 $x=y$ 时，$\delta(x,y)=1$，否则 $\delta(x,y)=0$。当上式为 0 时，$L(f,\lambda)$ 达到极值点，将上式整理得到：

$$f(s_u|t_v) = \lambda_{t_v}^{-1}\underbrace{\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=1}^{l} f(s_j|t_i)}_{P(\boldsymbol{s}|\boldsymbol{t})}\underbrace{\sum_{j=1}^{m}\delta(s_j,s_u)\cdot\sum_{i=1}^{l}\delta(t_i,t_v)}_{(s_u,t_v)\text{的配对次数}}\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}, \tag{8.17}$$

记单词 t_v 翻译到 s_u 的期望频次如下：

$$c_{\mathbb{E}}(s_u|t_v) \equiv \sum_{j=1}^{m}\delta(s_j,s_u)\cdot\sum_{i=1}^{l}\delta(t_i,t_v)\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)} \tag{8.18}$$

则等式8.17可以简写为如下形式：

$$f(s_u|t_v) = \lambda_{t_v}^{-1} P(\boldsymbol{s}|\boldsymbol{t}) c_{\mathbb{E}}(s_u|t_v) = (\lambda'_{t_v})^{-1} c_{\mathbb{E}}(s_u|t_v), \tag{8.19}$$

此处 $(\lambda'_{t_v})^{-1} = \lambda_{t_v}^{-1} P(\boldsymbol{s}|\boldsymbol{t})$，结合翻译概率 $f(s_u|t_v)$ 的归一化约束，容易得到:

$$\lambda'_{t_v} = \sum_{s'_u} c_{\mathbb{E}}(s'_u|t_v) \tag{8.20}$$

这样，原始的等式8.17就被转换为更简洁的如下形式：

$$f(s_u|t_v) = \frac{c_{\mathbb{E}}(s_u|t_v)}{\sum_{s'_u} c_{\mathbb{E}}(s'_u|t_v)}, \tag{8.21}$$

模型的参数通过上式所示的迭代过程逐步优化得到。通常，我们会有一个较大的平行语料数据集 $\mathcal{D} = \{(\boldsymbol{s}_i, \boldsymbol{t}_i)\}_{i=0}^{n}$，单词对之间的期望频次可以通过下式计算：

$$c_{\mathbb{E}}(s'_u|t_v) = \sum_{i=0}^{n} c_{\mathbb{E}}(s_u|t_v; (\boldsymbol{s}_i, \boldsymbol{t}_i)) \tag{8.22}$$

完整的训练过程如下所示：

代码 8.2: IBM 模型 1 训练算法

输入: 平行语料数据集 $\mathcal{D} = \{(\boldsymbol{s}_i, \boldsymbol{t}_i)\}_{i=0}^{n}$
输出: 参数 $f(\cdot|\cdot)$ 的最优值
初始化 $f(\cdot|\cdot)$ // 例如说可以初始化为均匀分布;
while $f(\cdot|\cdot)$ 不收敛 **do**
  for $i = 1$ **to** n **do**
    $c_{\mathbb{E}}(s_u|t_v; (\boldsymbol{s}_i, \boldsymbol{t}_i)) = \sum_{j=1}^{m} \delta(s_j, s_u) \cdot \sum_{i=1}^{l} \delta(t_i, t_v) \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$;
  end
  for 目标语言中所有可能单词 t_v **do**
    $\lambda'_{t_v} = \sum_{s'_u} \sum_{i=0}^{n} c_{\mathbb{E}}(s_u|t_v; (\boldsymbol{s}_i, \boldsymbol{t}_i))$;
    for 源语言中所有可能单词 s_u **do**
      $f(s_u|t_v) = \sum_{i=0}^{n} c_{\mathbb{E}}(s_u|t_v; (\boldsymbol{s}_i, \boldsymbol{t}_i)) \cdot (\lambda'_{t_v})^{-1}$;
    end
  end
end
return $f(\cdot|\cdot)$

8.2.3 IBM 模型 II

IBM 模型 I 虽然很好地简化了模型的复杂程度使得翻译的建模成为了可能，但其中的一些简化与真实情况存在着较大地差异，导致翻译性能受到了较大的限制。最突出的问题是词对齐的概率服从均匀分布。图8.5给出了词对齐的倾向性的简单样例。以单词“翻译”为例，在 IBM 模型 I 当中，它对齐到目标语言序列的 3 个单词上的概率是均等的。但是，很显然“翻译”这一单词应该对齐到目标语言序列当中的第三个位置的“Translation”上，也即 $a_2 = 3$。这种词对齐是具有倾向性的，绝大部分情况下概率不是均等的。

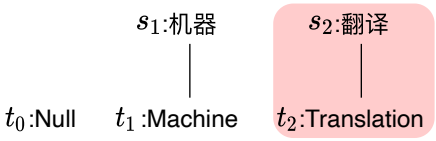


图 8.5 词对齐的倾向性

IBM 模型 II 对这一问题作出了修正，它认为词对齐存在着一定的倾向性。具体来说，IBM 模型 II 假设源语言单词 x_j 的对齐位置 a_j 的生成概率与它所在的位置 j 和源语言序列长度 m 以及目标语言序列长度 l 有关，形式化表示为：

$$P(a_j|\boldsymbol{a}_1^{j-1}, \boldsymbol{s}_1^{j-1}, m, t) \equiv a(a_j|j, m, l), \tag{8.23}$$

此处的 $a(a_j|j,m,l)$ 表示源语言序列中第 j 个位置词对齐的生成概率，它被建模为 IBM 模型 II 中的一个需要学习的参数。除了词对齐假设之外，其余的模型假设均与 IBM 模型 I 相同，将新的词对齐生成概率按照上一小节所述的建模过程能够得到 IBM 模型 II 的翻译建模表达式为：

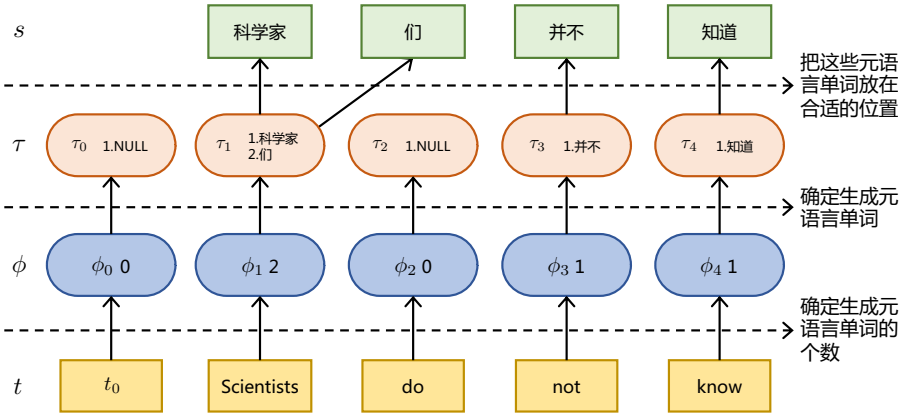
$$P(\boldsymbol{s}|\boldsymbol{t}) = \epsilon \prod_{j=1}^{m} \sum_{i=1}^{l} a(i|j,m,l) f(s_j|t_i), \tag{8.24}$$

8.2.4 IBM 模型 III

IBM 模型 I 和 II 存在的一个共同的问题是将单词翻译的过程建模为了一个独立的过程，这就导致它们不能很好地描述多个源语言单词对齐到同一个目标语言单词的情况。IBM 模型 III 是一种基于繁衍率的模型，可以在一定程度上解决上述问题。这里的**繁衍率**(Fertility)是指每个目标语言单词生成源语言单词的个数。接下来，我们简单描述基于繁衍率的模型的整体翻译流程。

如图8.6所示，模型首先确定每个目标语言单词的繁衍率 ϕ_i，接下来，依据繁衍率确定目标语言对应的源语言单词是什么，这样就得到了每个目标语言单词所对应的源语言单词列表 τ_i。最后将所有单词列表中的单词放置在合适的位置上就得到了源语言单词序列 $\boldsymbol{s}$。以单词 Scientists 的生

成过程为例，从图8.6当中可以看出，它的繁衍率 $\phi_1 = 2$，这意味着在源语言当中它对应两个单词。接下来，确定 Scientists 的源语言单词列表为 $\tau_1 = \{\tau_{11} =$“科学家”$, \tau_{12} =$“们”$\}$。



$$\phi_0 = 0, \phi_1 = 2, \phi_3 = 0, \phi_4 = 1$$

$$\tau_0 = \{\}, \tau_1 = \{\tau_{11} = '科学家', \tau_{12} = '们'\}, \phi_3 = \{\tau_{31} = '并不'\}, \phi_4 = \{\tau_{41} = '知道'\}$$

$$\pi_0 = \{\}, \pi_1 = \{\pi_{11} = 1, \pi_{12} = 2\}, \pi_2 = \{\}, \pi_3 = \{\pi_{31} = 3\}, \pi_4 = \{\pi_{41} = 4\}$$

图 8.6　基于繁衍率的模型示意图

最后，将单词列表中的单词放置在合适的位置，Scientists 对应的单词列表 τ_1 中的单词分别应该被放置在源语言序列中国呢的第 1 和第 2 个位置。因此，$\pi_1 = \pi_{11} = 1, \pi_{12} = 2$。最后，直接给出 IBM 模型 III 的翻译概率的形式，详细过程可以参阅文献 [9] 了解这一模型的建模过程：

$$\begin{aligned} P(\boldsymbol{s}|\boldsymbol{t}) = \sum_{\boldsymbol{a}} [\binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|t_i) \cdot \prod_{j=1}^{m} t(s_j|t_{a_j}) \cdot \prod_{j=1, a_j \neq 0}^{m} d(j|a_j, m, l)], \\ s.t. \quad \sum_{s_x} t(s_x|t_y) = 1, \\ \sum_{j} d(j|i, m, l) = 1, \\ \sum_{\phi} n(\phi|t_y) = 1, \\ p_0 + p_1 = 1, \end{aligned} \tag{8.25}$$

其中 $n(\phi_i|t_i) = P(\phi_i|t_i)$ 指繁衍率的分布，s, t, m, l 分别表示源语言句子、目标语言译文、源语言和目标语言序列长度，ϕ, τ, π 分别表示繁衍率，生成的源语言单词以及它们在源语言序列中的位置，$d(j|i, m, l)$ 通常被称为扭曲度函数。

8.2.5 IBM 模型 IV

当一个目标语言单词对应多个源语言单词时，这些源语言单词往往会构成一个整体，也即一个短语。然而前面所述的三个 IBM 模型并没有对与这种情况做特殊的设计，这就导致了源语言中的单词短语可能会被打散。针对这个问题，IBM 模型 IV 做出了进一步的修正。它将原本单词之间的对应关系拓宽到了**概念** (Concept) 之间的对应。这里的概念是指具有独立语法或语义的一组单词。

IBM 模型将目标语言的概念约束为那些非空对齐的目标语言单词，且要求所有的目标语言概念都只能由一个单词构成。例如，图8.7所给出的单词“mind”是目标语言序列中的第 3 个概念，而单词“my”不是概念。在后面的叙述当中，标记目标语言中第 i 个概念所在的位置为 $[i]$，$\odot_i$ 表示目标语言中第 i 个概念对应的源语言位置的平均值，若这个平均值不是整数则向上取整。

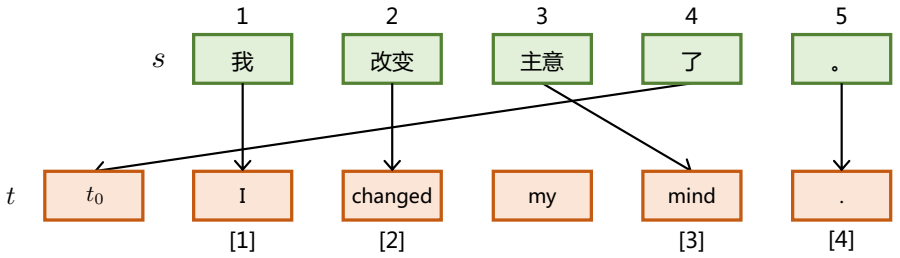


图 8.7　概念对齐示意图

IBM 模型 IV 所做的修正主要体现在扭曲度的建模，对于 $[i]$ 对应的源语言单词列表中的第一个单词 $\tau_{[i]1}$，它的扭曲度计算公式如下：

$$P(\pi_{[i]1} = j|\pi_1^{[i]-1}, \tau_0^l, \phi_0^l, t) = d_1(j - \odot_{i-1}|A(t_{[i-1]}), B(s_j)), \tag{8.26}$$

此处的 π_{ik} 表示目标语言序列中第 i 个单词所对应的源语言列表中的第 k 个单词的位置。对于列表中其他单词的扭曲度，则使用如下公式进行计算：

$$P(\pi_{[i]k} = j|\pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, t) = d_{>1}(j - \pi_{[i]k-1}|B(s_j)), \tag{8.27}$$

其中 $A(\cdot)$ 和 $B(\cdot)$ 分别表示从源语言、目标语言单词向单词词类映射的函数。这一扭曲度函数的改进背后的思想是，在生成 $t_{[i]}$ 的第一个源语言单词时，要考虑平均位置 $\odot_{[i]}$ 和这个源语言单词之间的绝对距离，随后生成的单词所放置的位置则要考虑前一个放置完的单词的相对位置以及当前源语言单词的词类。这个过程实际上使得同一个目标语言单词所生成的源语言单词之间可以相互影响，从而避免了独立生成各个源语言单词所带来的冲突问题。

8.2.6 IBM 模型 V

相对于前面叙述的 4 个模型，IBM 模型 V 针对词对齐的过程做了进一步的约束。它认为同一个源语言单词不应当由多个目标语言单词转换而来。如图8.8所示，前面 4 中词对齐方式都是合法的。然而，对于词对齐 a_5 和 a_6 来说，源语言单词“吃”和“早饭”分别对应着两个目标语言单词。为

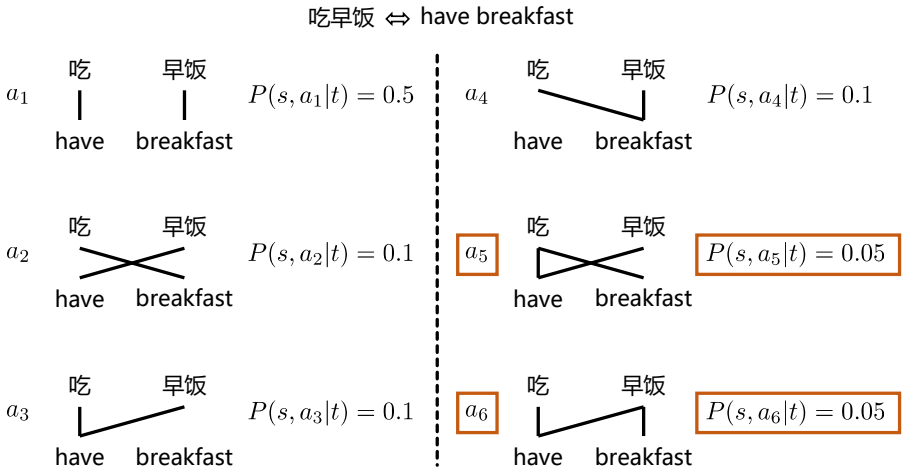


图 8.8 词对齐冲突示意图

了约束这种情况的出现，IBM 模型 V 在放置每一个源语言单词时都会检查这个位置是否已经放置了其他单词。为了实现这一点，引入一个新的变量 $v(j, \tau_1^{[i]-1}, \tau_{[i]1}^{k-1})$，它表示在放置 $\tau_{[i]k}$ 之前，源语言的前 j 个位置还有多少空余。为了简便起见，后续记这个变量为 v_j。这样，对于单词 $[i]$ 所对应的源语言单词列表中的第一个单词 $\tau_{[i]1}$ 有：

$$P(\tau_{[i]1} = j|\pi_1^{[i]-1}, \tau_0^l, \phi_0^l, t) = d_1(v_j|B(s_j), v_{\odot_{i-1}}, v_m - (\phi_{[i]} - 1)) \cdot (1 - \delta(v_j, v_{j-1})), \tag{8.28}$$

对于其他单词 $\tau_{[i]k}, 1 < k \leqslant \phi_{[i]}$，有：

$$\begin{aligned} &P(\pi_{[i]k} = j|\pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, t) \\ &= d_{>1}(v_j - v_{\pi_{[i]k-1}}|B(s_j), v_m - v_{\pi_{[i]k-1}} - \phi_{[i]} + k) \cdot (1 - \delta(v_j, v_{j-1})), \end{aligned} \tag{8.29}$$

此处的 $1 - \delta(v_j, v_{j-1})$ 是用来判断第 j 个位置是否为空。如果第 j 个位置为空，则 $v_j = v_{j-1}$，这样 $P(\pi_{[i]1} = j|\pi_1^{[i]-1}, \tau_0^l, \phi_0^l, t) = 0$。这样就避免了词对齐的冲突问题。

8.3 基于神经网络的机器翻译方法

传统机器翻译方法高度依赖于繁杂的特征工程，合理特征的设计对系统构建者的语言学背景具有较高的要求，同时需要在不断地试错过程中修正[10, 11]。这些特征往往不能够完整地反映输入文本的语义。举例来说，语言模型作为传统机器翻译模型的重要组成部分，为了降低模型复杂度而引入的马尔可夫假设使得上下文窗口之外的语义依赖无法被建模[12]；从输入文本表示的角度来说，经典的词袋模型（bag-of-words，BOW）则忽略了词序对输入文本表示的影响[13]。与之相对，神经网络模型作为一个强力的特征抽取器，能够自动地学习输入文本的最优表征从而在很大程度上降低对领域知识的要求及繁琐的特征工程预处理步骤[14]。此外，尽管传统机器翻译方法经过多年的发展已经能够实现不错的翻译性能，但存在一些固有缺陷影响其进一步提升。以最具代表性的基于短语的统计机器翻译方法为例，翻译通过将输入的源语言切分成短语并替换为目标语言的过程完成，短语范围之外的长程依赖在这一过程中被完全忽略进而造成翻译结果中的错误和不一致性。同时，为了提升翻译的准确性和流畅度，越来越多的功能模块不断被设计并添加到统计翻译模型当中（如语言模型、调序模型、长度调整模型等）[15–17]。复杂的翻译组件使得系统的整体调优和稳定性受到一定程度的影响。而以循环神经网络、Transformer 为代表的神经机器翻译方法能够有效地建模长程依赖，端到端的特性也使得系统的整体结构变得更加紧凑易于调整。

现代神经机器翻译模型大多依据序列到序列的方式对任务进行建模。在给定源语言输入文本，训练目标是找到最合适的目标语言句子作为译文。如图8.9所示，这一过程大体上可以划分为编码和解码两个模块步骤。编码器旨在将源语言句子转化为对应的语义向量，而解码器通过这些向量预测出合适的目标语言句子。

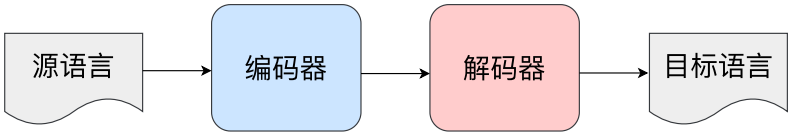


图 8.9　神经机器翻译的编码器-解码器框架

形式化地来说，给定源语言序列 $S = \{s_1, s_2, ..., s_n\}$，神经机器翻译模型试图找到具有最大条件概率 $P(T|S)$ 的目标语言序列 $T = \{t_1, t_2, ..., t_m\}$，$n$ 和 m 分别表示源语言和目标语言的长度。在生成目标语言句子的每个单词时，源语言和已经生成的目标语言信息会被使用。因此，神经机器翻译的整体过程可以按照如下公式描述：

$$\arg\max \prod_{i=1}^{m} P(t_i|t_{j<i}, S), \tag{8.30}$$

基于上述的总体目标，神经机器翻译利用不同的网络结构针对后验概率 $P(T|S)$ 进行建模，常见

的结构包括卷积神经网络、循环神经网络、自注意力神经网络等，本节将分别介绍上述常见神经机器翻译网络模型。

8.3.1 循环神经网络翻译模型

如前所述，神经机器翻译模型大多基于序列到序列的架构完成从源语言到目标语言的转换过程。不同神经机器翻译模型的主要区别在于编码器和解码器所采用的结构上的差异。自然语言文本可以看做一种时间序列数据，因此一种常见做法是采用基于循环神经网络的结构完成对源语言文本的编码以及目标语言文本的生成。基于循环神经网络的机器翻译模型整体结构如图8.10所示。其中，左侧为编码器部分，源语言单词按照其在文本序列中的先后顺序被依次送入到循环神经网络（RNN）当中。在每个时间步 t 中，模型依据送入的源语言单词 x_t 对应修改维护其模型内部的隐状态 $\boldsymbol{h}_t$，这个隐状态编码了输入的源语言序列前 t 个时刻的所有必要信息。按照这种方式当 m 个输入全部被送入到编码器之后，所对应的 $\boldsymbol{h}_m$ 可以认为包含了源语言序列的所有信息。

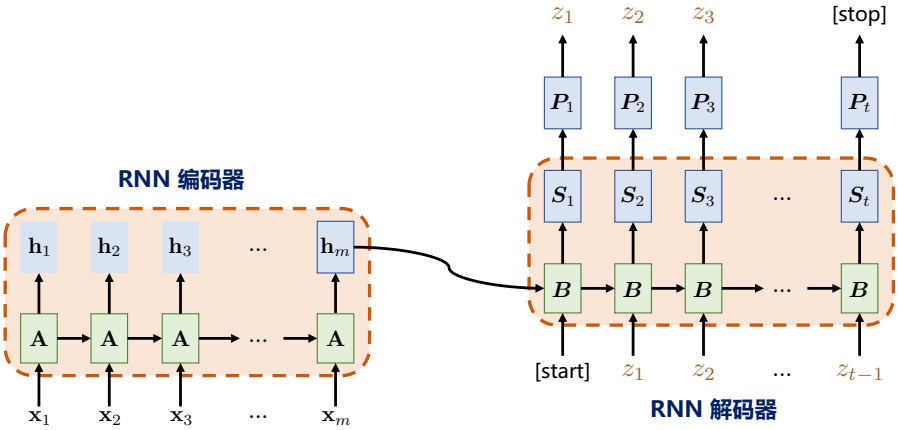


图 8.10 循环神经网络翻译模型

图8.10的右半部分是 RNN 解码器部分，它接收编码器输出的编码源语言句子信息的向量 $\boldsymbol{h}_m$ 作为初始隐状态 $\boldsymbol{s}_0$。由于 RNN 的循环过程在每个时间步都要求一个输入单词，为了启动解码过程，一般会使用一个保留的特殊符号“[Start]”作为翻译开始的标记送入到 RNN 解码器当中并解码出目标语言序列的第一个单词 z_1。接下来，z_1 会作为下一个时刻的输入被送入到循环神经网络当中并按照不断迭代产生后续的预测。由于目标语言序列的长度无法被提前预知，因此使用另一个保留符号“[Stop]”作为预测结束的标志。当某一个时刻 t 预测出的目标语言单词为 z_t =“[Stop]”时，解码过程动态地停止。在上述过程当中，主要涉及到两步运算，第一步是 RNN 接收前一时刻隐状态 $\boldsymbol{s}_{t-1}$ 并依据当前时刻输入 $\boldsymbol{z}_{t-1}$（目标语言单词 z_{t-1} 对应的语义嵌入）对隐状态进行维护并生

成 $\boldsymbol{s}_t$ 的运算过程，第二步是依据当前步骤隐状态生成目标语言单词的过程：

$$\boldsymbol{s}_t = \tanh(\boldsymbol{z}_{t-1}\boldsymbol{U} + \boldsymbol{s}_{t-1}\boldsymbol{W}) \tag{8.31}$$

$$\boldsymbol{p}_t = \text{Softmax}(\boldsymbol{s}_t\boldsymbol{V}), \tag{8.32}$$

其中 $\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}$ 是可学习的参数。$\boldsymbol{U}, \boldsymbol{W}$ 负责维护循环状态，而 $\boldsymbol{V}$ 负责将当前时刻状态转换到词表大小的概率分布 $\boldsymbol{p} \in \mathbb{R}^{vocab\_size}$，从中可以采样得到目标语言单词 z_t。

通过循环网络对源语言文本进行编码，并生成目标语言翻译结果的过程十分简单。然而，它仅仅使用一个定长的向量 $\boldsymbol{h}_m$ 编码整个源语言序列。这对于较短的源语言文本没有什么问题，但随着文本序列长度的逐渐加长，单一的一个向量 $\boldsymbol{h}_m$ 可能不足以承载源语言序列当中的所有信息。如图8.11所示，蓝色的线代表上述简单循环神经网络性能随源语言文本长度的变化趋势。当文本长度在 20 个单词以内时，单一向量能够承载源语言文本中的必要信息。随着文本序列的进一步增加，翻译性能的评价指标 BLEU 的值就开始出现明显地下降。因此，这就启发我们使用更加有效地机制从编码器向解码器传递源语言信息，这就是接下来要讲到的注意力机制。

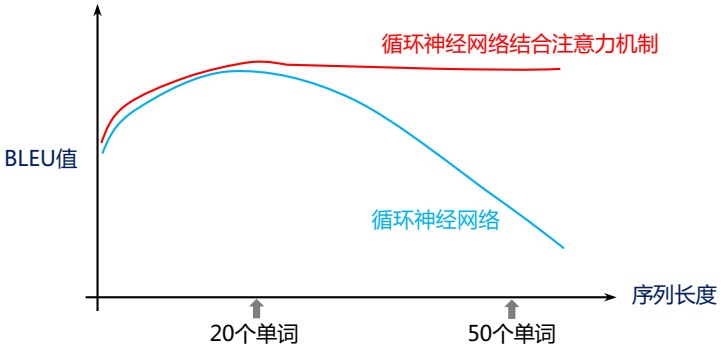


图 8.11　基于简单循环网络的机器翻译模型的瓶颈

引入注意力机制的循环机器翻译架构与基于简单循环网络的机器翻译模型大体结构相似，均采用循环神经网络作为编码器与解码器的实现。关键的不同点在于注意力机制的引入使得不再需要把原始文本中的所有必要信息压缩到一个向量当中。引入注意力机制的循环机器翻译架构如图8.12所示。

具体来说，给定源语言序列经过编码器输出的向量序列 $\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_m$，注意力机制旨在依据解码端翻译的需要，自适应地从这个向量序列中查找对应的信息。与简单循环网络相类似，在 $t = 4$ 时刻，旨在通过 $t - 1 = 3$ 时刻的隐状态 $\boldsymbol{s}_3$ 以及 $t = 4$ 时刻的输入 $\boldsymbol{x}'_4$ 维护循环隐状态并生成当前时刻目标语言翻译结果 $\boldsymbol{x}'_5$。为了更高效地考虑源语言上下文语义来提高翻译质量，注意力机制通过通过计算一组匹配分数 $\{\text{score}_i\}_{i=1}^m$ 并利用 softmax 归一化为一组权重 $\{\alpha_i\}_{i=1}^m$ 自适应地确定源

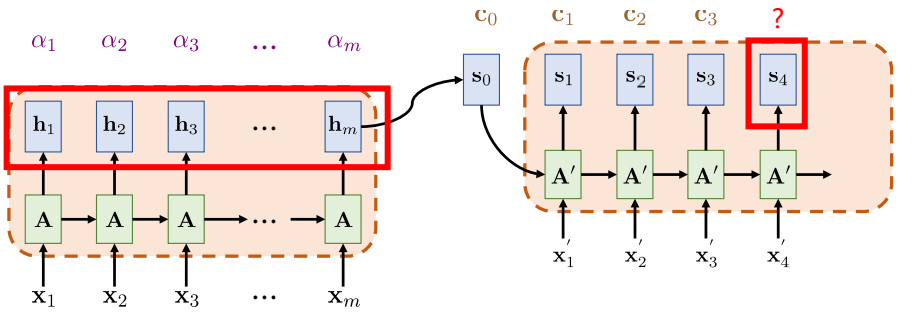


图 8.12 引入注意力机制的循环机器翻译架构

语言中需要聚焦的部分。具体计算公式如下：

$$\{\alpha_i\}_{i=1}^m = \text{Softmax}(\{\text{score}_i\}_{i=1}^m) \tag{8.33}$$

$$\text{score}_i = \begin{cases} \boldsymbol{s}_{t-1}\boldsymbol{h}_i^T & \text{向量乘} \\ \cos(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i^T) & \text{向量夹角} \\ \boldsymbol{s}_{t-1}\boldsymbol{W}\boldsymbol{h}_i^T & \text{线性模型} \\ \tanh(\boldsymbol{W}[\boldsymbol{s}_{t-1}, \boldsymbol{h}_i])\boldsymbol{v}^T & \text{单层网络,} \end{cases} \tag{8.34}$$

其中 $\boldsymbol{W}, \boldsymbol{v}$ 表示可训练的参数矩阵或向量，$[\cdot,\cdot]$ 表示拼接算符。基于上述权重能够得到生成译文 $\boldsymbol{x}'_t$ 所必要的源语言信息 $\boldsymbol{c}_t$，进一步地，可以将这部分源语言信息与当前时刻的输入 $\boldsymbol{x}'_t$ 拼接送入 RNN 作为新的输入：

$$\boldsymbol{c}_t = \sum_i \alpha_i \boldsymbol{h}_i \tag{8.35}$$

$$\boldsymbol{s}_t = \tanh([\boldsymbol{x}'_t, \boldsymbol{c}_t]\boldsymbol{U} + \boldsymbol{s}_{t-1}\boldsymbol{W}), \tag{8.36}$$

通过这样的修改，在维护 RNN 任意时刻隐藏状态并生成译文的过程中，能够自适应地考虑源语言中的哪部分信息需要被聚焦，从而生成更加高质量的译文。

8.3.2 卷积神经网络翻译模型

卷积神经网络也是一种经典的神经网络结构，被广泛地使用在自然语言处理的各项任务当中。相较于循环神经网络来说，卷积神经网络每一步卷积操作并不依赖于前一时间步的计算结果因而能够充分并行化以更好地利用 GPU 的计算资源。在本章当中，将以 ConvS2S，一种全卷积、高并行、序列到序列的模型为例，介绍卷积神经网络是如何被应用在机器翻译任务当中的。

ConvS2S 的整体结构如图8.13所示，它采用卷积神经网络作为编码器（图片上侧）和解码器（图片左下侧）的具体实现，完成对源语言和目标语言的特征提取，这种模型结构使得每一层的网

络计算可以完全并行化，不再收到循环结构中时序依赖的限制。同时，通过堆叠多层卷积结构，上下文窗口的范围得以不断扩大，从而逐渐建模输入文本中的远距离依赖。同时相较于循环神经网络，它的信息传递路径更短，更有利于优化。

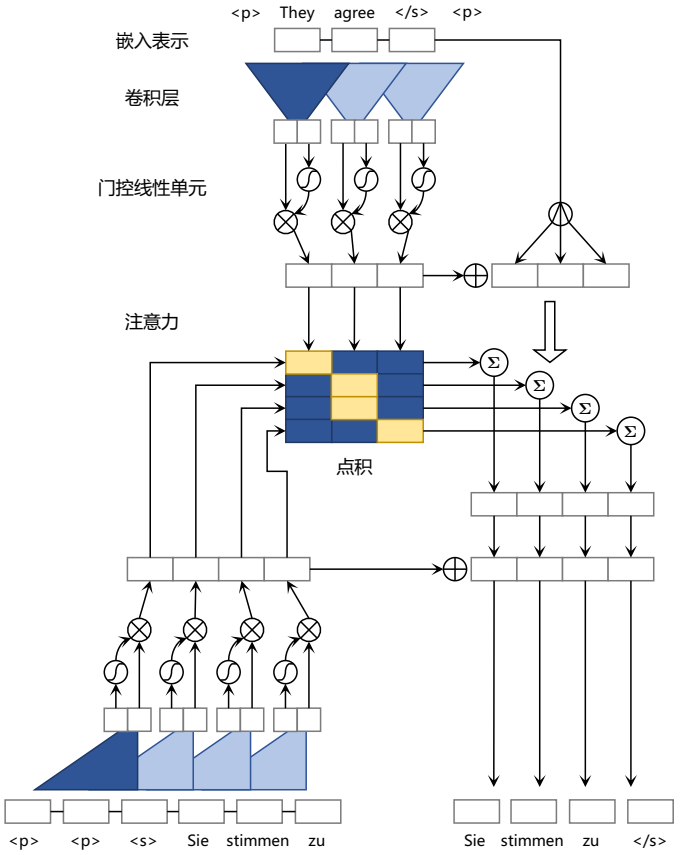


图 8.13 卷积序列到序列模型结构

ConvS2S 作为一种经典的神经机器翻译模型，实现了优越的翻译质量及高效的翻译效率，受到了学术界的广泛关注。它主要由下述几个部件构成：

- **位置编码**：由于 ConvS2S 摒弃了循环结构，因此需要在输入层引入位置编码来标识输入序列中词与词之间的相对位置关系。
- **卷积层与门控线性单元**：这部分是编码器与解码器的实现模块，分别用于抽取源语言和目标语言的上下文语义特征。
- **残差连接**：这部分也被添加在编码器和解码器中堆叠的多层卷积结构当中，直接连接每一层的输入与输出，从而提高信息传播效率，减小模型的优化难度。

- **多步注意力机制**：这里与上一小节中的循环神经机器翻译模型类似，都采用注意力机制自适应地从源语言端检索译文对应的源语言信息。不同的是，此处的注意力计算在解码器的每一层当中都会出现，因而被称为“多步”注意力。

接下来，将详细介绍 ConvS2S 模型当中涉及到的关键模块的技术细节：

位置编码：由于 ConvS2S 不再使用基于循环的结构编码输入序列，因此模型失去了对于输入文本中词与词之间相对位置关系的感知。因此位置编码旨在重新给予模型这部分信息。具体来说，给定输入源语言序列 $S = \{s_1, s_2, ..., s_n\}$ 及其在嵌入空间中的表示序列 $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_n\}$，其中 $\boldsymbol{s}_j \in \mathbb{R}^f$ 是词嵌入矩阵 $D^{V\times f}$ 中单词 s_j 对应的表示。为了使得卷积模型能够感知到输入序列中单词的相对位置关系，一个额外的位置嵌入 $\boldsymbol{P} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n\}$ 被用于标识每个单词在句子中的绝对位置，其中 $\boldsymbol{p}_j \in \mathbb{R}^f$。词嵌入和位置嵌入同时被使用作为编码器和解码器的输入 $\boldsymbol{E} = \{\boldsymbol{s}_1 + \boldsymbol{p}_1, \boldsymbol{s}_2 + \boldsymbol{p}_2, ..., \boldsymbol{s}_n + \boldsymbol{p}_n\}$。

门控卷积结构：在 ConvSeq2Seq 模型中，编码器和解码器均采用门控卷积结构作为建模源语言和目标语言的基本部件，这一部件由序列维度上的一维卷积运算和非线性门控机制结合而成。卷积过程能够有效地建模待处理文本中的局部上下文信息，而序列中的长程依赖问题则可以通过多层卷积结构的堆叠得到缓解。非线性门控机制使得我们能够建模输入视野下更加复杂的依赖关系。具体来说，对于嵌入层输入的文本表示 $\boldsymbol{E} \in \mathcal{R}^{n\times f}$，通过一个线性映射将维度转换到 d 维之后，我们能够得到卷积操作的每个上下文窗口的输入 $\boldsymbol{X} \in \mathcal{R}^{k\times d}$。对其进行卷积运算如下：

$$\boldsymbol{A} = \boldsymbol{X} \times \boldsymbol{W}_A + \boldsymbol{b}_A \tag{8.37}$$

$$\boldsymbol{B} = \boldsymbol{X} \times \boldsymbol{W}_B + \boldsymbol{b}_B, \tag{8.38}$$

这里存在两组卷积操作，每组由 d 个卷积核组成，其参数包括 $\boldsymbol{W}_A \in \mathcal{R}^{k\times d\times d}$，$\boldsymbol{W}_B \in \mathcal{R}^{k\times d\times d}$，$\boldsymbol{b}_A \in \mathcal{R}^d$，$\boldsymbol{b}_B \in \mathcal{R}^d$。对 $\boldsymbol{X}$ 进行卷积操作得到对应两组输出 $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{R}^d$ 后，基于门控线性单元（Gated Linear Units）的非线性变换被用作激活函数得到最终输出：

$$\boldsymbol{h} = \boldsymbol{A} \otimes \sigma(\boldsymbol{B}) \tag{8.39}$$

其中 $\otimes$ 表示逐点乘法运算，非线性门控 $\sigma(\boldsymbol{B})$ 主要被用于建模 $\boldsymbol{A}$ 中的哪部分上下文信息是相关的，哪些需要被遗忘。此外，为了克服卷积神经网络无法建模长程依赖的问题，多层卷积的堆叠是必要的。举例来说，堆叠 6 层上下文窗口大小 $k = 5$ 的卷积单元就能够将窗口大小扩大到 25，也即输出能够依赖 25 个单元的输入。为了有效地训练深层神经网络结构，残差链接（residual connections）被引入到模型构建当中。具体来说，每一层卷积单元的输入被直接连接到输出当中如下所示：

$$\boldsymbol{h}^l = \boldsymbol{A}^l \otimes \sigma(\boldsymbol{B}^l) + \boldsymbol{h}^{l-1}. \tag{8.40}$$

多步自注意力机制：在解码译文的过程中，对源语言的参考是必不可少的。ConvS2S 结构中，

解码器同样采用了堆叠的多层门控卷积结构完成对目标语言的解码，并在每一层门控卷积之后通过注意力机制参考源语言信息。以解码器第 l 层第 i 个时间步的注意力计算为例，为了确定需要参考源语言中的哪部分信息，当前时刻的解码器状态 $\boldsymbol{z}_i^l$ 以及前一个时刻解码出的目标语言嵌入 $\boldsymbol{g}_i$ 被用于作出决策的依据：

$$\boldsymbol{d}_i^l = \boldsymbol{z}_i^l \boldsymbol{W}_d^l + \boldsymbol{b}_d^l + \boldsymbol{g}_i, \tag{8.41}$$

$$\boldsymbol{z}_i^l = \text{Conv}(\boldsymbol{s}_i^l), \tag{8.42}$$

其中 $\boldsymbol{W}_d^l$ 和 $\boldsymbol{b}_d^l$ 是可训练的参数。基于当前位置的状态依据 $\boldsymbol{d}_i^l$，目标语言位置 i 相对源语言第 j 个单词的注意力得分 a_{ij}^l 可以通过 $\boldsymbol{d}_i^l$ 和源语言编码器对应位置的输出 $\boldsymbol{y}_j$ 计算得到：

$$a_{ij}^l = \frac{exp(\boldsymbol{d}_i^l \cdot \boldsymbol{y}_j)}{\sum_{t=1}^{n} exp(\boldsymbol{d}_i^l \cdot \boldsymbol{h}_t)} \tag{8.43}$$

基于上述过程得到的注意力得分，可以对源语言不同位置的信息进行加权整合得到为了预测当前位置目标语言单词所需的依据：

$$\boldsymbol{c}_i^l = \sum_{j=1}^{m} a_{ij}^l (\boldsymbol{h}_j + \boldsymbol{e}_j), \tag{8.44}$$

这里的源语言端同时利用了编码器的输出 $\boldsymbol{h}_j$ 以及对应位置的输入词嵌入 $\boldsymbol{e}_j$。这两者对应着更加全面的源语言信息，在实践中被证明十分有效。基于上述源语言信息，可以得到解码器端第 l 层的输出为：

$$\boldsymbol{s}_i^{l+1} = \boldsymbol{c}_i^l + \boldsymbol{z}_i^l \tag{8.45}$$

上述多步注意力机制中的“多步”一词主要从两个方面体现。首先从多层卷积堆叠的角度来说，前一层中通过注意力机制动态地决定哪些相关信息需要被关注并传递到下一层当中，而下一层在计算对源语言不同位置的注意力得分过程中又会考虑到这些信息。从时间步的角度来说，在计算目标语言每个位置 i 的注意力分布时，前 k 个位置的注意力历史信息 $\boldsymbol{c}_{i-k}^{l-1}, ..., \boldsymbol{c}_i^{l-1}$ 都会作为输入的一部分被考虑。这就使得我们在计算当前位置需要参考源语言中的哪些信息时能够有效地判断哪些信息在之前的时间步中已经被参考过了。与之相对，在循环神经网络中，这一过程需要历经多个时间步中的非线性转换才能够被利用，这极有可能造成相关信息的丢失。

8.3.3 自注意力神经网络翻译模型

基于循环或卷积神经网络的序列到序列建模方法是现存机器翻译任务中的经典方法。然而，它们在建模文本长程依赖方面都存在一定的局限性。对于卷积神经网络来说，受限的上下文窗口在建模长文本方面天然地存在不足。而对于循环神经网络来说，上下文的语义依赖是通过维护循环单元中的隐藏状态实现的。在编码过程中，每一个时间步的输入建模都涉及到对隐藏状态的修改。

随着序列长度的增加，编码在隐藏状态中的序列早期的上下文信息被逐渐遗忘。尽管注意力机制的引入在一定程度上缓解了这个问题，但循环网络在编码效率方面仍存在很大的不足之处。由于编码端和解码端的每一个时间步的隐藏状态都依赖于前一时间步的计算结果，这就造成了在训练和推断阶段的低效。

Transformer[18] 是由谷歌在 2017 年提出的一种 Seq2Seq 架构。它的出现使得机器翻译的性能和效率迈向了一个新的阶段。它摒弃了循环结构，并完全通过注意力机制完成对源语言序列和目标语言序列全局依赖的建模。在抽取每个单词的上下文特征时，Transformer 通过自注意力机制（self-attention）衡量上下文中每一个单词对当前单词的重要程度，在这个过程当中没有任何的循环单元参与计算。这种高度可并行化的编码过程使得模型的运行变得十分高效。

基于 Transformer 的机器翻译模型架构如图8.14所示，左侧和右侧分别对应着 Seq2Seq 模型的编码器和解码器结构。它们均由若干个基本的 Transformer 层组成（对应着图中的灰色框）。每个 Transformer 层都接收一个向量序列 $\{\boldsymbol{x}_i\}_{i=1}^t$ 作为输入，并输出一个等长的向量序列作为输出 $\{\boldsymbol{y}_i\}_{i=1}^t$。这里的 $\boldsymbol{x}_i$ 和 $\boldsymbol{y}_i$ 分别对应着文本序列中的一个单词的表示。而 $\boldsymbol{y}_i$ 是当前 Transformer 对输入 $\boldsymbol{x}_i$ 进一步整合其上下文语义后对应的输出。在从输入 $\{\boldsymbol{x}_i\}_{i=1}^t$ 到输出 $\{\boldsymbol{y}_i\}_{i=1}^t$ 的语义抽象过程中，主要涉及到如下几个模块：

- **自注意力子层**：对应图中的 Multi-Head Attention 部分。使用自注意力机制整合上下文语义，它使得序列中任意两个单词之间的依赖关系可以直接被建模而不基于传统的循环结构，从而更好地解决文本的长程依赖。
- **前馈子层**：对应图中的 Feed Forward 部分。通过全连接层对输入文本序列中的每个单词表示进行更复杂的变换。
- **残差连接**：对应图中的 Add 部分。它是一条分别作用在上述两个子层当中直连通路，被用于连接它们的输入与输出。从而使得信息流动更加高效，有利于模型的优化。
- **层标准化**：对应图中的 Norm 部分。作用于上述两个子层的输出表示序列中，对表示序列进行层标准化操作，同样起到稳定优化的作用。

相比于编码器端，解码器端要更复杂一些。具体来说，解码器的每个 Transformer 层的第一个自注意力子层额外增加了注意力掩码，对应图中的掩码多头注意力（Masked Multi-Head Attention）部分。这主要是因为在翻译的过程中，编码器端主要用于编码源语言序列的信息，而这个序列是完全已知的，因而编码器仅需要考虑如何融合上下文语义信息即可。而解码端则负责生成目标语言序列，这一生成过程是自回归的，即对于每一个单词的生成过程，仅有当前单词之前的目标语言序列是可以被观测的，因此这一额外增加的掩码是用来掩盖后续的文本信息，以防模型在训练阶段直接看到后续的文本序列进而无法得到有效地训练。此外，解码器端还额外增加了一个多头注意力（Multi-Head Attention）模块，需要注意的是它同时接收来自编码器端的输出以及当前 Transformer 层第一个掩码注意力层的输出。它的作用是在翻译的过程当中，为了生成合理的目标语言序列需要观测待翻译的源语言序列是什么。基于上述的编码器和解码器结构，一个待翻译的源语言文本

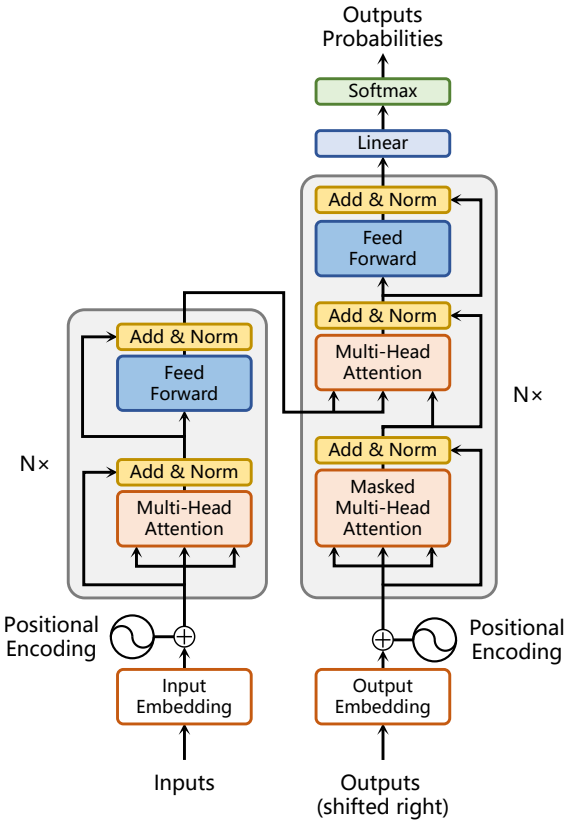


图 8.14　基于 Transformer 的机器翻译模型架构[18]

首先经过编码器端的每个 Transformer 层对其上下文语义的层层抽象，最终输出每一个源语言单词上下文相关的表示。解码器端以自回归的方式生成目标语言文本，即每个时间步 t 参考编码器端输出的所有源语言文本表示以及前 $t-1$ 个时刻生成的目标语言文本生成当前时刻的目标语言单词。接下来详细介绍翻译过程当中所涉及到的不同模块的技术细节。

位置编码：对于待翻译的文本序列，首先通过输入嵌入层（Input Embedding）将每个单词转换为其相对应的向量表示。在送入编码器端建模其上下文语义之前，一个非常重要的操作是在词嵌入中加入位置编码这一特征。由于 Transfomer 不再使用基于循环的方式建模文本输入，序列中不再有任何信息能够提示模型单词之间的相对位置关系。因此补充这部分信息是十分必要的。具体来说，序列中每一个单词所在的位置都对应一个实值向量。这一实值向量会与单词表示对应相加并送入到后续模块中做进一步处理。在训练的过程当中，模型会自动地学习到如何利用这部分

位置信息。为了得到不同位置对应的编码，Transformer 使用不同频率的正余弦函数如下所示：

$$\mathrm{PE}(\mathrm{pos}, 2i) = \sin(\frac{\mathrm{pos}}{10000^{2i/d}}) \tag{8.46}$$

$$\mathrm{PE}(\mathrm{pos}, 2i+1) = \cos(\frac{\mathrm{pos}}{10000^{2i/d}}), \tag{8.47}$$

其中，pos 表示单词所在的位置，$2i$ 和 $2i+1$ 表示位置编码向量中的对应维度，d 则对应位置编码的总维度。通过上面这种方式计算位置编码有这样几个好处：首先，正余弦函数的范围是在 [-1,+1]，导出的位置编码与原词嵌入相加不会使得结果偏离过远而破坏原有单词的语义信息。其次，依据三角函数的基本性质，可以得到第 $\mathrm{pos}+k$ 个位置的编码是第 pos 个位置的编码的线性组合，这就意味着位置编码中蕴含着单词之间的距离信息。

自注意力子层：自注意力（Self-Attention）机制是基于 Transformer 的机器翻译模型的基本操作，在源语言的编码和目标语言的生成中频繁地被使用以建模源语言、目标语言任意两个单词之间的依赖关系。给定由单词语义嵌入及其位置编码叠加得到的输入表示 $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^t$，为了实现对上下文语义依赖的建模，我们进一步引入在自注意力机制中设计到的三个元素：查询 $\boldsymbol{q}_i$（Query），键 $\boldsymbol{k}_i$（Key），值 $\boldsymbol{v}_i$（Value）。在编码输入序列中每一个单词的表示的过程中，这三个元素被用于计算上下文单词所对应的权重得分。直观地说，这些权重反映了在编码当前单词的表示时对于上下文不同部分所需要的关注程度。具体来说，如图8.15所示，通过三个线性变换 $\boldsymbol{W}^Q \in \mathbb{R}^{d\times d_k}$，$\boldsymbol{W}^K \in \mathbb{R}^{d\times d_k}$，$\boldsymbol{W}^V \in \mathbb{R}^{d\times d_v}$ 将将输入序列中的每一个单词表示 $\boldsymbol{x}_i$ 转换为其对应的 $\boldsymbol{q}_i \in \mathbb{R}^{d_k}$，$\boldsymbol{k}_i \in \mathbb{R}^{d_k}$，$\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ 向量。

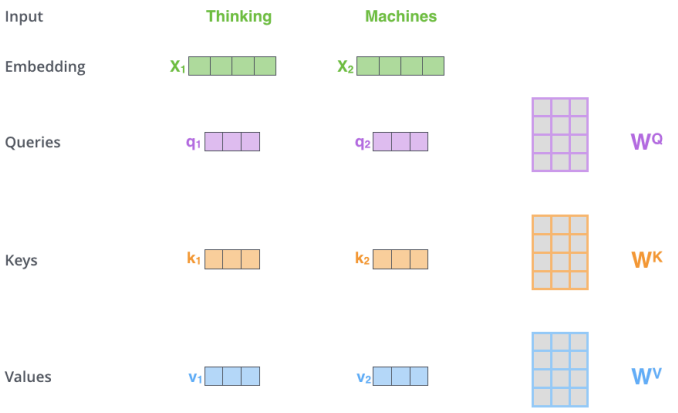


图 8.15　自注意力机制中的查询、键、值向量

为了得到编码单词 x_i 时所需要关注的上下文信息，通过位置 i 查询向量与其他位置的键向量做点积得到匹配分数 $\boldsymbol{q}_1 \cdot \boldsymbol{k}_1, \boldsymbol{q}_2 \cdot \boldsymbol{k}_2, ..., \boldsymbol{q}_t \cdot \boldsymbol{k}_t$。为了防止过大的匹配分数在后续 softmax 计算过程

中导致的梯度爆炸以及收敛效率的问题，这些得分会除放缩因子 $\sqrt{d}$ 以稳定优化。放缩后的得分经过 softmax 归一化为概率之后与其他位置的值向量相乘来聚合我们希望关注的上下文信息并最小化不相关信息的干扰。上述计算过程可以被形式化地表述如下：

$$\boldsymbol{Z} = \text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V}, \tag{8.48}$$

其中 $\boldsymbol{Q} \in \mathbb{R}^{L\times d_k}, \boldsymbol{K} \in \mathbb{R}^{L\times d_k}, \boldsymbol{V} \in \mathbb{R}^{d\times d_v}$ 分别表示输入序列中的不同单词的 $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ 向量拼接组成的矩阵，L 表示序列长度，$\boldsymbol{Z} \in \mathbb{R}^{L\times d_v}$ 表示自注意力操作的输出。为了进一步增强自注意力机制聚合上下文信息的能力，一种被称为多头（Multi-head）自注意力的机制被提出以从关注上下文的不同侧面。具体来说，上下文中每一个单词的表示 x_i 经过多组线性映射 $\{\boldsymbol{W}_j^Q, \boldsymbol{W}_j^K, \boldsymbol{W}_j^V\}_{j=1}^N$ 到不同的表示子空间当中，公式8.48会在不同的子空间中分别计算并得到不同的上下文相关的单词序列表示 $\{\boldsymbol{Z}_j\}_{j=1}^N$。最终，一个线性变换 $\boldsymbol{W}^O \in \mathbb{R}^{(Nd_v)\times d}$ 被用于综合不同子空间中的上下文表示并形成自注意力层最终的输出 $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^t$。

前馈子层：前馈子层接受自注意力子层的输出作为输入，并通过一个带有 `Relu` 激活函数的两层全连接网络对输入进行更加复杂的非线性变换。实验证明，这一非线性变换会对模型最终的性能产生十分重要的影响。

$$FFN(\boldsymbol{x}) = \texttt{Relu}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{8.49}$$

$$\texttt{Relu}(\boldsymbol{x}) = \max(0, \boldsymbol{x}), \tag{8.50}$$

其中 $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2$ 表示前馈子层的参数。实验结果表明，增大前馈子层隐状态的维度有利于提升最终翻译结果的质量，因此，前馈子层隐状态的维度一般比自注意力子层要大。

残差连接与层标准化：事实上 Transformer 是一个非常庞大的网络结构。它的编码器和解码器均由 6 层基本的 Transformer 层组成，每一层当中都包含复杂的非线性映射，这就导致模型的训练比较困难。因此，研究者们在 Transformer 层中进一步引入了残差连接与层标准化技术以进一步提升训练的稳定性。具体来说，残差连接主要是指使用一条直连通道直接将对应子层的输入连接到输出上去，从而避免由于网络过深在优化过程中潜在的梯度消失问题：

$$\boldsymbol{x}^{l+1} = f(\boldsymbol{x}^l) + \boldsymbol{x}^l, \tag{8.51}$$

其中 $\boldsymbol{x}^l$ 表示第 l 层的输入，$f(\cdot)$ 表示一个映射函数。此外，为了进一步使得每一层的输入输出范围稳定在一个合理的范围内，层标准化技术被进一步引入 Transformer 的每一层当中：

$$LN(\boldsymbol{x}) = g \cdot \frac{\boldsymbol{x} - \mu}{\sigma} + b \tag{8.52}$$

其中 μ 和 σ 分别表示均值和方差，用于将数据平移缩放到均值为 0，方差为 1 的标准分布，g 和 b

是可学习的参数。层标准化技术可以有效地缓解优化过程中潜在的不稳定、收敛速度慢等问题。

8.4 机器翻译语料库

数据集是评测机器学习算法的基础，本小节针对机器翻译领域广泛使用的基准数据集进行一个简单的介绍，读者可以基于这些数据集复现现存主流的机器翻译模型并进行比较。

- **WMT 数据集**：该数据集是一个以英语为主的多语言机器翻译数据集，涉及英中、英德翻译等多种任务，数据来源于新闻、医学、翻译等领域。
- **IWSLT 数据集**：该数据集是一个来自于 TED 演讲的文本翻译数据集，语料规模较小，涉及英德、英中翻译等任务。
- **NIST 数据集**：该数据集是一个新闻翻译领域的高质量数据集，评测集包括 4 句参考译文，涉及中英、英捷翻译等任务。
- **TVsub**：该数据集是一个抽取自电视剧字幕的翻译数据集，适合于对话中的长距离上下文依赖研究，涉及中英翻译任务。
- **Flickr30k**：该数据集是多模态机器翻译的主流数据集之一，包含 31783 张图片，每张图片对应 5 个语句标注，涉及英德翻译任务。
- **Multi30k**：该数据集是多模态机器翻译的主流数据集之一，包含 31014 张图片，每张图片对应 5 个语句标注，设计英德、英法翻译任务。
- **IAPRTC-12**：该数据集是多么台翻译的主流数据集之一，包含 20000 张图片及其对应标注，涉及英德翻译任务。
- **IKEA**：该数据集是多模态翻译领域的主流数据集之一，包含 3600 张图片及其对应标注，涉及英德、英法翻译任务。

此外，机器翻译模型的训练需要大规模双语数据，这里针对一些主流公开的双语平行预料进行了简单汇总。

- **News Commentary Corpus**：该语料库爬取自 Project Syndicate 网站的政治经济评论，涉及中文、英语等 12 个语种，64 个语言对的双语数据。
- **CWMT Corpus**：该语料库是由中国计算机翻译研讨会社区收集和共享的中英平行预料数据，来源于新闻、电影、小说、政府文档等多个领域。
- **Common Crawl Corpus**：该语料库是爬取自互联网网页的数据，涵盖捷克语、德语、法语、俄语等 4 种语言到英语的平行语料。
- **Europarl Corpus**：该语料库的数据来源是欧洲议会记录，涵盖了保加利亚语、捷克语等 20 种欧洲语言到英语的平行语料。
- **ParaCrawl Corpus**：该语料库的数据来源依然是互联网爬取，包含了 23 种欧洲语言到英语的双语语料。
- **United Nations Parallel Corpus**：该语料库的数据来源是联合国公共领域的官方记录和其他

会议文件，涵盖了阿拉伯语、英语、西班牙语、法语、俄语、汉语等 6 种联合国正式语言。

- **TED Corpus**：该语料库的数据来源是 TED 大会演讲在其网站公开的自 2007 年以来的演讲字幕，以及超过 100 种语言的翻译版本。
- **OpenSubtitle**：该语料库是由 P.Lison 和 J.Tiedemann 收集自 opensubtitles 电影字幕网站，是一个涵盖了 62 中语言、1782 个语言对的平行预料的大规模数据集。
- **Wikititles Corpus**：该语料库的数据来源是维基百科的标题，涵盖了古吉拉特语等 14 个语种，11 个语言对的平行预料数据。
- **CzEng**：该语料库的数据来源是欧洲法律、信息技术和小说领域，涵盖了捷克语和英语的平行语料。
- **Yandex Corpus**：该语料库的数据均爬取自互联网网页，涵盖了俄语和英语的平行语料。
- **Tilde MODEL Corpus**：该语料库由多个来自于经济、新闻、政府、旅游等门户网站的数据集组成，涵盖了欧洲多种语言的开放数据。
- **Setimes Corpus**：该语料库的数据来源是东南欧时报的新闻报道，涵盖了克罗地亚语、阿尔巴尼亚语等 9 种巴尔干语言，72 个语言对的双语数据。
- **TVsub**：该语料库的数据来源是电视剧字幕的中英文对话的语料，主要用于对话和长距离上下文信息依赖的研究。
- **Recipe Corpus**：该语料库是一个包含 10 万多个句对的由 Cookpad 公司创建的日英食谱语料库。

8.5 延伸阅读

从基于规则、统计的模型到如今基于神经网络的模型，机器翻译任务已经经历了数十年的发展并取得了辉煌的成绩。伴随着硬件计算能力的提升和大规模训练数据的积累，我们在很多富资源语言的翻译问题上已经能够实现很好的翻译效果。尽管如此，机器翻译领域中仍存在着很多问题尚未解决，包括神经网络结构优化、低资源翻译、多模态翻译等方面。

如何对现有的神经网络结构进行优化使其更加适配机器翻译任务一直是学术界研究的重点。在这一方面，有如下几个问题值得进一步关注。首先是针对 Transformer 模型的多头注意力模块，许多研究人员发现其中一部分注意力头具有有意义的语言学解释并在模型编码过程中发挥着重要的作用，而存在另一部分注意力头则并没有发挥什么作用。因此如何通过对 Transformer 模型进一步剪枝在不影响性能的前提下提升效率具有很大的使用价值 [19]。此外，通过引入正则化技术 [20]，多尺度表示 [21]，定义隐变量等方式提升源语言和目标语言表示 [22]。最后，现存的 Transformer 模型对于超长文本序列建模能力存在一定的不足之处，如何解决长文本建模的能力也是一个值得研究的方向 [23–25]。

尽管现存的神经机器翻译模型在大规模数据的基础之上实现了令人惊叹的翻译质量，但世界上的大多数语言无法获取到如此大规模的平行语料数据。因此如何更高效地利用已有的单、双语

数据始终是学术界的一个研究热点。现有的方法从数据增强的角度探索如何对已有数据进行增强 [26**?** –29]；从预训练模型的角度探索如何将预训练知识更加高效的迁移到下游任务 [30, 31]；以及通过多任务学习的方式探索语言间的资源共享 [32, 33]。最近，零资源翻译也逐渐受到了广泛的关注。这一任务旨在使用少量的平行语料库（覆盖 k 个语言），就能够实现在任何 k(k-1) 个语言对之间进行翻译 [34]。

针对跨模态的数据进行翻译也是一个极具潜力的研究方向。典型的任务包括语音翻译、图像翻译等。其中，语音翻译的一个重要应用是机器同声传译，其最大的难点在于不同的语言表达语序不同，现存的解决方案包括等待源语言 k 个单词后再进行翻译 [35, 36]，或者通过束搜索的方式预测未来的词序列从而提升准确度等 [37, 38]。此外，数据稀缺也是多模态机器翻译任务的一大难点。一些研究者常识通过调整训练策略使得模型更容易捕获上下文信息 [39] 或数据增强策略 [40] 来提升整体数据量等。

8.6 习题

参考文献

[1] Forcada M L, Ginestí-Rosell M, Nordfalk J, et al. Apertium: A free/open-source platform for rule-based machine translation[J]. Machine Translation, 2011, 25(2):127-144.

[2] Koehn P, Och F J, Marcu D. Statistical phrase-based translation[C/OL]//NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1. USA: Association for Computational Linguistics, 2003: 48–54. https://doi.org/10.3115/1073445.1073462.

[3] Koehn P, Hoang H, Birch A, et al. Moses: Open source toolkit for statistical machine translation [C/OL]//Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions. Prague, Czech Republic: Association for Computational Linguistics, 2007: 177-180. https://aclanthology.org/P07-2045.

[4] Chrisman L. Learning recursive distributed representations for holistic computation[J]. Connection Science, 1991, 3(4):345-366.

[5] Allen R B. Several studies on natural language and back-propagation[C]//Proceedings of the IEEE First International Conference on Neural Networks: volume 2. IEEE Piscataway, NJ, 1987: 341.

[6] Wu Y, Schuster M, Chen Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[Z]. 2016.

[7] Akhbardeh F, Arkhangorodsky A, Biesialska M, et al. Findings of the 2021 conference on machine translation (wmt21)[C]//Proceedings of the Sixth Conference on Machine Translation. 2021: 1-88.

[8] Khashabi D, Stanovsky G, Bragg J, et al. Genie: A leaderboard for human-in-the-loop evaluation of text generation[J]. arXiv preprint arXiv:2101.06561, 2021.

[9] 肖桐, 朱靖波. 机器翻译：基础与模型[M]. 北京: 电子工业出版社, 2021.

[10] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(ARTICLE):2493-2537.

[11] Young T, Hazarika D, Poria S, et al. Recent trends in deep learning based natural language processing [J]. ieee Computational intelligenCe magazine, 2018, 13(3):55-75.

[12] Song F, Croft W B. A general language model for information retrieval[C]//Proceedings of the eighth international conference on Information and knowledge management. 1999: 316-321.

[13] Wallach H M. Topic modeling: Beyond bag-of-words[C/OL]//ICML '06: Proceedings of the 23rd International Conference on Machine Learning. New York, NY, USA: Association for Computing Machinery, 2006: 977–984. https://doi.org/10.1145/1143844.1143967.

[14] Britz D, Goldie A, Luong M T, et al. Massive exploration of neural machine translation architectures [J]. arXiv preprint arXiv:1703.03906, 2017.

[15] Galley M, Manning C D. A simple and effective hierarchical phrase reordering model[C/OL]// Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. Honolulu, Hawaii: Association for Computational Linguistics, 2008: 848-856. https://aclanthology.org/D08-1089.

[16] Chiang D, Knight K, Wang W. 11,001 new features for statistical machine translation[C]//Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics. 2009: 218-226.

[17] Green S, Wang S I, Cer D, et al. Fast and adaptive online training of feature-rich translation models[C]//Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2013: 311-321.

[18] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C/OL]//Guyon I, Luxburg U V, Bengio S, et al. Advances in Neural Information Processing Systems: volume 30. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[19] Voita E, Talbot D, Moiseev F, et al. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned[C/OL]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 5797-5808. https://aclanthology.org/P19-1580. DOI: 10.18653/v1/P19-1580.

[20] Li J, Tu Z, Yang B, et al. Multi-head attention with disagreement regularization[C/OL]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium:

Association for Computational Linguistics, 2018: 2897-2903. https://aclanthology.org/D18-1317. DOI: 10.18653/v1/D18-1317.

[21] Hao J, Wang X, Shi S, et al. Multi-granularity self-attention for neural machine translation[C/OL]// Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 887-897. https://aclanthology.org/D19-1082. DOI: 10.18653/v1/D19-1082.

[22] Setiawan H, Sperber M, Nallasamy U, et al. Variational neural machine translation with normalizing flows[C/OL]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020: 7771-7777. https://aclanthology.org/2020.acl-main.694. DOI: 10.18653/v1/2020.acl-main.694.

[23] Beltagy I, Peters M E, Cohan A. Longformer: The long-document transformer[J/OL]. CoRR, 2020, abs/2004.05150. https://arxiv.org/abs/2004.05150.

[24] Choromanski K M, Likhosherstov V, Dohan D, et al. Rethinking attention with performers[C/OL]// International Conference on Learning Representations. 2021. https://openreview.net/forum?id=Ua6zuk0WRH.

[25] Guo Q, Qiu X, Liu P, et al. Star-transformer[C/OL]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 1315-1325. https://aclanthology.org/N19-1133. DOI: 10.18653/v1/N19-1133.

[26] Wu L, Wang Y, Xia Y, et al. Exploiting monolingual data at scale for neural machine translation [C/OL]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 4207-4216. https://aclanthology.org/D19-1430. DOI: 10.18653/v1/D19-1430.

[27] Zhang J, Zong C. Exploiting source-side monolingual data in neural machine translation[C/OL]// Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, 2016: 1535-1545. https://aclanthology.org/D16-1160. DOI: 10.18653/v1/D16-1160.

[28] Dou Z Y, Anastasopoulos A, Neubig G. Dynamic data selection and weighting for iterative back-translation[C/OL]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language

Processing (EMNLP). Online: Association for Computational Linguistics, 2020: 5894-5904. https://www.aclweb.org/anthology/2020.emnlp-main.475. DOI: 10.18653/v1/2020.emnlp-main.475.

[29] Wang S, Liu Y, Wang C, et al. Improving back-translation with uncertainty-based confidence estimation[C/OL]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 791-802. https://aclanthology.org/D19-1073. DOI: 10.18653/v1/D19-1073.

[30] Peters M E, Ruder S, Smith N A. To tune or not to tune? adapting pretrained representations to diverse tasks[C/OL]//Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019). Florence, Italy: Association for Computational Linguistics, 2019: 7-14. https://aclanthology.org/W19-4302. DOI: 10.18653/v1/W19-4302.

[31] Sun C, Qiu X, Xu Y, et al. How to fine-tune BERT for text classification?[J/OL]. CoRR, 2019, abs/1905.05583. http://arxiv.org/abs/1905.05583.

[32] Dong D, Wu H, He W, et al. Multi-task learning for multiple language translation[C/OL]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, 2015: 1723-1732. https://aclanthology.org/P15-1166. DOI: 10.3115/v1/P15-1166.

[33] Firat O, Cho K, Bengio Y. Multi-way, multilingual neural machine translation with a shared attention mechanism[C/OL]//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California: Association for Computational Linguistics, 2016: 866-875. https://aclanthology.org/N16-1101. DOI: 10.18653/v1/N16-1101.

[34] Al-Shedivat M, Parikh A. Consistency by agreement in zero-shot neural machine translation [C/OL]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 1184-1197. https://aclanthology.org/N19-1121. DOI: 10.18653/v1/N19-1121.

[35] Gu J, Neubig G, Cho K, et al. Learning to translate in real-time with neural machine translation [C/OL]//Proceedings of the 15th Conference of the European Chapter of the Association for Com-

putational Linguistics: Volume 1, Long Papers. Valencia, Spain: Association for Computational Linguistics, 2017: 1053-1062. https://aclanthology.org/E17-1099.

[36] Grissom II A, He H, Boyd-Graber J, et al. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation[C/OL]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1342-1352. https://aclanthology.org/D14-1140. DOI: 10.3115/v1/D14-1140.

[37] Ma M, Huang L, Xiong H, et al. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework[C/OL]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 3025-3036. https://aclanthology.org/P19-1289. DOI: 10.18653/v1/P19-1289.

[38] Zheng R, Ma M, Zheng B, et al. Speculative beam search for simultaneous translation[C/OL]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 1395-1402. https://aclanthology.org/D19-1144. DOI: 10.18653/v1/D19-1144.

[39] Jean S, Cho K. Context-aware learning for neural machine translation[J/OL]. CoRR, 2019, abs/1903.04715. http://arxiv.org/abs/1903.04715.

[40] Sugiyama A, Yoshinaga N. Data augmentation using back-translation for context-aware neural machine translation[C/OL]//Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019). Hong Kong, China: Association for Computational Linguistics, 2019: 35-44. https://aclanthology.org/D19-6504. DOI: 10.18653/v1/D19-6504.

索引