

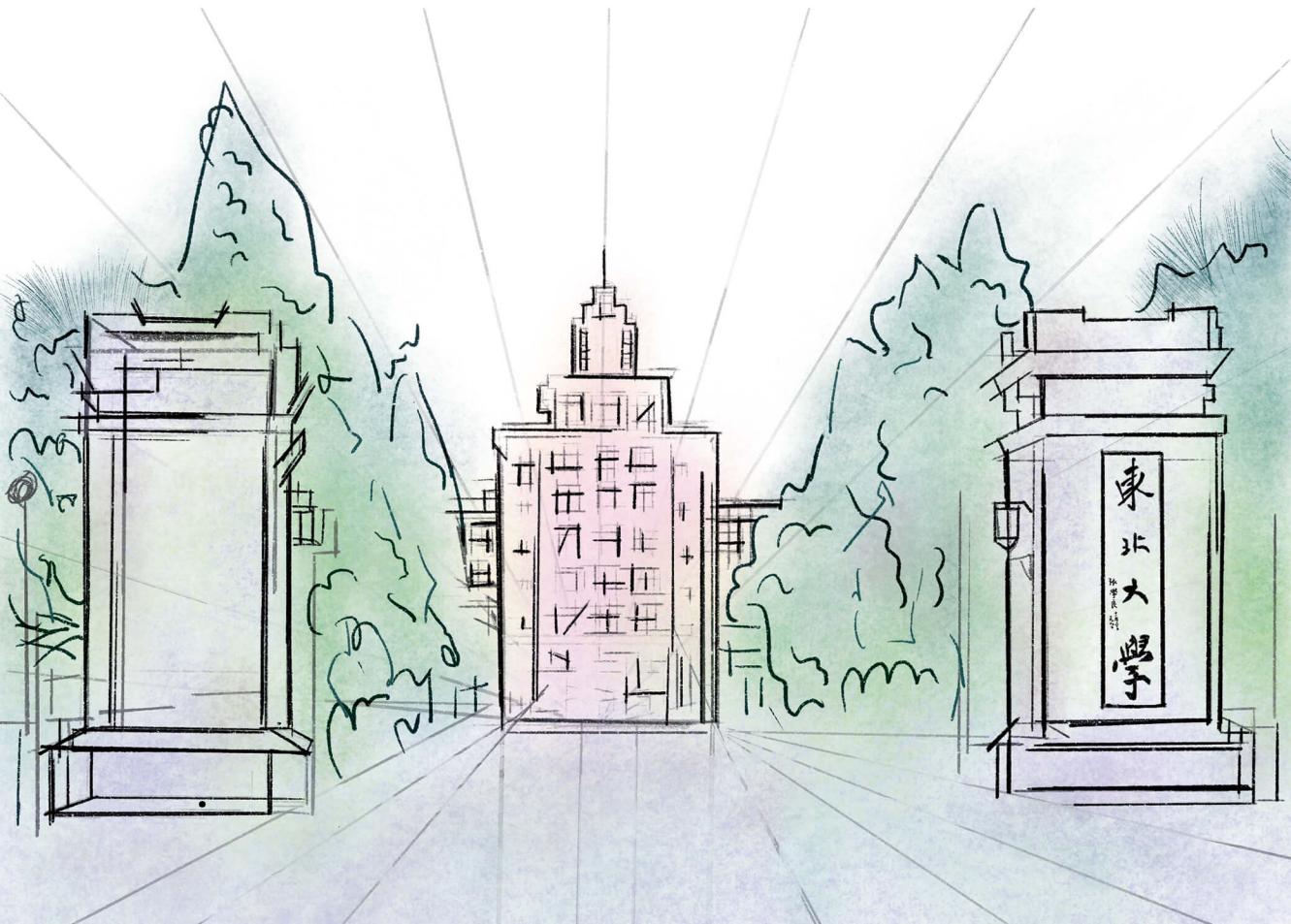
# 机器翻译

## 基础与模型

Machine Translation: Foundations and Models

肖桐 朱靖波 著

东北大学自然语言处理实验室 · 小牛翻译



Copyright © 2020 肖桐 朱靖波

东北大学自然语言处理实验室 · 小牛翻译

顾问：姚天顺 王宝库

[HTTPS://OPENSOURCE.NIUTRANS.COM/MTBOOK/HOME PAGE.HTML](https://opensource.niutrans.com/mtbook/homepage.html)

[HTTPS://GITHUB.COM/NIUTRANS/MTBOOK](https://github.com/NIUTRANS/MTBOOK)

Licensed under the Creative Commons Attribution-NonCommercial 4.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/4.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

*February 1, 2021*

## 在此感谢为本书做出贡献的人

曹润柘、曾信、孟霞、单韦乔、周涛、周书含、许诺、李北、许晨、林野、李垠桥、王子扬、刘辉、张裕浩、冯凯、罗应峰、魏冰浩、王屹超、李炎洋、胡驰、姜雨帆、田丰宁、刘继强、张哲旸、陈贺轩、牛蕊、杜权、张春良、王会珍、张俐、马安香、胡明涵

# 前　　言

## 1. 本书的由来

让计算机进行自然语言的翻译是人类长期的梦想，也是人工智能的终极目标之一。自上世纪九十年代起，机器翻译迈入了基于统计建模的时代，发展到今天，已经大量应用了深度学习等机器学习方法，并且取得了令人瞩目的进步。在这个时代背景下，对机器翻译的模型、方法和实现技术进行深入了解是自然语言处理领域研究者和实践者所渴望的。

与所有从事机器翻译研究的人一样，笔者也梦想着有朝一日机器翻译能够完全实现。这个想法可以追溯到 1980 年，姚天顺教授和王宝库教授创立了东北大学自然语言处理实验室，把机器翻译作为毕生为之奋斗的目标。这也影响了包括笔者在内的许多人。虽然，那时的机器翻译技术并不先进，研究条件也异常艰苦，但是实现机器翻译的梦想从未改变。

步入二十一世纪后，统计学习方法的兴起给机器翻译带来了全新的思路，同时也带来了巨大的技术进步。笔者有幸经历了那个时代，同时也加入到机器翻译研究的浪潮中。笔者从 2007 年开始研发 NiuTrans 开源系统，在 2012 年对 NiuTrans 机器翻译系统进行产业化，并创立了小牛翻译。在此过程中，笔者目睹了机器翻译的成长，并不断地被机器翻译取得的进步所感动。那时，笔者就曾经思考过将机器翻译的模型和方法进行总结，形成资料供人阅读。虽然粗略写过一些文字，但是未成体系，只是在相关的教学环节中进行使用，供实验室同学闲暇时参考阅读。

但是机器翻译领域进展之快是无法预见的。2016 年之后，随着深度学习方法在机器翻译中的进一步应用，机器翻译迎来了前所未有的大好机遇。新的技术方法层出不穷，机器翻译系统也得到了广泛应用。这时，笔者心里又涌现出将机器翻译的技术内容编撰成书的想法。这种强烈的念头使得笔者完成了本书的第一个版本（包含七章），并开源供人广泛阅读。承蒙同行们厚爱，得到了很多反馈，包括一些批评意见。这些使得笔者可以更加全面地梳理思路。

最初，笔者的想法仅仅是将机器翻译的技术内容做成资料供人阅读。但是，朋友、同事们一直鼓励将内容正式出版。虽然担心书的内容不够精致，无法给同行作为参考，但是最终还是下定决心重构内容。所幸，得到电子工业出版社的支持，形成新版，共十八章。

写作中，每当笔者翻起以前的资料，都会想起当年的一些故事。与其说这部书是写给读者，还不如说这本书是写给笔者自己，写给所有同笔者一样，经历过或正在经历机器翻译蓬勃发展年代的人。希望本书可以作为一个时代的记录，但是这个时代并未结束，还将继续，并更加美好。

## 2. 本书的特色

本书全面回顾了近三十年内机器翻译的技术发展历程，并围绕**机器翻译的建模和深度学习方法**这两个主题对机器翻译的技术方法进行了全面介绍。在写作中，笔者力求用朴实的语言和简洁的实例阐述机器翻译的基本模型，同时对相关的技术前沿进行讨论。其中也会涉及大量的实践经验，包括许多机器翻译系统开发的细节。从这个角度来说，本书不单单是一本理论书籍，它还结合了机器翻译的应用，给读者提供了很多机器翻译技术落地的具体思路。

本书可以供计算机相关专业高年级本科生及研究生学习之用，也可以作为自然语言处理领域，特别是机器翻译方向相关研究人员的参考资料。此外，本书各章主题明确，内容紧凑。因此，读者也可将每章作为某一专题的学习资料。

**用最简单的方式阐述机器翻译的基本思想**是笔者所期望达到的目标。但是，书中不可避免会使用一些形式化定义和算法的抽象描述，因此，笔者尽所能通过图例进行解释（本书共 396 张插图）。不过，本书所包含的内容较为广泛，难免会有疏漏，望读者海涵，并指出不当之处。

## 3. 本书的内容

本书共分为四个部分，十八章。章节的顺序参考了机器翻译技术发展的时间脉络，同时兼顾了机器翻译知识体系的内在逻辑。本书的主要内容包括：

- 第一部分：机器翻译基础
  - 第一章 机器翻译简介
  - 第二章 统计语言建模基础
  - 第三章 词法分析和语法分析基础
  - 第四章 翻译质量评价
- 第二部分：统计机器翻译
  - 第五章 基于词的机器翻译建模
  - 第六章 基于扭曲度和繁衍率的模型
  - 第七章 基于短语的模型
  - 第八章 基于句法的模型
- 第三部分：神经机器翻译
  - 第九章 人工神经网络和神经语言建模
  - 第十章 基于循环神经网络的模型
  - 第十一章 基于卷积神经网络的模型
  - 第十二章 基于自注意力的模型
- 第四部分：机器翻译前沿
  - 第十三章 神经机器翻译模型训练
  - 第十四章 神经机器翻译模型推断

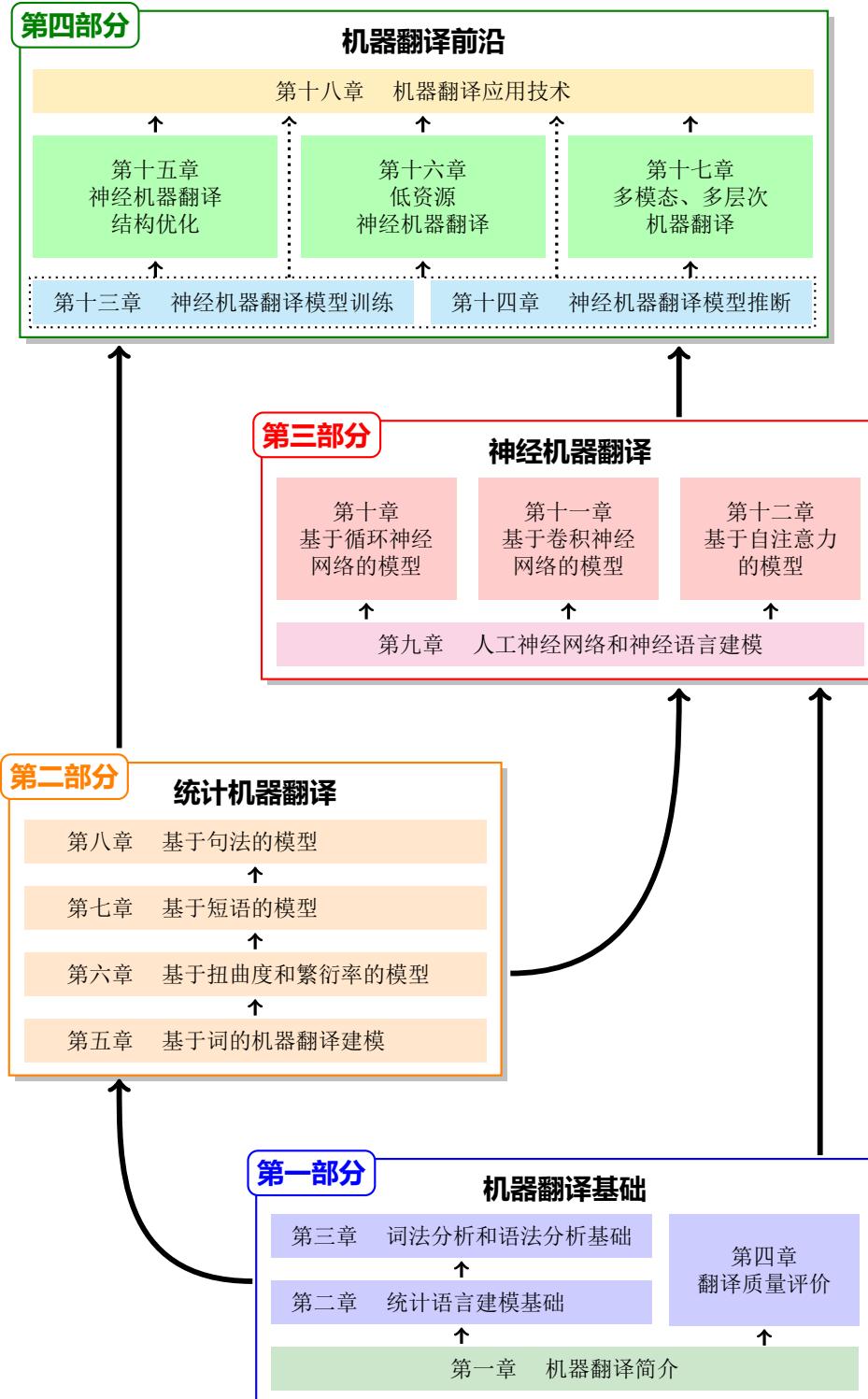
- 第十五章 神经机器翻译结构优化
- 第十六章 低资源机器翻译
- 第十七章 多模态、多层次机器翻译
- 第十八章 机器翻译应用技术

其中，第一部分是本书的基础知识部分，包含统计建模、语言分析、机器翻译评价等。在第一章对机器翻译的历史及现状进行介绍之后，第二章通过语言建模任务将统计建模的思想阐述出来，同时这部分内容也会作为后续机器翻译模型及方法的基础。第三章重点介绍机器翻译所涉及的词法和句法分析方法，旨在为后续相关概念的使用进行铺垫，同时进一步展示统计建模思想在相关问题上的应用。第四章相对独立，系统地介绍了机器翻译结果的评价方法，这部分内容也是机器翻译建模及系统设计所需的前置知识。

本书的第二部分主要介绍统计机器翻译的基本模型。第五章是整个机器翻译建模的基础。第六章进一步对扭曲度和产出率两个概念进行介绍，同时给出相关的翻译模型，这些模型在后续章节的内容中都有涉及。第七章和第八章分别介绍了基于短语和句法的模型。它们都是统计机器翻译的经典模型，其思想也构成了机器翻译成长过程中最精华的部分。

本书的第三部分主要介绍神经机器翻译模型，该模型也是近些年机器翻译的热点。第九章介绍了神经网络和深度学习的基础知识以保证本书知识体系的完备性。同时，第九章也介绍了基于神经网络的语言模型，其建模思想在神经机器翻译中被大量使用。第十、十一、十二章分别对三种经典的神经机器翻译模型进行介绍，以模型提出的时间为序，从最初的基于循环网络的模型，到最新的 Transformer 模型均有涉及。其中也会对编码器-解码器框架、注意力机制等经典方法和技术进行介绍。

本书的第四部分会进一步对机器翻译的前沿技术进行讨论，以神经机器翻译为主。第十三、十四、十五章是神经机器翻译研发的三个主要方面，也是近几年机器翻译领域讨论最多的几个方向。第十六章也是机器翻译的热门方向之一，包括无监督翻译等主题都会在这里被讨论。第十七章会对语音、图像翻译等多模态方法以及篇章级翻译等方法进行介绍，它们可以被看作是机器翻译在更多任务上的扩展。第十八章会结合笔者在各种机器翻译比赛和机器翻译产品研发的经验，对机器翻译的应用技术进行讨论。







# Contents

I

## 机器翻译基础

<b>1</b>	<b>机器翻译简介</b>	<b>23</b>
1.1	机器翻译的概念	23
1.2	机器翻译简史	25
1.2.1	人工翻译	25
1.2.2	机器翻译的萌芽	26
1.2.3	机器翻译的受挫	27
1.2.4	机器翻译的快速成长	28
1.2.5	机器翻译的爆发	29
<b>1.3</b>	<b>机器翻译现状及挑战</b>	<b>30</b>
<b>1.4</b>	<b>基于规则的方法</b>	<b>33</b>
1.4.1	规则的定义	33
1.4.2	转换法	34
1.4.3	基于中间语言的方法	36
1.4.4	规则方法的优缺点	37
<b>1.5</b>	<b>数据驱动的方法</b>	<b>37</b>
1.5.1	基于实例的机器翻译	37

1.5.2	统计机器翻译 . . . . .	38
1.5.3	神经机器翻译 . . . . .	39
1.5.4	对比分析 . . . . .	40
<b>1.6</b>	<b>推荐学习资源 . . . . .</b>	<b>41</b>
1.6.1	经典书籍 . . . . .	41
1.6.2	相关学术会议 . . . . .	42
<b>2</b>	<b>统计语言建模基础 . . . . .</b>	<b>45</b>
<b>2.1</b>	<b>概率论基础 . . . . .</b>	<b>45</b>
2.1.1	随机变量和概率 . . . . .	45
2.1.2	联合概率、条件概率和边缘概率 . . . . .	47
2.1.3	链式法则 . . . . .	48
2.1.4	贝叶斯法则 . . . . .	49
2.1.5	KL 距离和熵 . . . . .	50
<b>2.2</b>	<b>掷骰子游戏 . . . . .</b>	<b>52</b>
<b>2.3</b>	<b><i>n</i>-gram 语言模型 . . . . .</b>	<b>56</b>
2.3.1	建模 . . . . .	56
2.3.2	参数估计和平滑算法 . . . . .	58
2.3.3	语言模型的评价 . . . . .	64
<b>2.4</b>	<b>预测与搜索 . . . . .</b>	<b>64</b>
2.4.1	搜索问题的建模 . . . . .	65
2.4.2	经典搜索 . . . . .	69
2.4.3	局部搜索 . . . . .	72
<b>2.5</b>	<b>小结及拓展阅读 . . . . .</b>	<b>74</b>
<b>3</b>	<b>词法分析和语法分析基础 . . . . .</b>	<b>77</b>
<b>3.1</b>	<b>问题概述 . . . . .</b>	<b>77</b>
<b>3.2</b>	<b>中文分词 . . . . .</b>	<b>79</b>
3.2.1	基于词典的分词方法 . . . . .	80
3.2.2	基于统计的分词方法 . . . . .	82
<b>3.3</b>	<b>命名实体识别 . . . . .</b>	<b>84</b>
3.3.1	序列标注任务 . . . . .	84
3.3.2	基于特征的统计学习 . . . . .	85
3.3.3	基于概率图模型的方法 . . . . .	86
3.3.4	基于分类器的方法 . . . . .	92

<b>3.4 句法分析</b>	95
3.4.1 句法树	95
3.4.2 上下文无关文法	96
3.4.3 规则和推导的概率	100
<b>3.5 小结及拓展阅读</b>	103
<b>4 翻译质量评价</b>	105
<b>  4.1 译文质量评价所面临的挑战</b>	105
<b>  4.2 人工评价</b>	107
4.2.1 评价策略	108
4.2.2 打分标准	109
<b>  4.3 有参考答案的自动评价</b>	111
4.3.1 基于词串比对的方法	111
4.3.2 基于词对齐的方法	114
4.3.3 基于检测点的方法	117
4.3.4 多策略融合的评价方法	118
4.3.5 译文多样性	119
4.3.6 相关性与显著性	123
<b>  4.4 无参考答案的自动评价</b>	126
4.4.1 质量评估任务	126
4.4.2 构建质量评估模型	131
4.4.3 质量评估的应用场景	132
<b>  4.5 小结及拓展阅读</b>	133

## II

# 统计机器翻译

<b>5 基于词的机器翻译建模</b>	139
<b>  5.1 词在翻译中的作用</b>	139
<b>  5.2 一个简单实例</b>	141
5.2.1 翻译的流程	141
5.2.2 统计机器翻译的基本框架	144
5.2.3 单词翻译概率	144
5.2.4 句子级翻译模型	147
5.2.5 解码	150

<b>5.3</b>	<b>噪声信道模型 . . . . .</b>	<b>153</b>
<b>5.4</b>	<b>统计机器翻译的三个基本问题 . . . . .</b>	<b>156</b>
5.4.1	词对齐 . . . . .	156
5.4.2	基于词对齐的翻译模型 . . . . .	157
5.4.3	基于词对齐的翻译实例 . . . . .	159
<b>5.5</b>	<b>IBM 模型 1 . . . . .</b>	<b>159</b>
5.5.1	IBM 模型 1 的建模 . . . . .	160
5.5.2	解码及计算优化 . . . . .	161
5.5.3	训练 . . . . .	162
<b>5.6</b>	<b>小结及拓展阅读 . . . . .</b>	<b>169</b>
<b>6</b>	<b>基于扭曲度和繁衍率的模型 . . . . .</b>	<b>171</b>
<b>6.1</b>	<b>基于扭曲度的模型 . . . . .</b>	<b>171</b>
6.1.1	什么是扭曲度 . . . . .	172
6.1.2	IBM 模型 2 . . . . .	173
6.1.3	隐马尔可夫模型 . . . . .	174
<b>6.2</b>	<b>基于繁衍率的模型 . . . . .</b>	<b>175</b>
6.2.1	什么是繁衍率 . . . . .	176
6.2.2	IBM 模型 3 . . . . .	178
6.2.3	IBM 模型 4 . . . . .	180
6.2.4	IBM 模型 5 . . . . .	181
<b>6.3</b>	<b>解码和训练 . . . . .</b>	<b>183</b>
<b>6.4</b>	<b>问题分析 . . . . .</b>	<b>183</b>
6.4.1	词对齐及对称化 . . . . .	184
6.4.2	“缺陷” 问题 . . . . .	184
6.4.3	句子长度 . . . . .	185
6.4.4	其他问题 . . . . .	186
<b>6.5</b>	<b>小结及拓展阅读 . . . . .</b>	<b>186</b>
<b>7</b>	<b>基于短语的模型 . . . . .</b>	<b>189</b>
<b>7.1</b>	<b>翻译中的短语信息 . . . . .</b>	<b>189</b>
7.1.1	词的翻译带来的问题 . . . . .	190
7.1.2	更大粒度的翻译单元 . . . . .	191
7.1.3	机器翻译中的短语 . . . . .	193

<b>7.2</b>	<b>数学建模</b>	<b>195</b>
7.2.1	基于翻译推导的建模	196
7.2.2	对数线性模型	197
7.2.3	判别式模型中的特征	198
7.2.4	搭建模型的基本流程	198
<b>7.3</b>	<b>短语抽取</b>	<b>199</b>
7.3.1	与词对齐一致的短语	200
7.3.2	获取词对齐	201
7.3.3	度量双语短语质量	202
<b>7.4</b>	<b>翻译调序建模</b>	<b>203</b>
7.4.1	基于距离的调序	204
7.4.2	基于方向的调序	204
7.4.3	基于分类的调序	206
<b>7.5</b>	<b>翻译特征</b>	<b>206</b>
<b>7.6</b>	<b>最小错误率训练</b>	<b>207</b>
<b>7.7</b>	<b>栈解码</b>	<b>211</b>
7.7.1	翻译候选匹配	212
7.7.2	翻译假设扩展	212
7.7.3	剪枝	213
7.7.4	解码中的栈结构	214
<b>7.8</b>	<b>小结及拓展阅读</b>	<b>216</b>
<b>8</b>	<b>基于句法的模型</b>	<b>219</b>
<b>8.1</b>	<b>翻译中句法信息的使用</b>	<b>219</b>
<b>8.2</b>	<b>基于层次短语的模型</b>	<b>221</b>
8.2.1	同步上下文无关文法	224
8.2.2	层次短语规则抽取	228
8.2.3	翻译特征	229
8.2.4	CKY 解码	230
8.2.5	立方剪枝	234
<b>8.3</b>	<b>基于语言学句法的模型</b>	<b>236</b>
8.3.1	基于句法的翻译模型分类	238
8.3.2	基于树结构的文法	240
8.3.3	树到串翻译规则抽取	245
8.3.4	树到树翻译规则抽取	253
8.3.5	句法翻译模型的特征	256
8.3.6	基于超图的推导空间表示	257

8.3.7	基于树的解码 vs 基于串的解码 . . . . .	260
8.4	<b>小结及拓展阅读 . . . . .</b>	<b>264</b>

### III

## 神经机器翻译

<b>9</b>	<b>人工神经网络和神经语言建模 . . . . .</b>	<b>269</b>
9.1	<b>深度学习与人工神经网络 . . . . .</b>	<b>269</b>
9.1.1	发展简史 . . . . .	270
9.1.2	为什么需要深度学习 . . . . .	272
9.2	<b>神经网络基础 . . . . .</b>	<b>275</b>
9.2.1	线性代数基础 . . . . .	275
9.2.2	人工神经元和感知机 . . . . .	280
9.2.3	多层神经网络 . . . . .	284
9.2.4	函数拟合能力 . . . . .	287
9.3	<b>神经网络的张量实现 . . . . .</b>	<b>291</b>
9.3.1	张量及其计算 . . . . .	291
9.3.2	张量的物理存储形式 . . . . .	294
9.3.3	张量的实现手段 . . . . .	295
9.3.4	前向传播与计算图 . . . . .	296
9.4	<b>神经网络的参数训练 . . . . .</b>	<b>297</b>
9.4.1	损失函数 . . . . .	298
9.4.2	基于梯度的参数优化 . . . . .	299
9.4.3	参数更新的并行化策略 . . . . .	308
9.4.4	梯度消失、梯度爆炸和稳定性训练 . . . . .	309
9.4.5	过拟合 . . . . .	311
9.4.6	反向传播 . . . . .	312
9.5	<b>神经语言模型 . . . . .</b>	<b>317</b>
9.5.1	基于前馈神经网络的语言模型 . . . . .	317
9.5.2	对于长序列的建模 . . . . .	321
9.5.3	单词表示模型 . . . . .	322
9.5.4	句子表示模型 . . . . .	325
9.6	<b>小结及拓展阅读 . . . . .</b>	<b>326</b>
<b>10</b>	<b>基于循环神经网络的模型 . . . . .</b>	<b>329</b>
10.1	<b>神经机器翻译的发展简史 . . . . .</b>	<b>329</b>
10.1.1	神经机器翻译的起源 . . . . .	331

10.1.2 神经机器翻译的品质 . . . . .	333
10.1.3 神经机器翻译的优势 . . . . .	336
<b>10.2 编码器-解码器框架 . . . . .</b>	<b>337</b>
10.2.1 框架结构 . . . . .	338
10.2.2 表示学习 . . . . .	339
10.2.3 简单的运行实例 . . . . .	340
10.2.4 机器翻译范式的对比 . . . . .	341
<b>10.3 基于循环神经网络的翻译建模 . . . . .</b>	<b>342</b>
10.3.1 建模 . . . . .	343
10.3.2 长短时记忆网络 . . . . .	346
10.3.3 门控循环单元 . . . . .	348
10.3.4 双向模型 . . . . .	349
10.3.5 多层神经网络 . . . . .	350
<b>10.4 注意力机制 . . . . .</b>	<b>350</b>
10.4.1 翻译中的注意力机制 . . . . .	351
10.4.2 上下文向量的计算 . . . . .	353
10.4.3 注意力机制的解读 . . . . .	356
10.4.4 实例 - GNMT . . . . .	357
<b>10.5 训练及推断 . . . . .</b>	<b>359</b>
10.5.1 训练 . . . . .	359
10.5.2 推断 . . . . .	364
<b>10.6 小结及拓展阅读 . . . . .</b>	<b>368</b>
<b>11 基于卷积神经网络的模型 . . . . .</b>	<b>371</b>
<b>11.1 卷积神经网络 . . . . .</b>	<b>371</b>
11.1.1 卷积核与卷积操作 . . . . .	372
11.1.2 步长与填充 . . . . .	374
11.1.3 池化 . . . . .	375
11.1.4 面向序列的卷积操作 . . . . .	376
<b>11.2 基于卷积神经网络的翻译建模 . . . . .</b>	<b>378</b>
11.2.1 位置编码 . . . . .	379
11.2.2 门控卷积神经网络 . . . . .	380
11.2.3 残差网络 . . . . .	382
11.2.4 多跳注意力机制 . . . . .	382
11.2.5 训练与推断 . . . . .	384

<b>11.3 局部模型的改进</b>	385
11.3.1 深度可分离卷积	386
11.3.2 轻量卷积和动态卷积	388
<b>11.4 小结及拓展阅读</b>	389
<b>12 基于自注意力的模型</b>	391
<b>12.1 自注意力机制</b>	391
<b>12.2 Transformer 架构</b>	393
12.2.1 Transformer 的优势	393
12.2.2 总体结构	394
<b>12.3 位置编码</b>	397
<b>12.4 基于点乘的多头注意力机制</b>	399
12.4.1 点乘注意力机制	399
12.4.2 多头注意力机制	401
12.4.3 掩码操作	402
<b>12.5 残差网络和层标准化</b>	403
<b>12.6 前馈全连接网络子层</b>	404
<b>12.7 训练</b>	405
<b>12.8 推断</b>	408
<b>12.9 小结及拓展阅读</b>	408

## IV

## 机器翻译前沿

<b>13 神经机器翻译模型训练</b>	413
<b>13.1 开放词表</b>	413
13.1.1 大词表和未登录词问题	414
13.1.2 子词	414
13.1.3 双字节编码	415
13.1.4 其他方法	416
<b>13.2 正则化</b>	418
13.2.1 L1/L2 正则化	420
13.2.2 标签平滑	421
13.2.3 Dropout	422

<b>13.3 对抗样本训练</b>	424
13.3.1 对抗样本及对抗攻击	424
13.3.2 基于黑盒攻击的方法	425
13.3.3 基于白盒攻击的方法	426
<b>13.4 学习策略</b>	428
13.4.1 极大似然估计的问题	428
13.4.2 非 Teacher-forcing 方法	428
13.4.3 强化学习方法	431
<b>13.5 知识蒸馏</b>	435
13.5.1 什么是知识蒸馏	435
13.5.2 知识蒸馏的基本方法	436
13.5.3 机器翻译中的知识蒸馏	438
<b>13.6 基于样本价值的学习</b>	439
13.6.1 数据选择	439
13.6.2 课程学习	442
13.6.3 持续学习	444
<b>13.7 小结及拓展阅读</b>	445
<b>14 神经机器翻译模型推断</b>	447
<b>14.1 面临的挑战</b>	447
<b>14.2 基本问题</b>	449
14.2.1 推断方向	449
14.2.2 译文长度控制	451
14.2.3 搜索终止条件	452
14.2.4 译文多样性	453
14.2.5 搜索错误	454
<b>14.3 轻量模型</b>	455
14.3.1 输出层的词汇选择	455
14.3.2 消除冗余计算	456
14.3.3 轻量解码器及小模型	458
14.3.4 批量推断	458
14.3.5 低精度运算	460
<b>14.4 非自回归翻译</b>	461
14.4.1 自回归 vs 非自回归	461
14.4.2 非自回归翻译模型的结构	462
14.4.3 更好的训练目标	465
14.4.4 引入自回归模块	466

14.4.5	基于迭代精化的非自回归翻译模型 . . . . .	467
<b>14.5</b>	<b>多模型集成 . . . . .</b>	<b>469</b>
14.5.1	假设选择 . . . . .	469
14.5.2	局部预测融合 . . . . .	470
14.5.3	译文重组 . . . . .	471
<b>14.6</b>	<b>小结与拓展阅读 . . . . .</b>	<b>473</b>
<b>15</b>	<b>神经机器翻译结构优化 . . . . .</b>	<b>475</b>
<b>15.1</b>	<b>注意力机制的改进 . . . . .</b>	<b>475</b>
15.1.1	局部信息建模 . . . . .	476
15.1.2	多分支结构 . . . . .	482
15.1.3	引入循环机制 . . . . .	484
15.1.4	高效的自注意力模型 . . . . .	485
<b>15.2</b>	<b>神经网络连接优化及深层模型 . . . . .</b>	<b>487</b>
15.2.1	Post-Norm vs Pre-Norm . . . . .	487
15.2.2	高效信息传递 . . . . .	489
15.2.3	面向深层模型的参数初始化策略 . . . . .	493
15.2.4	深层模型的训练加速 . . . . .	497
15.2.5	深层模型的健壮性训练 . . . . .	500
<b>15.3</b>	<b>基于句法的神经机器翻译模型 . . . . .</b>	<b>501</b>
15.3.1	编码器使用句法信息 . . . . .	502
15.3.2	解码器使用句法信息 . . . . .	507
<b>15.4</b>	<b>基于结构搜索的翻译模型优化 . . . . .</b>	<b>509</b>
15.4.1	神经网络结构搜索 . . . . .	509
15.4.2	结构搜索的基本方法 . . . . .	510
15.4.3	机器翻译任务下的结构搜索 . . . . .	514
<b>15.5</b>	<b>小结及拓展阅读 . . . . .</b>	<b>516</b>
<b>16</b>	<b>低资源神经机器翻译 . . . . .</b>	<b>517</b>
<b>16.1</b>	<b>数据的有效使用 . . . . .</b>	<b>517</b>
16.1.1	数据增强 . . . . .	518
16.1.2	基于语言模型的方法 . . . . .	523
<b>16.2</b>	<b>双向翻译模型 . . . . .</b>	<b>529</b>
16.2.1	双向训练 . . . . .	529
16.2.2	对偶学习 . . . . .	530

<b>16.3 多语言翻译模型</b>	532
16.3.1 基于枢轴语言的方法	532
16.3.2 基于知识蒸馏的方法	533
16.3.3 基于迁移学习的方法	534
<b>16.4 无监督机器翻译</b>	537
16.4.1 无监督词典归纳	538
16.4.2 无监督统计机器翻译	541
16.4.3 无监督神经机器翻译	542
<b>16.5 领域适应</b>	546
16.5.1 基于数据的方法	547
16.5.2 基于模型的方法	548
<b>16.6 小结及拓展阅读</b>	550
<b>17 多模态、多层次机器翻译</b>	553
<b>17.1 机器翻译需要更多的上下文</b>	553
<b>17.2 语音翻译</b>	554
17.2.1 音频处理	555
17.2.2 级联式语音翻译	556
17.2.3 端到端语音翻译	558
<b>17.3 图像翻译</b>	562
17.3.1 基于图像增强的文本翻译	563
17.3.2 图像到文本的翻译	565
17.3.3 图像、文本到图像的翻译	568
<b>17.4 篇章级翻译</b>	568
17.4.1 篇章级翻译的挑战	569
17.4.2 篇章级翻译的评价	570
17.4.3 篇章级翻译的建模	570
17.4.4 在推断阶段结合篇章上下文	574
<b>17.5 小结及拓展阅读</b>	576
<b>18 机器翻译应用技术</b>	577
<b>18.1 机器翻译的应用并不简单</b>	577
<b>18.2 增量式模型优化</b>	578
<b>18.3 交互式机器翻译</b>	580
<b>18.4 翻译结果的可干预性</b>	581
<b>18.5 小设备机器翻译</b>	583

18.6	机器翻译系统的部署.....	584
18.7	机器翻译的应用场景.....	586
	<b>随笔 .....</b>	<b>588</b>
	<b>后记 .....</b>	<b>593</b>

V

## 附录

<b>A</b>	<b>附录 A .....</b>	<b>597</b>
A.1	统计机器翻译开源系统.....	597
A.2	神经机器翻译开源系统.....	599
<b>B</b>	<b>附录 B .....</b>	<b>601</b>
B.1	公开评测任务 .....	601
B.2	基准数据集 .....	603
B.3	平行语料.....	604
<b>C</b>	<b>附录 C .....</b>	<b>607</b>
C.1	IBM 模型 2 训练方法 .....	607
C.2	IBM 模型 3 训练方法 .....	608
C.3	IBM 模型 4 训练方法 .....	610
C.4	IBM 模型 5 训练方法 .....	612

# 机器翻译基础

<b>1</b>	<b>机器翻译简介</b>	<b>23</b>
1.1	机器翻译的概念	
1.2	机器翻译简史	
1.3	机器翻译现状及挑战	
1.4	基于规则的方法	
1.5	数据驱动的方法	
1.6	推荐学习资源	
<b>2</b>	<b>统计语言建模基础</b>	<b>45</b>
2.1	概率论基础	
2.2	掷骰子游戏	
2.3	$n$ -gram 语言模型	
2.4	预测与搜索	
2.5	小结及拓展阅读	
<b>3</b>	<b>词法分析和语法分析基础</b>	<b>77</b>
3.1	问题概述	
3.2	中文分词	
3.3	命名实体识别	
3.4	句法分析	
3.5	小结及拓展阅读	
<b>4</b>	<b>翻译质量评价</b>	<b>105</b>
4.1	译文质量评价所面临的挑战	
4.2	人工评价	
4.3	有参考答案的自动评价	
4.4	无参考答案的自动评价	
4.5	小结及拓展阅读	





# 1. 机器翻译简介

## 1.1 机器翻译的概念

从广义上来讲，“翻译”是指把一个事物转化为另一个事物的过程。这个概念多使用在对序列的转化上，比如，计算机程序的编译、自然语言文字的翻译、生物蛋白质的合成等。在程序编译中，高级语言编写的程序经过一系列的处理后转化为可执行的目标程序，这是一种从高级程序语言到低级程序语言的“翻译”。在人类语言的翻译中，一种语言文字通过人脑转化为另一种语言表达，这是一种自然语言的“翻译”。在蛋白质合成的第一步，RNA分子序列转化为特定的氨基酸序列，这是一种生物学遗传信息的“翻译”。甚至说给上联对出下联、给一幅图片写出图片的主题等都可以被看作是“翻译”的过程。

这里更加关注人类语言之间的翻译问题，即自然语言的翻译。如图1.1所示，通过计算机可以将一段汉语文字自动转化为英语文字，汉语被称为**源语言**（Source Language），英语被称为**目标语言**（Target Language）。

一直以来，文字的翻译往往是由人完成。让计算机像人一样进行翻译似乎还是电影中的桥段，因为很难想象语言的多样性和复杂性可以用计算机语言进行描述。但是时至今日，人工智能技术的发展已经大大超越了人类传统的认知，用计算机进行自动翻译也不再是一种梦想，它已经深入到人们生活的很多方面，并发挥着重要作用。而这种由计算机进行自动翻译的过程也被称作**机器翻译**（Machine Translation）。类似地，自动翻译、智能翻译、多语言自动转换等概念也是指同样的事情。如果将今天的机器翻译和人工翻译进行对比，可以发现机器翻译系统所生成的译文还不够完

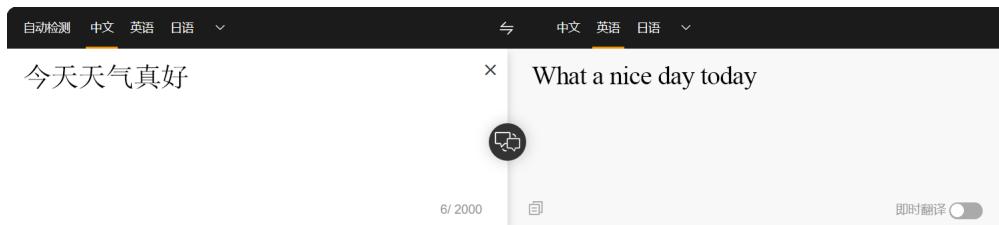


图 1.1 通过计算机将汉语翻译为英语

美，甚至有时翻译质量非常差，但是它的生成速度快且成本低廉，更为重要的是机器翻译系统可以从大量数据中不断学习和进化。

人工翻译尽管精度很高，但是费时费力。当需要翻译大量的文本且精度要求不那么高时，比如海量数据的浏览型任务，机器翻译的优势就体现出来了。对于人工作业无法完成的事情，使用机器翻译可能只需花费几个小时甚至几分钟就能完成。这就类似于拿着锄头耕地种庄稼和使用现代化机器作业之间的区别。

实现机器翻译往往需要多个学科知识的融合，如数学、语言学、计算机科学、心理学等等。而最终呈现给使用者的是一套软件系统——机器翻译系统。通俗来讲，机器翻译系统就是一个可以在计算机上运行的软件工具，与人们使用的其他软件一样，只不过机器翻译系统是由“不可见的程序”组成。虽然这个系统非常复杂，但是呈现出来的形式却很简单，比如输入是待翻译的句子或文本，输出是译文句子或文本。

用机器进行翻译的想法可以追溯到电子计算机产生之前，发展过程中也经历了多个范式的变迁，现代机器翻译系统大多是基于数据驱动的方法——从数据中自动学习翻译知识，并运用这些知识对新的文本进行翻译。

从机器翻译系统的组成上来看，通常可以抽象为两个部分，如图1.2所示：

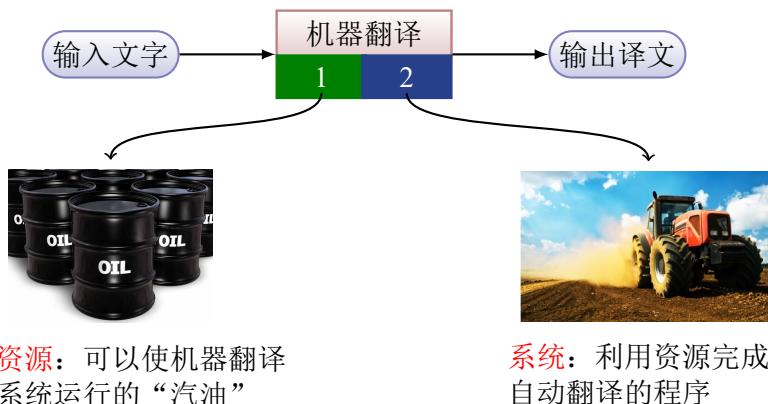


图 1.2 机器翻译系统的组成

- **资源：**如果把机器翻译系统比作一辆汽车，资源就好比是可以使汽车运行的“汽

油”，它包括很多内容，如翻译规则、双（单）语数据、知识库等翻译知识，且这些“知识”都是计算机可读的。值得一提的是，如果没有翻译资源的支持，任何机器翻译系统都无法运行起来。

- **系统：**机器翻译算法的程序实现被称作系统，也就是机器翻译研究人员开发的软件。无论是翻译规则、翻译模板还是统计模型中的参数都需要通过机器翻译系统进行读取和使用。

构建一个强大的机器翻译系统需要“资源”和“系统”两方面共同作用。在资源方面，随着语料库语言学的发展，已经有大量的高质量的双语和单语数据（称为语料）被整理并且被电子化存储，因此可以说具备了研发机器翻译系统所需要的语料基础。特别是像英语、汉语等世界主流语种，相关语料资源已经非常丰富，这也大大加速了相关研究的进展。当然，对于一些稀缺资源语种或者特殊的领域，语料库中的语料仍然匮乏，但是这些并不影响机器翻译领域整体的发展速度。因此在现有语料库的基础上，很多研究者把精力集中在“系统”研发上。

## 1.2 机器翻译简史

虽然翻译这个概念在人类历史中已经存在了上千年，但机器翻译发展至今只有七十余年历史。纵观机器翻译的发展，历程曲折又耐人寻味，可以说，回顾机器翻译的历史对深入理解相关技术方法会有很好的启发，甚至对了解整个自然语言处理领域的发展也有启示作用。

### 1.2.1 人工翻译

人类形成语言文字的过程中逐渐形成了翻译的概念。一个著名的标志性证据是罗塞塔石碑（Rosetta Stone），如图1.3所示。这个石碑制作于公元前196年，据说是可供考证的最久远的记载平行文字的历史遗迹。石碑由上至下刻有同一段埃及国王诏书的三种语言版本，最上面是古埃及象形文，中间是埃及草书，最下面是古希腊文。可以明显看出石碑上中下雕刻的文字的纹理是不同的。尽管用不同的语言文字描述同一件事在今天看来很常见，但是这在生产力低下的两千年前是很罕见的。很多人认为罗塞塔石碑是标志翻译或人工翻译的一个起点。目前罗塞塔石碑保存于大英博物馆，并成为该馆最具代表性的镇馆之宝之一。

在此之后，更多的翻译工作在文化和知识传播中开展。其中一个典型代表是宗教文献的翻译。宗教是人类意识形态的一个重要载体，为了宣传教义，人们编写了大量的宗教文献。在西方，一项最早被记录的翻译活动是将旧约圣经（希伯来文及埃兰文）翻译为希腊文版本。迄今为止人类历史上翻译版本最多的书就是圣经。在中国唐代，有一位世界性的文化人物——玄奘，他不仅是佛学家、旅行家，还是翻译家。玄奘西行求法归来后把全部的心血和智慧奉献给了译经事业，在助手们的帮助下，共翻译佛教经论74部，1335卷，每卷万字左右，合计1335万字，占去整个

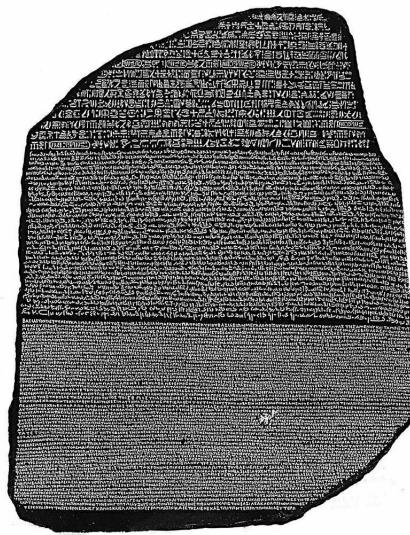


图 1.3 罗塞塔石碑

唐代译经总数的一半以上<sup>[1]</sup>，树立了我国古代翻译思想的光辉典范。

翻译在人类历史长河中起到了重要的作用。一方面，由于语言文字、文化和地理位置的差异性，使得翻译成为一个重要的需求；另一方面，翻译也加速了不同文明的融会贯通，促进了世界的发展。今天，翻译已经成为重要的行业之一，包括各个高校也都设立了翻译及相关专业，相关人才不断涌现。据《2019 年中国语言服务行业发展报告》<sup>[2]</sup> 统计：全球语言服务产值预计将首次接近 500 亿美元；中国涉及语言服务的在营企业 360,000 余家，语言服务为主营业务的在营企业近万家，总产值超过 300 亿元，年增长 3% 以上；全国开设外语类专业的高校数量多达上千所，其中设立有翻译硕士（MTI）和翻译本科（BTI）专业的院校分别有 250 余所和 280 余所，其中仅 MTI 的累计招生数就高达 6 万余人<sup>[3]</sup>。当然，面对着巨大的需求，如何使用机器辅助翻译等技术手段提高人工翻译效率，也是人工翻译和机器翻译领域需要共同探索的方向。

## 1.2.2 机器翻译的萌芽

人工翻译已经存在了上千年，而机器翻译又起源于什么时候呢？机器翻译跌宕起伏的发展史可以分为萌芽期、受挫期、快速成长期和爆发期四个阶段。

早在 17 世纪，如 Descartes、Leibniz、Cave Beck、Athanasius Kircher 和 Johann Joachim Becher 等很多学者就提出采用机器词典（电子词典）来克服语言障碍的想法<sup>[4]</sup>，这种想法在当时是很超前的。随着语言学、计算机科学等学科的发展，在 19 世纪 30 年代使用计算模型进行自动翻译的思想开始萌芽，如当时法国科学家 Georges Artsrouni 就提出用机器来进行翻译的想法。只是那时依然没有合适的实现手段，所以这种想法的合理性无法被证实。

随着第二次世界大战爆发，对文字进行加密和解密成为重要的军事需求，这也使得数学和密码学变得相当发达。在战争结束一年后，世界上第一台通用电子数字计算机于 1946 年研制成功，至此使用机器进行翻译有了真正实现的可能。

基于战时密码学领域与通讯领域的研究，Claude Elwood Shannon 在 1948 年提出使用“噪声信道”描述语言的传输过程，并借用热力学中的“熵”(Entropy) 来刻画消息中的信息量<sup>[5]</sup>。次年，Shannon 与 Warren Weaver 更是合著了著名的 *The Mathematical Theory of Communication*<sup>[6]</sup>，这些工作都为后期的统计机器翻译打下了理论基础。

1949 年，Weaver 撰写了一篇名为 *TRANSLATION* 的备忘录<sup>[7]</sup>，在这个备忘录中 Weaver 提出用密码学的方法解决人类语言翻译任务的想法，比如把汉语看成英语的一个加密文本，那么将汉语翻译成英语就类似于解密的过程。并且在这篇备忘录中第一次提出了机器翻译，正式开创了机器翻译的概念，这个概念一直沿用至今。虽然，在那个年代进行机器翻译的研究条件并不成熟，包括使用加密解密技术进行自动翻译的很多尝试很快也被验证是不可行的，但是这些早期的探索为后来机器翻译的发展提供了思想的火种。

### 1.2.3 机器翻译的受挫

随着电子计算机的发展，研究者开始尝试使用计算机来进行自动翻译。1954 年，美国乔治敦大学在 IBM 公司支持下，启动了第一次真正的机器翻译实验。翻译的目标是将几个简单的俄语句子翻译成为英语，翻译系统包含 6 条翻译规则和 250 词汇。这次翻译实验中测试了 50 个化学文本句子，取得了初步成功。在某种意义上来说，这个实验显示了采用基于词典和翻译规则的方法可以实现机器翻译过程。虽然只是取得了初步成功，但却引起了苏联、英国和日本研究机构的机器翻译研究热，大大推动了早期机器翻译的研究进展。

1957 年，Noam Chomsky 在 *Syntactic Structures* 中描述了转换生成语法<sup>[8]</sup>，并使用数学方法来研究自然语言，建立了包括上下文有关语法、上下文无关语法等 4 种类型的语法。这些工作最终为今天计算机中广泛使用的“形式语言”奠定了基础。而他的思想也深深地影响了同时期的语言学和自然语言处理领域的学者。特别的是，早期基于规则的机器翻译中也大量使用了这些思想。

虽然在这段时间，使用机器进行翻译的议题越加火热，但是事情并不总是一帆风顺，怀疑论者对机器翻译一直存有质疑，并很容易找出一些机器翻译无法解决的问题。自然地，人们也期望能够客观地评估一下机器翻译的可行性。当时美国基金资助组织委任自动语言处理咨询会承担了这项任务。经过近两年的调查与分析，该委员会于 1966 年 11 月公布了一个题为 *LANGUAGE AND MACHINES* 的报告（图1.4），即 ALPAC 报告。该报告全面否定了机器翻译的可行性，为机器翻译的研究泼了一盆冷水。

随后美国政府终止了对机器翻译研究的支持，这导致整个产业界和学术界都开始回避机器翻译。没有了政府的支持，企业也无法进行大规模投入，机器翻译的研

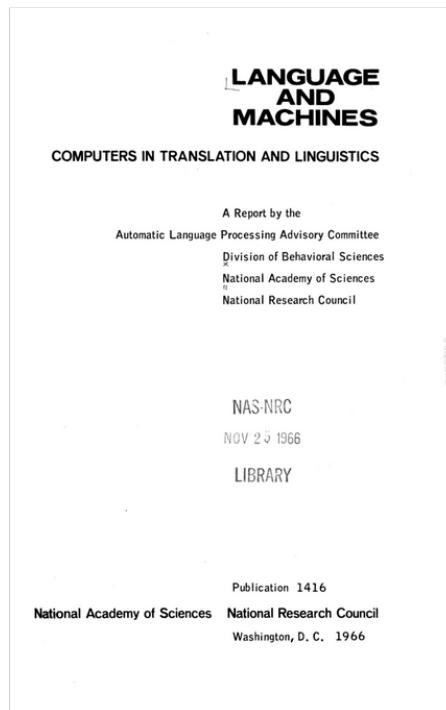


图 1.4 ALPAC 报告

究就此受挫。

从历史上看，包括机器翻译在内，很多人工智能领域在那个年代并不受“待见”，其主要原因在于当时的技术水平还比较低，而大家又对机器翻译等技术的期望过高。最后发现，当时的机器翻译水平无法满足实际需要，因此转而排斥它。但是，也正是这一盆冷水，让研究人员可以更加冷静地思考机器翻译的发展方向，为后来的爆发蓄力。

#### 1.2.4 机器翻译的快速成长

事物的发展都是螺旋式上升的，机器翻译也是一样。早期基于规则的机器翻译方法需要人来书写规则，虽然对少部分句子具有较高的翻译精度，可是对翻译现象的覆盖度有限，而且对规则或者模板中的噪声非常敏感，系统健壮性差。

上世纪 70 年代中后期，特别是 80 年代到 90 年代初，国家之间往来日益密切，而不同语言之间形成的交流障碍愈发严重，传统的人工作业方式已经远远不能满足需求。与此同时，语料库语言学的发展也为机器翻译提供了新的思路。一方面，随着传统纸质文字资料不断电子化，计算机可读的语料越来越多，这使得人们可以用计算机对语言规律进行统计分析。另一方面，随着可用数据越来越多，用数学模型描述这些数据中的规律并进行推理逐渐成为可能。这也衍生出一类数学建模方法——**数据驱动**（Data-driven）的方法。同时这类方法也成为了随后出现的统计机器翻译的基础，比如，IBM 研究人员提出的基于噪声信道模型的 5 种统计翻译模型<sup>[9, 10]</sup>。

基于数据驱动的方法不依赖于人书写的规则，机器翻译的建模、训练和推断都可以自动地从数据中学习。这使得整个机器翻译的范式发生了翻天覆地的变化，比如，日本学者长尾真提出的基于实例的方法<sup>[11, 12]</sup> 和统计机器翻译<sup>[9, 10]</sup> 就是在此期间兴起的。此外，这样的方法使得机器翻译系统的开发代价大大降低。

从上世纪 90 年代到本世纪初，随着语料库的完善与高性能计算机的发展，统计机器翻译很快成为了当时机器翻译研究与应用的代表性方法。一个标志性的事件是谷歌公司推出了一个在线的免费自动翻译服务，也就是大家熟知的谷歌翻译。这使得机器翻译这种“高大上”的技术快速进入人们的生活，而不再是束之高阁的科研想法。随着机器翻译不断走向实用，机器翻译的应用也越来越多，这反过来促进了机器翻译的研究进程。比如，在 2005-2015 年间，统计机器翻译这个主题几乎统治了 ACL 等自然语言处理相关方向顶级会议的论文，可见其在当时的影响力。

### 1.2.5 机器翻译的爆发

进入二十一世纪，统计机器翻译拉开了黄金发展期的序幕。在这一时期，各种基于统计机器翻译模型层出不穷，经典的基于短语的模型和基于句法的模型也先后被提出。在 2013 年以后，机器学习的进步带来了机器翻译技术的进一步提升。特别是基于神经网络的深度学习方法在机器视觉、语音识别中被成功应用，带来性能的飞跃式提升。很快，深度学习方法也被用于机器翻译。

实际上，对于机器翻译任务来说，深度学习方法被广泛使用也是一种必然，原因如下：

- 第一，端到端学习不依赖于过多的先验假设。在统计机器翻译时代，模型设计或多或少会对翻译的过程进行假设，称为隐藏结构假设。比如基于短语的模型假设：源语言和目标语言都会被切分成短语序列，这些短语之间存在某种对齐关系。这种假设既有优点也有缺点：一方面，该假设有助于模型融入人类的先验知识，比如，统计机器翻译中一些规则的设计就借鉴了语言学的相关概念；另一方面，假设越多模型受到的限制也越多。如果假设是正确的，模型可以很好地描述问题。但如果假设错误，那么模型就可能产生偏差。深度学习不依赖于先验知识，也不需要手工设计特征，模型直接从输入和输出的映射上进行学习（端到端学习），这样也在一定程度上避免了隐藏结构假设造成的偏差。
- 第二，神经网络的连续空间模型有更强的表示能力。机器翻译中的一个基本问题是：如何表示一个句子？统计机器翻译把句子的生成过程看作是短语或者规则的推导，这本质上是一个离散空间上的符号系统。深度学习把传统的基于离散化的表示变成了连续空间的表示。比如，用实数空间的分布式表示代替了离散化的词语表示，而整个句子可以被描述为一个实数向量。这使得翻译问题可以在连续空间上描述，进而大大缓解了传统离散空间模型维度灾难等问题。更重要的是，连续空间模型可以用梯度下降等方法进行优化，具有很好的数学性质并且易于实现。

- 第三，深度网络学习算法的发展和 GPU（Graphics Processing Unit）等并行计算设备为训练神经网络提供了可能。早期的基于神经网络的方法一直没有在机器翻译甚至自然语言处理领域得到大规模应用，其中一个重要的原因是这类方法需要大量的浮点运算，但是以前计算机的计算能力无法达到这个要求。随着 GPU 等并行计算设备的进步，训练大规模神经网络也变为了可能。现在已经可以在几亿、几十亿，甚至上百亿句对上训练机器翻译系统，系统研发的周期越来越短，进展日新月异。

今天，神经机器翻译已经成为新的范式，与统计机器翻译一同推动了机器翻译技术与应用产品的发展。比如，从世界上著名的机器翻译比赛 WMT 和 CCMT 中就可以看出这个趋势。如图1.5所示，其中左图是 WMT 19 国际机器翻译比赛的参赛队伍的截图，这些参赛队伍基本上都在使用深度学习完成机器翻译的建模。而在 WMT 19 各个项目夺冠系统中（1.5右图），神经机器翻译也占据了主导地位。

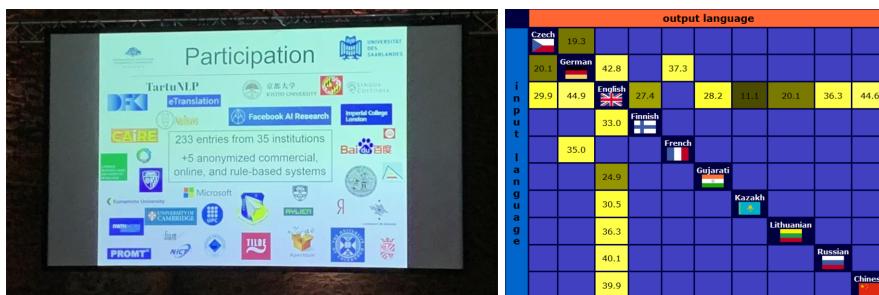


图 1.5 WMT 19 国际机器翻译大赛（左：WMT 19 参赛队伍；右：WMT 19 各项目的最好分数）

值得一提的是，近些年神经机器翻译的快速发展也得益于产业界的关注。各大互联网企业和机器翻译技术研发机构都对神经机器翻译的模型和实践方法给予了很大贡献。很多企业凭借自身人才和基础设施方面的优势，先后推出了以神经机器翻译为内核的产品及服务，相关技术方法已经在大规模应用中得到验证，大大推动了机器翻译的产业化进程，而且这种趋势在不断加强，机器翻译的前景也更加宽广。

## 1.3 机器翻译现状及挑战

机器翻译技术发展到今天已经过无数次迭代，技术范式也经过若干次更替，近些年机器翻译的应用也如雨后春笋相继浮现。今天的机器翻译的质量究竟如何呢？乐观地说，在很多特定的条件下，机器翻译的译文结果是非常不错的，甚至可以接近人工翻译的结果。然而，在开放式翻译任务中，机器翻译的结果还并不完美。更严格来说，机器翻译的质量远没有达到人们所期望的程度。对于有些人提到的“机器翻译将代替人工翻译”也并不是事实。比如，在高精度同声传译任务中，机器翻译仍需要更多打磨；再比如，针对于小说的翻译，机器翻译还无法做到与人工翻译媲美；甚至有人尝试用机器翻译系统翻译中国古代诗词，这里更多的是娱乐的味道。但是毫无疑问的是，机器翻译可以帮助人类，甚至有朝一日可以代替一些低端的人工翻

译工作。

图1.6展示了机器翻译和人工翻译质量的一个对比结果。在汉语到英语的新闻翻译任务中，如果对译文进行人工评价（五分制），那么机器翻译的译文得分为3.9分，人工译文得分为4.7分（人的翻译也不是完美的）。可见，在这个任务中机器翻译表现不错，但是与人还有一定差距。如果换一种方式评价，把人的译文作为参考答案，用机器翻译的译文与其进行比对（百分制），会发现机器翻译的得分只有47分。当然，这个结果并不是说机器翻译的译文质量很差，它更多的是表明机器翻译系统可以生成一些与人工翻译不同的译文，机器翻译也具有一定的创造性。这也类似于，很多围棋选手都想向AlphaGo学习，因为智能围棋系统也可以走出一些人类从未走过的妙招。

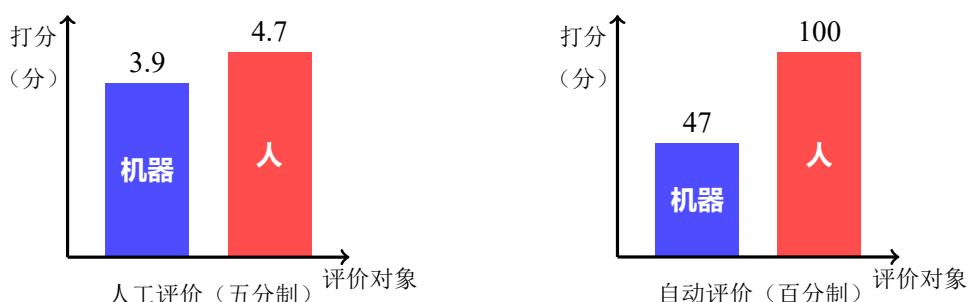


图 1.6 机器翻译与人工翻译性能对比 (汉英新闻领域翻译)

图1.7展示了一个真实的汉语到英语翻译实例。对比发现，机器翻译与人工翻译还是存在差距的，特别是在翻译一些具有感情色彩的词语时，机器翻译的译文缺一些味道。那么，机器翻译一点用都没有吗？显然不是。实际上，如果考虑翻译速度与翻译代价，机器翻译的价值是无可比拟的。还是同一个例子，翻译一篇短文如果人工翻译需要30分钟甚至更长时间，那么机器翻译仅仅需要两秒，换种情况思考，如果有100万篇这样的文档，其人工翻译的成本根本无法想象，消耗的时间更是难以计算，而计算机集群仅仅需要一天，而且只有电力的消耗。

源语 言：从前有一个小岛，上面住着快乐、悲哀、知识和爱，还有其他各种情感。一天，情感们得知小岛快要下沉了。于是，大家都准备船只，离开小岛，只有爱决定留下来，她想坚持到最后一刻。过了几天，小岛真的要下沉了，爱想请人帮忙。
机器翻译：Once upon a time there was an island <u>on which</u> lived happiness,sorrow,knowledge,love and other emotions. One day, <u>the</u> emotions learned that the island was going to sink. As a result,everyone pre-pared the boat and <u>left the island</u> . Only Love decided to stay. She <u>wanted to stick</u> to it until the last moment. After a few days, the island was really going to sink and love <u>wanted help</u> .
人工翻译：Once upon a time, there was a small island <u>where</u> lived all kinds of emotions like JOY,SADNESS, KNOWLEDGE, and LOVE. One day, <u>these</u> emotions found that the island was sinking, so one by one they prepared the boat and <u>planned to leave</u> . None but LOVE chose to stay there. She <u>was determined to persist</u> till the last moment. A few days later, almost the whole island sunk into the sea, and LOVE had to <u>seek for help</u> .

图 1.7 机器翻译与人工翻译结果对比实例

虽然机器翻译有上述优点，但仍然面临一些挑战：

- **自然语言翻译问题的复杂性极高。**语言是人类进化的最高成就之一，自然语言具有高度的概括性、灵活性、多样性，这些都很难用几个简单的模型和算法进行描述。因此，翻译问题的数学建模和计算机程序实现难度很大。虽然近几年AlphaGo等人工智能系统在围棋等领域取得了令人瞩目的成绩，但是，相比翻译来说，围棋等棋类任务仍然“简单”。正如不同人对同一句话的理解不尽相同，一个句子往往不存在绝对的标准译文，其潜在的译文几乎是不可穷尽的。甚至人类译员在翻译一个句子、一个单词的时候，都要考虑整个篇章的上下文语境。这些难点都不是传统棋类任务所具有的。
- **计算机的“理解”与人类的“理解”存在鸿沟。**人类一直希望把自己翻译时所使用的知识描述出来，并用计算机程序进行实现，例如早期基于规则的机器翻译方法就源自这个思想。但是，经过实践发现，人和计算机在“理解”自然语言上存在着明显差异。首先，人类的语言能力是经过长时间在多种外部环境因素共同作用下形成的，这种能力很难用计算机准确地刻画。况且人类的语言知识本身就很难描述，更不用说让计算机来理解；其次，人和机器翻译系统理解语言的目的不一样。人理解和使用语言是为了进行生活和工作，而机器翻译系统更多的是为了对某些数学上定义的目标函数进行优化。也就是说，机器翻译系统关注的是翻译这个单一目标，而并不是像人一样进行复杂的活动；此外，人和计算机的运行方式有着本质区别。人类语言能力的生物学机理与机器翻译系统所使用的计算模型本质上是不同的，机器翻译系统使用的是其自身能够理解的“知识”，比如，统计学上的词语表示。这种“知识”并不需要人来理解，当然从系统开发的角度，计算机也并不需要理解人是如何思考的。
- **单一的方法无法解决多样的翻译问题。**首先，**语种的多样性**会导致任意两种语言之间的翻译实际上都是不同的翻译任务。比如，世界上存在的语言多达几千种，如果选择任意两种语言进行互译就会产生上百万种翻译方向。虽然已经有研究者尝试用同一个框架甚至同一个翻译系统进行全语种的翻译，但是这类系统离真正可用还有很远的距离；其次，**不同的领域**，**不同的应用场景**对翻译也有不同的需求。比如，文学作品的翻译和新闻的翻译就有不同、口译和笔译也有不同，类似的情况不胜枚举。以上这些都增加了计算机对翻译进行建模的难度；再次，对于机器翻译来说，充足的高质量数据是必要的，但是不同语种、不同领域、不同应用场景所拥有的**数据量**有明显差异，很多语种甚至几乎没有可用的数据，这时开发机器翻译系统的难度可想而知。值得注意的是，现在的机器翻译还无法像人类一样在学习少量样例的情况下进行举一反三，因此数据稀缺情况下的机器翻译也给研究者带来了很大的挑战。

显然，实现机器翻译并不简单，甚至有人把机器翻译看作是实现人工智能的终极目标。幸运的是，今天的机器翻译无论从技术方法上还是从应用上都有了巨大的

飞跃，很多问题在不断被求解。如果读者看到过十年前机器翻译的结果，再对比今天的结果，一定会感叹翻译质量的今非昔比，很多译文已经非常准确且流畅。从当今机器翻译的前沿技术看，近三十年机器翻译的进步更多得益于基于数据驱动方法和统计建模方法的使用。特别是近些年深度学习等基于表示学习的端到端方法使得机器翻译的水平达到了新高度。因此，本书将会对基于统计建模和深度学习方法的机器翻译模型、方法和系统实现进行全面介绍和分析，希望这些论述可以对相关内容的学习和科研工作提供参考。

## 1.4 基于规则的方法

机器翻译技术大体上可以分为三种方法，分别为基于规则的机器翻译、统计机器翻译以及神经机器翻译。第一代机器翻译技术是主要使用基于规则的机器翻译方法，其主要思想是通过形式文法定义的规则引入源语言和目标语中的语言学知识。此类方法在机器翻译技术诞生之初就被人所关注，特别是在上世纪 70 年代，以基于规则方法为代表的专家系统是人工智能中最具代表性的研究领域。甚至到了统计机器翻译时代，很多系统中也大量地使用了基于规则的翻译知识表达形式。

早期，基于规则的机器翻译大多依赖人工定义及书写的规则。主要有两类方法<sup>[13, 14, 15]</sup>：一类是基于转换规则的机器翻译方法，简称转换法。另一类是基于中间语言的方法。它们都以词典和人工书写的规则库作为翻译知识，用一系列规则的组合完成翻译。

### 1.4.1 规则的定义

规则就像语言中的“If-then”语句，如果满足条件，则执行相应的语义动作。比如，可以将待翻译句子中的某个词，使用目标语言单词进行替换，但是这种替换并非随意的，而是在语言学知识的指导下进行的。

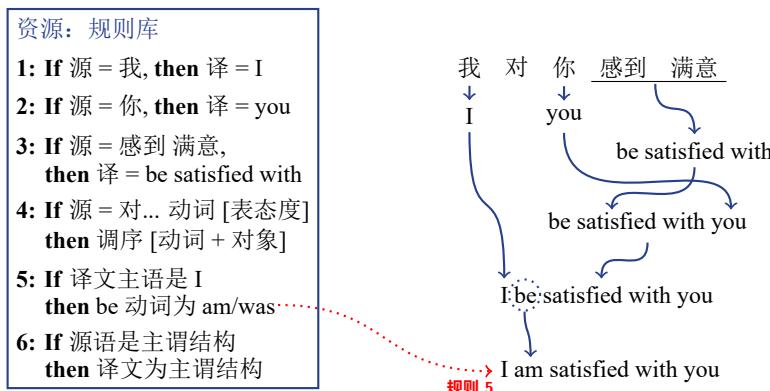


图 1.8 基于规则的机器翻译的示例图（左：规则库；右：规则匹配结果）

图1.8展示了一个使用转换法进行翻译的实例。这里，利用一个简单的汉译英规

则库完成对句子“我对你感到满意”的翻译。当翻译“我”时，从规则库中找到规则1，该规则表示遇到单词“我”就翻译为“I”；类似地，也可以从规则库中找到规则4，该规则表示翻译调序，即将单词“you”放到“be satisfied with”后面。这种通过规则表示单词之间的对应关系也为统计机器翻译方法提供了思路。如统计机器翻译中，基于短语的翻译模型使用短语对对原文进行替换，详细描述可以参考第七章。

在上述例子中可以发现，规则不仅仅可以翻译句子之间单词的对应，如规则1，还可以表示句法甚至语法之间的对应，如规则6。因此基于规则的方法可以分成多个层次，如图1.9所示。图中不同的层次表示采用不同的知识来书写规则，进而完成机器翻译过程。对于翻译问题，可以构建不同层次的基于规则的机器翻译系统。这里包括四个层次，分别为：词汇转换、句法转换、语义转换和中间语言层。其中，上层可以继承下层的翻译知识，比如说句法转换层会利用词汇转换层知识。早期基于规则的方法属于词汇转换层。

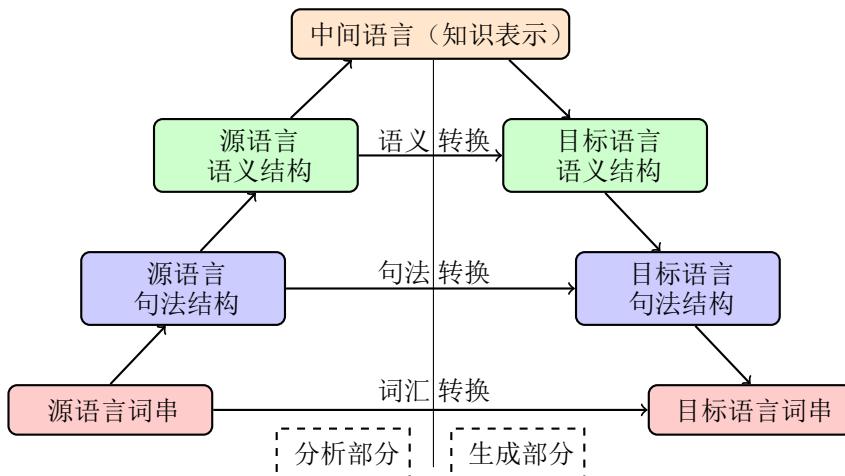


图 1.9 基于规则的机器翻译方法的四个层次<sup>[16]</sup>

### 1.4.2 转换法

通常一个典型的**基于转换规则的机器翻译**（Transfer-based Translation）的过程可以被视为“独立分析-相关转换-独立生成”的过程<sup>[17]</sup>。如图1.10所示，这些过程可以分成六个步骤，其中每一个步骤都是通过相应的翻译规则来完成。比如，第一个步骤中需要构建源语词法分析规则，第二个步骤中需要构建源语句法分析规则，第三个和第四个步骤中需要构建转换规则，其中包括源语言-目标语言词汇和结构转换规则等等。

转换法的目标就是使用规则定义的词法和句法，将源语言句子分解成为一个蕴含语言学标志的结构。如一个汉语句子“她把一束花放在桌上。”，经过词法和句法分析之后可以被表示成如图1.11所示的结构，这个结构就是图1.10中的源语言句子结

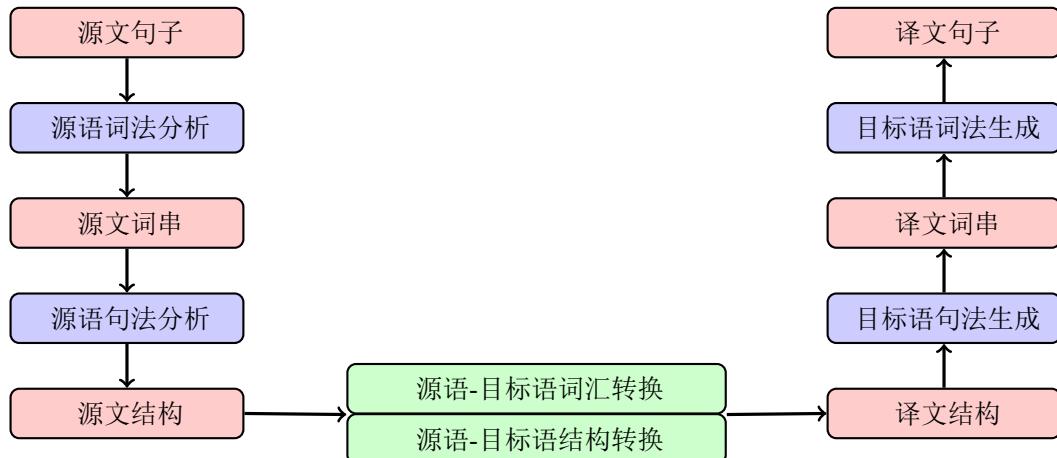


图 1.10 基于转换规则的机器翻译过程

构。这种使用语言学提取句子结构化表示，并使用某种规则匹配源语言结构和目标语言结构的方式也为第八章将要介绍的基于语言学句法的模型提供了思路。

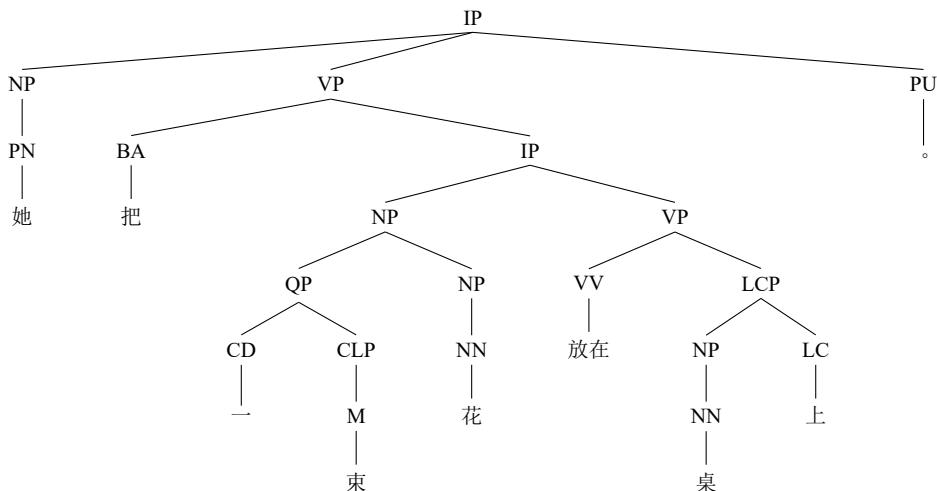


图 1.11 一个汉语句子的结构表示 (句法树)

在转换法中，翻译规则通常会分成两类：通用规则和个性规则。所谓通用的规则主要用于句法分析、语义分析、结构转换和句法生成等，是不具体依赖于某个源语言或者目标语言词汇而设计的翻译规则；个性规则通常以具体源语言词汇来做索引，比如图1.8中规则 5 就是针对主语是 “I”的个性规则，它直接针对某个具体词汇进行分析和翻译。

### 1.4.3 基于中间语言的方法

基于转换的方法可以通过词汇层、句法层和语义层完成从源语到目标语的转换过程，虽然采用了独立分析和独立生成两个子过程，但中间包含一个从源语到目标语的相关转换过程。这就会导致一个实际问题，假设需要实现  $N$  个语言之间互译的机器翻译系统，采用基于转换的方法，需要构建  $N(N - 1)$  个不同的机器翻译系统，这个构建代价是非常高的。为了解决这个问题，一种有效的解决方案是使用**基于中间语言的机器翻译**（Interlingua-based Translation）方法。

如图1.12所示，基于中间语言方法的最大特点就是采用了一个称之为“中间语言”的知识表示结构，将“中间语言”作为独立源语言分析和独立目标语生成的桥梁，真正实现独立分析和独立生成。并且在基于中间语言的方法中不涉及“相关转换”这个过程，这一点与基于转换的方法有很大区别。

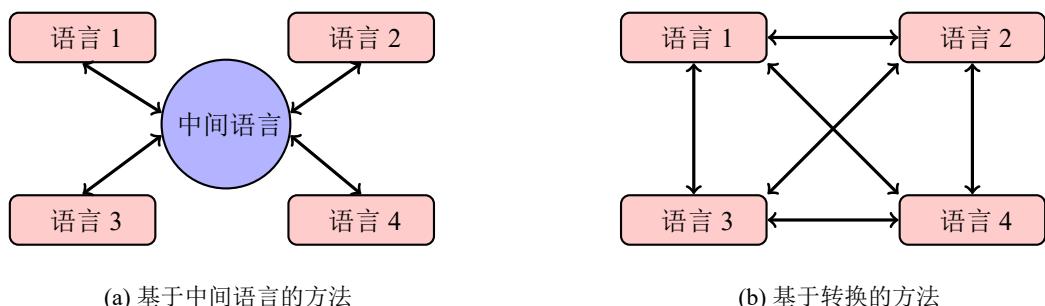


图 1.12 基于中间语言的方法 (a) 与基于转换的方法 (b)

从图1.9可以发现，中间语言（知识表示）处于最顶端，本质上是独立于源语言和目标语言的，这也是基于中间语言的方法可以将分析过程和生成过程分开的原因。

虽然基于中间语言的方法有上述优点，但如何定义中间语言是一个关键问题。严格说，所谓中间语言本身是一种知识表示结构，承载着源语言句子的分析结果，应该包含和体现尽可能多的源语言知识。如果中间语言的表示能力不强，会导致源语言句子信息丢失，这自然会影响目标语生成结果。

在基于规则的机器翻译方法中，构建中间语言结构的知识表示方式有很多，比较常见的是语法树、语义网、逻辑结构表示或者多种结构的融合等。但不管哪种方法，实际上都无法充分地表达源语言句子所携带的信息。因此，在早期的基于规则的机器翻译研究中，基于中间语言的方法明显弱于基于转换的机器翻译方法。不过，近些年随着神经机器翻译等方法的兴起，使用统一的中间表示来刻画句子又受到了广泛关注。但是，神经机器翻译中的“中间表示”并不是规则系统中的中间语言，二者有着本质区别，这部分内容将会在第十章进行介绍。

#### 1.4.4 规则方法的优缺点

在基于规则的机器翻译时代，机器翻译技术研究有一个特点就是语法（Grammar）和算法（Algorithm）分开，相当于是把语言分析和程序设计分开。传统方式使用程序代码来实现翻译规则，并把所谓的翻译规则隐含在程序代码实现中。其中最大问题是一旦翻译规则发生修改，程序代码也需要进行相应修改，导致维护代价非常高。此外书写翻译规则的语言学家与编代码的程序员沟通代价也非常高，有时候会出现鸡同鸭讲的感觉。把语法和算法分开对于基于规则的机器翻译技术来说最大好处就是可以将语言学家和程序员的工作分开，各自发挥自己的优势。

这种语言分析和程序设计分开的实现方式也使得基于人工书写翻译规则的机器翻译方法非常直观，语言学家可以很容易地将翻译知识利用规则的方法表达出来，并且不需要修改系统代码。例如：1991年，东北大学自然语言处理实验室王宝库教授提出的规则描述语言（CTRDL）<sup>[18]</sup>。以及1995年，同为东北大学自然语言处理实验室的姚天顺教授提出的词汇语义驱动算法<sup>[19]</sup>，都是在这种思想上对机器翻译方法的一种改进。此外，使用规则本身就具有一定的优势：

- 翻译规则的书写颗粒度具有很大的可伸缩性。
- 较大颗粒度的翻译规则有很强的概括能力，较小颗粒度的翻译规则具有精细的描述能力。
- 翻译规则便于处理复杂的句法结构和进行深层次的语义理解，比如解决翻译过程中的长距离依赖问题。

通过图1.8中规则的翻译实例中可以看出，规则的使用和人类进行翻译时所使用的思想非常类似，可以说基于规则的方法实际上在试图描述人类进行翻译的思维过程。虽然直接模仿人类的翻译方式对翻译问题建模是合理的，但是这一定程度上也暴露了基于规则的方法的弱点。基于规则的机器翻译方法中，人工书写翻译规则的主观因素重，有时与客观事实有一定差距。并且人工书写翻译规则的难度大，代价非常高，这也成为了后来基于数据驱动的机器翻译方法主要改进的方向。

### 1.5 数据驱动的方法

虽然基于规则的方法有种种优势，但是该方法人工代价过高。所以研究者们开始尝试，是否可以更好地利用数据，从数据中学习到某些规律，而不是完全依靠人类来制定规则。在这样的思想下，基于数据驱动的方法诞生了。

#### 1.5.1 基于实例的机器翻译

在实际使用上，1.4章提到的基于规则的方法更多地被使用在受限翻译场景中，比如受限词汇集的翻译。针对基于规则的方法存在的问题，基于实例的机器翻译于上世纪80年代中期被提出<sup>[11]</sup>。该方法的基本思想是在双语句库中找到与待翻译句子

相似的实例，之后对实例的译文进行修改，如对译文进行替换、增加、删除等一系列操作，从而得到最终译文。这个过程可以类比人类学习并运用语言的过程：人会先学习一些翻译实例或者模板，当遇到新的句子时，会用以前的实例和模板作对比，之后得到新的句子的翻译结果。这也是一种举一反三的思想。

图1.13展示了基于实例的机器翻译过程。它利用简单的翻译实例库与翻译词典完成对句子“我对你感到满意”的翻译。首先，使用待翻译句子的源语言端在翻译实例库中进行比较，根据相似度大小找到相似的实例“我对他感到高兴”。然后，标记实例中不匹配的部分，即“你”和“他”，“满意”和“高兴”。再查询翻译词典得到词“你”和“满意”所对应的翻译结果“you”和“satisfied”，用这两个词分别替换实例中的“him”和“happy”，从而得到最终译文。

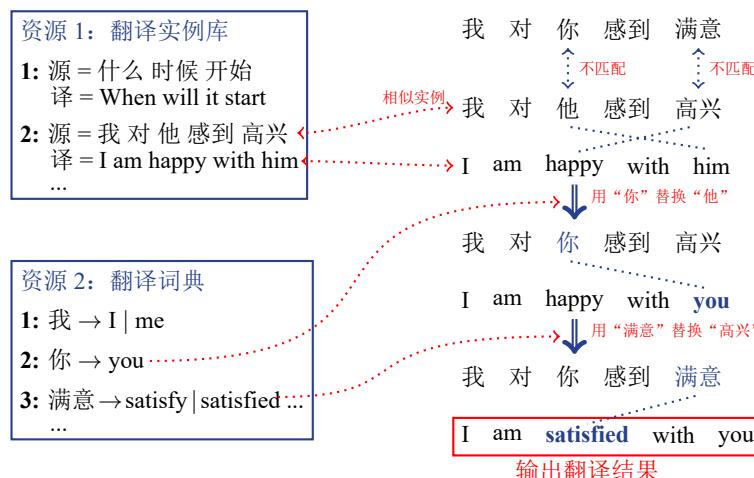


图 1.13 基于实例的机器翻译的示例图（左：实例库；右：翻译结果）

当然，基于实例的机器翻译也并不完美：

- 这种方法对翻译实例的精确度要求非常高，一个实例的错误可能会导致一个句型都无法翻译正确。
- 实例维护较为困难，实例库的构建通常需要单词级对齐的标注，而保证词对齐的质量是非常困难的工作，这也大大增加了实例库维护的难度。
- 尽管可以通过实例或者模板进行翻译，但是其覆盖度仍然有限。在实际应用中，很多句子无法找到可以匹配的实例或者模板。

## 1.5.2 统计机器翻译

统计机器翻译兴起于上世纪 90 年代<sup>[9, 20]</sup>，它利用统计模型从单/双语语料中自动学习翻译知识。具体来说，可以使用单语语料学习语言模型，使用双语平行语料学习翻译模型，并使用这些统计模型完成对翻译过程的建模。整个过程不需要人工编

写规则，也不需要从实例中构建翻译模板。无论是词还是短语，甚至是句法结构，统计机器翻译系统都可以自动学习。人更多的是定义翻译所需的特征和基本翻译单元的形式，而翻译知识都保存在模型的参数中。

图1.14展示了统计机器翻译系统的运行示例。整个系统需要两个模型：翻译模型和语言模型。其中，翻译模型从双语平行语料中学习翻译知识，得到短语表，短语表包含了各种词汇的翻译及其概率，这样可以度量源语言和目标语言片段之间互为翻译的可能性大小；语言模型从单语语料中学习目标语的词序列生成规律，来衡量目标语言译文的流畅性。最后，将这两种模型联合使用，通过翻译引擎来搜索尽可能多的翻译结果，并计算不同翻译结果的可能性大小，最后将概率最大的译文作为最终结果输出。这个过程并没有显性地使用人工翻译规则和模板，译文的生成仅仅依赖翻译模型和语言模型中的统计参数。

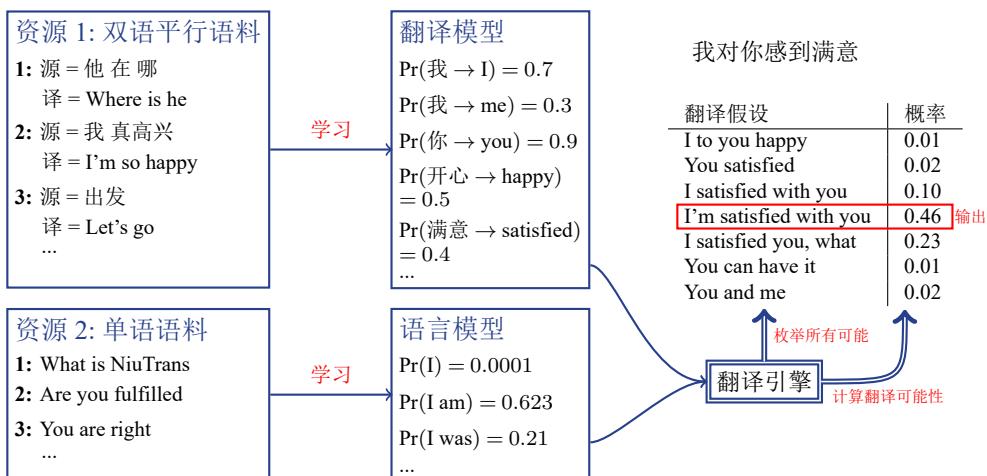


图 1.14 统计机器翻译的示例图（左：语料资源；中：翻译模型与语言模型；右：翻译假设与翻译引擎）

由于没有对翻译过程进行过多的限制，统计机器翻译有很灵活的译文生成方式，因此系统可以处理更加多样的句子。但是这种方法也带来了一些问题：首先，虽然并不需要人工定义翻译规则或模板，但统计机器翻译系统仍然需要人工定义翻译特征。提升翻译品质往往需要大量的特征工程，这导致人工特征设计的好坏会对系统产生决定性影响；其次，统计机器翻译的模块较多，系统研发比较复杂；再次，随着训练数据增多，统计机器翻译的模型（比如短语翻译表）会明显增大，对系统存储资源消耗较大。

### 1.5.3 神经机器翻译

随着机器学习技术的发展，基于深度学习的神经机器翻译逐渐兴起。自 2014 年开始，它在短短几年内已经在大部分任务上取得了明显的优势<sup>[21, 22, 23, 24, 25]</sup>。在神经机器翻译中，词串被表示成实数向量，即分布式向量表示。这样，翻译过程并不是在离散化的单词和短语上进行，而是在实数向量空间上计算。因此与之前的技术相比，

它在词序列表示的方式上有着本质的改变。通常，机器翻译可以被看作一个序列到另一个序列的转化。在神经机器翻译中，序列到序列的转化过程可以由**编码器-解码器**（Encoder-Decoder）框架实现。其中，编码器把源语言序列进行编码，并提取源语言中的信息进行分布式表示，之后解码器再把这种信息转换为另一种语言的表达。

图1.15展示了一个神经机器翻译的实例。首先，通过编码器，源语言序列“我对  
你感到满意”经过多层神经网络编码生成一个向量表示，即图中的向量（0.2, -1, 6,  
5, 0.7, -2）。再将该向量作为输入送到解码器中，解码器把这个向量解码成目标语  
言序列。注意，目标语言序列的生成是逐词进行的（虽然图中展示的是解码器一次  
生成了整个序列，但是在具体实现时是由左至右逐个单词地生成目标语译文），产生  
某个词的时候依赖之前生成的目标语言的历史信息，直到产生句子结束符为止。

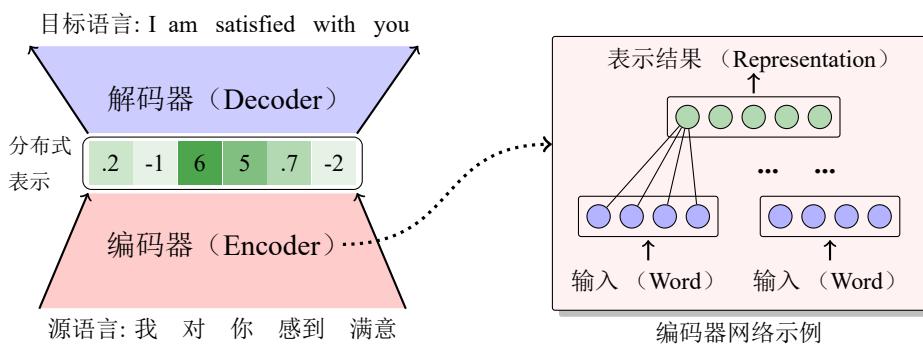


图 1.15 神经机器翻译的示例图（左：编码器-解码器网络；右：编码器示例网络）

与统计机器翻译相比，神经机器翻译的优势体现在其不需要特征工程，所有信息由神经网络自动从原始输入中提取。而且，相比于统计机器翻译中所使用的离散化的表示。神经机器翻译中词和句子的分布式连续空间表示可以为建模提供更为丰富的信息，同时可以使用相对成熟的基于梯度的方法优化模型。此外，神经网络的存储需求较小，天然适合小设备上的应用。当然，神经机器翻译也存在问题：

- 虽然脱离了特征工程，但神经网络的结构需要人工设计，即使设计好结构，系统的调优、超参数的设置等仍然依赖大量的实验。
- 神经机器翻译现在缺乏可解释性，其过程和人的认知差异很大，通过人的先验知识干预的程度差。
- 神经机器翻译对数据的依赖很大，数据规模、质量对性能都有很大影响，特别是在数据稀缺的情况下，充分训练神经网络很有挑战性。

#### 1.5.4 对比分析

不同机器翻译方法有不同的特点。表1.1对比了这些方法，不难看出：

- 规则系统需要人工编写规则并维护，人工代价较高。统计和神经网络方法仅需

要设计特征或者神经网络结构，对人工依赖较少（语言相关的）。

- 基于实例、统计和神经网络的方法都需要依赖语料库（数据），其中统计和神经网络方法具有一定的抗噪能力，因此也更适合大规模数据情况下的机器翻译系统研发。
- 基于规则和基于实例的方法在受限场景下有较好的精度，但是在开放领域的翻译上统计和神经网络方法更具优势。

表 1.1 不同机器翻译方法的对比

	规则	实例	统计	神经
人工写规则	是	否	否	否
人工代价	高	一般	几乎没有	几乎没有
数据驱动	否	是	是	是
依赖数据质量	N/A	高	低	较低
抗噪声能力	低	低	高	较高
使用范围	受限领域	受限领域	通用领域	通用领域
翻译精度	高	较高	不确定	不确定

从现在机器翻译的研究和应用情况来看，基于统计建模的方法（统计机器翻译和神经机器翻译）是主流。这主要是由于它们的系统研发周期短，通过搜集一定量的数据即可实现快速原型。但是随着互联网等信息的不断开放，低成本的数据获取让神经机器翻译系统更快得以实现。因此最近神经机器翻译凭借其高质量的译文，受到越来越多研究人员和开发者的青睐。当然，对不同方法进行融合也是有价值的研究方向，也有很多有趣的探索，比如无指导机器翻译中会同时使用统计机器翻译和神经机器翻译方法，这也是一种典型的融合多种方法的思路。

## 1.6 推荐学习资源

### 1.6.1 经典书籍

首先，推荐一本书 *Statistical Machine Translation*<sup>[26]</sup>，其作者是机器翻译领域著名学者 Philipp Koehn 教授。该书是机器翻译领域内的经典之作，介绍了统计机器翻译技术的进展。该书从语言学和概率学两个方面介绍了统计机器翻译的构成要素，然后介绍了统计机器翻译的主要模型：基于词、基于短语和基于树的模型，以及机器翻译评价、语言建模、判别式训练等方法。此外，作者在该书的最新版本中增加了神经机器翻译的章节，方便研究人员全面了解机器翻译的最新发展趋势<sup>[27]</sup>。

*Foundations of Statistical Natural Language Processing*<sup>[28]</sup> 中文译名《统计自然语言处理基础》，作者是自然语言处理领域的权威 Chris Manning 教授和 Hinrich Schütze 教授。该书对统计自然语言处理方法进行了全面介绍。书中讲解了统计自然语言处

理所需的语言学和概率论基础知识，介绍了机器翻译评价、语言建模、判别式训练以及整合语言学信息等基础方法。其中也包含了构建自然语言处理工具所需的基本理论和算法，并且涵盖了数学和语言学基础内容以及相关的统计方法。

《统计自然语言处理（第2版）》<sup>[29]</sup> 由中国科学院自动化所宗成庆教授所著。该书中系统介绍了统计自然语言处理的基本概念、理论方法和最新研究进展，既有对基础知识和理论模型的介绍，也有对相关问题的研究背景、实现方法和技术现状的详细阐述。可供从事自然语言处理、机器翻译等研究的相关人员参考。

由 Ian Goodfellow、Yoshua Bengio、Aaron Courville 三位机器学习领域的学者所写的 *Deep Learning*<sup>[30]</sup> 也是值得一读的参考书。其讲解了有关深度学习常用的方法，其中很多都会在深度学习模型设计和使用中用到。同时在该书的应用一章中也简单讲解了神经机器翻译的任务定义和发展过程。

*Neural Network Methods for Natural Language Processing*<sup>[31]</sup> 是 Yoav Goldberg 编写的面向自然语言处理的深度学习参考书。相比 *Deep Learning*，该书聚焦在自然语言处理中的深度学习方法，内容更加易读，非常适合刚入门自然语言处理及深度学习应用的人员参考。

《机器学习》<sup>[32]</sup> 由南京大学周志华教授所著，作为机器学习领域入门教材，该书尽可能地涵盖了机器学习基础知识的各个方面，试图尽可能少地使用数学知识介绍机器学习方法与思想。

《统计学习方法（第2版）》<sup>[33]</sup> 由李航博士所著，该书对机器学习的有监督和无监督等方法进行了全面而系统的介绍。可以作为梳理机器学习的知识体系，同时了解相关基础概念的参考读物。

《神经网络与深度学习》<sup>[34]</sup> 由复旦大学邱锡鹏教授所著，全面地介绍了神经网络和深度学习的基本概念和常用技术，同时涉及了许多深度学习的前沿方法。该书适合初学者阅读，同时又不失为一本面向专业人士的参考书。

## 1.6.2 相关学术会议

许多自然语言处理的相关学术组织会定期举办学术会议。以**计算语言学**（Computational Linguistics）和**自然语言处理**（Natural Language Processing）方面的会议为主。与机器翻译相关的部分会议有：

- AACL，全称 Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics，为国际权威组织计算语言学会（Association for Computational Linguistics，ACL）亚太地区分会。2020 年会议首次召开，是亚洲地区自然语言处理领域最具影响力的会议之一。
- AAMT，全称 Asia-Pacific Association for Machine Translation Annual Conference，为亚洲-太平洋地区机器翻译协会举办的年会，旨在推进亚洲及泛太平洋地区机器翻译的研究和产业化。特别是对亚洲国家语言的机器翻译研究有很好的促进，因此也成为了该地区十分受关注的会议之一。

- ACL，全称 Annual Conference of the Association for Computational Linguistics，是自然语言处理领域最高级别的会议。由计算语言学会组织，每年举办一次，主题涵盖计算语言学的所有方向。
- AMTA，全称 Biennial Conference of the Association for Machine Translation in the Americas，美国机器翻译协会组织的会议，每两年举办一次。AMTA 会议汇聚了学术界、产业界和政府的研究人员、开发人员和用户，让工业界和学术界进行交流。
- CCL，全称 China National Conference on Computational Linguistics，中文为中国计算语言学大会。中国计算语言学大会创办于 1991 年，由中国中文信息学会计算语言学专业委员会负责组织。经过 20 余年的发展，中国计算语言学大会已成为国内自然语言处理领域权威性最高、规模和影响最大的学术会议。作为中国中文信息学会（国内一级学会）的旗舰会议，CCL 聚焦于中国境内各类语言的智能计算和信息处理，为研讨和传播计算语言学最新学术和技术成果提供了最广泛的高层次交流平台。
- CCMT，全称 China Conference on Machine Translation，中国机器翻译研讨会，由中国中文信息学会主办，旨在为国内外机器翻译界同行提供一个平台，促进中国机器翻译事业。CCMT 不仅是国内机器翻译领域最具影响力、最权威的学术和评测活动，而且也代表着汉语与民族语言翻译技术的最高水准，对民族语言技术发展具有重要意义。
- COLING，全称 International Conference on Computational Linguistics，自然语言处理老牌顶级会议之一。该会议始于 1965 年，是由 ICCL 国际计算语言学委员会主办。会议简称为 COLING，是谐音瑞典著名作家 Albert Engström 小说中的虚构人物 Kolingen。COLING 每两年举办一次。
- EACL，全称 Conference of the European Chapter of the Association for Computational Linguistics，为 ACL 欧洲分会，虽然在欧洲召开，会议也吸引了全世界的大量学者投稿并参会。
- EAMT，全称 Annual Conference of the European Association for Machine Translation，欧洲机器翻译协会的年会。该会议汇聚了欧洲机器翻译研究、产业化等方面的成果，同时也吸引了世界范围的关注。
- EMNLP，全称 Conference on Empirical Methods in Natural Language Processing，自然语言处理另一个顶级会议之一，由 ACL 当中对语言数据和经验方法有特殊兴趣的团体主办，始于 1996 年。会议比较偏重于方法和经验性结果。
- MT Summit，全称 Machine Translation Summit，是机器翻译领域的重要峰会。该会议的特色是与产业结合，在探讨机器翻译技术问题的同时，更多的关注机器翻译的应用落地工作，因此备受产业界关注。该会议每两年举办一次，通常由欧洲

机器翻译协会 (The European Association for Machine Translation, EAMT)、美国机器翻译协会 (The Association for Machine Translation in the Americas, AMTA)、亚洲-太平洋地区机器翻译协会 (Asia-Pacific Association for Machine Translation, AAMT) 举办。

- NAACL, 全称 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 为 ACL 北美分会, 在自然语言处理领域也属于顶级会议, 每年会选择一个北美城市召开会议。
- NLPCC, 全称 CCF International Conference on Natural Language Processing and Chinese Computing。NLPCC 是由中国计算机学会 (CCF) 主办的 CCF 中文信息技术专业委员会年度学术会议, 专注于自然语言处理及中文处理领域的研究和应用创新。会议自 2012 年开始举办, 主要活动有主题演讲、论文报告、技术测评等多种形式。
- WMT, 全称 Conference on Machine Translation, 前身为 Workshop on Statistical Machine Translation。机器翻译领域一年一度的国际会议。其举办的机器翻译评测是国际公认的顶级机器翻译赛事之一。

除了会议之外,《中文信息学报》、*Computational Linguistics*、*Machine Translation*、*Transactions of the Association for Computational Linguistics*、*IEEE/ACM Transactions on Audio, Speech, and Language Processing*、*ACM Transactions on Asian and Low Resource Language Information Processing*、*Natural Language Engineering* 等期刊也发表了许多与机器翻译相关的重要论文。



## 2. 统计语言建模基础

世间万物的运行都是不确定的，大到宇宙的运转，小到分子的运动，都是如此。自然语言也同样充满着不确定性和灵活性。建立统计模型正是描述这种不确定性的一种手段，包括机器翻译在内对众多自然语言处理问题的求解都大量依赖于这些统计模型。

本章将会对统计建模的基础数学工具进行介绍，并在此基础上对语言建模问题展开讨论。而统计建模与语言建模任务的结合也产生了自然语言处理的一个重要方向——**统计语言建模**（Statistical Language Modeling）。它与机器翻译有很多相似之处，比如，二者都在描述单词串生成的过程，因此在解决问题的思想上是相通的。此外，统计语言模型也常常被作为机器翻译系统的组件，对于机器翻译系统研发有着重要意义。本章所讨论的内容对本书后续章节有很好的铺垫作用。本书也会大量运用统计模型的手段对自然语言处理问题进行描述。

### 2.1 概率论基础

为了便于后续内容的介绍，首先对本书中使用的概率和统计学概念进行简要说明。

#### 2.1.1 随机变量和概率

在自然界中，很多**事件**（Event）是否会发生是不确定的。例如，明天会下雨、掷一枚硬币是正面朝上、扔一个骰子的点数是 1 等。这些事件可能会发生也可能不会

发生。通过大量的重复试验，能发现具有某种规律性的事件叫做**随机事件**。

**随机变量** (Random Variable) 是对随机事件发生可能状态的描述，是随机事件的数量表征。设  $\Omega = \{\omega\}$  为一个随机试验的样本空间， $X = X(\omega)$  就是定义在样本空间  $\Omega$  上的单值实数函数，即  $X = X(\omega)$  为随机变量，记为  $X$ 。随机变量是一种能随机选取数值的变量，常用大写的英语字母或希腊字母表示，其取值通常用小写字母来表示。例如，用  $A$  表示一个随机变量，用  $a$  表示变量  $A$  的一个取值。根据随机变量可以选取的值的某些性质，可以将其划分为离散变量和连续变量。

离散变量是在其取值区间内可以被一一列举、总数有限并且可计算的数值变量。例如，用随机变量  $X$  代表某次投骰子出现的点数，点数只可能取 1~6 这 6 个整数， $X$  就是一个离散变量。

连续变量是在其取值区间内连续取值无法被一一列举、具有无限个取值的变量。例如，图书馆的开馆时间是 8:30-22:00，用  $X$  代表某人进入图书馆的时间，时间的取值范围是 [8:30, 22:00] 这个时间区间， $X$  就是一个连续变量。

**概率** (Probability) 是度量随机事件呈现其每个可能状态的可能性的数值，本质上它是一个测度函数<sup>[35, 36]</sup>。概率的大小表征了随机事件在一次试验中发生的可能性大小。用  $P(\cdot)$  表示一个随机事件的可能性，即事件发生的概率。比如  $P(\text{太阳从东方升起})$  表示“太阳从东方升起”的可能性，同理， $P(A = B)$  表示的就是“ $A = B$ ”这件事的可能性。

在实际问题中，往往需要得到随机变量的概率值。但是，真实的概率值可能是无法准确知道的，这时就需要对概率进行**估计** (Estimation)，得到的结果是概率的**估计值** (Estimate)。概率值的估计是概率论和统计学中的经典问题，有十分多样的方法可以选择。比如，一个很简单的方法是利用相对频次作为概率的估计值。如果  $\{x_1, x_2, \dots, x_n\}$  是一个试验的样本空间，在相同情况下重复试验  $N$  次，观察到样本  $x_i (1 \leq i \leq n)$  的次数为  $n(x_i)$ ，那么  $x_i$  在这  $N$  次试验中的相对频率是  $\frac{n(x_i)}{N}$ 。当  $N$  越来越大时，相对概率也就越来越接近真实概率  $P(x_i)$ ，即  $\lim_{N \rightarrow \infty} \frac{n(x_i)}{N} = P(x_i)$ 。实际上，很多概率模型都等同于相对频次估计。比如，对于一个服从多项式分布的变量，它的极大似然估计就可以用相对频次估计实现。

概率函数是用函数形式给出离散变量每个取值发生的概率，其实就是将变量的概率分布转化为数学表达形式。如果把  $A$  看做一个离散变量， $a$  看做变量  $A$  的一个取值，那么  $P(A)$  被称作变量  $A$  的概率函数， $P(A = a)$  被称作  $A = a$  的概率值，简记为  $P(a)$ 。例如，在相同条件下掷一个骰子 50 次，用  $A$  表示投骰子出现的点数这个离散变量， $a_i$  表示点数的取值， $P_i$  表示  $A = a_i$  的概率值。表2.1为  $A$  的概率分布，给出了  $A$  的所有取值及其概率。

表 2.1 离散变量  $A$  的概率分布

$A$	$a_1 = 1$	$a_2 = 2$	$a_3 = 3$	$a_4 = 4$	$a_5 = 5$	$a_6 = 6$
$P_i$	$P_1 = \frac{4}{25}$	$P_2 = \frac{3}{25}$	$P_3 = \frac{4}{25}$	$P_4 = \frac{6}{25}$	$P_5 = \frac{3}{25}$	$P_6 = \frac{5}{25}$

除此之外，概率函数  $P(\cdot)$  还具有非负性、归一性等特点。非负性是指，所有的概率函数  $P(\cdot)$  的数值都必须大于等于 0，概率函数中不可能出现负数，即  $\forall x, P(x) \geq 0$ 。归一性，又称规范性，简单来说就是所有可能发生的事件的概率总和为 1，即  $\sum_x P(x) = 1$ 。

对于离散变量  $A$ ， $P(A = a)$  是个确定的值，可以表示事件  $A = a$  的可能性大小；而对于连续变量，求在某个定点处的概率是无意义的，只能求其落在某个取值区间内的概率。因此，用 **概率分布函数**  $F(x)$  和 **概率密度函数**  $f(x)$  来统一描述随机变量取值的分布情况（如图2.1）。概率分布函数  $F(x)$  表示取值小于等于某个值的概率，是概率的累加（或积分）形式。假设  $A$  是一个随机变量， $a$  是任意实数，将函数  $F(a) = P\{A \leq a\}$  定义为  $A$  的分布函数。通过分布函数，可以清晰地表示任何随机变量的概率分布情况。

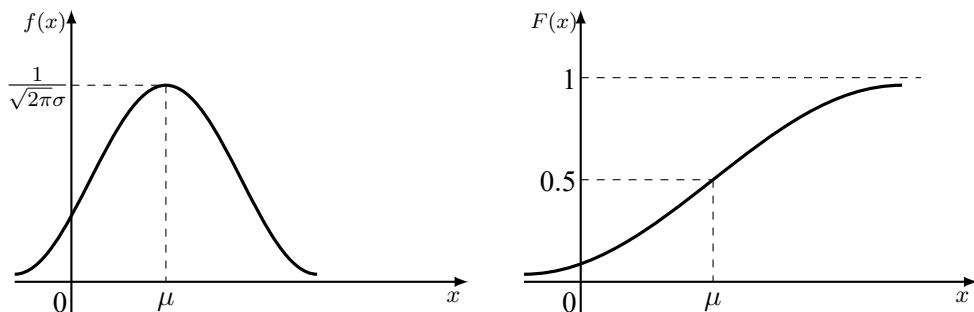


图 2.1 一个概率密度函数（左）与其对应的分布函数（右）

概率密度函数反映了变量在某个区间内的概率变化快慢，概率密度函数的值是概率的变化率，该连续变量的概率分布函数也就是对概率密度函数求积分得到的结果。设  $f(x) \geq 0$  是连续变量  $X$  的概率密度函数， $X$  的分布函数就可以用如下公式定义：

$$F(x) = \int_{-\infty}^x f(x)dx \quad (2.1)$$

## 2.1.2 联合概率、条件概率和边缘概率

**联合概率** (Joint Probability) 是指多个事件共同发生，每个随机变量满足各自条件的概率。如事件  $A$  和事件  $B$  的联合概率可以表示为  $P(AB)$  或  $P(A \cap B)$ 。**条件概率** (Conditional Probability) 是指  $A$ 、 $B$  为任意的两个事件，在事件  $A$  已出现的前提下，事件  $B$  出现的概率，使用  $P(B | A)$  表示。

贝叶斯法则（见2.1.4小节）是条件概率计算时的重要依据，条件概率可以表示

为:

$$\begin{aligned}
 P(B|A) &= \frac{P(A \cap B)}{P(A)} \\
 &= \frac{P(A)P(B|A)}{P(A)} \\
 &= \frac{P(B)P(A|B)}{P(A)}
 \end{aligned} \tag{2.2}$$

**边缘概率** (Marginal Probability) 是和联合概率对应的, 它指的是  $P(X = a)$  或  $P(Y = b)$ , 即仅与单个随机变量有关的概率。对于离散随机变量  $X$  和  $Y$ , 如果知道  $P(X, Y)$ , 则边缘概率  $P(X)$  可以通过求和的方式得到。对于  $\forall x \in X$ , 有:

$$P(X = x) = \sum_y P(X = x, Y = y) \tag{2.3}$$

对于连续变量, 边缘概率  $P(X)$  需要通过积分得到, 如下式所示:

$$P(X = x) = \int P(x, y) dy \tag{2.4}$$

为了更好地区分条件概率、边缘概率和联合概率, 这里用一个图形面积的计算来举例说明。如图2.2所示, 矩形  $A$  代表事件  $X$  发生所对应的所有可能状态, 矩形  $B$  代表事件  $Y$  发生所对应的所有可能状态, 矩形  $C$  代表  $A$  和  $B$  的交集, 则:

- **边缘概率:** 矩形  $A$  或者矩形  $B$  的面积。
- **联合概率:** 矩形  $C$  的面积。
- **条件概率:** 联合概率/对应的边缘概率, 如:  $P(A | B) =$  矩形  $C$  的面积/矩形  $B$  的面积。

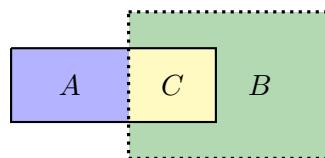


图 2.2  $A$ 、 $B$ 、 $C$  事件所对应概率的图形化表示

### 2.1.3 链式法则

条件概率公式  $P(A | B) = P(AB)/P(B)$  反映了事件  $B$  发生的条件下事件  $A$  发生的概率。如果将其推广到三个事件  $A$ 、 $B$ 、 $C$ , 为了计算  $P(A, B, C)$ , 可以运用两

次  $P(A | B) = P(AB)/P(B)$ , 计算过程如下:

$$\begin{aligned} P(A, B, C) &= P(A | B, C)P(B, C) \\ &= P(A | B, C)P(B | C)P(C) \end{aligned} \quad (2.5)$$

推广到  $n$  个事件, 可以得到**链式法则** (Chain Rule) 的公式:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | x_1, \dots, x_{i-1}) \quad (2.6)$$

链式法则经常被用于对事件序列的建模。比如, 在事件  $A$  与事件  $C$  相互独立时, 事件  $A$ 、 $B$ 、 $C$  的联合概率可以被表示为:

$$\begin{aligned} P(A, B, C) &= P(A)P(B | A)P(C | A, B) \\ &= P(A)P(B | A)P(C | B) \end{aligned} \quad (2.7)$$

## 2.1.4 贝叶斯法则

首先介绍一下全概率公式: **全概率公式** (Law of Total Probability) 是概率论中重要的公式, 它可以将一个复杂事件发生的概率分解成不同情况的小事件发生概率的和。这里先介绍一个概念——划分。集合  $\Sigma$  的一个划分事件为  $\{B_1, \dots, B_n\}$  是指它们满足  $\bigcup_{i=1}^n B_i = S$  且  $B_i B_j = \emptyset, i, j = 1, \dots, n, i \neq j$ 。此时事件  $A$  的全概率公式可以被描述为:

$$P(A) = \sum_{k=1}^n P(A | B_k)P(B_k) \quad (2.8)$$

举个例子, 小张从家到公司有三条路分别为  $a$ ,  $b$ ,  $c$ , 选择每条路的概率分别为 0.5, 0.3, 0.2。令:

- $S_a$ : 小张选择  $a$  路去上班
- $S_b$ : 小张选择  $b$  路去上班
- $S_c$ : 小张选择  $c$  路去上班
- $S$ : 小张去上班

显然,  $S_a$ ,  $S_b$ ,  $S_c$  是  $S$  的划分。如果三条路不拥堵的概率分别为  $P(S'_a)=0.2$ ,

$P(S'_b)=0.4$ ,  $P(S'_c)=0.7$ , 那么事件  $L$ : 小张上班没有遇到拥堵情况的概率就是:

$$\begin{aligned} P(L) &= P(L|S_a)P(S_a) + P(L|S_b)P(S_b) + P(L|S_c)P(S_c) \\ &= P(S'_a)P(S_a) + P(S'_b)P(S_b) + P(S'_c)P(S_c) \\ &= 0.36 \end{aligned} \quad (2.9)$$

**贝叶斯法则** (Bayes' Rule) 是概率论中的一个经典公式, 通常用于已知  $P(A | B)$  求  $P(B | A)$ 。可以表述为: 设  $\{B_1, \dots, B_n\}$  是某个集合  $\Sigma$  的一个划分,  $A$  为事件, 则对于  $i = 1, \dots, n$ , 有如下公式:

$$\begin{aligned} P(B_i | A) &= \frac{P(AB_i)}{P(A)} \\ &= \frac{P(A | B_i)P(B_i)}{\sum_{k=1}^n P(A | B_k)P(B_k)} \end{aligned} \quad (2.10)$$

其中, 等式右端的分母部分使用了全概率公式。进一步, 令  $\bar{B}$  表示事件  $B$  不发生的情况, 由上式, 也可以得到贝叶斯公式的另外一种写法:

$$\begin{aligned} P(B | A) &= \frac{P(A | B)P(B)}{P(A)} \\ &= \frac{P(A | B)P(B)}{P(A | B)P(B) + P(A | \bar{B})P(\bar{B})} \end{aligned} \quad (2.11)$$

贝叶斯公式常用于根据已知的结果来推断使之发生的各因素的可能性。

## 2.1.5 KL 距离和熵

### 1. 信息熵

熵是热力学中的一个概念, 同时也是对系统无序性的一种度量标准。在自然语言处理领域也会使用到信息熵这一概念, 比如描述文字的信息量大小。一条信息的信息量可以被看作是这条信息的不确定性。如果需要确认一件非常不确定甚至于一无所知的事情, 那么需要理解大量的相关信息才能进行确认; 同样的, 如果对某件事已经非常确定, 那么就不需要太多的信息就可以把它搞清楚。如下就是两个例子,

#### 实例 2.1 确定性和不确定性的事件

“太阳从东方升起”

“明天天气多云”

在这两句话中, “太阳从东方升起”是一件确定性事件 (在地球上), 几乎不需要查阅更多信息就可以确认, 因此这件事的信息熵相对较低; 而“明天天气多云”这件事, 需要关注天气预报, 才能大概率确定这件事, 它的不确定性很高, 因而它的信息熵也就相对较高。因此, 信息熵也是对事件不确定性的度量。进一步, 一个事件  $X$

的**自信息**（Self-information）的表达式为：

$$I(x) = -\log P(x) \quad (2.12)$$

其中， $x$  是  $X$  的一个取值， $P(x)$  表示  $x$  发生的概率。自信息用来衡量单一事件发生时所包含的信息多少，当底数为  $e$  时，单位为 nats，其中 1nats 是通过观察概率为  $1/e$  的事件而获得的信息量；当底数为 2 时，单位为 bits 或 shannons。 $I(x)$  和  $P(x)$  的函数关系如图2.3 所示。

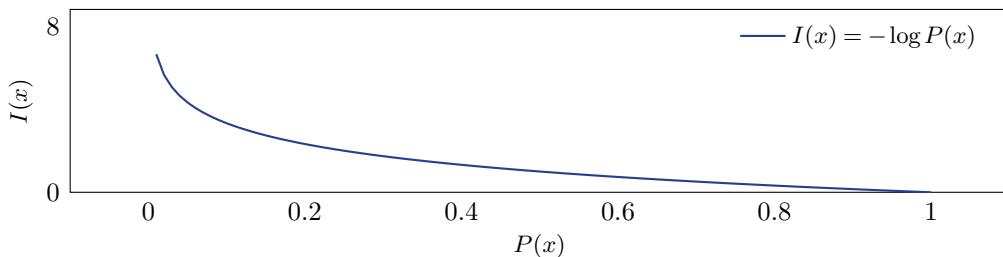


图 2.3 自信息函数  $I(x)$  关于  $P(x)$  的曲线

自信息处理的是变量单一取值的情况。若量化整个概率分布中的不确定性或信息量，可以用信息熵，记为  $H(x)$ 。其公式如下：

$$\begin{aligned} H(x) &= \sum_{x \in X} [P(x)I(x)] \\ &= -\sum_{x \in X} [P(x)\log(P(x))] \end{aligned} \quad (2.13)$$

一个分布的信息熵也就是从该分布中得到的一个事件的期望信息量。比如， $a$ 、 $b$ 、 $c$ 、 $d$  四支球队，四支队伍夺冠的概率分别是  $P_1$ 、 $P_2$ 、 $P_3$ 、 $P_4$ ，某个人对比赛不感兴趣但是又想知道哪只球队夺冠，使用 2 次二分法就能确定哪支球队夺冠了。但假设这四只球队中  $c$  的实力可以碾压其他球队，那么猜 1 次就可以确定。所以对于前面这种情况，哪只球队夺冠的信息量较高，信息熵也相对较高；对于后面这种情况，因为结果是容易猜到的，信息量和信息熵也就相对较低。因此可以得知：分布越尖锐熵越低，分布越均匀熵越高。

## 2. KL 距离

如果同一个随机变量  $X$  上有两个概率分布  $P(x)$  和  $Q(x)$ ，那么可以使用 **Kullback-Leibler 距离**或 **KL 距离**（KL Distance）来衡量这两个分布的不同（也称作 KL 散度）。

这种度量就是**相对熵** (Relative Entropy) , 其公式如下:

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in X} [P(x) \log \frac{P(x)}{Q(x)}] \\ &= \sum_{x \in X} [P(x)(\log P(x) - \log Q(x))] \end{aligned} \quad (2.14)$$

其中, 概率分布  $P(x)$  对应的每个事件的可能性。相对熵的意义在于: 在一个事件空间里, 若用概率分布  $Q(x)$  来编码  $P(x)$ , 相比于用概率分布  $P(x)$  来编码  $P(x)$  时信息量增加了多少。它衡量的是同一个事件空间里两个概率分布的差异。**KL** 距离有两条重要的性质:

- **非负性**, 即  $D_{\text{KL}}(P \parallel Q) \geq 0$ , 等号成立条件是  $P$  和  $Q$  相等。
- **不对称性**, 即  $D_{\text{KL}}(P \parallel Q) \neq D_{\text{KL}}(Q \parallel P)$ , 所以 **KL** 距离并不是常用的欧式空间中的距离。为了消除这种不确定性, 有时也会使用  $D_{\text{KL}}(P \parallel Q) + D_{\text{KL}}(Q \parallel P)$  作为度量两个分布差异性的函数。

### 3. 交叉熵

**交叉熵** (Cross-entropy) 是一个与 **KL** 距离密切相关的概念, 它的公式是:

$$H(P, Q) = - \sum_{x \in X} [P(x) \log Q(x)] \quad (2.15)$$

结合相对熵公式可知, 交叉熵是 **KL** 距离公式中的右半部分。因此, 当概率分布  $P(x)$  固定时, 求关于  $Q$  的交叉熵的最小值等价于求 **KL** 距离的最小值。从实践的角度来说, 交叉熵与 **KL** 距离的目的相同: 都是用来描述两个分布的差异。由于交叉熵计算上更加直观方便, 因此在机器翻译中被广泛应用。

## 2.2 掷骰子游戏

在阐述统计建模方法前, 先看一个有趣的实例 (图2.4)。掷骰子, 一个生活中比较常见的游戏, 掷一个骰子, 玩家猜一个数字, 猜中就算赢。按照常识来说, 随便选一个数字, 获胜的概率是一样的, 即所有选择的获胜概率都是  $1/6$ 。因此这个游戏玩家很难获胜, 除非运气很好。假设进行一次游戏, 玩家随意选了一个数字, 比如是 1。当投掷 30 次骰子 (如图2.4), 发现运气不错, 命中 7 次, 好于预期 ( $7/30 > 1/6$ )。

此时玩家的胜利似乎只能来源于运气。不过, 这里的假设“随便选一个数字, 获胜的概率是一样的”本身就是一个概率模型, 它对骰子六个面的出现做了均匀分布假设:

$$P(1) = P(2) = \dots = P(5) = P(6) = 1/6 \quad (2.16)$$

2	3	1	4	4	1	5	1	4	4
5	6	4	4	3	2	1	4	5	1
4	2	2	3	4	1	5	1	3	4

图 2.4 骰子结果

但是在这个游戏中没有人规定骰子是均匀的。如果骰子的六个面不均匀呢？这里可以用一种更加“聪明”的方式定义一种新的模型，即定义骰子的每一个面都以一定的概率出现，而不是相同的概率。描述如下：

$$\begin{aligned}
 P(1) &= \theta_1 \\
 P(2) &= \theta_2 \\
 P(3) &= \theta_3 \\
 P(4) &= \theta_4 \\
 P(5) &= \theta_5 \\
 P(6) &= 1 - \sum_{1 \leq i \leq 5} \theta_i \quad \triangleleft \text{归一性}
 \end{aligned} \tag{2.17}$$

这里， $\theta_1 \sim \theta_5$  可以被看作是模型的参数，因此这个模型的自由度是 5。对于这样的模型，参数确定了，模型也就确定了。但是一个新的问题出现了，在定义骰子每个面的概率后，如何求出具体的概率值呢？一种常用的方法是，从大量实例中学习模型参数，这个方法也是常说的**参数估计**（Parameter Estimation）。可以将这个不均匀的骰子先实验性地掷很多次，这可以被看作是独立同分布的若干次采样。比如投掷骰子  $X$  次，发现 1 出现  $X_1$  次，2 出现  $X_2$  次，以此类推，可以得到各个面出现的次数。假设掷骰子中每个面出现的概率符合多项式分布，那么通过简单的概率论知识可以知道每个面出现概率的极大似然估计为：

$$P(i) = \frac{X_i}{X} \tag{2.18}$$

当  $X$  足够大时， $X_i/X$  可以无限逼近  $P(i)$  的真实值，因此可以通过大量的实验推算出掷骰子各个面的概率的准确估计值。

回归到原始的问题，如果在正式开始游戏前，预先掷骰子 30 次，得到如图 2.5 的结果。

此时，可以注意到，这是一个有倾向性的模型（图 2.6）：在这样的预先实验基础上，可以知道这个骰子是不均匀的，如果用这个骰子玩掷骰子游戏，选择数字 4 获胜的可能性是最大的。

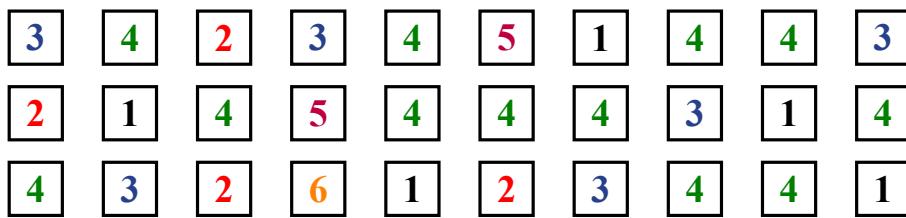


图 2.5 实验性投掷骰子的结果

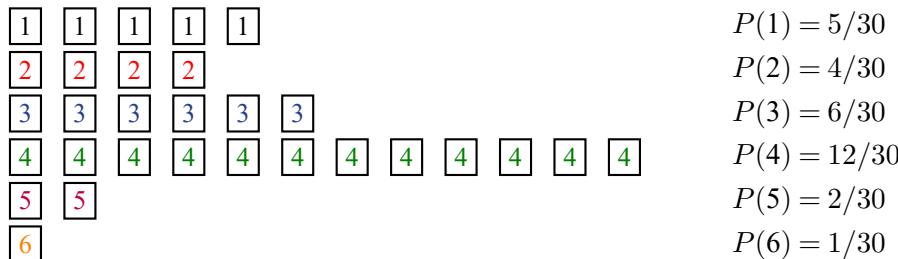


图 2.6 投骰子模型

与上面这个掷骰子游戏类似，世界上的事物并不是平等出现的。在“公平”的世界中，没有任何一个模型可以学到有价值的事情。从机器学习的角度来看，所谓的“不公平”实际上是客观事物中蕴含的一种**偏置**（Bias），也就是很多事情天然地就有对某些情况有倾向。而图像处理、自然语言处理等问题中绝大多数都存在着偏置。比如，当翻译一个英语单词的时候，它最可能的翻译结果往往就是那几个词。设计统计模型的目的正是要学习这种偏置，之后利用这种偏置对新的问题做出足够好的决策。

在处理自然语言问题时，为了评价哪些词更容易在一个句子中出现，或者哪些句子在某些语境下更合理，常常也会使用统计方法对词或句子出现的可能性建模。与掷骰子游戏类似，词出现的概率可以这样理解：每个单词的出现就好比掷一个巨大的骰子，与前面的例子中有所不同的是：

- 骰子有很多个面，每个面代表一个单词。
- 骰子是不均匀的，代表常用单词的那些面的出现次数会远远多于罕见单词。

如果投掷这个新的骰子，可能会得到图2.7这样的结果。如果把这些数字换成汉语中的词，比如：

88 = 这

87 = 是

45 = 一

...

就可以得到图2.8所示的结果。于是，可以假设有一个不均匀的多面骰子，每个

面都对应一个单词。在获取一些文本数据后，可以统计每个单词出现的次数，进而利用极大似然估计推算出每个单词在语料库中出现的概率的估计值。图2.9给出了一个实例

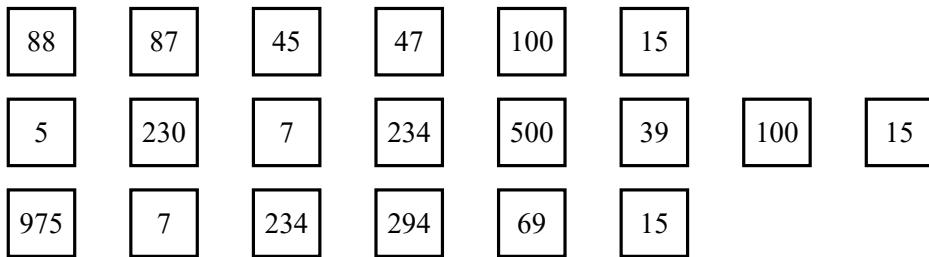


图 2.7 投掷一个很多面骰子的结果

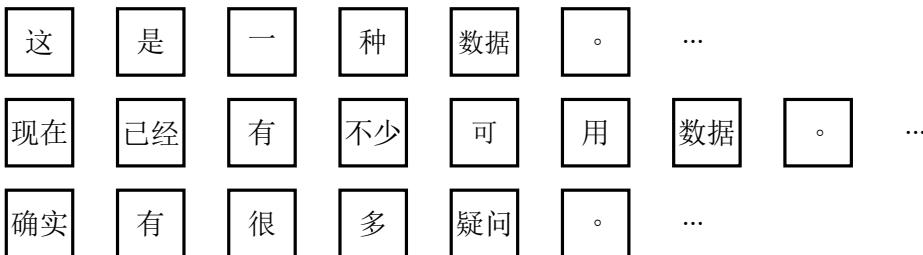


图 2.8 掷骰子游戏中把数字换成汉字后的结果

$$\text{总词数: } 6 + 8 + 5 = 20$$

$$P(\text{很}) = 1/20 = 0.05$$

$$P(\text{。}) = 3/20 = 0.15$$

$$P(\text{确实}) = 1/20 = 0.05$$

$$\text{更多数据-总词数: } 1 \text{ 百万个词}$$

$$P(\text{很}) = 0.000010$$

$$P(\text{。}) = 0.001812$$

$$P(\text{确实}) = 0.000001$$

图 2.9 单词概率的估计结果

通过这个学习过程，就可以得到每个词出现的概率，成功使用统计方法对“单词的频率”这个问题进行建模。那么该如何计算一个句子的概率呢？在自然语言处理领域中，句子可以被看作是由单词组成的序列，因而句子的概率可以被建模为若干单词的联合概率，即  $P(w_1 w_2 \dots w_m)$ 。其中， $w_i$  表示句子中的一个单词。

为了求  $P(w_1 w_2 \dots w_m)$ ，最直接的方式是统计所有可能出现的词串  $w_1 w_2 \dots w_m$  在数据中出现的次数  $c(w_1 w_2 \dots w_m)$ ，之后利用极大似然估计算  $P(w_1 w_2 \dots w_m)$ ：

$$P(w_1 w_2 \dots w_m) = \frac{c(w_1 w_2 \dots w_m)}{\sum_{w'_1, w'_2, \dots, w'_m \in V} c(w'_1 w'_2 \dots w'_m)} \quad (2.19)$$

其中， $V$  为词汇表。本质上，这个方法和计算单词出现概率  $P(w_i)$  的方法是一样的。但是这里的问题是：当  $m$  较大时，词串  $w_1 w_2 \dots w_m$  可能非常低频，甚至在数据中没

有出现过。这时，由于  $c(w_1 w_2 \dots w_m) \approx 0$ ，公式(2.19)的结果会不准确，甚至产生 0 概率的情况。这是观测低频事件时经常出现的问题。对于这个问题，另一种思路是对多个联合出现的事件进行独立性假设，这里可以假设  $w_1, w_2 \dots w_m$  的出现是相互独立的，于是：

$$P(w_1 w_2 \dots w_m) = P(w_1)P(w_2) \dots P(w_m) \quad (2.20)$$

这样，单词序列的出现的概率被转化为每个单词概率的乘积。由于单词的概率估计是相对准确的，因此整个序列的概率会比较合理。但是，这种独立性假设也破坏了句子中单词之间的依赖关系，造成概率估计结果的偏差。那如何更加合理的计算一个单词序列的概率呢？下面介绍的  $n$ -gram 语言建模方法可以很好地回答这个问题。

## 2.3 $n$ -gram 语言模型

在骰子游戏中，可以通过一种统计的方式，估计出在文本中词和句子出现的概率。但是在计算句子概率时往往会因为句子的样本过少而无法正确估计出句子出现的概率。为了解决这个问题，这里引入了计算整个单词序列概率  $P(w_1 w_2 \dots w_m)$  的方法——统计语言模型。下面将重点介绍  $n$ -gram 语言模型。它是一种经典的统计语言模型，而且在机器翻译及其他自然语言处理任务中有非常广泛的应用。

### 2.3.1 建模

**语言模型** (Language Model) 的目的是描述文字序列出现的规律，其对问题建模的过程被称作**语言建模** (Language Modeling)。如果使用统计建模的方式，语言模型可以被定义为计算  $P(w_1 w_2 \dots w_m)$  的问题，也就是计算整个词序列  $w_1 w_2 \dots w_m$  出现的可能性大小。具体定义如下，

**定义 2.3.1** 词汇表  $V$  上的语言模型是一个函数  $P(w_1 w_2 \dots w_m)$ ，它表示  $V^+$  上的一个概率分布。其中，对于任何词串  $w_1 w_2 \dots w_m \in V^+$ ，有  $P(w_1 w_2 \dots w_m) \geq 0$ 。而且对于所有的词串，函数满足归一化条件  $\sum_{w_1 w_2 \dots w_m \in V^+} P(w_1 w_2 \dots w_m) = 1$ 。

直接求  $P(w_1 w_2 \dots w_m)$  并不简单，因为如果把整个词串  $w_1 w_2 \dots w_m$  作为一个变量，模型的参数量会非常大。 $w_1 w_2 \dots w_m$  有  $|V|^m$  种可能性，这里  $|V|$  表示词汇表大小。显然，当  $m$  增大时，模型的复杂度会急剧增加，甚至都无法进行存储和计算。既然把  $w_1 w_2 \dots w_m$  作为一个变量不好处理，就可以考虑对这个序列的生成过程进行分解。使用链式法则（见2.1.3 节），很容易得到：

$$P(w_1 w_2 \dots w_m) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_m|w_1 w_2 \dots w_{m-1}) \quad (2.21)$$

这样， $w_1 w_2 \dots w_m$  的生成可以被看作是逐个生成每个单词的过程，即首先生成

$w_1$ , 然后根据  $w_1$  再生成  $w_2$ , 然后根据  $w_1w_2$  再生成  $w_3$ , 以此类推, 直到根据所有前  $m-1$  个词生成序列的最后一个单词  $w_m$ 。这个模型把联合概率  $P(w_1w_2\dots w_m)$  分解为多个条件概率的乘积, 虽然对生成序列的过程进行了分解, 但是模型的复杂度和以前是一样的, 比如,  $P(w_m|w_1w_2\dots w_{m-1})$  仍然不好计算。

换一个角度看,  $P(w_m|w_1w_2\dots w_{m-1})$  体现了一种基于“历史”的单词生成模型, 也就是把前面生成的所有单词作为“历史”, 并参考这个“历史”生成当前单词。但是这个“历史”的长度和整个序列长度是相关的, 也是一种长度变化的历史序列。为了化简问题, 一种简单的想法是使用定长历史, 比如, 每次只考虑前面  $n-1$  个历史单词来生成当前单词。这就是  $n$ -gram 语言模型, 其中  $n$ -gram 表示  $n$  个连续单词构成的单元, 也被称作  **$n$  元语法单元**。这个模型的数学描述如下:

$$P(w_m|w_1w_2\dots w_{m-1}) = P(w_m|w_{m-n+1}\dots w_{m-1}) \quad (2.22)$$

如表2.2所示, 整个序列  $w_1w_2\dots w_m$  的生成概率可以被重新定义为:

表 2.2 基于  $n$ -gram 的序列生成概率

链式法则	1-gram	2-gram	...	$n$ -gram
$P(w_1w_2\dots w_m) =$	$P(w_1w_2\dots w_m) =$	$P(w_1w_2\dots w_m) =$	$\dots$	$P(w_1w_2\dots w_m) =$
$P(w_1)\times$	$P(w_1)\times$	$P(w_1)\times$	$\dots$	$P(w_1)\times$
$P(w_2 w_1)\times$	$P(w_2)\times$	$P(w_2 w_1)\times$	$\dots$	$P(w_2 w_1)\times$
$P(w_3 w_1w_2)\times$	$P(w_3)\times$	$P(w_3 w_2)\times$	$\dots$	$P(w_3 w_1w_2)\times$
$P(w_4 w_1w_2w_3)\times$	$P(w_4)\times$	$P(w_4 w_3)\times$	$\dots$	$P(w_4 w_1w_2w_3)\times$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$P(w_m w_1\dots w_{m-1})$	$P(w_m)$	$P(w_m w_{m-1})$	$\dots$	$P(w_m w_{m-n+1}\dots w_{m-1})$

可以看到, 1-gram 语言模型只是  $n$ -gram 语言模型的一种特殊形式。基于独立性假设, 1-gram 假定当前单词出现与否与任何历史都无关, 这种方法大大简化了求解句子概率的复杂度。比如, 上一节中公式(2.20)就是一个 1-gram 语言模型。但是, 句子中的单词并非完全相互独立的, 这种独立性假设并不能完美地描述客观世界的问题。如果需要更精确地获取句子的概率, 就需要使用更长的“历史”信息, 比如, 2-gram、3-gram、甚至更高阶的语言模型。

$n$ -gram 的优点在于, 它所使用的历史信息是有限的, 即  $n-1$  个单词。这种性质也反映了经典的马尔可夫链的思想<sup>[37, 38]</sup>, 有时也被称作马尔可夫假设或者马尔可夫属性。因此  $n$ -gram 也可以被看作是变长序列上的一种马尔可夫模型, 比如, 2-gram 语言模型对应着 1 阶马尔可夫模型, 3-gram 语言模型对应着 2 阶马尔可夫模型, 以此类推。

那么, 如何计算  $P(w_m|w_{m-n+1}\dots w_{m-1})$  呢? 有很多种选择, 比如:

- **基于频次的方法**。直接利用词序列在训练数据中出现的频次计算出  $P(w_m|w_{m-n+1}\dots w_{m-1})$

$\dots w_{m-1}$ ):

$$P(w_m|w_{m-n+1} \dots w_{m-1}) = \frac{c(w_{m-n+1} \dots w_m)}{c(w_{m-n+1} \dots w_{m-1})} \quad (2.23)$$

其中,  $c(\cdot)$  是在训练数据中统计频次的函数。

- **人工神经网络方法**。构建一个人工神经网络估计  $P(w_m|w_{m-n+1} \dots w_{m-1})$  的值, 比如, 可以构建一个前馈神经网络来对  $n$ -gram 进行建模。

极大似然估计方法(基于频次的方法)和掷骰子游戏中介绍的统计词汇概率的方法是一致的, 它的核心是使用  $n$ -gram 出现的频次进行参数估计。基于人工神经网络的方法在近些年也非常受关注, 它直接利用多层神经网络对问题的输入  $w_{m-n+1} \dots w_{m-1}$  和输出  $P(w_m|w_{m-n+1} \dots w_{m-1})$  进行建模, 而模型的参数通过网络中神经元之间连接的权重进行体现。严格来说, 基于人工神经网络的方法并不算基于  $n$ -gram 的方法, 或者说它并没有显性记录  $n$ -gram 的生成概率, 也不依赖  $n$ -gram 的频次进行参数估计。为了保证内容的连贯性, 接下来仍以传统  $n$ -gram 语言模型为基础进行讨论, 基于人工神经网络的方法将会在第九章进行详细介绍。

$n$ -gram 语言模型的使用非常简单。可以直接用它来对词序列出现的概率进行计算。比如, 可以使用一个 2-gram 语言模型计算一个句子出现的概率, 其中单词之间用斜杠分隔, 如下:

$$\begin{aligned} & P_{2\text{-gram}}(\text{确实}/\text{现在}/\text{数据}/\text{很}/\text{多}) \\ &= P(\text{确实}) \times P(\text{现在}|\text{确实}) \times P(\text{数据}|\text{现在}) \times \\ & \quad P(\text{很}|\text{数据}) \times P(\text{多}|\text{很}) \end{aligned} \quad (2.24)$$

以  $n$ -gram 语言模型为代表的统计语言模型的应用非常广泛。除了将要在第三章中介绍的全概率分词方法, 在文本生成、信息检索、摘要等自然语言处理任务中, 语言模型都有举足轻重的地位。包括近些年非常受关注的预训练模型, 本质上也是统计语言模型。这些技术都会在后续章节进行介绍。值得注意的是, 统计语言模型为解决自然语言处理问题提供了一个非常好的建模思路, 即: 把整个序列生成的问题转化为逐个生成单词的问题。实际上, 这种建模方式会被广泛地用于机器翻译建模, 在统计机器翻译和神经机器翻译中都会有具体的体现。

### 2.3.2 参数估计和平滑算法

对于  $n$ -gram 语言模型, 每个  $P(w_m|w_{m-n+1} \dots w_{m-1})$  都可以被看作是模型的 **参数** (Parameter)。而  $n$ -gram 语言模型的一个核心任务是估计这些参数的值, 即参数估计。通常, 参数估计可以通过在数据上的统计得到。一种简单的方法是: 给定一定数量的句子, 统计每个  $n$ -gram 出现的频次, 并利用公式(2.23)得到每个参数  $P(w_m|w_{m-n+1} \dots w_{m-1})$  的值。这个过程也被称作模型的**训练** (Training)。对于自然

语言处理任务来说，统计模型的训练是至关重要的。在本书后面的内容中也会看到，不同的问题可能需要不同的模型以及不同的模型训练方法，并且很多研究工作也都集中在优化模型训练的效果上。

回到  $n$ -gram 语言模型上。前面所使用的参数估计方法并不完美，因为它无法很好地处理低频或者未见现象。比如，在式(2.24)所示的例子中，如果语料中从没有“确实”和“现在”两个词连续出现的情况，即  $c(\text{确实}/\text{现在}) = 0$ 。那么使用 2-gram 计算句子“确实/现在/数据/很/多”的概率时，会出现如下情况：

$$\begin{aligned} P(\text{现在}|\text{确实}) &= \frac{c(\text{确实}/\text{现在})}{c(\text{确实})} \\ &= \frac{0}{c(\text{确实})} \\ &= 0 \end{aligned} \tag{2.25}$$

显然，这个结果是不合理的。因为即使语料中没有“确实”和“现在”两个词连续出现，这种搭配也是客观存在的。这时简单地用极大似然估计得到概率却是 0，导致整个句子出现的概率为 0。更常见的问题是那些根本没有出现在词表中的词，称为**未登录词**（Out-of-vocabulary Word, OOV Word），比如一些生僻词，可能模型训练阶段从来没有看到过，这时模型仍然会给出 0 概率。图2.10展示了一个真实语料库中词语出现频次的分布，可以看到绝大多数词都是低频词。

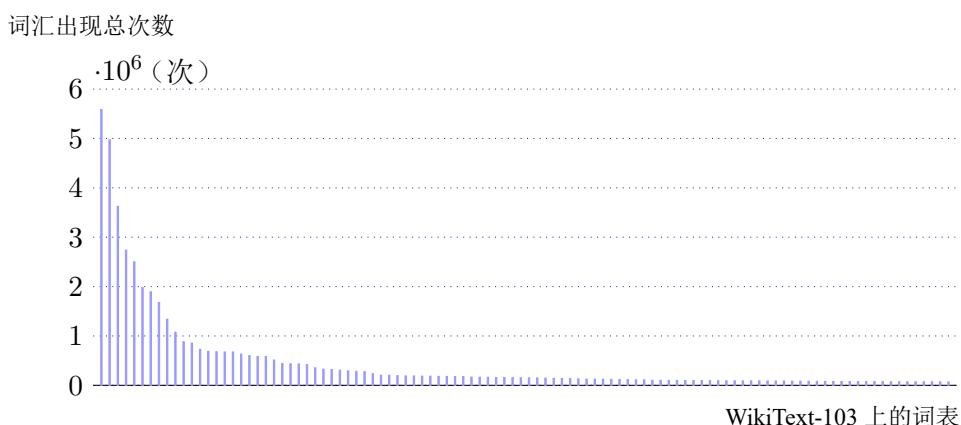


图 2.10 词汇出现频次的分布

为了解决未登录词引起的零概率问题，常用的做法是对模型进行**平滑**（Smoothing），也就是给可能出现零概率的情况一个非零的概率，使得模型不会对整个序列给出零概率。平滑可以用“劫富济贫”这一思想理解，在保证所有情况的概率和为 1 的前提下，使极低概率的部分可以从高概率的部分分配到一部分概率，从而达到平滑的目的。

语言模型使用的平滑算法有很多。在本节中，主要介绍三种平滑方法：加法平

滑法、古德-图灵估计法和 Kneser-Ney 平滑。这些方法也可以被应用到其他任务的概率平滑操作中。

## 1. 加法平滑方法

**加法平滑** (Additive Smoothing) 是一种简单的平滑技术。通常情况下，系统开发者会利用采集到的语料库来模拟真实的全部语料库。当然，没有一个语料库能覆盖所有的语言现象。假设有一个语料库  $C$ ，其中从未出现“确实现在”这样的 2-gram，现在要计算一个句子  $S = \text{“确实/现在/物价/很/高”}$  的概率。当计算“确实/现在”的概率时， $P(S) = 0$ ，导致整个句子的概率为 0。

加法平滑方法假设每个  $n$ -gram 出现的次数比实际统计次数多  $\theta$  次， $0 < \theta \leq 1$ 。这样，计算概率的时候分子部分不会为 0。重新计算  $P(\text{现在}|\text{确实})$ ，可以得到：

$$\begin{aligned} P(\text{现在}|\text{确实}) &= \frac{\theta + c(\text{确实}/\text{现在})}{\sum_w^{|V|} (\theta + c(\text{确实}/w))} \\ &= \frac{\theta + c(\text{确实}/\text{现在})}{\theta |V| + c(\text{确实})} \end{aligned} \quad (2.26)$$

其中， $V$  表示词表， $|V|$  为词表中单词的个数， $w$  为词表中的一个词， $c$  表示统计单词或短语出现的次数。有时候，加法平滑方法会将  $\theta$  取 1，这时称之为加一平滑或是拉普拉斯平滑。这种方法比较容易理解，也比较简单，因此也往往被用于对系统的快速原型中。

举一个例子。假设在一个英语文档中随机采样一些单词（词表大小  $|V| = 20$ ），各个单词出现的次数为：“look”出现 4 次，“people”出现 3 次，“am”出现 2 次，“what”出现 1 次，“want”出现 1 次，“do”出现 1 次。图 2.11 给出了在平滑之前和平滑之后的概率分布。

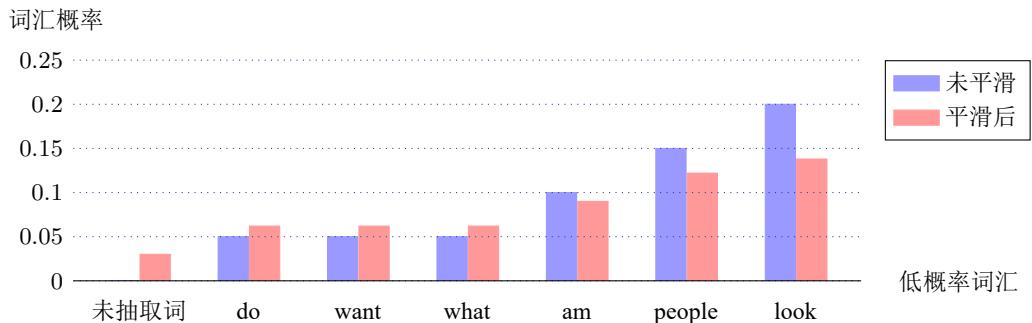


图 2.11 无平滑和有平滑的概率分布

## 2. 古德-图灵估计方法

**古德-图灵估计**(Good-Turing Estimate)是 Alan Turing 和他的助手 Irving John Good 开发的，作为他们在二战期间破解德国密码机 Enigma 所使用方法的一部分，在 1953 年 Irving John Good 将其发表。这一方法也是很多平滑算法的核心，其基本思路是：把非零的  $n$  元语法单元的概率降低，匀给一些低概率  $n$  元语法单元，以减小最大似然估计与真实概率之间的偏离<sup>[39, 40]</sup>。

假定在语料库中出现  $r$  次的  $n$ -gram 有  $n_r$  个，特别的，出现 0 次的  $n$ -gram(即未登录词及词串)有  $n_0$  个。语料库中全部单词的总个数为  $N$ ，显然：

$$N = \sum_{r=1}^{\infty} r n_r \quad (2.27)$$

这时，出现  $r$  次的  $n$ -gram 的相对频率为  $r/N$ ，也就是不做平滑处理时的概率估计。为了解决零概率问题，对于任何一个出现  $r$  次的  $n$ -gram，古德-图灵估计法利用出现  $r+1$  次的  $n$ -gram 统计量重新假设它出现  $r^*$  次：

$$r^* = (r+1) \frac{n_{r+1}}{n_r} \quad (2.28)$$

基于这个公式，就可以估计所有 0 次  $n$ -gram 的频次  $n_0 r^* = (r+1)n_1 = n_1$ 。要把这个重新估计的统计数转化为概率，需要进行归一化处理。对于每个统计数为  $r$  的事件，其概率为：

$$P_r = \frac{r^*}{N} \quad (2.29)$$

其中：

$$\begin{aligned} N &= \sum_{r=0}^{\infty} r^* n_r \\ &= \sum_{r=0}^{\infty} (r+1) n_{r+1} \\ &= \sum_{r=1}^{\infty} r n_r \end{aligned} \quad (2.30)$$

也就是说，公式(2.30)中使用的  $N$  仍然为这个整个样本分布最初的计数。所有出

现事件（即  $r > 0$ ）的概率之和为：

$$\begin{aligned} P(r > 0) &= \sum_{r>0} P_r \\ &= 1 - \frac{n_1}{N} \\ &< 1 \end{aligned} \tag{2.31}$$

其中  $n_1/N$  就是分配给所有出现为 0 次事件的概率。古德-图灵方法最终通过出现 1 次的  $n$ -gram 估计了出现为 0 次的事件概率，达到了平滑的效果。

下面通过一个例子来说明这个方法是如何对事件出现的可能性进行平滑的。仍然考虑在加法平滑法中统计单词的例子，根据古德-图灵方法进行修正如表2.3所示。

表 2.3 单词出现频次及古德-图灵平滑结果

$r$	$n_r$	$r^*$	$P_r$
0	14	0.21	0.018
1	3	0.67	0.056
2	1	3	0.25
3	1	4	0.333
4	1	-	-

但是在  $r$  很大的时候经常会出现  $n_{r+1} = 0$  的情况。通常，古德-图灵方法可能无法很好的处理这种复杂的情况，不过该方法仍然是其他一些平滑方法的基础。

### 3. Kneser-Ney 平滑方法

Kneser-Ney 平滑方法是由 Reinhard Kneser 和 Hermann Ney 于 1995 年提出的用于计算  $n$  元语法概率分布的方法<sup>[41, 42]</sup>，并被广泛认为是最有效的平滑方法之一。这种平滑方法改进了 Absolute Discounting<sup>[43, 44]</sup> 中与高阶分布相结合的低阶分布的计算方法，使不同阶分布得到充分的利用。这种算法也综合利用了其他多种平滑算法的思想。

首先介绍一下 Absolute Discounting 平滑算法，公式如下所示：

$$P_{\text{AbsDiscount}}(w_i | w_{i-1}) = \frac{c(w_{i-1} w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1}) P(w_i) \tag{2.32}$$

其中  $d$  表示被裁剪的值， $\lambda$  是一个正则化常数。可以看到第一项是经过减值调整后的 2-gram 的概率值，第二项则相当于一个带权重  $\lambda$  的 1-gram 的插值项。然而这种插值模型极易受到原始 1-gram 模型  $P(w_i)$  的干扰。

假设这里使用 2-gram 和 1-gram 的插值模型预测下面句子中下划线处的词

I cannot see without my reading \_\_\_\_\_

直觉上应该会猜测这个地方的词应该是“glasses”，但是在训练语料库中“Francisco”出现的频率非常高。如果在预测时仍然使用的是标准的 1-gram 模型，那么系统会高概率选择“Francisco”填入下划线处，这个结果显然是不合理的。当使用混合的插值模型时，如果“reading Francisco”这种二元语法并没有出现在语料中，就会导致 1-gram 对结果的影响变大，仍然会做出与标准 1-gram 模型相同的结果，犯下相同的错误。

观察语料中的 2-gram 发现，“Francisco”的前一个词仅可能是“San”，不会出现“reading”。这个分析证实了，考虑前一个词的影响是有帮助的，比如仅在前一个词是“San”时，才给“Francisco”赋予一个较高的概率值。基于这种想法，改进原有的 1-gram 模型，创造一个新的 1-gram 模型  $P_{\text{continuation}}$ ，简写为  $P_{\text{cont}}$ 。这个模型可以通过考虑前一个词的影响评估当前词作为第二个词出现的可能性。

为了评估  $P_{\text{cont}}$ ，统计使用当前词作为第二个词所出现 2-gram 的种类，2-gram 种类越多，这个词作为第二个词出现的可能性越高：

$$P_{\text{cont}}(w_i) \propto |\{w_{i-1} : c(w_{i-1}w_i) > 0\}| \quad (2.33)$$

其中，公式(2.33)右端表示求出在  $w_i$  之前出现过的  $w_{i-1}$  的数量。接下来通过对全部的二元语法单元的种类做归一化可得到评估公式：

$$P_{\text{cont}}(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}w_i) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}w_j) > 0\}|} \quad (2.34)$$

分母中对二元语法单元种类的统计还可以写为另一种形式：

$$P_{\text{cont}}(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}w_i) > 0\}|}{\sum_{w'_i} |\{w'_{i-1} : c(w'_{i-1}w'_i) > 0\}|} \quad (2.35)$$

结合基础的 Absolute discounting 计算公式，可以得到 Kneser-Ney 平滑方法的公式：

$$P_{\text{KN}}(w_i | w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{\text{cont}}(w_i) \quad (2.36)$$

其中：

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w_i : c(w_{i-1}w_i) > 0\}| \quad (2.37)$$

这里  $\max(\cdot)$  保证了分子部分为不小 0 的数，原始 1-gram 更新成  $P_{\text{cont}}$  概率分布， $\lambda$  是正则化项。

为了更具普适性，不局限于 2-gram 和 1-gram 的插值模型，利用递归的方式可以

得到更通用的 Kneser-Ney 平滑公式：

$$P_{\text{KN}}(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{\max(c_{\text{KN}}(w_{i-n+1} \dots w_i) - d, 0)}{c_{\text{KN}}(w_{i-n+1} \dots w_{i-1})} + \lambda(w_{i-n+1} \dots w_{i-1}) P_{\text{KN}}(w_i | w_{i-n+2} \dots w_{i-1}) \quad (2.38)$$

$$\lambda(w_{i-n+1} \dots w_{i-1}) = \frac{d}{c_{\text{KN}}(w_{i-n+1}^{i-1})} |\{w_i : c_{\text{KN}}(w_{i-n+1} \dots w_{i-1} w_i) > 0\}| \quad (2.39)$$

$$c_{\text{KN}}(\cdot) = \begin{cases} c(\cdot) & \text{当计算最高阶模型时} \\ \text{catcount}(\cdot) & \text{当计算低阶模型时} \end{cases} \quad (2.40)$$

其中  $\text{catcount}(\cdot)$  表示的是单词  $w_i$  作为  $n$ -gram 中第  $n$  个词时  $w_{i-n+1} \dots w_i$  的种类数目。

Kneser-Ney 平滑是很多语言模型工具的基础<sup>[45, 46]</sup>。还有很多以此为基础衍生出来的算法，感兴趣的读者可以通过参考文献自行了解<sup>[17, 42, 44]</sup>。

### 2.3.3 语言模型的评价

在使用语言模型时，往往需要知道模型的质量。**困惑度** (Perplexity, PPL) 是一种衡量语言模型的好坏的指标。对于一个真实的词序列  $w_1 \dots w_m$ ，困惑度被定义为：

$$\text{PPL} = P(w_1 \dots w_m)^{-\frac{1}{m}} \quad (2.41)$$

本质上，PPL 反映了语言模型对序列可能性预测能力的一种评估。如果  $w_1 \dots w_m$  是真实的自然语言，“完美”的模型会得到  $P(w_1 \dots w_m) = 1$ ，它对应了最低的困惑度  $\text{PPL}=1$ ，这说明模型可以完美地对词序列出现的可能性进行预测。当然，真实的语言模型是无法达到  $\text{PPL}=1$  的，比如，在著名的 Penn Treebank (PTB) 数据上最好的语言模型的 PPL 值也只能到达 35 左右。可见自然语言处理任务的困难程度。

## 2.4 预测与搜索

给定模型结构，统计语言模型的使用可以分为两个阶段：

- **训练** (Training)：从训练数据上估计出语言模型的参数。
- **预测** (Prediction)：用训练好的语言模型对新输入的句子进行概率评估，或者生成新的句子。

模型训练的内容已经在前文进行了介绍，这里重点讨论语言模型的预测。实际上，预测是统计自然语言处理中的常用概念。比如，深度学习中的**推断** (Inference)、

统计机器翻译中的**解码**（Decoding）本质上都是预测。具体到语言建模的问题上，预测通常对应两类问题：

- 预测输入句子的可能性。比如，有如下两个句子：

The boy caught the cat.

The caught boy the cat.

可以利用语言模型对其进行打分，即计算句子的生成概率，之后把语言模型的得分作为判断句子合理性的依据。显然，在这个例子中，第一句的语言模型得分更高，因此句子也更加合理。

- 预测可能生成的单词或者单词序列。比如，对于如下的例子

The boy caught \_\_\_\_\_

下划线的部分是缺失的内容，现在要将缺失的部分生成出来。理论上，所有可能的单词串都可以构成缺失部分的内容。这时可以使用语言模型得到所有可能词串构成句子的概率，之后找到概率最高的词串填入下划线处。

从词序列建模的角度看，这两类预测问题本质上是一样的。因为，它们都在使用语言模型对词序列进行概率评估。但是，从实现上看，词序列的生成问题更难。因为，它不仅要对所有可能的词序列进行打分，同时要“找到”最好的词序列。由于潜在的词序列不计其数，因此这个“找”最优词序列的过程并不简单。

实际上，生成最优词序列的问题也是自然语言处理中的一大类问题——**序列生成**（Sequence Generation）。机器翻译就是一个非常典型的序列生成问题：在机器翻译任务中，需要根据源语言词序列生成与之相对应的目标语言词序列。但是语言模型本身并不能“制造”单词序列的。因此，严格地说，序列生成问题的本质并非让语言模型凭空“生成”序列，而是使用语言模型在所有候选的单词序列中“找出”最佳序列。这个过程对应着经典的**搜索问题**（Search Problem）。下面将着重介绍序列生成背后的建模方法，以及在序列生成里常用的搜索技术。

### 2.4.1 搜索问题的建模

基于语言模型的序列生成问题可以被定义为：在无数任意排列的单词序列中找到概率最高的序列。这里单词序列  $w = w_1 w_2 \dots w_m$  的语言模型得分  $P(w)$  度量了这个序列的合理性和流畅性。在序列生成任务中，基于语言模型的搜索问题可以被描述为：

$$\hat{w} = \arg \max_{w \in \chi} P(w) \quad (2.42)$$

这里  $\arg$  即 argument(参数),  $\arg \max_x f(x)$  表示返回使  $f(x)$  达到最大的  $x$ 。 $\arg \max_{w \in \chi}$

$P(w)$  表示找到使语言模型得分  $P(w)$  达到最大的单词序列  $w$ 。 $\chi$  是搜索问题的解空间，它是所有可能的单词序列  $w$  的集合。 $\hat{w}$  可以被看做该搜索问题中的“最优解”，即概率最大的单词序列。

在序列生成任务中，最简单的策略就是对词表中的词汇进行任意组合，通过这种枚举的方式得到全部可能的序列。但是，很多时候待生成序列的长度是无法预先知道的。比如，机器翻译中目标语序列的长度是任意的。那么怎样判断一个序列何时完成了生成过程呢？这里借用现代人类书写中文和英文的过程：句子的生成首先从一片空白开始，然后从左到右逐词生成，除了第一个单词，所有单词的生成都依赖于前面已经生成的单词。为了方便计算机实现，通常定义单词序列从一个特殊的符号  $\langle \text{sos} \rangle$  后开始生成。同样地，一个单词序列的结束也用一个特殊的符号  $\langle \text{eos} \rangle$  来表示。

对于一个序列  $\langle \text{sos} \rangle \text{ I agree } \langle \text{eos} \rangle$ ，图2.12展示语言模型视角下该序列的生成过程。该过程通过在序列的末尾不断附加词表中的单词来逐渐扩展序列，直到这段序列结束。这种生成单词序列的过程被称作**自左向右生成**（Left-to-Right Generation）。注意，这种序列生成策略与  $n$ -gram 的思想天然契合，因为  $n$ -gram 语言模型中，每个词的生成概率依赖前面（左侧）若干词，因此  $n$ -gram 语言模型也是一种自左向右的计算模型。

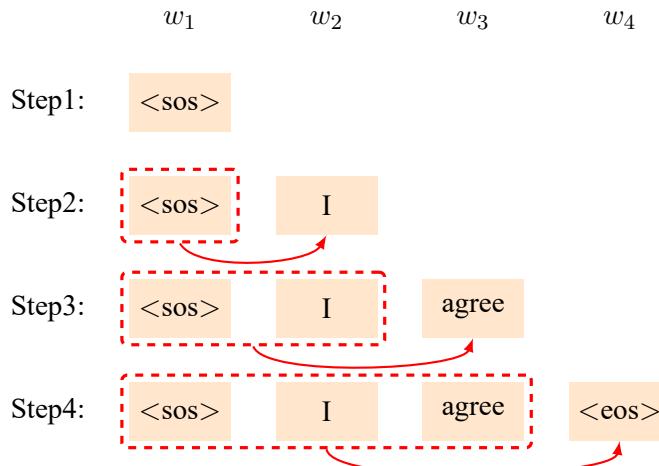


图 2.12 序列生成过程

在这种序列生成方式的基础上，实现搜索通常有两种方法——深度优先遍历和宽度优先遍历<sup>[47]</sup>。在深度优先遍历中，每次从词表中可重复地选择一个单词，然后从左至右地生成序列，直到  $\langle \text{eos} \rangle$  被选择，此时一个完整的单词序列被生成出来。然后从  $\langle \text{eos} \rangle$  回退到上一个单词，选择之前词表中未被选择到的候选单词代替  $\langle \text{eos} \rangle$ ，并继续挑选下一个单词直到  $\langle \text{eos} \rangle$  被选到，如果上一个单词的所有可能都被枚举过，那么回退到上一个单词继续枚举，直到回退到  $\langle \text{sos} \rangle$ ，这时候枚举结束。在宽度优

先遍历中，每次不是只选择一个单词，而是枚举所有单词。

有一个简单的例子。假设词表只含两个单词  $\{a, b\}$ ，从  $\langle \text{sos} \rangle$  开始枚举所有候选，有三种可能：

$$\{\langle \text{sos} \rangle a, \langle \text{sos} \rangle b, \langle \text{sos} \rangle \langle \text{eos} \rangle\}$$

其中可以划分成长度为 0 的完整的单词序列集合  $\{\langle \text{sos} \rangle \langle \text{eos} \rangle\}$  和长度为 1 的未结束的单词序列片段集合  $\{\langle \text{sos} \rangle a, \langle \text{sos} \rangle b\}$ ，然后下一步对未结束的单词序列枚举词表中的所有单词，可以生成：

$$\{\langle \text{sos} \rangle a a, \langle \text{sos} \rangle a b, \langle \text{sos} \rangle a \langle \text{eos} \rangle, \langle \text{sos} \rangle b a, \langle \text{sos} \rangle b b, \langle \text{sos} \rangle b \langle \text{eos} \rangle\}$$

此时可以划分出长度为 1 的完整单词序列集合  $\{\langle \text{sos} \rangle a \langle \text{eos} \rangle, \langle \text{sos} \rangle b \langle \text{eos} \rangle\}$ ，以及长度为 2 的未结束单词序列片段集合  $\{\langle \text{sos} \rangle a a, \langle \text{sos} \rangle a b, \langle \text{sos} \rangle b a, \langle \text{sos} \rangle b b\}$ 。以此类推，继续生成未结束序列，直到单词序列的长度达到所允许的最大长度。

对于这两种搜索算法，通常可以从以下四个方面评价：

- **完备性：**当问题有解时，使用该策略能否找到问题的解。
- **最优性：**搜索策略能否找到最优解。
- **时间复杂度：**找到最优解需要多长时间。
- **空间复杂度：**执行策略需要多少内存。

当任务对单词序列长度没有限制时，上述两种方法枚举出的单词序列也是无穷无尽的。因此这两种枚举策略并不具备完备性而且会导致枚举过程无法停止。由于日常生活中通常不会见到特别长的句子，因此可以通过限制单词序列的最大长度来避免这个问题。一旦单词序列的最大长度被确定，以上两种枚举策略就可以在一定时间内枚举出所有可能的单词序列，因而一定可以找到最优的单词序列，即具备最优性。

表 2.4 枚举的两种实现方式比较

	时间复杂度	空间复杂度
深度优先	$O(( V +1)^{m-1})$	$O(m)$
宽度优先	$O(( V +1)^{m-1})$	$O(( V +1)^m)$

此时上述生成策略虽然可以满足完备性和最优性，但其仍然算不上是优秀的生成策略，因为这两种算法在时间复杂度和空间复杂度上的表现很差，如表2.4所示。其中  $|V|$  为词表大小， $m$  为序列长度。值得注意的是，在之前的遍历过程中，除了在序列开头一定会挑选  $\langle \text{sos} \rangle$  之外，其他位置每次可挑选的单词并不只有词表中的单词，还有结束符号  $\langle \text{eos} \rangle$ ，因此实际上生成过程中每个位置的单词候选数量为  $|V| + 1$ 。

那么是否有比枚举策略更高效的方法呢？答案是肯定的。一种直观的方法是将搜索的过程表示成树型结构，称为解空间树。它包含了搜索过程中可生成的全部序列。该树的根节点恒为  $\langle \text{sos} \rangle$ ，代表序列均从  $\langle \text{sos} \rangle$  开始。该树结构中非叶子节点的兄弟节点有  $|V| + 1$  个，由词表和结束符号  $\langle \text{eos} \rangle$  构成。从图2.13可以看到，对于一个最大长度为 4 的序列的搜索过程，生成某个单词序列的过程实际上就是访问解空间树中从根节点  $\langle \text{sos} \rangle$  开始一直到叶子节点  $\langle \text{eos} \rangle$  结束的某条路径，而这条的路径上节点按顺序组成了一段独特的单词序列。此时对所有可能单词序列的枚举就变成了对解空间树的遍历。并且枚举的过程与语言模型打分的过程也是一致的，每枚举一个词  $i$  也就是在图2.13选择  $w_i$  一列的一个节点，语言模型就可以为当前的树节点  $w_i$  给出一个分值，即  $P(w_i | w_1 w_2 \dots w_{i-1})$ 。对于  $n$ -gram 语言模型，这个分值可以表示为  $P(w_i | w_1 w_2 \dots w_{i-1}) = P(w_i | w_{i-n+1} \dots w_{i-1})$

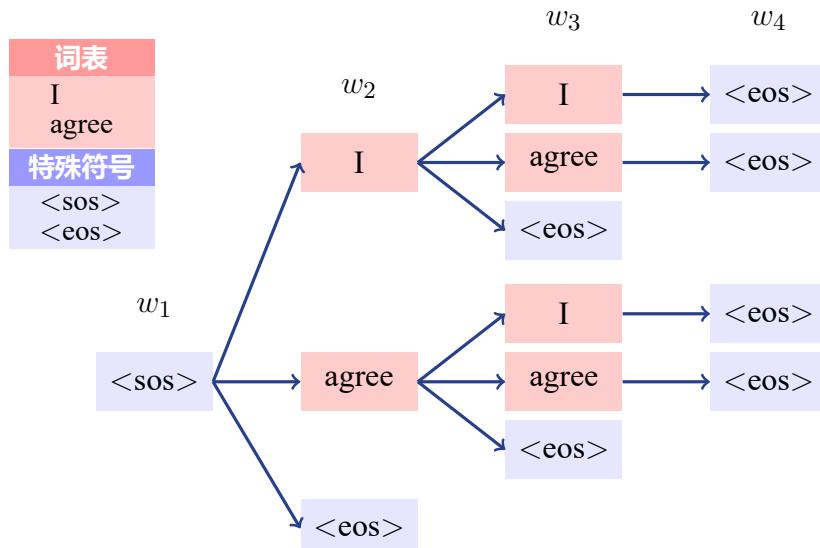


图 2.13 对有限长序列进行枚举搜索时的解空间树

从这个角度来看，在树的遍历中，可以很自然地引入语言模型打分：在解空间树中引入节点的权重——将当前节点  $i$  的得分重设为语言模型打分  $\log P(w_i | w_1 w_2 \dots w_{i-1})$ ，其中  $w_1 w_2 \dots w_{i-1}$  是该节点的全部祖先。与先前不同的是，由于在使用语言模型打分时，词的概率通常小于 1，因此句子很长时概率会非常小，容易造成浮点误差，所以这里使用概率的对数形式  $\log P(w_i | w_1 w_2 \dots w_{i-1})$  代替  $P(w_i | w_1 w_2 \dots w_{i-1})$ 。此时对于图中一条包含  $\langle \text{eos} \rangle$  的完整序列来说，它的最终得分  $\text{score}(\cdot)$  可以被定义为：

$$\begin{aligned} \text{score}(w_1 w_2 \dots w_m) &= \log P(w_1 w_2 \dots w_m) \\ &= \sum_{i=1}^m \log P(w_i | w_1 w_2 \dots w_{i-1}) \end{aligned} \quad (2.43)$$

通常， $\text{score}(\cdot)$  也被称作**模型得分**（Model Score）。如图2.14所示，可知红线所示单词序列“<sos> I agree <eos>”的模型得分为：

$$\begin{aligned}
 & \text{score}(<\text{sos}> \text{ I agree } <\text{eos}>) \\
 = & \log P(<\text{sos}>) + \log P(\text{I}|<\text{sos}>) + \log P(\text{agree}|<\text{sos}> \text{ I}) + \log P(<\text{sos}>|<\text{sos}> \text{ I agree}) \\
 = & 0 - 0.5 - 0.2 - 0.8 \\
 = & -1.5
 \end{aligned} \tag{2.44}$$

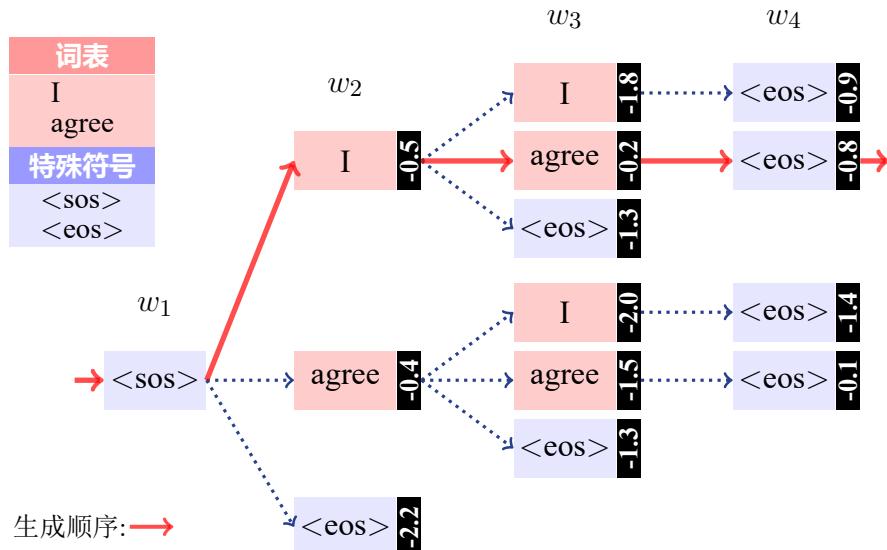


图 2.14 通过语言模型对解空间树打分

这样，语言模型的打分与解空间树的遍历就融合在一起了。于是，序列生成的问题可以被重新描述为：寻找所有单词序列组成的解空间树中权重总和最大的一条路径。在这个定义下，前面提到的两种枚举词序列的方法就是经典的**深度优先搜索**（Depth-first Search）和**宽度优先搜索**（Breadth-first Search）的雏形<sup>[48, 49]</sup>。在后面的内容中，从遍历解空间树的角度出发，可以对这些原始的搜索策略的效率进行优化。

## 2.4.2 经典搜索

人工智能领域有很多经典的搜索策略，这里将对无信息搜索和启发性搜索进行简要介绍。

### 1. 无信息搜索

在解空间树中，在每次对一个节点进行扩展的时候，可以借助语言模型计算当前节点的权重。因此很自然的一个想法是：使用权重信息可以帮助系统更快地找到合适的解。

在深度优先搜索中，每次总是先挑选一个单词，等枚举完当前单词全部子节点构成的序列后，才会选择下一个兄弟节点继续进行搜索。但是在挑选过程中先枚举词表中的哪个词是未定义的，也就是先选择哪个兄弟节点进行搜索是随机的。既然最终目标是寻找权重之和最大的路径，那么可以优先挑选分数较高的单词进行枚举。如图2.15所示，红色线表示了第一次搜索的路径。在路径长度有限的情况下，权重和最大的路径上每个节点的权重也会比较大，先尝试分数较大的单词可以让系统更快地找到最优解，这是对深度优先搜索的一个自然的扩展。

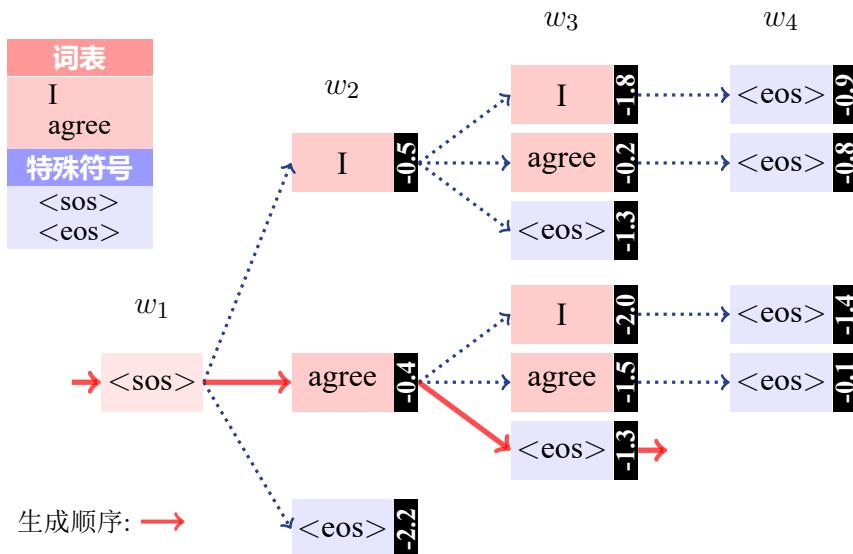


图 2.15 深度优先搜索扩展方法实例

类似的思想也可以应用于宽度优先搜索，由于宽度优先搜索每次都选择了所有的单词，因此简单使用节点的权重来选择单词是不可行的。重新回顾宽度优先搜索的过程：它维护了一个未结束单词序列的集合，每次扩展单词序列后根据长度往集合里面加入单词序列。而搜索问题关心的是单词序列的得分而非其长度。因此可以在搜索过程中维护未结束的单词序列集合里每个单词序列的得分，然后优先扩展该集合中得分最高的单词序列，使得扩展过程中未结束的单词序列集合包含的单词序列分数逐渐变高。如图2.16所示，由于“<sos> I”在图右侧的5条路径中分数最高，因此下一步将要扩展  $w_2$  一列 “I” 节点后的全部后继。图中绿色节点表示下一步将要扩展的节点。普通宽度优先搜索中，扩展后生成的单词序列长度相同，但是分数却参差不齐。而改造后的宽度优先搜索则不同，它会优先生成得分较高的单词序列，这种宽度优先搜索也叫做**一致代价搜索**（Uniform-cost Search）<sup>[50]</sup>。

上面描述的两个改进后的搜索方法属于**无信息搜索**（Uninformed Search）<sup>[51]</sup>，因为它们依赖的信息仍然来源于问题本身而不是问题以外。改进后的方法虽然有机会更早地找到分数最高的单词序列（也就是最优解）。但是由于没有一个通用的办法来判断当前找到的解是否为最优解，这种策略不会在找到最优解后自动停止，因此最

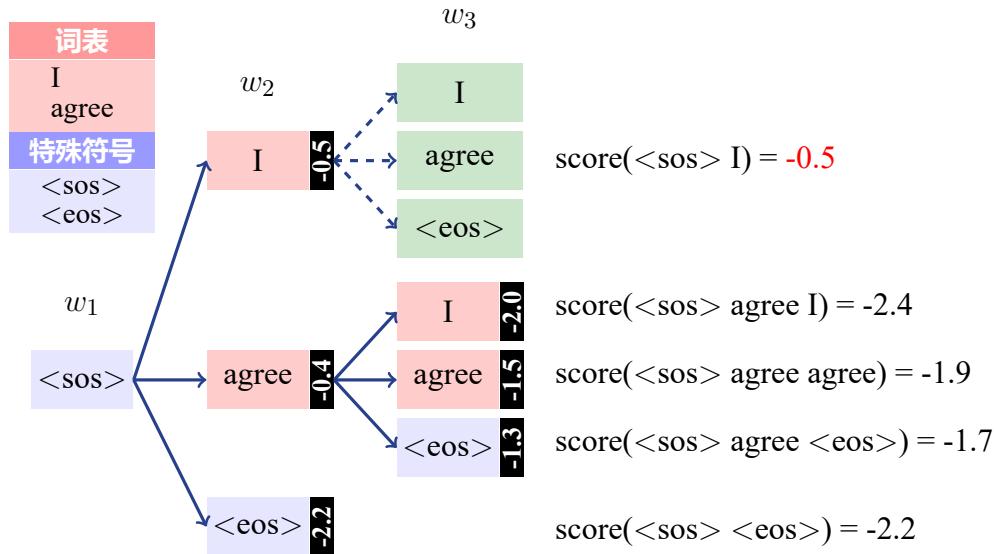


图 2.16 一致代价搜索实例

终仍然需要枚举所有可能的单词序列，寻找最优解需要的时间复杂度没有产生任何改变。尽管如此，如果只是需要一个相对好的解而不是最优解，改进后的搜索策略仍然是比原始的枚举策略更优秀的算法。

此外，由于搜索过程中将语言模型的打分作为搜索树的节点权重，另一种改进思路是：能否借助语言模型的特殊性质来对搜索树进行剪枝（Pruning），从而避免在搜索空间中访问一些不可能产生比当前解更好的结果的区域，提高搜索策略在实际运用当中的效率。简单来说，剪枝是一种可以缩小搜索空间的手段，比如，在搜索的过程中，动态的“丢弃”一些搜索路径，从而减少搜索的总代价。剪枝的程度在一定范围内影响了搜索系统的效率，剪枝越多搜索效率越高，一般找到最优解的可能性也越低；反之，搜索效率越低，但是找到最优解的可能性越大。在本章后面即将介绍的贪婪搜索和束搜索都可以被看作是剪枝方法的一种特例。

## 2. 启发式搜索

在搜索问题中，一个单词序列的生成可以分为两部分：已生成部分和未生成部分。既然最终目标是使得一个完整的单词序列得分最高，那么关注未生成部分的得分也许能为搜索策略的改进提供思路。

但是，问题在于未生成部分来自搜索树中未被搜索过的区域，因此也无法直接计算其得分。既然仅依赖于问题本身的信息无法得到未生成部分的得分，那么是否可以通过一些外部信息来估计未生成部分的得分呢？在前面所提到的剪枝技术中，借助语言模型的特性可以使得搜索变得高效。与其类似，利用语言模型的其他特性也可以实现对未生成部分得分的估计。这个对未生成部分得分的估计通常被称为**启发式函数**（Heuristic Function）。在扩展假设过程中，可以优先挑选当前得分  $\log P(w_1 w_2 \dots w_m)$

和启发式函数值  $h(w_1 w_2 \dots w_m)$  最大的候选进行扩展，从而大大提高搜索的效率。这时，模型得分可以被定义为：

$$\text{score}(w_1 w_2 \dots w_m) = \log P(w_1 w_2 \dots w_m) + h(w_1 w_2 \dots w_m) \quad (2.45)$$

这种基于启发式函数的一致代价搜索通常也被称为 A\* 搜索或**启发式搜索**(Heuristic Search)<sup>[52]</sup>。通常可以把启发式函数看成是计算当前状态跟最优解的距离的一种方法，并把关于最优解的一些性质的猜测放到启发式函数里。比如，在序列生成中，一般认为最优序列应该在某个特定的长度附近，那么就可以把启发式函数定义成该长度与当前单词序列长度的差值。这样，在搜索过程中，启发式函数会引导搜索优先生成当前得分高且序列长度接近预设长度的单词序列。

### 2.4.3 局部搜索

由于全局搜索策略要遍历整个解空间，所以它的时间、空间复杂度一般都比较高。在对于完备性与最优性要求不那么严格的搜索问题上，可以使用非经典搜索策略。非经典搜索涵盖的内容非常广泛，其中包括局部搜索<sup>[53]</sup>、连续空间搜索<sup>[54]</sup>、信念状态搜索<sup>[55]</sup> 和实时搜索<sup>[56]</sup> 等等。局部搜索是非经典搜索里的一个重要方面，局部搜索策略不必遍历完整的解空间，因此降低了时间、空间复杂度，但是这也导致可能会丢失最优解甚至找不到解，所以局部搜索都是不完备的而且非最优的。但是，在自然语言处理中，很多问题由于搜索空间过大无法使用全局搜索，因此使用局部搜索是非常普遍的。

#### 1. 贪婪搜索

**贪婪搜索**(Greedy Search) 基于一种思想：当一个问题可以拆分为多个子问题时，如果一直选择子问题的最优解就能得到原问题的最优解，那么就可以不必遍历原始的解空间，而是使用这种“贪婪”的策略进行搜索。基于这种思想，它每次都优先挑选得分最高的词进行扩展，这一点与改进过的深度优先搜索类似。但是它们的区别在于，贪婪搜索在搜索到一个完整的序列，也就是搜索到  $\langle \text{eos} \rangle$  即停止，而改进的深度优先搜索会遍历整个解空间。因此贪婪搜索非常高效，其时间和空间复杂度仅为  $O(m)$ ，这里  $m$  为单词序列的长度。

由于贪婪搜索并没有遍历整个解空间，所以该方法不保证一定能找到最优解。比如对于如图2.17所示的一个搜索结构，贪婪搜索将选择红线所示的序列，该序列的最终得分是-1.7。但是，对比图2.15可以发现，在另一条路径上有得分更高的序列“ $\langle \text{sos} \rangle$  I agree  $\langle \text{eos} \rangle$ ”，它的得分为-1.5。此时贪婪搜索并没有找到最优解，由于贪婪搜索选择的单词是当前步骤得分最高的，但是最后生成的单词序列的得分取决于它未生成部分的得分。因此当得分最高的单词的子树中未生成部分的得分远小于其他子树时，贪婪搜索提供的解的质量会非常差。同样的问题可以出现在使用贪婪搜索的任意时刻。但是，即使是这样，凭借其简单的思想以及在真实问题上的效果，贪婪搜索

在很多场景中仍然得到了深入应用。

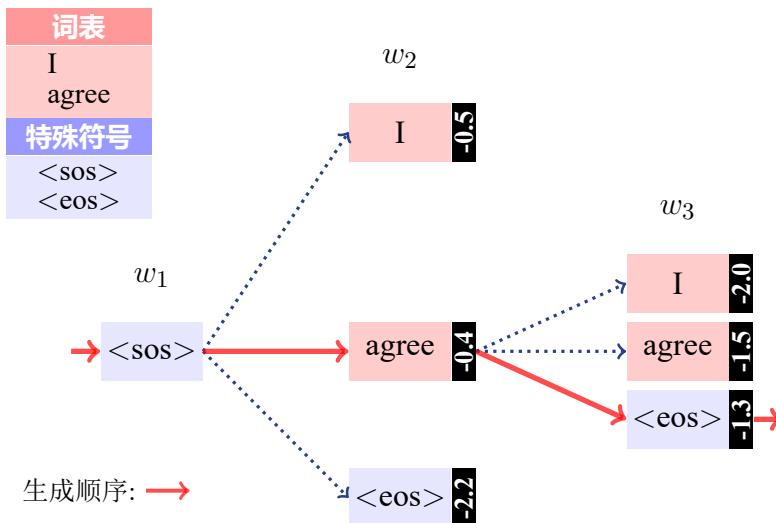


图 2.17 贪婪搜索实例

## 2. 束搜索

贪婪搜索会产生质量比较差的解是由于当前单词的错误选择造成的。既然每次只挑选一个单词可能会产生错误，那么可以通过同时考虑更多候选单词来缓解这个问题，也就是对于一个位置，可以同时将其扩展到若干个节点。这样就扩大了搜索的范围，进而使得优质解被找到的概率增大。

常见的做法是每一次生成新单词的时候都挑选得分最高的前  $B$  个单词，然后扩展这  $B$  个单词的  $T$  个孩子节点，得到  $BT$  条新路径，最后保留其中得分最高的  $B$  条路径。从另外一个角度理解，它相当于比贪婪搜索看到了更多的路径，因而它更有可能找到好的解。这个方法通常被称为**束搜索**（Beam Search）。图2.18展示了一个束大小为 3 的例子，其中束大小代表每次选择单词时保留的词数。比起贪婪搜索，束搜索在实际表现中非常优秀，它的时间、空间复杂度仅为贪婪搜索的常数倍，也就是  $O(Bm)$ 。

束搜索也有很多的改进版本。回忆一下，在无信息搜索策略中可以使用剪枝技术来提升搜索的效率。而实际上，束搜索本身也是一种剪枝方法。因此有时也把束搜索称作**束剪枝**（Beam Pruning）。在这里有很多其它的剪枝策略可供选择，例如可以只保留与当前最佳路径得分相差在  $\theta$  之内的路径，也就是进行搜索时只保留得分差距在一定范围内的路径，这种方法也被称作**直方图剪枝**（Histogram Pruning）。

对于语言模型来说，当多个路径中最高得分比当前搜索到的最好的解的得分低时，可以立刻停止搜索。因为此时序列越长语言模型得分  $\log P(w_1 w_2 \dots w_m)$  会越低，继续扩展这些路径不会产生更好的结果。这个技术通常也被称为**最佳停止条件**（Optimal Stopping Criteria）。类似的思想也被用于机器翻译等任务<sup>[57, 58]</sup>。

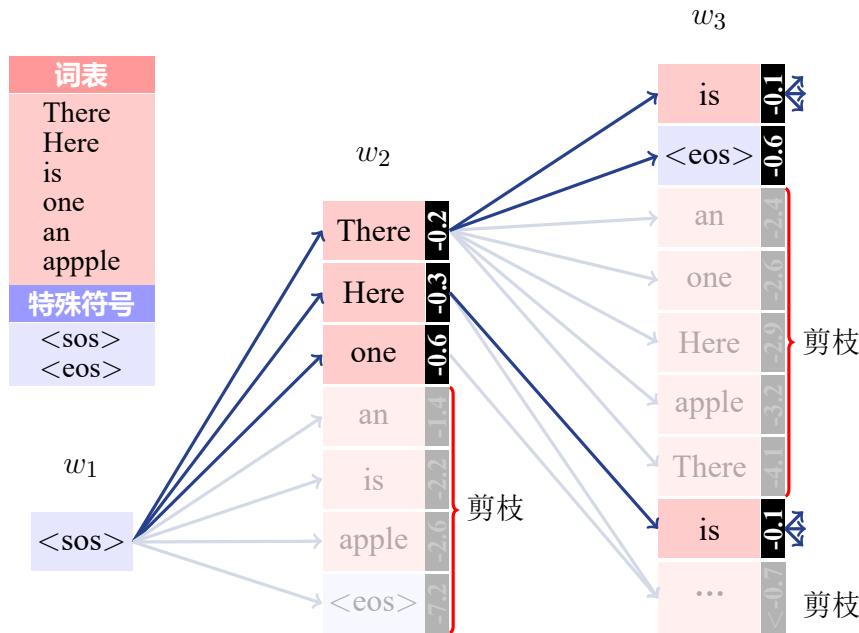


图 2.18 束搜索实例

总的来说，虽然局部搜索没有遍历完整的解空间，使得这类方法无法保证找到最优解。但是，局部搜索算法大大降低了搜索过程的时间、空间复杂度。因此在语言模型生成和机器翻译的解码过程中常常使用局部搜索算法。在第七章、第十章中还将介绍这些算法的具体应用。

## 2.5 小结及拓展阅读

本章重点介绍了如何对自然语言处理问题进行统计建模，并从数据中自动学习统计模型的参数，最终使用学习到的模型对新的问题进行处理。之后，将这种思想应用到语言建模任务中，该任务与机器翻译有着紧密的联系。通过系统化的建模，可以发现：经过适当的假设和化简，统计模型可以很好地描述复杂的自然语言处理问题。进一步，本章对面向语言模型预测的搜索方法进行了介绍。相关概念和方法也会在后续章节的内容中被广泛使用。

此外，有几方面内容，读者可以继续深入了解：

- 在  $n$ -gram 语言模型中，由于语料中往往存在大量的低频词以及未登录词，模型会产生不合理的概率预测结果。因此本章介绍了三种平滑方法，以解决上述问题。实际上，平滑方法是语言建模中的重要研究方向。除了上文中介绍的三种平滑方法之外，还有如 Jelinek-Mercer 平滑<sup>[59]</sup>、Katz 平滑<sup>[60]</sup>以及 Witten-Bell 平滑等等<sup>[61, 62]</sup> 的平滑方法。相关工作也对这些平滑方法进行了详细对比<sup>[42, 63]</sup>。

- 除了平滑方法，也有很多工作对  $n$ -gram 语言模型进行改进。比如，对于形态学丰富的语言，可以考虑对单词的形态学变化进行建模。这类语言模型在一些机器翻译系统中也体现出了很好的潜力<sup>[64, 65, 66]</sup>。此外，如何使用超大规模数据进行语言模型训练也是备受关注的研究方向。比如，有研究者探索了对超大语言模型进行压缩和存储的方法<sup>[45, 67, 68]</sup>。另一个有趣的方向是，利用随机存储算法对大规模语言模型进行有效存储<sup>[69, 70]</sup>，比如，在语言模型中使用 Bloom Filter 等随机存储的数据结构。
- 本章更多地关注了语言模型的基本问题和求解思路，但是基于  $n$ -gram 的方法并不是语言建模的唯一方法。从现在自然语言处理的前沿看，端到端的深度学习方法在很多任务中都取得了领先的性能。语言模型同样可以使用这些方法<sup>[71]</sup>，而且在近些年取得了巨大成功。例如，最早提出的前馈神经语言模型<sup>[72]</sup>和后来的基于循环单元的语言模型<sup>[73]</sup>、基于长短期记忆单元的语言模型<sup>[74]</sup>以及现在非常流行的 Transformer<sup>[23]</sup>。关于神经语言模型的内容，会在第九章进行进一步介绍。
- 最后，本章结合语言模型的序列生成任务对搜索技术进行了介绍。类似地，机器翻译任务也需要从大量的翻译候选中快速寻找最优译文。因此在机器翻译任务中也使用了搜索方法，这个过程通常被称作解码。例如，有研究者在基于词的翻译模型中尝试使用启发式搜索<sup>[75, 76, 77]</sup>以及贪婪搜索方法<sup>[78][79]</sup>，也有研究者探索基于短语的栈解码方法<sup>[80, 81]</sup>。此外，解码方法还包括有限状态机解码<sup>[82][83]</sup>以及基于语言学约束的解码<sup>[84, 85, 86, 87, 88]</sup>。相关内容将在第八章和第十四章进行介绍。





### 3. 词法分析和语法分析基础

机器翻译并非是一个孤立的系统，它依赖于很多模块，并且需要多个学科知识的融合。其中就会用到许多自然语言处理工具来对不同语言的文字进行分析。因此，在正式开始介绍机器翻译的内容之前，本章会对相关的词法分析和语法分析知识进行概述，包括：分词、命名实体识别、短语结构句法分析。它们都是自然语言处理中的经典问题，而且在机器翻译中被广泛使用。本章会重点介绍这些任务的定义和求解问题的思路。其中也会使用到统计建模方法，因此本章也可以被看作是第二章内容的延伸。

#### 3.1 问题概述

很多时候机器翻译系统被看作是孤立的“黑盒”系统（图3.1(a)）。将一段文本作为输入送入机器翻译系统之后，系统输出翻译好的译文。但是真实的机器翻译系统非常复杂，因为系统看到的输入和输出实际上只是一些符号串，这些符号并没有任何意义，因此需要进一步对这些符号串进行处理才能更好的使用它们。比如，需要定义翻译中最基本的单元是什么？符号串是否具有结构信息？如何用数学工具刻画这些基本单元和结构？

图3.1(b)展示了一个机器翻译系统的输入和输出形式。可以看到，输入的中文词串“猫喜欢吃鱼”被加工成一个新的结构（图3.2）。直觉上，这个结构有些奇怪，因为上面多了很多新的符号，而且还有一些线将不同符号进行连接。实际上这就是一种常见的句法表示——短语结构树。生成这样的结构会涉及两方面问题：

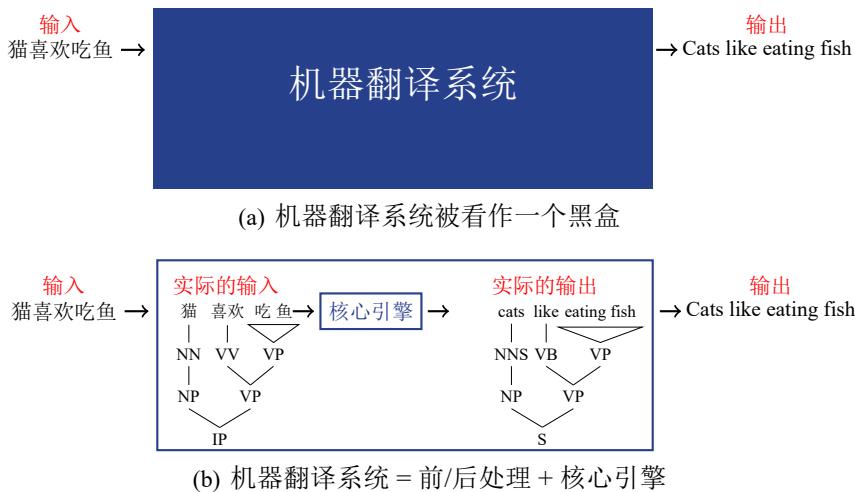


图 3.1 机器翻译系统的结构

- **分词 (Segmentation)**：这个过程会把词串进行切分，切割成最小的具有完整功能的单元——**单词 (Word)**。因为只有知道了什么是单词，机器翻译系统才能完成对句子的表示、分析和生成。
- **句法分析 (Parsing)**：这个过程会对分词的结果进行进一步分析。比如，可以对句子进行浅层分析，得到句子中实体的信息（如人名、地名等）。也可以对句子进行更深层次的分析，得到完整的句法结构，类似于图3.2中的结果。这种结构可以被看作是对句子的进一步抽象，被称为短语结构树，比如，NP+VP 就可以表示由名词短语 (Noun Phrase, NP) 和动词短语 (Verb Phrase, VP) 构成的主谓结构。利用这些信息，机器翻译可以更加准确地对句子的结构进行分析和生成。

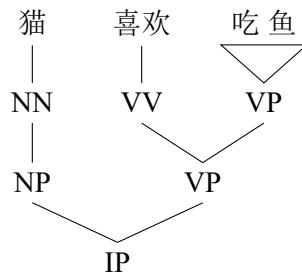


图 3.2 汉语句子“猫喜欢吃鱼”的分析结果（分词和句法分析）

类似地，机器翻译输出的结果也可以包含同样的信息。甚至系统输出英语译文之后，还有一个额外的步骤来把部分英语单词的大小写恢复出来，比如，句首单词的首字母要大写。

一般来说，在送入机器翻译系统前需要对文字序列进行处理和加工，这个过程被称为**预处理**（Pre-processing）。类似地，在机器翻译模型输出译文后进行的处理被称作**后处理**（Post-processing）。这两个过程对机器翻译性能影响很大，比如，对于神经机器翻译系统来说，不同的分词策略可能会造成翻译性能的天差地别。

值得注意的是，有些观点认为，对于机器翻译来说，不论是分词还是句法分析，并不要求符合人的认知和语言学约束。换句话说，机器翻译所使用的“单词”和“结构”本身并不是为了符合人类的解释，它们更直接目的是为了进行翻译。从系统开发的角度，有时候即使使用一些与人类的语言习惯有差别的处理，仍然会带来性能的提升，比如在神经机器翻译中，在传统分词的基础上进一步使用**双字节编码**（Byte Pair Encoding, BPE）子词切分<sup>[89]</sup>会使得机器翻译性能大幅提高。当然，自然语言处理中语言学信息的使用一直是学界关注的焦点。甚至关于语言学结构对机器翻译是否有作用这个问题也有一些不同的观点。但是不能否认的是，无论是语言学的知识，还是计算机自己学习到的知识，对机器翻译都是有价值的。在后续章节会看到，这两种类型的知识对机器翻译帮助很大。

剩下的问题是进行句子的切分和结构的分析。思路有很多，一种常用的方法是对问题进行概率化，用统计模型来描述问题并求解之。比如，一个句子切分的好坏，并不是非零即一的判断，而是要估计出这种切分的可能性大小，最终选择可能性最大的结果进行输出。这也是一种典型的用统计建模的方式来描述自然语言处理问题的方法。

本章将会对上述问题及求解问题的方法进行介绍。并将统计建模应用到中文分词、命名实体识别和短语结构句法分析等任务中。

## 3.2 中文分词

对于机器翻译系统而言，输入的是已经切分好的单词序列，而不是原始的字符串（图3.3）。比如，对于一个中文句子，单词之间是没有间隔的，因此需要把一个个的单词切分出来，这样机器翻译系统可以区分不同的翻译单元。甚至，可以对语言学上的单词进行进一步切分，得到词片段序列（比如：中国人→中国/人）。广义上，可以把上述过程看作是一种分词过程，即：将一个输入的自然语言字符串切割成单元序列，每个**单元**（Token）都对应可以处理的最小单位。



图 3.3 一个简单的预处理流程

分词得到的单元序列既可以是语言学上的词序列，也可以是根据其他方式定义的基本处理单元。在本章中，把分词得到的一个个单元称为单词或词，尽管这些单元可以不是语言学上的完整单词，而这个过程也被称作**词法分析**（Lexical Analysis）。除了汉语，词法分析在日语、泰语等单词之间无明确分割符的语言中有着广泛的应

用，芬兰语、维吾尔语等一些形态学十分丰富的语言也需要使用词法分析来解决复杂的词尾、词缀变化等形态学变化。

在机器翻译中，分词系统的好坏往往会影响译文的质量。分词的目的是定义系统处理的基本单元，那么什么叫做“词”呢？关于词的定义有很多，比如：

### 定义 3.2.1 词

语言里最小的可以独立运用的单位。

——《新华字典》

单词，含有语义内容或语用内容，且能被单独念出来的的最小单位。

——维基百科

语句中具有完整概念，能独立自由运用的基本单位。

——《国语辞典》

从语言学的角度来看，人们普遍认为词是可以单独运用的、包含意义的基本单位。这样可以使用有限的词组合出无限的句子，这也正体现出自然语言的奇妙之处。不过，机器翻译并不仅仅局限于语言学定义的单词。比如，神经机器翻译中广泛使用的 BPE 子词切分方法，可以被理解为将词的一部分切分出来，将得到的词片段送给机器翻译系统使用。比如，对如下英语字符串，可以得到切分结果：

Interesting → Interest/ing	selection → se/lect/ion	procession → pro/cess/ion
Interested → Interest/ed	selecting → se/lect/ing	processing → pro/cess/ing
Interests → Interest/s	selected → se/lect/ed	processed → pro/cess/ed

词法分析的重要性在自然语言处理领域已经有共识。如果切分的颗粒度很大，获得单词的歧义通常比较小，比如“中华人民共和国”整体作为一个单词不存在歧义，而如果单独的一个单词“国”，可能会代表“中国”、“美国”等不同的国家，存在歧义。但是随着切分颗粒度的增大，特定单词出现的频次也随之降低，低频词容易和噪音混淆，系统很难对其进行学习。因此，处理这些问题并开发适合翻译任务的分词系统是机器翻译的第一步。

### 3.2.1 基于词典的分词方法

计算机并不能像人类一样在概念上理解“词”，因此需要使用其他方式让计算机“学会”如何分词。一个最简单的方法就是给定一个词典，在这个词典中出现的汉字组合就是所定义的“词”。也就是说，可以通过一个词典定义一个标准，符合这个标准定义的字符串都是合法的“词”。

在使用基于词典的分词方法时，只需预先加载词典到计算机中，扫描输入句子，查询其中的每个词串是否出现在词典中。如图3.4所示，有一个包含六个词的词典，给

定输入句子“确实现在物价很高”后，分词系统自左至右遍历输入句子的每个字，发现词串“确实”在词典中出现，说明“确实”是一个“词”。之后，重复这个过程。

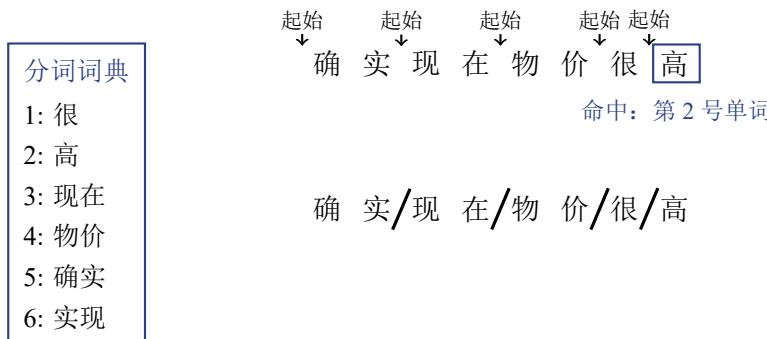


图 3.4 基于词典进行分词的实例

但是，基于词典的分词方法很“硬”。这是因为自然语言非常灵活，经常出现歧义。图3.5就给出了上面例子中的交叉型歧义，从词典中查看，“实现”和“现在”都是合法的单词，但是在句子中二者有重叠，因此词典无法告诉系统哪个结果是正确的。

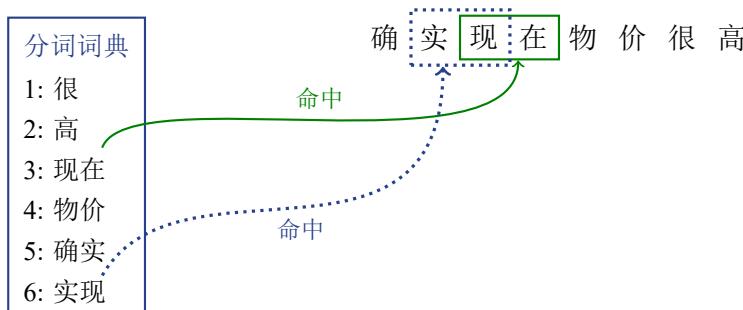


图 3.5 交叉型分词歧义

类似的例子在生活中也很常见。再比如“答辩结束的和尚未答辩的同学都请留在教室”一句中，正常的分词结果是“答辩/结束/的/和/尚未/答辩/的/同学/都/请/留在/教室”，但是由于“尚未”、“和尚”都是常见词汇，使用基于词典的分词方法在这时很容易出现切分错误。

基于词典的分词方法是典型的基于规则的方法，完全依赖于人工给定的词典。在遇到歧义时，需要人工定义消除歧义的规则，比如，可以自左向右扫描每次匹配最长的单词，这是一种简单的启发式消歧策略。图3.4中的例子实际上就是使用这种策略得到的分词结果。但是，启发式的消歧方法仍然需要人工设计启发式规则，而且启发式规则也不能处理所有的情况。所以说简单的基于词典的方法还不能很好的解决分词问题。

### 3.2.2 基于统计的分词方法

既然基于词典的方法有很多问题，那么就需要一种更为有效的方法。在上文中提到，想要搭建一个分词系统，需要让计算机知道什么是“词”，那么可不可以给出已经切分好的分词数据，让计算机在这些数据中学习到规律呢？答案是肯定的，利用“数据”来让计算机明白“词”的定义，让计算机直接在数据中学到知识，这就是一个典型的基于统计建模的学习过程。

#### 1. 统计模型的学习与推断

统计分词也是一种典型的数据驱动方法。这种方法将已经经过分词的数据“喂”给系统，这个数据也被称作**标注数据**（Annotated Data）。在获得标注数据后，系统自动学习一个统计模型来描述分词的过程，而这个模型会把分词的“知识”作为参数保存在模型中。当送入一个新的需要分词的句子时，可以利用学习到的模型对可能的分词结果进行概率化的描述，最终选择概率最大的结果作为输出。这个方法就是基于统计的分词方法，其与第二章介绍的统计语言建模方法本质上是一样的。具体来说，可以分为两个步骤：

- **训练**。利用标注数据，对统计模型的参数进行学习。
- **预测**。利用学习到的模型和参数，对新的句子进行切分。这个过程也可以被看作是利用学习到的模型在新的数据上进行推断。

图3.6给出了一个基于统计建模的汉语分词实例。左侧是标注数据，其中每个句子是已经经过人工标注的分词结果（单词用斜杠分开）。之后，建立一个统计模型，记为 $P(\cdot)$ 。模型通过在标注数据上的学习来对问题进行描述，即学习 $P(\cdot)$ 。最后，对于新的未分词的句子，使用模型 $P(\cdot)$ 对每个可能的切分方式进行概率估计，之后选择概率最高的切分结果输出。

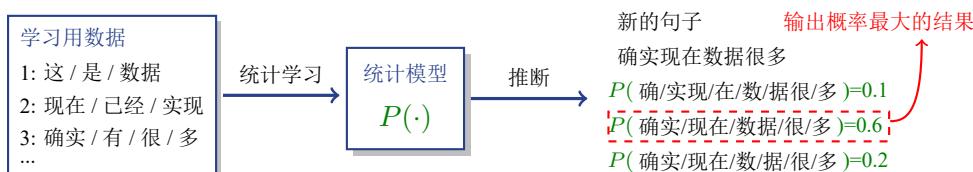


图 3.6 基于统计的自动分词流程

#### 2. 全概率分词方法

上述过程的核心在于从标注数据中学习一种对分词现象的统计描述，即句子的分词结果概率 $P(\cdot)$ 。如何让计算机利用分好词的数据学习到分词知识呢？本书的第二章曾介绍如何对单词概率进行统计建模，而对分词现象的统计描述就是在单词概率的基础上，基于独立性假设获取的<sup>1</sup>。虽然独立性假设并不能完美描述分词过程中

<sup>1</sup>即假定所有词的出现都是相互独立的。

单词之间的关系，但是它大大简化了分词问题的复杂度。

如图3.7所示，可以利用大量人工标注好的分词数据，通过统计学习方法获得一个统计模型  $P(\cdot)$ ，给定任意分词结果  $W = w_1 w_2 \dots w_m$ ，都能通过  $P(W) = P(w_1) \cdot P(w_2) \cdot \dots \cdot P(w_m)$  计算这种切分的概率值。

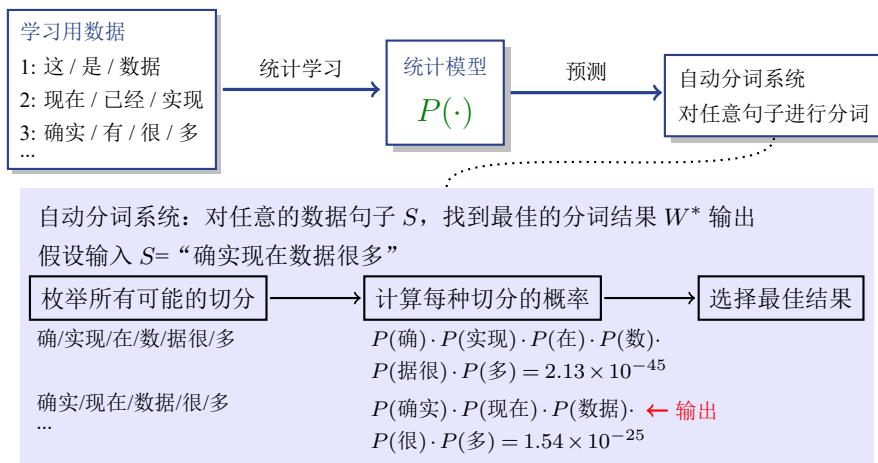


图 3.7 基于 1-gram 语言模型的中文分词实例

以“确实现在数据很多”这个实例来说，如果把这句话按照“确实/现在/数据/很/多”这样的方式进行切分，这个句子切分的概率  $P(\text{确实}/\text{现在}/\text{数据}/\text{很}/\text{多})$  可以通过每个词出现概率相乘的方式进行计算。

$$\begin{aligned} & P(\text{确实}/\text{现在}/\text{数据}/\text{很}/\text{多}) \\ &= P(\text{确实}) \cdot P(\text{现在}) \cdot P(\text{数据}) \cdot P(\text{很}) \cdot P(\text{多}) \end{aligned} \quad (3.1)$$

经过充分训练的统计模型  $P(\cdot)$  就是我们所说的分词模型。对于输入的新句子  $S$ ，通过这个模型找到最佳的分词结果输出。假设输入句子  $S$  是“确实现在数据很多”，可以通过列举获得不同切分方式的概率，其中概率最高的切分方式，就是系统的目标输出。

这种分词方法也被称作基于 1-gram 语言模型的分词，或全概率分词<sup>[90, 91]</sup>。全概率分词最大的优点在于方法简单、效率高，因此被广泛应用在工业界系统里。它本质上就是一个 1-gram 语言模型，因此可以直接复用  $n$ -gram 语言模型的训练方法和未登录词处理方法。与传统  $n$ -gram 语言模型稍有不同的是，分词的预测过程需要找到一个在给定字符串所有可能切分中 1-gram 语言模型得分最高的切分。因此，可以使用第二章中所描述的搜索算法实现这个预测过程，也可以使用动态规划方法<sup>[92]</sup> 快速找到最优切分结果。由于本节的重点是介绍中文分词的基础方法和统计建模思想，因此不会对相关搜索算法进行进一步介绍，有兴趣的读者可以参考第二章和本章3.5节的相关文献做进一步深入研究。

### 3.3 命名实体识别

在人类使用语言的过程中，单词往往不是独立出现的。很多时候，多个单词会组合成一个更大的单元来表达特定的意思。其中，最典型的代表是**命名实体**（Named Entity）。通常，命名实体是指名词性的专用短语，例如公司名称、品牌名称、产品名称等专有名词和行业术语。准确地识别出这些命名实体，是提高机器翻译质量的关键。比如，在翻译技术文献时，往往需要对术语进行识别并进行准确翻译，因此引入**命名实体识别**（Named Entity Recognition）可以帮助系统对特定术语进行更加细致的处理。

从句法分析的角度来说，命名实体识别是一种浅层句法分析任务。它在分词的基础上，进一步对句子浅层结构进行识别，包括词性标注、组块识别在内的很多任务都可以被看作是浅层句法分析的内容。本节会以命名实体识别为例，对基于序列标注的浅层句法分析方法进行介绍。

#### 3.3.1 序列标注任务

命名实体识别是一种典型的**序列标注**（Sequence Labeling）任务，对于一个输入序列，它会生成一个相同长度的输出序列。输入序列的每一个位置，都有一个与之对应的输出，输出的内容是这个位置所对应的标签（或者类别）。比如，对于命名实体识别，每个位置的标签可以被看作是一种命名实体“开始”和“结束”的标志，而命名实体识别的目标就是得到这种“开始”和“结束”标注的序列。不仅如此，分词、词性标注、组块识别等也都可以被看作是序列标注任务。

通常来说，序列标注任务中首先需要定义标注策略，即使用什么样的格式来对序列进行标注。为了便于描述，这里假设输入序列为一个个单词<sup>2</sup>。常用的标注策略有：

- **BIO 格式**（Beginning-inside-outside）。以命名实体识别为例，B 代表一个命名实体的开始，I 表示一个命名实体的其它部分，O 表示一个非命名实体单元。
- **BIOES 格式**。与 BIO 格式相比，多出了标签 E（End）和 S（Single）。仍然以命名实体识别为例，E 和 S 分别用于标注一个命名实体的结束位置和仅含一个单词的命名实体。

图3.8给出了不同标注格式所对应的标注结果。可以看出文本序列中的非命名实体直接被标注为“O”，而命名实体的标注则被分为了两部分：位置和命名实体类别，图中的“B”、“I”、“E”等标注出了位置信息，而“CIT”和“CNT”则标注出了命名实体类别（“CIT”表示城市，“CNT”表示国家）。可以看到，命名实体的识别结果可以通过 BIO、BIOES 这类序列标注结果归纳出来：例如在 BIOES 格式中，标签“B-CNT”后面的标签只会是“I-CNT”或“E-CNT”，而不会是其他的标签。同时，

<sup>2</sup>广义上，序列标注任务并不限制输入序列的形式，比如，字符、单词、多个单词构成的词组都可以作为序列标注的输入单元。

在命名实体识别任务中涉及到实体边界确定，而“BIO”或“BIOES”的标注格式本身就暗含着边界问题：在“BIO”格式下，实体左边界只能在“B”的左侧，右边界只能在“B”或“I”的右侧；在“BIOES”格式下，实体左边界只能在“B”或“S”的左侧，右边界只能在“E”和“S”的右侧。

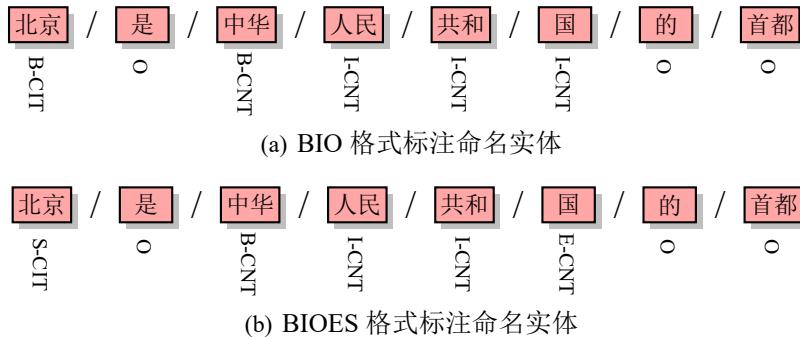


图 3.8 BIO 和 BIOES 格式对比

需要注意的是，虽然图3.8中的命名实体识别以单词为基本单位进行标注，但真实系统中也可以在字序列上进行命名实体识别，其方法与基于词序列的命名实体识别是一样的。因此，这里仍然以基于词序列的方法为例进行介绍。

对于像命名实体识别这样的任务，早期的方法主要是基于词典和规则的方法。这些方法依赖于人工构造的识别规则，通过字符串匹配的方式识别出文本中的命名实体<sup>[93, 94, 95]</sup>。严格意义上来说，那时命名实体识别还并没有被看作是一种序列标注问题。

序列标注这个概念更多的是出现在基于统计建模的方法中。许多统计机器学习方法都被成功应用于命名实体识别任务，例如**隐马尔可夫模型**（Hidden Markov Model, HMM）<sup>[96]</sup>、**条件随机场**（Conditional Random Fields, CRFs）<sup>[97]</sup>、**最大熵**（Maximum Entropy, ME）模型<sup>[98]</sup> 和**支持向量机**（Support Vector Machine, SVM）<sup>[99]</sup> 等。此外，近些年深度学习的兴起也给命名实体识别带来了新的思路<sup>[100]</sup>。而命名实体识别也成为了验证机器学习方法有效性的重要任务之一。本节将对序列标注中几类基础的方法进行介绍。其中会涉及概率图模型、统计分类模型等方法。特别是统计分类的概念，在后续章节中也会被使用到。

### 3.3.2 基于特征的统计学习

基于特征的统计学习是解决序列标注的有效方法之一。这种方法中，系统研发人员通过定义不同的特征来完成对问题的描述，之后利用统计模型完成对这些特征的某种融合，并得到最终的预测结果。

在开始介绍序列标注模型之前，先来看一下统计学习所涉及的重要概念——**特征**（Feature）。简单来说，特征是指能够反映事物在某方面表现或行为的一种属性，如现实生活中小鸟的羽毛颜色、喙的形状、翼展长度等就是小鸟的特征；命名实体识别任务中的每个词的词根、词性和上下文组合也可以被看做是识别出命名实体可

以采用的特征。

从统计建模的角度看，特征的形式可以非常灵活。比如，可以分为连续型特征和离散型特征，前者通常用于表示取值蕴含数值大小关系的信息，如人的身高和体重，后者通常用于表示取值不蕴含数值大小关系的信息，如人的性别。正是由于这种灵活性，系统开发者可以通过定义多样的特征来从多个不同的角度对目标问题进行建模。而这种设计特征的过程也被称作**特征工程**（Feature Engineering）。

设计更好的特征也成为了很多机器学习方法的关键。通常有两个因素需要进行考虑：

- **样本在这些特征上的差异度**，即特征对于样本的区分能力。比如，可以考虑优先选择样本特征值方差较大即区分能力强的特征<sup>3</sup>；
- **特征与任务目标的相关性**。优先选择相关性高的特征。

回到命名实体识别任务上来。对于输入的每个单词，可以将其表示为一个单词和对应的**词特征**（Word Feature）的组合，记作 $\langle w, f \rangle$ 。通过这样的表示，就可以将原始的单词序列转换为词特征序列。命名实体识别中的特征可以分为两大类，一种是单词对应各个标签的特征，另一种是标签之间组合的特征。常用的特征包括词根、词缀、词性或者标签的固定搭配等。表3.1展示了命名实体识别任务中一些典型的特征。

表 3.1 命名实体识别中常用的特征

特征名	示例文本	释义
LocSuffix	沈阳市	地名后缀
FourDigitYear	2020	四位数年份
OtherDigit	202020	其他数字
NamePrefix	张三	姓名前缀
ShortName	东大 成立 120 周年	缩略词

在相当长的一段时期内，基于特征工程的方法都是自然语言处理领域的主流范式。虽然深度学习技术的进步使得系统研发人员可以逐步摆脱繁重的特征设计工作，但是很多传统的模型和方法在今天仍然被广泛使用。比如，在当今最先进的序列标注模型中<sup>[101]</sup>，条件随机场模型仍然是一个主要部件，本节即将对其进行介绍。

### 3.3.3 基于概率图模型的方法

**概率图模型**（Probabilistic Graphical Model）是使用图表示变量及变量间概率依赖关系的方法。在概率图模型中，可以根据可观测变量推测出未知变量的条件概率分

<sup>3</sup> 方差如果很小，意味着样本在这个特征上基本上没有差异，那么这个特征对于样本的区分并没有什么用。

布等信息。如果把序列标注任务中的输入序列看作观测变量，而把输出序列看作需要预测的未知变量，那么就可以把概率图模型应用于命名实体识别等序列标注任务。

## 1. 隐马尔可夫模型

隐马尔可夫模型是一种经典的序列模型<sup>[96, 102, 103]</sup>。它在语音识别、自然语言处理的很多领域得到了广泛的应用。隐马尔可夫模型的本质就是概率化的马尔可夫过程，这个过程隐含着状态间转移和可见状态生成的概率。

这里用一个简单的“抛硬币”游戏来对这些概念进行说明：假设有三枚质地不同的硬币  $A$ 、 $B$ 、 $C$ ，已知这三个硬币抛出正面的概率分别为 0.3、0.5、0.7，在游戏中，游戏发起者在上述三枚硬币中选择一枚硬币上抛，每枚硬币被挑选到的概率可能会受上次被挑选的硬币的影响，且每枚硬币正面向上的概率都各不相同。不停的重复挑选硬币、上抛硬币的过程，会得到一串硬币的正反序列，例如：抛硬币 6 次，得到：正正反反正反。游戏挑战者通过观察 6 次后获得的硬币正反序列，猜测每次选择的究竟是哪一枚硬币。

在上面的例子中，每次挑选并上抛硬币后得到的“正面”或“反面”即为“可见状态”，再次挑选并上抛硬币会获得新的“可见状态”，这个过程即为“状态的转移”，经过 6 次反复挑选上抛后得到的硬币正反序列叫做可见状态序列，由每个回合的可见状态构成。此外，在这个游戏中还暗含着一个会对最终“可见状态序列”产生影响的“隐含状态序列”——每次挑选的硬币形成的序列，例如  $CBABCA$ 。

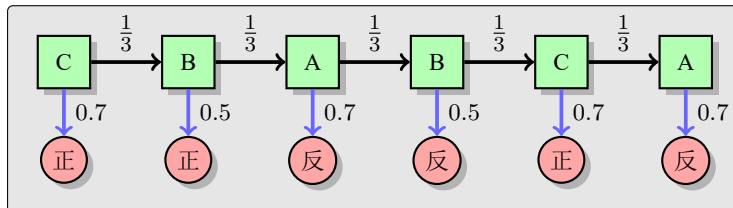
实际上，隐马尔科夫模型在处理序列问题时的关键依据是两个至关重要的概率关系，并且这两个概率关系也始终贯穿于“抛硬币”的游戏中。一方面，隐马尔可夫模型中用**发射概率**（Emission Probability）来描述隐含状态和可见状态之间存在的输出概率（即  $A$ 、 $B$ 、 $C$  抛出正面的输出概率为 0.3、0.5、0.7），同样的，隐马尔可夫模型还会描述系统隐含状态的**转移概率**（Transition Probability），在这个例子中， $A$  的下一个状态是  $A$ 、 $B$ 、 $C$  的概率都是  $1/3$ ， $B$ 、 $C$  的下一个状态是  $A$ 、 $B$ 、 $C$  的转移概率也同样是  $1/3$ 。图3.9展示了在“抛硬币”游戏中的转移概率和发射概率，它们都可以被看做是条件概率矩阵。

		转移概率 $P(\text{第 } i+1 \text{ 次}   \text{第 } i \text{ 次})$			发射概率 $P(\text{可见状态}   \text{隐含状态})$		
		硬币 A	硬币 B	硬币 C	可见 隐含	正面	反面
第 $i$ 次	第 $i+1$ 次	硬币 A	硬币 B	硬币 C	硬币 A	0.3	0.7
	硬币 A	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	硬币 B	0.5	0.5
	硬币 B	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	硬币 C	0.7	0.3
第 $i$ 次	第 $i+1$ 次	硬币 A	硬币 B	硬币 C	硬币 A	0.3	0.7
	硬币 B	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	硬币 B	0.5	0.5
	硬币 C	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	硬币 C	0.7	0.3

图 3.9 “抛硬币”游戏中的转移概率和发射概率

由于隐含状态序列之间存在转移概率，并且隐马尔可夫模型中隐含状态和可见

状态之间存在着发射概率，因此根据可见状态的转移猜测隐含状态序列并非无迹可循。图3.10描述了如何使用隐马尔可夫模型来根据“抛硬币”结果推测挑选的硬币序列。可见，通过隐含状态之间的联系（绿色方框及它们之间的连线）可以对有序的状态进行描述，进而得到隐含状态序列所对应的可见状态序列（红色圆圈）。



图示说明：

一个隐含状态

→ 从一个隐含状态到下一个隐含状态的转换，该过程隐含着转移概率

一个可见状态

↓ 从一个隐含状态到可见状态的输出，该过程隐含着发射概率

图 3.10 抛硬币的隐马尔可夫模型实例

从统计建模的角度看，上述过程本质上是在描述隐含状态和可见状态出现的联合概率。这里，用  $x = (x_1, \dots, x_m)$  表示可见状态序列，用  $y = (y_1, \dots, y_m)$  表示隐含状态序列。（一阶）隐马尔可夫模型假设：

- 当前位置的隐含状态仅与前一个位置的隐含状态相关，即  $y_i$  仅与  $y_{i-1}$  相关；
- 当前位置的可见状态仅与当前位置的隐含状态相关，即  $x_i$  仅与  $y_i$  相关。

于是，联合概率  $P(x, y)$  可以被定义为：

$$\begin{aligned}
 P(x, y) &= P(x|y)P(y) \\
 &= P(x_1, \dots, x_m | y_1, \dots, y_m)P(y_1, \dots, y_m) \\
 &= \prod_{i=1}^m P(x_i | x_1, \dots, x_{i-1}, y_1, \dots, y_m) \prod_{i=1}^m P(y_i | y_{i-1}) \\
 &= \prod_{i=1}^m P(x_i | y_i) \prod_{i=1}^m P(y_i | y_{i-1}) \\
 &= \prod_{i=1}^m P(x_i | y_i)P(y_i | y_{i-1})
 \end{aligned} \tag{3.2}$$

这里， $y_0$  表示一个虚拟的隐含状态。这样，可以定义  $P(y_1 | y_0) \equiv P(y_1)$ ，它表示起始隐含状态出现的概率。隐马尔可夫模型的假设也大大简化了问题，因此可以通过式(3.2)很容易地计算隐含状态序列和可见状态序列出现的概率。值得注意的是，发射概率和转移概率都可以被看作是描述序列生成过程的“特征”。但是，这些“特征”

并不是随意定义的，而是符合问题的概率解释。而这种基于事件发生的逻辑所定义的概率生成模型，通常可以被看作是一种**生成式模型**（Generative Model）。

一般来说，隐马尔可夫模型中包含下面三个问题：

- **隐含状态序列的概率计算**，即给定模型（转移概率和发射概率），根据可见状态序列（抛硬币的结果）计算在该模型下得到这个结果的概率，这个问题的求解需要用到前后向算法<sup>[103]</sup>。
- **参数学习**，即给定硬币种类（隐含状态数量），根据多个可见状态序列（抛硬币的结果）估计模型的参数（转移概率），这个问题的求解需要用到 EM 算法<sup>[104]</sup>。
- **解码**，即给定模型（转移概率和发射概率）和可见状态序列（抛硬币的结果），计算在可见状态序列的情况下，最可能出现的对应的状态序列，这个问题的求解需要用到基于动态规划的方法，通常也被称作**维特比算法**（Viterbi Algorithm）<sup>[105]</sup>。

隐马尔可夫模型处理序列标注问题的基本思路是：

- 第一步：根据可见状态序列（输入序列）和其对应的隐含状态序列（标记序列）样本，估算模型的转移概率和发射概率；
- 第二步：对于给定的可见状态序列，预测概率最大的隐含状态序列，比如，根据输入的词序列预测最有可能的命名实体标记序列

一种简单的办法是使用相对频次估计得到转移概率和发射概率估计值。令  $x_i$  表示第  $i$  个位置的可见状态， $y_i$  表示第  $i$  个位置的隐含状态， $P(y_i|y_{i-1})$  表示第  $i-1$  个位置到第  $i$  个位置的状态转移概率， $P(x_i|y_i)$  表示第  $i$  个位置的发射概率，于是有：

$$P(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \quad (3.3)$$

$$P(x_i|y_i) = \frac{c(x_i, y_i)}{c(y_i)} \quad (3.4)$$

其中， $c(\cdot)$  统计训练集中某种现象出现的次数。

在获得转移概率和发射概率的基础上，对于一个句子进行命名实体识别可以被描述为：在观测序列  $x$ （可见状态，即输入的词序列）的条件下，最大化标签序列  $y$ （隐含状态，即标记序列）的概率，即：

$$\hat{y} = \arg \max_y P(y|x) \quad (3.5)$$

根据贝叶斯定理，该概率被分解为  $P(y|x) = \frac{P(x,y)}{P(x)}$ ，其中  $P(x)$  是固定概率，因为  $x$  在这个过程中是确定的不变量。因此只需考虑如何求解分子，即将求条件概率

$P(y|x)$  的问题转化为求联合概率  $P(y,x)$  的问题:

$$\hat{y} = \arg \max_y P(x,y) \quad (3.6)$$

将式(3.2)带入式(3.6)可以得到最终计算公式, 如下:

$$\hat{y} = \arg \max_y \prod_{i=1}^m P(x_i|y_i)P(y_i|y_{i-1}) \quad (3.7)$$

图3.11展示了基于隐马尔可夫模型的命名实体识别模型。实际上, 这种描述序列生成的过程也可以被应用于机器翻译, 在第五章还将看到隐马尔可夫模型在翻译建模中的应用。

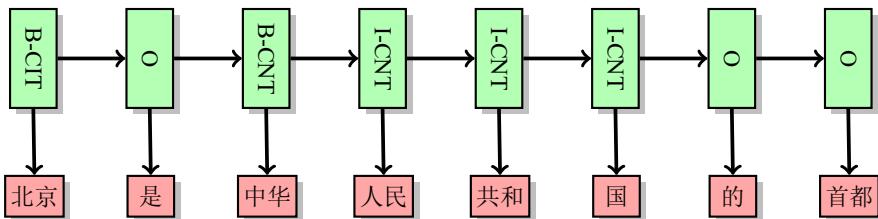


图 3.11 基于隐马尔可夫模型的命名实体识别

## 2. 条件随机场

隐马尔可夫模型有一个很强的假设: 一个隐含状态出现的概率仅由上一个隐含状态决定。这个假设也会带来一些问题, 举个例子: 在某个隐马尔可夫模型中, 隐含状态集合为  $\{A, B, C, D\}$ , 可见状态集合为  $\{T, F\}$ , 其中隐含状态  $A$  可能的后继隐含状态集合为  $\{A, B\}$ , 隐含状态  $B$  可能的后继隐含状态集合为  $\{A, B, C, D\}$ , 于是有:

$$P(A|A) + P(A|B) = 1 \quad (3.8)$$

$$P(A|B) + P(B|B) + P(C|B) + P(D|B) = 1 \quad (3.9)$$

其中,  $P(b|a)$  表示由状态  $a$  转移到状态  $b$  的概率, 由于式(3.8)中的分式数量少于式(3.9), 这就导致在统计中获得的  $P(A|A)、P(A|B)$  的值很可能会比  $P(A|B)、P(B|B)、P(C|B)、P(D|B)$  要大。

图3.12展示了一个具体的例子, 有一个可见状态序列  $TFFT$ , 假设初始隐含状态是  $A$ , 图中线上的概率值是对应的转移概率与发射概率的乘积, 比如图中隐含状态  $A$  开始, 下一个隐含状态是  $A$  且可见状态是  $F$  的概率是 0.45, 下一个隐含状态是  $B$  且可见状态是  $F$  的概率是 0.55。图中可以看出, 由于有较大的值, 当可见状态序列为  $TFFT$  时, 隐马尔可夫计算出的最有可能的隐含状态序列为  $AAAA$ 。但是如果对训练集进行统计可能会发现, 当可见序列为  $TFFT$  时, 对应的隐含状态是  $AAAA$

的概率可能是比较大的，但也可能是比较小的。这个例子中出现预测偏差的主要原因是：由于比其他状态转移概率要大得多，隐含状态的预测一直停留在状态  $A$ 。

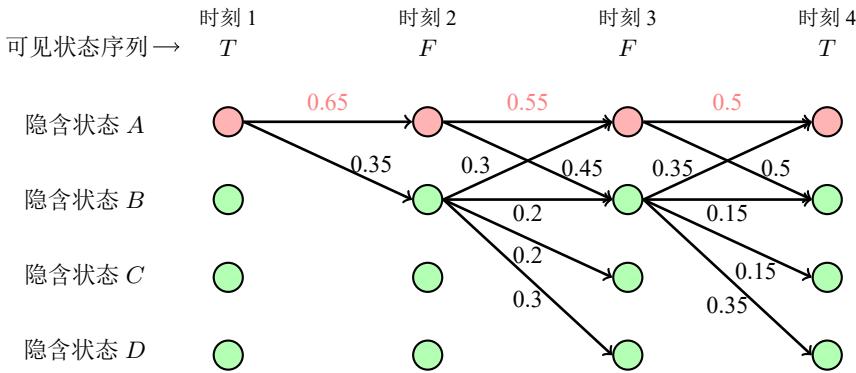


图 3.12 隐马尔可夫实例

上述现象也被称作**标注偏置**（Label Bias）。条件随机场模型在隐马尔可夫模型的基础上，解决了这个问题<sup>[97]</sup>。在条件随机场模型中，以全局范围的统计归一化代替了隐马尔可夫模型中的局部归一化。除此之外，条件随机场模型中并非使用概率计算而是特征函数的方式对可见状态序列  $x$  对应的隐含状态序列  $y$  的概率进行计算。

条件随机场中一般有若干个特征函数，都是经过设计的、能够反映序列规律的一些二元函数<sup>4</sup>，并且每个特征函数都有其对应的权重  $\lambda$ 。特征函数一般由两部分组成：能够反映隐含状态序列之间转移规则的转移特征  $t(y_{i-1}, y_i, x, i)$  和状态特征  $s(y_i, x, i)$ 。其中  $y_i$  和  $y_{i-1}$  分别是位置  $i$  和前一个位置的隐含状态， $x$  则是可见状态序列。转移特征  $t(y_{i-1}, y_i, x, i)$  反映了两个相邻的隐含状态之间的转换关系，而状态特征  $s(y_i, x, i)$  则反映了第  $i$  个可见状态应该对应什么样的隐含状态，这两部分共同组成了一个特征函数  $F(y_{i-1}, y_i, x, i)$ ，即

$$F(y_{i-1}, y_i, x, i) = t(y_{i-1}, y_i, x, i) + s(y_i, x, i) \quad (3.10)$$

实际上，基于特征函数的方法更像是对隐含状态序列的一种打分：根据人为设计的模板（特征函数），测试隐含状态之间的转换以及隐含状态与可见状态之间的对应关系是否符合这种模板。在处理序列问题时，假设可见状态序列  $x$  的长度和待预测隐含状态序列  $y$  的长度均为  $m$ ，且共设计了  $k$  个特征函数，则有：

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^m \sum_{j=1}^k \lambda_j F_j(y_{i-1}, y_i, x, i)\right) \quad (3.11)$$

公式(3.11)中的  $Z(x)$  即为上面提到的实现全局统计归一化的归一化因子，其计

<sup>4</sup>二元函数的函数值一般非 1 即 0

算方式为：

$$Z(x) = \sum_y \exp\left(\sum_{i=1}^m \sum_{j=1}^k \lambda_j F_j(y_{i-1}, y_i, x, i)\right) \quad (3.12)$$

由公式(3.12)可以看出，归一化因子的求解依赖于整个可见状态序列和每个位置的隐含状态，因此条件随机场模型中的归一化是一种全局范围的归一化方式。图3.13为条件随机场模型处理序列问题的示意图。

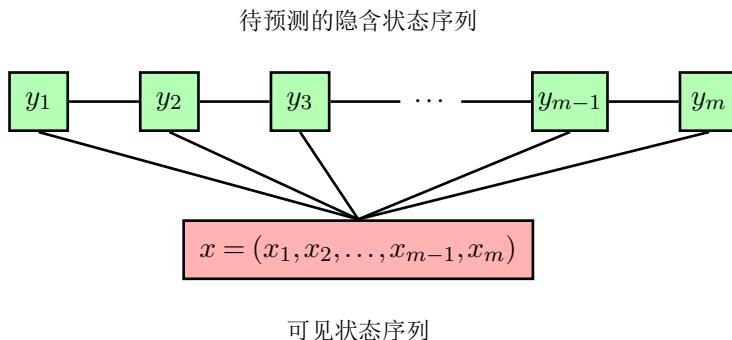


图 3.13 条件随机场模型处理序列问题

虽然，式(3.11)和式(3.12)的表述相较于隐马尔可夫模型更加复杂，但是其实现有非常高效的方式。比如，可以使用动态规划方法完成整个条件随机场模型的计算<sup>[97]</sup>。

条件随机场模型处理命名实体识别任务时，可见状态序列对应着文本内容，隐含状态序列对应着待预测的标签。对于命名实体识别任务，需要单独设计若干适合命名实体识别任务的特征函数。例如在使用 BIOES 标准标注命名实体识别任务时，标签“B-ORG”<sup>5</sup>后面的标签必然是“I-ORG”或是“E-ORG”，而不可能是“O”，针对此规则可以设计相应特征函数。

### 3.3.4 基于分类器的方法

基于概率图的模型将序列表示为有向图或无向图，如图3.14(a)、(b)所示。这种方法增加了建模的复杂度。既然要得到每个位置的类别输出，另一种更加直接的方法是使用分类器对每个位置进行独立预测。分类器是机器学习中广泛使用的方法，它可以根据输入自动地对类别进行预测。如图3.14(c)所示，对于序列标注任务，分类器把每一个位置所对应的所有特征看作是输入，而把这个位置对应的标签看作输出。从这个角度说，隐马尔可夫模型等方法实际上也是在进行一种“分类”操作，只不过这些方法考虑了不同位置输出（或隐含状态）之间的依赖。

值得注意的是分类模型可以被应用于序列标注之外的很多任务，在后面的章节中还会看到，机器翻译中的很多模块也借鉴了统计分类的思想。其中使用到的基础

<sup>5</sup>ORG 表示机构实体

数学模型和特征定义形式，与这里提到的分类器本质上是一样的。

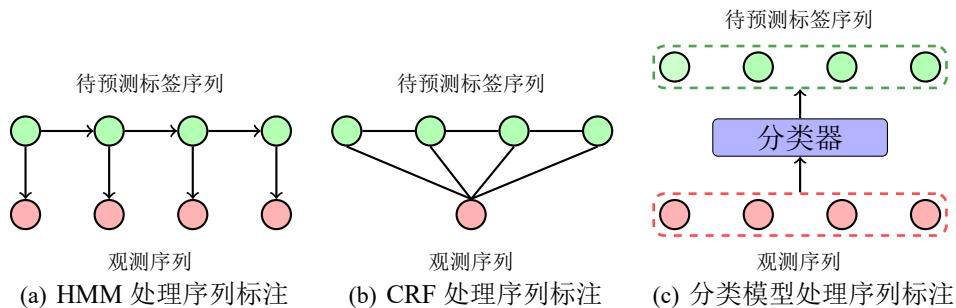


图 3.14 HMM、CRF、分类算法三种方法对比

## 1. 分类任务与分类器

无论在日常生活中还是在研究工作中，都会遇到各种各样的分类问题，例如挑选西瓜时需要区分“好瓜”和“坏瓜”、编辑看到一篇新闻稿件时要对稿件进行分门别类。事实上，在机器学习中，对“分类任务”的定义会更宽泛而并不拘泥于“类别”的概念，在对样本进行预测时，只要预测标签集合是有限的且预测标签是离散的，就可认定其为分类任务。

具体来说，分类任务目标是训练一个可以根据输入数据预测离散标签的**分类器**（Classifier），也可称为分类模型。在有监督的分类任务中<sup>6</sup>，训练数据集合通常由形似 $(\mathbf{x}^{[i]}, y^{[i]})$ 的带标注数据构成， $\mathbf{x}^{[i]} = (x_1^{[i]}, \dots, x_k^{[i]})$ 作为分类器的输入数据（通常被称为一个训练样本），其中 $x_j^{[i]}$ 表示样本 $\mathbf{x}^{[i]}$ 的第 $j$ 个特征； $y^{[i]}$ 作为输入数据对应的**标签**（Label），反映了输入数据对应的“类别”。若标签集合大小为 $n$ ，则分类任务的本质是通过对训练数据集合的学习，建立一个从 $k$ 维样本空间到 $n$ 维标签空间的映射关系。更确切地说，分类任务的最终目标是学习一个条件概率分布 $P(y|\mathbf{x})$ ，这样对于输入 $\mathbf{x}$ 可以找到概率最大的 $y$ 作为分类结果输出。

与概率图模型一样，分类模型中也依赖特征定义。其定义形式与3.3.2节的描述一致，这里不再赘述。分类任务一般根据类别数量分为二分类任务和多分类任务，二分类任务是最经典的分类任务，只需要对输出进行非零即一的预测。多分类任务则可以有多种处理手段，比如，可以将其“拆解”为多个二分类任务求解，或者直接让模型输出多个类别中的一个。在命名实体识别中，往往使用多类别分类模型。比如，在 BIO 标注下，有三个类别（B、I 和 O）。一般来说，类别数量越大分类的难度也越大。比如，BIOES 标注包含 5 个类别，因此使用同样的分类器，它要比 BIO 标注下的分类问题难度大。此外，更多的类别有助于准确的刻画目标问题。因此在实践中需要在类别数量和分类难度之间找到一种平衡。

<sup>6</sup>与之相对应的，还有无监督、半监督分类任务，不过这些内容不是本书讨论的重点。读者可以参看参考文献<sup>[32, 33]</sup>对相关概念进行了解。

在机器翻译和语言建模中也会遇到类似的问题，比如，生成单词的过程可以被看做是一个分类问题，类别数量就是词表的大小。显然，词表越大可以覆盖更多的单词和更多种类的单词形态学变化，但是过大的词表里会包含很多低频词，其计算复杂度会显著增加。然而，过小的词表又无法包含足够多的单词。因此，在设计这类系统的时候对词表大小的选择（类别数量的选择）是十分重要的，往往要通过大量的实验得到最优的设置。

## 2. 经典的分类模型

经过多年的发展，研究者提出了很多分类模型。由于篇幅所限，本书无法一一列举这些模型，这里仅列出了部分经典的模型。关于分类模型更全面的介绍可以参考相关文献<sup>[33, 106]</sup>。

- ***K*-近邻分类算法**。*K*-近邻分类算法通过计算不同特征值之间的距离进行分类，这种方法适用于可以提取到数值型特征<sup>7</sup>的分类问题。该方法的基本思想为：将提取到的特征分别作为坐标轴，建立一个*k*维坐标系（对应特征数量为*k*的情况），此时每个样本都将成为该*k*维空间的一个点，将未知样本与已知类别样本的空间距离作为分类依据进行分类，比如，考虑与输入样本最近的*K*个样本的类别进行分类。
- **支持向量机**。支持向量机是一种二分类模型，其思想是通过线性超平面将不同输入划分为正例和负例，并使线性超平面与不同输入的距离都达到最大。与*K*-近邻分类算法类似，支持向量机也适用于可以提取到数值型特征的分类问题。
- **最大熵模型**。最大熵模型是根据最大熵原理提出的一种分类模型，其基本思想是：以在训练数据集中学习到的经验知识作为一种“约束”，并在符合约束的前提下，在若干合理的条件概率分布中选择“使条件熵最大”的模型。
- **决策树分类算法**。决策树分类算法是一种基于实例的归纳学习方法：将样本中某些决定性特征作为决策树的节点，根据特征表现进行对样本划分，最终根节点到每个叶子节点均形成一条分类的路径规则。这种分类方法适用于可以提取到离散型特征<sup>8</sup>的分类问题。
- **朴素贝叶斯分类算法**。朴素贝叶斯算法是以贝叶斯定理为基础并且假设特征之间相互独立的方法，以特征之间相互独立作为前提假设，学习从输入到输出的联合概率分布，并以后验概率最大的输出作为最终类别。

<sup>7</sup>即可以用数值大小对某方面特征进行衡量。

<sup>8</sup>即特征值是离散的。

## 3.4 句法分析

前面已经介绍了什么叫做“词”以及如何对分词问题进行统计建模。同时，也介绍了如何对多个单词构成的命名实体进行识别。无论是分词还是命名实体识别都是句子浅层信息的一种表示。对于一个自然语言句子来说，它更深层次的结构信息可以通过更完整的句法结构来描述，而句法信息也是机器翻译和自然语言处理其他任务中常用的知识之一。

### 3.4.1 句法树

**句法** (Syntax) 是研究句子的每个组成部分和它们之间的组合方式。一般来说，句法和语言是相关的，比如，英文是主谓宾结构，而日语是主宾谓结构，因此不同的语言也会有不同的句法描述方式。自然语言处理领域最常用的两种句法分析形式是**短语结构分析** (Phrase Structure Parsing) 和**依存分析** (Dependency Parsing)。图3.15展示了这两种的句法表示形式的实例。其中，左侧是短语结构树，它描述的是短语的结构功能，比如“吃”是动词（记为 VV），“鱼”是名词（记为 NN），“吃/鱼”组成动词短语，这个短语再与“喜欢”这一动词组成新的动词短语。短语结构树的每个子树都是一个句法功能单元，比如，子树 VP(VV(吃) NN(鱼)) 就表示了“吃/鱼”这个动词短语的结构，其中子树根节点 VP 是句法功能标记。短语结构树利用嵌套的方式描述了语言学的功能，短语结构树中，每个词都有词性（或词类），不同的词或者短语可以组成名动结构、动宾结构等语言学短语结构，短语结构分析一般也被称为**成分分析** (Constituency Parsing) 或**完全分析** (Full Parsing)。

图3.15右侧展示的是另一种句法结构，被称作依存句法树。依存句法树表示了句子中单词和单词之间的依存关系。比如，从这个例子可以了解，“猫”依赖“喜欢”，“吃”依赖“喜欢”，“鱼”依赖“吃”。

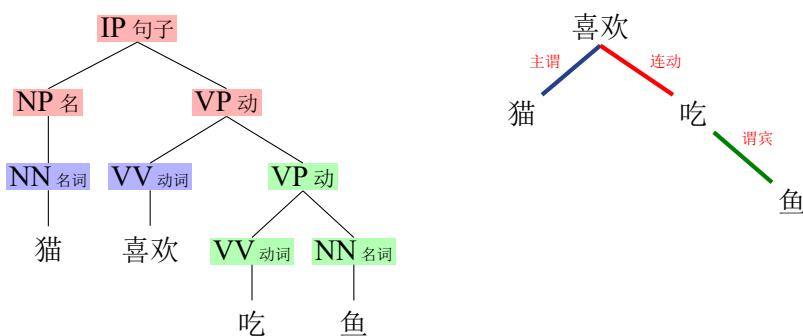


图 3.15 短语结构树 (左) 和依存树 (右)

短语结构树和依存句法树的结构和功能有很大不同。短语结构树的叶子节点是单词，中间节点是词性或者短语句法标记。在短语结构分析中，通常把单词称作**终结符** (Terminal)，把词性称为**预终结符** (Pre-terminal)，而把其他句法标记称为**非终结符** (Non-terminal)。依存句法树没有预终结符和非终结符，所有的节点都是句子里

的单词，通过不同节点间的连线表示句子中各个单词之间的依存关系。每个依存关系实际上都是有方向的，头和尾分别指向“接受”和“发出”依存关系的词。依存关系也可以进行分类，例如，图3.15中的对每个依存关系的类型都有一个标记，这也被称作是有标记的依存分析。如果不生成这些标记，这样的句法分析被称作无标记的依存分析。

虽然短语结构树和依存树的句法表现形式有很大不同，但是它们在某些条件下能相互转化。比如，可以使用启发性规则将短语结构树自动转化为依存树。从应用的角度，依存分析由于形式更加简单，而且直接建模词语之间的依赖，因此在自然语言处理领域中受到很多关注。在机器翻译中，无论是哪种句法树结构，都已经被证明会对机器翻译系统产生帮助。特别是短语结构树，在机器翻译中的应用历史更长，研究更为深入，因此本节将会以短语结构分析为例介绍句法分析的相关概念。

而句法分析到底是什么呢？简单的理解，句法分析就是在小学语文课程中学习的句子成分的分析，以及对句子中各个成分内部、外部关系的判断。更规范一些的定义，可以参照百度百科和维基百科关于句法分析的解释。

#### 定义 3.4.1 句法分析

句法分析就是指对句子中的词语语法功能进行分析。

——百度百科

在自然语言或者计算机语言中，句法分析是利用形式化的文法规则对一个符号串进行分析的过程。

——维基百科（译文）

上面的定义中，句法分析包含三个重要的概念：

- 形式化的文法：描述语言结构的定义，由文法规则组成。
- 符号串：在本节中，符号串就是指词串，由前面提到的分词系统生成。
- 分析：使用形式文法对符号串进行分析的具体方法，在这里指实现分析的计算机算法。

以上三点是实现一个句法分析器的要素，本节的后半部分会对相关的概念和技术方法进行介绍。

#### 3.4.2 上下文无关文法

句法树是对句子的一种抽象，这种树形结构表达了一种对句子结构的归纳过程，比如，从树的叶子开始，把每一个树节点看作一次抽象，最终形成一个根节点。那这个过程如何用计算机来实现呢？这就需要使用到形式文法。

形式文法是分析自然语言的一种重要工具。根据乔姆斯基的定义<sup>[8]</sup>，形式文法分为四种类型：无限制文法（0型文法）、上下文有关文法（1型文法）、上下文无关文

法（2型文法）和正规文法（3型文法）。不同类型的文法有不同的应用，比如，正规文法可以用来描述有限状态自动机，因此也会被使用在语言模型等系统中。对于短语结构分析问题，常用的是**上下文无关文法**（Context-free Grammar）。上下文无关文法的具体形式如下：

### 定义 3.4.2 上下文无关文法

一个上下文无关文法可以被视为一个系统  $G = \langle N, \Sigma, R, S \rangle$ ，其中

- $N$  为一个非终结符集合；
- $\Sigma$  为一个终结符集合；
- $R$  为一个规则（产生式）集合，每条规则  $r \in R$  的形式为  $X \rightarrow Y_1 Y_2 \dots Y_n$ ，其中  $X \in N, Y_i \in N \cup \Sigma$ ；
- $S$  为一个起始符号集合且  $S \subseteq N$ 。

举例说明，假设有上下文无关文法  $G = \langle N, \Sigma, R, S \rangle$ ，可以用它描述一个简单汉语句法结构。其中非终结符集合为不同的汉语句法标记

$$N = \{\text{NN}, \text{VV}, \text{NP}, \text{VP}, \text{IP}\}$$

这里，NN 代表名词，VV 代表动词，NP 代表名词短语，VP 代表动词短语，IP 代表单句。进一步，把终结符集合定义为

$$\Sigma = \{\text{猫, 喜欢, 吃, 鱼}\}$$

再定义起始符集合为

$$S = \{\text{IP}\}$$

最后，文法的规则集定义图3.16所示（其中  $r_i$  为规则的编号）。这个文法蕴含了不同“层次”的句法信息。比如，规则  $r_1, r_2, r_3$  和  $r_4$  表达了词性对单词的抽象；规则  $r_6, r_7$  和  $r_8$  是表达了短语结构的抽象，其中，规则  $r_8$  描述了汉语中名词短语（主语）+ 动词短语（谓语）的结构。在实际应用中，像  $r_8$  这样的规则可以覆盖很大的片段（试想一下一个包含 50 个词的主谓结构的句子，可以使用  $r_8$  进行描述）。

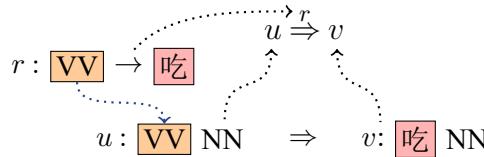
上下文无关文法的规则是一种**产生式规则**（Production Rule），形如  $\alpha \rightarrow \beta$ ，它表示把规则左端的非终结符  $\alpha$  替换为规则右端的符号序列  $\beta$ 。通常， $\alpha$  被称作规则的**左部**（Left-hand Side）， $\beta$  被称作规则的**右部**（Right-hand Side）。使用右部  $\beta$  替换左部  $\alpha$  的过程也被称作规则的使用，而这个过程的逆过程称为规约。规则的使用可以如下定义：

$r_1: \text{NN} \rightarrow \text{猫}$	$r_2: \text{VV} \rightarrow \text{喜欢} \rightarrow \text{喜欢}$
$r_3: \text{VV} \rightarrow \text{吃}$	$r_4: \text{NN} \rightarrow \text{鱼} \rightarrow \text{鱼}$
$r_5: \text{NP} \rightarrow \text{NN}$	$r_6: \text{VP} \rightarrow \text{VV NN} / \text{V NN}$
$r_7: \text{VP} \rightarrow \text{VV VP}$	$r_8: \text{IP} \rightarrow \text{NP VP NP VP}$
$r_1, r_2, r_3, r_4$ 为生成单词词性的规则	
$r_5$ 为单变量规则，它将词性 NN 进一步抽象为名词短语 NP	
$r_6, r_7, r_8$ 为句法结构规则，比如 $r_8$ 表示了主 (NP)+ 谓 (VP) 结构	

图 3.16 一个示例文法的规则集

**定义 3.4.3** 上下文无关文法规则的使用

一个符号序列  $u$  可以通过使用规则  $r$  替换其中的某个非终结符，并得到符号序列  $v$ ，于是  $v$  是在  $u$  上使用  $r$  的结果，记为  $u \xrightarrow{r} v$ ：



给定起始非终结符，可以不断地使用规则，最终生成一个终结字符串，这个过程也被称为**推导**（Derivation）。形式化的定义为：

**定义 3.4.4** 推导

给定一个文法  $G = < N, \Sigma, R, S >$ ，对于一个字符串序列  $s_0, s_1, \dots, s_n$  和规则序列  $r_1, r_2, \dots, r_n$ ，满足

$$s_0 \xrightarrow{r_1} s_1 \xrightarrow{r_2} s_2 \xrightarrow{r_3} \dots \xrightarrow{r_n} s_n$$

且

- $\forall i \in [0, n], s_i \in (N \cup \Sigma)^*$   $\triangleleft s_i$  为合法的字符串
- $\forall j \in [1, n], r_j \in R$   $\triangleleft r_j$  为  $G$  的规则
- $s_0 \in S$   $\triangleleft s_0$  为起始非终结符
- $s_n \in \Sigma^*$   $\triangleleft s_n$  为终结符序列

则  $s_0 \xrightarrow{r_1} s_1 \xrightarrow{r_2} s_2 \xrightarrow{r_3} \dots \xrightarrow{r_n} s_n$  为一个推导

比如，使用前面的示例文法，可以对“猫/喜欢/吃/鱼”进行分析，并形成句法分析树（图3.17）。从起始非终结符 IP 开始，使用唯一拥有 IP 作为左部的规则  $r_8$  推导出 NP 和 VP，之后依次使用规则  $r_5$ 、 $r_1$ 、 $r_7$ 、 $r_2$ 、 $r_6$ 、 $r_3$ 、 $r_4$ ，得到了完整的句法树。

通常，可以把推导简记为  $d = r_1 \circ r_2 \circ \dots \circ r_n$ ，其中  $\circ$  表示规则的组合。显然， $d$

也对应了树形结构，也就是句法分析结果。从这个角度看，推导就是描述句法分析树的一种方式。此外，规则的推导也把规则的使用过程与生成的字符串对应起来。一个推导所生成的字符串，也被称作文法所产生的一个**句子**（Sentence）。而一个文法所能生成的所有句子的集合是这个文法所对应的**语言**（Language）。

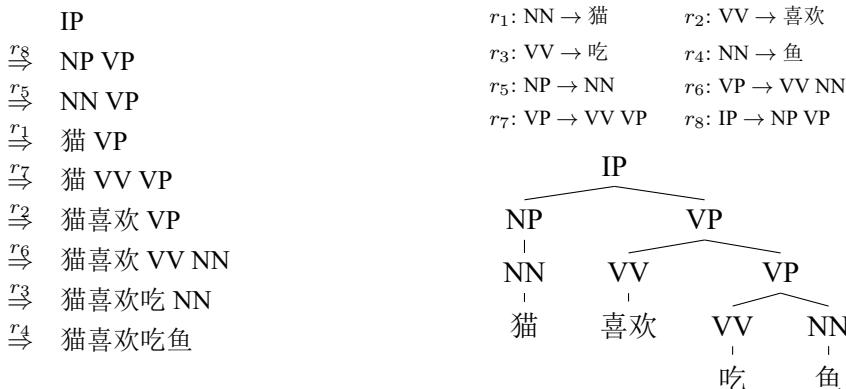


图 3.17 上下文无关文法推导实例

但是，句子和规则的推导并不是一一对应的。同一个句子，往往有很多推导的方式，这种现象被称为**歧义**（Ambiguity）。甚至同一棵句法树，也可以对应不同的推导，图3.18给出同一棵句法树所对应的两种不同的规则推导。

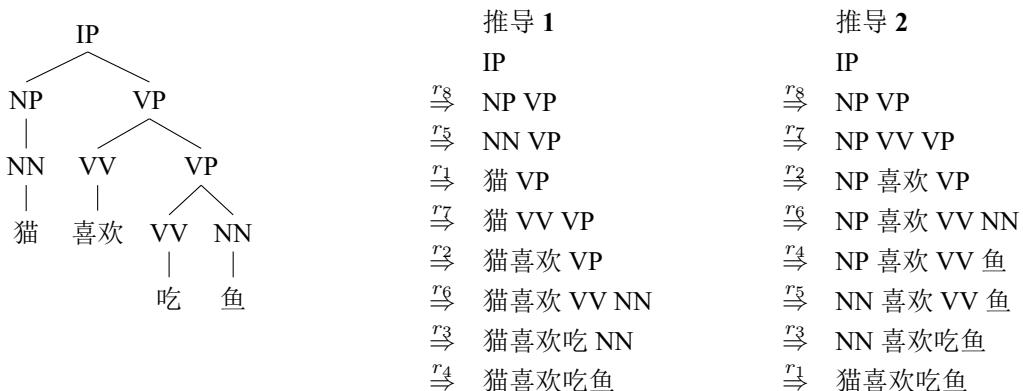


图 3.18 同一棵句法树对应的不同规则推导

显然，规则顺序的不同会导致句法树的推导这一确定的过程变得不确定，因此，需要进行**消歧**（Disambiguation）。这里，可以使用启发式方法：要求规则使用都服从最左优先原则，这样得到的推导被称为**最左优先推导**（Left-most Derivation）。图3.18中的推导 1 就是符合最左优先原则的推导。

这样，对于一个上下文无关文法，每一棵句法树都有唯一的最左推导与之对应。

于是，句法分析可以被描述为：对于一个句子找到能够生成它的最佳推导，这个推导所对应的句法树就是这个句子的句法分析结果。

不过问题又回来了，怎样才能知道什么样的推导或者句法树是“最佳”的呢？如图3.19所示，对于语言学专家，他们可以很确定地分辨出哪些句法树是正确的，哪些句法树是错误。甚至普通人也可以通过一些课本中学到的知识产生一些模糊的判断。而计算机如何进行判别呢？沿着前面介绍的统计建模的思想，计算机可以得出不同句法树出现的概率，进而选择概率最高的句法树作为输出，而这正是统计句法分析所做的事情。

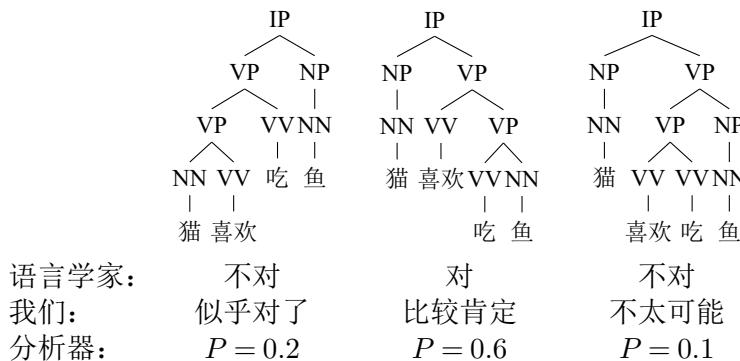


图 3.19 如何选择最佳的句法分析结果 - 专家、普通人和句法分析器的视角

在统计句法分析中，需要对每个推导进行统计建模，于是定义一个模型  $P(\cdot)$ ，对于任意的推导  $d$ ，都可以用  $P(d)$  计算出推导  $d$  的概率。这样，给定一个输入句子，我们可以对所有可能的推导用  $P(d)$  计算其概率值，并选择概率最大的结果作为句法分析的结果输出（图3.20）。

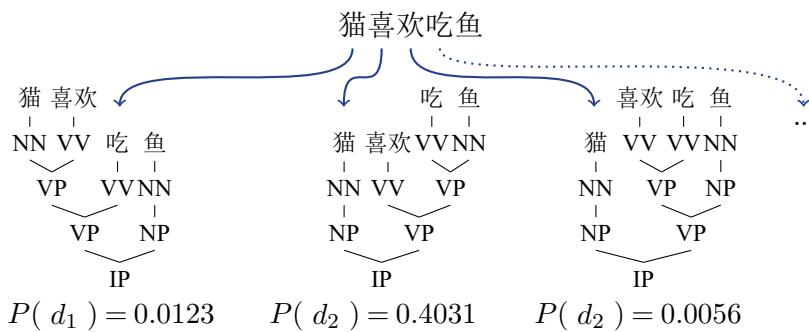


图 3.20 不同推导（句法树）对应的概率值

### 3.4.3 规则和推导的概率

对句法树进行概率化，首先要对使用的规则进行概率化。为了达到这个目的，可以使用**概率上下文无关文法**（Probabilistic Context-free Grammar），它是上下文无关文

法的一种扩展。

#### 定义 3.4.5 概率上下文无关文法

一个概率上下文无关文法可以被视为一个系统  $G = \langle N, \Sigma, R, S \rangle$ , 其中

- $N$  为一个非终结符集合;
- $\Sigma$  为一个终结符集合;
- $R$  为一个规则(产生式)集合, 每条规则  $r \in R$  的形式为  $p: X \rightarrow Y_1 Y_2 \dots Y_n$ , 其中  $X \in N, Y_i \in N \cup \Sigma$ , 每个  $r$  都对应一个概率  $p$ , 表示其生成的可能性;
- $S$  为一个起始符号集合且  $S \subseteq N$ 。

概率上下文无关文法与传统上下文无关文法的区别在于, 每条规则都会有一个概率, 描述规则生成的可能性。具体来说, 规则  $P(\alpha \rightarrow \beta)$  的概率可以被定义为:

$$P(\alpha \rightarrow \beta) = P(\beta | \alpha) \quad (3.13)$$

即, 在给定规则左部的情况下生成规则右部的可能性。进一步, 在上下文无关文法中, 每条规则之间的使用都是相互独立的<sup>9</sup>, 因此可以把  $P(d)$  分解为规则概率的乘积:

$$\begin{aligned} P(d) &= P(r_1 \cdot r_2 \cdot \dots \cdot r_n) \\ &= P(r_1) \cdot P(r_2) \cdots P(r_n) \end{aligned} \quad (3.14)$$

这个模型可以很好的解释词串的生成过程。比如, 对于规则集

$$\begin{aligned} r_3 : \quad &\text{VV} \rightarrow \text{吃} \\ r_4 : \quad &\text{NN} \rightarrow \text{鱼} \\ r_6 : \quad &\text{VP} \rightarrow \text{VV NN} \end{aligned}$$

可以得到  $d_1 = r_3 \cdot r_4 \cdot r_6$  的概率为

$$\begin{aligned} P(d_1) &= P(r_3) \cdot P(r_4) \cdot P(r_6) \\ &= P(\text{VV} \rightarrow \text{吃}) \cdot P(\text{NN} \rightarrow \text{鱼}) \cdot P(\text{VP} \rightarrow \text{VV NN}) \end{aligned} \quad (3.15)$$

这也对应了词串“吃/鱼”的生成过程。首先, 从起始非终结符 VP 开始, 使用规则  $r_6$  生成两个非终结符 VV 和 NN; 进一步, 分别使用规则  $r_3$  和  $r_4$  从 VV 和 NN 进一步生成单词“吃”和“鱼”。整个过程的概率等于三条规则概率的乘积。

---

<sup>9</sup>如果是上下文有关文法, 规则会形如  $a\alpha b \rightarrow a\beta b$ , 这时  $\alpha \rightarrow \beta$  的过程会依赖前后上下文  $a$  和  $b$

新的问题又来了，如何得到规则的概率呢？这里仍然可以从数据中学习文法规则的概率。假设有人工标注的数据，它包括很多人工标注句法树的句法，称之为**树库**（Treebank）。然后，对于规则  $r: \alpha \rightarrow \beta$  可以使用基于频次的方法：

$$P(r) = \frac{\text{规则 } r \text{ 在树库中出现的次数}}{\alpha \text{ 在树库中出现的次数}} \quad (3.16)$$

图3.21展示了通过这种方法计算规则概率的过程。与词法分析类似，可以统计树库中规则左部和右部同时出现的次数，除以规则左部出现的全部次数，所得的结果就是所求规则的概率。这种方法也是典型的相对频次估计。但是如果规则左部和右部同时出现的次数为 0 时是否代表这个规则概率是 0 呢？遇到这种情况，可以使用平滑方法对概率进行平滑处理，具体思路可参考第二章的相关内容。

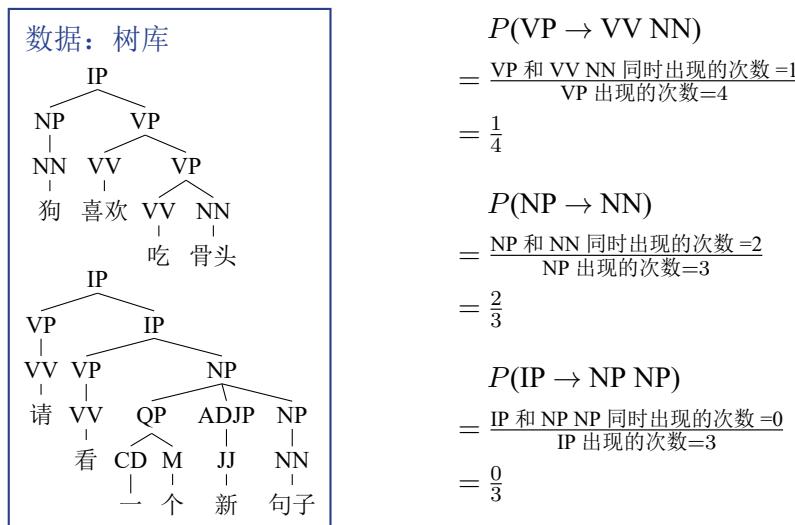


图 3.21 上下文无关文法规则概率估计

图3.22展示了基于统计的句法分析的流程。首先，通过树库上的统计，获得各个规则的概率，这样就得到了一个上下文无关句法分析模型  $P(\cdot)$ 。对于任意句法分析结果  $d = r_1 \circ r_2 \circ \dots \circ r_n$ ，都能通过如下公式计算其概率值：

$$P(d) = \prod_{i=1}^n P(r_i) \quad (3.17)$$

在获取统计分析模型后，就可以使用模型对任意句子进行分析，计算每个句法分析树的概率，并输出概率最高的树作为句法分析的结果。



图 3.22 统计句法分析的流程

## 3.5 小结及拓展阅读

本章将统计建模的思想应用到三个自然语言处理任务中，包括：中文分词、命名实体识别、短语结构句法分析。它们和机器翻译有着紧密的联系，往往作为机器翻译系统输入和输出的数据加工方法。可以发现：经过适当的假设和化简，统计模型可以很好的描述复杂的自然语言处理问题。这种建模手段也会在后续章节的内容中被广泛使用。

由于本章重点介绍如何用统计方法对自然语言处理任务进行建模，因此并没有对具体的问题展开深入讨论。有几方面内容，读者可以继续关注：

- 在建模方面，本章描述了基于 1-gram 语言模型的分词、基于上下文无关文法的句法分析等，它们都是基于人工先验知识进行模型设计的思路。也就是，问题所表达的现象被“一步一步”生成出来。这是一种典型的生成式建模思想，它把要解决的问题看作一些观测结果的隐含变量（比如，句子是观测结果，分词结果是隐含在背后的变量），之后通过对隐含变量生成观测结果的过程进行建模，以达到对问题进行数学描述的目的。这类模型一般需要依赖一些独立性假设，假设的合理性对最终的性能有较大影响。相对于生成式模型，另一类方法是**判别式模型** (Discriminative Model)。本章序列标注内容中提到一些模型就是判别式模型，如条件随机场<sup>[97]</sup>。它直接描述了从隐含变量生成观测结果的过程，这样对问题的建模更加直接，同时这类模型可以更加灵活的引入不同的特征。判别模型在自然语言处理中也有广泛应用<sup>[107, 108, 109, 110, 111]</sup>。在本书的第七章也会使用到判别式模型。
- 事实上，本章并没有对分词、句法分析中的预测问题进行深入介绍。比如，如何找到概率最大的分词结果？这个问题的解决可以直接借鉴第二章中介绍的搜索方法：对于基于  $n$ -gram 语言模型的分词方法，可以使用动态规划方法<sup>[112]</sup> 进行搜索；在不满足动态规划的使用条件时，可以考虑使用更加复杂的搜索策略，并配合一定的剪枝方法找到最终的分词结果。实际上，无论是基于  $n$ -gram 语言模型的分词还是简单的上下文无关文法都有高效的推断方法。比如， $n$ -gram 语言模型可以被视为概率有限状态自动机，因此可以直接使用成熟的自动机工具<sup>[113]</sup>。对于更复杂的句法分析问题，可以考虑使用移进- 规约方法来解决预测问题<sup>[114]</sup>。
- 从自然语言处理的角度来看，词法分析和语法分析中的很多问题都是序列标

注问题，例如本章介绍的分词和命名实体识别。此外序列标注还可以被扩展到词性标注<sup>[115]</sup>、组块识别<sup>[116]</sup>、关键词抽取<sup>[117]</sup>、词义角色标注<sup>[118]</sup>等任务，本章着重介绍了传统的方法，前沿方法大多与深度学习相结合，感兴趣的读者可以自行了解，其中比较有代表性的使用双向长短时记忆网络对序列进行建模，之后于不同模型进行融合得到最终的结果，例如，与条件随机场相结合的模型（BiLSTM-CRF）<sup>[119]</sup>、与卷积神经网络相结合的模型（BiLSTM-CNNs）<sup>[120]</sup>、与简单的Softmax结构相结合的模型<sup>[121]</sup>等。此外，对于序列标注任务，模型性能很大程度上依赖对输入序列的表示能力，因此基于预训练语言模型的方法也非常流行<sup>[122]</sup>，如：BERT<sup>[123]</sup>、GPT<sup>[124]</sup>、XLM<sup>[125]</sup>等。



## 4. 翻译质量评价

人们在使用机器翻译系统时需要评估系统输出结果的质量。这个过程也被称作机器翻译译文质量评价，简称为**译文质量评价**（Quality Evaluation of Translation）。在机器翻译的发展进程中，译文质量评价有着非常重要的作用。不论在系统研发的反复迭代中，还是在诸多的机器翻译应用场景中，都存在大量的译文质量评价环节。从某种意义上说，没有译文质量评价，机器翻译也不会发展成今天的样子。比如，本世纪初研究人员提出了译文质量自动评价方法 BLEU<sup>[126]</sup>。该方法使得机器系统的评价变得自动、快速、便捷，而且评价过程可以重复。正是由于 BLEU 等自动评价方法的提出，机器翻译研究人员可以在更短的时间内得到译文质量的评价结果，加速系统研发的进程。

时至今日，译文质量评价方法已经非常丰富，针对不同的使用场景研究人员陆续提出了不同的方法。本章将会对其中的典型方法进行介绍，包括：人工评价、有参考答案自动评价、无参考答案自动评价等。相关方法及概念也会在本章的后续章节中被广泛使用。

### 4.1 译文质量评价所面临的挑战

一般来说，译文质量评价可以被看作是一个对译文进行打分或者排序的过程，打分或者排序的结果代表了翻译质量的好坏。比如，表4.1展示一个汉译英的译文质量评价结果。这里采用了5分制打分，1代表最低分，5代表最高分。可以看出，流畅的高质量译文得分较高，相反，存在问题的译文得分较低。

表 4.1 汉译英译文质量评价实例

源文	那/只/敏捷/的/棕色/狐狸/跳过/了/那/只/懒惰/的/狗/。	评价得分
机器译文 1	The quick brown fox jumped over the lazy dog .	5
机器译文 2	The fast brown fox jumped over a sleepy dog .	4
机器译文 3	The fast brown fox jumps over the dog .	3
机器译文 4	The quick brown fox jumps over dog .	2
机器译文 5	A fast fox jump dog .	1

这里的一个核心问题是：从哪个角度对译文质量进行评价呢？常用的标准有：**流畅度**（Fluency）和**忠诚度**（Fidelity）<sup>[127]</sup>。其中流畅度是指译文在目标语言中的流畅程度，越通顺的译文流畅度越高；忠诚度是指译文表达源文意思的程度，如果译文能够全面、准确的表达源文的意思，那么它具有较高的翻译忠诚度。在一些极端的情况下，译文可以非常流畅，但是与源文完全不对应。或者，译文可以非常好的对应源文，但是读起来非常不连贯。这些译文都不是很好的译文。

传统观点把翻译分为“信”、“达”、“雅”三个层次，而忠诚度体现的是一种“信”的思想，而流畅度体现的是一种“达”的思想。不过“雅”在机器翻译评价中还不是一个常用的标准，而且机器翻译还没有达到“雅”的水平，是未来所追求的目标。

给定评价标准，译文质量评价有很多实现方式。比如，可以使用人工评价的方式让评委对每个译文进行打分（4.2节），也可以用自动评价的方式让计算机比对译文和参考答案之间的匹配的程度（4.3节）。但是，自然语言的翻译是最复杂的人工智能问题之一。这不仅仅体现在相关问题的建模和系统实现的复杂性上，译文质量评价也同样面临着诸多挑战。

- **译文不唯一。**自然语言表达的丰富性决定了同一个意思往往有很多种表达方式。同一句话，由不同译者的翻译也往往存在差异。译者的背景、翻译水平、翻译所处的语境，甚至译者的情绪都会对译文产生影响。如何在评价过程中尽可能考虑多样的译文，是译文质量评价中最具挑战的问题之一。
- **评价标准不唯一。**虽然流畅度和忠诚度给译文质量评价提供了很好的参考依据，但是在实践中往往会有更多样的需求。比如，在专利翻译中，术语翻译的准确性就是必须要考虑的因素，一个术语的翻译错误会导致整个译文不可用。此外，术语翻译的一致性也是非常重要的，即使同一个术语有多种正确的译文，但是在同一个专利文档中，术语翻译需要保持一致。不同的需求使得很难用统一的标准对译文质量进行评价。在实践中，往往需要针对不同应用场景设计不同的评价标准。
- **自动评价与人工评价存在着偏差。**固然使用人工的方式可以准确地评估译文质量，但是这种方式费时、费力。而且由于人工评价的主观性，其结果不易重现，也就是不同人的评价结果会有差异。这些因素也造成了人工评价不能被过于频繁

的使用。翻译质量的自动评价可以充分利用计算机的计算能力，对译文与参考答案进行比对，具有速度快、结果可重现的优点，但是其精度不如人工评价。使用何种评价方法也是实践中需要考虑的重要问题之一。

- **参考答案不容易获得。**很多情况下，译文的正确答案并不容易获取。甚至对于某些低资源语种，相关的语言学家都很稀缺。这时很难进行基于标准答案的评价。如何在没有参考答案的情况下对译文质量进行估计是极具应用前景且颇具挑战的方向。

针对以上问题，研究人员设计出多种不同的译文质量评价方法。根据人工参与方式的不同，可以分为人工评价、有参考答案的自动评价、无参考答案的自动评价。这些方法也对应了不同的使用场景。

- **人工评价。**当需要对系统进行准确的评估时，往往采用人工评价。比如，对于机器翻译的一些互联网应用，在系统上线前都会采用人工评价对机器翻译系统性能进行测试。当然，这种方法的时间和人力成本是最高的。
- **有参考答案的自动评价。**由于机器翻译系统研发过程中需要频繁地对系统性能进行评价，这时可以让人标注一些正确的译文，之后把这些译文作为参考答案与机器翻译系统输出的结果进行比对。这种自动评价的结果获取成本低，可以多次重复，而且可以用于对系统结果的快速反馈，指导系统优化的方向。
- **无参考答案的自动评价。**在很多应用场景中，在系统输出译文时，使用者希望提前知道译文的质量，即使这时并没有可比对的参考答案。这样，系统使用者可以根据这个对质量的“估计”结果有选择地使用机器翻译译文。严格意义上说，这并不是一个传统的译文质量评价方法，而是一种对译文置信度和可能性的估计。

图4.1给出了机器翻译译文评价方法的逻辑关系图。需要注意的是，很多时候，译文质量评价结果是用于机器翻译系统优化的。在随后的章节中也会看到，译文评价的结果会被用于不同的机器翻译模型优化中。甚至很多统计指标（如极大似然估计）也可以被看作是一种对译文的“评价”，这样就可以把机器翻译的建模和译文评价联系在了一起。本章的后半部分将重点介绍传统的译文质量评价方法。与译文质量评价相关的模型优化方法将会在后续章节详细论述。

## 4.2 人工评价

顾名思义，人工评价是指评价者根据翻译结果好坏对译文进行评价。例如，可以根据句子的忠诚度和流畅度对其进行打分，这样能够准确评定出译文是否准确翻译出源文的意思以及译文是否通顺。在人工评价时，一般由多个评价者匿名对译文打分，之后综合所有评价者的评价结果给出最终的得分。人工评价可以准确反映句子的翻译质量，是最权威、可信度最高的评价方法，但是其缺点也十分明显：需要耗

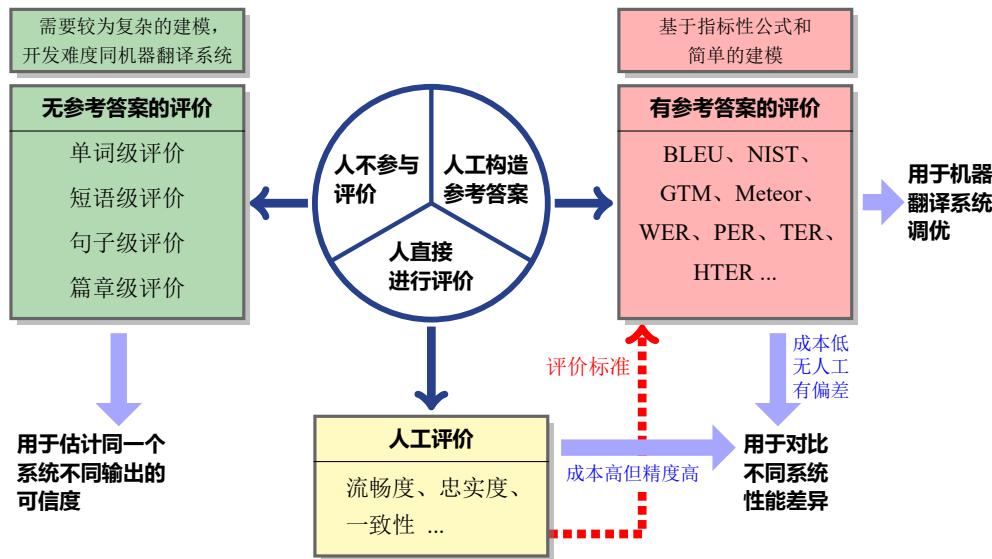


图 4.1 译文质量评价方法逻辑图

费人力物力，而且评价的周期长，不能及时得到有效的反馈。因此在实际系统开发中，纯人工评价不会过于频繁地被使用，它往往和自动评价一起配合，帮助系统研发人员准确的了解当前系统的状态。

#### 4.2.1 评价策略

合理的评价指标是人工评价得以顺利进行的基础。机器译文质量的人工评价可以追溯到 1966 年，自然语言处理咨询委员会提出**可理解度**（Intelligibility）和**忠诚度**作为机器译文质量人工评价指标<sup>[128]</sup>。1994 年，**充分性**（Adequacy）、流畅度和**信息性**（Informativeness）成为 ARPA MT<sup>1</sup> 的人工评价标准<sup>[129]</sup>。此后，有不少研究者提出了更多的机器译文质量人工评估指标，例如将**清晰度**（Clarity）和**连贯性**（Coherence）加入人工评价指标中<sup>[130]</sup>。甚至有人将各种人工评价指标集中在一起，组成了尽可能全面的机器翻译评估框架<sup>[131]</sup>。

人工评价的策略非常多。考虑不同的因素，往往会使用不同的评价方案，比如：

- **是否呈现源语言文本**。在进行人工评价时，可以向评价者提供源语言文本或参考答案，也可以同时提供源语言文本和参考答案。从评价的角度，参考答案已经能够帮助评价者进行正确评价，但是源语言文本可以提供更多信息帮助评估译文的准确性。
- **评价者选择**。理想情况下，评价者应同时具有源语言和目标语言的语言能力。但是，很多时候具备双语能力的评价者很难招募，因此这时会考虑使用目标语为母语的评价者。配合参考答案，单语评价者也可以准确地评价译文质量。

<sup>1</sup> ARPA MT 计划是美国高级研究计划局软件和智能系统技术处人类语言技术计划的一部分。

- **多个系统同时评价。**如果有多个不同系统的译文需要评价，可以直接使用每个系统单独打分的方法。但是，如果仅仅是想了解不同译文之间的相对好坏，也可以采用竞评的方式：对每个待翻译的源语言句子，根据各个机器翻译系统输出的译文质量对所有待评价的机器翻译系统进行排序，这样做的效率会高于直接打分，而且评价准确性也能够得到保证。
- **数据选择。**评价数据一般需要根据目标任务进行采集，为了避免和系统训练数据重复，往往会搜集最新的数据。而且，评价数据的规模越大，评价结果越科学。常用的做法是搜集一定量的评价数据，之后从中采样出所需的数据。由于不同的采样会得到不同的评价集合，这样的方法可以复用多次，得到不同的测试集。
- **面向应用的评价。**除了人工直接打分，一种更有效的方法是把机器翻译的译文嵌入到下游应用中，通过机器翻译对下游应用的改善效果评估机器翻译译文质量。比如，可以把机器翻译放入译后编辑流程中，通过对比译员翻译效率的提升来评价译文质量。再比如，把机器翻译放入线上应用中，通过点击率或者用户反馈来评价机器翻译的品质。

#### 4.2.2 打分标准

如何对译文进行打分是机器翻译评价的核心问题。在人工评价方法中，一种被广泛使用的方法是**直接评估**（Direct Assessment, DA）<sup>[129]</sup>，这种评价方法需要评价者给出对机器译文的绝对评分：在给定一个机器译文和一个参考答案的情况下，评价者直接给出 1-100 的分数用来表征机器译文的质量。与其类似的策略是对机器翻译质量进行等级评定<sup>[132]</sup>，常见的是在 5 级或 7 级标准中指定单一等级用以反映机器翻译质量。也有研究者提出利用语言测试技术对机器翻译质量进行评价<sup>[133]</sup>，其中涉及多等级内容的评价：第一等级测试简单的短语、成语、词汇等；第二等级利用简单的句子测试机器翻译在简单文本上的表现；第三等级利用稍复杂的句子测试机器翻译在复杂语法结构上的表现；第四等级测试引入更加复杂的补语结构和附加语等等。

除了对译文进行简单的打分，另一种经典的人工评价方法是**相对排序**（Relative Ranking, RR）<sup>[134]</sup>。这种方法通过对不同机器翻译的译文质量进行相对排序得到最终的评价结果。举例来说：

- 在每次评价过程中，若干个等待评价的机器翻译系统被分为 5 个一组，评价者被提供 3 个连续的源文片段和 1 组机器翻译系统的相应译文；
- 评价者需要对本组的机器译文根据其质量进行排序，不过评价者并不需要一次性将 5 个译文排序，而是将其两两进行比较，判出胜负或是平局。在评价过程中，由于排序是两两一组进行的，为了评价的公平性，将采用排列组合的方式进行分组和比较，若共有  $n$  个机器翻译系统，则会为被分为  $C_n^5$  组，组内每个系统都将与其他 4 个系统进行比较，由于需要针对 3 个源文片段进行评价对比，

则意味着每个系统都需要被比较  $C_n^5 \times 4 \times 3$  次；

- 最终根据多次比较的结果，对所有参与评价的系统进行总体排名。对于如何获取合理的总体排序，有三种常见的策略：

- **根据系统胜出的次数进行排序**<sup>[135]</sup>。以系统  $S_j$  和系统  $S_k$  为例，两个系统都被比较了  $C_n^5 \times 4 \times 3$  次，其中系统  $S_j$  获胜 20 次，系统  $S_k$  获胜 30 次，总体排名中系统  $S_k$  优于系统  $S_j$ 。
- **根据冲突次数进行排序**<sup>[136]</sup>。第一种排序策略中存在冲突现象：例如在每次两两比较中，系统  $S_j$  胜过系统  $S_k$  的次数比系统  $S_j$  不敌系统  $S_k$  的次数多，若待评价系统仅有系统  $S_j$ 、 $S_k$ ，显然系统  $S_j$  的排名高于系统  $S_k$ 。但当待评价系统很多时，可能系统  $S_j$  在所有比较中获胜的次数低于系统  $S_k$ ，此时就出现了总体排序与局部排序不一致的冲突。因此，有研究者提出，能够与局部排序冲突最少的总体排序才是最合理的。令  $O$  表示一个对若干个系统的排序，该排序所对应的冲突定义为：

$$\text{conflict}(O) = \sum_{S_j, S_k \in O, j \neq k} \max(0, \text{count}_{\text{win}}(S_j, S_k) - \text{count}_{\text{loss}}(S_j, S_k)) \quad (4.1)$$

其中， $S_j$  和  $S_k$  是成对比较的两个系统， $\text{count}_{\text{win}}(S_j, S_k)$  和  $\text{count}_{\text{loss}}(S_j, S_k)$  分别是  $S_j, S_k$  进行成对比较时系统  $S_j$  胜利和失败的次数。而使得  $\text{conflict}(O)$  最低的  $O$  就是最终的系统排序结果。

- **根据某系统最终获胜的期望进行排序**<sup>[137]</sup>。以系统  $S_j$  为例，若共有  $n$  个待评价的系统，则进行总体排序时系统  $S_j$  的得分为其最终获胜的期望，即：

$$\text{score}(S_j) = \frac{1}{n} \sum_{k, k \neq j} \frac{\text{count}_{\text{win}}(S_j, S_k)}{\text{count}_{\text{win}}(S_j, S_k) + \text{count}_{\text{loss}}(S_j, S_k)} \quad (4.2)$$

根据公式(4.2)可以看出，该策略去除了平局的影响。

与相对排序相比，直接评估方法虽然更加直观，但是过度依赖评价者的主观性，因而直接评估适用于直观反映某机器翻译系统性能，而不适合用来比较机器翻译系统之间的性能差距。在需要对大量系统进行快速人工评价时，找出不同译文质量之间的相关关系要比直接准确评估译文质量简单得多，基于排序的评价方法可以大大降低评价者的工作量，所以也被系统研发人员经常使用。

在实际应用中，研究者可以根据实际情况选择不同的人工评价方案，人工评价也没有统一的标准。WMT<sup>[138]</sup> 和 CCMT<sup>[139]</sup> 机器翻译评测都有配套的人工评价方案，可以作为业界的参考标准。

## 4.3 有参考答案的自动评价

人工评价费事费力，同时具有一定的主观性，甚至不同人在不同时刻面对同一篇文章的理解都会不同。为了克服这些问题，另一种思路是将人类专家翻译的结果看作是参考答案，将译文与答案的近似程度作为评价结果。即译文与答案越接近，评价结果越好；反之，评价结果较差。这种评价方式叫做**自动评价**（Automatic Evaluation）。自动评价具有速度快，成本低、一致性高的优点，因此自动评价也是机器翻译系统研发人员所青睐的方法。

随着评价技术的不断发展，自动评价结果已经具有了比较好的指导性，可以帮助使用者快速了解当前译文的质量。在机器翻译领域，自动评价已经成为了一个重要的研究分支。至今，已经有不下几十种自动评价方法被提出。这里无法对这些方法一一列举，为了便于后续章节中对自动评价方法的使用，这里仅对一些代表性的方法进行简要介绍。

### 4.3.1 基于词串比对的方法

这种方法比较关注译文单词及  $n$ -gram 的翻译准确性。其思想是将译文看成是符号序列，通过计算参考答案和机器译文间的序列相似性来评价机器翻译的质量。

#### 1. 基于距离的方法

基于距离的自动评价方法的基本思想是：将机器译文转化为参考答案所需要的**最小编辑步骤数**作为译文质量的度量，基于此类思想的自动评价方法主要有**单词错误率**（Word Error Rate, WER）<sup>[140]</sup>、**与位置无关的单词错误率**（Position-independent word Error Rate, PER）<sup>[141]</sup> 和**翻译错误率**（Translation Error Rate, TER）<sup>[142]</sup> 等。下面介绍其中比较有代表性的方法——翻译错误率，即 TER。

TER 是一种典型的基于距离的评价方法，通过评定机器译文的译后编辑工作量来衡量机器译文质量。在这里“距离”被定义为将一个序列转换成另一个序列所需要的最少编辑操作次数，操作次数越多，距离越大，序列之间的相似性越低；相反距离越小，表示一个句子越容易改写成另一个句子，序列之间的相似性越高。TER 使用的编辑操作包括：**增加、删除、替换和移位**。其中增加、删除、替换操作计算得到的距离被称为编辑距离。TER 根据错误率的形式给出评分：

$$\text{score} = \frac{\text{edit}(o, g)}{l} \quad (4.3)$$

其中， $\text{edit}(o, g)$  表示系统生成的译文  $o$  和参考答案  $g$  之间的距离， $l$  是归一化因子，通常为参考答案的长度。在距离计算中所有的操作的代价都为 1。在计算距离时，优先考虑移位操作，再计算编辑距离（即增加、删除和替换操作的次数）。直到增加、移位操作无法减少编辑距离时，将编辑距离和移位操作的次数累加得到 TER 计算的距离。

**实例 4.1** 机器译文: A cat is standing in the ground .

参考答案: The cat is standing on the ground .

在这个实例中, 将机器译文序列转换为参考答案序列, 需要进行两次替换操作, 将 “A” 替换为 “The”, 将 “in” 替换为 “on”。所以  $\text{edit}(c, r) = 2$ , 归一化因子  $l$  为参考答案的长度 8 (包括标点符号), 所以该机器译文的 TER 结果为 2/8。

PER 与 WER 的基本思想与 TER 相同, 这三种方法的主要区别在于对 “错误”的定义和考虑的操作类型略有不同。WER 使用的编辑操作包括: 增加、删除、替换, 由于没有移位操作, 当机器译文出现词序问题时, 会发生多次替代, 因而一般会低估译文质量; 而 PER 只考虑增加和删除两个动作, 在不考虑词序的情况下, PER 计算两个句子中出现相同单词的次数, 根据机器译文与参考答案的长度差距, 其余操作无非是插入词或删除词, 这样往往会高估译文质量。

## 2. 基于 $n$ -gram 的方法

BLEU 是目前使用最广泛的自动评价指标。BLEU 是 Bilingual Evaluation Under-study 的缩写, 由 IBM 的研究人员在 2002 年提出<sup>[126]</sup>。通过采用  $n$ -gram 匹配的方式评定机器翻译结果和参考答案之间的相似度, 机器译文越接近参考答案就认定它的质量越高。 $n$ -gram 是指  $n$  个连续单词组成的单元, 称为  $n$  元语法单元 (见第三章)。 $n$  越大表示评价时考虑的匹配片段越大。

BLEU 的计算首先考虑待评价机器译文中  $n$ -gram 在参考答案中的匹配率, 称为  **$n$ -gram 准确率** ( $n$ -gram Precision)。其计算方法如下:

$$P_n = \frac{\text{count}_{\text{hit}}}{\text{count}_{\text{output}}} \quad (4.4)$$

其中,  $\text{count}_{\text{hit}}$  表示机器译文中  $n$ -gram 在参考答案中命中的次数,  $\text{count}_{\text{output}}$  表示机器译文中总共有多少  $n$ -gram。为了避免同一个词被重复计算, BLEU 的定义中使用了截断的方式定义  $\text{count}_{\text{hit}}$  和  $\text{count}_{\text{output}}$ 。

**实例 4.2** 机器译文: the the the the

参考答案: The cat is standing on the ground .

在引入截断方式之前, 该机器译文的 1-gram 准确率为  $4/4 = 1$ , 这显然是不合理的。在引入截断的方式之后, “the” 在译文中出现 4 次, 在参考答案中出现 2 次, 截断操作则是取二者的最小值, 即  $\text{count}_{\text{hit}}=2$ ,  $\text{count}_{\text{output}}=4$ , 该译文的 1-gram 准确率为  $2/4$ 。

令  $N$  表示考虑的最大  $n$ -gram 的大小, 则译文整体的准确率等于各  $n$ -gram 的加权平均:

$$P_{\text{avg}} = \exp\left(\sum_{n=1}^N w_n \cdot \log P_n\right) \quad (4.5)$$

但是，该方法更倾向于对短句子打出更高的分数。一个极端的例子是译文只有很少的几个词，但是都命中答案，准确率很高可显然不是好的译文。因此，BLEU 引入**短句惩罚因子**（Brevity Penalty，BP）的概念，对短句进行惩罚：

$$\text{BP} = \begin{cases} 1 & c > r \\ \exp(1 - \frac{r}{c}) & c \leq r \end{cases} \quad (4.6)$$

其中， $c$  表示机器译文的句子长度， $r$  表示参考答案的句子长度。最终 BLEU 的计算公式为：

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^N w_n \cdot \log P_n\right) \quad (4.7)$$

实际上，BLEU 的计算也是一种综合考虑**准确率**（Precision）和**召回率**（Recall）的方法。公式中， $\exp\left(\sum_{n=1}^N w_n \cdot \log P_n\right)$  是一种准确率的表示。BP 本是一种召回率的度量，它会惩罚过短的结果。这种设计同分类系统中评价指标 F1 值是有相通之处的<sup>[143]</sup>。

从机器翻译的发展来看，BLEU 的意义在于它给系统研发人员提供了一种简单、高效、可重复的自动评价手段，在研发机器翻译系统时可以不需要依赖人工评价。同时，BLEU 也有很多创新之处，包括引入  $n$ -gram 的匹配，截断计数和短句惩罚等等，NIST 等很多评价指标都是受到 BLEU 的启发。此外，BLEU 本身也有很多不同的实现方式，包括 IBM-BLEU<sup>[126]</sup>、NIST-BLEU<sup>2</sup><sup>[144]</sup>、BLEU-SBP<sup>[144]</sup>、ScareBLEU<sup>[145]</sup> 等，使用不同实现方式得到评价结果会有差异。因此在实际使用 BLEU 进行评价时需要确认其实现细节，以保证结果与相关工作评价要求相符。

还需要注意的是，BLEU 的评价结果与所使用的参考答案数量有很大相关性。如果参考答案数量多， $n$ -gram 匹配的几率变大，BLEU 的结果也会偏高。同一个系统，在不同数量的参考答案下进行 BLEU 评价，结果相差 10 个点都十分正常。此外，考虑测试的同源性等因素，相似系统在不同测试条件下的 BLEU 结果差异可能会更大，这时可以采用人工评价的方式以得到更准确的评价结果。

虽然 BLEU 被广泛使用，但也并不完美，甚至经常被人诟病。比如，它需要依赖参考答案，而且评价结果有时与人工评价不一致，同时 BLEU 评价只是单纯地从词串匹配的角度思考翻译质量的好坏，并没有真正考虑句子的语义是否翻译正确。但是，毫无疑问，BLEU 仍然是机器翻译中最常用的评价方法。在没有找到更好的替代方案之前，BLEU 还是机器翻译研究中最重要的评价指标之一。

---

<sup>2</sup>NIST-BLEU 是指美国国家标准与技术研究院（NIST）开发的机器翻译评价工具 mteval 中实现的一种 BLEU 计算的方法。

### 4.3.2 基于词对齐的方法

基于词对齐的方法，顾名思义就是根据参考答案中的单词与译文中的单词之间的对齐关系对机器翻译译文进行评价。词对齐的概念也被用于统计机器翻译的建模（第五章），这里借用了相同的思想来度量机器译文与参考答案之间的匹配程度。在基于  $n$ -gram 匹配的评价方法中（如 BLEU），BP 可以起到一些度量召回率的作用，但是这类方法并没有对召回率进行准确的定义。与其不同的是，基于词对齐的方法在机器译文和参考答案的单词之间建立一对一的对应关系，这种评价方法在引入准确率的同时还能显性引入召回率作为评价所考虑的因素。

在基于词对齐的自动评价方法中，一种典型的方法是 Meteor。该方法通过计算精确的**单词到单词**（Word-to-Word）的匹配来度量一个译文的质量<sup>[146]</sup>，并且在“绝对”匹配之外，还引入了“波特词干匹配”和“同义词”匹配。在下面的内容中，将利用实例对 Meteor 方法进行介绍。

#### 实例 4.3 机器译文：Can I have it like he ?

参考答案：Can I eat this can like him ?

在 Meteor 方法中，首先在机器译文与参考答案之间建立单词之间的对应关系，再根据其对应关系计算准确率和召回率。

- 在机器译文与参考答案之间建立单词之间的对应关系。单词之间的对应关系在建立过程中主要涉及三个模型，在对齐过程中依次使用这三个模型进行匹配：
    - “**绝对**”**匹配模型**（Exact Model）。绝对匹配模型在建立单词对应关系时，要求机器译文端的单词与参考答案端的单词完全一致，并且在参考答案端至多有 1 个单词与机器译文端的单词对应，否则会将其视为多种对应情况。
- 对于实例4.3，使用“绝对”匹配模型，共有两种匹配结果，如图4.2所示。

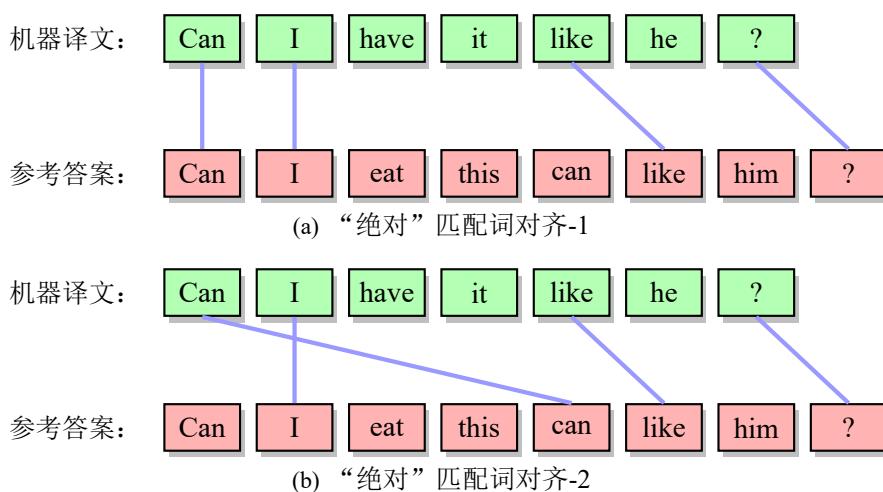


图 4.2 “绝对”匹配模型

- “**波特词干**”匹配模型（Porter Stem Model）。该模型在“绝对”匹配结果的基础上，对尚未对齐的单词进行基于词干的匹配，只需机器译文端单词与参考答案端单词的词干相同即可，如上文中的“do”和“did”。对于图4.2中显示的词对齐结果，再使用“波特词干”匹配模型，得到如图4.3所示的结果。

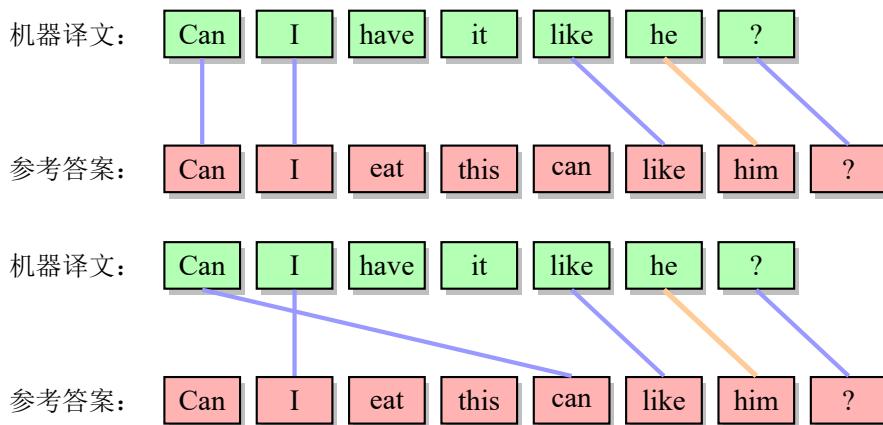


图 4.3 “波特词干”匹配词对齐

- “**同义词**”匹配模型（WN Synonymy Model）。该模型在前两个模型匹配结果的基础上，对尚未对齐的单词进行同义词的匹配，即基于 WordNet 词典匹配机器译文与参考答案中的同义词。如实例4.3中的“eat”和“have”。图4.4给出一个真实例子。

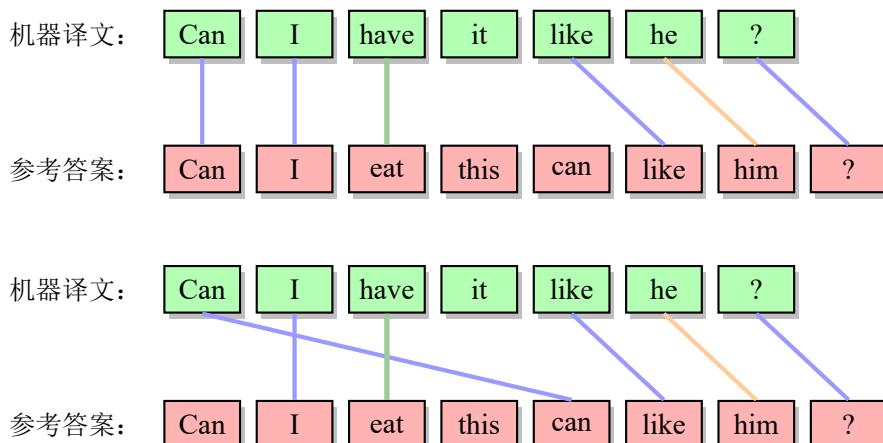


图 4.4 “同义词”匹配词对齐

经过上面的处理，可以得到机器译文与参考答案之间的单词对齐关系。下一步需要从中确定一个拥有最大的子集的对齐关系，即机器译文中被对齐的单词个数最多的对齐关系。但是在上例中的两种对齐关系子集基数相同，这种情况下，

需要选择一个对齐关系中交叉现象出现最少的对齐关系。于是，最终的对齐关系如图4.5所示。

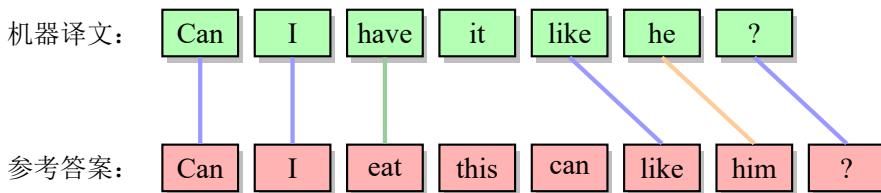


图 4.5 确定最终词对齐

- 在得到机器译文与参考答案的对齐关系后，需要基于对齐关系计算准确率和召回率。

准确率：机器译文中命中单词数与机器译文单词总数的比值。即：

$$P = \frac{\text{count}_{\text{hit}}}{\text{count}_{\text{candidate}}} \quad (4.8)$$

召回率：机器译文中命中单词数与参考答案单词总数的比值。即：

$$R = \frac{\text{count}_{\text{hit}}}{\text{count}_{\text{reference}}} \quad (4.9)$$

- 最后，计算机器译文的得分。利用**调和均值**（Harmonic-mean）将准确率和召回率结合起来，并加大召回率的重要性将其权重调大，例如将召回率的权重设置为9：

$$F_{\text{mean}} = \frac{10PR}{R+9P} \quad (4.10)$$

在上文提到的评价指标中，无论是准确率、召回率还是  $F_{\text{mean}}$ ，都是基于单个词汇信息衡量译文质量，而忽略了语序问题。为了将语序问题考虑进来，Meteor 会考虑更长的匹配：将机器译文按照最长匹配长度分块，由于“块数”较多的机器译文与参考答案的对齐更加散乱，意味着其语序问题更多，因此 Meteor 会对这样的译文给予惩罚。例如图4.5显示的最终词对齐结果中，机器译文被分为了三个“块”——“Can I have it”、“like he”、“?”在这种情况下，看起来上例中的准确率、召回率都还不错，但最终会受到很严重的惩罚。这种罚分机制能够识别出机器译文中的词序问题，因为当待测译文词序与参考答案相差较大时，机器译文将会被分割得比较零散，这种惩罚机制的计算公式如式(4.11)，其中  $\text{count}_{\text{chunks}}$  表示匹配的块数。

$$\text{Penalty} = 0.5 \cdot \left( \frac{\text{count}_{\text{chunks}}}{\text{count}_{\text{hit}}} \right)^3 \quad (4.11)$$

Meteor 评价方法的最终评分为：

$$\text{score} = F_{\text{mean}} \cdot (1 - \text{Penalty}) \quad (4.12)$$

Meteor 方法是经典的自动评价方法之一。它的创新点在于引入了词干匹配和同义词匹配，扩大了词汇匹配的范围。Meteor 方法被提出后，很多人尝试对其进行改进，使其评价结果与人工评价结果更相近。例如 Meteor-next 在 Meteor 的基础上增加**释义匹配器**（Paraphrase Matcher），利用该匹配器能够捕获机器译文中与参考答案意思相近的短语，从而在短语层面进行匹配。此外这种方法还引入了**可调权值向量**（Tunable Weight Vector），用于调节每个匹配类型的相应贡献<sup>[147]</sup>；Meteor 1.3 在 Meteor 的基础上增加了改进的**文本规范器**（Meteor Normalizer）、更高精度的释义匹配以及区分内容词和功能词等指标，其中文本规范器能够根据一些规范化规则，将机器译文中意义等价的标点减少到通用的形式。而区分内容词和功能词则能够得到更为准确的词汇对应关系<sup>[148]</sup>；Meteor Universial 则通过机器学习方法学习不同语言的可调权值，在对低资源语言进行评价时可对其进行复用，从而实现对低资源语言的译文更准确的评价<sup>[149]</sup>。

由于召回率反映参考答案在何种程度上覆盖目标译文的全部内容，而 Meteor 在评价过程中显式引入召回率，所以 Meteor 的评价与人工评价更为接近。但 Meteor 方法需要借助同义词表、功能词表等外部数据，当外部数据中的目标词对应不正确或缺失相应的目标词时，评价水准就会降低。特别是，针对汉语等与英语差异较大的语言，使用 Meteor 方法也会面临很多挑战。不仅如此，超参数的设置和使用，对于评分也有较大影响。

### 4.3.3 基于检测点的方法

基于词串比对和基于词对齐的自动评价方法中提出的 BLEU、TER 等评价指标可以对译文的整体质量进行评估，但是缺乏对具体问题的细致评价。很多情况下，研究人员需要知道系统是否能够处理特定类型的翻译问题，而不是得到一个笼统的评价结果。基于检测点的方法正是基于此想法<sup>[150]</sup>。这种评价方法的优点在于对机器翻译系统给出一个总体评价的同时针对系统在各个具体问题上的翻译能力进行评估，方便比较不同翻译模型的性能。这种方法也被多次用于机器翻译比赛的译文质量评估。

基于检测点的评价根据事先定义好的语言学检测点对译文的相应部分进行打分。如下是几个英中翻译中的检测点实例：

**实例 4.4** They got up at six this morning .

他们/今天/早晨/六点钟/起床。

检测点：时间词的顺序

**实例 4.5** There are nine cows on the farm .

农场/里/有/九/头/牛/。

检测点：量词“头”

#### 实例 4.6 His house is on the south bank of the river .

他/的/房子/在/河/的/南岸/。

We keep our money in a bank .

我们/在/一家/银行/存钱/。

检测点：bank 的多义翻译

该方法的关键在于检测点的获取。有工作曾提出一种从平行双语句子中自动提取检查点的方法<sup>[151]</sup>，借助大量的双语词对齐平行语料，利用自然语言处理工具对其进行词性标注、句法分析等处理，利用预先构建的词典和人工定义的规则，识别语料中不同类别的检查点，从而构建检查点数据库。其中，将检查点分别设计为单词级（如介词、歧义词等）、短语级（如固定搭配）、句子级（特殊句型、复合句型等）三个层面，在对机器翻译系统进行评价时，在检查点数据库中分别选取不同类别检查点对应的测试数据进行测试，从而了解机器翻译系统在各种重要语言现象方面的翻译能力。除此之外，这种方法也能应用于机器翻译系统之间的性能比较中，通过为各个检查点分配合理的权重，用翻译系统在各个检查点得分的加权平均作为系统得分，从而对机器翻译系统的整体水平作出评价。

基于检测点的评价方法的意义在于，它并不是简单给出一个分数，反而更像是一种诊断型评估方法，能够帮助系统研发人员定位系统问题。因此这类方法更多地使用在对机器翻译系统的翻译能力进行分析上，是对 BLEU 等整体评价指标的一种很好的补充。

#### 4.3.4 多策略融合的评价方法

前面介绍的几种自动评价方法中，大多是从某个单一的角度比对机器译文与参考答案之间的相似度，例如 BLEU 更关注  $n$ -gram 是否命中、Meteor 更关注机器译文与参考答案之间的词对齐信息、WER、PER 与 TER 等方法只关注机器译文与参考译文之间的编辑距离，此外还有一些方法比较关注机器译文和参考译文在语法、句法方面的相似度。但无一例外的是，每种自动评价的关注点都是单一的，无法对译文质量进行全面、综合的评价。为了克服这种限制，研究员们提出了一些基于多策略融合的译文质量评估方法，以期提高自动评价与人工评价结果的一致性。

基于策略融合的自动评价方法往往会将多个基于词汇、句法和语义的自动评价方法融合在内，其中比较核心的问题是如何将多个评价方法进行合理地组合。目前提出的方法中颇具代表性的是使用参数化方式和非参数化方式对多种自动评价方法进行筛选和组合。

参数化组合方法的实现主要有两种方式：一种方式是广泛使用不同的译文质量评价作为特征，借助回归算法实现多种评价策略的融合<sup>[152, 153]</sup>；另一种方式则是对各种译文质量评价方法的结果进行加权求和，并借助机器学习算法更新内部的权重参

数，从而实现多种评价策略的融合<sup>[154]</sup>。

非参数化组合方法的思想与贪心算法异曲同工：将多个自动评价方法以与人工评价的相关度为标准进行降序排列，依次尝试将其加入最优策略集合中，如果能提高最优策略集合的“性能”，则将该自动评价方法加入最优策略集合中，否则不加入。其中最优策略集合的“性能”用QUEEN定义<sup>[155]</sup>。该方法是首次尝试使用非参数的组合方式将多种自动评价方法进行融合，也不可避免地存在一些瑕疵。一方面在评价最优策略集合性能时，对于一个源文需要至少三个参考答案；另一方面，这种“贪心”的组合策略很有可能会得到局部最优的组合。

与单一的译文评价方法相比，多策略融合的评价方法能够对机器译文从多角度进行综合评价，这显然是一个模拟人工评价的过程，因而多策略融合的评价结果也与人工评价结果更加接近。但是对于不同的语言，多策略融合的评价方法需要不断调整最优策略集合或是调整组合方法内部的参数才能达到最佳的评价效果，这个过程势必要比单一的自动评价方法更繁琐些。

#### 4.3.5 译文多样性

在自然语言中，由于句子的灵活排序和大量同义词的存在，导致同一个源语言句子可能对应几百个合理的目标语言译文，甚至更多。然而上文提到的几种人工评价仅仅比较机器译文与有限数量的参考答案之间的差距，得出的评价结果往往会低估了机器译文的质量。为了改变这种窘况，比较直观的想法是增大参考答案集或是直接比较机器译文与参考答案在词法、句法和语义等方面的差距。

##### 1. 增大参考答案集

BLEU、Meteor、TER等自动评价方法的结果往往与人工评价结果存在差距。这些自动评价方法直接比对机器译文与有限数量的参考答案之间的“外在差异”，由于参考答案集可覆盖的人类译文数量过少，当机器译文本来十分合理但却未被包含在参考答案集中时，其质量就会被过分低估。

针对这个问题，HyTER自动评价方法致力于得到所有可能译文的紧凑编码，从而实现自动评价过程中访问所有合理的译文<sup>[156]</sup>。这种评价方法的原理非常简单直观：

- 通过注释工具标记出一个短语的所有备选含义（同义词）并存储在一起作为一个同义单元。可以认为每个同义单元表达了一个语义概念。在生成参考答案时，通过对某参考答案中的短语用同义单元进行替换生成一个新的参考答案。例如，将中文句子“对提案的支持率接近于0”翻译为英文，同义单元有以下几种：

[THE-SUPPORT-RATE]:

<the level of approval; the approval level; the approval rate ; the support rate>

[CLOSE-TO]:

<close to; about equal to; practically>

- 通过已有同义单元和附加单词的组合用于覆盖更大的语言片段。在生成参考答案时就是采用这种方式不断覆盖更大的语言片段，直到将所有可能的参考答案覆盖进去。例如可以将短语 [THE-SUPPORT-RATE] 与 “the proposal” 组合为 “[THE-SUPPORT-RATE] for the proposal”。
- 利用同义单元的组合将所有所有合理的人类译文都编码出来。将中文句子“对提案的支持率接近于 0” 翻译为英文，图4.6展示了其参考答案的编码结果。

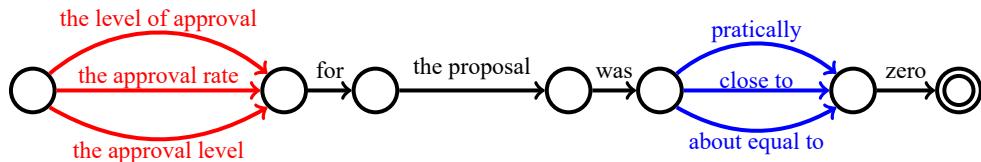


图 4.6 HyTER 中参考答案集的表示方式

从图4.6中可以看出，HyTER 方法通过构造同义单元的方式，可以列举出译文中每个片段的所有可能的表达方式，从而增大参考答案的数量，图4.6中的每一条路径都代表一个参考答案。但是这种对参考答案集的编码方式存在问题，同义单元之间的组合往往存在一定的限制关系<sup>[157]</sup>，使用 HyTER 方法会导致参考答案集中包含有错误的参考答案。

**实例 4.7** 将中文“市政府批准了一项新规定”分别翻译为英语和捷克语，使用 HyTER 构造的参考答案集分别如图4.7(a) 和 (b) 所示<sup>[157]</sup>：

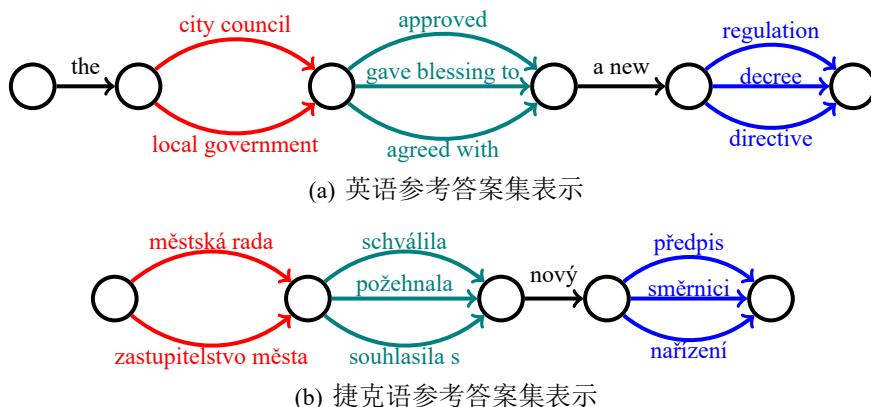


图 4.7 使用 HyTER 构造的参考答案集

但是在捷克语中主语 “městská rada” 或是 “zastupitelstvo města”的性别必须由动词来反映，那么上述捷克语的参考答案集中有部分存在语法错误。为了避免此类现象的出现，研究人员在同义单元中加入了将同义单元组合在一起必须满足的限制条件<sup>[157]</sup>，从而在增大参考答案集的同时确保了每个参考答案的准确性

将参考答案集扩大后，可以继续沿用 BLEU 或 NIST 等基于  $n$  元语法的方法进

行自动评价，但是传统方法往往会忽略多重参考答案中的重复信息，于是对每个  $n$  元语法进行加权的自动评价方法被提出<sup>[158]</sup>。该方法根据每个  $n$  元语法单元的长度、在参考答案集中出现的次数、被虚词（如“the”，“by”，“a”等）分开后的分散度等方面，确定其在计算最终分数时所占的权重。以 BLEU 方法为例（4.3.1节），可以将式(4.7)改写为：

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^N w_n \cdot \log(I_n \cdot P_n)\right) \quad (4.13)$$

$$I_n = n\text{-gram}_{\text{diver}} \cdot \log\left(n + \frac{M}{\text{count}_{\text{ref}}}\right) \quad (4.14)$$

其中， $I_n$  即为某个  $n$  元语法单元分配的权重， $M$  为参考答案集中出现该  $n$ -gram 中的参考答案数量， $\text{count}_{\text{ref}}$  为参考答案集大小。 $n\text{-gram}_{\text{diver}}$  为该  $n$ -gram 的分散度，用  $n$ -gram 种类数量与语法单元总数的比值计算。

需要注意的是，HyTER 方法对参考译文的标注有特殊要求，因此需要单独培训译员并开发相应的标注系统。这在一定程度上也增加了该方法被使用的难度。

## 2. 利用分布式表示进行质量评价

**词嵌入**（Word Embedding）技术是近些年自然语言处理中的重要成果，其思想是把每个单词映射为多维实数空间中的一个点（具体表现为一个实数向量），这种技术也被称作单词的**分布式表示**（Distributed Representation）。在这项技术中，单词之间的关系可以通过空间的几何性质进行刻画，意义相近的单词之间的欧式距离也十分相近（单词分布式表示的具体内容，将在书的第九章详细介绍，在此不再赘述）。

受词嵌入技术的启发，研究人员尝试借助参考答案和机器译文的分布式表示来进行译文质量评价，为译文质量评价提供了新思路。在自然语言的上下文中，表示是与每个单词、句子或文档相关联的数学对象。这个对象通常是一个向量，其中每个元素的值在某种程度上描述了相关单词、句子或文档的语义或句法属性。基于这个想法，研究人员提出了**分布式表示评价度量**（Distributed Representations Evaluation Metrics, DREEM）<sup>[159]</sup>。这种方法将单词或句子的分布式表示映射到连续的低维空间，发现在该空间中，具有相似句法和语义属性的单词彼此接近，类似的结论也出现在相关工作中，如参考文献[72, 160, 161]所示。而这个特点可以被应用到译文质量评估中。

在 DREEM 中，分布式表示的选取是一个十分关键的问题，理想的情况下，分布式表示应该涵盖句子在词汇、句法、语法、语义、依存关系等各个方面信息。目前常见的分布式表示方式如表4.2所示。除此之外，还可以通过词袋模型、循环神经网络等将词向量表示转换为句子向量表示。

DREEM 方法中选取了能够反映句子中使用的特定词汇的 One-hot 向量、能够反映词汇信息的词嵌入向量<sup>[72]</sup>、能够反映句子的合成语义信息的**递归自动编码**（Recur-

sive Auto-encoder Embedding, RAE)，这三种表示级联在一起，最终形成句子的向量表示。在得到机器译文和参考答案的上述分布式表示后，利用余弦相似度和长度惩罚对机器译文质量进行评价。机器译文  $o$  和参考答案  $g$  之间的相似度如公式(4.15)所示，其中  $v_i(o)$  和  $v_i(g)$  分别是机器译文和参考答案的向量表示中的第  $i$  个元素， $N$  是向量表示的维度大小。

$$\cos(t, r) = \frac{\sum_{i=1}^N v_i(o) \cdot v_i(g)}{\sqrt{\sum_{i=1}^N v_i^2(o)} \sqrt{\sum_{i=1}^N v_i^2(g)}} \quad (4.15)$$

表 4.2 常见的单词及句子分布表示

单词分布表示	句子分布表示
One-hot 词向量	RAE 编码 <sup>[160]</sup>
Word2Vec 词向量 <sup>[162]</sup>	Doc2Vec 向量 <sup>[163]</sup>
Prob-fasttext 词向量 <sup>[164]</sup>	ELMO 预训练句子表示 <sup>[165]</sup>
GloVe 词向量 <sup>[166]</sup>	GPT 句子表示 <sup>[124]</sup>
ELMO 预训练词向量 <sup>[165]</sup>	BERT 预训练句子表示 <sup>[123]</sup>
BERT 预训练词向量 <sup>[123]</sup>	Skip-thought 向量 <sup>[167]</sup>

在此基础上，DREEM 方法还引入了长度惩罚项，对与参考答案长度相差太多的机器译文进行惩罚，长度惩罚项如公式(4.16)所示，其中  $l_o$  和  $l_g$  分别是机器译文和参考答案长度：

$$\text{BP} = \begin{cases} \exp(1 - l_g/l_o) & l_o < l_g \\ \exp(1 - l_o/l_g) & l_o \geq l_g \end{cases} \quad (4.16)$$

机器译文的最终得分如下，其中  $\alpha$  是一个需要手动设置的参数：

$$\text{score}(o, g) = \cos^\alpha(o, g) \times \text{BP} \quad (4.17)$$

本质上，分布式表示是一种对句子语义的一种统计表示。因此，它可以帮助评价系统捕捉一些从简单的词或者句子片段中不易发现的现象，进而进行更深层的句子匹配。

在 DREEM 方法取得成功后，基于词嵌入的词对齐自动评价方法被提出<sup>[168]</sup>，该方法中先得到机器译文与参考答案的词对齐关系后，通过对齐关系中两者的词嵌入相似度来计算机器译文与参考答案的相似度，公式如(4.18)。其中， $o$  是机器译文， $g$  是参考答案， $m$  表示译文  $o$  的长度， $l$  表示参考答案  $g$  的长度，函数  $\varphi(o, g, i, j)$  用来

计算  $o$  中第  $i$  个词和  $g$  中第  $j$  个词之间对齐关系的相似度。:

$$\text{ASS}(o, g) = \frac{1}{m \cdot l} \sum_{i=1}^m \sum_{j=1}^l \varphi(o, g, i, j) \quad (4.18)$$

此外，将分布式表示与相对排序融合也是一个很有趣的想法<sup>[169]</sup>，在这个尝试中，研究人员利用分布式表示提取参考答案和多个机器译文中的句法信息和语义信息，利用神经网络模型对多个机器译文进行排序。

在基于分布式表示的这类译文质量评价方法中，译文和参考答案的所有词汇信息和句法语义信息都被包含在句子的分布式表示中，克服了单一参考答案的限制。但是同时也带来了新的问题，一方面将句子转化成分布式表示使评价过程变得不那么具有可解释性，另一方面分布式表示的质量也会对评价结果有较大的影响。

### 4.3.6 相关性与显著性

近年来，随着多种有参考答案的自动评价方法的提出，译文质量评价已经渐渐从大量的人力工作中解脱转而依赖于自动评价技术。然而，一些自动评价结果的可靠性、置信性以及参考价值仍有待商榷。自动评价结果与人工评价结果的相关性以及其自身的统计显著性，都是衡量其可靠性、置信性以及参考价值的重要标准。

#### 1. 自动评价与人工评价的相关性

**相关性** (Correlation) 是统计学中的概念，当两个变量之间存在密切的依赖或制约关系，但却无法确切地表示时，可以认为两个变量之间存在“相关关系”，并往往用“相关性”作为衡量关系密切程度的标准<sup>[170]</sup>。对于相关关系，虽然无法求解两个变量之间确定的函数关系，但是通过大量的观测数据，能够发现变量之间存在的统计规律性，而“相关性”也同样可以利用统计手段获取。

在机器译文质量评价工作中，相比人工评价，有参考答案的自动评价具有效率高、成本低的优点，因而广受机器翻译系统研发人员青睐。在这种情况下，自动评价结果的可信度一般取决于它们与可靠的人工评价之间的相关性。随着越来越多有参考答案的自动评价方法的提出，“与人工评价之间的相关性”也被视为衡量一种新的自动评价方法是否可靠的衡量标准。

很多研究工作中都曾对 BLEU、NIST 等有参考答案的自动评价与人工评价的相关性进行研究和讨论，其中也有很多工作对“相关性”的统计过程作过比较详细的阐述。在“相关性”的统计过程中，一般是分别利用人工评价方法和某种有参考答案的自动评价方法对若干个机器翻译系统的输出进行等级评价<sup>[171]</sup> 或是相对排序<sup>[172]</sup>，从而对比两种评价手段的评价结果是否一致。该过程中的几个关键问题可能会对最终结果产生影响。

- **源语言句子的选择。**由于机器翻译系统一般以单句作为翻译单元，因而评价过程

中涉及的源语言句子是脱离上下文语境的单句<sup>[171]</sup>。

- **人工评估结果的产生。**人工评价过程中采用只提供标准高质量参考答案的单语评价方法，由多位评委对译文质量做出评价后进行平均作为最终的人工评价结果<sup>[171]</sup>。
- **自动评价中参考答案的数量。**在有参考答案的自动评价过程中，为了使评价结果更加准确，一般会设置多个参考答案。参考答案数量的设置会对自动评价与人工评价的相关性产生影响，也有很多工作对此进行了研究。例如人们发现有参考答案的自动评价方法在区分人类翻译和机器翻译时，设置 4 个参考答案的区分效果远远高于 2 个参考答案<sup>[173]</sup>；也有人曾专注于研究怎样设置参考答案数量能够产生最高的相关性<sup>[174]</sup>。
- **自动评价中参考答案的质量。**从直觉上，自动评价中参考答案的质量一般会影响最终的评价结果，从而对相关性的计算产生影响。然而，有相关实验表明，只要参考答案的质量不是过分低劣，很多情况下自动评价都能得到相同的评价结果<sup>[175]</sup>。

目前在机器译文质量评价的领域中，有很多研究工作尝试比较各种有参考答案的自动评价方法（主要以 BLEU、NIST 等基于  $n$ -gram 的方法为主）与人工评价方法的相关性。整体来看，这些方法与人工评价具有一定的相关性，自动评价结果能够较好地反映译文质量<sup>[171, 176]</sup>。

但是也有相关研究指出，不应该对有参考答案的自动评价方法过于乐观，而应该存谨慎态度，因为目前的自动评价方法对于流利度的评价并不可靠，同时参考答案的体裁和风格往往会对自动评价结果产生很大影响<sup>[173]</sup>。同时，有研究人员提出，机器翻译研究过程中，在忽略实际示例翻译的前提下，BLEU 分数的提高并不意味着翻译质量的真正提高，而在一些情况下，为了实现翻译质量的显著提高，并不需要提高 BLEU 分数<sup>[177]</sup>。

## 2. 自动评价方法的统计显著性

使用自动评价的目的是比较不同系统之间性能的差别。比如，对某个机器翻译系统进行改进后，它的 BLEU 值从 40.0% 提升到 40.5%，能否说改进后的系统真的比改进前的翻译品质更好吗？实际上，这也是统计学中经典的**统计假设检验**（Statistical Hypothesis Testing）问题<sup>[178]</sup>。统计假设检验的基本原理是：如果对样本总体的某种假设是真的，那么不支持该假设的小概率事件几乎是不可能发生的；一旦这种小概率事件在某次试验中发生了，那就有理由拒绝原始的假设。例如，对于上面提到了例子，可以假设：

- **原始假设：**改进后比改进前翻译品质更好；
- **小概率事件**（备择假设）：改进后和改进前比，翻译品质相同甚至更差。

统计假设检验的流程如图4.8所示。其中的一个关键步骤是检验一个样本集合中是否发生了小概率事件。但是，怎样才算是小概率事件呢？比如，可以定义概率不超过0.1的事件就是小概率事件，甚至可以定义这个概率为0.05、0.01。通常，这个概率被记为 $\alpha$ ，也就是常说的**显著性水平**（Significance Level）。而显著性水平更准确的定义是“去真错误”的概率，即：原假设为真但是拒绝了它的概率。

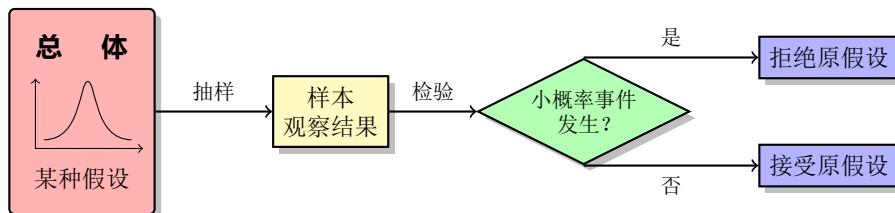


图 4.8 统计假设检验的流程

回到机器翻译的问题中来。一个更加基础的问题是：一个系统评价结果的变化在多大范围内是不显著的。利用假设检验的原理，这个问题可以被描述为：评价结果落在 $[x-d, x+d]$ 区间的置信度是 $1-\alpha$ 。换句话说，当系统性能落在 $[x-d, x+d]$ 外，就可以说这个结果与原始的结果有显著性差异。这里 $x$ 通常是系统译文的BLEU计算结果， $[x-d, x+d]$ 是其对应的置信区间。而 $d$ 和 $\alpha$ 有很多计算方法，比如，如果假设评价结果服从正态分布，可以简单的计算 $d$ 。

$$d = t \frac{s}{\sqrt{n}} \quad (4.19)$$

其中， $s$ 是标准差， $n$ 是样本数。 $t$ 是一个统计量，它与假设检验的方式、显著性水平、样本数量有关。

而机器翻译评价使用假设检验的另一个问题是如何进行抽样。需要注意的是，这里的样本是指一个机器翻译的测试集，因为BLEU等指标都是在整个测试集上计算的，而非简单的通过句子级评价结果进行累加。为了保证假设检验的充分性，需要构建多个测试集，以模拟从所有潜在的测试集空间中采样的行为。

最常用的方法是使用**Bootstrap重采样技术**<sup>[179]</sup>从一个固定测试集中采样不同的句子组成不同的测试集，之后在这些测试集上进行假设检验<sup>[180]</sup>。此后，有工作指出了Bootstrap重采样方法存在隐含假设的不合理之处，并提出了使用近似随机化<sup>[181]</sup>方法计算自动评价方法统计显著性<sup>[182]</sup>。另有研究工作着眼于研究自动评价结果差距大小、测试集规模、系统相似性等因素对统计显著性的影响，以及在不同领域的测试语料中计算的统计显著性是否具有通用性的问题<sup>[183]</sup>。

在所有自然语言处理系统的结果对比中，显著性检验是十分必要的。很多时候不同系统性能的差异性很小，因此需要确定一些微小的进步是否是“真”的，还是只是一些随机事件。但是从实践的角度看，当某个系统性能的提升达到一个绝对值，这种性能提升效果往往是显著的。比如，在机器翻译，BLEU提升0.5%一般都是比

较明显的进步。也有研究对这种观点进行了论证，也发现其中具有一定的科学性<sup>[183]</sup>。因此，在机器翻译系统研发中类似的方式也是可以采用的。

## 4.4 无参考答案的自动评价

无参考答案自动评价在机器翻译领域又被称作**质量评估**（Quality Estimation, QE）。与传统的译文质量评价方法不同，质量评估旨在不参照标准译文的情况下，对机器翻译系统的输出在单词、短语、句子、文档等各个层次进行评价。

人们对于无参考答案自动评价的需求大多来源于机器翻译的实际应用。例如，在机器翻译的译后编辑过程中，译员不仅仅希望了解机器翻译系统的整体翻译质量，还需要了解该系统在某个句子上的表现如何：该机器译文的质量是否很差？需要修改的内容有多少？是否值得进行后编辑？这时，译员更加关注系统在单个数据点上（比如一段话）的可信度而非系统在测试数据集上的平均质量。这时，太多的人工介入就无法保证使用机器翻译所带来的高效性，因此在机器翻译输出译文的同时，需要质量评估系统给出对译文质量的预估结果。这些需求也促使研究人员在质量评估问题上投入了更多的研究力量。包括WMT、CCMT等知名机器翻译评测中也都设置了相关任务，受到了业界的关注。

### 4.4.1 质量评估任务

质量评估任务本质上是通过预测一个能够反映评价单元的质量标签，在各个层次上对译文进行质量评价。在上文中已经提到，质量评估任务通常被划分为单词级、短语级、句子级和文档级，在接下来的内容中，将对各个级别的任务进行更加详细的介绍。

#### 1. 单词级质量评估

机器翻译系统在翻译某个句子时，会出现各种类型的错误，这些错误多是一些单词翻译问题，例如单词出现歧义、单词漏译、单词错译、词形转化错误等等。单词级质量评价以单词为评估单元，目的是确定译文句子中每个单词的所在位置是否存在翻译错误和单词漏译现象。

单词级质量评估任务可以被定义为：参照源语言句子，以单词为评价单位，自动标记出机器译文中的错误。其中的“错误”包括单词错译、单词词形错误、单词漏译等。在单词级质量评估任务中，输入是机器译文和源语言句子，输出是一系列标签序列，即图4.9中的Source tags、MT tags、Gap tags，标签序列中的每个标签对应翻译中的每个单词（或其间隙），并表明该位置是否出现错误。

下面以实例4.8为例介绍该任务的具体内容，在实例4.8中加入后编辑结果是方便读者理解任务内容，实际上质量评估任务在预测质量标签时并不依赖后编辑结果：

##### 实例 4.8 单词级质量评估任务

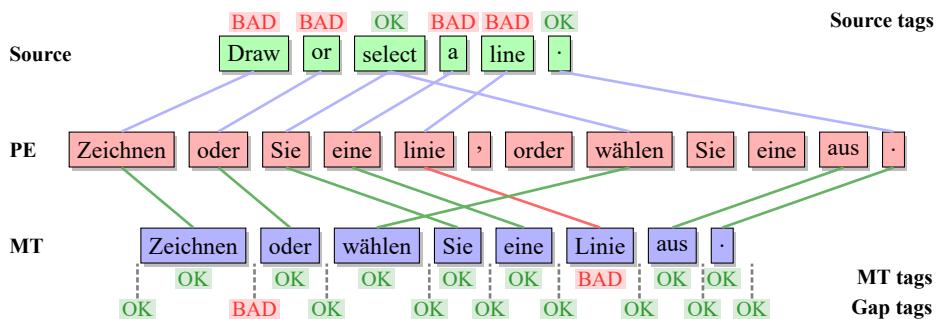


图 4.9 单词级质量评估任务示意图

源句 (Source): Draw or select a line. (英语)

机器译文 (MT): Zeichnen oder wählen Sie eine Linie aus. (德语)

后编辑结果 (PE): Zeichnen oder Sie eine Linie, oder wählen Sie eine aus. (德语)

单词级质量评估主要通过以下三类错误评价译文好坏:

- 找出译文中翻译错误的单词。**单词级质量评估任务要求预测一个与译文等长的质量标签序列，该标签序列反映译文端的每个单词是否能够准确表达出其对应的源端单词的含义，若是可以，则标签为“OK”，反之则为“BAD”。图4.9中的连线表示单词之间的对齐关系，图4.9中的 MT tags 即为该过程中需要预测的质量标签序列。
- 找出原文中导致翻译错误的单词。**单词级质量评估任务还要求预测一个与源文等长的质量标签序列，该标签序列反映源文端的每个单词是否会导致本次翻译出现错误，若是不会，则标签为“OK”，反之则为“BAD”。图4.9中的 Source tags 即为该过程中的质量标签序列。在具体应用时，质量评估系统往往先预测译文端的质量标签序列，并根据原文与译文之间的对齐关系，推测源端的质量标签序列。
- 找出在翻译句子时出现漏译现象的位置。**单词级质量评估任务同时也要求预测一个能够捕捉到漏译现象的质量标签序列，在译文端单词的两侧位置进行预测，若某位置未出现漏译，则该位置的质量标签为“OK”，否则为“BAD”。图4.9中的 Gap tags 即为该过程中的质量标签序列。为了检测句子翻译中的漏译现象，需要在译文中标记缺口，即译文中的每个单词两边都各有一个“GAP”标记，如图4.9所示。

## 2. 短语级质量评估

短语级质量评估可以看做是单词级质量评估任务的扩展：机器翻译系统引发的错误往往都是相互关联的，解码过程中某个单词出错会导致更多的错误，特别是在其局部上下文当中，以单词的“局部上下文”为基本单元进行质量评估即为短语级

质量评估。

短语级质量评估与单词级质量评估类似，其目标是找出短语中翻译错误、短语内部语序问题及漏译问题。短语级质量评估任务可以被定义为：以若干个连续单词组成的短语为基本评估单位，参照源语言句子，自动标记出短语内部短语错误以及短语之间是否存在漏译。其中的短语错误包括短语内部单词的错译和漏译、短语内部单词的语序错误，而漏译问题则特指短语之间的漏译错误。在短语级质量评估任务中，输入是机器译文和源语言句子，输出是一系列标签序列，即图4.10中的 Phrase-target tags、Gap tags，标签序列中的每个标签对应翻译中的每个单词，并表明该位置是否出现错误。

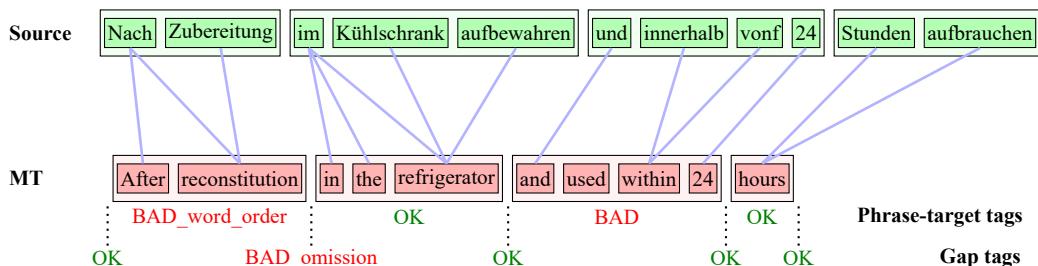


图 4.10 短语级质量评估任务示意图

下面以实例4.9为例介绍该任务的具体内容：

#### 实例 4.9 短语级质量评估任务（短语间用 || 分隔）

源句（Source）: Nach Zubereitung || im Kühlschrank aufbewahren || und innerhalb von 24 || Stunden aufbrauchen . (德语)

机器译文（MT）: After reconstitution || in the refrigerator || and used within 24 || hours . (英语)

短语级质量评估任务主要通过以下两类类错误评价译文好坏：

- **找出译文中翻译错误的短语。**要求预测出一个能够捕捉短语内部单词翻译错误、单词漏译以及单词顺序错误的标签序列。该序列中每个标签都对应着一个短语，若是短语不存在任何错误，则标签为“OK”；若是短语内部存在单词翻译错误和单词漏译，则标签为“BAD”；若短语内部的单词顺序存在问题，则标签为“BAD\_word\_order”。图4.10中的连线表示单词之间的对齐关系，蓝色虚线框标出了每个短语的范围，图4.10中的 Phrase-target tags 即为该过程中需要预测的质量标签序列。
- **找出译文中短语之间漏译错误。**短语级质量评估任务同时也要求预测一个能够捕捉到短语间的漏译现象的质量标签序列，在译文端短语的两侧位置进行预测，若某位置未出现漏译，则该位置的质量标签为“OK”，否则为“BAD\_omission”。图4.10中的 Gap tags 即为该过程中的质量标签序列。

为了检测句子翻译中的漏译现象，参与者也被要求在译文中短语之间标记缺口，即译文中的每对短语之间都有两个“GAP”标记，一个在短语前面，一个在短语后面，与单词级类似。

### 3. 句子级质量评估

迄今为止，质量评估的大部分工作都集中在句子层次的预测上，这是因为多数情况下机器翻译系统的处理都是逐句进行，系统用户也总是每次翻译一个句子或是以句子为单位组成的文本块（段落、文档等），因此以句子作为质量评估的基本单元是很自然的。

句子级质量评估的目标是生成能够反映译文句子整体质量的标签——可以是离散型的表示某种质量等级的标签，也可以是连续型的基于评分的标签。虽然以不同的标准进行评估，同一个译文句子的质量标签可能有所不同，但可以肯定的是句子的最终质量绝不是句子中单词质量的简单累加。因为与词级的质量评估相比，句子级质量评估也会关注是否保留源句的语义、译文的语义是否连贯、译文中的单词顺序是否合理等因素。

句子级质量系统需要根据某种评价标准，通过建立预测模型来生成一个反映句子质量的标签。人们可以根据句子翻译的目的、后编辑的工作难度、是否达到发表要求或是是否能让非母语者读懂等各个角度、各个标准去设定句子级质量评估的标准。句子级质量评估任务有多种形式：

- **区分“人工翻译”和“机器翻译”。**在早期的工作中，研究人员试图训练一个能够区分人工翻译和机器翻译的二分类器完成句子级的质量评估<sup>[184]</sup>，将被分类器判断为“人工翻译”的机器译文视为优秀的译文，将被分类器判断为“机器翻译”的机器译文视为较差的译文。一方面，这种评估方式不够直观，另一方面，这种评估方式并不十分准确，因为通过人工比对发现很多被判定为“机器翻译”的译文具有与人们期望的人类翻译相同质量水平。
- **预测反映译文句子质量的“质量标签”。**在同一时期，研究人们也尝试使用人工为机器译文分配能够反映译文质量的标签<sup>[185]</sup>，例如“不可接受”、“一定程度上可接受”、“可接受”、“理想”等类型的质量标签，同时将获取机器译文的质量标签作为句子级质量评估的任务目标。
- **预测译文句子的相对排名。**当相对排序（详见4.2.2节）的译文评价方法被引入后，给出机器译文的相对排名成为句子级质量评估的任务目标。
- **预测译文句子的后编辑工作量。**在最近的研究中，句子级的质量评估一直在探索各种类型的离散或连续的后编辑标签。例如，通过测量以秒为单位的后编辑时间对译文句子进行评分；通过测量预测后编辑过程所需的击键数对译文句子进行评分；通过计算**人工译后编辑距离**（Human Translation Error Rate, HTER），即在后编辑过程中编辑（插入/删除/替换）数量与参考翻译长度的占比率对译文

句子进行评分。HTER 的计算公式为：

$$\text{HTER} = \frac{\text{编辑操作数目}}{\text{翻译后编辑结果长度}} \quad (4.20)$$

这种质量评估方式往往以单词级质量评估为基础，在其结果的基础上进行计算。以实例4.8中词级质量评估结果为例，与编辑后结果相比较，机器翻译译文中有四处漏译（“Mit”、“können”、“Sie”、“einzelne”）、三处误译（“dem”、“Scharfzeichner”、“scharfzeichnen”分别被误译为“Der”、“Schärfen-Werkezug”、“Schärfer”）、一处多译（“erscheint”），因而需要进行4次插入操作、3次替换操作和1次删除操作，而最终译文长度为12，则有  $\text{HTER} = (4+3+1)/12 = 0.667$ 。需要注意的是，即便这种评估方式以单词级质量评估为基础，也不意味这句子级质量评估只是在单词级质量评估的结果上通过简单的计算来获得其得分，在实际研究中，常将其视为一个回归问题，利用大量数据学习其评分规则。

#### 4. 文档级质量评估

文档级质量评估的主要目的是对机器翻译得到的整个译文文档进行打分。文档级质量评估中，“文档”很多时候并不单单指一整篇文档，而是指包含多个句子的文本，例如包含3到5个句子的段落或是像新闻文章一样的长文本。

传统的机器翻译任务中，往往以一个句子作为输入和翻译的单元，而忽略了文档中句子之间的联系，这可能会使文档的论述要素受到影响，最终导致整个文档的语义不连贯。如实例4.10所示，在第二句中“he”原本指代第一句中的“housewife”，这里出现了错误，但这种错误在句子级的质量评估中并不能被发现。

##### 实例 4.10 文档级质量评估任务

上文信息：A **housewife** won the first prize in the supermarket's anniversary celebration .

机器译文：A few days ago, **he** contacted the News Channel and said that the supermarket owner refused to give **him** the prize .

在文档级质量评估中，有两种衡量文档译文的质量的方式：

- **阅读理解测试得分情况**。以往衡量文档译文质量的主要方法是采用理解测试<sup>[186]</sup>，即利用提前设计好的与文档相关的阅读理解题目（包括多项选择题类型和问答题类型）对母语为目标语言的多个测试者进行测试，将代表测试者在给定文档上的问卷中的所有问题所得到的分数作为质量标签。
- **后编辑工作量**。最近的研究工作中，多是采用对文档译文进行后编辑的工作量评估文档译文的质量。为了准确获取文档后编辑的工作量，两阶段后编辑方法被提出<sup>[187]</sup>，即第一阶段对文档中的句子单独在无语境情况下进行后编辑，第二阶段将所有句子重新合并成文档后再进行后编辑。两阶段中后编辑工作量的总

和越多，意味着文档译文质量越差。

在文档级质量评估任务中，需要对译文文档做一些更细粒度的注释，注释内容包括错误位置、错误类型和错误的严重程度，最终在注释的基础上对译文文档质量进行评估。

与更细粒度的词级和句子级的质量评价相比，文档级质量评估更加复杂。其难点之一在于文档级的质量评估过程中需要根据一些主观的质量标准去对文档进行评分，例如在注释的过程中，对于错误的严重程度并没有严格的界限和规定，只能靠评测人员主观判断，这就意味着随着出现主观偏差的注释的增多，文档级质量评估的参考价值会大打折扣。另一方面，根据所有注释（错误位置、错误类型及其严重程度）对整个文档进行评分本身就具有不合理性，因为译文中有些在抛开上下文语境时可以并判定为“翻译得不错的”单词和句子，一旦被放在上下文语境中就可能变得不合理，而某些在无语境条件下看起来翻译得“糟糕透了”的单词和句子，一旦被放在文档中的语境中可能会变得恰到好处。此外，构建一个质量评测模型势必需要大量的标注数据，而文档级质量评测所需要的带有注释的数据的获取代价相当高。

实际上，文档级质量评估与其它文档级自然语言处理任务面临的问题是一样的。由于数据稀缺，无论是系统研发，还是结果评价都面临很大挑战。这些问题也会在本书的第十六章和第十七章进行讨论。

#### 4.4.2 构建质量评估模型

不同于有参考答案的自动评价，质量评估方法的实现较为复杂。质量评估可以被看作是一个统计推断问题，即：如何根据以往得到的经验对从未见过的机器译文的质量做出预测。从这个角度说，质量评估和机器翻译问题一样，都需要设计模型进行求解，而无法像 BLEU 计算一样直接使用指标性的公式计算就能得到结果。

实际上，质量评估的灵感最初来源于语音识别中的置信度评价，所以最初研究人员也尝试通过翻译模型中的后验概率来直接评价翻译质量<sup>[188]</sup>，然而仅仅依靠概率值作为评价标准显然是远远不够的，其效果也让人大失所望。之后，质量评估被定义为一个有监督的机器学习问题。这也形成了质量评估的新范式：使用机器学习算法利用句子的某种表示对译文质量进行评价。

研究人员将质量评估模型的基本框架设计为两部分：

- **表示/特征学习模块：**用于在数据中提取能够反映翻译结果质量的“特征”；
- **质量评估模块：**基于句子的表示结果，利用机器学习算法预测翻译结果的质量。

在传统机器学习的观点下，句子都是由某些特征表示的。因此需要人工设计能够对译文质量评估有指导性作用的特征<sup>[189, 190, 191, 192, 193]</sup>，常用的特征有：

- **复杂度特征：**反映了翻译一个源文的难易程度，翻译难度越大，译文质量低的可能性就越大；
- **流畅度特征：**反映了译文的自然度、流畅度、语法合理程度；

- **置信度特征:** 反映了机器翻译系统对输出的译文的置信程度；
- **充分度特征:** 反映了源文和机器译文在不同语言层次上的密切程度或关联程度。

随着深度学习技术的发展，另一种思路是使用表示学习技术生成句子的分布式表示，并在此基础上利用神经网络自动提取高度抽象的句子特征<sup>[194, 195, 196]</sup>，这样就避免了人工设计特征所带来的时间以及人工代价，同时表示学习所得到的分布式表示可以涵盖更多人工设计难以捕获到的特征，更加全面地反映句子的特点，因此在质量评估任务上也取得了很好的效果<sup>[197, 198, 199, 200, 201]</sup>。比如，最近的一些工作中大量使用了神经机器翻译模型来获得双语句子的表示结果，并用于质量评估<sup>[202, 203, 204, 205]</sup>。这样做的好处在于，质量评估可以直接复用机器翻译的模型，从某种意义上降低了质量评估系统开发的代价。此外，随着近几年各种预训练模型的出现，使用预训练模型来获取用于质量评估的句子表示也成为一大流行趋势，这种方法大大减少了质量评估模型自身的训练时间，在该领域内的表现也十分亮眼<sup>[206, 207, 208]</sup>。关于表示学习、神经机器翻译、预训练模型的内容在第九章和第十章会有进一步介绍。

在得到句子表示之后，可以使用质量评估模块对译文质量进行预测。质量评估模型通常由回归算法或分类算法实现：

- 句子级和文档级质量评估目前大多通过回归算法实现。由于在句子级和文档级的质量评估中，标签是使用连续数字（得分情况）表示的，因此回归算法是最合适的选择。最初的工作中，研究人员们多采用传统的机器学习回归算法<sup>[189, 192, 209]</sup>，而近年来，研究人员则更青睐于使用神经网络方法进行句子级和文档级质量评估；
- 单词级和短语级质量评估多由分类算法实现。在单词级质量评估任务中，需要对每个位置的单词标记“OK”或“BAD”，这对应了经典的二分类问题，因此可以使用分类算法对其进行预测。自动分类算法在第三章已经涉及，质量评估中直接使用成熟的分类器即可。此外，使用神经网络方法进行分类也是不错的选择。

值得一提的是，近年来的研究工作中，模型集成已经成为了提高质量评估模型性能的重要手段之一，该方法能够有效减缓使用单一模型时可能存在的性能不稳定，提升译文质量评估模型在不同测试集下的鲁棒性，最终获得更高的预测准确度<sup>[195, 204, 205, 206, 210]</sup>。

#### 4.4.3 质量评估的应用场景

很多情况下参考答案是很难获取的，例如，在很多人工翻译生产环节中，译员的任务就是“创造”翻译。如果已经有了答案，译员根本不需要工作，也谈不上应用机器翻译技术了。这时更多的是希望通过质量评估帮助译员有效地选择机器翻译结果。质量评估的应用场景还有很多，例如：

- **判断人工后编辑的工作量。**人工后编辑工作中有两个不可避免的问题：1) 待编辑的机器译文是否值得改？2) 待编辑的机器译文需要修改哪里？对于一些质量较差的机器译文来说，人工重译远远比修改译文的效率高，后编辑人员可以借助质量评估系统提供的指标筛选出值得进行后编辑的机器译文，另一方面，质量评估模型可以为每条机器译文提供错误内容、错误类型、错误严重程度的注释，这些内容将帮助后编辑人员准确定位到需要修改的位置，同时在一定程度上提示后编辑人员采取何种修改策略，势必能大大减少后编辑的工作内容。
- **自动识别并更正翻译错误。**质量评估和**自动后编辑**（Automatic Post-editing, APE）也是很有潜力的应用方向。因为质量评估可以预测出错的位置，进而可以使用自动方法修正这些错误。但是，在这种应用模式中，质量评估的精度是非常关键的，因为如果预测错误可能会产生错误的修改，甚至带来整体译文质量的下降。
- **辅助外语交流和学习。**例如，在很多社交网站上，用户会利用外语进行交流。质量评估模型可以提示该用户输入的内容中存在的用词、语法等问题，这样用户可以重新对内容进行修改。甚至质量评估可以帮助外语学习者发现外语使用中的问题，例如，对于一个英语初学者，如果能提示他/她写的句子中的明显错误，对他/她的外语学习是非常有帮助的。

需要注意的是，质量评估的应用模式还没有完全得到验证。这一方面是由于，质量评估的应用非常依赖与人的交互过程。但是，改变人的工作习惯是很困难的，因此质量评估系统在实际场景中的应用往往需要很长时间，或者说人也要适应质量评估系统的行为。另一方面，质量评估的很多应用场景还没有完全被发掘出来，需要更长的时间进行探索。

## 4.5 小结及拓展阅读

译文的质量评价是机器翻译研究中不可或缺的环节。与其他任务不同，由于自然语言高度的歧义性和表达方式的多样性，机器翻译的参考答案本身就不唯一。此外，对译文准确、全面的评价准则很难制定，导致译文质量的自动评价变得异常艰难，因此也成为了广受关注的研究课题。本章系统阐述了译文质量评估的研究现状和主要挑战。从人类参与程度和标注类型两个角度对译文质量评价中的经典方法进行介绍，力求让读者对领域内的经典及热点内容有更加全面的了解。不过，由于篇幅限制笔者无法对译文评价的相关工作进行面面俱到的描述，还有很多研究方向值得关注：

- 基于句法和语义的机器译文质量自动评价方法。本章内容中介绍的自动评价多是基于表面字符串形式判定机器翻译结果和参考译文之间的相似度，而忽略了更抽象的语言层次的信息。基于句法和语义的机器译文质量自动评价方法在评

价度量标准中加入能反映句法信息<sup>[211]</sup> 和语义信息<sup>[212]</sup> 的相关内容，通过比较机器译文与参考答案之间的句法相似度和语义等价性<sup>[213]</sup>，能够大大提高自动评价与人工评价之间的相关性。其中句法信息往往能够对机器译文流利度方面的评价起到促进作用<sup>[211]</sup>，常见的句法信息包括语法成分<sup>[211]</sup>、依存关系<sup>[214, 215, 216]</sup> 等。语义信息则对机器翻译的充分性评价更有帮助<sup>[217, 218]</sup>，近年来也有很多用于机器译文质量评估的语义框架被提出，如 AM-FM<sup>[217]</sup>、XMEANT<sup>[219]</sup> 等。

- 对机器译文中的错误分析和错误分类。无论是人工评价还是自动评价手段，其评价结果只能反映机器翻译系统性能，而无法确切表明机器翻译系统的优点和弱点是什么、系统最常犯什么类型的错误、一个特定的修改是否改善了系统的某一方面、排名较差的系统是否在任何方面都优于排名较好的系统等等。对机器译文进行错误分析和错误分类有助于找出机器翻译系统中存在的主要问题，以便集中精力进行研究改进<sup>[220]</sup>。相关的研究工作中，一些致力于错误分类方法的设计，如手动的机器译文错误分类框架<sup>[220]</sup>、自动的机器译文错误分类框架<sup>[221]</sup>、基于语言学的错误分类方法<sup>[222]</sup> 以及目前被用作篇章级质量评估注释标准的 MQM 错误分类框架<sup>[223]</sup>；其他的研究工作则致力于对机器译文进行错误分析，如引入形态句法信息的自动错误分析框架<sup>[224]</sup>、引入词错误率（WER）和位置无关词错误率（PER）的错误分析框架<sup>[225]</sup>、基于检索的错误分析工具 tSEARCH<sup>[226]</sup> 等等。
- 译文质量的多角度评价。章节内主要介绍的几种经典方法如 BLEU、TER、METEOR 等，大都是从某个单一的角度计算机器译文和参考答案的相似性，如何对译文从多个角度进行综合评价是需要进一步思考的问题，4.3.4 节中介绍的多策略融合评价方法就可以看作是一种多角度评价方法，其思想是将各种评价方法下的译文得分通过某种方式进行组合，从而实现对译文的综合评价。译文质量多角度评价的另一种思路则是直接将 BLEU、TER、Meteor 等多种指标看做是某种特征，使用分类<sup>[227, 228]</sup>、回归<sup>[229]</sup>、排序<sup>[230]</sup> 等机器学习手段形成一种综合度量。此外，也有相关工作专注于多等级的译文质量评价，使用聚类算法将大致译文按其质量分为不同等级，并对不同质量等级的译文按照不同权重组合几种不同的评价方法<sup>[231]</sup>。
- 不同评价方法的应用场景有明显不同：人工评价主要用于需要对机器翻译系统进行准确的评估的场合。例如，在系统对比中利用人工评价方法对不同系统进行人工评价、给出最终排名，或上线机器翻译服务时对翻译品质进行详细的测试；有参考答案的自动评价则可以为机器翻译系统提供快速、相对可靠的评价。在机器翻译系统的快速研发过程中，一般都使用有参考答案的自动评价方法对最终模型的性能进行评估。有相关研究工作专注于在机器翻译模型的训练过程中利用评价信息（如 BLEU 分数）进行参数调优，其中比较有代表性的工作包括最小错误率训练<sup>[232]</sup>、最小风险训练<sup>[233, 234]</sup> 等。这部分内容可以参考第七章和

第十三章进行进一步阅读；无参考答案的质量评估主要用来对译文质量做出预测，经常被应用在一些无法提供参考译文的实时翻译场景中，例如人机交互过程、自动纠错、后编辑等<sup>[235]</sup>。

- 另一个比较值得关注的一个研究问题是如何使模型更加鲁棒，因为通常情况下，一个质量评估模型会受语种、评价策略等问题的约束，设计一个能应用于任何语种，同时从单词、短语、句子等各个等级对译文质量进行评估的模型是很有难度的。Biçici 等人最先关注质量评估的鲁棒性问题，并设计开发了一种与语言无关的机器翻译性能预测器<sup>[236]</sup>，此后又在该工作的基础上研究如何利用外在的、与语言无关的特征对译文进行句子级别的质量评估<sup>[191]</sup>，该项研究的最终成果是一个与语言无关，可以从各个等级对译文质量进行评估的模型——RTMs（Referential Translation Machines）<sup>[237]</sup>。



# 统计机器翻译

<b>5</b>	<b>基于词的机器翻译建模</b>	139
5.1	词在翻译中的作用	
5.2	一个简单实例	
5.3	噪声信道模型	
5.4	统计机器翻译的三个基本问题	
5.5	IBM 模型 1	
5.6	小结及拓展阅读	
<b>6</b>	<b>基于扭曲度和繁衍率的模型</b>	171
6.1	基于扭曲度的模型	
6.2	基于繁衍率的模型	
6.3	解码和训练	
6.4	问题分析	
6.5	小结及拓展阅读	
<b>7</b>	<b>基于短语的模型</b>	189
7.1	翻译中的短语信息	
7.2	数学建模	
7.3	短语抽取	
7.4	翻译调序建模	
7.5	翻译特征	
7.6	最小错误率训练	
7.7	栈解码	
7.8	小结及拓展阅读	
<b>8</b>	<b>基于句法的模型</b>	219
8.1	翻译中句法信息的使用	
8.2	基于层次短语的模型	
8.3	基于语言学句法的模型	
8.4	小结及拓展阅读	





## 5. 基于词的机器翻译建模

使用统计方法对翻译问题进行建模是机器翻译发展中的重要里程碑。这种思想也影响了当今的统计机器翻译和神经机器翻译范式。虽然技术不断发展，传统的统计模型已经不再“新鲜”，但它对于今天机器翻译的研究仍然有着重要的启示作用。在了解前沿、展望未来的同时，更要冷静地思考前人给我们带来了什么。基于此，这里将介绍统计机器翻译的开山之作——IBM 模型，它提出了使用统计模型进行翻译的思想，并在建模中引入了单词对齐这一重要概念。

IBM 模型由 Peter F. Brown 等人于上世纪九十年代初提出<sup>[10]</sup>。客观地说，这项工作的视野和对问题的理解，已经超过当时很多人所能看到的东西，其衍生出来的一系列方法和新的问题还被后人花费将近 10 年的时间来进行研究与讨论。时至今日，IBM 模型中的一些思想仍然影响着很多研究工作。本章将重点介绍一种简单的基于单词的统计翻译模型（IBM 模型 1），以及在这种建模方式下的模型训练方法。这些内容可以作为后续章节中统计机器翻译和神经机器翻译建模方法的基础。

### 5.1 词在翻译中的作用

在翻译任务中，我们希望得到一个源语言到目标语言的翻译。对于人类来说这个问题很简单，但是让计算机做这样的工作却很困难。这里面临的第一个问题是：如何对翻译进行建模？从计算机的角度来看，这就需要把自然语言的翻译问题转换为计算机可计算的问题。

那么，基于单词的统计机器翻译模型又是如何描述翻译问题的呢？Peter F. Brown

等人提出了一个观点<sup>[10]</sup>: 在翻译一个句子时, 可以把其中的每个单词翻译成对应的目标语言单词, 然后调整这些目标语言单词的顺序, 最后得到整个句子的翻译结果, 而这个过程可以用统计模型来描述。尽管在人看来使用两个语言单词之间的对应进行翻译是很自然的事, 但是对于计算机来说可是向前迈出了一大步。

先来看一个例子。图 5.1 展示了一个汉语翻译到英语的例子。首先, 可以把源语言句子中的单词“我”、“对”、“你”、“感到”和“满意”分别翻译为“I”、“with”、“you”、“am”和“satisfied”, 然后调整单词的顺序, 比如, “am”放在译文的第 2 个位置, “you”应该放在最后的位置等等, 最后得到译文 “I am satisfied with you”。

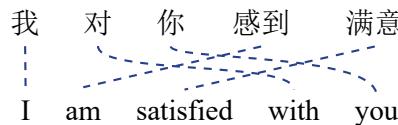


图 5.1 汉语到英语的翻译实例及两种语言单词之间的对应关系

上面的例子反映了人在做翻译时所使用的一些知识: 首先, 两种语言单词的顺序可能不一致, 而且译文需要符合目标语的习惯, 这也就是常说的翻译的流畅度; 其次, 源语言单词需要准确地被翻译出来, 也就是常说的翻译的准确性问题和充分性问题。为了达到以上目的, 传统观点认为翻译过程需要包含三个步骤<sup>[17]</sup>:

- **分析:** 将源语言句子表示为适合机器翻译的结构。在基于词的翻译模型中, 处理单元是单词, 因此在这里也可以简单地将分析理解为分词<sup>1</sup>。
- **转换:** 把源语言句子中的每个单词翻译成目标语言单词。
- **生成:** 基于转换的结果, 将目标语译文变成通顺且合乎语法的句子。

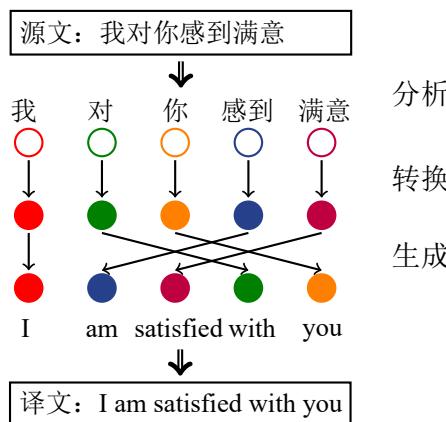


图 5.2 翻译过程中的分析、转换和生成

<sup>1</sup>在后续章节中会看到, 分析也包括对句子深层次结构的生成, 但是这里为了突出基于单词的概念, 因此把问题简化为最简单的情况。

图5.2给出了上述过程的一个示例。对于今天的自然语言处理研究，“分析、转换和生成”依然是一个非常深刻的观点。包括机器翻译在内的很多自然语言处理问题都可以用这个过程来解释。比如，对于现在比较前沿的神经机器翻译方法，从大的框架来说，依然在做分析（编码器）、转换（编码-解码注意力）和生成（解码器），只不过这些过程隐含在神经网络的设计中。当然，这里并不会对“分析、转换和生成”的架构展开过多的讨论，随着后面技术内容讨论的深入，这个观念会进一步体现。

## 5.2 一个简单实例

本节首先对比人工翻译和机器翻译过程的异同点，从中归纳出实现机器翻译过程的两个主要步骤：训练和解码。之后，会从学习翻译知识和运用翻译知识两个方面描述如何构建一个简单的机器翻译系统。

### 5.2.1 翻译的流程

#### 1. 人工翻译流程

当人翻译一个句子时，首先会快速地分析出句子的（单词）构成，然后根据以往的知识，得到每个词可能的翻译，最后利用对目标语的理解拼出来一个译文。尽管这个过程并不是严格来自心理学或者脑科学的相关结论，但至少可以帮助我们理解人在翻译时的思考方式。

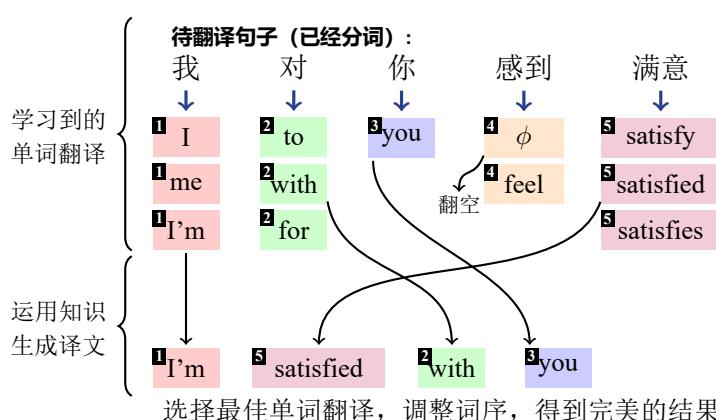


图 5.3 人工翻译的过程

图5.3展示了人在翻译“我/对/你/感到/满意”时可能会思考的内容<sup>2</sup>。具体来说，有如下两方面内容：

- **翻译知识的学习:** 对于输入的源语言句子，首先需要知道每个单词可能的翻译有什么，这些翻译被称为**翻译候选**（Translation Candidate）。比如，汉语单词“对”可能的译文有“to”、“with”和“for”等。对于人来说，可以通过阅读、背诵、

<sup>2</sup>这里用斜杠表示单词之间的分隔。

做题或者老师教等途径获得翻译知识，这些知识就包含了源语言与目标语言单词之间的对应关系。通常，也把这个过程称之为学习过程。

- **运用知识生成译文：**当翻译一个从未见过的句子时，可以运用学习到的翻译知识，得到新的句子中每个单词的译文，并处理常见的单词搭配、主谓一致等问题，比如，英语中“*satisfied*”后面常常使用介词“*with*”构成搭配。基于这些知识可以快速生成译文。

当然，每个人进行翻译时所使用的方法和技巧都不相同，所谓人工翻译也没有固定的流程。但是，可以确定的是，人在进行翻译时也需要“学习”和“运用”翻译知识。对翻译知识“学习”和“运用”的好与坏，直接决定了人工翻译结果的质量。

## 2. 机器翻译流程

人进行翻译的过程比较容易理解，那计算机是如何完成翻译的呢？虽然人工智能这个概念显得很神奇，但是计算机远没有人那么智能，有时甚至还很“笨”。一方面，它没有能力像人一样，在教室里和老师一起学习语言知识；另一方面，即使能列举出每个单词的候选译文，但是还是不知道这些译文是怎么拼装成句的，甚至不知道哪些译文是对的。为了更加直观地理解机器在翻译时要解决的挑战，可以将问题归纳如下：

- 如何让计算机获得每个单词的译文，然后将这些单词的译文拼装成句？
- 如果可以形成整句的译文，如何让计算机知道不同译文的好坏？

对于第一个问题，可以给计算机一个翻译词典，这样计算机可以发挥计算方面的优势，尽可能多地把翻译结果拼装出来。比如，可以把每个翻译结果看作是对单词翻译的拼装，这可以被形象地比作贯穿多个单词的一条路径，计算机所做的就是尽可能多地生成这样的路径。图5.4中蓝色和红色的折线就分别表示了两条不同的译文选择路径，区别在于“满意”和“对”的翻译候选是不一样的，蓝色折线选择的是“*satisfy*”和“*to*”，而红色折线是“*satisfied*”和“*with*”。换句话说，不同的译文对应不同的路径（即使词序不同也会对应不同的路径）。

对于第二个问题，尽管机器能够找到很多译文选择路径，但它并不知道哪些路径是好的。说得再直白一些，简单地枚举路径实际上就是一个体力活，没有太多的智能。因此计算机还需要再聪明一些，运用它的能够“掌握”的知识判断翻译结果的好与坏。这一步是最具挑战的，当然也有很多思路来解决这个问题。在统计机器翻译中，这个问题被定义为：设计一种统计模型，它可以给每个译文一个可能性，而这个可能性越高表明译文越接近人工翻译。

如图5.4所示，每个单词翻译候选的右侧黑色框里的数字就是单词的翻译概率，使用这些单词的翻译概率，可以得到整句译文的概率（用符号  $P$  表示）。这样，就用概率化的模型描述了每个翻译候选的可能性。基于这些翻译候选的可能性，机器翻译系统可以对所有的翻译路径进行打分，比如，图5.4中第一条路径的分为 0.042，

第二条是 0.006，以此类推。最后，系统可以选择分数最高的路径作为源语言句子的最终译文。

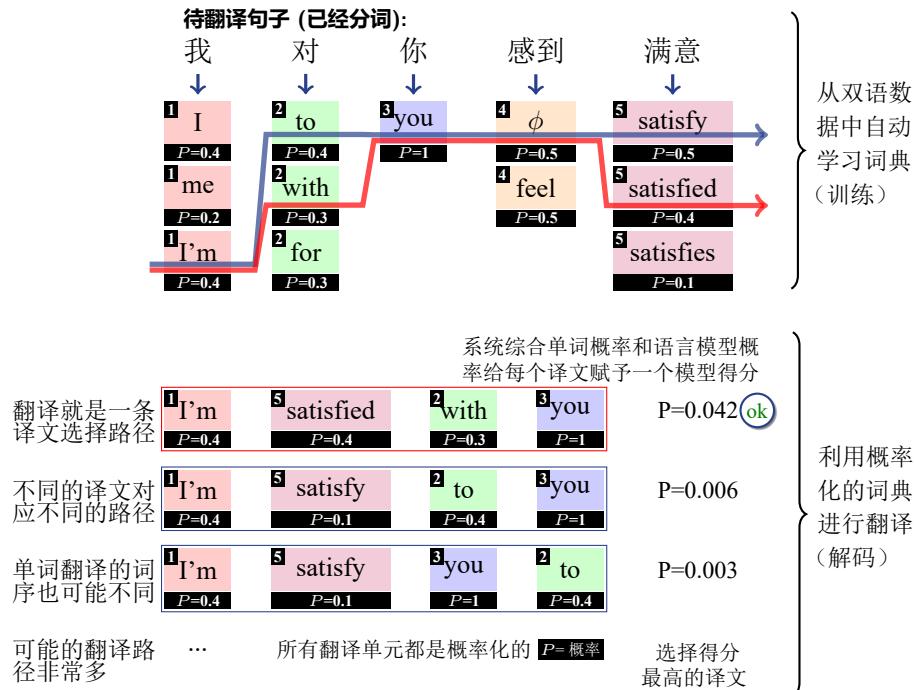


图 5.4 机器翻译的过程——把单词的译文进行拼装，并找到最优的拼装路径

### 3. 人工翻译 vs 机器翻译

人在翻译时的决策是非常确定并且快速的，但计算机处理这个问题时却充满了概率化的思想。当然它们也有类似的地方。首先，计算机使用统计模型的目的是把翻译知识变得可计算，并把这些“知识”储存在模型参数中，这个模型和人类大脑的作用是类似的<sup>3</sup>；其次，计算机对统计模型进行训练相当于人类对知识的学习，二者都可以被看作是理解、加工知识的过程；再有，计算机使用学习到的模型对新句子进行翻译的过程相当于人运用知识的过程。在统计机器翻译中，模型学习的过程被称为训练，目的是从双语平行数据中自动学习翻译“知识”；而使用模型处理新句子的过程是一个典型的预测过程，也被称为解码或推断。图5.4的右侧标注在翻译过程中训练和解码的作用。最终，统计机器翻译的核心由三部分构成——建模、训练和解码。本章后续内容会围绕这三个问题展开讨论。

<sup>3</sup> 这里并不是要把统计模型等同于生物学或者认知科学上的人脑，这里是指它们处理翻译问题时发挥的作用类似。

## 5.2.2 统计机器翻译的基本框架

为了对统计机器翻译有一个直观的认识，下面将介绍如何构建一个非常简单的统计机器翻译系统，其中涉及到的很多思想来自 IBM 模型。这里，仍然使用数据驱动的统计建模方法。图5.5展示了系统的主要流程，包括两个步骤：

- 训练：**从双语平行数据中学习翻译模型，记为  $P(t|s)$ ，其中  $s$  表示源语言句子， $t$  表示目标语句子。 $P(t|s)$  表示把  $s$  翻译为  $t$  的概率。简言之，这一步需要从大量的双语平行数据中学习到  $P(t|s)$  的准确表达。
- 解码：**当面对一个新的句子时，需要使用学习到的模型进行预测。预测可以被视为一个搜索和计算的过程，也就是，尽可能搜索更多的翻译结果，然后用训练好的模型对每个翻译结果进行打分，最后选择得分最高的翻译结果作为输出。



图 5.5 简单的统计机器翻译流程

接下来，本节将介绍统计机器翻译模型训练和解码的方法。在模型学习中，会分两小节进行描述——单词级翻译和句子级翻译。实现单词级翻译是实现句子级翻译的基础。换言之，句子级翻译的统计模型是建立在单词翻译之上的。在解码中，本节将介绍一个高效的搜索算法，其中也使用到了剪枝和启发式搜索的思想。

## 5.2.3 单词翻译概率

### 1. 什么是单词翻译概率？

单词翻译概率描述的是一个源语言单词与目标语言译文构成正确翻译的可能性，这个概率越高表明单词翻译越可靠。使用单词翻译概率，可以帮助机器翻译系统解决翻译时的“择词”问题，即选择什么样的目标语译文是合适的。当人在翻译某个单词时，可以利用积累的知识，快速得到它的高质量候选译文。

以汉译英为例，当翻译“我”这个单词时，可能直接会想到用“I”、“me”或“I'm”作为它的译文，而几乎不会选择“you”、“satisfied”等含义相差太远的译文。这是为什么呢？如果从统计学的角度来看，无论是何种语料，包括教材、新闻、小说等，绝大部分情况下“我”都翻译成了“I”、“me”等，几乎不会看到我被翻译成“you”或“satisfied”的情况。可以说“我”翻译成“I”、“me”等属于高频事件，而翻译成“you”、“satisfied”等属于低频或小概率事件。因此人在翻译时也是选择在统计意义上概率更大的译文，这也间接反映出统计模型可以在一定程度上描述人的翻译习惯和模式。

表5.1展示了汉语到英语的单词翻译实例及相应的翻译概率。可以看到，“我”翻译成“I”的概率最高，为 0.5。这是符合人类对翻译的认知的。此外，这种概率化的

模型避免了非 0 即 1 的判断，所有的译文都是可能的，只是概率不同。这也使得统计模型可以覆盖更多的翻译现象，甚至捕捉到一些人所忽略的情况。

表 5.1 汉译英单词翻译概率

源语言	目标语言	翻译概率
我	I	0.50
	me	0.20
	I'm	0.10
	we	0.05
	am	0.10
	...	...

## 2. 如何从双语平行数据中进行学习？

假设有一定数量的双语对照的平行数据，是否可以从中自动获得两种语言单词之间的翻译概率呢？回忆一下第二章中的掷骰子游戏，其中使用了相对频次估计方法来自动获得骰子不同面出现概率的估计值。其中，重复投掷骰子很多次，然后统计“1”到“6”各面出现的次数，再除以投掷的总次数，最后得到它们出现的概率的极大似然估计。这里，可以使用类似的方式计算单词翻译概率。但是，现在有的是句子一级对齐的数据，并不知道两种语言之间单词的对应关系。也就是，要从句子级对齐的平行数据中学习单词之间对齐的概率。这里，需要使用稍微“复杂”一些的模型来描述这个问题。

令  $X$  和  $Y$  分别表示源语言和目标语言的词汇表。对于任意源语言单词  $x \in X$ ，所有的目标语单词  $y \in Y$  都可能是它的译文。给定一个互译的句对  $(s, t)$ ，可以把  $P(x \leftrightarrow y; s, t)$  定义为：在观测到  $(s, t)$  的前提下  $x$  和  $y$  互译的概率。其中  $x$  是属于句子  $s$  中的词，而  $y$  是属于句子  $t$  中的词。 $P(x \leftrightarrow y; s, t)$  的计算公式描述如下：

$$\begin{aligned} P(x \leftrightarrow y; s, t) &\equiv P(x, y; s, t) \\ &= \frac{c(x, y; s, t)}{\sum_{x', y'} c(x', y'; s, t)} \end{aligned} \quad (5.1)$$

其中， $\equiv$  表示定义式。分子  $c(x, y; s, t)$  表示  $x$  和  $y$  在句对  $(s, t)$  中共现的总次数，分母  $\sum_{x', y'} c(x', y'; s, t)$  表示任意的源语言单词  $x'$  和任意的目标语言单词  $y'$  在  $(s, t)$  共同出现的总次数。

看一个具体的例子，如例 5.1 所示，有一个汉英互译的句对  $(s, t)$ 。

### 实例 5.1 一个汉英互译的句对

$s = \text{机器 翻译 就是用计算机来生成翻译的过程}$

$t = \text{machine translation is a process of generating a translation by computer}$

假设， $x = \text{“翻译”}$ ， $y = \text{“translation”}$ ，现在要计算  $x$  和  $y$  共现的总次数。“翻译”

和“translation”分别在  $s$  和  $t$  中出现了 2 次，因此  $c(\text{“翻译”}, \text{“translation”}; s, t)$  等于 4。而对于  $\sum_{x',y'} c(x', y'; s, t)$ ，因为  $x'$  和  $y'$  分别表示的是  $s$  和  $t$  中的任意词，所以  $\sum_{x',y'} c(x', y'; s, t)$  表示所有单词对的数量——即  $s$  的词数乘以  $t$  的词数。最后，“翻译”和“translation”的单词翻译概率为：

$$\begin{aligned} P(\text{翻译, translation}; s, t) &= \frac{c(\text{翻译, translation}; s, t)}{\sum_{x',y'} c(x', y'; s, t)} \\ &= \frac{4}{|s| \times |t|} \\ &= \frac{4}{121} \end{aligned} \quad (5.2)$$

这里运算  $|\cdot|$  表示句子长度。类似的，可以得到“机器”和“translation”、“机器”和“look”的单词翻译概率：

$$P(\text{机器, translation}; s, t) = \frac{2}{121} \quad (5.3)$$

$$P(\text{机器, look}; s, t) = \frac{0}{121} \quad (5.4)$$

注意，由于“look”没有出现在数据中，因此  $P(\text{机器, look}; s, t) = 0$ 。这时，可以使用第二章介绍的平滑算法赋予它一个非零的值，以保证在后续的步骤中整个翻译模型不会出现零概率的情况。

### 3. 如何从大量的双语平行数据中进行学习？

如果有更多的句子，上面的方法同样适用。假设，有  $K$  个互译句对  $\{(s^{[1]}, t^{[1]}), \dots, (s^{[K]}, t^{[K]})\}$ 。仍然可以使用基于相对频次的方法估计翻译概率  $P(x, y)$ ，具体方法如下：

$$P(x, y) = \frac{\sum_{k=1}^K c(x, y; s^{[k]}, t^{[k]})}{\sum_{k=1}^K \sum_{x',y'} c(x', y'; s^{[k]}, t^{[k]})} \quad (5.5)$$

与公式(5.1)相比，公式(5.5)的分子、分母都多了一项累加符号  $\sum_{k=1}^K \cdot$ ，它表示遍历语料库中所有的句对。换句话说，当计算词的共现次数时，需要对每个句对上的计数结果进行累加。从统计学习的角度，使用更大规模的数据进行参数估计可以提高结果的可靠性。计算单词的翻译概率也是一样，在小规模的数据上看，很多翻译现象的特征并不突出，但是当使用的数据量增加到一定程度，翻译的规律会很明显的体现出来。

举个例子，实例5.2展示了一个由两个句对构成的平行语料库。

#### 实例 5.2 两个汉英互译的句对

$s^{[1]}$  = 机器 翻译 就是用计算机来生成翻译的过程

$t^{[1]} = \text{machine translation}$  is a process of generating a translation by computer

$s^{[2]} = \text{那 人工 翻译 呢 ?}$

$t^{[2]} = \text{So , what is human } \underline{\text{translation}} \text{ ?}$

其中,  $s^{[1]}$  和  $s^{[2]}$  分别表示第一个句对和第二个句对的源语言句子,  $t^{[1]}$  和  $t^{[2]}$  表示对应的目标语言句子。于是, “翻译” 和 “translation” 的翻译概率为

$$\begin{aligned} P(\text{翻译}, \text{translation}) &= \frac{c(\text{翻译}, \text{translation}; s^{[1]}, t^{[1]}) + c(\text{翻译}, \text{translation}; s^{[2]}, t^{[2]})}{\sum_{x', y'} c(x', y'; s^{[1]}, t^{[1]}) + \sum_{x', y'} c(x', y'; s^{[2]}, t^{[2]})} \\ &= \frac{4+1}{|s^{[1]}| \times |t^{[1]}| + |s^{[2]}| \times |t^{[2]}|} \\ &= \frac{4+1}{11 \times 11 + 5 \times 7} \\ &= \frac{5}{156} \end{aligned} \quad (5.6)$$

公式(5.6)所展示的计算过程很简单, 分子是两个句对中“翻译”和“translation”共现次数的累计, 分母是两个句对的源语言单词和目标语言单词的组合数的累加。显然, 这个方法也很容易推广到处理更多句子的情况。

## 5.2.4 句子级翻译模型

下面继续回答如何获取句子级翻译概率的问题, 即: 对于源语言句子  $s$  和目标语言句子  $t$ , 计算  $P(t|s)$ 。这也是整个句子级翻译模型的核心, 一方面需要从数据中学习这个模型的参数, 另一方面, 对于新输入的句子, 需要使用这个模型得到最佳的译文。下面介绍句子级翻译的建模方法。

### 1. 基础模型

计算句子级翻译概率并不简单。因为自然语言非常灵活, 任何数据无法覆盖足够多的句子, 因此, 无法像公式(5.5)一样直接用简单计数的方式对句子的翻译概率进行估计。这里, 采用一个退而求其次的方法: 找到一个函数  $g(s, t) \geq 0$  来模拟翻译概率对译文可能性进行估计。可以定义一个新的函数  $g(s, t)$ , 令其满足: 给定  $s$ , 翻译结果  $t$  出现的可能性越大,  $g(s, t)$  的值越大;  $t$  出现的可能性越小,  $g(s, t)$  的值越小。换句话说,  $g(s, t)$  和翻译概率  $P(t|s)$  呈正相关。如果存在这样的函数  $g(s, t)$ , 可以利用  $g(s, t)$  近似表示  $P(t|s)$ , 如下:

$$P(t|s) \equiv \frac{g(s, t)}{\sum_{t'} g(s, t')} \quad (5.7)$$

公式(5.7)相当于在函数  $g(\cdot)$  上做了归一化, 这样等式右端的结果具有一些概率的属性, 比如,  $0 \leq \frac{g(s, t)}{\sum_{t'} g(s, t')} \leq 1$ 。具体来说, 对于源语言句子  $s$ , 枚举其所有的翻译结果, 并把所对应的函数  $g(\cdot)$  相加作为分母, 而分子是某个翻译结果  $t$  所对应的  $g(\cdot)$  的值。

上述过程初步建立了句子级翻译模型，并没有直接求  $P(t|s)$ ，而是把问题转化为对  $g(\cdot)$  的设计和计算上。但是，面临着两个新的问题：

- 如何定义函数  $g(s, t)$ ？即，在知道单词翻译概率的前提下，如何计算  $g(s, t)$ ；
- 公式(5.7)中分母  $\sum_{seqt'} g(s, t')$  需要累加所有翻译结果的  $g(s, t')$ ，但枚举所有  $t'$  是不现实的。

当然，这里最核心的问题还是函数  $g(s, t)$  的定义。而第二个问题其实不需要解决，因为机器翻译只关注于可能性最大的翻译结果，即  $g(s, t)$  的计算结果最大时对应的译文。这个问题会在后面进行讨论。

回到设计  $g(s, t)$  的问题上。这里，采用“大题小作”的方法，这个技巧在第二章已经进行了充分的介绍。具体来说，直接建模句子之间的对应比较困难，但可以利用单词之间的对应来描述句子之间的对应关系。这就用到了5.2.3小节所介绍的单词翻译概率。

首先引入一个非常重要的概念——**词对齐** (Word Alignment)，它是统计机器翻译中最核心的概念之一。词对齐描述了平行句对中单词之间的对应关系，它体现了一种观点：本质上句子之间的对应是由单词之间的对应表示的。当然，这个观点在神经机器翻译或者其他模型中可能会有不同的理解，但是翻译句子的过程中考虑词级的对应关系是符合人类对语言的认知的。

图5.6展示了一个句对  $s$  和  $t$ ，单词的右下标数字表示了该词在句中的位置，而虚线表示的是句子  $s$  和  $t$  中的词对齐关系。比如，“满意”的右下标数字 5 表示在句子  $s$  中处于第 5 个位置，“satisfied” 的右下标数字 3 表示在句子  $t$  中处于第 3 个位置，“满意”和“satisfied”之间的虚线表示两个单词之间是对齐的。为方便描述，用二元组  $(j, i)$  来描述词对齐，它表示源语言句子的第  $j$  个单词对应目标语言句子的第  $i$  个单词，即单词  $s_j$  和  $t_i$  对应。通常，也会把  $(j, i)$  称作一条**词对齐连接** (Word Alignment Link)。图5.6中共有 5 条虚线，表示有 5 组单词之间的词对齐连接。可以把这些词对齐连接构成的集合作为词对齐的一种表示，记为  $A$ ，即  $A = \{(1,1), (2,4), (3,5), (4,2), (5,3)\}$ 。

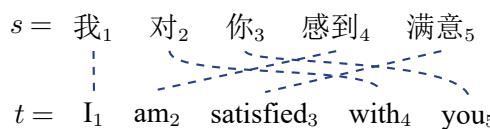


图 5.6 汉英互译句对及词对齐连接（蓝色虚线）

对于句对  $(s, t)$ ，假设可以得到最优词对齐  $\hat{A}$ ，于是可以使用单词翻译概率计算  $g(s, t)$ ，如下

$$g(s, t) = \prod_{(j, i) \in \hat{A}} P(s_j, t_i) \quad (5.8)$$

其中  $g(s, t)$  被定义为句子  $s$  中的单词和句子  $t$  中的单词的翻译概率的乘积，并且这

两个单词之间必须有词对齐连接。 $P(s_j, t_i)$  表示具有词对齐连接的源语言单词  $s_j$  和目标语言单词  $t_i$  的单词翻译概率。以图5.6中的句对为例，其中“我”与“I”、“对”与“with”、“你”与“you”等相互对应，可以把它们的翻译概率相乘得到  $g(s, t)$  的计算结果，如下：

$$\begin{aligned} g(s, t) = & P(\text{我}, \text{I}) \times P(\text{对}, \text{with}) \times P(\text{你}, \text{you}) \times \\ & P(\text{感到}, \text{am}) \times P(\text{满意}, \text{satisfied}) \end{aligned} \quad (5.9)$$

显然，如果每个词对齐连接所对应的翻译概率变大，那么整个句子翻译的得分也会提高。也就是说，词对齐越准确，翻译模型的打分越高， $s$  和  $t$  之间存在翻译关系的可能性越大。

## 2. 生成流畅的译文

公式(5.8)定义的  $g(s, t)$  存在的问题是没有考虑词序信息。这里用一个简单的例子说明这个问题。如图5.7所示，源语言句子“我对你感到满意”有两个翻译结果，第一个翻译结果是“*I am satisfied with you*”，第二个是“*I with you am satisfied*”。虽然这两个译文包含的目标语单词是一样的，但词序存在很大差异。比如，它们都选择了“satisfied”作为源语单词“满意”的译文，但是在第一个翻译结果中“satisfied”处于第3个位置，而第二个结果中处于最后的位置。显然第一个翻译结果更符合英语的表达习惯，翻译的质量更高。遗憾的是，对于有明显差异的两个译文，公式(5.8)计算得到的函数  $g(\cdot)$  的值却是一样的。

源语言句子“我对你感到满意”的不同翻译结果					$\prod_{(j,i) \in \hat{A}} P(s_j, t_i)$
$s =$ 我 <sub>1</sub>	对 <sub>2</sub>	你 <sub>3</sub>	感到 <sub>4</sub>	满意 <sub>5</sub>	
$t' =$ I <sub>1</sub>	am <sub>2</sub>	satisfied <sub>3</sub>	with <sub>4</sub>	you <sub>5</sub>	0.0023
$s =$ 我 <sub>1</sub>	对 <sub>2</sub>	你 <sub>3</sub>	感到 <sub>4</sub>	满意 <sub>5</sub>	
$t'' =$ I <sub>1</sub>	with <sub>2</sub>	you <sub>3</sub>	am <sub>4</sub>	satisfied <sub>5</sub>	0.0023

图 5.7 同一个源语言句子的不同译文所对应的  $g(\cdot)$  得分

如何在  $g(s, t)$  引入词序信息呢？理想情况下，函数  $g(s, t)$  对符合自然语言表达习惯的翻译结果给出更高的分数，对于不符合的或不通顺的句子给出更低的分数。这里很自然想到使用语言模型，因为语言模型可以度量一个句子出现的可能性。流畅的句子语言模型得分越高，反之越低。

这里可以使用第二章介绍的  $n$ -gram 语言模型，它也是统计机器翻译中确保流畅翻译结果的重要手段之一。 $n$ -gram 语言模型用概率化方法描述了句子的生成过程。

以 2-gram 语言模型为例，可以使用如下公式计算一个词串的概率：

$$\begin{aligned} P_{\text{lm}}(t) &= P_{\text{lm}}(t_1 \dots t_l) \\ &= P(t_1) \times P(t_2|t_1) \times P(t_3|t_2) \times \dots \times P(t_l|t_{l-1}) \end{aligned} \quad (5.10)$$

其中， $t = \{t_1 \dots t_l\}$  表示由  $l$  个单词组成的句子， $P_{\text{lm}}(t)$  表示语言模型给句子  $t$  的打分。具体而言， $P_{\text{lm}}(t)$  被定义为  $P(t_i|t_{i-1})(i=1, 2, \dots, l)$  的连乘<sup>4</sup>，其中  $P(t_i|t_{i-1})(i=1, 2, \dots, l)$  表示前面一个单词为  $t_{i-1}$  时，当前单词为  $t_i$  的概率。语言模型的训练方法可以参看第二章相关内容。

回到建模问题上来。既然语言模型可以帮助系统度量每个译文的流畅度，那么可以使用它对翻译进行打分。一种简单的方法是把语言模型  $P_{\text{lm}}(t)$  和公式(5.8)中的  $g(s, t)$  相乘，这样就得到了一个新的  $g(s, t)$ ，它同时考虑了翻译准确性 ( $\prod_{j, i \in \hat{A}} P(s_j, t_i)$ ) 和流畅度 ( $P_{\text{lm}}(t)$ )：

$$g(s, t) \equiv \prod_{j, i \in \hat{A}} P(s_j, t_i) \times P_{\text{lm}}(t) \quad (5.11)$$

如图5.8所示，语言模型  $P_{\text{lm}}(t)$  分别给  $t'$  和  $t$  赋予 0.0107 和 0.0009 的概率，这表明句子  $t'$  更符合英文的表达，这与期望是相吻合的。它们再分别乘以  $\prod_{j, i \in \hat{A}} P(s_j, t_i)$  的值，就得到公式(5.11)定义的函数  $g(\cdot)$  的值。显然句子  $t'$  的分数更高。至此，完成了对函数  $g(s, t)$  的一个简单定义，把它带入公式(5.7)就得到了同时考虑准确性和流畅性的句子级统计翻译模型。

源语言句子“我对你感到满意”的不同翻译结果					$\prod_{(j, i) \in \hat{A}} P(s_j, t_i) \times P_{\text{lm}}(t)$
$s =$	我 <sub>1</sub>	对 <sub>2</sub>	你 <sub>3</sub>	感到 <sub>4</sub>	满意 <sub>5</sub>
$t' =$	I <sub>1</sub>	am <sub>2</sub>	satisfied <sub>3</sub>	with <sub>4</sub>	you <sub>5</sub>
$s =$	我 <sub>1</sub>	对 <sub>2</sub>	你 <sub>3</sub>	感到 <sub>4</sub>	满意 <sub>5</sub>
$t'' =$	I <sub>1</sub>	with <sub>2</sub>	you <sub>3</sub>	am <sub>4</sub>	satisfied <sub>5</sub>

图 5.8 同一个源语言句子的不同译文所对应的语言模型得分和翻译模型得分

## 5.2.5 解码

解码是指在得到翻译模型后，对于新输入的句子生成最佳译文的过程。具体来说，当给定任意的源语言句子  $s$ ，解码系统要找到翻译概率最大的目标语译文  $\hat{t}$ 。这

<sup>4</sup>为了确保数学表达的准确性，本书中定义  $P(t_1|t_0) \equiv P(t_1)$

个过程可以被形式化描述为：

$$\hat{t} = \arg \max_t P(t|s) \quad (5.12)$$

其中  $\arg \max_t P(t|s)$  表示找到使  $P(t|s)$  达到最大时的译文  $t$ 。结合5.2.4小节中关于  $P(t|s)$  的定义，把公式(5.7)带入公式(5.12)得到：

$$\hat{t} = \arg \max_t \frac{g(s,t)}{\sum_{t'} g(s,t')} \quad (5.13)$$

在公式(5.13)中，可以发现  $\sum_{t'} g(s,t')$  是一个关于  $s$  的函数，当给定源语句  $s$  时，它是一个常数，而且  $g(\cdot) \geq 0$ ，因此  $\sum_{t'} g(s,t')$  不影响对  $\hat{t}$  的求解，也不需要计算。基于此，公式(5.13)可以被化简为：

$$\hat{t} = \arg \max_t g(s,t) \quad (5.14)$$

公式(5.14)定义了解码的目标，剩下的问题是实现  $\arg \max$ ，以快速准确地找到最佳译文  $\hat{t}$ 。但是，简单遍历所有可能的译文并计算  $g(s,t)$  的值是不可行的，因为所有潜在译文构成的搜索空间是十分巨大的。为了理解机器翻译的搜索空间的规模，假设源语言句子  $s$  有  $m$  个词，每个词有  $n$  个可能的翻译候选。如果从左到右一步步翻译每个源语言单词，那么简单的顺序翻译会有  $n^m$  种组合。如果进一步考虑目标语单词的任意调序，每一种对翻译候选进行选择的结果又会对应  $m!$  种不同的排序。因此，源语句子  $s$  至少有  $n^m \cdot m!$  个不同的译文。

$n^m \cdot m!$  是什么样的概念呢？如表5.2所示，当  $m$  和  $n$  分别为 2 和 10 时，译文只有 200 个，不算多。但是当  $m$  和  $n$  分别为 20 和 10 时，即源语言句子的长度 20，每个词有 10 个候选译文，系统会面对  $2.4329 \times 10^{38}$  个不同的译文，这几乎是不可计算的。

表 5.2 机器翻译搜索空间大小的示例

句子长度 $m$	单词翻译候选数量 $n$	译文数量 $n^m \cdot m!$
1	1	1
1	10	10
2	10	200
10	10	3628800000000000000
20	10	$2.43290200817664 \times 10^{38}$
20	30	$8.48300477127188 \times 10^{47}$

已经有工作证明机器翻译问题是 NP 难的<sup>[238]</sup>。对于如此巨大的搜索空间，需要一种十分高效的搜索算法才能实现机器翻译的解码。在第二章已经介绍一些常用的

搜索方法。这里使用一种贪婪的搜索方法实现机器翻译的解码。它把解码分成若干步骤，每步只翻译一个单词，并保留当前“最好”的结果，直至所有源语言单词都被翻译完毕。

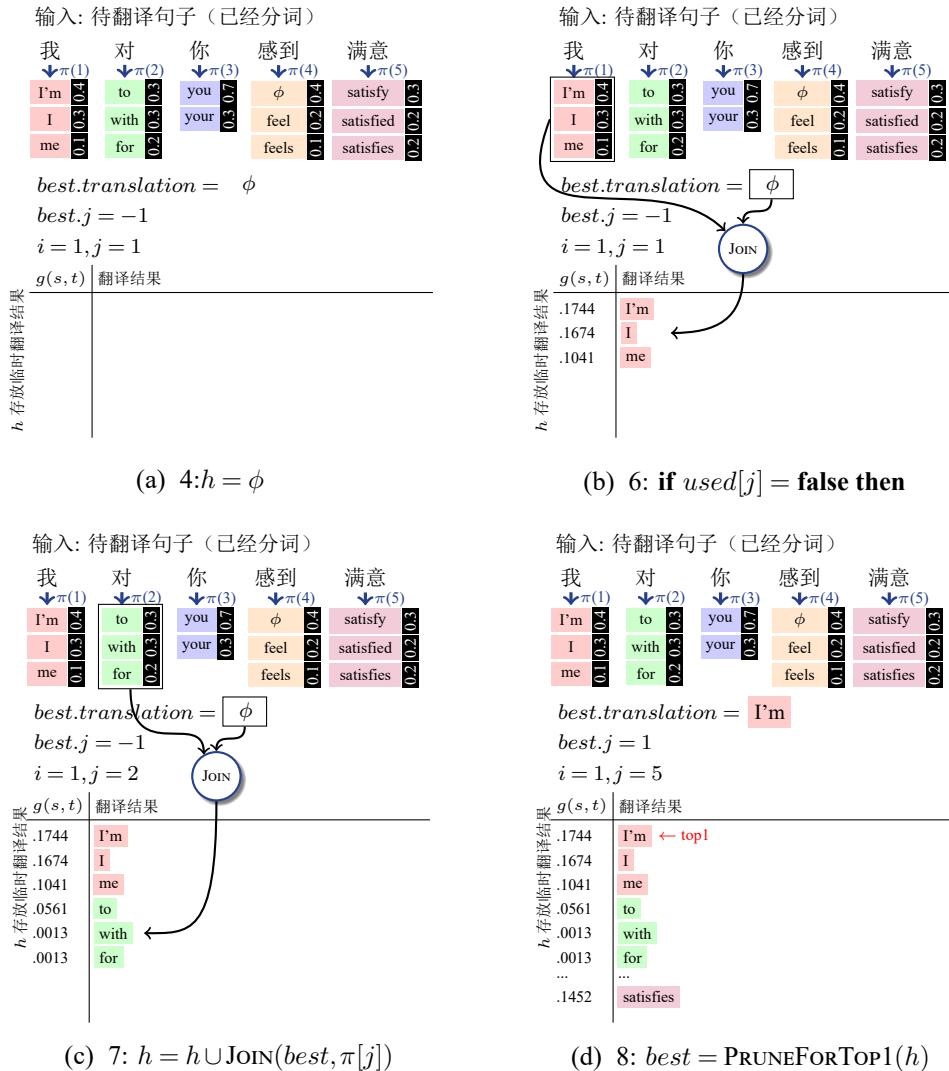


图 5.9 贪婪的机器翻译解码过程实例

图5.10给出了贪婪解码算法的伪代码。其中 $\pi$ 保存所有源语单词的候选译文， $\pi[j]$ 表示第 $j$ 个源语单词的翻译候选的集合， $best$ 保存当前最好的翻译结果， $h$ 保存当前步生成的所有译文候选。算法的主体有两层循环，在内层循环中如果第 $j$ 个源语单词没有被翻译过，则用 $best$ 和它的候选译文 $\pi[j]$ 生成新的翻译，再存于 $h$ 中，即操作 $h = h \cup \text{Join}(best, \pi[j])$ 。外层循环再从 $h$ 中选择得分最高的结果存于 $best$ 中，即操作 $best = \text{PruneForTop1}(h)$ ；同时标记相应的源语言单词状态为已翻译，即 $used[best.j] = true$ 。

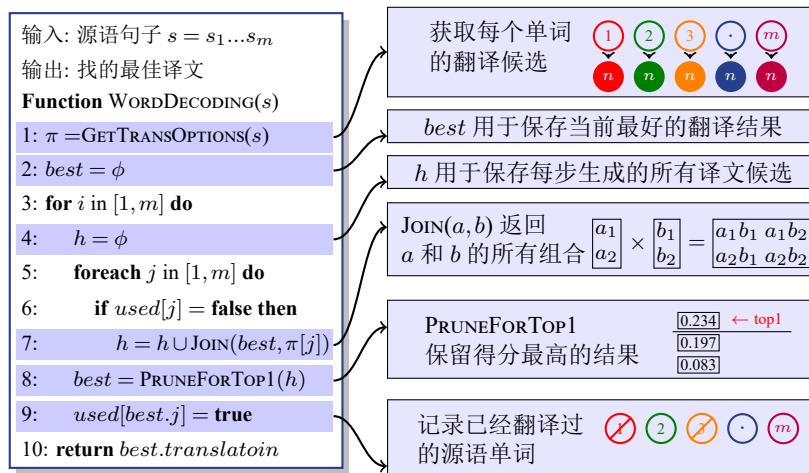


图 5.10 贪婪的机器翻译解码算法的伪代码

该算法的核心在于，系统一直维护一个当前最好的结果，之后每一步考虑扩展这个结果的所有可能，并计算模型得分，然后再保留扩展后的最好结果。注意，在每一步中，只有排名第一的结果才会被保留，其他结果都会被丢弃。这也体现了贪婪的思想。显然这个方法不能保证搜索到全局最优的结果，但是由于每次扩展只考虑一个最好的结果，因此该方法速度很快。图5.9给出了算法执行过程的简单示例。当然，机器翻译的解码方法有很多，这里仅仅使用简单的贪婪搜索方法来解决机器翻译的解码问题，在后续章节会对更加优秀的解码方法进行介绍。

## 5.3 噪声信道模型

在5.2节中，我们实现了一个简单的基于词的统计机器翻译模型，内容涉及建模、训练和解码。但是，还有很多问题还没有进行深入讨论，比如，如何处理空翻译？如何对调序问题进行建模？如何用更严密的数学模型描述翻译过程？如何对更加复杂的统计模型进行训练？等等。针对以上问题，本节将系统地介绍 IBM 统计机器翻译模型。作为经典的机器翻译模型，对 IBM 模型的学习将有助于对自然语言处理问题建立系统化建模思想，特别是对问题的数学描述方法将会成为理解本书后续内容的基础工具。

首先，重新思考一下人类进行翻译的过程。对于给定的源语句  $s$ ，人不会像计算机一样尝试很多的可能，而是快速准确地翻译出一个或者少数几个正确的译文。在人看来，除了正确的译文外，其他的翻译都是不正确的，或者说除了少数的译文人甚至都不会考虑太多其他的可能性。但是，在统计机器翻译的世界里，没有译文是不可能的。换句话说，对于源语言句子  $s$ ，所有目标语词串  $t$  都是可能的译文，只是可能性大小不同。这个思想可以通过统计模型实现：每对  $(s, t)$  都有一个概率值  $P(t|s)$  来描述  $s$  翻译为  $t$  的好与坏（图5.11）。

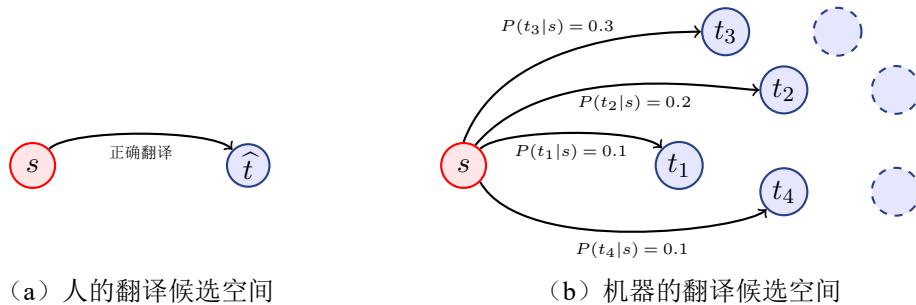


图 5.11 不同翻译候选空间的对比：人（左）vs 机器翻译（右）

IBM 模型也是建立在如上统计模型之上。具体来说，IBM 模型的基础是**噪声信道模型**（Noise Channel Model），它是由 Shannon 在上世纪 40 年代末提出来的<sup>[239]</sup>，并于上世纪 80 年代应用在语言识别领域，后来又被 Brown 等人用于统计机器翻译中<sup>[9, 10]</sup>。

在噪声信道模型中，源语言句子  $s$ （信宿）被看作是由目标语言句子  $t$ （信源）经过一个有噪声的信道得到的。如果知道了  $s$  和信道的性质，可以通过  $P(t|s)$  得到信源的信息，这个过程如图 5.12 所示。

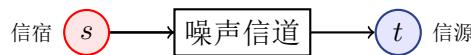


图 5.12 噪声信道模型

举个例子，对于汉译英的翻译任务，英语句子  $t$  可以被看作是汉语句子  $s$  加入噪声通过信道后得到的结果。换句话说，汉语句子经过噪声-信道传输时发生了变化，在信道的输出端呈现为英语句子。于是需要根据观察到的汉语特征，通过概率  $P(t|s)$  猜测最为可能的英语句子。这个找到最可能的目标语句（信源）的过程也被称为**解码**（Decoding）。直到今天，解码这个概念也被广泛地使用在机器翻译及相关任务中。这个过程也可以表述为：给定输入  $s$ ，找到最可能的输出  $t$ ，使得  $P(t|s)$  达到最大：

$$\hat{t} = \arg \max_t P(t|s) \quad (5.15)$$

公式(5.15)的核心内容之一是定义  $P(t|s)$ 。在 IBM 模型中，可以使用贝叶斯准则对  $P(t|s)$  进行如下变换：

$$\begin{aligned} P(t|s) &= \frac{P(s,t)}{P(s)} \\ &= \frac{P(s|t)P(t)}{P(s)} \end{aligned} \quad (5.16)$$

公式(5.16)把  $s$  到  $t$  的翻译概率转化为  $\frac{P(s|t)P(t)}{P(s)}$ ，它包括三个部分：

- 第一部分是由译文  $t$  到源语言句子  $s$  的翻译概率  $P(s|t)$ ，也被称为**翻译模型**。

它表示给定目标语句  $t$  生成源语句  $s$  的概率。需要注意是翻译的方向已经从  $P(t|s)$  转向了  $P(s|t)$ ，但无须刻意地区分，可以简单地理解为翻译模型描述了  $s$  和  $t$  的翻译对应程度；

- 第二部分是  $P(t)$ ，也被称为语言模型。它表示的是目标语言句子  $t$  出现的可能性；
- 第三部分是  $P(s)$ ，表示源语言句子  $s$  出现的可能性。因为  $s$  是输入的不变量，而且  $P(s) > 0$ ，所以省略分母部分  $P(s)$  不会影响  $\frac{P(s|t)P(t)}{P(s)}$  最大值的求解。

于是，机器翻译的目标可以被重新定义为：给定源语言句子  $s$ ，寻找这样的目标语言译文  $t$ ，它使得翻译模型  $P(s|t)$  和语言模型  $P(t)$  乘积最大：

$$\begin{aligned}\hat{t} &= \arg \max_t P(t|s) \\ &= \arg \max_t \frac{P(s|t)P(t)}{P(s)} \\ &= \arg \max_t P(s|t)P(t)\end{aligned}\quad (5.17)$$

公式(5.17)展示了 IBM 模型最基础的建模方式，它把模型分解为两项：(反向)翻译模型  $P(s|t)$  和语言模型  $P(t)$ 。仔细观察公式(5.17)的推导过程，我们很容易发现一个问题：直接用  $P(t|s)$  定义翻译问题不就可以了吗，为什么要用  $P(s|t)$  和  $P(t)$  的联合模型？从理论上来说，正向翻译模型  $P(t|s)$  和反向翻译模型  $P(s|t)$  的数学建模可以是一样的，因为我们只需要在建模的过程中把两个语言调换即可。使用  $P(s|t)$  和  $P(t)$  的联合模型的意义在于引入了语言模型，它可以很好地对译文的流畅度进行评价，确保结果是通顺的目标语言句子。

可以回忆一下5.2.4节中讨论的问题，如果只使用翻译模型可能会造成一个局面：译文的单词都和源语言单词对应的很好，但是由于语序的问题，读起来却不像人说的话。从这个角度说，引入语言模型是十分必要的。这个问题在 Brown 等人的论文中也有讨论<sup>[10]</sup>，他们提到单纯使用  $P(s|t)$  会把概率分配给一些翻译对应比较好但是不通顺甚至不合逻辑的目标语言句子，而且这部分概率可能会很大，影响模型的决策。这也正体现了 IBM 模型的创新之处，作者用数学技巧把  $P(t)$  引入进来，保证了系统的输出是通顺的译文。语言模型也被广泛使用在语音识别等领域以保证结果的流畅性，甚至应用的历史比机器翻译要长得多，这里的方法也有借鉴相关工作的味道。

实际上，在机器翻译中引入语言模型这个概念十分重要。在 IBM 模型之后相当长的时间里，语言模型一直是机器翻译各个部件中最重要的部分。对译文连贯性的建模也是所有系统中需要包含的内容（即使隐形体现）。

## 5.4 统计机器翻译的三个基本问题

公式(5.17)给出了统计机器翻译的数学描述。为了实现这个过程，面临着三个基本问题：

- **建模** (Modeling): 如何建立  $P(s|t)$  和  $P(t)$  的数学模型。换句话说，需要用可计算的方式对翻译问题和语言建模问题进行描述，这也是最核心的问题。
- **训练** (Training): 如何获得  $P(s|t)$  和  $P(t)$  所需的参数。即从数据中得到模型的最优参数。
- **解码** (Decoding): 如何完成搜索最优解的过程。即完成  $\arg \max$ 。

为了理解以上的问题，可以先回忆一下5.2.4小节中的公式(5.11)，即  $g(s,t)$  函数的定义，它用于评估一个译文的好与坏。如图5.13所示， $g(s,t)$  函数与公式(5.17)的建模方式非常一致，即  $g(s,t)$  函数中红色部分描述译文  $t$  的可能性大小，对应翻译模型  $P(s|t)$ ；蓝色部分描述译文的平滑或流畅程度，对应语言模型  $P(t)$ 。尽管这种对应并不十分严格的，但也可以看出在处理机器翻译问题上，很多想法的本质是一样的。

$$g(s,t) = \prod_{(j,i) \in \hat{A}} P(s_j, t_i) \times P_{lm}(t)$$

$P(s|t)$ 
翻译模型
 $P(t)$ 
语言模型

图 5.13 IBM 模型与公式(5.11)的对应关系

但  $g(s,t)$  函数的建模很粗糙，因此下面将介绍的 IBM 模型对问题有着更严谨的定义与建模。对于语言模型  $P(t)$  和解码过程在前面的内容中都有介绍，所以本章的后半部分会重点介绍如何定义翻译模型  $P(s|t)$  以及如何训练模型参数。

### 5.4.1 词对齐

IBM 模型的一个基本的假设是词对齐假设。词对齐描述了源语言句子和目标语句子之间单词级别的对应。具体来说，给定源语句子  $s = \{s_1 \dots s_m\}$  和目标语译文  $t = \{t_1 \dots t_l\}$ ，IBM 模型假设词对齐具有如下两个性质。

- 一个源语言单词只能对应一个目标语言单词。如图5.14所示，(a) 和 (c) 都满足该条件，尽管 (c) 中的“谢谢”和“你”都对应“thanks”，但并不违背这个约束。而 (b) 不满足约束，因为“谢谢”同时对应到了两个目标语单词上。这个约束条件也导致这里的词对齐变成一种**非对称的词对齐** (Asymmetric Word Alignment)，因为它只对源语言做了约束，但是目标语言没有。使用这样的约束的目的是为了减少建模的复杂度。在 IBM 模型之后的方法中也提出了双向词对齐，用于建模一个源语言单词对应到多个目标语单词的情况<sup>[240]</sup>。
- 源语言单词可以翻译为空，这时它对应到一个虚拟或伪造的目标语单词  $t_0$ 。在图5.15所示的例子中，“在”没有对应到“on the table”中的任意一个词，而是把

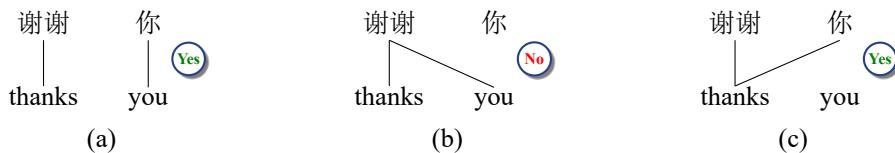


图 5.14 不同词对齐对比

它对应到  $t_0$  上。这样，所有的源语言单词都能找到一个目标语单词对应。这种设计也很好地引入了**空对齐**（Empty Alignment）的思想，即源语言单词不对应任何真实存在的单词的情况。而这种空对齐的情况在翻译中是频繁出现的，比如虚词的翻译。

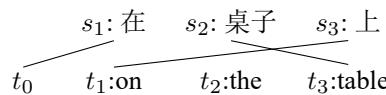


图 5.15 词对齐实例 (“在” 对应到  $t_0$ )

通常，把词对齐记为  $a$ ，它由  $a_1$  到  $a_m$  共  $m$  个词对齐连接组成，即  $a = \{a_1 \dots a_m\}$ 。 $a_j$  表示第  $j$  个源语单词  $s_j$  对应的目标语单词的位置。在图5.15的例子中，词对齐关系可以记为  $a_1 = 0, a_2 = 3, a_3 = 1$ ，即第 1 个源语单词“在”对应到目标语译文的第 0 个位置，第 2 个源语单词“桌子”对应到目标语译文的第 3 个位置，第 3 个源语单词“上”对应到目标语译文的第 1 个位置。

## 5.4.2 基于词对齐的翻译模型

直接准确估计  $P(s|t)$  很难，训练数据只能覆盖整个样本空间非常小的一部分，绝大多数句子在训练数据中一次也没出现过。为了解决这个问题，IBM 模型假设：句子之间的对应可以由单词之间的对应进行表示。于是，翻译句子的概率可以被转化为词对齐生成的概率：

$$P(s|t) = \sum_a P(s,a|t) \quad (5.18)$$

公式(5.18)使用了简单的全概率公式把  $P(s|t)$  进行展开。通过访问  $s$  和  $t$  之间所有可能的词对齐  $a$ ，并把对应的对齐概率进行求和，得到了  $t$  到  $s$  的翻译概率。这里，可以把词对齐看作翻译的隐含变量，这样从  $t$  到  $s$  的生成就变为从  $t$  同时生成  $s$  和隐含变量  $a$  的问题。引入隐含变量是生成式模型常用的手段，通过使用隐含变量，可以把较为困难的端到端学习问题转化为分步学习问题。

举个例子说明公式(5.18)的实际意义。如图5.16所示，可以把从“谢谢你”到“thank you”的翻译分解为9种可能的词对齐。因为源语言句子 $s$ 有2个词，目标语言句子 $t$ 加上空标记 $t_0$ 共3个词，因此每个源语言单词有3个可能对齐的位置，整

一个句子共有  $3 \times 3 = 9$  种可能的词对齐。

$$\begin{aligned}
 & P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + \\
 & P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + \\
 & P(t_0 \text{ 谢谢 } t_0 \text{ thank you}) + P(t_0 \text{ 谢謝 } t_0 \text{ thank you}) + P(t_0 \text{ 谢謝 } t_0 \text{ thank you}) = P(s|t)
 \end{aligned}$$

图 5.16 一个汉译英句对的所有词对齐可能

接下来的问题是如何定义  $P(s, a|t)$  —— 即定义词对齐的生成概率。但是，隐含变量  $a$  仍然很复杂，因此直接定义  $P(s, a|t)$  也很困难，在 IBM 模型中，为了化简问题， $P(s, a|t)$  被进一步分解。使用链式法则，可以得到：

$$P(s, a|t) = P(m|t) \prod_{j=1}^m P(a_j | a_1^{j-1}, s_1^{j-1}, m, t) P(s_j | a_1^j, s_1^{j-1}, m, t) \quad (5.19)$$

其中  $s_j$  和  $a_j$  分别表示第  $j$  个源语言单词及第  $j$  个源语言单词对齐到的目标位置， $s_1^{j-1}$  表示前  $j-1$  个源语言单词（即  $s_1^{j-1} = \{s_1 \dots s_{j-1}\}$ ）， $a_1^{j-1}$  表示前  $j-1$  个源语言的词对齐（即  $a_1^{j-1} = \{a_1 \dots a_{j-1}\}$ ）， $m$  表示源语句子的长度。公式(5.19)将  $P(s, a|t)$  分解为四个部分，具体含义如下：

- 根据译文  $t$  选择源文  $s$  的长度  $m$ ，用  $P(m|t)$  表示；
- 当确定源语言句子的长度  $m$  后，循环每个位置  $j$  逐次生成每个源语言单词  $s_j$ ，也就是  $\prod_{j=1}^m \cdot$  计算的内容；
- 对于每个位置  $j$ ，根据译文  $t$ 、源文长度  $m$ 、已经生成的源语言单词  $s_1^{j-1}$  和对齐  $a_1^{j-1}$ ，生成第  $j$  个位置的对齐结果  $a_j$ ，用  $P(a_j | a_1^{j-1}, s_1^{j-1}, m, t)$  表示；
- 对于每个位置  $j$ ，根据译文  $t$ 、源文长度  $m$ 、已经生成的源语言单词  $s_1^{j-1}$  和对齐  $a_1^j$ ，生成第  $j$  个位置的源语言单词  $s_j$ ，用  $P(s_j | a_1^j, s_1^{j-1}, m, t)$  表示。

换句话说，当求  $P(s, a|t)$  时，首先根据译文  $t$  确定源语言句子  $s$  的长度  $m$ ；当知道源语言句子有多少个单词后，循环  $m$  次，依次生成第 1 个到第  $m$  个源语言单词；当生成第  $j$  个源语言单词时，要先确定它是由哪个目标语译文单词生成的，即确定生成的源语言单词对应的译文单词的**位置**；当知道了目标语译文单词的位置，就能确定第  $j$  个位置的源语言单词。

需要注意的是公式(5.19)定义的模型并没有做任何化简和假设，也就是说公式的左右两端是严格相等的。在后面的内容中会看到，这种将一个整体进行拆分的方法

可以有助于分步骤化简并处理问题。

### 5.4.3 基于词对齐的翻译实例

用前面图5.15中例子来对公式(5.19)进行说明。例子中，源语言句子“在 桌子上”目标语译文“on the table”之间的词对齐为  $a = \{1-0, 2-3, 3-1\}$ 。公式(5.19)的计算过程如下：

- 首先根据译文确定源文  $s$  的单词数量 ( $m = 3$ )，即  $P(m = 3 | t_0 \text{ on the table})$ ；
- 再确定源语言单词  $s_1$  由谁生成的且生成的是什么。可以看到  $s_1$  由第 0 个目标语单词生成，也就是  $t_0$ ，表示为  $P(a_1 = 0 | \phi, \phi, 3, t_0 \text{ on the table})$ ，其中  $\phi$  表示空。当知道了  $s_1$  是由  $t_0$  生成的，就可以通过  $t_0$  生成源语言第一个单词“在”，即  $P(s_1 = “在” | \{1-0\}, \phi, 3, t_0 \text{ on the table})$ ；
- 类似于生成  $s_1$ ，依次确定源语言单词  $s_2$  和  $s_3$  由谁生成且生成的是什么；

最后得到基于词对齐  $a$  的翻译概率为：

$$\begin{aligned}
 P(s, a | t) &= P(m | t) \prod_{j=1}^m P(a_j | a_1^{j-1}, s_1^{j-1}, m, t) P(s_j | a_1^j, s_1^{j-1}, m, t) \\
 &= P(m = 3 | t_0 \text{ on the table}) \times \\
 &\quad P(a_1 = 0 | \phi, \phi, 3, t_0 \text{ on the table}) \times \\
 &\quad P(s_1 = 在 | \{1-0\}, \phi, 3, t_0 \text{ on the table}) \times \\
 &\quad P(a_2 = 3 | \{1-0\}, 在, 3, t_0 \text{ on the table}) \times \\
 &\quad P(s_2 = 桌子 | \{1-0, 2-3\}, 在, 3, t_0 \text{ on the table}) \times \\
 &\quad P(a_3 = 1 | \{1-0, 2-3\}, 在 桌子, 3, t_0 \text{ on the table}) \times \\
 &\quad P(s_3 = 上 | \{1-0, 2-3, 3-1\}, 在 桌子, 3, t_0 \text{ on the table}) \quad (5.20)
 \end{aligned}$$

## 5.5 IBM 模型 1

公式(5.18)和公式(5.19)把翻译问题定义为对译文和词对齐同时进行生成的问题。其中有两个问题：

- 首先，公式(5.18)的右端 ( $\sum_a P(s, a | t)$ ) 要求对所有的词对齐概率进行求和，但是词对齐的数量随着句子长度是呈指数增长，如何遍历所有的对齐  $a$ ？
- 其次，公式(5.19)虽然对词对齐的问题进行了描述，但是模型中的很多参数仍然很复杂，如何计算  $P(m | t)$ 、 $P(a_j | a_1^{j-1}, s_1^{j-1}, m, t)$  和  $P(s_j | a_1^j, s_1^{j-1}, m, t)$ ？

针对这些问题，Brown 等人总共提出了 5 种解决方案，这也就是被后人所熟知的 5 个 IBM 翻译模型。第一个问题可以通过一定的数学或者工程技巧进行求解；第二

个问题可以通过一些假设进行化简，依据化简的层次和复杂度不同，可以分为 IBM 模型 1、IBM 模型 2、IBM 模型 3、IBM 模型 4 以及 IBM 模型 5。本节首先介绍较为简单的 IBM 模型 1。

### 5.5.1 IBM 模型 1 的建模

IBM 模型 1 对公式(5.19)中的三项进行了简化。具体方法如下：

- 假设  $P(m|t)$  为常数  $\varepsilon$ ，即源语言句子长度的生成概率服从均匀分布，如下：

$$P(m|t) \equiv \varepsilon \quad (5.21)$$

- 对齐概率  $P(a_j|a_1^{j-1}, s_1^{j-1}, m, t)$  仅依赖于译文长度  $l$ ，即每个词对齐连接的生成概率也服从均匀分布。换句话说，对于任意源语言位置  $j$  对齐到目标语言任意位置都是等概率的。比如译文为“on the table”，再加上  $t_0$  共 4 个位置，相应的，任意源语单词对齐到这 4 个位置的概率是一样的。具体描述如下：

$$P(a_j|a_1^{j-1}, s_1^{j-1}, m, t) \equiv \frac{1}{l+1} \quad (5.22)$$

- 源语单词  $s_j$  的生成概率  $P(s_j|a_1^j, s_1^{j-1}, m, t)$  仅依赖与其对齐的译文单词  $t_{a_j}$ ，即词汇翻译概率  $f(s_j|t_{a_j})$ 。此时词汇翻译概率满足  $\sum_{s_j} f(s_j|t_{a_j}) = 1$ 。比如在图5.17表示的例子中，源语单词“上”出现的概率只和与它对齐的单词“on”有关系，与其他单词没有关系。

$$P(s_j|a_1^j, s_1^{j-1}, m, t) \equiv f(s_j|t_{a_j}) \quad (5.23)$$

用一个简单的例子对公式(5.23)进行说明。比如，在图5.17中，“桌子”对齐到“table”，可被描述为  $f(s_2|t_{a_2}) = f(\text{“桌子”} | \text{“table”})$ ，表示给定“table”翻译为“桌子”的概率。通常， $f(s_2|t_{a_2})$  被认为是一种概率词典，它反应了两种语言词汇一级的对应关系。

将上述三个假设和公式(5.19)代入公式(5.18)中，得到  $P(s|t)$  的表达式：

$$\begin{aligned} P(s|t) &= \sum_a P(s, a|t) \\ &= \sum_a P(m|t) \prod_{j=1}^m P(a_j|a_1^{j-1}, s_1^{j-1}, m, t) P(s_j|a_1^j, s_1^{j-1}, m, t) \\ &= \sum_a \varepsilon \prod_{j=1}^m \frac{1}{l+1} f(s_j|t_{a_j}) \\ &= \sum_a \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m f(s_j|t_{a_j}) \end{aligned} \quad (5.24)$$

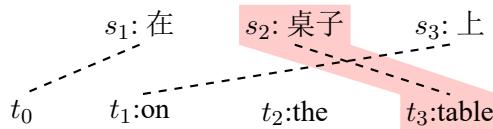


图 5.17 汉译英双语句对及词对齐

在公式(5.24)中，需要遍历所有的词对齐，即  $\sum_a \cdot$ 。但这种表示不够直观，因此可以把这个过程重新表示为如下形式：

$$P(s|t) = \sum_{a_1=0}^l \cdots \sum_{a_m=0}^l \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m f(s_j|t_{a_j}) \quad (5.25)$$

公式(5.25)分为两个主要部分。第一部分：遍历所有的对齐  $a$ 。其中  $a$  由  $\{a_1, \dots, a_m\}$  组成，每个  $a_j \in \{a_1, \dots, a_m\}$  从译文的开始位置 (0) 循环到截止位置 ( $l$ )。如图5.18表示的例子，描述的是源语单词  $s_3$  从译文的开始  $t_0$  遍历到结尾  $t_3$ ，即  $a_3$  的取值范围。第二部分：对于每个  $a$  累加对齐概率  $P(s, a|t) = \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m f(s_j|t_{a_j})$ 。

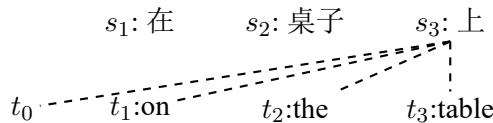


图 5.18 公式(5.25)第一部分实例

这样就得到了 IBM 模型 1 中句子翻译概率的计算式。可以看出 IBM 模型 1 的假设把翻译模型化简成了非常简单的形式。对于给定的  $s$ ,  $a$  和  $t$ , 只要知道  $\varepsilon$  和  $f(s_j|t_{a_j})$  就可以计算出  $P(s|t)$ 。

## 5.5.2 解码及计算优化

如果模型参数给定，可以使用 IBM 模型 1 对新的句子进行翻译。比如，可以使用5.2.5节描述的解码方法搜索最优译文。在搜索过程中，只需要通过公式(5.25)计算每个译文候选的 IBM 模型翻译概率。但是，公式(5.25)的高计算复杂度导致这些模型很难直接使用。以 IBM 模型 1 为例，这里把公式(5.25)重写为：

$$P(s|t) = \frac{\varepsilon}{(l+1)^m} \underbrace{\sum_{a_1=0}^l \cdots \sum_{a_m=0}^l}_{(l+1)^m \text{ 次循环}} \underbrace{\prod_{j=1}^m f(s_j|t_{a_j})}_{m \text{ 次循环}} \quad (5.26)$$

可以看到，遍历所有的词对齐需要  $(l+1)^m$  次循环，遍历所有源语言位置累计  $f(s_j|t_{a_j})$  需要  $m$  次循环，因此这个模型的计算复杂度为  $O((l+1)^m m)$ 。当  $m$  较大时，计算这样的模型几乎是不可能的。不过，经过仔细观察，可以发现公式右端的部分有另外

一种计算方法，如下：

$$\sum_{a_1=0}^l \dots \sum_{a_m=0}^l \prod_{j=1}^m f(s_j|t_{a_j}) = \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \quad (5.27)$$

公式(5.27)的技巧在于把若干个乘积的加法（等式左手端）转化为若干加法结果的乘积（等式右手端），这样省去了多次循环，把  $O((l+1)^m m)$  的计算复杂度降为  $O((l+1)m)$ 。此外，公式(5.27)相比公式(5.26)的另一个优点在于，公式(5.27)中乘法的数量更少，因为现代计算机中乘法运算的代价要高于加法，因此公式(5.27)的计算机实现效率更高。图5.19 对这个过程进行了进一步解释。

$$\begin{aligned}
 & \alpha(1,0)\alpha(2,0) + \alpha(1,0)\alpha(2,1) + \alpha(1,0)\alpha(2,2) + \\
 & \alpha(1,1)\alpha(2,0) + \alpha(1,1)\alpha(2,1) + \alpha(1,1)\alpha(2,2) + \\
 & \alpha(1,2)\alpha(2,0) + \alpha(1,2)\alpha(2,1) + \alpha(1,2)\alpha(2,2) \\
 \\ 
 & \sum_{y_1=0}^2 \sum_{y_2=0}^2 \alpha(1,y_1)\alpha(2,y_2) = (\alpha(1,0) + \alpha(1,1) + \alpha(1,2)) \cdot \\
 & \sum_{y_1=0}^2 \sum_{y_2=0}^2 \prod_{x=1}^l \alpha(x,y_x) = (\alpha(2,0) + \alpha(2,1) + \alpha(2,2)) \\
 \\ 
 & \sum_{a_1=0}^l \dots \sum_{a_m=0}^l \prod_{j=1}^m f(s_j|t_{a_j}) = \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)
 \end{aligned}$$

图 5.19  $\sum_{a_1=0}^l \dots \sum_{a_m=0}^l \prod_{j=1}^m f(s_j|t_{a_j}) = \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)$  的实例

接着，利用公式(5.27)的方式，可以把公式(5.25)重写表示为：

$$\text{IBM 模型 1: } P(s|t) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \quad (5.28)$$

公式(5.28)是 IBM 模型 1 的最终表达式，在解码和训练中可以被直接使用。

### 5.5.3 训练

在完成了建模和解码的基础上，剩下的问题是如何得到模型的参数。这也是整个统计机器翻译里最重要的内容。下面将会对 IBM 模型 1 的参数估计方法进行介绍。

## 1. 目标函数

统计机器翻译模型的训练是一个典型的优化问题。简单来说，训练是指在给定数据集（训练集）上调整参数使得目标函数的值达到最大（或最小），此时得到的参数被称为是该模型在该目标函数下的最优解（图5.20）。

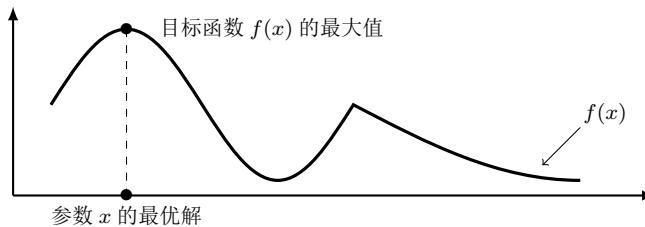


图 5.20 一个目标函数的最优解

在 IBM 模型中，优化的目标函数被定义为  $P(s|t)$ 。也就是，对于给定的句对  $(s, t)$ ，最大化翻译概率  $P(s|t)$ 。这里用符号  $P_\theta(s|t)$  表示模型由参数  $\theta$  决定，模型训练可以被描述为对目标函数  $P_\theta(s|t)$  的优化过程：

$$\hat{\theta} = \arg \max_{\theta} P_\theta(s|t) \quad (5.29)$$

其中， $\arg \max_{\theta}$  表示求最优参数的过程（或优化过程）。

公式(5.29)实际上也是一种基于极大似然的模型训练方法。这里，可以把  $P_\theta(s|t)$  看作是模型对数据描述的一个似然函数，记作  $L(s, t; \theta)$ 。也就是，优化目标是对似然函数的优化： $\{\hat{\theta}\} = \{\arg \max_{\theta \in \Theta} L(s, t; \theta)\}$ ，其中  $\{\hat{\theta}\}$  表示可能有多个结果， $\Theta$  表示参数空间。

回到 IBM 模型的优化问题上。以 IBM 模型 1 为例，优化的目标是最大化翻译概率  $P(s|t)$ 。使用公式(5.28)，可以把这个目标表述为：

$$\begin{aligned} & \max \left( \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right) \\ \text{s.t. } & \text{任意单词 } t_y : \sum_{s_x} f(s_x|t_y) = 1 \end{aligned}$$

其中， $\max(\cdot)$  表示最大化， $\frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)$  是目标函数， $f(s_j|t_i)$  是模型的参数， $\sum_{s_x} f(s_x|t_y) = 1$  是优化的约束条件，以保证翻译概率满足归一化的要求。需要注意的是  $\{f(s_x|t_y)\}$  对应了很多参数，每个源语言单词和每个目标语单词的组合都对应一个参数  $f(s_x|t_y)$ 。

## 2. 优化

可以看到，IBM 模型的参数训练问题本质上是带约束的目标函数优化问题。由于目标函数是可微分函数，解决这类问题的一种常用手法是把带约束的优化问题转化为不带约束的优化问题。这里用到了**拉格朗日乘数法**（Lagrange Multiplier Method），它的基本思想是把含有  $n$  个变量和  $m$  个约束条件的优化问题转化为含有  $n+m$  个变量的无约束优化问题。

这里的目地是  $\max(P_\theta(s|t))$ ，约束条件是对于任意的目标语单词  $t_y$  有  $\sum_{s_x} P(s_x|t_y) = 1$ 。根据拉格朗日乘数法，可以把上述优化问题重新定义为最大化如下拉格朗日函数的问题：

$$L(f, \lambda) = \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) - \sum_{t_y} \lambda_{t_y} (\sum_{s_x} f(s_x|t_y) - 1) \quad (5.30)$$

$L(f, \lambda)$  包含两部分， $\frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)$  是原始的目标函数， $\sum_{t_y} \lambda_{t_y} (\sum_{s_x} f(s_x|t_y) - 1)$  是原始的约束条件乘以拉格朗日乘数  $\lambda_{t_y}$ ，拉格朗日乘数的数量和约束条件的数量相同。图5.21通过图例说明了  $L(f, \lambda)$  各部分的意义。

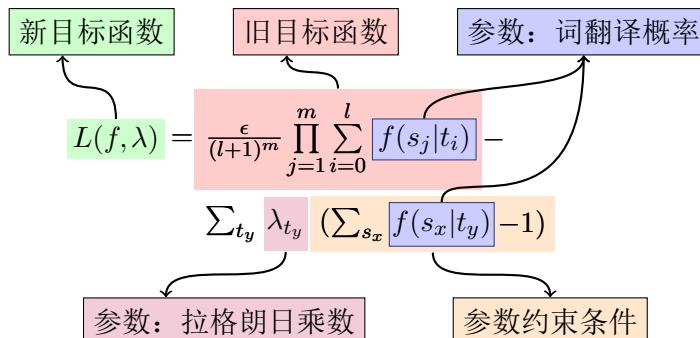


图 5.21 拉格朗日函数的解释 (IBM 模型 1)

因为  $L(f, \lambda)$  是可微分函数，因此可以通过计算  $L(f, \lambda)$  导数为零的点得到极值点。因为这个模型里仅有  $f(s_x|t_y)$  一种类型的参数，只需要对如下导数进行计算。

$$\begin{aligned} \frac{\partial L(f, \lambda)}{\partial f(s_u|t_v)} &= \frac{\partial \left[ \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial f(s_u|t_v)} - \\ &\quad \frac{\partial \left[ \sum_{t_y} \lambda_{t_y} (\sum_{s_x} f(s_x|t_y) - 1) \right]}{\partial f(s_u|t_v)} \\ &= \frac{\varepsilon}{(l+1)^m} \cdot \frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial f(s_u|t_v)} - \lambda_{t_v} \end{aligned} \quad (5.31)$$

这里  $s_u$  和  $t_v$  分别表示源语言和目标语言词表中的某一个单词。为了求  $\frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial f(s_u|t_v)}$ , 这里引入一个辅助函数。令  $g(z) = \alpha z^\beta$  为变量  $z$  的函数, 显然,  $\frac{\partial g(z)}{\partial z} = \alpha \beta z^{\beta-1} = \frac{\beta}{z} \alpha z^\beta = \frac{\beta}{z} g(z)$ 。这里可以把  $\prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)$  看做  $g(z) = \alpha z^\beta$  的实例。首先, 令  $z = \sum_{i=0}^l f(s_u|t_i)$ , 注意  $s_u$  为给定的源语单词。然后, 把  $\beta$  定义为  $\sum_{i=0}^l f(s_u|t_i)$  在  $\prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)$  中出现的次数, 即源语句子中与  $s_u$  相同的单词的个数。

$$\beta = \sum_{j=1}^m \delta(s_j, s_u) \quad (5.32)$$

其中, 当  $x = y$  时,  $\delta(x, y) = 1$ , 否则为 0。

根据  $\frac{\partial g(z)}{\partial z} = \frac{\beta}{z} g(z)$ , 可以得到

$$\begin{aligned} \frac{\partial g(z)}{\partial z} &= \frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial \left[ \sum_{i=0}^l f(s_u|t_i) \right]} \\ &= \frac{\sum_{j=1}^m \delta(s_j, s_u)}{\sum_{i=0}^l f(s_u|t_i)} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \end{aligned} \quad (5.33)$$

根据  $\frac{\partial g(z)}{\partial z}$  和  $\frac{\partial z}{\partial f}$  计算的结果, 可以得到

$$\begin{aligned} \frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial f(s_u|t_v)} &= \frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial \left[ \sum_{i=0}^l f(s_u|t_i) \right]} \cdot \frac{\partial \left[ \sum_{i=0}^l f(s_u|t_i) \right]}{\partial f(s_u|t_v)} \\ &= \frac{\sum_{j=1}^m \delta(s_j, s_u)}{\sum_{i=0}^l f(s_u|t_i)} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \cdot \sum_{i=0}^l \delta(t_i, t_v) \end{aligned} \quad (5.34)$$

将  $\frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \right]}{\partial f(s_u|t_v)}$  进一步代入  $\frac{\partial L(f, \lambda)}{\partial f(s_u|t_v)}$ , 得到  $L(f, \lambda)$  的导数

$$\begin{aligned} \frac{\partial L(f, \lambda)}{\partial f(s_u|t_v)} &= \frac{\varepsilon}{(l+1)^m} \cdot \frac{\partial \left[ \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_{a_j}) \right]}{\partial f(s_u|t_v)} - \lambda_{t_v} \\ &= \frac{\varepsilon}{(l+1)^m} \frac{\sum_{j=1}^m \delta(s_j, s_u) \cdot \sum_{i=0}^l \delta(t_i, t_v)}{\sum_{i=0}^l f(s_u|t_i)} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) - \lambda_{t_v} \end{aligned} \quad (5.35)$$

令  $\frac{\partial L(f, \lambda)}{\partial f(s_u|t_v)} = 0$ , 有

$$f(s_u|t_v) = \frac{\lambda_{t_v}^{-1} \varepsilon}{(l+1)^m} \cdot \frac{\sum_{j=1}^m \delta(s_j, s_u) \cdot \sum_{i=0}^l \delta(t_i, t_v)}{\sum_{i=0}^l f(s_u|t_i)} \prod_{j=1}^m \prod_{i=0}^l f(s_j|t_i) \cdot f(s_u|t_v) \quad (5.36)$$

将上式稍作调整得到下式:

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i) \sum_{j=1}^m \delta(s_j, s_u) \sum_{i=0}^l \delta(t_i, t_v) \frac{f(s_u|t_v)}{\sum_{i=0}^l f(s_u|t_i)} \quad (5.37)$$

可以看出, 这不是一个计算  $f(s_u|t_v)$  的解析式, 因为等式右端仍含有  $f(s_u|t_v)$ 。不过它蕴含着一种非常经典的方法——**期望最大化** (Expectation Maximization) 方法, 简称**EM方法** (或算法)。使用 EM 方法可以利用式5.37迭代地计算  $f(s_u|t_v)$ , 使其最终收敛到最优值。EM 方法的思想是: 用当前的参数, 求似然函数的期望, 之后最大化这个期望同时得到新的一组参数的值。对于 IBM 模型来说, 其迭代过程就是反复使用公式(5.37), 具体如图5.22所示。

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \frac{\varepsilon}{(l+1)^m} \underbrace{\prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)}_{\text{新的参数值}} \underbrace{\sum_{j=1}^m \delta(s_j, s_u) \sum_{i=0}^l \delta(t_i, t_v)}_{\text{旧的参数值}} \underbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^l f(s_u|t_i)}}_{\text{新的参数值}}$$

图 5.22 IBM 模型迭代过程示意图

为了化简  $f(s_u|t_v)$  的计算, 在此对公式(5.37)进行了重新组织, 见图5.23。其中, 红色部分表示翻译概率  $P(s|t)$ ; 蓝色部分表示  $(s_u, t_v)$  在句对  $(s, t)$  中配对的总次数, 即 “ $t_v$  翻译为  $s_u$ ” 在所有对齐中出现的次数; 绿色部分表示  $f(s_u|t_v)$  对于所有的  $t_i$  的相对值, 即 “ $t_v$  翻译为  $s_u$ ” 在所有对齐中出现的相对概率; 蓝色与绿色部分相乘表示 “ $t_v$  翻译为  $s_u$ ” 这个事件出现次数的期望的估计, 称之为**期望频次** (Expected Count)。

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \frac{\varepsilon}{(l+1)^m} \underbrace{\prod_{j=1}^m \sum_{i=0}^l f(s_j|t_i)}_{P(s|t)} \underbrace{\sum_{j=1}^m \delta(s_j, s_u) \sum_{i=0}^l \delta(t_i, t_v)}_{(s_u, t_v) \text{ 在句对 } (s, t) \text{ 中配对的总次数}} \underbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^l f(s_u|t_i)}}_{f(s_u|t_v) \text{ 对于所有的 } t_i \text{ 的相对值}}$$

||

“ $t_v$  翻译为  $s_u$ ” 这个事件  
出现次数的期望的估计  
称之为期望频次

图 5.23 公式(5.37)的解释

期望频次是事件在其分布下出现次数的期望。另  $c_{\mathbb{E}}(X)$  为事件  $X$  的期望频次,

其计算公式为：

$$c_{\mathbb{E}}(X) = \sum_i c(x_i) \cdot P(x_i) \quad (5.38)$$

其中  $c(x_i)$  表示  $X$  取  $x_i$  时出现的次数， $P(x_i)$  表示  $X = x_i$  出现的概率。图5.24展示了事件  $X$  的期望频次的详细计算过程。其中  $x_1$ 、 $x_2$  和  $x_3$  分别表示事件  $X$  出现 2 次、1 次和 5 次的情况。

$x_i$	$c(x_i)$	$x_i$	$c(x_i)$	$P(x_i)$	$c(x_i) \cdot P(x_i)$
$x_1$	2	$x_1$	2	0.1	0.2
$x_2$	1	$x_2$	1	0.3	0.3
$x_3$	5	$x_3$	5	0.2	1.0
总频次 = 8					$c_{\mathbb{E}}(X) = 0.2 + 0.3 + 1.0 = 1.5$

图 5.24 总频次（左）和期望频次（右）的实例

因为在  $P(s|t)$  中， $t_v$  翻译（连接）到  $s_u$  的期望频次为：

$$c_{\mathbb{E}}(s_u|t_v; s, t) \equiv \sum_{j=1}^m \delta(s_j, s_u) \cdot \sum_{i=0}^l \delta(t_i, t_v) \cdot \frac{f(s_u|t_v)}{\sum_{i=0}^l f(s_u|t_i)} \quad (5.39)$$

所以公式5.37可重写为：

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \cdot P(s|t) \cdot c_{\mathbb{E}}(s_u|t_v; s, t) \quad (5.40)$$

在此如果令  $\lambda'_{t_v} = \frac{\lambda_{t_v}}{P(s|t)}$ ， 可得：

$$\begin{aligned} f(s_u|t_v) &= \lambda_{t_v}^{-1} \cdot P(s|t) \cdot c_{\mathbb{E}}(s_u|t_v; s, t) \\ &= (\lambda'_{t_v})^{-1} \cdot c_{\mathbb{E}}(s_u|t_v; s, t) \end{aligned} \quad (5.41)$$

又因为 IBM 模型对  $f(\cdot|\cdot)$  的约束如下：

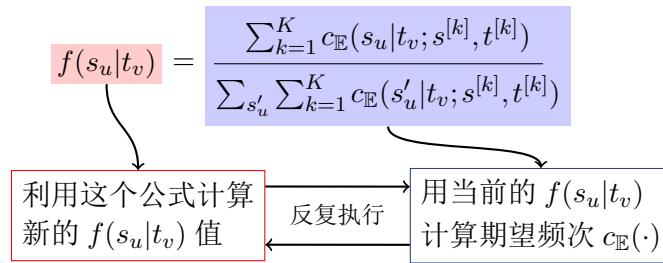
$$\forall t_y : \sum_{s_x} f(s_x|t_y) = 1 \quad (5.42)$$

为了满足  $f(\cdot|\cdot)$  的概率归一化约束，易得  $\lambda'_{t_v}$  为：

$$\lambda'_{t_v} = \sum_{s'_u} c_{\mathbb{E}}(s'_u|t_v; s, t) \quad (5.43)$$

因此， $f(s_u|t_v)$  的计算式可再一步变换为下式：

$$f(s_u|t_v) = \frac{c_{\mathbb{E}}(s_u|t_v; s, t)}{\sum_{s'_u} c_{\mathbb{E}}(s'_u|t_v; s, t)} \quad (5.44)$$

图 5.25  $f(s_u|t_v)$  的计算公式和迭代过程

进一步，假设有  $K$  个互译的句对（称作平行语料）： $\{(s^{[1]}, t^{[1]}), \dots, (s^{[K]}, t^{[K]})\}$ ， $f(s_u|t_v)$  的期望频次为：

$$c_{\mathbb{E}}(s_u|t_v) = \sum_{k=1}^K c_{\mathbb{E}}(s_u|t_v; s^{[k]}, t^{[k]}) \quad (5.45)$$

于是有  $f(s_u|t_v)$  的计算公式和迭代过程图5.25所示。完整的 EM 算法如图5.26所示。其中 E-Step 对应 4-5 行，目的是计算  $c_{\mathbb{E}}(\cdot)$ ；M-Step 对应 6-9 行，目的是计算  $f(\cdot|\cdot)$ 。

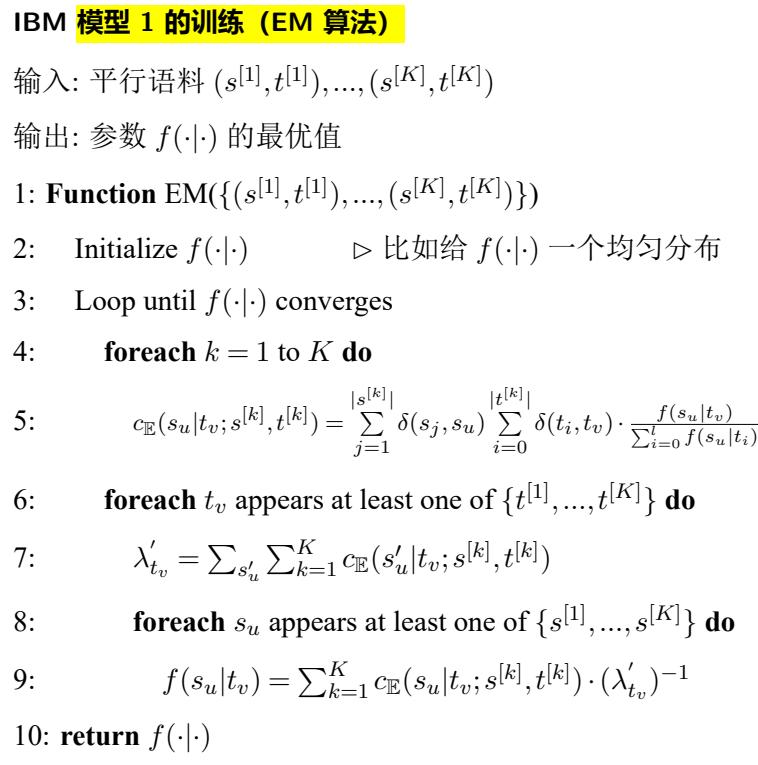


图 5.26 EM 算法流程图 (IBM 模型 1)

至此，本章完成了对 IBM 模型 1 训练方法的介绍。其可以通过图5.25所示的算

法进行实现。算法最终的形式并不复杂，因为只需要遍历每个句对，之后计算  $f(\cdot|\cdot)$  的期望频次，最后估计新的  $f(\cdot|\cdot)$ ，这个过程迭代直至  $f(\cdot|\cdot)$  收敛至稳定状态。

## 5.6 小结及拓展阅读

本章对 IBM 系列模型中的 IBM 模型 1 进行了详细的介绍和讨论，从一个简单的基于单词的翻译模型开始，本章从建模、解码、训练多个维度对统计机器翻译进行了描述，期间涉及了词对齐、优化等多个重要概念。IBM 模型共分为 5 个模型，对翻译问题的建模依次由浅入深，同时模型复杂度也依次增加，我们将在第六章对 IBM 模型 2-5 进行详细的介绍和讨论。IBM 模型作为入门统计机器翻译的“必经之路”，其思想对今天的机器翻译仍然产生着影响。虽然单独使用 IBM 模型进行机器翻译现在已经不多见，甚至很多从事神经机器翻译等前沿研究的人对 IBM 模型已经逐渐淡忘，但是不能否认 IBM 模型标志着一个时代的开始。从某种意义上讲，当使用公式  $\hat{t} = \arg \max_t P(t|s)$  描述机器翻译问题的时候，或多或少都在使用与 IBM 模型相似的思想。

当然，本书也无法涵盖 IBM 模型的所有内涵，很多内容需要感兴趣的读者继续研究和挖掘。其中最值得关注的是统计词对齐问题。由于词对齐是 IBM 模型训练的间接产物，因此 IBM 模型成为了自动词对齐的重要方法。比如 IBM 模型训练装置 GIZA++ 更多的是被用于自动词对齐任务，而非简单的训练 IBM 模型参数<sup>[240]</sup>。

- 在 IBM 基础模型之上，有很多改进的工作。例如，对空对齐、低频词进行额外处理<sup>[241]</sup>；考虑源语言-目标语言和目标语言-源语言双向词对齐进行更好地词对齐对称化<sup>[242]</sup>；使用词典、命名实体等多种信息对模型进行改进<sup>[243]</sup>；通过引入短语增强 IBM 基础模型<sup>[244]</sup>；引入相邻单词对齐之间的依赖关系增加模型健壮性<sup>[245]</sup>等；也可以对 IBM 模型的正向和反向结果进行对称化处理，以得到更加准确词对齐结果<sup>[240]</sup>。
- 随着词对齐概念的不断深入，也有很多词对齐方面的工作并不依赖 IBM 模型。比如，可以直接使用判别式模型利用分类器解决词对齐问题<sup>[246]</sup>；使用带参数控制的动态规划方法来提高词对齐准确率<sup>[247]</sup>；甚至可以把对齐的思想用于短语和句法结构的双语对应<sup>[248]</sup>；无监督的对称词对齐方法，正向和反向模型联合训练，结合数据的相似性<sup>[249]</sup>；除了 GIZA++，研究人员也开发了很多优秀的自动对齐工具，比如，FastAlign<sup>[250]</sup>、Berkeley Word Aligner<sup>[251]</sup>等，这些工具现在也有很广泛的应用。
- 一种较为通用的词对齐评价标准是对齐错误率（Alignment Error Rate, AER）<sup>[252]</sup>。在此基础之上也可以对词对齐评价方法进行改进，以提高对齐质量与机器翻译评价得分 BLEU 的相关性<sup>[253, 254, 255]</sup>。也有工作通过统计机器翻译系统性能的提升来评价对齐质量<sup>[252]</sup>。不过，在相当长的时间内，词对齐质量对机器翻译系统的影响究竟如何并没有统一的结论。有些时候，词对齐的错误率下降了，但是

机器翻译系统的译文品质没有带来性能提升。但是，这个问题比较复杂，需要进一步的论证。不过，可以肯定的是，词对齐可以帮助人们分析机器翻译的行为。甚至在最新的神经机器翻译中，如何在神经网络模型中寻求两种语言单词之间的对应关系也是对模型进行解释的有效手段之一<sup>[256]</sup>。

- 基于单词的翻译模型的解码问题也是早期研究者所关注的。比较经典的方法的是贪婪方法<sup>[79]</sup>。也有研究者对不同的解码方法进行了对比<sup>[78]</sup>，并给出了一些加速解码的思路。随后，也有工作进一步对这些方法进行改进<sup>[257, 258]</sup>。实际上，基于单词的模型的解码是一个 NP 完全问题<sup>[238]</sup>，这也是为什么机器翻译的解码十分困难的原因。关于翻译模型解码算法的时间复杂度也有很多讨论<sup>[259, 260, 261]</sup>。



## 6. 基于扭曲度和繁衍率的模型

第五章展示了一种基于单词的翻译模型。这种模型的形式非常简单，而且其隐含的词对齐信息具有较好的可解释性。不过，语言翻译的复杂性远远超出人们的想象。语言翻译主要有两方面挑战——如何对“调序”问题进行建模以及如何对“一对多翻译”问题进行建模。一方面，调序是翻译问题中所特有的现象，比如，汉语到日语的翻译中，需要对谓词进行调序。另一方面，一个单词在另一种语言中可能会被翻译为多个连续的词，比如，汉语“联合国”翻译到英语会对应三个单词“The United Nations”。这种现象也被称作一对多翻译，它与句子长度预测有着密切的联系。

无论是调序还是一对多翻译，简单的翻译模型（如 IBM 模型 1）都无法对其进行很好的处理。因此，需要考虑对这两个问题单独进行建模。本章将会对机器翻译中两个常用的概念进行介绍——扭曲度（Distortion）和繁衍率（Fertility）。它们可以被看作是对调序和一对多翻译现象的一种统计描述。基于此，本章会进一步介绍基于扭曲度和繁衍率的翻译模型，建立相对完整的基于单词的统计建模体系。相关的技术和概念在后续章节也会被进一步应用。

### 6.1 基于扭曲度的模型

下面将介绍扭曲度在机器翻译中的定义及使用方法。这也带来了两个新的翻译模型——IBM 模型 2<sup>[10]</sup> 和 HMM<sup>[262]</sup>。

### 6.1.1 什么是扭曲度

**调序** (Reordering) 是自然语言翻译中特有的语言现象。造成这个现象的主要原因在于不同语言之间语序的差异，比如，汉语是“主谓宾”结构，而日语是“主宾谓”结构。即使在句子整体结构相似的语言上进行翻译，调序也是频繁出现的现象。如图6.1所示，当一个主动语态的汉语句子翻译为一个被动语态的英语句子时，如果直接顺序翻译，那么翻译结果 “I with you am satisfied” 很明显不符合英语语法。这里就需要采取一些方法和手段在翻译过程中对词或短语进行调序，从而得到正确的翻译结果。

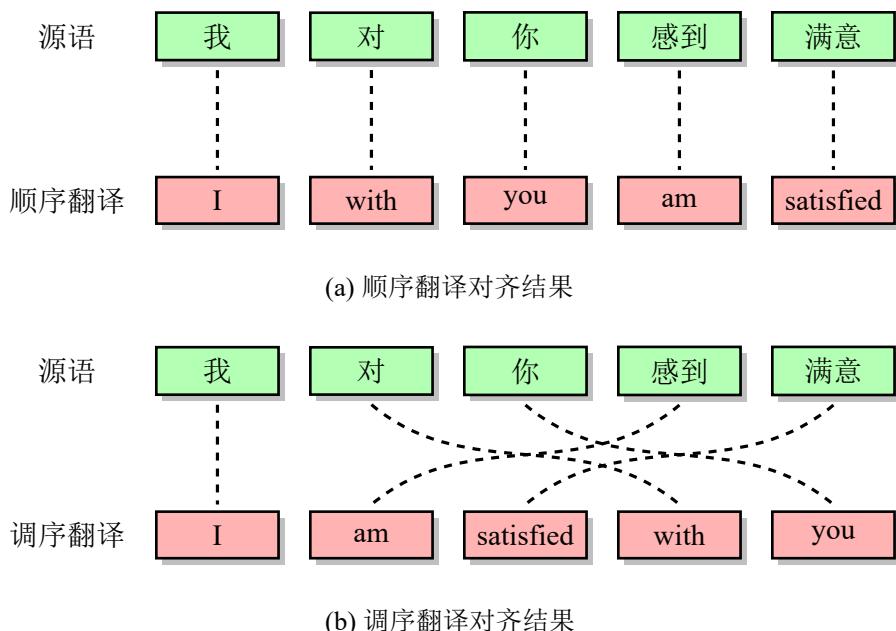


图 6.1 顺序翻译和调序翻译的实例对比

在对调序问题进行建模的方法中，最基本的是使用调序距离方法。这里，可以假设完全进行顺序翻译时，调序的“代价”是最低的。当调序出现时，可以用调序相对于顺序翻译产生的位置偏移来度量调序的程度，也被称为调序距离。图6.2展示了翻译时两种语言中词的对齐矩阵。比如，在图6.2(a) 中，系统需要跳过“对”和“你”来翻译“感到”和“满意”，之后再回过头翻译“对”和“你”，这样就完成了对单词的调序。这时可以简单地把需要跳过的单词数看作一种距离。

可以看到，调序距离实际上是在度量目标语言词序相对于源语言词序的一种扭曲程度。因此，也常常把这种调序距离称作**扭曲度** (Distortion)。调序距离越大对应的扭曲度也越大。比如，可以明显看出图6.2(b) 中调序的扭曲度要比图6.2(a) 中调序的扭曲度大，因此6.2(b) 实例的调序代价也更大。

在机器翻译中使用扭曲度进行翻译建模是一种十分自然的想法。接下来，会介绍两个基于扭曲度的翻译模型，分别是 IBM 模型 2 和隐马尔可夫模型。不同于 IBM

模型 1，它们利用了单词的位置信息定义了扭曲度，并将扭曲度融入翻译模型中，使得对翻译问题的建模更加合理。

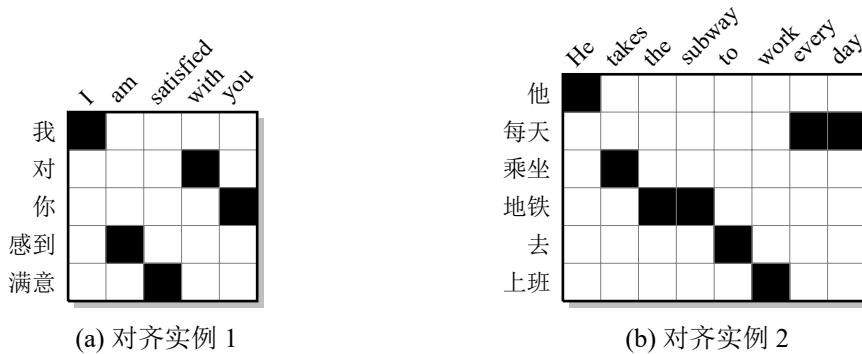


图 6.2 汉语到英语翻译的对齐矩阵

### 6.1.2 IBM 模型 2

对于建模来说，IBM 模型 1 很好地化简了翻译问题，但是由于使用了很强的假设，导致模型和实际情况有较大差异。其中一个比较严重的问题是假设词对齐的生成概率服从均匀分布。IBM 模型 2 抛弃了这个假设<sup>[10]</sup>。它认为词对齐是有倾向性的，它与源语言单词的位置和目标语言单词的位置有关。具体来说，对齐位置  $a_j$  的生成概率与位置  $j$ 、源语言句子长度  $m$  和目标语言句子长度  $l$  有关，形式化表述为：

$$P(a_j | a_1^{j-1}, s_1^{j-1}, m, t) \equiv a(a_j | j, m, l) \quad (6.1)$$

这里还用第五章中的例子（图6.3）来进行说明。在 IBM 模型 1 中，“桌子”对齐到目标语言四个位置的概率是一样的。但在 IBM 模型 2 中，“桌子”对齐到“table”被形式化为  $a(a_j | j, m, l) = a(3 | 2, 3, 3)$ ，意思是对于源语言位置 2 ( $j = 2$ ) 的词，如果它的源语言和目标语言都是 3 个词 ( $l = 3, m = 3$ )，对齐到目标语言位置 3 ( $a_j = 3$ ) 的概率是多少？因为  $a(a_j | j, m, l)$  也是模型需要学习的参数，因此“桌子”对齐到不同目标语言单词的概率也是不一样的。理想的情况下，通过  $a(a_j | j, m, l)$ ，“桌子”对齐到“table”应该得到更高的概率。

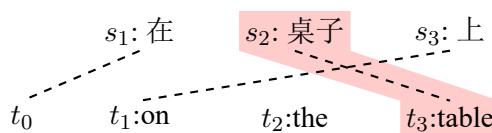


图 6.3 汉译英句对及词对齐

IBM 模型 2 的其他假设均与模型 1 相同，即源语言长度预测概率及源语言单词生成概率被定义为：

$$P(m|t) \equiv \varepsilon \quad (6.2)$$

$$P(s_j|a_1^j, s_1^{j-1}, m, t) \equiv f(s_j|t_{a_j}) \quad (6.3)$$

把公式(6.1)、(6.2)和(6.3)重新带入公式  $P(s, a|t) = P(m|t) \prod_{j=1}^m P(a_j|a_1^{j-1}, s_1^{j-1}, m, t)$   $P(s_j|a_1^j, s_1^{j-1}, m, t)$  和  $P(s|t) = \sum_a P(s, a|t)$ , 可以得到 IBM 模型 2 的数学描述:

$$\begin{aligned} P(s|t) &= \sum_a P(s, a|t) \\ &= \sum_{a_1=0}^l \cdots \sum_{a_m=0}^l \varepsilon \prod_{j=1}^m a(a_j|j, m, l) f(s_j|t_{a_j}) \end{aligned} \quad (6.4)$$

类似于模型 1, 模型 2 的表达式(6.4)也能被拆分为两部分进行理解。第一部分: 遍历所有的  $a$ ; 第二部分: 对于每个  $a$  累加对齐概率  $P(s, a|t)$ , 即计算对齐概率  $a(a_j|j, m, l)$  和词汇翻译概率  $f(s_j|t_{a_j})$  对于所有源语言位置的乘积。

同样的, 模型 2 的解码及训练优化和模型 1 的十分相似, 在此不再赘述, 详细推导过程可以参看第五章5.5小节解码及计算优化部分。这里直接给出 IBM 模型 2 的最终表达式:

$$P(s|t) = \varepsilon \prod_{j=1}^m \sum_{i=0}^l a(i|j, m, l) f(s_j|t_i) \quad (6.5)$$

### 6.1.3 隐马尔可夫模型

IBM 模型把翻译问题定义为生成词对齐的问题, 模型翻译质量的好坏与词对齐有着非常紧密的联系。IBM 模型 1 假设对齐概率仅依赖于目标语言句子长度, 即对齐概率服从均匀分布; IBM 模型 2 假设对齐概率与源语言、目标语言的句子长度以及源语言位置和目标语言位置相关。虽然 IBM 模型 2 已经覆盖了一部分词对齐问题, 但是该模型只考虑到了单词的绝对位置, 并未考虑到相邻单词间的关系。图6.4 展示了一个简单的实例, 可以看到的是, 汉语的每个单词都被分配给了英语句子中的每一个单词, 但是单词并不是任意分布在各个位置上的, 而是倾向于生成簇。也就是说, 如果源语言的两个单词位置越近, 它们的译文在目标语言句子中的位置也越近。

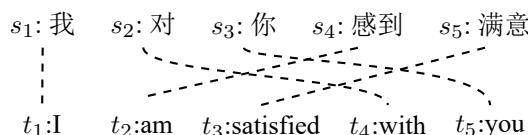


图 6.4 汉译英句对及对齐

针对此问题，基于 HMM 的词对齐模型抛弃了 IBM 模型 1-2 的绝对位置假设，将一阶隐马尔可夫模型用于词对齐问题<sup>[262]</sup>。HMM 词对齐模型认为，单词与单词之间并不是毫无联系的，对齐概率应该取决于对齐位置的差异而不是本身单词所在的位置。具体来说，位置  $j$  的对齐概率  $a_j$  与前一个位置  $j-1$  的对齐位置  $a_{j-1}$  和译文长度  $l$  有关，形式化的表述为：

$$P(a_j|a_1^{j-1}, s_1^{j-1}, m, t) \equiv P(a_j|a_{j-1}, l) \quad (6.6)$$

这里用图6.4的例子对公式进行说明。在 IBM 模型 1-2 中，单词的对齐都是与单词所在的绝对位置有关。但在 HMM 词对齐模型中，“你”对齐到“you”被形式化为  $P(a_j|a_{j-1}, l) = P(5|4, 5)$ ，意思是对于源语言位置  $3(j=3)$  上的单词，如果它的译文是第 5 个目标语言单词，上一个对齐位置是 4( $a_2=4$ )，对齐到目标语言位置 5( $a_j=5$ ) 的概率是多少？理想的情况下，通过  $P(a_j|a_{j-1}, l)$ ，“你”对齐到“you”应该得到更高的概率，并且由于源语言单词“对”和“你”距离很近，因此其对应的对齐位置“with”和“you”的距离也应该很近。

把公式  $P(s_j|a_1^j, s_1^{j-1}, m, t) \equiv f(s_j|t_{a_j})$  和(6.6)重新带入公式  $P(s, a|t) = P(m|t)$   $\prod_{j=1}^m P(a_j|a_1^{j-1}, s_1^{j-1}, m, t)P(s_j|a_1^j, s_1^{j-1}, m, t)$  和  $P(s|t) = \sum_a P(s, a|t)$ ，可得 HMM 词对齐模型的数学描述：

$$P(s|t) = \sum_a P(m|t) \prod_{j=1}^m P(a_j|a_{j-1}, l) f(s_j|t_{a_j}) \quad (6.7)$$

此外，为了使得 HMM 的对齐概率  $P(a_j|a_{j-1}, l)$  满足归一化的条件，这里还假设其对齐概率只取决于  $a_j - a_{j-1}$ ，即：

$$P(a_j|a_{j-1}, l) = \frac{\mu(a_j - a_{j-1})}{\sum_{i=1}^l \mu(i - a_{j-1})} \quad (6.8)$$

其中， $\mu(\cdot)$  是隐马尔可夫模型的参数，可以通过训练得到。

需要注意的是，公式(6.7)之所以被看作是一种隐马尔可夫模型，是由于其形式与标准的一阶隐马尔可夫模型无异。 $P(a_j|a_{j-1}, l)$  可以被看作是一种状态转移概率， $f(s_j|t_{a_j})$  可以被看作是一种发射概率。关于隐马尔可夫模型具体的数学描述也可参考第三章中的相关内容。

## 6.2 基于繁衍率的模型

下面介绍翻译中的一对多问题，以及这个问题所带来的句子长度预测问题。

### 6.2.1 什么是繁衍率

从前面的介绍可知，IBM 模型 1 和模型 2 把不同的源语言单词看作相互独立的单元来进行词对齐和翻译。换句话说，即使某个源语言短语中的两个单词都对齐到同一个目标语单词，它们之间也是相互独立的。这样 IBM 模型 1 和模型 2 对于多个源语言单词对齐到同一个目标语单词的情况并不能很好地进行描述。

这里将会给出另一个翻译模型，能在一定程度上解决上面提到的问题<sup>[10, 240]</sup>。该模型把目标语言生成源语言的过程分解为如下几个步骤：首先，确定每个目标语言单词生成源语言单词的个数，这里把它称为繁衍率或产出率（Fertility）；其次，决定目标语言句子中每个单词生成的源语言单词都是什么，即决定生成的第一个源语言单词是什么，生成的第二个源语言单词是什么，以此类推。这样每个目标语言单词就对应了一个源语言单词列表；最后把各组源语言单词列表中的每个单词都放置到合适的位置上，完成目标语言译文到源语言句子的生成。

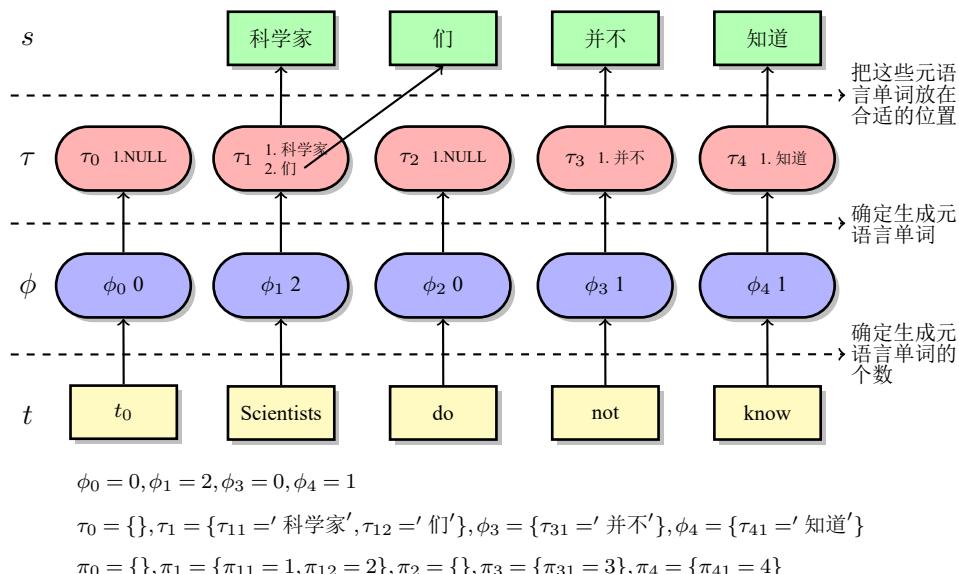


图 6.5 基于产出率的翻译模型执行过程

对于句对  $(s, t)$ ，令  $\varphi$  表示产出率，同时令  $\tau$  表示每个目标语言单词对应的源语言单词列表。图6.5描述了一个英语句子生成汉语句子的过程。

- 首先，对于每个英语单词  $t_i$  确定它的产出率  $\varphi_i$ 。比如“Scientists”的产出率是 2，可表示为  $\varphi_1 = 2$ 。这表明它会生成 2 个汉语单词；
- 其次，确定英语句子中每个单词生成的汉语单词列表。比如“Scientists”生成“科学家”和“们”两个汉语单词，可表示为  $\tau_1 = \{\tau_{11} = “科学家”, \tau_{12} = “们”\}$ 。这里用特殊的空标记 NULL 表示翻译对空的情况；
- 最后，把生成的所有汉语单词放在合适的位置。比如“科学家”和“们”分别放在  $s$  的位置 1 和位置 2。可以用符号  $\pi$  记录生成的单词在源语言句子  $s$  中的位置。

比如“Scientists”生成的汉语单词在  $s$  中的位置表示为  $\pi_1 = \{\pi_{11} = 1, \pi_{12} = 2\}$ 。

为了表述清晰，这里重新说明每个符号的含义。 $s$ 、 $t$ 、 $m$  和  $l$  分别表示源语言句子、目标语言译文、源语言单词数量以及译文单词数量。 $\varphi$ 、 $\tau$  和  $\pi$  分别表示产出率、生成的源语言单词以及它们在源语言句子中的位置。 $\varphi_i$  表示第  $i$  个目标语言单词  $t_i$  的产出率。 $\tau_i$  和  $\pi_i$  分别表示  $t_i$  生成的源语言单词列表及其在源语言句子  $s$  中的位置列表。

可以看出，一组  $\tau$  和  $\pi$ （记为  $\langle \tau, \pi \rangle$ ）可以决定一个对齐  $a$  和一个源语句子  $s$ 。相反的，一个对齐  $a$  和一个源语句子  $s$  可以对应多组  $\langle \tau, \pi \rangle$ 。如图6.6所示，不同的  $\langle \tau, \pi \rangle$  对应同一个源语言句子和词对齐。它们的区别在于目标语单词“Scientists”生成的源语言单词“科学家”和“们”的顺序不同。这里把不同的  $\langle \tau, \pi \rangle$  对应到的相同的源语句子  $s$  和对齐  $a$  记为  $\langle s, a \rangle$ 。因此计算  $P(s, a | t)$  时需要把每个可能结果的概率加起来，如下：

$$P(s, a | t) = \sum_{\langle \tau, \pi \rangle \in \langle s, a \rangle} P(\tau, \pi | t) \quad (6.9)$$

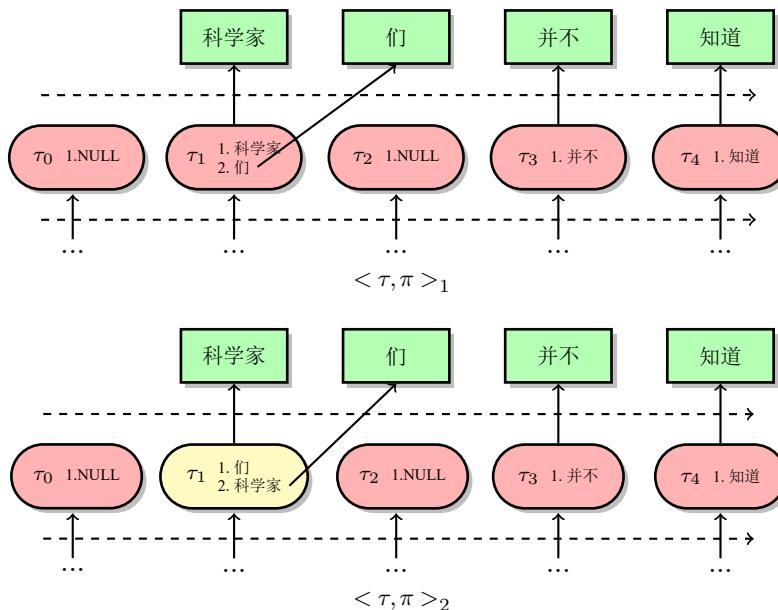


图 6.6 不同  $\tau$  和  $\pi$  对应相同的源语言句子和词对齐的情况

不过  $\langle s, a \rangle$  中有多少组  $\langle \tau, \pi \rangle$  呢？通过图6.5中的例子，可以推出  $\langle s, a \rangle$  应该包含  $\prod_{i=0}^l \varphi_i!$  个不同的二元组  $\langle \tau, \pi \rangle$ 。这是因为在给定源语言句子和词对齐时，对于每一个  $\tau_i$  都有  $\varphi_i!$  种排列。

进一步， $P(\tau, \pi | t)$  可以被表示如图6.7的形式。其中  $\tau_{i1}^{k-1}$  表示  $\tau_{i1} \cdots \tau_{i(k-1)}$ ， $\pi_{i1}^{k-1}$  表示  $\pi_{i1} \cdots \pi_{i(k-1)}$ 。可以把图6.7中的公式分为 5 个部分，并用不同的序号和颜色进行标注。每部分的具体含义是：

$$\begin{aligned}
P(\tau, \pi | t) = & \prod_{i=1}^l P(\varphi_i | \varphi_1^{i-1}, t) \times P(\varphi_0 | \varphi_1^l, t) \times \\
& \prod_{i=0}^l \prod_{k=1}^{\varphi_i} P(\tau_{ik} | \tau_{i1}^{k-1}, \tau_1^{i-1}, \varphi_0^l, t) \times \\
& \prod_{i=1}^l \prod_{k=1}^{\varphi_i} P(\pi_{ik} | \pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^l, \varphi_0^l, t) \times \\
& \prod_{k=1}^{\varphi_0} P(\pi_{0k} | \pi_{01}^{k-1}, \pi_1^l, \tau_0^l, \varphi_0^l, t)
\end{aligned}$$

图 6.7  $P(\tau, \pi | t)$  的详细表达式

- 第一部分：对每个  $i \in [1, l]$  的目标语单词的产出率建模（红色），即  $\varphi_i$  的生成概率。它依赖于  $t$  和区间  $[1, i-1]$  的目标语单词的产出率  $\varphi_1^{i-1}$ 。<sup>1</sup>
- 第二部分：对  $i=0$  时的产出率建模（蓝色），即空标记  $t_0$  的产出率生成概率。它依赖于  $t$  和区间  $[1, i-1]$  的目标语单词的产出率  $\varphi_1^l$ 。
- 第三部分：对词汇翻译建模（绿色），目标语言单词  $t_i$  生成第  $k$  个源语言单词  $\tau_{ik}$  时的概率，依赖于  $t$ 、所有目标语言单词的产出率  $\varphi_0^l$ 、区间  $i \in [1, l]$  的目标语言单词生成的源语言单词  $\tau_1^{i-1}$  和目标语单词  $t_i$  生成的前  $k$  个源语言单词  $\tau_{i1}^{k-1}$ 。
- 第四部分：对于每个  $i \in [1, l]$  的目标语言单词生成的源语言单词的扭曲度建模（黄色），即第  $i$  个目标语言单词生成的第  $k$  个源语言单词在源文中的位置  $\pi_{ik}$  的概率。其中  $\pi_1^{i-1}$  表示区间  $[1, i-1]$  的目标语言单词生成的源语言单词的扭曲度， $\pi_{i1}^{k-1}$  表示第  $i$  目标语言单词生成的前  $k-1$  个源语言单词的扭曲度。
- 第五部分：对  $i=0$  时的扭曲度建模（灰色），即空标记  $t_0$  生成源语言位置的概率。

### 6.2.2 IBM 模型 3

IBM 模型 3 通过一些假设对图 6.7 所表示的基本模型进行了化简。具体来说，对于每个  $i \in [1, l]$ ，假设  $P(\varphi_i | \varphi_1^{i-1}, t)$  仅依赖于  $\varphi_i$  和  $t_i$ ， $P(\pi_{ik} | \pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^l, \varphi_0^l, t)$  仅依赖于  $\pi_{ik}$ 、 $i$ 、 $m$  和  $l$ 。而对于所有的  $i \in [0, l]$ ，假设  $P(\tau_{ik} | \tau_{i1}^{k-1}, \tau_1^{i-1}, \varphi_0^l, t)$  仅依赖于  $\tau_{ik}$  和  $t_i$ 。这些假设的形式化描述为：

$$P(\varphi_i | \varphi_1^{i-1}, t) = P(\varphi_i | t_i) \quad (6.10)$$

$$P(\tau_{ik} = s_j | \tau_{i1}^{k-1}, \tau_1^{i-1}, \varphi_0^l, t) = t(s_j | t_i) \quad (6.11)$$

$$P(\pi_{ik} = j | \pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^l, \varphi_0^l, t) = d(j | i, m, l) \quad (6.12)$$

通常把  $d(j | i, m, l)$  称为扭曲度函数。这里  $P(\varphi_i | \varphi_1^{i-1}, t) = P(\varphi_i | t_i)$  和  $P(\pi_{ik} = j | \pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^l, \varphi_0^l, t) = t(s_j | t_i)$ 。

<sup>1</sup> 这里约定，当  $i=1$  时， $\varphi_1^0$  表示空。

$\pi_1^{i-1}, \tau_0^l, \varphi_0^l, t) = d(j|i, m, l)$  仅对  $1 \leq i \leq l$  成立。这样就完成了图6.7中第1、3和4部分的建模。

对于  $i=0$  的情况需要单独进行考虑。实际上,  $t_0$  只是一个虚拟的单词。它要对应  $s$  中原本为空对齐的单词。这里假设: 要等其他非空对应单词都被生成(放置)后, 才考虑这些空对齐单词的生成(放置)。即非空对单词都被生成后, 在那些还有空的位置上放置这些空对的源语言单词。此外, 在任何的空位置上放置空对的源语言单词都是等概率的, 即放置空对齐源语言单词服从均匀分布。这样在已经放置了  $k$  个空对齐源语言单词的时候, 应该还有  $\varphi_0 - k$  个空位置。如果第  $j$  个源语言位置为空, 那么

$$P(\pi_{0k} = j | \pi_{01}^{k-1}, \pi_1^l, \tau_0^l, \varphi_0^l, t) = \frac{1}{\varphi_0 - k} \quad (6.13)$$

否则

$$P(\pi_{0k} = j | \pi_{01}^{k-1}, \pi_1^l, \tau_0^l, \varphi_0^l, t) = 0 \quad (6.14)$$

这样对于  $t_0$  所对应的  $\tau_0$ , 就有

$$\prod_{k=1}^{\varphi_0} P(\pi_{0k} | \pi_{01}^{k-1}, \pi_1^l, \tau_0^l, \varphi_0^l, t) = \frac{1}{\varphi_0!} \quad (6.15)$$

而上面提到的  $t_0$  所对应的这些空位置是如何生成的呢? 即如何确定哪些位置是要放置空对齐的源语言单词。在 IBM 模型 3 中, 假设在所有的非空对齐源语言单词都被生成出来后(共  $\varphi_1 + \dots + \varphi_l$  个非空对源语单词), 这些单词后面都以  $p_1$  概率随机地产生一个“槽”用来放置空对齐单词。这样,  $\varphi_0$  就服从了一个二项分布。于是得到

$$P(\varphi_0 | t) = \binom{\varphi_1 + \dots + \varphi_l}{\varphi_0} p_0^{\varphi_1 + \dots + \varphi_l - \varphi_0} p_1^{\varphi_0} \quad (6.16)$$

其中,  $p_0 + p_1 = 1$ 。到此为止, 已经完成了图6.7中第2和5部分的建模。最终根据这些假设可以得到  $P(s|t)$  的形式为:

$$\begin{aligned} P(s|t) &= \sum_{a_1=0}^l \cdots \sum_{a_m=0}^l \left[ \binom{m - \varphi_0}{\varphi_0} p_0^{m-2\varphi_0} p_1^{\varphi_0} \prod_{i=1}^l \varphi_i! n(\varphi_i | t_i) \right. \\ &\quad \times \prod_{j=1}^m t(s_j | t_{a_j}) \times \left. \prod_{j=1, a_j \neq 0}^m d(j | a_j, m, l) \right] \end{aligned} \quad (6.17)$$

其中,  $n(\varphi_i|t_i) = P(\varphi_i|t_i)$  表示产出率的分布。这里的约束条件为:

$$\sum_{s_x} t(s_x|t_y) = 1 \quad (6.18)$$

$$\sum_j d(j|i, m, l) = 1 \quad (6.19)$$

$$\sum_{\varphi} n(\varphi|t_y) = 1 \quad (6.20)$$

$$p_0 + p_1 = 1 \quad (6.21)$$

### 6.2.3 IBM 模型 4

IBM 模型 3 仍然存在问题, 比如, 它不能很好地处理一个目标语言单词生成多个源语言单词的情况。这个问题在模型 1 和模型 2 中也存在。如果一个目标语言单词对应多个源语言单词, 则这些源语言单词往往构成短语。但是模型 1-3 把这些源语言单词看成独立的单元, 而实际上它们是一个整体。这就造成了在模型 1-3 中这些源语言单词可能会“分散”开。为了解决这个问题, 模型 4 对模型 3 进行了进一步修正。

为了更清楚地阐述, 这里引入新的术语——**概念单元**或**概念**(Concept)。词对齐可以被看作概念之间的对应。这里的概念是指具有独立语法或语义功能的一组单词。依照 Brown 等人的表示方法<sup>[10]</sup>, 可以把概念记为 cept.. 每个句子都可以被表示成一系列的 cept.. 这里要注意的是, 源语言句子中的 cept. 数量不一定等于目标句子中的 cept. 数量。因为有些 cept. 可以为空, 因此可以把那些空对的单词看作空 cept.. 比如, 在图6.8的实例中, “了”就对应一个空 cept..

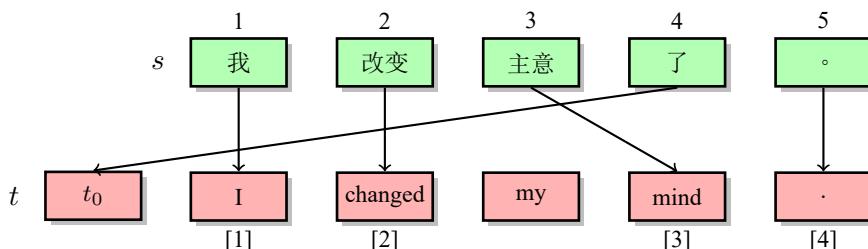


图 6.8 词对齐的汉译英句对及独立单词 cept. 的位置 (记为 [i])

在 IBM 模型的词对齐框架下, 目标语的 cept. 只能是那些非空对齐的目标语单词, 而且每个 cept. 只能由一个目标语单词组成 (通常把这类由一个单词组成的 cept. 称为独立单词 cept.). 这里用  $[i]$  表示第  $i$  个独立单词 cept. 在目标语句子中的位置。换句话说,  $[i]$  表示第  $i$  个非空对的目标语单词的位置。比如在本例中“mind”在 *t* 中的位置表示为 [3]。

另外，可以用  $\odot_i$  表示位置为  $[i]$  的目标语言单词对应的那些源语言单词位置的平均值，如果这个平均值不是整数则对它向上取整。比如在本例中，目标语句中第 4 个 cept. (“.”) 对应在源语言句子中的第 5 个单词。可表示为  $\odot_4 = 5$ 。

利用这些新引进的概念，模型 4 对模型 3 的扭曲度进行了修改。主要是把扭曲度分解为两类参数。对于  $[i]$  对应的源语言单词列表  $(\tau_{[i]})$  中的第一个单词  $(\tau_{[i]1})$ ，且  $[i] > 0$ ，它的扭曲度用如下公式计算：

$$P(\pi_{[i]1} = j | \pi_1^{[i]-1}, \tau_0^l, \varphi_0^l, t) = d_1(j - \odot_{i-1} | A(t_{[i-1]}), B(s_j)) \quad (6.22)$$

其中，第  $i$  个目标语言单词生成的第  $k$  个源语言单词的位置用变量  $\pi_{ik}$  表示。而对于列表  $(\tau_{[i]})$  中的其他的单词  $(\tau_{[i]k}, 1 < k \leq \varphi_{[i]})$  的扭曲度，且  $[i] > 0$ ，用如下公式计算：

$$P(\pi_{[i]k} = j | \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \varphi_0^l, t) = d_{>1}(j - \pi_{[i]k-1} | B(s_j)) \quad (6.23)$$

这里的函数  $A(\cdot)$  和函数  $B(\cdot)$  分别把目标语言和源语言的单词映射到单词的词类。这么做的目的是要减小参数空间的大小。词类信息通常可以通过外部工具得到，比如 Brown 聚类等。另一种简单的方法是把单词直接映射为它的词性。这样可以直接用现在已经非常成熟的词性标注工具解决问题。

从上面改进的扭曲度模型可以看出，对于  $t_{[i]}$  生成的第一个源语言单词，要考虑中心  $\odot_{[i]}$  和这个源语言单词之间的绝对距离。实际上也就要把  $t_{[i]}$  生成的所有源语言单词看成一个整体并把它放置在合适的位置。这个过程要依据第一个源语言单词的词类和对应的源语中心位置，以及前一个非空的目标语言单词  $t_{[i-1]}$  的词类。而对于  $t_{[i]}$  生成的其他源语言单词，只需要考虑它与前一个刚放置完的源语言单词的相对位置和这个源语言单词的词类。

实际上，上述过程要先用  $t_{[i]}$  生成的第一个源语言单词代表整个  $t_{[i]}$  生成的单词列表，并把第一个源语言单词放置在合适的位置。然后，相对于前一个刚生成的源语言单词，把列表中的其他单词放置在合适的地方。这样就可以在一定程度上保证由同一个目标语言单词生成的源语言单词之间可以相互影响，达到了改进的目的。

#### 6.2.4 IBM 模型 5

模型 3 和模型 4 并不是“准确”的模型。这两个模型会把一部分概率分配给一些根本就不存在的句子。这个问题被称作 IBM 模型 3 和模型 4 的缺陷（Deficiency）。说得具体一些，模型 3 和模型 4 中并没有这样的约束：如果已经放置了某个源语言单词的位置不能再放置其他单词，也就是说句子的任何位置只能放置一个词，不能多也不能少。由于缺乏这个约束，模型 3 和模型 4 中在所有合法的词对齐上概率和不等于 1。这部分缺失的概率被分配到其他不合法的词对齐上。举例来说，如图 6.9 所示，“吃/早饭”和“have breakfast”之间的合法词对齐用直线表示。但是在模型 3 和

模型 4 中，它们的概率和为  $0.9 < 1$ 。损失掉的概率被分配到像 a5 和 a6 这样的对齐上了（红色）。虽然 IBM 模型并不支持一对多的对齐，但是模型 3 和模型 4 把概率分配给这些“不合法”的词对齐上，因此也就产生所谓的缺陷。

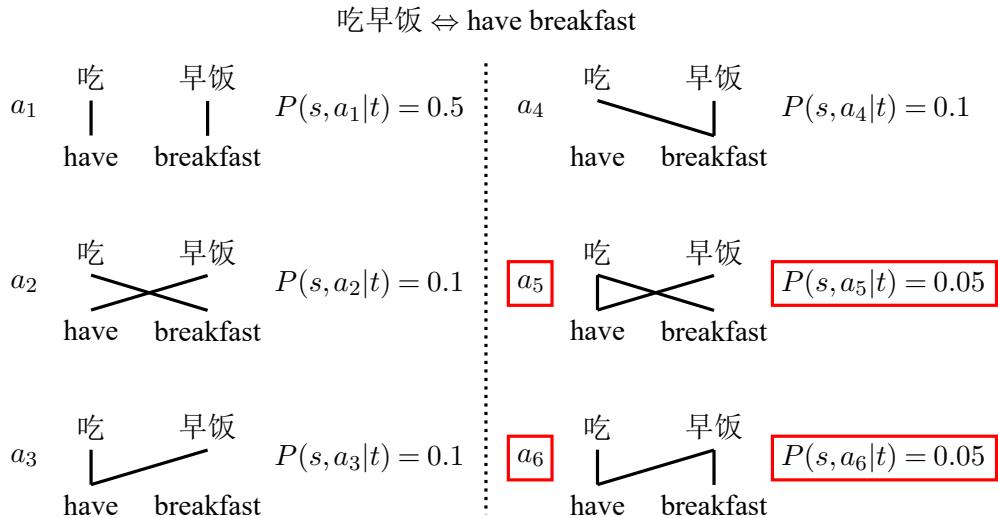


图 6.9 IBM 模型 3 的词对齐及概率分配

为了解决这个问题，模型 5 在模型中增加了额外的约束。基本想法是，在放置一个源语言单词的时候检查这个位置是否已经放置了单词，如果可以则把这个放置过程赋予一定的概率，否则把它作为不可能事件。基于这个想法，就需要在逐个放置源语言单词的时候判断源语言句子的哪些位置为空。这里引入一个变量  $v(j, \tau_1^{[i]-1}, \tau_{[i]1}^{k-1})$ ，它表示在放置  $\tau_{[i]k}$  之前 ( $\tau_1^{[i]-1}$  和  $\tau_{[i]1}^{k-1}$  已经被放置完了)，从源语言句子的第一个位置到位置  $j$  (包含  $j$ ) 为止还有多少个空位置。这里，把这个变量简写为  $v_j$ 。于是，对于  $[i]$  所对应的源语言单词列表  $(\tau_{[i]})$  中的第一个单词  $(\tau_{[i]1})$ ，有：

$$P(\pi_{[i]1} = j | \pi_1^{[i]-1}, \tau_0^l, \varphi_0^l, t) = d_1(v_j | B(s_j), v_{\odot_{i-1}}, v_m - (\varphi_{[i]} - 1)) \cdot (1 - \delta(v_j, v_{j-1})) \quad (6.24)$$

对于其他单词  $(\tau_{[i]k}, 1 < k \leq \varphi_{[i]} )$ , 有:

$$P(\pi_{[i]k} = j | \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \varphi_0^l, t) \\ = d_{>1}(v_j - v_{\pi_{[i]k-1}} | B(s_j), v_m - v_{\pi_{[i]k-1}} - \varphi_{[i]} + k) \cdot (1 - \delta(v_j, v_{j-1})) \quad (6.25)$$

这里，因子  $1 - \delta(v_j, v_{j-1})$  是用来判断第  $j$  个位置是不是为空。如果第  $j$  个位置为空则  $v_j = v_{j-1}$ ，这样  $P(\pi_{[i]1} = j | \pi_1^{[i]-1}, \tau_0^l, \varphi_0^l, t) = 0$ 。这样就从模型上避免了模型 3 和模型 4 中生成不存在的字符串的问题。这里还要注意的是，对于放置第一个单词的情况，影响放置的因素有  $v_i$ ,  $B(s_i)$  和  $v_{i-1}$ 。此外还要考虑位置  $j$  放置了第一个源

语言单词以后它的右边是不是还有足够的位置留给剩下的  $k - 1$  个源语言单词。参数  $v_m - (\varphi_{[i]} - 1)$  正是为了解决这个问题，这里  $v_m$  表示整个源语言句子中还有多少空位置， $\varphi_{[i]} - 1$  表示源语言位置  $j$  右边至少还要留出的空格数。对于放置非第一个单词的情况，主要是要考虑它和前一个放置位置的相对位置。这主要体现在参数  $v_j - v_{\varphi_{[i]} k - 1}$  上。式(6.25)的其他部分都可以用上面的理论解释，这里不再赘述。

实际上，模型 5 和模型 4 的思想基本一致，即，先确定  $\tau_{[i]1}$  的绝对位置，然后再确定  $\tau_{[i]}$  中剩余单词的相对位置。**模型 5 消除了产生不存在的句子的可能性**，不过模型 5 的复杂性也大大增加了。

## 6.3 解码和训练

与 IBM 模型 1一样，IBM 模型 2-5 和隐马尔可夫模型的解码可以直接使用第五章所描述的方法。基本思路与第二章所描述的自左向右搜索方法一致，即：对译文自左向右生成，每次扩展一个源语言单词的翻译，即把源语言单词的译文放到已经生成的译文的右侧。每次扩展可以选择不同的源语言单词或者同一个源语言单词的不同翻译候选，这样就可以得到多个不同的扩展译文。在这个过程中，同时计算翻译模型和语言模型的得分，对每个得到的译文候选打分。最终，保留一个或者多个译文。这个过程重复执行直至所有源语言单词被翻译完。

类似的，IBM 模型 2-5 和隐马尔可夫模型也都可以使用期望最大化（EM）方法进行模型训练。相关数学推导可参考附录B的内容。通常，可以使用这些模型获得双语句子间的词对齐结果，比如使用 GIZA++ 工具。这时，往往会使用多个模型，**把简单的模型训练后的参数作为初始值传给后面更加复杂的模型**。比如，先用 IBM 模型 1 训练，之后把参数送给 IBM 模型 2，再训练，之后把参数送给隐马尔可夫模型等。值得注意的是，并不是所有的模型使用 EM 算法都能找到全局最优解。特别是 IBM 模型 3-5 的训练中使用一些剪枝和近似的方法，优化的真实目标函数会更加复杂。不过，**IBM 模型 1 是一个凸函数**（Convex Function），因此理论上使用 **EM 方法能够找到全局最优解**。更实际的好处是，IBM 模型 1 训练的最终结果与参数的初始化过程无关。这也是为什么在使用 IBM 系列模型时，往往会**使用 IBM 模型 1 作为起始模型的原因**。

## 6.4 问题分析

IBM 模型是一个时代的经典，但也留下了一些值得思考的问题。这一方面体现了科学技术发展需要一步步前行，而非简单的一蹴而就。另一方面也体现了机器翻译问题的困难程度。下面对 IBM 存在的问题进行分析，同时给出一些解决问题的思路，希望通过这些讨论可以使我们对机器翻译问题有更深层次的理解。

### 6.4.1 词对齐及对称化

IBM 五个模型都是基于一个词对齐的假设——一个源语言单词最多只能对齐到一个目标语言单词。这个约束大大降低了建模的难度。在法英翻译中一对多的对齐情况并不多见，这个假设带来的问题也不是那么严重。但是，在像汉英翻译这样的任务中，一个汉语单词对应多个英语单词的翻译很常见，这时 IBM 模型的词对齐假设就表现出了明显的问题。比如在翻译“我/会/试一试。”→“I will have a try.”时，IBM 模型根本不能把单词“试一试”对齐到三个单词“have a try”，因而可能无法得到正确的翻译结果。

本质上，IBM 模型词对齐的“不完整”问题是 IBM 模型本身的缺陷。解决这个问题有很多思路。一种思路是，反向训练后，合并源语言单词，然后再正向训练。这里用汉英翻译为例来解释这个方法。首先反向训练，就是把英语当作待翻译语言，而把汉语当作目标语言进行训练（参数估计）。这样可以得到一个词对齐结果（参数估计的中间结果）。在这个词对齐结果里面，一个汉语单词可对应多个英语单词。之后，扫描每个英语句子，如果有多个英语单词对应同一个汉语单词，就把这些英语单词合并成一个英语单词。处理完之后，再把汉语当作源语言而把英语当作目标语言进行训练。这样就可以把一个汉语单词对应到合并的英语单词上。虽然从模型上看，还是一个汉语单词对应一个英语“单词”，但实质上已经把这个汉语单词对应到多个英语单词上了。训练完之后，再利用这些参数进行翻译（解码）时，就能把一个中文单词翻译成多个英文单词了。但是反向训练后再训练也存在一些问题。首先，合并英语单词会使数据变得更稀疏，训练不充分。其次，由于 IBM 模型的词对齐结果并不是高精度的，利用它的词对齐结果来合并一些英文单词可能造成严重的错误，比如：把本来很独立的几个单词合在了一起。因此，还要考虑实际需要和问题的严重程度来决定是否使用该方法。

另一种思路是双向对齐之后进行词对齐对称化（Symmetrization）。这个方法可以在 IBM 词对齐的基础上获得对称的词对齐结果。思路很简单，用正向（汉语为源语言，英语为目标语言）和反向（汉语为目标语言，英语为源语言）同时训练。这样可以得到两个词对齐结果。然后利用一些启发性方法用这两个词对齐生成对称的结果（比如，取“并集”、“交集”等），这样就可以得到包含一对多和多对多的词对齐结果<sup>[240]</sup>。比如，在基于短语的统计机器翻译中已经很成功地使用了这种词对齐信息进行短语的获取。直到今天，对称化仍然是很多自然语言处理系统中的一个关键步骤。

### 6.4.2 “缺陷”问题

IBM 模型的缺陷是指翻译模型会把一部分概率分配给一些根本不存在的源语言字符串。如果用  $P(\text{well}|t)$  表示  $P(s|t)$  在所有的正确的（可以理解为语法上正确的）

$s$  上的和，即

$$P(\text{well}|t) = \sum_{s \text{ is well formed}} P(s|t) \quad (6.26)$$

类似地，用  $P(\text{ill}|t)$  表示  $P(s|t)$  在所有的错误的（可以理解为语法上错误的） $s$  上的和。如果  $P(\text{well}|t) + P(\text{ill}|t) < 1$ ，就把剩余的部分定义为  $P(\text{failure}|t)$ 。它的形式化定义为，

$$P(\text{failure}|t) = 1 - P(\text{well}|t) - P(\text{ill}|t) \quad (6.27)$$

本质上，模型 3 和模型 4 就是对应  $P(\text{failure}|t) > 0$  的情况。这部分概率是模型损失掉的。有时候也把这类缺陷称为**物理缺陷** (Physical Deficiency) 或**技术缺陷** (Technical Deficiency)。还有一种缺陷被称作**精神缺陷** (Spiritual Deficiency) 或**逻辑缺陷** (Logical Deficiency)，它是指  $P(\text{well}|t) + P(\text{ill}|t) = 1$  且  $P(\text{ill}|t) > 0$  的情况。模型 1 和模型 2 就有逻辑缺陷。可以注意到，**技术缺陷只存在于模型 3 和模型 4 中**，模型 1 和模型 2 并没有技术缺陷问题。根本原因在于模型 1 和模型 2 的词对齐是从源语言出发对应到目标语言， $t$  到  $s$  的翻译过程实际上是从单词  $s_1$  开始到单词  $s_m$  结束，依次把每个源语言单词  $s_j$  对应到唯一一个目标语言位置。显然，这个过程能够保证每个源语言单词仅对应一个目标语言单词。但是，模型 3 和模型 4 中对齐是从目标语言出发对应到源语言， $t$  到  $s$  的翻译过程从  $t_1$  开始  $t_l$  结束，依次把目标语言单词  $t_i$  生成的单词对应到某个源语言位置上。但是这个过程不能保证  $t_i$  中生成的单词所对应的位置没有被其他单词占用，因此也就产生了缺陷。

这里还要强调的是，技术缺陷是模型 3 和模型 4 是模型本身的缺陷造成的，如果有一个“更好”的模型就可以完全避免这个问题。而逻辑缺陷几乎是不能从模型上根本解决的，因为对于任意一种语言都不能枚举所有的句子 ( $P(\text{ill}|t)$  实际上是得不到的)。

IBM 的模型 5 已经解决了技术缺陷问题。但逻辑缺陷的解决很困难，因为即使对于人来说也很难判断一个句子是不是“良好”的句子。当然可以考虑用语言模型来缓解这个问题，不过由于在翻译的时候源语言句子都是定义“良好”的句子， $P(\text{ill}|t)$  对  $P(s|t)$  的影响并不大。但用输入的源语言句子  $s$  的“良好性”并不能解决技术缺陷，因为技术缺陷是模型的问题或者模型参数估计方法的问题。无论输入什么样的  $s$ ，模型 3 和模型 4 的技术缺陷问题都存在。

#### 6.4.3 句子长度

在 IBM 模型中， $P(t)P(s|t)$  会随着目标语言句子长度的增加而减少，因为这种模型有多个概率化的因素组成，乘积项越多结果的值越小。这也就是说，IBM 模型会更倾向选择**长度短一些的目标语言句子**。显然这种对短句子的偏向性并不是机器翻译所期望的。

这个问题在很多机器翻译系统中都存在。它实际上也是一种**系统偏置**（System Bias）的体现。为了消除这种偏置，可以通过在模型中增加一个短句子惩罚因子来抵消掉模型对短句子的倾向性。比如，可以定义一个惩罚因子，它的值随着长度的减少而增加。不过，简单引入这样的惩罚因子会导致模型并不符合一个严格的噪声信道模型。它对应一个基于判别式框架的翻译模型，这部分内容会在第七章进行介绍。

#### 6.4.4 其他问题

模型 5 的意义是什么？模型 5 的提出是为了消除模型 3 和模型 4 的缺陷。缺陷的本质是， $P(s, a|t)$  在所有合理的对齐上概率和不为 1。但是，在这里更关心是哪个对齐  $a$  使  $P(s, a|t)$  达到最大，即使  $P(s, a|t)$  不符合概率分布的定义，也并不影响我们寻找理想的对齐  $a$ 。从工程的角度说， $P(s, a|t)$  不归一并不是一个十分严重的问题。遗憾的是，实际上到现在为止有太多对 IBM 模型 3 和模型 4 中的缺陷进行系统性的实验和分析，但对于这个问题到底有多严重并没有定论。当然用模型 5 是可以解决这个问题。但是如果用一个非常复杂的模型去解决了一个并不产生严重后果的问题，那这个模型也就没有太大意义了（从实践的角度）。

概念（cept.）的意义是什么？经过前面的分析可知，IBM 模型的词对齐模型使用了 *cept.* 这个概念。但是，在 IBM 模型中使用的 *cept.* 最多只能对应一个目标语言单词（模型并没有用到源语言 *cept.* 的概念）。因此可以直接用单词代替 *cept.*。这样，即使不引入 *cept.* 的概念，也并不影响 IBM 模型的建模。实际上，*cept.* 的引入确实可以帮助我们从语法和语义的角度解释词对齐过程。不过，这个方法在 IBM 模型中的效果究竟如何还没有定论。

### 6.5 小结及拓展阅读

本章在 IBM 模型 1 的基础上进一步介绍了 IBM 模型 2-5 以及 HMM。同时，本章引入了两个新的概念——扭曲度和繁衍率。它们都是机器翻译中的经典概念，也经常出现在机器翻译的建模中。另一方面，通过对上述模型的分析，本章进一步探讨建模中的若干基础问题，例如，如何把翻译问题分解为若干步骤，并建立合理的模型解释这些步骤；如何对复杂问题进行化简，以得到可以计算的模型等等。这些思想也在很多自然语言处理问题中被使用。此外，关于扭曲度和繁衍率还有一些问题值得关注：

- 扭曲度是机器翻译中的一个经典概念。广义上来说，事物位置的变换都可以用扭曲度进行描述，比如，在物理成像系统中，扭曲度模型可以帮助进行镜头校正<sup>[263, 264]</sup>。在机器翻译中，扭曲度本质上在描述源语言和目标语言单词顺序的偏差。这种偏差可以用于对调序的建模。因此扭曲度的使用也可以被看作是一种对调序问题的描述，这也是机器翻译区别于语音识别等任务的主要因素之一。在早期的统计机器翻译系统中，如 Pharaoh<sup>[81]</sup>，大量使用了扭曲度这个概念。虽

然，随着机器翻译的发展，更复杂的调序模型被提出<sup>[23, 265, 266, 267, 268, 269]</sup>，但是扭曲度所引发的对调序问题的思考是非常深刻的，这也是 IBM 模型最大的贡献之一。

- IBM 模型的另一个贡献是在机器翻译中引入了繁衍率的概念。本质上，繁衍率是一种对翻译长度的建模。在 IBM 模型中，通过计算单词的繁衍率就可以得到整个句子的长度。需要注意的是，在机器翻译中译文长度对翻译性能有着至关重要的影响。虽然，在很多机器翻译模型中并没有直接使用繁衍率这个概念，但是几乎所有的现代机器翻译系统中都有译文长度的控制模块。比如，在统计机器翻译和神经机器翻译中，都把译文单词数量作为一个特征用于生成合理长度的译文<sup>[22, 80, 270]</sup>。此外，在神经机器翻译中，非自回归的解码中也使用繁衍率模型对译文长度进行预测<sup>[271]</sup>。





## 7. 基于短语的模型

机器翻译的一个基本问题是定义翻译的基本单元是什么。比如，可以像第五章介绍的那样，以单词为单位进行翻译，即把句子的翻译看作是单词之间对应关系的一种组合。基于单词的模型是符合人类对翻译问题的认知的，因为单词本身就是人类加工语言的一种基本单元。然而，在进行翻译时也可以使用一些更“复杂”的知识。比如，很多词语间的搭配需要根据语境的变化进行调整，而且对于句子结构的翻译往往需要更上层的知识，如句法知识。因此，在对单词翻译进行建模的基础上，需要探索其他类型的翻译知识，使得搭配和结构翻译等问题可以更好地被建模。

本章会介绍基于短语的机器翻译模型。在过去二十年中，它一直是机器翻译的主流方法。相比于基于单词的模型，基于短语的模型可以更好地对单词之间搭配和小范围依赖关系进行描述。这种方法也在相当长的一段时期内占据着机器翻译的统治地位。即使近些年神经机器翻译逐渐崛起，基于短语的模型仍然是机器翻译的主要框架之一，其中的思想和很多技术手段对今天的机器翻译研究仍然有很好的借鉴意义。

### 7.1 翻译中的短语信息

不难发现，基于单词的模型并不能很好地捕捉单词间的搭配关系。相比之下，使用更大颗粒度的翻译单元是一种对搭配进行处理的方法。下面来一起看看，基于单词的模型所产生的问题以及如何使用基于短语的模型来缓解该问题。

### 7.1.1 词的翻译带来的问题

首先，回顾一下基于单词的统计翻译模型是如何完成翻译的。图7.1展示了一个实例。其中，左侧是一个单词的“翻译表”，它记录了源语言（汉语）单词和目标语言（英语）单词之间的对应关系，以及这种对应的可能大小（用  $P$  表示）。在翻译时，会使用这些单词一级的对应，生成译文。图7.1右侧就展示了一个基于词的模型生成的翻译结果，其中  $s$  和  $t$  分别表示源语言和目标语言句子，单词之间的连线表示两个句子中单词一级的对应。

单词翻译表	$P$	$s =$	我	喜欢	绿	茶
我 → I	0.6			~	/	
喜欢 → like	0.3			~	/	
绿 → green	0.9	$t =$	I	like	green	tea
茶 → tea	0.8					

图 7.1 基于单词的翻译实例

图7.1体现的是一个典型的基于单词对应关系的翻译方法。它非常适合**组合性翻译** (Compositional Translation) 的情况，也就是通常说的直译。不过，自然语言作为人类创造的高级智能的载体，远比想象的复杂。比如，即使是同一个单词，词义也会根据不同的语境产生变化。

图7.2给出了一个新的例子，其中为了便于阅读，单词之间用空格或者斜杠进行分割。如果同样使用概率化的单词翻译对问题进行建模，对于输入的句子“我/喜欢/红/茶”，翻译概率最大的译文是“*I like red tea*”。显然，“red tea”并不是英语中“红/茶”的说法，正确的译文应该是“*black tea*”。

单词翻译表	$P$	$s =$	我	喜欢	红	茶
我 → I	0.6			~	/	
喜欢 → like	0.3			~	/	
红 → red	0.8	$t =$	I	like	red	tea
红 → black	0.1					
茶 → tea	0.8					

“红茶”为一种搭配，应该翻译为“black tea”

图 7.2 基于单词的模型对固定搭配“红/茶”进行翻译

这里的问题在于，“*black tea*”不能通过“红”和“茶”这两个单词直译的结果组合而成，也就是，把“红”翻译为“red”并不符合“红/茶”这种特殊搭配的翻译。虽然在训练数据中“红”有很高的概率被翻译为“red”，但是在这个例子中，应该选择概率更低的译文“black”。那如何做到这一点呢？如果让人来做，这个事不难，因为所有人学习英语的时候都知道“红”和“茶”放在一起构成了一个短语，或者说一种搭配，这种搭配的译文是固定的，记住就好。同理，如果机器翻译系统也能学习并记住这样的搭配，显然可以做得更好。这也就形成了基于短语的机器翻译建模的基本思想。

本思路。

### 7.1.2 更大粒度的翻译单元

既然仅仅使用单词的直译不能覆盖所有的翻译现象，那就可以考虑在翻译中使用更大颗粒度的单元，这样能够对更大范围的搭配和依赖关系进行建模。一种非常简单的方法是把单词扩展为  $n$ -gram，这里视为**短语**（Phrase）。也就是，翻译的基本单元是一个个连续的词串，而非一个个相互独立的单词。

图7.3展示了一个引入短语之后的翻译结果。其中的翻译表不仅包含源语言和目标语言单词之间的对应，同时也包括短语（ $n$ -gram）的翻译。这样，“红/茶”可以作为一个短语包含在翻译表中，它所对应译文是“black tea”。对于待翻译句子，可以使用单词翻译的组合得到“红/茶”的译文“red tea”，也可以直接使用短语翻译得到“black tea”。由于短语翻译“红/茶 → black tea”的概率更高，因此最终会输出正确的译文“black tea”。

词串翻译表	$P$	
我 → I	0.6	
喜欢 → like	0.3	$s =$ 我 喜欢 红 茶
红 → red	0.8	
红 → black	0.1	
茶 → tea	0.8	
我/喜欢 → I like	0.3	$t =$ I like red tea
我/喜欢 → I liked	0.2	$s =$ 我 喜欢 红 茶
绿/茶 → green tea	0.5	
绿/茶 → the green tea	0.1	
红/茶 → black tea	0.7	$t =$ I like black tea
...		

图 7.3 基于短语（ $n$ -gram）的翻译的实例

一般来说，统计机器翻译的建模对应着一个两阶段的过程：首先，得到每个翻译单元所有可能的译文；然后，通过对这些译文的组合得到可能的句子翻译结果，并选择最佳的目标语言句子输出。如果基本的翻译单元被定义下来，机器翻译系统可以学习这些单元翻译所对应的翻译知识（对应训练过程），之后运用这些知识对新的句子进行翻译（对应解码过程）。

图7.4给出了基于单词的机器翻译过程的一个示例。首先，每个单词的候选译文都被列举出来，而机器翻译系统就是要找到覆盖所有源语言单词的一条路径，且对应的译文概率是最高的。比如，图中的红色折线就代表了一条翻译路径，也就是一个单词译文的序列<sup>1</sup>。

在引入短语翻译之后，并不需要对上述过程进行太大的修改。仍然可以把翻译

<sup>1</sup>为了简化问题，这里没有描述单词译文的调序。对于调序的建模，可以把它当作是对目标语单词串的排列，这个排列的好坏需要用额外的调序模型进行描述。详细内容见7.4节。

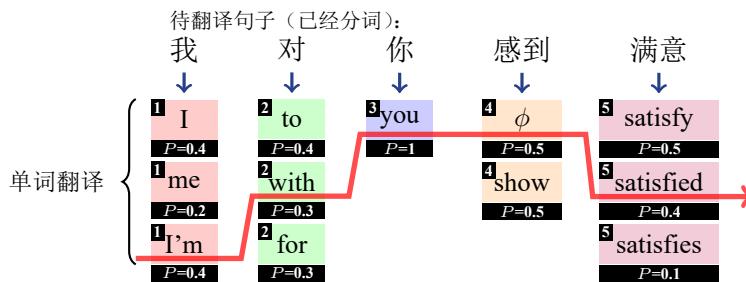


图 7.4 基于单词的翻译被看作是一条“路径”

当作是一条贯穿源语言所有单词译文的路径，只是这条路径中会包含短语，而非一个个单词。图7.5给出了一个实例，其中的蓝色折线表示包含短语的翻译路径。

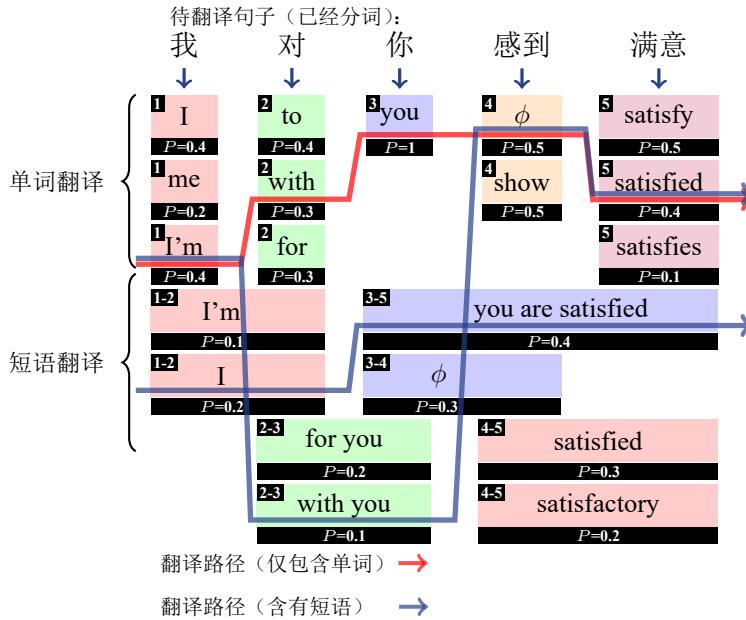


图 7.5 翻译被看作是由单词和短语组成的“路径”

实际上，单词本身也是一种短语。从这个角度说，基于单词的翻译模型是包含在基于短语的翻译模型中的。而这里所说的短语包括多个连续的单词，可以直接捕捉翻译中的一些局部依赖。而且，由于引入了更多样的翻译单元，可选择的翻译路径数量也大大增加。本质上，引入更大颗粒度的翻译单元给模型增加了灵活性，同时增大了翻译假设空间。如果建模合理，更多的翻译路径会增加找到高质量译文的机会。在7.2节还将看到，基于短语的模型会从多个角度对翻译问题进行描述，包括基础数学建模、调序等等。

### 7.1.3 机器翻译中的短语

基于短语的机器翻译的基本假设是：双语句子的生成可以用短语之间的对应关系进行表示。图7.6展示了一个基于短语的翻译实例。可以看到，这里的翻译单元是连续的词串。比如，“进口”的译文“The imports have”就包含了三个单词，而“下降/了”也是一个包含两个单词的源语言片段。

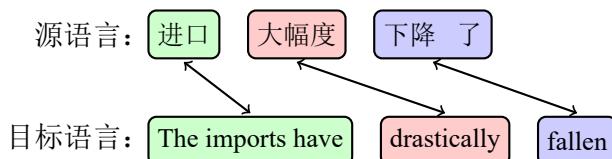


图 7.6 基于短语的汉英翻译实例

不过，这里所说的短语并不是语言学上的短语，本身也没有任何语言学句法的结构约束。在基于短语的模型中，可以把短语简单地理解为一个词串。具体来说，有如下定义。

#### 定义 7.1.1 短语

对于一个句子  $w = \{w_1 \dots w_n\}$ ，任意子串  $\{w_i \dots w_j\} (i \leq j \text{ 且 } 0 \leq i, j \leq n)$  都是句子  $w$  的一个**短语**。

根据这个定义，对于一个由  $n$  个单词构成的句子，可以包含  $\frac{n(n-1)}{2}$  个短语（子串）。进一步，可以把每个句子看作是由一系列短语构成的序列。组成这个句子的短语序列也可以被看作是句子的一个**短语切分**（Phrasal Segmentation）。

#### 定义 7.1.2 句子的短语切分

如果一个句子  $w = \{w_1 \dots w_n\}$  可以被切分为  $m$  个子串，则称  $w$  由  $m$  个短语组成，记为  $w = \{p_1 \dots p_m\}$ ，其中  $p_i$  是  $w$  的一个短语， $\{p_1, \dots, p_m\}$  也被称作句子  $w$  的一个**短语切分**。

比如，对于一个句子，“机器/翻译/是/一/项/很有/挑战/的/任务”，一种可能的短语切分为：

$$\begin{aligned}
 p_1 &= \text{机器/翻译} \\
 p_2 &= \text{是/一/项} \\
 p_3 &= \text{很有/挑战/的} \\
 p_4 &= \text{任务}
 \end{aligned}$$

进一步，把单语短语的概念推广到双语的情况：

**定义 7.1.3 双语短语（或短语对）**

对于源语言和目标语言句对  $(s, t)$ ,  $s$  中的一个短语  $\bar{s}_i$  和  $t$  中的一个短语  $\bar{t}_j$  可以构成一个双语短语对  $(\bar{s}_i, \bar{t}_j)$ , 简称**短语对** (Phrase Pairs)  $(\bar{s}_i, \bar{t}_j)$ 。

也就是说, 源语言句子中任意的短语和目标语言句子中任意的短语都构成一个双语短语。这里用  $\leftrightarrow$  表示互译关系。对于一个双语句对“牛肉的/进口/大幅度/下降/了  $\leftrightarrow$  the import of beef has drastically fallen”, 可以得到很多双语短语, 比如:

大幅度  $\leftrightarrow$  drastically  
 大幅度/下降  $\leftrightarrow$  has drastically fallen  
 牛肉的/进口  $\leftrightarrow$  import of beef  
 进口/大幅度  $\leftrightarrow$  import has drastically  
 大幅度/下降/了  $\leftrightarrow$  drastically fallen  
 了  $\leftrightarrow$  have drastically  
 ...

接下来的问题是, 如何使用双语短语描述双语句子的生成, 即句子翻译的建模问题。在基于词的翻译模型里, 可以用词对齐来描述双语句子的对应关系。类似的, 也可以使用双语短语描述句子的翻译。这里, 借用形式文法中推导的概念。把生成双语句对的过程定义为一个基于短语的翻译推导:

**定义 7.1.4 基于短语的翻译推导**

对于源语言和目标语言句对  $(s, t)$ , 分别有短语切分  $\{\bar{s}_i\}$  和  $\{\bar{t}_j\}$ , 且  $\{\bar{s}_i\}$  和  $\{\bar{t}_j\}$  之间存在一一对应的关系。令  $\{\bar{a}_j\}$  表示  $\{\bar{t}_j\}$  中每个短语对应到源语言短语的编号, 则称短语对  $\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\}$  构成了  $s$  到  $t$  的**基于短语的翻译推导** (简称推导), 记为  $d(\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\}, s, t)$  (简记为  $d(\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\})$  或  $d$ )。

基于短语的翻译推导定义了一种从源语言短语序列到目标语言短语序列的对应, 其中源语言短语序列是源语言句子的一种切分, 同样的, 目标语言短语序列是目标语言句子的一种切分。翻译推导提供了一种描述翻译过程的手段: 对于一个源语言句子, 可以找到从它出发的翻译推导, 推导中短语的目标语部分就构成了译文。也就是, 每个源语言句子  $s$  上的一个推导  $d$  都蕴含着一个目标语句子  $t$ 。

图7.7给出了一个由三个双语短语  $\{(\bar{s}_{\bar{a}_1}, \bar{t}_1), (\bar{s}_{\bar{a}_2}, \bar{t}_2), (\bar{s}_{\bar{a}_3}, \bar{t}_3)\}$  构成的汉英互译句对, 其中短语对齐信息为  $\bar{a}_1 = 1$ ,  $\bar{a}_2 = 2$ ,  $\bar{a}_3 = 3$ 。这里, 可以把这三个短语对的组合看作是翻译推导, 形式化表示为如下公式:

$$d = (\bar{s}_{\bar{a}_1}, \bar{t}_1) \circ (\bar{s}_{\bar{a}_2}, \bar{t}_2) \circ (\bar{s}_{\bar{a}_3}, \bar{t}_3) \quad (7.1)$$

其中， $\circ$  表示短语的组合<sup>2</sup>。

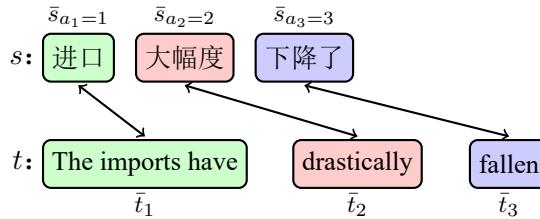


图 7.7 三个双语短语  $\{(\bar{s}_{\bar{a}_1}, \bar{t}_1), (\bar{s}_{\bar{a}_2}, \bar{t}_2), (\bar{s}_{\bar{a}_3}, \bar{t}_3)\}$  构成的翻译推导

到此为止，就得到了一个基于短语的翻译模型。对于每个双语句对  $(s, t)$ ，每个翻译推导  $d$  都对应了一个基于短语的翻译过程。而基于短语的机器翻译的目标就是对  $d$  进行描述。为了实现基于短语的翻译模型，有四个基本问题需要解决：

- 如何用统计模型描述每个翻译推导的好坏——即翻译的统计建模问题。
- 如何获得可使用的双语短语对——即短语翻译获取问题。
- 如何对翻译中的调序问题进行建模——即调序问题。
- 如何找到输入句子  $s$  的最佳译文——即解码问题。

这四个问题也构成了基于短语的翻译模型的核心，下面对其逐一展开讨论。

## 7.2 数学建模

对于统计机器翻译，其目的是找到输入句子的可能性最大的译文：

$$\hat{t} = \arg \max_t P(t|s) \quad (7.2)$$

其中， $s$  是输入的源语言句子， $t$  是一个目标语言译文。 $P(t|s)$  被称为翻译模型，它描述了把  $s$  翻译为  $t$  的可能性。通过  $\arg \max P(t|s)$  可以找到使  $P(t|s)$  达到最大的  $t$ 。

这里的第一个问题是如何定义  $P(t|s)$ 。直接描述  $P(t|s)$  是非常困难的，因为  $s$  和  $t$  分别对应了巨大的样本空间，而在训练数据中能观测到的只是空间中的一小部分样本。直接用有限的训练数据描述这两个空间中样本的对应关系会面临着严重的数据稀疏问题。对于这个问题，常用的解决办法是把复杂的问题转化为容易计算的简单问题。

<sup>2</sup> 短语的组合是指将两个短语  $a$  和  $b$  进行拼接，形成新的短语  $ab$ 。在机器翻译中，可以把双语短语的组合看作是对目标语短语的组合。比如，对于两个双语短语  $(\bar{s}_{\bar{a}_1}, \bar{t}_1), (\bar{s}_{\bar{a}_2}, \bar{t}_2)$ ，短语的组合表示将  $\bar{t}_1$  和  $\bar{t}_2$  进行组合，而源语言端作为输入已经给定，因此直接匹配源语言句子中相应的部分即可。根据两个短语在源语言中位置的不同，通常又分为顺序翻译、反序翻译、不连续翻译。关于这部分内容将会在 7.4 节调序模型的部分进行介绍。

### 7.2.1 基于翻译推导的建模

基于短语的翻译模型假设  $s$  到  $t$  的翻译可以用翻译推导进行描述，这些翻译推导都是由双语短语组成。于是，两个句子之间的映射就可以被看作是一个个短语的映射。显然短语翻译的建模要比整个句子翻译的建模简单得多。从模型上看，可以把翻译推导  $d$  当作是从  $s$  到  $t$  翻译的一种隐含结构。这种结构定义了对问题的一种描述，即翻译由一系列短语组成。根据这个假设，可以把句子的翻译概率定义为：

$$P(t|s) = \sum_d P(d, t|s) \quad (7.3)$$

公式(7.3)中， $P(d, t|s)$  表示翻译推导的概率。公式(7.3)把翻译问题转化为翻译推导的生成问题。但是，由于翻译推导的数量十分巨大<sup>3</sup>，公式(7.3)的右端需要对所有可能的推导进行枚举并求和，这几乎是无法计算的。

对于这个问题，常用的一种解决办法是利用一个化简的模型来近似完整的模型。如果把翻译推导的全体看作一个空间  $D$ ，可以从  $D$  中选取一部分样本参与计算，而不是对整个  $D$  进行计算。比如，可以用最好的  $n$  个翻译推导来代表整个空间  $D$ 。令  $D_{n\text{-best}}$  表示最好的  $n$  个翻译推导所构成的空间，于是可以定义：

$$P(t|s) \approx \sum_{d \in D_{n\text{-best}}} P(d, t|s) \quad (7.4)$$

进一步，把公式(7.4)带入公式(7.2)，可以得到翻译的目标为：

$$\hat{t} = \arg \max_t \sum_{d \in D_{n\text{-best}}} P(d, t|s) \quad (7.5)$$

另一种常用的方法是直接用  $P(d, t|s)$  的最大值代表整个翻译推导的概率和。这种方法假设翻译概率是非常尖锐的，“最好”的推导会占有概率的主要部分。它被形式化为：

$$P(t|s) \approx \max P(d, t|s) \quad (7.6)$$

于是，翻译的目标可以被重新定义：

$$\hat{t} = \arg \max_t (\max P(d, t|s)) \quad (7.7)$$

值得注意的是，翻译推导中蕴含着译文的信息，因此每个翻译推导都与一个译

<sup>3</sup> 如果把推导看作是一种树结构，推导的数量与词串的长度成指数关系。

文对应。因此可以把公式(7.7)所描述的问题重新定义为：

$$\hat{d} = \arg \max_d P(d, t|s) \quad (7.8)$$

也就是，给定一个输入句子  $s$ ，找到从它出发的最优翻译推导  $\hat{d}$ ，把这个翻译推导所对应的目标语词串看作最优的译文。假设函数  $t(\cdot)$  可以返回一个推导的目标语词串，则最优译文也可以被看作是：

$$\hat{t} = t(\hat{d}) \quad (7.9)$$

注意，公式(7.8)-(7.9)和公式(7.7)本质上是一样的。它们也构成了统计机器翻译中最常用的方法——Viterbi 方法<sup>[272]</sup>。在后面机器翻译的解码中还会看到它们的应用。而公式(7.5)也被称作  $n$ -best 方法，常常作为 Viterbi 方法的一种改进。

## 7.2.2 对数线性模型

对于如何定义  $P(d, t|s)$  有很多种思路，比如，可以把  $d$  拆解为若干步骤，然后对这些步骤分别建模，最后形成描述  $d$  的生成式模型。这种方法在第五章和第六章的 IBM 模型中也大量使用。但是，生成式模型的每一步推导需要有严格概率解释，这也限制了研究人员从更多的角度对  $d$  进行描述。这里，可以使用另外一种方法——判别式模型来对  $P(d, t|s)$  进行描述<sup>[111]</sup>。其模型形式如下：

$$P(d, t|s) = \frac{\exp(\text{score}(d, t, s))}{\sum_{d', t'} \exp(\text{score}(d', t', s))} \quad (7.10)$$

其中，

$$\text{score}(d, t, s) = \sum_{i=1}^M \lambda_i \cdot h_i(d, t, s) \quad (7.11)$$

公式(7.11)是一种典型的**对数线性模型** (Log-linear Model)。所谓“对数线性”体现在对多个量求和后进行指数运算 ( $\exp(\cdot)$ )，这相当于对多个因素进行乘法。公式(7.10)的右端是一种归一化操作。分子部分可以被看作是一种对翻译推导  $d$  的对数线性建模。具体来说，对于每个  $d$ ，用  $M$  个特征对其进行描述，每个特征用函数  $h_i(d, t, s)$  表示，它对应一个权重  $\lambda_i$ ，表示特征  $i$  的重要性。 $\sum_{i=1}^M \lambda_i \cdot h_i(d, t, s)$  表示了对这些特征的线性加权和，值越大表示模型得分越高，相应的  $d$  和  $t$  的质量越高。公式(7.10)的分母部分实际上不需要计算，因为其值与求解最佳推导的过程无关。把

公式(7.10)带入公式(7.8)得到：

$$\begin{aligned}\hat{d} &= \arg \max_d \frac{\exp(\text{score}(d, t, s))}{\sum_{d', t'} \exp(\text{score}(d', t', s))} \\ &= \arg \max_d \exp(\text{score}(d, t, s))\end{aligned}\quad (7.12)$$

公式(7.12)中， $\exp(\text{score}(d, t, s))$  表示指数化的模型得分，记为  $\text{mscore}(d, t, s) = \exp(\text{score}(d, t, s))$ 。于是，翻译问题就可以被描述为：找到使函数  $\text{mscore}(d, t, s)$  达到最大的  $d$ 。由于， $\exp(\text{score}(d, t, s))$  和  $\text{score}(d, t, s)$  是单调一致的，因此有时也直接把  $\text{score}(d, t, s)$  当做模型得分。

### 7.2.3 判别式模型中的特征

判别式模型最大的好处在于它可以更灵活地引入特征。某种意义上，每个特征都是在描述翻译的某方面属性。在各种统计分类模型中，也大量使用了“特征”这个概念（见第三章）。比如，要判别一篇新闻是体育方面的还是文化方面的，可以设计一个分类器，用词作为特征。这个分类器就会有能力区分“体育”和“文化”两个类别的特征，最终决定这篇文章属于哪个类别。统计机器翻译也在做类似的事情。系统研发者可以通过设计翻译相关的特征，来区分不同翻译结果的好坏。翻译模型会综合这些特征对所有可能的译文进行打分和排序，并选择得分最高的译文输出。

在判别式模型中，系统开发者可以设计任意的特征来描述翻译，特征的设计甚至都不需要统计上的解释，包括 0-1 特征、计数特征等。比如，可以设计特征来回答“you 这个单词是否出现在译文中？”。如果答案为真，这个特征的值为 1，否则为 0。再比如，可以设计特征来回答“译文里有多少个单词？”。这个特征相当于一个统计目标语单词数的函数，它的值即为译文的长度。此外，还可以设计更加复杂的实数特征，甚至具有概率意义的特征。在随后的内容中还将看到，翻译的调序、译文流畅度等都会被建模为特征，而机器翻译系统会融合这些特征，综合得到最优的输出译文。

此外，判别式模型并不需要像生成式模型那样对问题进行具有统计学意义的“分解”，更不需要对每个步骤进行严格的数学推导。相反，它直接对问题的后验概率进行建模。由于不像生成式模型那样需要引入假设来对每个生成步骤进行化简，判别式模型对问题的刻画更加直接，因此也受到自然语言处理研究者的青睐。

### 7.2.4 搭建模型的基本流程

对于翻译的判别式建模，需要回答两个问题：

- 如何设计特征函数  $\{h_i(d, t|s)\}$ ？
- 如何获得最好的特征权重  $\{\lambda_i\}$ ？

在基于短语的翻译模型中，通常包含三类特征：短语翻译特征、调序特征、语言模型相关的特征。这些特征都需要从训练数据中学习。

图7.8展示了一个基于短语的机器翻译模型的搭建流程。其中的训练数据包括双语平行语料和目标语言单语语料。首先，需要从双语平行数据中学习短语的翻译，并形成一个短语翻译表；然后，再从双语平行数据中学习调序模型；最后，从目标语单语数据中学习语言模型。短语翻译表、调序模型、语言模型都会作为特征被送入判别式模型，由解码器完成对新句子的翻译。而这些特征的权重可以在额外的开发集上进行调优。关于短语抽取、调序模型和翻译特征的学习，会在本章的7.3-7.6节进行介绍。

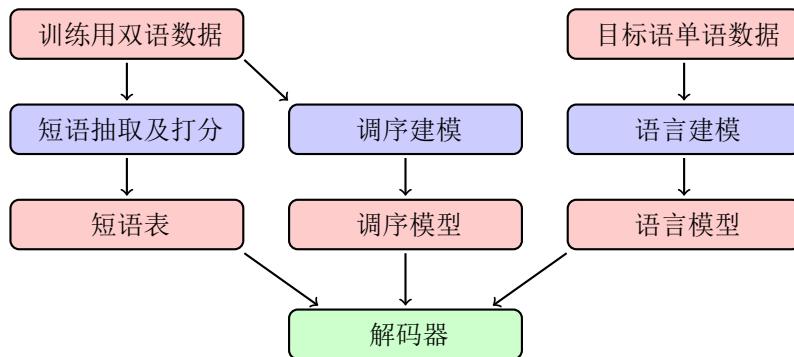


图 7.8 基于短语的机器翻译的系统流程

### 7.3 短语抽取

在基于短语的模型中，学习短语翻译是重要的步骤之一。获得短语翻译的方法有很多种，最常用的方法是从双语平行语料中进行**短语抽取**（Phrase Extraction）。前面已经介绍过短语的概念，句子中任意的连续子串都被称为短语。例如在图7.9中，用点阵的形式来表示双语之间的对应关系，那么图中任意一个矩形框都可以构成一个双语短语（或短语对），例如“什么/都/没”对应“learned nothing ?”。

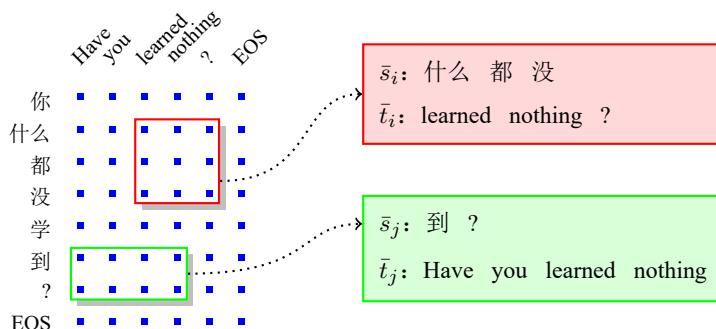


图 7.9 无限制的短语抽取

按照上述抽取短语的方式可以找到所有可能的双语短语，但是这种不加限制的抽取是十分低效的。一是可抽取的短语数量爆炸，二是抽取得到的大部分短语是没有意义的，如上面的例子中抽取到“到/?”对应“Have you learned nothing”这样的短语对在翻译中并没有什么意义。对于这个问题，一种解决方法是基于词对齐进行短语抽取，另一种是抽取与词对齐相一致的短语。

### 7.3.1 与词对齐一致的短语

图7.10中大蓝色方块代表词对齐。通过词对齐信息，可以很容易地获得双语短语“天气  $\leftrightarrow$  The weather”。这里称其为与词对齐一致（兼容）的双语短语。具体定义如下：

#### 定义 7.3.1 与词对齐一致（兼容）的双语短语

对于源语言句子  $s$  和目标语句子  $t$ ，存在  $s$  和  $t$  之间的词对齐。如果有  $(s, t)$  中的双语短语  $(\bar{s}, \bar{t})$ ，且  $\bar{s}$  中所有单词仅对齐到  $\bar{t}$  中的单词，同时  $\bar{t}$  中所有单词仅对齐到  $\bar{s}$  中的单词，那么称  $(\bar{s}, \bar{t})$  与是与词对齐一致的（兼容的）双语短语。

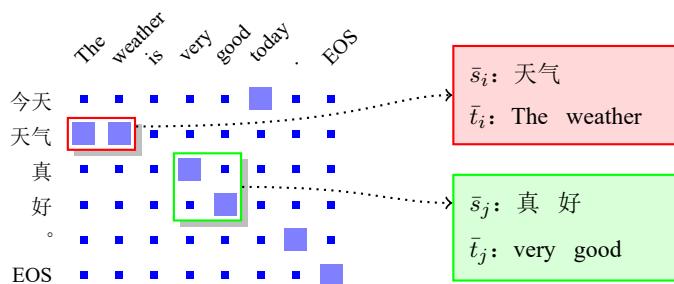


图 7.10 与词对齐一致的短语抽取

如图7.11所示，左边的例子中的  $t_1$  和  $t_2$  严格地对应到  $s_1$ 、 $s_2$ 、 $s_3$ ，所以短语是与词对齐相一致的；中间例子中的  $t_2$  对应到短语  $s_1$  和  $s_2$  的外面，所以短语是与词对齐不一致的；类似的，右边的例子中短语与词对齐也是相一致的。

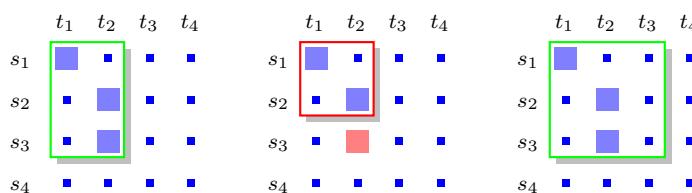


图 7.11 词对齐一致性示例

图7.12展示了与词对齐一致的短语抽取过程，首先判断抽取得到的双语短语是

否与词对齐保持一致，若一致，则抽取出来。在实际抽取过程中，通常需要对短语的最大长度进行限制，以免抽取过多的无用短语。比如，在实际系统中，最大短语长度一般是 5-7 个词。

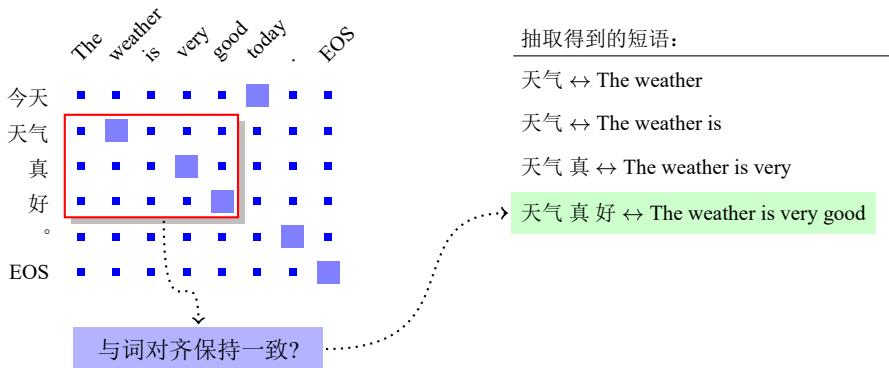


图 7.12 与词对齐一致的短语抽取

### 7.3.2 获取词对齐

如何获得词对齐呢？第五章和第六章介绍的 IBM 模型本身就是一个词对齐模型，因此一种常用的方法是直接使用 IBM 模型生成词对齐。IBM 模型约定每个源语言单词必须对应、也只能对应到一个目标语单词。因此，IBM 模型得到的词对齐结果是不对称的。正常情况下词对齐可以是一个源语言单词对应多个目标语言单词，或者多对一，甚至多对多的情况。为了获得对称的词对齐，一种简单的方法是，分别进行正向翻译和反向翻译的词对齐，然后利用启发性方法生成对称的词对齐，例如，双向词对齐取交集、并集等。

如图7.13中，左边两个图就是正向和反向两种词对齐的结果。右边的图是融合双向词对齐的结果，取交集是蓝色的方框，取并集是红色的方框。当然，还可以设计更多的启发性规则生成词对齐<sup>[273]</sup>。

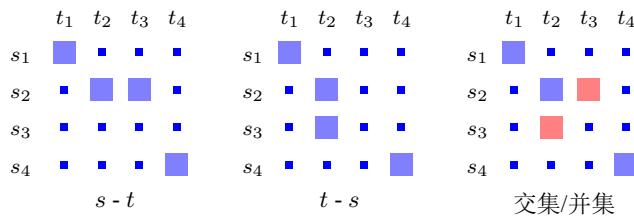


图 7.13 词对齐的获取

除此之外，一些外部工具也可以用来获取词对齐，如 Fastalign<sup>[250]</sup>、Berkeley Word Aligner<sup>[251]</sup> 等。词对齐的质量通常使用词对齐错误率（AER）来评价<sup>[274]</sup>，但是词对齐

并不是一个独立的系统，它一般会服务于其他任务。因此，也可以使用下游任务来评价词对齐的好坏。比如，改进词对齐后观察机器翻译系统性能的变化。

### 7.3.3 度量双语短语质量

抽取双语短语之后，需要对每个双语短语的质量进行评价。这样，在使用这些双语短语时，可以更有效地估计整个句子翻译的好坏。在统计机器翻译中，一般用双语短语出现的可能性大小来度量双语短语的好坏。这里，使用相对频次估计对短语的翻译条件概率进行计算，公式如下：

$$P(\bar{t}|\bar{s}) = \frac{c(\bar{s}, \bar{t})}{c(\bar{s})} \quad (7.13)$$

给定一个双语句对  $(s, t)$ ， $c(\bar{s})$  表示短语  $\bar{s}$  在  $s$  中出现的次数， $c(\bar{s}, \bar{t})$  表示双语短语  $(\bar{s}, \bar{t})$  在  $(s, t)$  中被抽取出来的次数。对于一个包含多个句子的语料库， $c(\bar{s})$  和  $c(\bar{s}, \bar{t})$  可以按句子进行累加。类似的，也可以用同样的方法，计算  $\bar{t}$  到  $\bar{s}$  的翻译概率，即  $P(\bar{s}|\bar{t})$ 。一般会同时使用  $P(\bar{t}|\bar{s})$  和  $P(\bar{s}|\bar{t})$  度量一个双语短语的好与坏。

当遇到低频短语时，短语翻译概率的估计可能会不准确。例如，短语  $\bar{s}$  和  $\bar{t}$  在语料中只出现了一次，且在一个句子中共现，那么  $\bar{s}$  到  $\bar{t}$  的翻译概率为  $P(\bar{t}|\bar{s}) = 1$ ，这显然是不合理的，因为  $\bar{s}$  和  $\bar{t}$  的出现完全可能是偶然事件。既然直接度量双语短语的好坏会面临数据稀疏问题，一个自然的想法就是把短语拆解成单词，利用双语短语中单词翻译的好坏间接度量双语短语的好坏。为了达到这个目的，可以使用**词汇化翻译概率**（Lexical Translation Probability）。前面借助词对齐信息完成了双语短语的抽取，因此，词对齐信息本身就包含了短语内部单词之间的对应关系。因此同样可以借助词对齐来计算词汇翻译概率，公式如下：

$$P_{\text{lex}}(\bar{t}|\bar{s}) = \prod_{j=1}^{|\bar{s}|} \frac{1}{|\{j|a(j,i) = 1\}|} \sum_{\forall(j,i):a(j,i)=1} \sigma(t_i|s_j) \quad (7.14)$$

它表达的意思是短语  $\bar{s}$  和  $\bar{t}$  存在词汇级的对应关系，其中  $a(j,i) = 1$  表示双语句对  $(s, t)$  中单词  $s_j$  和单词  $t_i$  对齐， $\sigma$  表示词汇翻译概率用来度量两个单词之间翻译的可能性大小（见第五章），作为两个词之间对应的强度。

	$t_1$	$t_2$	$t_3$	$t_4$	$N$	$P_{\text{lex}}(\bar{t} \bar{s}) = \sigma(t_1 s_1) \times$
$s_1$	■	■	■	■	■	$\frac{1}{2}(\sigma(t_2 s_2) + \sigma(t_3 s_2)) \times$
$s_2$	■	■	■	■	■	$\sigma(N s_3) \times$
$s_3$	■	■	■	■	■	$\sigma(t_4 s_4) \times$
$s_4$	■	■	■	■	■	

图 7.14 词汇翻译概率实例

来看一个具体的例子，如图7.14所示。对于一个双语短语，将它们的词对齐关系代入到公式(7.14)就会得到短语的词汇翻译概率。对于词汇翻译概率，可以使用 IBM 模型中的单词翻译表，也可以通过统计获得<sup>[275]</sup>。如果一个单词的词对齐为空，则用  $N$  表示它翻译为空的概率。和短语翻译概率一样，可以使用双向的词汇化翻译概率来评价双语短语的好坏。

经过上面的介绍，可以从双语平行语料中把双语短语抽取出来，同时得到相应的翻译概率（即特征），组成**短语表**（Phrase Table）。图7.15展示了一个真实短语表的片段。其中包括源语言短语和目标语言短语，用 ||| 进行分割。每个双语对应的得分，包括正向和反向的词汇翻译概率以及短语翻译概率，还包括词对齐信息（0-0、1-1）等其他信息。

```
...
报告认为 ||| report holds that ||| -2.62 -5.81 -0.91 -2.85 1 0 ||| 4 ||| 0-0 1-1 1-2
, 悲伤 ||| , sadness ||| -1.946 -3.659 0 -3.709 1 0 ||| 1 ||| 0-0 1-1
, 北京等 ||| , beijing , and other ||| 0 -7.98 0 -3.84 1 0 ||| 2 ||| 0-0 1-1 2-2 2-3 2-4
, 北京及 ||| , beijing , and ||| -0.69 -1.45 -0.92 -4.80 1 0 ||| 2 ||| 0-0 1-1 2-2
一个世界 ||| one world ||| 0 -1.725 0 -1.636 1 0 ||| 2 ||| 1-1 2-2
...
...
```

图 7.15 短语表实例

## 7.4 翻译调序建模

尽管已经知道了如何将一个源语言短语翻译成目标语言短语，但是想要获得一个高质量的译文，仅有互译的双语短语是远远不够的。

如图7.16所示，按照从左到右的顺序对一个句子“在/桌子/上/的/苹果”进行翻译，得到的译文“on the table the apple”的语序是不对的。虽然可以使用  $n$ -gram 语言模型对语序进行建模，但是此处仍然需要用更加准确的方式描述目标语短语间的次序。一般，把这个问题称为短语调序，或者简称**调序**（Reordering）。通常，基于短语的调序模型会作为判别式模型的特征参与到翻译过程中来。接下来，会介绍 3 种不同的调序方法，分别是基于距离的调序、基于方向的调序（MSD 模型）以及基于分类的调序。

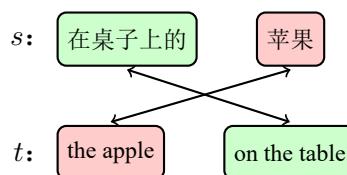


图 7.16 基于短语翻译的调序

### 7.4.1 基于距离的调序

基于距离的调序是最简单的一种调序模型。第六章中所讨论的“扭曲度”本质上就是一种调序模型。只不过第六章所涉及的扭曲度描述的是单词的调序问题，而这里需要把类似的概念推广到短语。

基于距离的调序的一个基本假设是：语言的翻译基本上都是顺序的，也就是，译文单词出现的顺序和源语言单词的顺序基本上是一致的。反过来说，如果译文和源语言单词（或短语）的顺序差别很大，就认为出现了调序。

基于距离的调序方法的核心思想就是度量当前翻译结果与顺序翻译之间的差距。对于译文中的第  $i$  个短语，令  $\text{start}_i$  表示它所对应的源语言短语中第一个词所在的位置， $\text{end}_i$  表示它所对应的源语言短语中最后一个词所在的位置。于是，这个短语（相对于前一个短语）的调序距离为：

$$dr = \text{start}_i - \text{end}_{i-1} - 1 \quad (7.15)$$

在图7.17的例子中，“the apple”所对应的调序距离为 4，“on the table”所对应的调序距离为  $-5$ 。显然，如果两个源语短语按顺序翻译，则  $\text{start}_i = \text{end}_{i-1} + 1$ ，这时调序距离为 0。

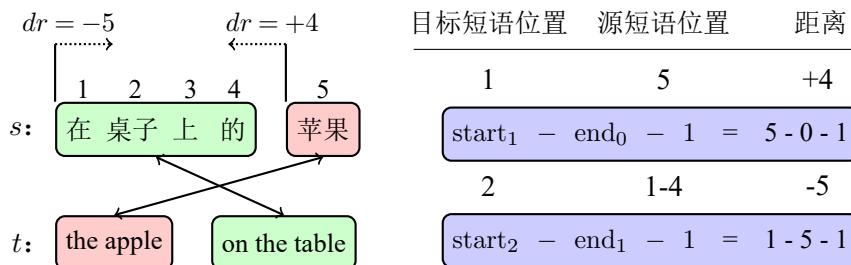


图 7.17 基于距离的调序

如果把调序距离作为特征，一般会使用指数函数  $f(dr) = a^{|dr|}$  作为特征函数（或者调序代价的函数），其中  $a$  是一个参数，控制调序距离对整个特征值的影响。调序距离  $dr$  的绝对值越大，调序代价越高。基于距离的调序模型比较适用于像法语到英语翻译这样的任务，因为两种语言的语序基本上是一致的。但是，对于汉语到日语翻译，由于句子结构存在很大差异（日语是谓词后置，而汉语中谓词放在宾语前），使用基于距离的调序会带来一些问题。因此，具体应用时应该根据语言之间的差异性有选择地使用该模型。

### 7.4.2 基于方向的调序

基于方向的调序模型是另一种常用的调序模型。该模型是一种典型的词汇化调序模型，因此调序的结果会根据不同短语有所不同。简单来说，在两个短语目标语

言端连续的情况下，该模型会判断两个双语短语在源语言端的调序情况，包含三种调序类型：顺序的单调翻译（M）、与前一个短语交换位置（S）、非连续翻译（D）。因此，这个模型也被称作 MSD 调序模型，也是 Moses 等经典的机器翻译系统所采用的调序模型<sup>[80]</sup>。

图7.18展示了这三种调序类型，当两个短语对在源语言和目标语言中都是按顺序排列时，它们就是单调的（如：从左边数前两个短语）；如果对应的短语顺序在目标语中是反过来的，属于交换调序（如：从左边数第三和第四个短语）；如果两个短语之间还有其他的短语，就是非连续调序（如：从右边数的前两个短语）。

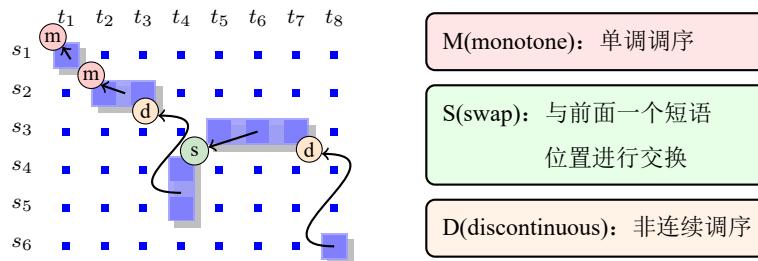


图 7.18 词汇化调序模型的三种调序类型

对于每种调序类型，都可以定义一个调序概率，如下：

$$P(o|s, t, a) = \prod_{i=1}^K P(o_i | \bar{s}_{a_i}, \bar{t}_i, a_{i-1}, a_i) \quad (7.16)$$

其中， $o_i$  表示（目标语言）第  $i$  个短语的调序方向， $o = \{o_i\}$  表示短语序列的调序方向， $K$  表示短语的数量。短语之间的调序概率是由双语短语以及短语对齐决定的， $o$  表示调序的种类，可以取 M、S、D 中的任意一种。而整个句子调序的好坏就是把相邻的短语之间的调序概率相乘（对应取 log 后的加法）。这样，公式(7.16)把调序的好坏定义为新的特征，对于 M、S、D 总共就有三个特征。除了当前短语和前一个短语的调序特征，还可以定义当前短语和后一个短语的调序特征，即将上述公式中的  $a_{i-1}$  换成  $a_{i+1}$ 。于是，又可以得到三个特征。因此在 MSD 调序中总共可以有 6 个特征。

具体实现时，通常使用词对齐对两个短语间的调序关系进行判断。图7.19展示了这个过程。先判断短语的左上角和右上角是否存在词对齐，再根据其位置对调序类型进行划分。每个短语对应的调序概率都可以用相对频次估计进行计算。而 MSD 调序模型也相当于在短语表中的每个双语短语后添加 6 个特征。不过，调序模型一般并不会和短语表一起存储，因此在系统中通常会看到两个独立的模型文件，分别保存短语表和调序模型。

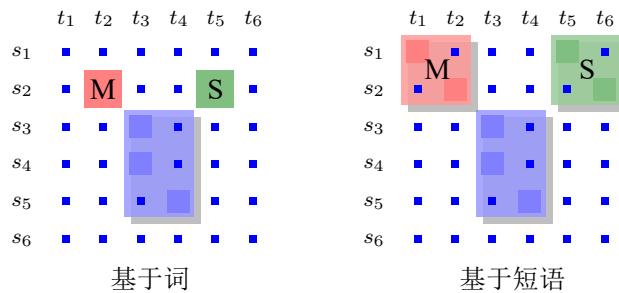


图 7.19 调序类型的判断

### 7.4.3 基于分类的调序

在 MSD 调序中，双语短语所对应的调序概率  $P(o_i|\bar{s}_{a_i}, \bar{t}_i, a_{i-1}, a_i)$  是用极大似然估计方法进行计算的。但是，这种方法也会面临数据稀疏问题，同时对调序产生影响的细致特征也没有考虑进来。另一种有效的方法是直接用统计分类模型对调序进行建模，比如，可以使用最大熵、SVM 等分类器输出调序概率或者得分<sup>[266, 267, 268]</sup>。对于基于分类的调序模型，有两方面问题需要考虑：

- 训练样本的生成。可以把 M、S、D 看作是类别标签，把所对应的短语及短语对齐信息看作是输入。这样就得到了大量分类器训练所需的样本。
- 分类特征设计。这部分是传统统计机器学习中的重要组成部分，好的特征会对分类结果产生很大影响。在调序模型中，一般直接使用单词作为特征，比如用短语的第一个单词和最后一个单词作为特征就可以达到很好的效果。

随着神经网络方法的兴起，也可以考虑使用多层神经网络构建调序模型<sup>[269]</sup>。这时，可以把短语直接送入一个神经网络，之后由神经网络完成对特征的抽取和表示，并输出最终的调序模型得分。

## 7.5 翻译特征

基于短语的模型使用判别式模型对翻译推导进行建模，给定双语句对  $(s, t)$ ，每个翻译推导  $d$  都有一个模型得分，由  $M$  个特征线性加权得到，记为  $\text{score}(d, t, s) = \sum_{i=1}^M \lambda_i \cdot h_i(d, t, s)$ ，其中  $\lambda_i$  表示特征权重， $h_i(d, t, s)$  表示特征函数（简记为  $h_i(d)$ ）。这些特征包含刚刚介绍过的短语翻译概率、调序模型得分等，除此之外，还包含语言模型等其他特征，它们共同组成了特征集合。这里列出了基于短语的模型中的一些基础特征：

- 短语翻译概率(取对数)，包含正向翻译概率  $\log(P(\bar{t}|\bar{s}))$  和反向翻译概率  $\log(P(\bar{s}|\bar{t}))$ ，它们是基于短语的模型中最主要的特征。

- 词汇化翻译概率（取对数），同样包含正向词汇化翻译概率  $\log(P_{\text{lex}}(\bar{t}|\bar{s}))$  和反向词汇化翻译概率  $\log(P_{\text{lex}}(\bar{s}|\bar{t}))$ ，它们用来描述双语短语中单词间对应的好坏。
- $n$ -gram 语言模型，用来度量译文的流畅程度，可以通过大规模目标端单语数据得到。
- 译文长度，避免模型倾向于短译文，同时让系统自动学习对译文长度的偏好。
- 翻译规则数量，为了避免模型仅使用少量特征构成翻译推导（规则数量少，短语翻译概率相乘的因子也会少，得分一般会大一些），同时让系统自动学习对规则数量的偏好。
- 被翻译为空的源语言单词数量。注意，空翻译特征有时也被称作**有害特征**（Evil Feature），这类特征在一些数据上对 BLEU 有很好的提升作用，但会造成人工评价结果的下降，需要谨慎使用。
- 基于 MSD 的调序模型，包括与前一个短语的调序模型  $f_{M\text{-pre}}(d)$ 、 $f_{S\text{-pre}}(d)$ 、 $f_{D\text{-pre}}(d)$  和与后一个短语的调序模型  $f_{M\text{-fol}}(d)$ 、 $f_{S\text{-fol}}(d)$ 、 $f_{D\text{-fol}}(d)$ ，共 6 个特征。

## 7.6 最小错误率训练

除了特征设计，统计机器翻译也需要找到每个特征所对应的最优权重  $\lambda_i$ 。这也就是机器学习中所说的模型训练问题。不过，需要指出的是，统计机器翻译关于模型训练的定义与传统机器学习稍有不同。在统计机器翻译中，短语抽取和翻译概率的估计被看作是**模型训练**（Model Training），也就是说这里的模型训练是指特征函数的学习；而特征权重的训练，一般被称作**权重调优**（Weight Tuning），而这个过程才真正对应了传统机器学习（如分类任务）中的模型训练过程。在本章中，如果没有特殊说明，权重调优就是指特征权重的学习，模型训练是指短语抽取和特征函数的学习。

想要得到最优的特征权重，最简单的方法是枚举所有特征权重可能的取值，然后评价每组权重所对应的翻译性能，最后选择最优的特征权重作为调优的结果。但是特征权重是一个实数值，因此可以考虑把实数权重进行量化，即把权重看作是在固定间隔上的取值，比如，每隔 0.01 取值。即使这样，同时枚举多个特征的权重也是非常耗时的工作，当特征数量增多时这种方法的效率仍然很低。

这里介绍一种更加高效的特征权重调优方法——**最小错误率训练**（Minimum Error Rate Training, MERT）。最小错误率训练是统计机器翻译发展中代表性工作，也是机器翻译领域原创的重要技术方法之一<sup>[232]</sup>。最小错误率训练假设：翻译结果相对于标准答案的错误是可度量的，进而可以通过降低错误数量的方式来找到最优的特征权重。假设有样本集合  $S = \{(s^{[1]}, r^{[1]}), \dots, (s^{[N]}, r^{[N]})\}$ ， $s^{[i]}$  为样本中第  $i$  个源语言句子， $r^{[i]}$  为相应的参考译文。注意， $r^{[i]}$  可以包含多个参考译文。 $S$  通常被称为**调优集合**（Tuning Set）。对于  $S$  中的每个源语句子  $s^{[i]}$ ，机器翻译模型会解码出  $n$ -best 推

导  $\hat{d}^{[i]} = \{\hat{d}_j^{[i]}\}$ , 其中  $\hat{d}_j^{[i]}$  表示对于源语言句子  $s^{[i]}$  得到的第  $j$  个最好的推导。 $\{\hat{d}_j^{[i]}\}$  可以被定义如下:

$$\{\hat{d}_j^{[i]}\} = \arg \max_{\{d_j^{[i]}\}} \sum_{i=1}^M \lambda_i \cdot h_i(d, t^{[i]}, s^{[i]}) \quad (7.17)$$

对于每个样本都可以得到  $n$ -best 推导集合, 整个数据集上的推导集合被记为  $\hat{D} = \{\hat{d}^{[1]}, \dots, \hat{d}^{[N]}\}$ 。进一步, 令所有样本的参考译文集合为  $R = \{r^{[1]}, \dots, r^{[N]}\}$ 。最小错误率训练的目标就是降低  $\hat{D}$  相对于  $R$  的错误。也就是, 通过调整不同特征的权重  $\lambda = \{\lambda_i\}$ , 让错误率最小, 形式化描述为:

$$\hat{\lambda} = \arg \min_{\lambda} \text{Error}(\hat{D}, R) \quad (7.18)$$

其中,  $\text{Error}(\cdot)$  是错误率函数。 $\text{Error}(\cdot)$  的定义方式有很多, 一般来说  $\text{Error}(\cdot)$  会与机器翻译的评价指标相关, 例如, 词错误率 (WER)、位置错误率 (PER)、BLEU 值、NIST 值等都可以用于  $\text{Error}(\cdot)$  的定义。这里使用 1-BLEU 作为错误率函数, 即  $\text{Error}(\hat{D}, R) = 1 - \text{BLEU}(\hat{D}, R)$ 。则公式(7.18)可改写为:

$$\begin{aligned} \hat{\lambda} &= \arg \min_{\lambda} (1 - \text{BLEU}(\hat{D}, R)) \\ &= \arg \max_{\lambda} \text{BLEU}(\hat{D}, R) \end{aligned} \quad (7.19)$$

需要注意的是, BLEU 本身是一个不可微分函数。因此, 无法使用梯度下降等方法对式(7.19)进行求解。那么如何能快速得到最优解? 这里会使用一种特殊的优化方法, 称作**线搜索** (Line Search), 它是 Powell 搜索的一种形式<sup>[276]</sup>。这种方法也构成了最小错误率训练的核心。

首先, 重新看一下特征权重的搜索空间。按照前面的介绍, 如果要进行暴力搜索, 需要把特征权重的取值按小的间隔进行划分。这样, 所有特征权重的取值可以用图7.20的网格来表示。

其中横坐标为所有的  $M$  个特征函数, 纵坐标为权重可能的取值。假设每个特征都有  $V$  种取值, 那么遍历所有特征权重取值的组合有  $M^V$  种。每组  $\lambda = \{\lambda_i\}$  的取值实际上就是一个贯穿所有特征权重的折线, 如图7.20中间蓝线所展示的路径。当然, 可以通过枚举得到很多这样的折线 (图7.20右)。假设计算 BLEU 的时间开销为  $B$ , 那么遍历所有的路径的时间复杂度为  $O(M^V \cdot B)$ , 由于  $V$  可能很大, 而且  $B$  往往也无法忽略, 因此这种计算方式的时间成本是极高的。如果考虑对每一组特征权重都需要重新解码得到  $n$ -best 译文, 那么基于这种简单枚举的方法是无法使用的。

对全搜索的一种改进是使用局部搜索。循环处理每个特征, 每一次只调整一个特征权重的值, 找到使 BLEU 达到最大的权重。反复执行该过程, 直到模型达到稳

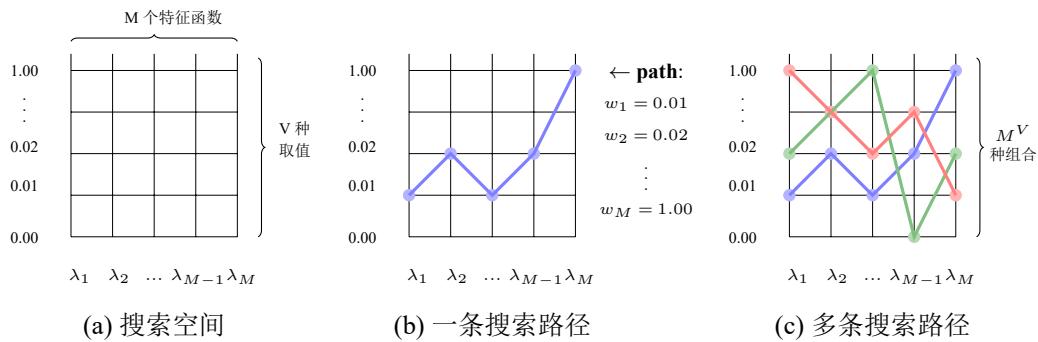


图 7.20 特征权重的搜索空间表示

定状态（例如 BLEU 不再降低）。

图 7.21 左侧展示了这种方法。其中蓝色部分为固定的权重，相应的虚线部分为当前权重所有可能的取值，这样搜索一个特征权重的时间复杂度为  $O(V \cdot B)$ 。而整个算法的时间复杂度为  $O(L \cdot V \cdot B)$ ，其中  $L$  为循环访问特征的总次数。这种方法也被称作**格搜索**（Grid Search）。

- 固定的权重
- 有效取值点
- 无效取值点

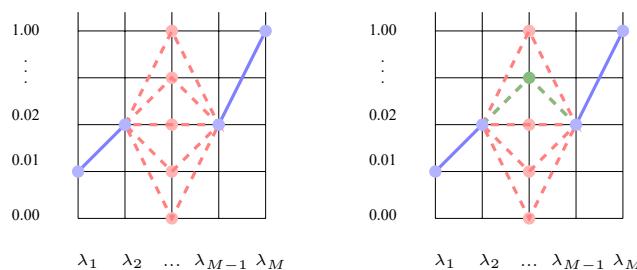


图 7.21 格搜索（左侧：所有点都访问（蓝色）；右侧：避开无效点（绿色））

格搜索的问题在于，每个特征都要访问  $V$  个点，且不说  $V$  个点无法对连续的特征权重进行表示，里面也会存在大量的无用访问。也就是说，这  $V$  个点中绝大多数点根本“不可能”成为最优的权重。可以把这样的点称为无效取值点。

能否避开这些权重取值点呢？再重新看一下优化的目标 BLEU。实际上，当一个特征权重发生变化时，BLEU 的变化只会出现在系统 1-best 译文发生变化的时候。那么，可以只关注使 1-best 译文发生变化的取值点，因为其他的取值点都不会使优化的目标函数产生变化。这也就构成了线搜索的思想。

假设对于每个输入的句子，翻译模型生成了两个推导  $d = \{d_1, d_2\}$ ，每个推导  $d$

的得分  $\text{score}(d)$  可以表示成关于第  $i$  个特征的权重  $\lambda_i$  的线性函数：

$$\begin{aligned}\text{score}(d) &= \sum_{k=1} \lambda_k \cdot h_k(d) \\ &= h_i(d) \cdot \lambda_i + \sum_{k \neq i} \lambda_k \cdot h_k(d) \\ &= a \cdot \lambda_i + b\end{aligned}\tag{7.20}$$

这里， $a = h_i(d)$  是直线的斜率， $b = \sum_{k \neq i} \lambda_k \cdot h_k(d)$  是截距。有了关于权重  $\lambda_i$  的直线表示，可以将  $d_1$  和  $d_2$  分别画成两条直线，如图7.22所示。在两条直线交叉点的左侧， $d_2$  是最优的翻译结果；在交叉点右侧， $d_1$  是最优的翻译结果。也就是说，只需知道交叉点左侧和右侧谁的 BLEU 值高， $\lambda_i$  的最优值就应该落在相应的范围，比如，这个例子中交叉点右侧（即  $d_2$ ）所对应的 BLEU 值更高，因此最优特征权重  $\hat{\lambda}_i$  应该在交叉点右侧 ( $\lambda_x \sim \lambda_i$  任意取值都可以)。

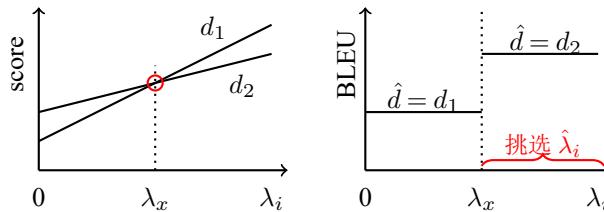


图 7.22 推导得分关于权重的函数（左）以及对应的 BLEU 值变化（右）

这样，最优权重搜索的问题就被转化为找到最优推导 BLEU 值发生变化的点的问题。理论上，对于  $n$ -best 翻译，交叉点计算最多需要  $\frac{n(n-1)}{2}$  次。由于  $n$  一般不会过大，这个时间成本完全是可以接受的。此外，在实现时还有一些技巧，比如，并不需要在每个交叉点处对整个数据集进行 BLEU 计算，可以只对 BLEU 产生变化的部分（比如  $n$ -gram 匹配的数量）进行调整，因此搜索的整体效率会进一步得到提高。相比于格搜索，线搜索可以确保在单个特征维度上的最优值，同时保证搜索的效率。

还有一些经验性的技巧用来完善基于线搜索的 MERT。例如：

- 随机生成特征权重的起始点。
- 搜索中，给权重加入一些微小的扰动，避免陷入局部最优。
- 随机选择特征优化的顺序。
- 使用先验知识来指导 MERT（对权重的取值范围进行约束）。
- 使用多轮迭代训练，最终对权重进行平均。

最小错误率训练最大的优点在于可以用于目标函数不可微、甚至不连续的情况。对于优化线性模型，最小错误率训练是一种很好的选择。但是，也有研究发现，直接使用最小错误率训练无法处理特征数量过多的情况。比如，用最小错误率训练优

化 10000 个稀疏特征的权重时，优化效果可能会不理想，而且收敛速度慢。这时也可以考虑使用在线学习等技术对大量特征的权重进行调优，比较有代表性的方法包括 MIRA<sup>[277]</sup> 和 PRO<sup>[278]</sup>。由于篇幅所限，这里不对这些方法做深入讨论，感兴趣的读者可以参考 7.8 节的内容，对相关文献进行查阅。

## 7.7 栈解码

解码的目的是根据模型以及输入，找到模型得分最高的推导，即：

$$\hat{d} = \arg \max_d \text{score}(d, t, s) \quad (7.21)$$

然而想要找到得分最高的翻译推导并不是一件简单的事情。对于每一句源语言句子，可能的翻译结果是指数级的。由于机器翻译解码是一个 NP 完全问题<sup>[238]</sup>，简单的暴力搜索显然不现实。因此，在机器翻译中会使用特殊的解码策略来确保搜索的效率。本节将介绍基于栈的自左向右解码方法。它是基于短语的模型中的经典解码方法，非常适于处理语言生成的各种任务。

首先，看一下翻译一个句子的基本流程。如图 7.23 所示，首先需要得到译文句子的第一个单词。在基于短语的模型中，可以从源语言端找出生成句首译文的短语，之后把译文放到目标语言端，例如，源语言的“有”对应的译文是“*There is*”。这个过程可以重复执行，直到生成完整句子的译文。但是，有两点需要注意：

- 源语言的每个单词（短语）只能被翻译一次。
- 译文的生成需自左向右连续进行。

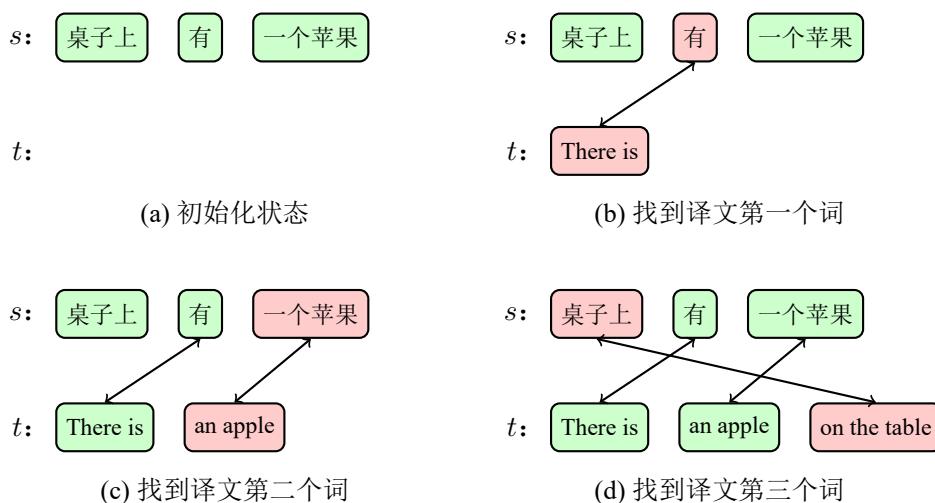


图 7.23 按目标语言短语自左向右生成的翻译实例

第一点对应了一种**覆盖度模型** (Coverage Model)；第二点定义了解码的方向，这

样可以确保  $n$ -gram 语言模型的计算是准确的。这样，就得到了一个简单的基于短语的机器翻译解码框架。每次从源语言句子中找到一个短语，作为译文最右侧的部分，重复执行直到整个译文被生成出来。

### 7.7.1 翻译候选匹配

在解码时，首先要知道每个源语言短语可能的译文都是什么。对于一个源语言短语，每个可能的译文也被称作翻译候选。实现翻译候选的匹配很简单。只需要遍历输入的源语言句子中所有可能的短语，之后在短语表中找到相应的翻译即可。比如，图7.24展示了句子“桌子/上/有/一个/苹果”的翻译候选匹配结果。可以看到，不同的短语会对应若干翻译候选。这些翻译候选会保存在所对应的范围（被称为跨度）中。这里，跨度  $[a, b]$  表示从第  $a+1$  个词开始到第  $b$  个词为止所表示的词串。比如，“upon the table” 是短语“桌子/上/有”的翻译候选，即对应源语言跨度 [0,3]。

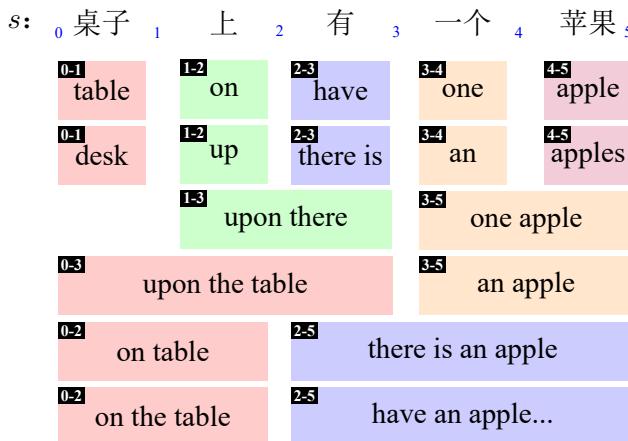


图 7.24 一个句子匹配的短语翻译候选

### 7.7.2 翻译假设扩展

接下来，需要使用这些翻译候选生成完整的译文。在机器翻译中，一个很重要的概念是**翻译假设** (Translation Hypothesis)。它可以被当作是一个局部译文所对应的短语翻译推导。在解码开始时，只有一个空假设，也就是任何译文单词都没有被生成出来。接着，可以挑选翻译选项来扩展当前的翻译假设。

图7.25展示了翻译假设扩展的过程。在翻译假设扩展时，需要保证新加入的翻译候选放置在旧翻译假设译文的右侧，也就是要确保翻译自左向右的连续性。而且，同一个翻译假设可以使用不同的翻译候选进行扩展。例如，扩展第一个翻译假设时，可以选择“桌子”的翻译候选“table”；也可以选择“有”的翻译候选“There is”。扩展完之后需要记录输入句子中已翻译的短语，同时计算当前所有翻译假设的模型得分。这个过程相当于生成了一个图的结构，每个节点代表了一个翻译假设。当翻译

假设覆盖了输入句子所有的短语，不能被继续扩展时，就生成了一个完整的翻译假设（译文）。最后需要找到得分最高的完整翻译假设，它对应了搜索图中的最优路径。

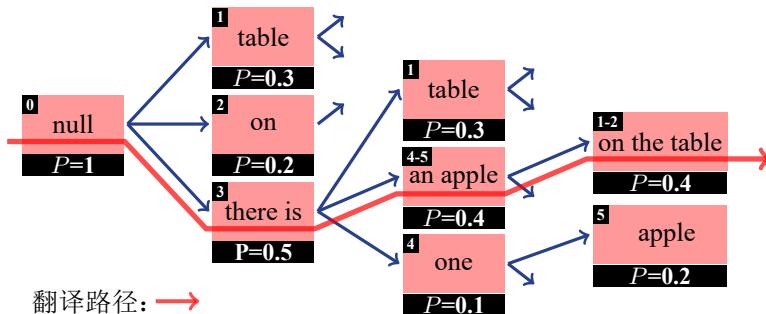


图 7.25 翻译假设扩展

### 7.7.3 剪枝

假设扩展建立了解码算法的基本框架。但是，当句子变长时，这种方法还是面临着搜索空间爆炸的问题。对于这个问题，常用的解决办法是剪枝，也就是在搜索图中排除掉一些节点。比如，可以使用束剪枝，确保每次翻译扩展时，最多生成  $k$  个新的翻译假设。这里  $k$  可以被看做是束的宽度。通过控制  $k$  的大小，可以在解码精度和速度之间进行平衡。这种基于束宽度进行剪枝的方法也被称作直方图剪枝。另一种思路是，每次扩展时只保留与最优翻译假设得分相差在  $\delta$  之内的翻译假设。 $\delta$  可以被看作是一种与最优翻译假设之间距离的阈值，超过这个阈值就被剪枝。这种方法也被称作**阈值剪枝**（Threshold Pruning）。

不过，即使引入束剪枝，解码过程中仍然会有很多冗余的翻译假设。有两种方法可以进一步加速解码：

- 对相同译文的翻译假设进行重新组合；
- 对低质量的翻译假设进行裁剪。

对翻译假设进行重新组合又被称作**假设重组**（Hypothesis Recombination）。其核心思想是，把代表同一个译文的不同翻译假设融合为一个翻译假设。如图7.26所示，对于给定的输入短语“一个 苹果”，系统可能将两个单词“一个”、“苹果”分别翻译成“an”和“apple”，也可能将这两个单词作为一个短语直接翻译成“an apple”。虽然这两个翻译假设得到的译文相同，并且覆盖了相同的源语言短语，但是却是两个不同的翻译假设，模型给它们的打分也是不一样的。这时，可以舍弃两个翻译假设中分数较低的那个，因为分数较低的翻译假设永远不可能成为最优路径的一部分。这也就相当于把两个翻译假设重组为一个假设。

即使翻译假设对应的译文不同也可以进行假设重组。图7.26的下半部分给出了一个这样的实例。在两个翻译假设中，第一个单词分别被翻译成了“it”和“he”，紧

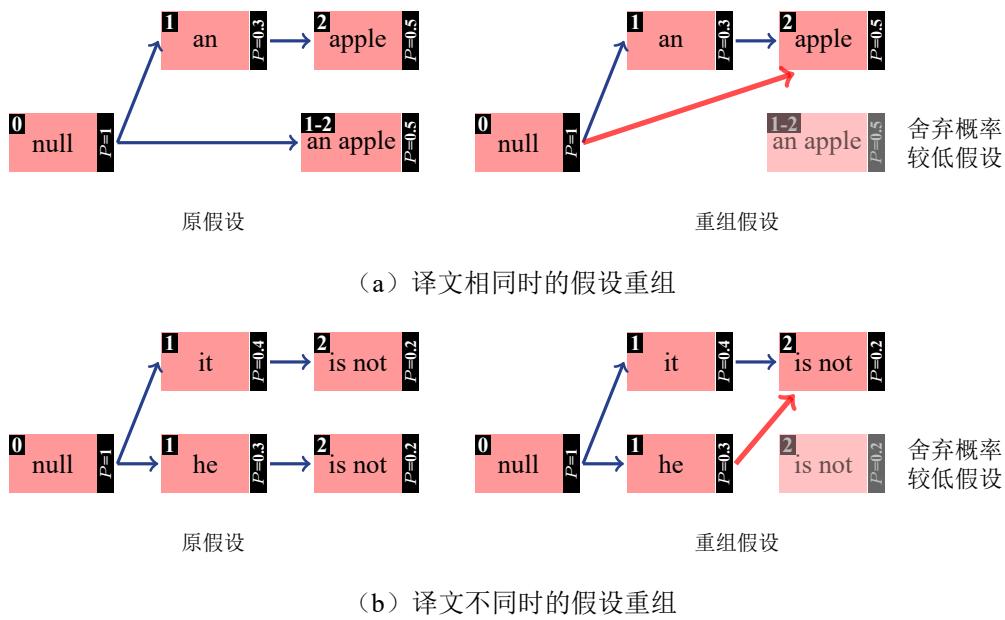


图 7.26 假设重组示例

接着它们后面的部分都被翻译成了“is not”。这两个翻译假设是非常相似的，因为它们译文的最后两个单词是相同的，而且翻译假设都覆盖了相同的源语言部分。这时，也可以对这两个翻译假设进行假设重组：如果得分较低的翻译假设和得分较高的翻译假设都使用相同的翻译候选进行扩展，且两个翻译假设都覆盖相同的源语言单词，分数低的翻译假设可以被剪枝掉。此外，还有两点需要注意：

- $n$ -gram 语言模型将前  $n - 1$  个单词作为历史信息，所以当两个假设最后  $n - 1$  个单词不相同时，不能进行假设重组，因为后续的扩展可能会得到不同的语言模型得分，并影响最终的模型得分。
- 调序模型通常是用来判断当前输入的短语与前一个输入短语之间的调序代价。因此当两个翻译假设对应短语在源语言中的顺序不同时，也不能被重新组合。

然而在实际处理中，并不需要“删掉”分数低的翻译假设，而是将它们与分数高的翻译假设连在了一起。对于搜索最优翻译，这些连接可能并没有什么作用，但是如果需要分数最高的前两个或前三个翻译，就可能需要用到这些连接。

翻译假设的重组有效地减少了解码过程中相同或者相似翻译假设带来的冗余。因此这些方法在机器翻译中被广泛使用。包括第八章将要介绍的基于句法的翻译模型解码中，也可以使用假设重组进行系统加速。

#### 7.7.4 解码中的栈结构

当质量较差的翻译假设在扩展早期出现时，这些翻译假设需要被剪枝掉，这样可以忽略所有从它扩展出来的翻译假设，进而有效地减小搜索空间。但是这样做也

存在着一定的问题：

- 删除的翻译假设可能会在后续的扩展过程中被重新搜索出来；
- 过早地删除某些翻译假设可能导致无法搜索到最优的翻译假设。

所以最好的情况是尽早删除质量差的翻译假设，这样就不会对整个搜索结果产生过大影响。但是这个“质量”从哪个方面来衡量，也是一个需要思考的问题。理想的情况就是从早期的翻译假设中，挑选一些可比的翻译假设进行筛选。

目前比较通用的做法是将翻译假设进行整理，放进一种栈结构中。这里所说的“栈”是为了描述方便的一种说法。它实际上就是保存多个翻译假设的一种数据结构<sup>4</sup>。当放入栈的翻译假设超过一定阈值时（比如 200），可以删除掉模型得分低的翻译假设。一般，会使用多个栈来保存翻译假设，每个栈代表覆盖源语言单词数量相同的翻译假设。

比如，第一个堆栈包含了覆盖一个源语言单词的翻译假设，第二个堆栈包含了覆盖两个源语言单词的翻译假设，以此类推。利用覆盖源语言单词数进行栈的划分的原因在于：翻译相同数量的单词所对应的翻译假设一般是“可比的”，因此在同一个栈里对它们进行剪枝带来的风险较小。

在基于栈的解码中，每次都会从所有的栈中弹出一个翻译假设，并选择一个或者若干个翻译假设进行扩展，之后把新得到的翻译假设重新压入解码栈中。这个过程不断执行，并可以配合束剪枝、假设重组等技术。最后在覆盖所有源语言单词的栈中得到整个句子的译文。图7.27展示了一个简单的栈解码过程。第一个栈（0号栈）用来存放空翻译假设。之后通过假设扩展，不断将翻译假设填入对应的栈中。

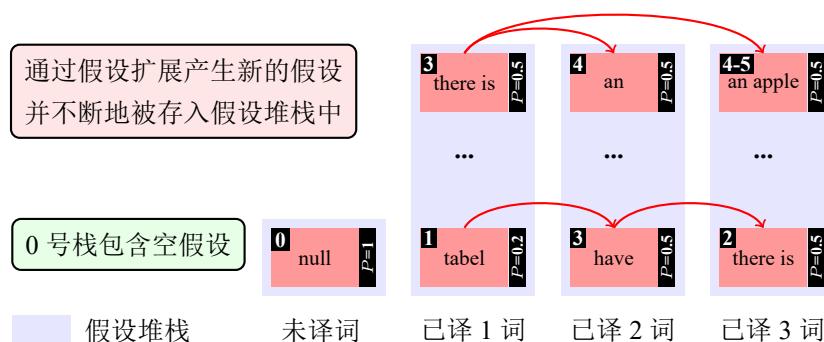


图 7.27 栈解码示例

<sup>4</sup>虽然被称作栈，实际上使用一个堆进行实现。这样可以根据模型得分对翻译假设进行排序。

## 7.8 小结及拓展阅读

统计机器翻译模型是近三十年内自然语言处理的重要里程碑之一。其统计建模的思想长期影响着自然语言处理的研究。无论是前面介绍的基于单词的模型，还是本章介绍的基于短语的模型，甚至后面即将介绍的基于句法的模型，大家都在尝试回答：究竟应该用什么样的知识对机器翻译进行统计建模？不过，这个问题至今还没有确定的答案。但是，显而易见，统计机器翻译为机器翻译的研究提供了一种范式，即让计算机用概率化的“知识”描述翻译问题。这些“知识”体现在统计模型的结构和参数中，并且可以从大量的双语和单语数据中自动学习。这种建模思想在今天的机器翻译研究中仍然随处可见。

本章对统计机器翻译中的基于短语的模型进行了介绍。可以说，基于短语的模型是机器翻译中最成功的机器翻译模型之一。其结构简单，而且翻译速度快，因此也被大量应用于机器翻译产品及服务中。此外，包括判别式模型、最小错误率训练、短语抽取等经典问题都是源自基于短语的模型。可是，基于短语的模型所涉及的非常丰富，很难通过一章的内容进行面面俱到的介绍。还有很多方向值得读者进一步了解：

- 基于短语的机器翻译的想法很早就出现了，比如直接对把机器翻译看作基于短语的生成问题<sup>[267, 279, 280]</sup>，或者单独对短语翻译进行建模，之后集成到基于单词的模型中<sup>[281, 282, 283]</sup>。现在，最通用的框架是 Koehn 等人提出的模型<sup>[284]</sup>，与其类似的还有 Zens 等人的工作<sup>[285, 286]</sup>。这类模型把短语翻译分解为短语学习问题和解码问题。因此，在随后相当长一段时间里，如何获取双语短语也是机器翻译领域的热点。比如，一些团队研究如何直接从双语句对中学习短语翻译，而不是通过简单的启发性规则进行短语抽取<sup>[287, 288]</sup>。也有研究者对短语边界的建模进行研究，以获得更高质量的短语，同时减小模型大小<sup>[289, 290, 291]</sup>。
- 调序是基于短语的模型中经典的问题之一。早期的模型都是词汇化的调序模型，这类模型把调序定义为短语之间的相对位置建模问题<sup>[268, 292, 293]</sup>。后来，也有一些工作使用判别式模型来集成更多的调序特征<sup>[266, 294, 295, 296]</sup>。实际上，除了基于短语的模型，调序也在基于句法的模型中被广泛讨论。因此，一些工作尝试将基于短语的调序模型集成到基于句法的机器翻译系统中<sup>[266, 297, 298, 299]</sup>。此外，也有研究者对不同的调序模型进行了系统化的对比和分析，可以作为相关研究的参考<sup>[300]</sup>。与在机器翻译系统中集成调序模型不同，预调序（Pre-ordering）也是一种解决调序问题的思路<sup>[301, 302, 303, 304]</sup>。机器翻译中的预调序是指将输入的源语言句子按目标语言的顺序进行排列，这样在翻译中就尽可能减少调序操作。这种方法大多依赖源语言的句法树进行调序的建模，不过它与机器翻译系统的耦合很小，因此很容易进行系统集成。
- 统计机器翻译中使用的栈解码方法源自 Tillmann 等人的工作<sup>[77]</sup>。这种方法在 Pharaoh<sup>[81]</sup>、Moses<sup>[80]</sup> 等开源系统中被成功的应用，在机器翻译领域产生了很大

的影响力。特别是，这种解码方法效率很高，因此在许多工业系统里也大量使用。对于栈解码也有很多改进工作，比如，早期的工作考虑剪枝或者限制调度范围以加快解码速度<sup>[76, 305, 306, 307]</sup>。随后，也有研究工作从解码算法和语言模型集成方式的角度对这类方法进行改进<sup>[308, 309, 310]</sup>。

- 统计机器翻译的成功很大程度上来自判别式模型引入任意特征的能力。因此，在统计机器翻译时代，很多工作都集中在新特征的设计上。比如，可以基于不同的统计特征和先验知识设计翻译特征<sup>[311, 312, 313]</sup>，也可以模仿分类任务设计大规模的稀疏特征<sup>[277]</sup>。模型训练和特征权重调优也是统计机器翻译中的重要问题，除了最小错误率训练，还有很多方法，比如，最大似然估计<sup>[10, 284]</sup>、判别式方法<sup>[314]</sup>、贝叶斯方法<sup>[315, 316]</sup>、最小风险训练<sup>[317, 318]</sup>、基于 Margin 的方法<sup>[312, 319]</sup>以及基于排序模型的方法（PRO）<sup>[278, 320]</sup>。实际上，统计机器翻译的训练和解码也存在不一致的问题，比如，特征值由双语数据上的极大似然估计得到（没有剪枝），而解码时却使用束剪枝，而且模型的目标是最大化机器翻译评价指标。对于这个问题也可以通过调整训练的目标函数进行缓解<sup>[321, 322]</sup>。
- 短语表是基于短语的系统中的重要模块。但是，简单地利用基于频次的方法估计得到的翻译概率无法很好地处理低频短语。这时就需要对短语表进行平滑<sup>[310, 323, 324, 325]</sup>。另一方面，随着数据量的增长和抽取短语长度的增大，短语表的体积会急剧膨胀，这也大大增加了系统的存储消耗，同时过大的短语表也会带来短语查询效率的下降。针对这个问题，很多工作尝试对短语表进行压缩。一种思路是限制短语的长度<sup>[326, 327]</sup>；另一种广泛使用的思路是使用一些指标或者分类器来对短语进行剪枝，其核心思想是判断每个短语的质量<sup>[328]</sup>，并过滤掉低质量的短语。代表性的方法有：基于假设检验的剪枝<sup>[329]</sup>、基于熵的剪枝<sup>[330]</sup>、两阶段短语抽取方法<sup>[331]</sup>、基于解码中短语使用频率的方法<sup>[332]</sup>等。此外，短语表的存储方式也是在实际使用中需要考虑的问题。因此，也有研究者尝试使用更加紧凑、高效的结构保存短语表。其中最具代表性的结构是后缀数组（Suffix Arrays），这种结构可以充分利用短语之间有重叠的性质，减少了重复存储<sup>[333, 334, 335]</sup>。





## 8. 基于句法的模型

人类的语言是具有结构的，这种结构往往体现在句子的句法信息上。比如，人们进行翻译时会将待翻译句子的主干确定下来，之后得到译文的主干，最后形成完整的译文。一个人学习外语时，也会先学习外语句子的基本构成，比如，主语、谓语等，之后用这种句子结构知识生成外语句子。

使用句法分析可以很好地处理翻译中的结构调序、远距离依赖等问题。因此，基于句法的机器翻译模型长期受到研究者关注。比如，早期基于规则的方法里就大量使用了句法信息来定义翻译规则。进入统计机器翻译时代，句法信息的使用同样是主要研究方向之一。这也产生了很多基于句法的机器翻译模型及方法，而且在很多任务上取得非常出色的结果。本章将对这些模型和方法进行介绍，内容涉及机器翻译中句法信息的表示、基于句法的翻译建模、句法翻译规则的学习等。

### 8.1 翻译中句法信息的使用

使用短语的优点在于可以捕捉到具有完整意思的连续词串，因此能够对局部上下文信息进行建模。当单词之间的搭配和依赖关系出现在连续词串中时，短语可以很好地对其进行描述。但是，当单词之间距离很远时，使用短语的“效率”很低。同  $n$ -gram 语言模型一样，当短语长度变长时，数据会变得非常稀疏。比如，很多实验已经证明，如果在测试数据中有一个超过 5 个单词的连续词串，那么它在训练数据中往往是很低频的现象，更长的短语甚至都很难在训练数据中找到。

当然，可以使用平滑算法对长短语的概率进行估计，但是使用过长的短语在实

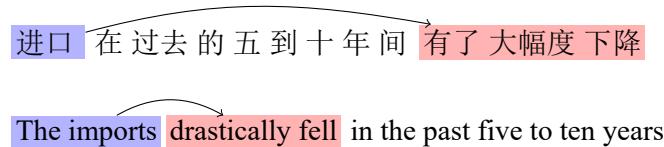


图 8.1 汉英翻译中的不同距离下的依赖

际系统研发中仍然不现实。图8.1展示了一个汉语到英语的翻译实例。源语言的两个短语（蓝色和红色高亮）在目标语言中产生了调序。但是，这两个短语在源语言句子中横跨 11 个单词。如果直接使用这 11 个单词构成的短语进行翻译，显然会有非常严重的数据稀疏问题，因为很难期望在训练数据中见到一模一样的短语。

仅使用连续词串不能处理所有的翻译问题，其根本原因在于句子的表层串很难描述片段之间大范围的依赖。一个新的思路是使用句子的层次结构信息进行建模。第三章已经介绍了句法分析基础。对于每个句子，都可以用句法树描述它的结构。

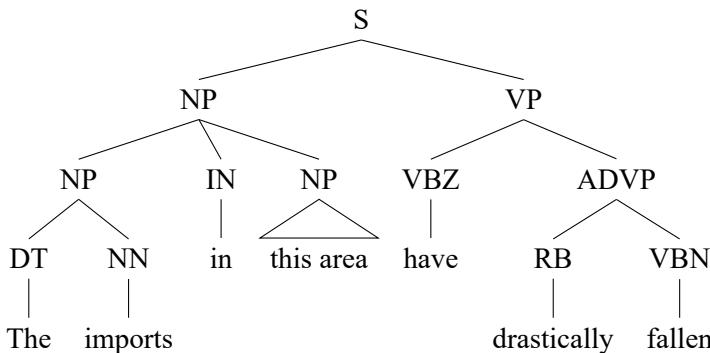


图 8.2 一棵英语句法树（短语结构树）

图8.2就展示了一棵英语句法树（短语结构树）。句法树描述了一种递归的结构，每个句法结构都可以用一个子树来描述，子树之间的组合可以构成更大的子树，最终完成整个句子的表示。相比线性的序列结构，树结构更容易处理大片段之间的关系。比如，两个在序列中距离“很远”的单词，在树结构中可能会“很近”。

句法树结构可以赋予机器翻译对语言进一步抽象的能力，这样，可以不需要使用连续词串，而是通过句法结构来对大范围的译文生成和调序进行建模。图8.3是一个在翻译中融入源语言（汉语）句法信息的实例。这个例子中，介词短语“在 ... 后”包含 12 个单词，因此，使用短语很难涵盖这样的片段。这时，系统会把“在 ... 后”错误地翻译为“In ...”。通过句法树，可以知道“在 ... 后”对应着一个完整的子树结构 PP（介词短语）。因此也很容易知道介词短语中“在 ... 后”是一个模板（红色），而“在”和“后”之间的部分构成从句部分（蓝色）。最终得到正确的译文“After ...”。

使用句法信息在机器翻译中并不新鲜。在基于规则和模板的翻译模型中，就大

**人工翻译:** After the school team won the Championship of the China University Basketball Association for the first time ...

**机器翻译:** In the school team won the Chinese College Basketball League Championship for the first time ...

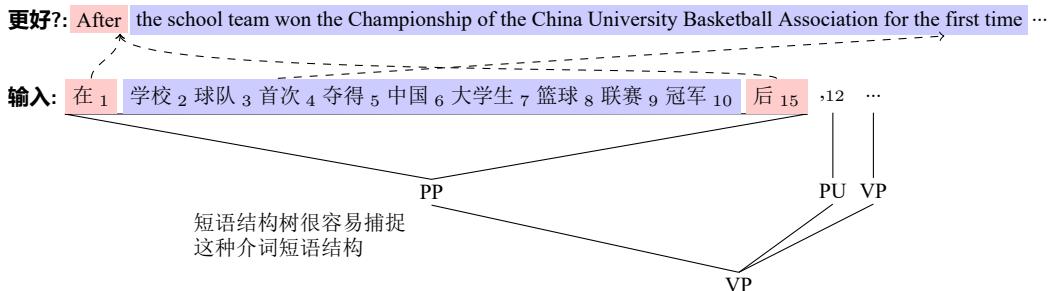


图 8.3 使用句法结构进行机器翻译的实例

量使用了句法等结构信息。只是由于早期句法分析技术不成熟，系统的整体效果并不突出。在数据驱动的方法中，句法可以很好地融合在统计建模中。通过概率化的句法设计，可以对翻译过程进行很好的描述。

## 8.2 基于层次短语的模型

在机器翻译中，如果翻译需要局部上下文的信息，把短语作为翻译单元是一种理想的方案。但是，单词之间的关系并不总是“局部”的，很多时候需要距离更远一些的搭配。比较典型的例子是含有从句的情况。比如：

我在今天早上没有吃早饭的情况下还是正常去上班了。

这句话的主语“我”和谓语“去上班”构成了主谓搭配，而二者之间的部分是状语。显然，用短语去捕捉这个搭配需要覆盖很长的词串，也就是整个“我...去上班”的部分。如果把这样的短语考虑到建模中，会面临非常严重的数据稀疏问题，因为无法保证这么长的词串在训练数据中能够出现。

表 8.1 不同短语在训练数据中出现的频次

短语（中文）	训练数据中出现的频次
包含	3341
包含多个	213
包含多个词	12
包含多个词的	8
包含多个词的短语	0
包含多个词的短语太多	0

实际上，随着短语长度变长，短语在数据中会变得越来越低频，相关的统计特

征也会越来越不可靠。表8.1就展示了不同长度的短语在一个训练数据中出现的频次。可以看到，长度超过3的短语已经非常低频了，更长的短语甚至在训练数据中一次也没有出现过。

显然，利用过长的短语来处理长距离的依赖并不是一种十分有效的方法。过于低频的长短语无法提供可靠的信息，而且使用长短语会导致模型体积急剧增加。

再来看一个翻译实例，图8.4是一个基于短语的机器翻译系统的翻译结果。这个例子中的调序有一些复杂，比如，“众多/高校/之一”和“与/东软/有/合作”的英文翻译都需要进行调序，分别是“one of the many universities”和“have collaborative relations with Neusoft”。基于短语的系统可以很好地处理这些调序问题，因为它们仅仅使用了局部的信息。但是，系统却无法在这两个短语（1和2）之间进行正确的调序。

源语言句子：东北大学 是 与 东软 有 合作 的 众多 高校 之一

系统输出：NEU is collaborative relations with Neusoft 1  
is one of the many universities 2

参考译文：NEU is one of the many universities that have  
collaborative relations with Neusoft

图 8.4 基于短语的机器翻译实例

这个例子也在一定程度上说明了长距离的调序需要额外的机制才能得到更好的处理。实际上，两个短语（1和2）之间的调序现象本身对应了一种结构，或者说模板。也就是汉语中的：

与 [什么 东西] 有 [什么 事]

可以翻译成：

have [什么 事] with [什么 东西]

这里[什么 东西]和[什么 事]表示模板中的变量，可以被其他词序列替换。通常，可以把这个模板形式化描述为：

$\langle \text{与 } X_1 \text{ 有 } X_2, \text{ have } X_2 \text{ with } X_1 \rangle$

其中，逗号分隔了源语言和目标语言部分， $X_1$ 和 $X_2$ 表示模板中需要替换的内容，或者说变量。源语言中的变量和目标语言中的变量是一一对应的，比如，源语言中的

$X_1$  和目标语言中的  $X_1$  代表这两个变量可以“同时”被替换。假设给定短语对：

〈东软， Neusoft 〉  
〈合作， collaborative relations 〉

可以使用第一个短语替换模板中的变量  $X_1$ ，得到：

〈与 [东软] 有  $X_2$ ， have  $X_2$  with [Neusoft] 〉

其中，[.] 表示被替换的部分。可以看到，在源语言和目标语言中， $X_1$  被同时替换为相应的短语。进一步，可以用第二个短语替换  $X_2$ ，得到：

〈与 东软 有 [合作]， have [collaborative relations] with Neusoft 〉

至此，就得到了一个完整词串的译文。类似的，还可以写出其他的翻译模板，如下：

〈 $X_1$  是  $X_2$ ，  $X_1$  is  $X_2$  〉  
〈 $X_1$  之一， one of  $X_1$  〉  
〈 $X_1$  的  $X_2$ ，  $X_2$  that have  $X_1$  〉

使用上面这种变量替换的方式，就可以得到一个完整句子的翻译。

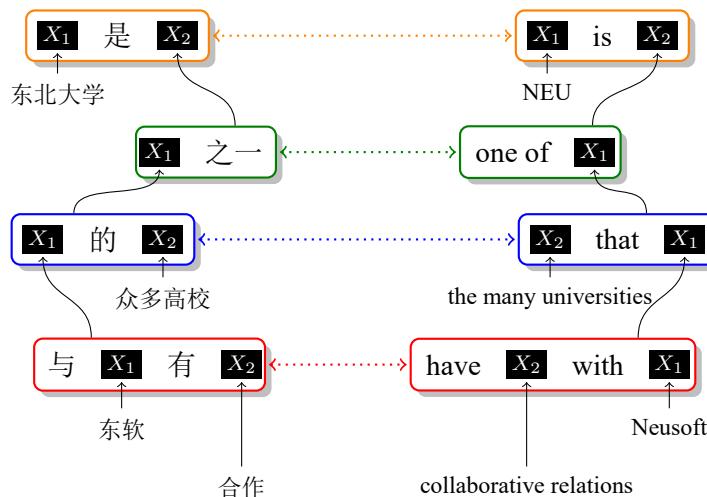


图 8.5 使用短语和翻译模板进行双语句子的同步生成

这个过程如图8.5所示。其中，左右相连接的方框表示翻译模版的源语言和目标语言部分。可以看到，模版中两种语言中的变量会被同步替换，替换的内容可以是

其他模版生成的结果。这也就对应了一种层次结构，或者说互译的句对可以被双语的层次结构同步生成出来。

实际上，在翻译中使用这样的模版就构成了层次短语模型的基本思想。下面就一起看看如何对翻译模版进行建模，以及如何自动学习并使用这些模版。

### 8.2.1 同步上下文无关文法

**基于层次短语的模型** (Hierarchical Phrase-based Model) 是一个经典的统计机器翻译模型<sup>[88, 336]</sup>。这个模型可以很好地解决短语系统对翻译中长距离调序建模不足的问题。基于层次短语的系统也在多项机器翻译比赛中取得了很好的成绩。这项工作也获得了自然语言处理领域顶级会议 ACL2015 的最佳论文奖。

层次短语模型的核心是把翻译问题归结为两种语言词串的同步生成问题。实际上，词串的生成问题是自然语言处理中的经典问题，早期的研究更多的是关注单语句子的生成，比如，如何使用句法树描述一个句子的生成过程。层次短语模型的创新之处是把传统单语词串的生成推广到双语词串的同步生成上。这使得机器翻译可以使用类似句法分析的方法进行求解。

#### 1. 文法定义

层次短语模型中一个重要的概念是**同步上下文无关文法** (Synchronous Context-free Grammar, SCFG)。SCFG 可以被看作是对源语言和目标语言上下文无关文法的融合，它要求源语言和目标语言的产生式及产生式中的变量具有对应关系。具体定义如下：

##### 定义 8.2.1 同步上下文无关文法

一个同步上下文无关文法由五部分构成  $(N, T_s, T_t, I, R)$ ，其中：

1.  $N$  是非终结符集合。
2.  $T_s$  和  $T_t$  分别是源语言和目标语言的终结符集合。
3.  $I \subseteq N$  是起始非终结符集合。
4.  $R$  是规则集合，每条规则  $r \in R$  有如下形式：

$$\text{LHS} \rightarrow <\alpha, \beta, \sim>$$

其中， $\text{LHS} \in N$  表示规则的左部，它是一个非终结符；规则的右部由三部分组成， $\alpha \in (N \cup T_s)^*$  表示由源语言终结符和非终结符组成的串； $\beta \in (N \cup T_t)^*$  表示由目标语言终结符和非终结符组成的串； $\sim$  表示  $\alpha$  和  $\beta$  中非终结符的 1-1 对应关系。

根据这个定义，源语言和目标语言有不同的终结符集合（单词），但是它们会共享同一个非终结符集合（变量）。每个产生式包括源语言和目标语言两个部分，分别表示由规则左部生成的源语言和目标语言符号串。由于产生式会同时生成两种语言的符号串，因此这是一种“同步”生成，可以很好地描述翻译中两个词串之间的对应。

下面是一个简单的 SCFG 实例：

$$\begin{aligned} S &\rightarrow \langle NP_1 \text{ 希望 } VP_2, \ NP_1 \text{ wish to } VP_2 \rangle \\ VP &\rightarrow \langle \text{对 } NP_1 \text{ 感到 } VP_2, \ be \ VP_2 \ wish \ NP_1 \rangle \\ NN &\rightarrow \langle \text{强大}, \ strong \rangle \end{aligned}$$

这里的 S、NP、VP 等符号可以被看作是具有句法功能的标记，因此这个文法和传统句法分析中的 CFG 很像，只是 CFG 是单语文法，而 SCFG 是双语同步文法。非终结符的下标表示对应关系，比如，源语言的  $NP_1$  和目标语言的  $NP_1$  是对应的。因此，在上面这种表示形式中，两种语言间非终结符的对应关系～是隐含在变量下标中的。当然，复杂的句法功能标记并不是必须的。比如，也可以使用更简单的文法形式：

$$\begin{aligned} X &\rightarrow \langle X_1 \text{ 希望 } X_2, \ X_1 \text{ wish to } X_2 \rangle \\ X &\rightarrow \langle \text{对 } X_1 \text{ 感到 } X_2, \ be \ X_2 \ wish \ X_1 \rangle \\ X &\rightarrow \langle \text{强大}, \ strong \rangle \end{aligned}$$

这个文法只有一种非终结符  $X$ ，因此所有的变量都可以使用任意的产生式进行推导。这就给翻译提供了更大的自由度，也就是说，规则可以被任意使用，进行自由组合。这也符合基于短语的模型中对短语进行灵活拼接的思想。基于此，层次短语系统中也使用这种并不依赖语言学句法标记的文法。在本章的内容中，如果没有特殊说明，把这种没有语言学句法标记的文法称作**基于层次短语的文法**（Hierarchical Phrase-based Grammar），或简称层次短语文法。

## 2. 推导

下面是一个完整的层次短语文法：

$$\begin{aligned} r_1: \quad X &\rightarrow \langle \text{进口 } X_1, \ The \ imports \ X_1 \rangle \\ r_2: \quad X &\rightarrow \langle X_1 \text{ 下降 } X_2, \ X_2 \ X_1 \ fallen \rangle \\ r_3: \quad X &\rightarrow \langle \text{大幅度}, \ drastically \rangle \\ r_4: \quad X &\rightarrow \langle \text{了}, \ have \rangle \end{aligned}$$

其中，规则  $r_1$  和  $r_2$  是含有变量的规则，这些变量可以被其他规则的右部替换；规则  $r_2$  是调序规则；规则  $r_3$  和  $r_4$  是纯词汇化规则，表示单词或者短语的翻译。

对于一个双语句对：

**源语言：** 进口 大幅度 下降 了

**目标语言：** The imports have drastically fallen

可以进行如下的推导（假设起始符号是 X）：

$$\begin{aligned}
 & \langle X_1, X_1 \rangle \\
 \xrightarrow{r_1} & \langle \text{进口 } X_2, \text{The imports } X_2 \rangle \\
 \xrightarrow{r_2} & \langle \text{进口 } X_3 \text{ 下降 } X_4, \text{The imports } X_4 X_3 \text{ fallen} \rangle \\
 \xrightarrow{r_3} & \langle \text{进口 大幅度 } \text{下降 } X_4, \\
 & \quad \text{The imports } X_4 \text{ drastically fallen} \rangle \\
 \xrightarrow{r_4} & \langle \text{进口 大幅度 } \text{下降 } \text{了}, \\
 & \quad \text{The imports have drastically fallen} \rangle
 \end{aligned}$$

其中，每使用一次规则就会同步替换源语言和目标语言符号串中的一个非终结符，替换结果用红色表示。通常，可以把上面这个过程称作翻译推导，记为：

$$d = r_1 \circ r_2 \circ r_3 \circ r_4 \quad (8.1)$$

在层次短语模型中，每个翻译推导都唯一地对应一个目标语译文。因此，可以用推导的概率  $P(d)$  描述翻译的好坏。同基于短语的模型是一样的（见第七章数学建模小节），层次短语翻译的目标是：求概率最高的翻译推导  $\hat{d} = \arg \max P(d)$ 。值得注意的是，基于推导的方法在句法分析中也十分常用。层次短语翻译实质上也是通过生成翻译规则的推导来对问题的表示空间进行建模。在8.3节还将看到，这种方法可以被扩展到语言学上基于句法的翻译模型中。而且这些模型都可以用一种被称作超图的结构来进行建模。从某种意义上讲，基于规则推导的方法将句法分析和机器翻译进行了形式上的统一。因此机器翻译也借用了很多句法分析的思想。

### 3. 胶水规则

由于翻译现象非常复杂，在实际系统中往往需要把两个局部翻译线性拼接到一起。在层次短语模型中，这个问题通过引入**胶水规则**（Glue Rule）来处理，形式如下：

$$\begin{aligned}
 S & \rightarrow \langle S_1 X_2, S_1 X_2 \rangle \\
 S & \rightarrow \langle X_1, X_1 \rangle
 \end{aligned}$$

胶水规则引入了一个新的非终结符 S，S 只能和 X 进行顺序拼接，或者 S 由 X

生成。如果把  $S$  看作文法的起始符，使用胶水规则后，相当于把句子划分为若干个部分，每个部分都被归纳为  $X$ 。之后，顺序地把这些  $X$  拼接到一起，得到最终的译文。比如，最极端的情况，整个句子会生成一个  $X$ ，之后再归纳为  $S$ ，这时并不需要进行胶水规则的顺序拼接；另一种极端的情况，每个单词都是独立地被翻译，被归纳为  $X$ ，之后先把最左边的  $X$  归纳为  $S$ ，再依次把剩下的  $X$  依次拼到一起。这样的推导形式如下：

$$\begin{aligned} S &\rightarrow \langle S_1 X_2, S_1 X_2 \rangle \\ &\rightarrow \langle S_3 X_4 X_2, S_3 X_4 X_2 \rangle \\ &\rightarrow \dots \\ &\rightarrow \langle X_n \dots X_4 X_2, X_n \dots X_4 X_2 \rangle \end{aligned}$$

实际上，胶水规则在很大程度上模拟了基于短语的系统中对字符串顺序翻译的操作，而且在实践中发现，这个步骤是十分必要的。特别是对法英翻译这样的任务，由于语言的结构基本上是顺序翻译的，因此引入顺序拼接的操作符合翻译的整体规律。同时，这种拼接给翻译增加了灵活性，系统会更加健壮。

需要说明的是，使用同步文法进行翻译时，由于单词的顺序是内嵌在翻译规则内的，因此这种模型并不依赖额外的调序模型。一旦文法确定下来，系统就可以进行翻译。

#### 4. 处理流程

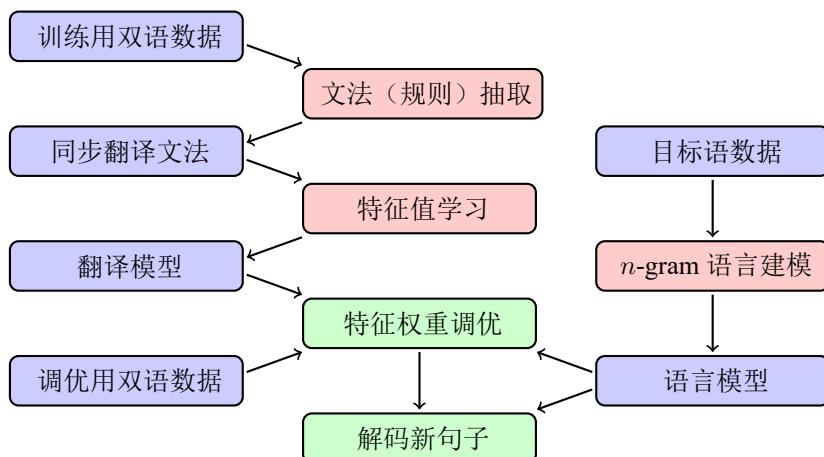


图 8.6 层次短语系统的处理流程

层次短语系统的流程如图8.6所示。其核心是从双语数据中学习同步翻译文法，并进行翻译特征的学习，形成翻译模型（即规则 + 特征）。同时，要从目标语言数据

中学习语言模型。最终，把翻译模型和语言模型一起送入解码器，在特征权重调优后，完成对新输入句子的翻译。

### 8.2.2 层次短语规则抽取

层次短语系统所使用的文法包括两部分：

- 不含变量的层次短语规则（短语翻译）；
- 含有变量的层次短语规则。短语翻译的抽取直接复用基于短语的系统即可。

此处重点讨论如何抽取含有变量的层次短语规则。在第七章短语抽取一节已经介绍了短语与词对齐相兼容的概念。这里，所有层次短语规则也是与词对齐相兼容（一致）的。

#### 定义 8.2.2 与词对齐相兼容的层次短语规则

对于句对  $(s, t)$  和它们之间的词对齐  $a$ ，令  $\Phi$  表示在句对  $(s, t)$  上与  $a$  相兼容的双语短语集合。则：

- 如果  $(x, y) \in \Phi$ ，则  $X \rightarrow \langle x, y, \Phi \rangle$  是与词对齐相兼容的层次短语规则。
- 对于  $(x, y) \in \Phi$ ，存在  $m$  个双语短语  $(x_i, y_j) \in \Phi$ ，同时存在  $(1, \dots, m)$  上面的一个排序  $\sim = \{\pi_1, \dots, \pi_m\}$ ，且：

$$x = \alpha_0 x_1 \dots \alpha_{m-1} x_m \alpha_m \quad (8.2)$$

$$y = \beta_0 y_{\pi_1} \dots \beta_{m-1} y_{\pi_m} \beta_m \quad (8.3)$$

其中， $\{\alpha_0, \dots, \alpha_m\}$  和  $\{\beta_0, \dots, \beta_m\}$  表示源语言和目标语言的若干个词串（包含空串）。则  $X \rightarrow \langle x, y, \sim \rangle$  是与词对齐相兼容的层次短语规则。这条规则包含  $m$  个变量，变量的对齐信息是  $\sim$ 。

这个定义中，所有规则都是由双语短语生成。如果规则中含有变量，则变量部分也需要满足与词对齐相兼容的定义。按上述定义实现层次短语规则抽取也很简单。只需要对短语抽取系统进行改造：对于一个短语，可以通过挖“槽”的方式生成含有变量的规则。每个“槽”就代表一个变量。

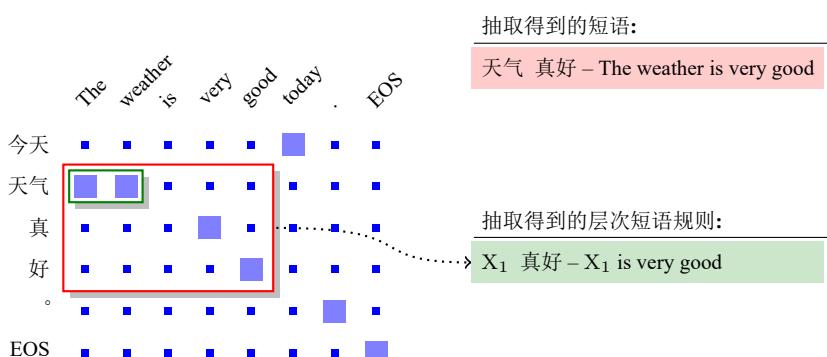


图 8.7 通过双语短语抽取层次短语规则

图8.7展示了一个通过双语短语抽取层次短语的示意图。可以看到，在获取一个“大”短语的基础上（红色），直接在其内部抽取得到另一个“小”短语（绿色），这样就生成了一个层次短语规则。

这种方式可以抽出大量的层次短语规则。但是，不加限制的抽取会带来规则集合的过度膨胀，对解码系统造成很大负担。比如，如果考虑任意长度的短语会使得层次短语规则过大，一方面这些规则很难在测试数据上被匹配，另一方面抽取这样的“长”规则会使得抽取算法变慢，而且规则数量猛增之后难以存储。还有，如果一个层次短语规则中含有过多的变量，也会导致解码算法变得更加复杂，不利于系统实现和调试。针对这些问题，在标准的层次短语系统中会考虑一些限制<sup>[336]</sup>，包括：

- 抽取的规则最多可以跨越 10 个词；
- 规则的（源语言端）变量个数不能超过 2；
- 规则的（源语言端）变量不能连续出现。

在具体实现时还会考虑其他的限制，比如，限定规则的源语言端终结符数量的上限等。

### 8.2.3 翻译特征

在层次短语模型中，每个翻译推导都有一个模型得分  $\text{score}(d, t, s)$ 。 $\text{score}(d, t, s)$  是若干特征的线性加权之和： $\text{score}(d, t, s) = \sum_{i=1}^M \lambda_i \cdot h_i(d, t, s)$ ，其中  $\lambda_i$  是特征权重， $h_i(d, t, s)$  是特征函数。层次短语模型的特征包括与规则相关的特征和语言模型特征，如下：

对于每一条翻译规则  $\text{LHS} \rightarrow \langle \alpha, \beta, \sim \rangle$ ，有：

- $(h_{1-2})$  短语翻译概率（取对数），即  $\log(P(\alpha | \beta))$  和  $\log(P(\beta | \alpha))$ ，特征的计算与基于短语的模型完全一样；
- $(h_{3-4})$  词汇化翻译概率（取对数），即  $\log(P_{\text{lex}}(\alpha | \beta))$  和  $\log(P_{\text{lex}}(\beta | \alpha))$ ，特征的计算与基于短语的模型完全一样；
- $(h_5)$  翻译规则数量，让模型自动学习对规则数量的偏好，同时避免使用过少规则造成分数偏高的现象；
- $(h_6)$  胶水规则数量，让模型自动学习使用胶水规则的偏好；
- $(h_7)$  短语规则数量，让模型自动学习使用纯短语规则的偏好。

这些特征可以被具体描述为：

$$h_i(d, t, s) = \sum_{r \in d} h_i(r) \quad (8.4)$$

公式(8.4)中， $r$  表示推导  $d$  中的一条规则， $h_i(r)$  表示规则  $r$  上的第  $i$  个特征。可

以看出，推导  $d$  的特征值就是所有包含在  $d$  中规则的特征值的和。进一步，可以定义

$$\text{rscore}(d, t, s) = \sum_{i=1}^7 \lambda_i \cdot h_i(d, t, s) \quad (8.5)$$

最终，模型得分被定义为：

$$\text{score}(d, t, s) = \text{rscore}(d, t, s) + \lambda_8 \log(P_{\text{lm}}(t)) + \lambda_9 |t| \quad (8.6)$$

其中：

- $\log(P_{\text{lm}}(t))$  表示语言模型得分；
- $|t|$  表示译文的长度。

在定义特征函数之后，特征权重  $\{\lambda_i\}$  可以通过最小错误率训练在开发集上进行调优。关于最小错误率训练方法可以参考第七章的相关内容。

#### 8.2.4 CKY 解码

层次短语模型解码的目标是找到模型得分最高的推导，即：

$$\hat{d} = \arg \max_d \text{score}(d, t, s) \quad (8.7)$$

这里， $\hat{d}$  的目标语部分即最佳译文  $\hat{t}$ 。令函数  $t(\cdot)$  返回翻译推导的目标语词串，于是有：

$$\hat{t} = t(\hat{d}) \quad (8.8)$$

由于层次短语规则本质上就是 CFG 规则，因此公式(8.7)代表了一个典型的句法分析过程。需要做的是，用模型源语言端的 CFG 对输入句子进行分析，同时用模型目标语言端的 CFG 生成译文。基于 CFG 的句法分析是自然语言处理中的经典问题。一种广泛使用的方法是：首先把 CFG 转化为  $\epsilon$ -free 的**乔姆斯基范式**（Chomsky Normal Form）<sup>5</sup>，之后采用 CKY 方法进行分析。

CKY 是形式语言中一种常用的句法分析方法<sup>[337, 338, 339]</sup>。它主要用于分析符合乔姆斯基范式的句子。由于乔姆斯基范式中每个规则最多包含两叉（或者说两个变量），因此 CKY 方法也可以被看作是基于二叉规则的一种分析方法。对于一个待分析的字符串，CKY 方法从小的“范围”开始，不断扩大分析的“范围”，最终完成对整个字符串的分析。在 CKY 方法中，一个重要的概念是**跨度**（Span），所谓跨度表示了一

<sup>5</sup>能够证明任意的 CFG 都可以被转换为乔姆斯基范式，即文法只包含形如  $A \rightarrow BC$  或  $A \rightarrow a$  的规则。这里，假设文法中不包含空串产生式  $A \rightarrow \epsilon$ ，其中  $\epsilon$  表示空字符串。

一个符号串的范围。这里可以把跨度简单地理解为从一个起始位置到一个结束位置中间的部分。

猫	喜欢	吃	鱼	
0	1	2	3	4

图 8.8 一个单词串及位置索引

比如, 如图8.8 所示, 每个单词左右都有一个数字来表示序号。可以用序号的范围来表示跨度, 例如:

$$\begin{aligned}span[0,1] &= \text{“猫”} \\span[2,4] &= \text{“吃 鱼”} \\span[0,4] &= \text{“猫 喜欢 吃 鱼”}\end{aligned}$$

CKY 方法是按跨度由小到大的次序执行的, 这也对应了一种**自下而上的分析**(Top-Down Parsing) 过程。对于每个跨度, 检查:

- 是否有形如  $A \rightarrow a$  的规则可以匹配;
- 是否有形如  $A \rightarrow BC$  的规则可以匹配。

对于第一种情况, 简单匹配字符串即可; 对于第二种情况, 需要把当前的跨度进一步分割为两部分, 并检查左半部分是否已经被归纳为 B, 右半部分是否已经被归纳为 C。如果可以匹配, 会在这个跨度上保存匹配结果。后面, 可以访问这个结果(也就是 A) 来生成更大跨度上的分析结果。

输入: 符合乔姆斯基范式的待分析字符串和一个上下文无关文法 (CFG)  
输出: 全部可能的字符串语法分析结果

参数:  $s$  为输入字符串。 $G$  为输入 CFG。 $J$  为待分析字符串长度。

```
Function CKY-Algorithm( $s, G$ )
  for  $j = 0$  to  $J - 1$ 
     $span[j, j + 1].Add(A \rightarrow a \in G)$ 
  for  $l = 1$  to  $J$            // 跨度长度
    for  $j = 0$  to  $J - l$       // 跨度起始位置
      for  $k = j$  to  $j + l$     // 跨度结束位置
         $hypos = Compose(span[j, k], span[k, j + l])$ 
         $span[j, j + l].Update(hypos)$ 
  return  $span[0, J]$ 
```

图 8.9 CKY 算法

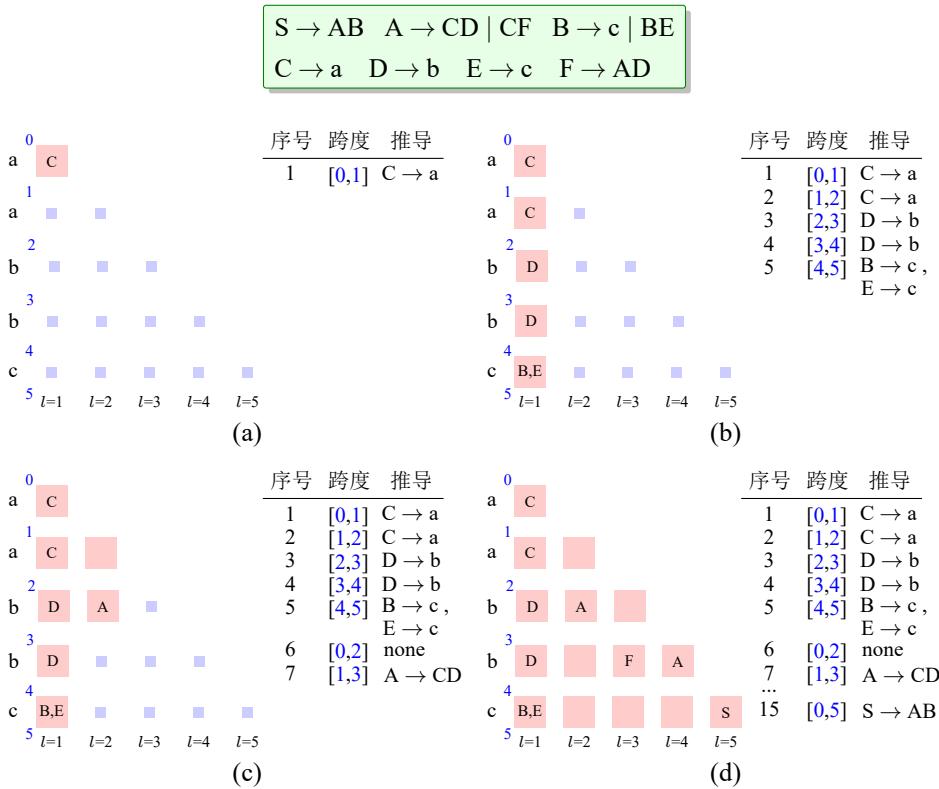


图 8.10 CKY 算法执行实例

CKY 算法的伪代码如图8.9所示。整个算法的执行顺序是按跨度的长度 ( $l$ ) 组织的。对于每个  $span[j, j+l]$ , 会在位置  $k$  进行切割。之后, 判断  $span[j, k]$  和  $span[k, j+l]$  是否可以形成一个规则的右部。也就是判断  $span[j, k]$  是否生成了 B, 同时判断  $span[k, j+l]$  是否生成了 C, 如果文法中有规则  $A \rightarrow BC$ , 则把这个规则放入  $span[j, j+l]$ 。这个过程由 Compose 函数完成。如果  $span[j, j+l]$  可以匹配多条规则, 所有生成的推导都会被记录在  $span[j, j+l]$  所对应的一个列表里<sup>6</sup>。

图8.10展示了 CKY 方法的一个运行实例 (输入词串是 aabbc)。算法在处理完最后一个跨度后会得到覆盖整个词串的分析结果, 即句法树的根结点 S。

不过, CKY 方法并不能直接用于层次短语模型, 主要有两个问题:

- 层次短语模型的文法不符合乔姆斯基范式;
- 机器翻译中需要语言模型。由于计算当前词的语言模型得分需要前面的词做条件, 因此机器翻译的解码过程并不是上下文无关的。

解决第一个问题有两个思路:

- 把层次短语文法转化为乔姆斯基范式, 这样可以直接使用原始的 CKY 方法进

<sup>6</sup>通常, 这个列表会用优先队列实现。这样可以对推导按模型得分进行排序, 方便后续的剪枝操作。

行分析；

- 对 CKY 方法进行改造。解码的核心任务要知道每个跨度是否能匹配规则的源语言部分。实际上，层次短语模型的文法是一种特殊的文法。这种文法规则的源语言部分最多包含两个变量，而且变量不能连续。这样的规则会对应一种特定类型的模版，比如，对于包含两个变量的规则，它的源语言部分形如  $\alpha_0 X_1 \alpha_1 X_2 \alpha_2$ 。其中， $\alpha_0$ 、 $\alpha_1$  和  $\alpha_2$  表示终结符串， $X_1$  和  $X_2$  是变量。显然，如果  $\alpha_0$ 、 $\alpha_1$  和  $\alpha_2$  确定下来那么  $X_1$  和  $X_2$  的位置也就确定了下来。因此，对于每一个词串，都可以很容易的生成这种模版，进而完成匹配。而  $X_1$  和  $X_2$  和原始 CKY 中匹配二叉规则本质上是一样的。由于这种方法并不需要对 CKY 方法进行过多调整，因此层次短语系统中广泛使用这种改造的 CKY 方法进行解码。

对于语言模型在解码中的集成问题，一种简单的办法是：在 CKY 分析的过程中，用语言模型对每个局部的翻译结果进行评价，并计算局部翻译（推导）的模型得分。注意，局部的语言模型得分可能是不准确的，比如，局部翻译片段最左边单词的概率计算需要依赖前面的单词。但是由于每个跨度下生成的翻译是局部的，当前跨度下看不到前面的译文。这时会用 1-gram 语言模型的得分代替真实的高阶语言模型得分。等这个局部翻译片段和其他片段组合之后，可以知道前文的内容，这时才会得出最终的语言模型得分。

另一种解决问题的思路是，先不加入语言模型，这样可以直接使用 CKY 方法进行分析。在得到最终的结果后，对最好的多个推导用含有语言模型的完整模型进行打分，选出最终的最优推导。

不过，在实践中发现，由于语言模型在机器翻译中起到至关重要的作用，因此对最终结果进行重排序会带来一定的性能损失。不过这种方法的优势在于速度快，而且容易实现。另外，在实践时，还需要考虑两方面问题：

- **剪枝：**在 CKY 中，每个跨度都可以生成非常多的推导（局部翻译假设）。理论上，这些推导的数量会和跨度大小成指数关系。显然不可能保存如此大量的翻译推导。对于这个问题，常用的办法是只保留 top- $k$  个推导。也就是每个局部结果只保留最好的  $k$  个，即束剪枝。在极端情况下，当  $k=1$  时，这个方法就变成了贪婪的方法；
- **$n$ -best 结果的生成：** $n$ -best 推导（译文）的生成是统计机器翻译必要的功能。比如，最小错误率训练中就需要最好的  $n$  个结果用于特征权重调优。在基于 CKY 的方法中，整个句子的翻译结果会被保存在最大跨度所对应的结构中。因此一种简单的  $n$ -best 生成方法是从这个结构中取出排名最靠前的  $n$  个结果。另外，也可以考虑自上而下遍历 CKY 生成的推导空间，得到更好的  $n$ -best 结果<sup>[340]</sup>。

### 8.2.5 立方剪枝

相比于基于短语的模型，基于层次短语的模型引入了“变量”的概念。这样，可以根据变量周围的上下文信息对变量进行调序。变量的内容由其所对应的跨度上的翻译假设进行填充。图8.11展示了一个层次短语规则匹配词串的实例。可以看到，规则匹配词串之后，变量  $X$  的位置对应了一个跨度。这个跨度上所有标记为  $X$  的局部推导都可以作为变量的内容。

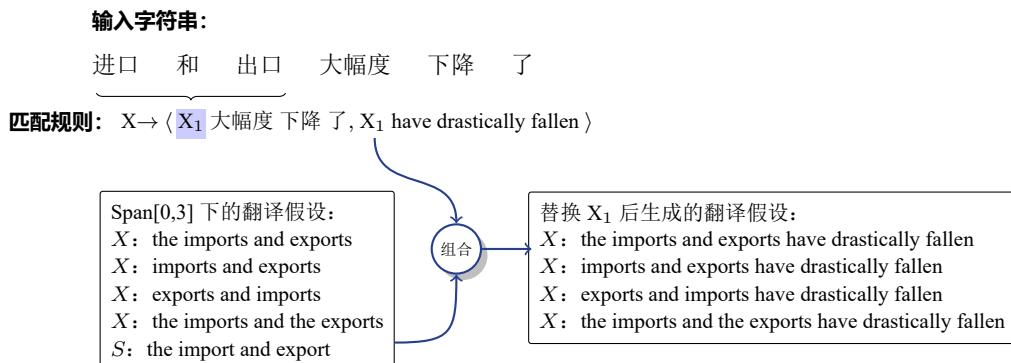


图 8.11 层次短语规则匹配及译文生成

真实的情况会更加复杂。对于一个规则的源语言端，可能会有多个不同的目标语言端与之对应。比如，如下规则的源语言端完全相同，但是译文不同：

$$\begin{aligned} X &\rightarrow \langle X_1 \text{ 大幅度 } \text{下降 } \text{了}, X_1 \text{ have drastically fallen} \rangle \\ X &\rightarrow \langle X_1 \text{ 大幅度 } \text{下降 } \text{了}, X_1 \text{ have fallen drastically} \rangle \\ X &\rightarrow \langle X_1 \text{ 大幅度 } \text{下降 } \text{了}, X_1 \text{ has drastically fallen} \rangle \end{aligned}$$

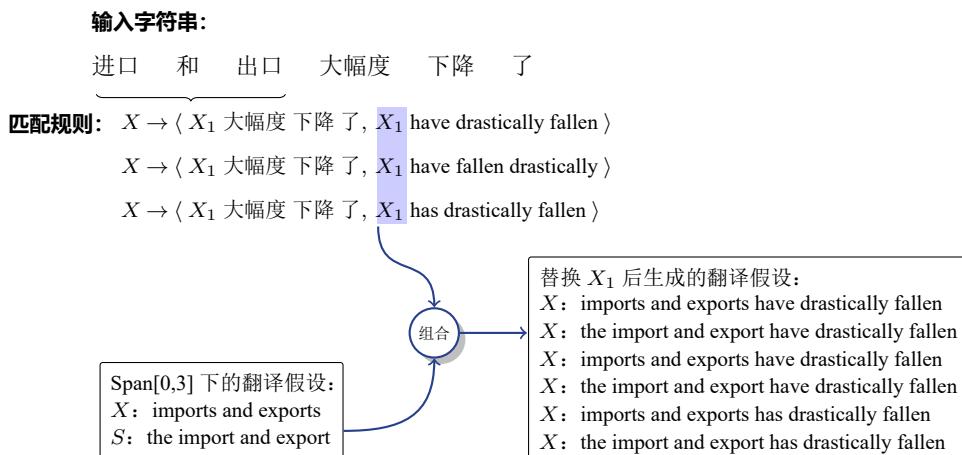


图 8.12 不同规则目标语端及变量译文的组合

这也就是说，当匹配规则的源语言部分“ $X_1$  大幅度 下降 了”时会有三个译文可以选择。而变量  $X_1$  部分又有很多不同的局部翻译结果。不同的规则译文和不同的

变量译文都可以组合出一个局部翻译结果。图8.12展示了这种情况的实例。

假设有  $n$  个规则的源语言端相同，规则中每个变量可以被替换为  $m$  个结果，对于只含有一个变量的规则，一共有  $nm$  种不同的组合。如果规则含有两个变量，这种组合的数量是  $nm^2$ 。由于翻译中会进行大量的规则匹配，如果每个匹配的源语言端都考虑所有  $nm^2$  种译文的组合，解码速度会很慢。

在层次短语系统中，会进一步对搜索空间剪枝。简言之，此时并不需要对所有  $nm^2$  种组合进行遍历，而是只考虑其中的一部分组合。这种方法也被称作**立方剪枝**（Cube Pruning）。所谓“立方”是指组合译文时的三个维度：规则的目标语端、第一个变量所对应的翻译候选、第二个变量所对应的翻译候选。立方剪枝假设所有的译文候选都经过排序，比如，按照短语翻译概率排序。这样，每个译文都对应一个坐标，比如， $(i, j, k)$  就表示第  $i$  个规则目标语端、第一个变量的第  $j$  个翻译候选、第二个变量的第  $k$  个翻译候选的组合。于是，可以把每种组合看作是一个三维空间中的一个点。在立方剪枝中，开始的时候会看到  $(0, 0, 0)$  这个翻译假设，并把这个翻译假设放入一个优先队列中。之后每次从这个优先队列里中弹出最好的结果，之后沿着三个维度分别将坐标加 1，比如，如果优先队列弹出  $(i, j, k)$ ，则会生成  $(i+1, j, k)$ 、 $(i, j+1, k)$  和  $(i, j, k+1)$  这三个新的翻译假设。之后，计算出它们的模型得分，并压入优先队列。这个过程不断被执行，直到达到终止条件，比如，扩展次数达到一个上限。

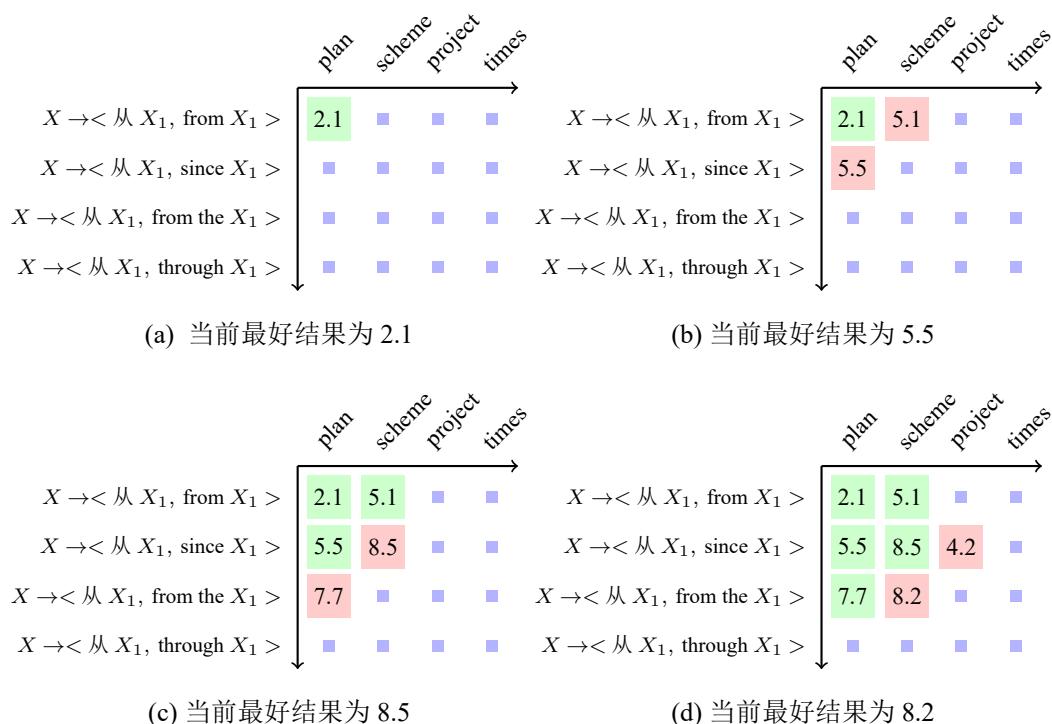


图 8.13 立方剪枝执行过程（行表示规则，列表示变量可替换的内容）

图8.13展示了立方剪枝的过程（规则只含有一个变量的情况）。可以看到，每个步骤中，算法只会扩展当前最好结果周围的两个点（对应两个维度，横轴对应变量被替换的内容，纵轴对应规则的目标语端）。

理论上，立方剪枝最多访问  $nm^2$  个点。但是在实践中发现，如果终止条件设计的合理，搜索的代价基本上与  $m$  或者  $n$  呈线性关系。因此，立方剪枝可以大大提高解码速度。立方剪枝实际上是一种启发性的搜索方法。它把搜索空间表示为一个三维空间。它假设：如果空间中某个点的模型得分较高，那么它“周围”的点的得分也很可能较高。这也是对模型得分沿着空间中不同维度具有连续性的一种假设。这种方法也可以使用在句法分析中，并取得了很好的效果。

### 8.3 基于语言学句法的模型

层次短语模型是一种典型的基于翻译文法的模型。它把翻译问题转化为语言分析问题。在翻译一个句子的时候，模型会生成一个树形结构，这样也就得到了句子结构的层次化表示。图8.14展示了一个使用层次短语模型进行翻译时所生成的翻译推导  $d$ ，以及这个推导所对应的树形结构（源语言）。这棵树体现了机器翻译的视角下的句子结构，尽管这个结构并不是人类语言学中的句法树。

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5 \circ r_2 \circ r_7 \circ r_6 \circ r_2$$

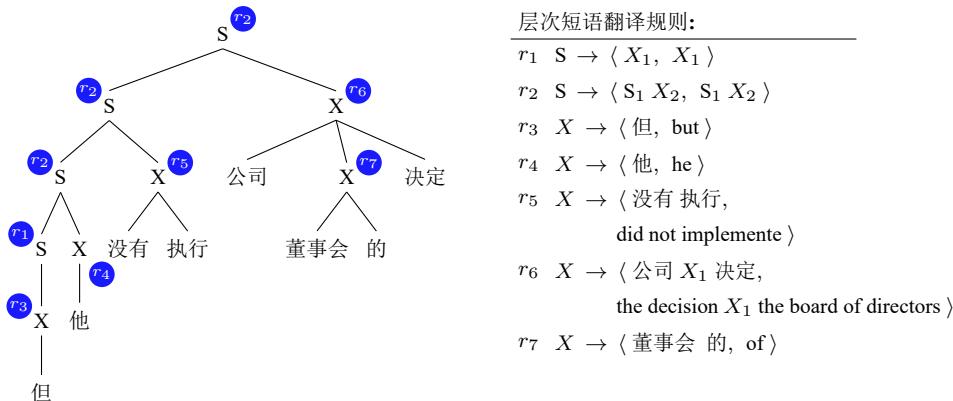


图 8.14 层次短语模型所对应的翻译推导及树结构（源语言）

在翻译中使用树结构的好处在于，模型可以更加有效地对句子的层次结构进行抽象。而且树结构可以作为对序列结构的一种补充，比如，在句子中距离较远的两个单词，在树结构中可以很近。不过，传统的层次短语模型也存在一些不足：

- 层次短语规则没有语言学句法标记，很多规则并不符合语言学认知，因此译文的生成和调序也无法保证遵循语言学规律。比如，层次短语系统经常会把完整的句法结构打散，或者“破坏”句法成分进行组合；
- 层次短语系统中有大量的工程化约束条件。比如，规则的源语言部分不允许两

个变量连续出现，而且变量个数也不能超过两个。这些约束在一定程度上限制了模型处理翻译问题的能力。

实际上，基于层次短语的方法可以被看作是介于基于短语的方法和基于语言学句法的方法之间的一种折中。它的优点在于，短语模型简单且灵活，同时，由于同步翻译文法可以对句子的层次结构进行表示，因此也能够处理一些较长距离的调序问题。但是，层次短语模型并不是一种“精细”的句法模型，当翻译需要复杂的结构信息时，这种模型可能会无能为力。

**参考答案：**The Xiyanghong star performance troupe presented a wonderful Peking opera as well as singing and dancing performance to the national audience .

**层次短语系统：**Star troupe of Xiyanghong, highlights of Peking opera and dance show to the audience of the national .

**句法系统：**The XYH star troupe [presented] [a wonderful Peking opera singing and dancing] [to] [the national audience] .

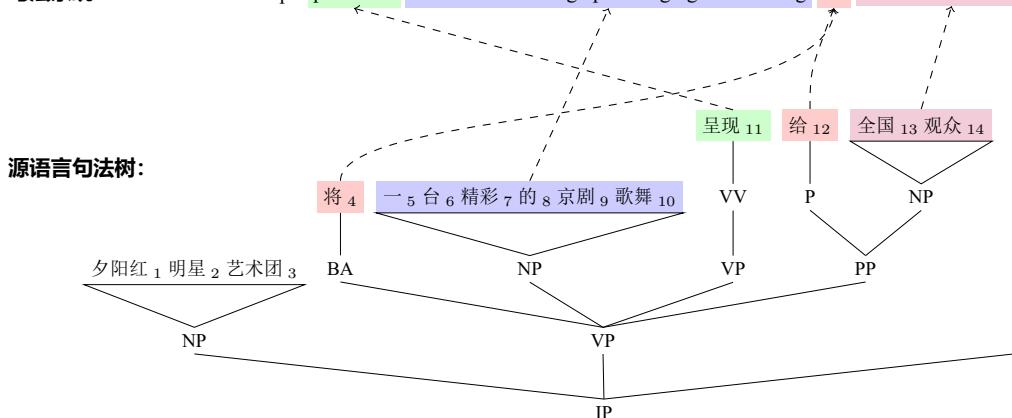


图 8.15 含有复杂调序的翻译实例（汉语翻译到英语）

图8.15展示了一个翻译实例，对图中句子进行翻译需要通过复杂的调序才能生成正确译文。为了完成这样的翻译，需要对多个结构（超过两个）进行调序，但是这种情况在标准的层次短语系统中是不允许的。

从这个例子中可以发现，如果知道源语言的句法结构，翻译其实并不“难”。比如，语言学句法结构可以告诉模型句子的主要成分是什么，而调序实际上是在这些成分之间进行的。从这个角度说，语言学句法可以帮助模型进行更上层结构的表示和调序。

显然，使用语言学句法对机器翻译进行建模也是一种不错的选择。不过，语言学句法有很多种，因此首先需要确定使用何种形式的句法。比如，在自然语言处理中经常使用的是短语结构分析和依存分析（图8.16）。二者的区别已经在第二章进行了讨论。

在机器翻译中，上述这两种句法信息都可以被使用。不过为了后续讨论的方便，这里仅介绍基于短语结构树的机器翻译建模。使用短语结构树的原因在于，它提供了较为丰富的句法信息，而且相关句法分析工具比较成熟。如果没有特殊说明，本

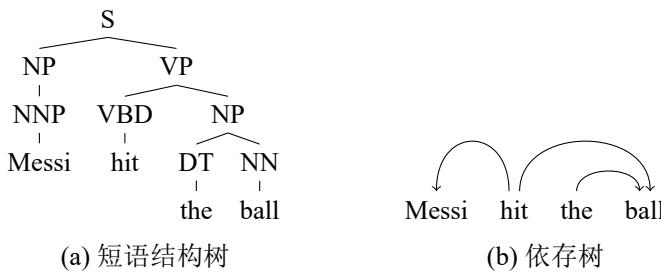


图 8.16 短语结构树 vs 依存树

章中所提到的句法树都是指短语结构树（或成分句法树），有时也会把句法树简称为树。此外，这里也假设所有句法树都可以由句法分析器自动生成<sup>7</sup>。

### 8.3.1 基于句法的翻译模型分类

可以说基于句法的翻译模型贯穿了现代统计机器翻译的发展历程。从概念上讲，不管是层次短语模型，还是语言学句法模型都是基于句法的模型。基于句法的机器翻译模型种类繁多，这里先对相关概念进行简要介绍，以避免后续论述中产生歧义。表8.2给出了基于句法的机器翻译中涉及的一些概念。

表 8.2 基于句法的机器翻译中常用概念

术语	说明
翻译规则	翻译的最小单元（或步骤）
推导	由一系列规则组成的分析或翻译过程，推导可以被看作是规则的序列
规则表	翻译规则的存储表示形式，可以高效进行查询
层次短语模型	基于同步上下文无关文法的翻译模型，非终结符只有 S 和 X 两种，文法并不需要符合语言学句法约束
树到串模型	一类翻译模型，它使用源语语言学句法树，因此翻译可以被看作是从句法树到词串的转换
串到树模型	一类翻译模型，它使用目标语语言学句法树，因此翻译可以被看作是从词串到句法树的转换
树到树模型	一类翻译模型，它同时使用源语和目标语语言学句法树，因此翻译可以被看作从句法树到句法树的转换
基于句法	使用语言学句法
基于树	（源语言）使用树结构（大多指句法树）

<sup>7</sup>对于汉语、英语等大语种，句法分析器的选择有很多。不过，对于一些小语种，句法标注数据有限，句法分析可能并不成熟，这时在机器翻译中使用语言学句法信息会面临较大的挑战。

术语	说明
基于串	(源语言) 使用词串, 比如串到树翻译系统的解码器一般都是基于串的解码方法
基于森林	(源语言) 使用句法森林, 这里森林只是对多个句法树的一种压缩结构表示
词汇化规则	含有终结符的规则
非词汇规则	不含有终结符的规则
句法软约束	不强制规则推导匹配语言学句法树, 通常把句法信息作为特征使用
句法硬约束	要求推导必须符合语言学句法树, 不符合的推导会被过滤掉

基于句法的翻译模型可以被分为两类: 基于形式化文法的模型和语言学上基于句法的模型(图8.17)。基于形式化文法的模型的典型代表包括, 基于反向转录文法的模型<sup>[341]</sup> 和基于层次短语的模型<sup>[336]</sup>。而语言学上基于句法的模型包括, 句法树到串的模型<sup>[86, 342]</sup>、串到句法树的模型<sup>[87, 343]</sup>、句法树到句法树的模型<sup>[344, 345]</sup>等。

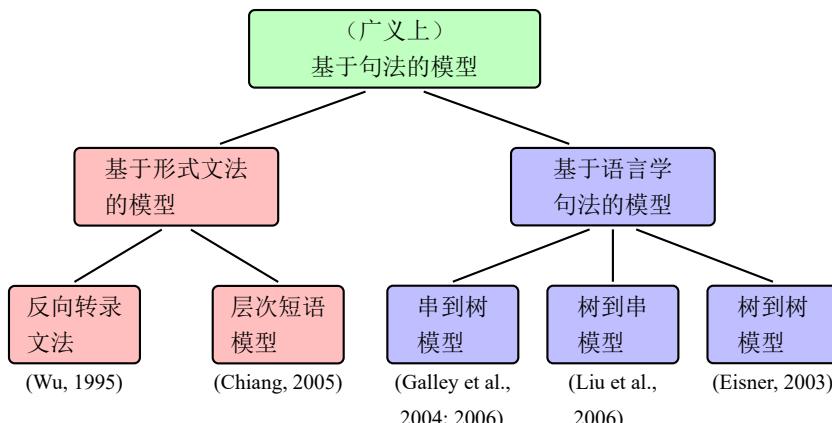


图 8.17 基于句法的机器翻译模型的分类

通常来说, 基于形式化文法的模型并不需要句法分析技术的支持。这类模型只是把翻译过程描述为一系列形式化文法规则的组合过程。而语言学上基于句法的模型则需要源语言和(或者)目标语言句法分析的支持, 以获取更丰富的语言学信息来提高模型的翻译能力。这也是本节所关注的重点。当然, 所谓分类也没有唯一的标准, 比如, 还可以把句法模型分为基于软约束的模型和基于硬约束的模型, 或者分为基于树的模型和基于串的模型。

表8.3进一步对比了不同模型的区别。其中, 树到串和树到树模型都使用了源语言句法信息, 串到树和树到树模型使用了目标语言句法信息。不过, 这些模型都依

表 8.3 基于句法的机器翻译模型对比

模型	形式句法	语言学句法		
		树到串	串到树	树到树
源语句法	否	是	否	是
目标语句法	否	否	是	是
基于串的解码	是	否	是	是
基于树的解码	否	是	否	是
健壮性	高	中	中	低

赖句法分析器的输出，因此会对句法分析的错误比较敏感。相比之下，基于形式文法的模型并不依赖句法分析器，因此会更健壮一些。

### 8.3.2 基于树结构的文法

基于句法的翻译模型的一个核心问题是需要对树结构进行建模，进而完成树之间或者树和串之间的转换。在计算机领域中，所谓树就是由一些节点组成的层次关系的集合。计算机领域的树和自然世界中的树没有任何关系，只是借用了相似的概念，因为这种层次结构很像一棵倒过来的树。在使用树时，经常会把树的层次结构转化为序列结构，称为树结构的**序列化**或者**线性化**（Linearization）。



图 8.18 树结构的不同表示形式

比如，使用树的先序遍历就可以得到一个树的序列表示。图8.18就对比了同一棵树的不同表示方式。实际上，树的序列表示是非常适合计算机进行读取和处理的。因此，本章也会使用树的序列化结果来表示句法结构。

在基于语言学句法的机器翻译中，两个句子间的转化仍然需要使用文法规则进行描述。有两种类型的规则：

- **树到串翻译规则**（Tree-to-String Translation Rule）：在树到串、串到树模型中使用；

- **树到树翻译规则** (Tree-to-Tree Translation Rule)：在树到树模型中使用。

树到串规则描述了一端是树结构而另一端是串的情况，因此树到串模型和串到树模型都可以使用这种形式的规则。树到树模型需要在两种语言上同时使用句法树结构，需要树到树翻译规则。

## 1. 树到树翻译规则

虽然树到串翻译规则和树到树翻译规则蕴含了不同类型的翻译知识，但是它们都在描述一个结构（树/串）到另一个结构（树/串）的映射。这里采用了一种更加通用的文法——基于树结构的文法——将树到串翻译规则和树到树翻译规则进行统一。定义如下：

### 定义 8.3.1 基于树结构的文法

一个基于树结构的文法由七部分构成  $(N_s, N_t, T_s, T_t, I_s, I_t, R)$ ，其中

1.  $N_s$  和  $N_t$  是源语言和目标语言非终结符集合；
2.  $T_s$  和  $T_t$  是源语言和目标语言终结符集合；
3.  $I_s \subseteq N_s$  和  $I_t \subseteq N_t$  是源语言和目标语言起始非终结符集合；
4.  $R$  是规则集合，每条规则  $r \in R$  有如下形式：

$$\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$$

其中，规则左部由非终结符  $\alpha_h \in N_s$  和  $\beta_h \in N_t$  构成；规则右部由三部分组成， $\alpha_r$  表示由源语言终结符和非终结符组成的树结构； $\beta_r$  表示由目标语言终结符和非终结符组成的树结构； $\sim$  表示  $\alpha_r$  和  $\beta_r$  中叶子非终结符的 1-1 对应关系。

基于树结构的规则非常适合于描述树结构到树结构的映射。比如，图8.19是一个汉语句法树结构到一个英语句法树结构的对应。其中的树结构可以被看作是完整句法树上的一个片段，称为**树片段** (Tree Fragment)。

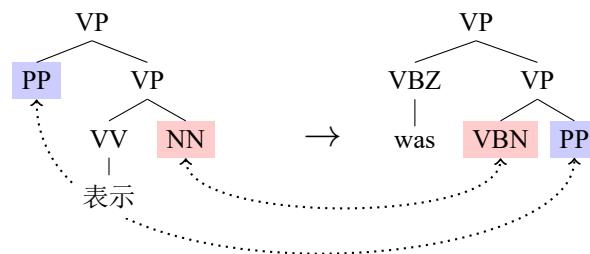


图 8.19 汉语句法树到英语句法树的结构对应

树片段的叶子节点既可以是终结符（单词）也可以是非终结符。当叶子节点为非终结符时，表示这个非终结符会被进一步替换，因此它可以被看作是变量。而源语言树结构和目标语言树结构中的变量是一一对应的，对应关系用虚线表示。

这个双语映射关系可以被表示为一个基于树结构的文法规则，套用规则的定义

$\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$  形式, 可以知道:

$$\begin{aligned}\alpha_h &= VP \\ \beta_h &= VP \\ \alpha_r &= VP(PP:x\ VP(VV(\text{表示})\ NN:x)) \\ \beta_r &= VP(VBZ(was)\ VP(VBN:x\ PP:x)) \\ \sim &= \{1-2, 2-1\}\end{aligned}$$

这里,  $\alpha_h$  和  $\beta_h$  表示规则的左部, 对应树片段的根节点;  $\alpha_r$  和  $\beta_r$  是两种语言的树结构 (序列化表示), 其中标记为  $x$  的非终结符是变量。 $\sim = \{1-2, 2-1\}$  表示源语言的第一个变量对应目标语言的第二个变量, 而源语言的第二个变量对应目标语言的第一个变量, 这也反应出两种语言句法结构中的调序现象。类似于层次短语规则, 可以把规则中变量的对应关系用下标进行表示。例如, 上面的规则也可以被写为如下形式:

$$\langle VP, VP \rangle \rightarrow \langle PP_1 VP(VV(\text{表示}) NN_2), VP(VBZ(was) VP(VBN_2 PP_1)) \rangle$$

其中, 两种语言中变量的对应关系为  $PP_1 \leftrightarrow PP_1$ ,  $NN_2 \leftrightarrow VBN_2$ 。

## 2. 基于树结构的翻译推导

规则中的变量预示着一种替换操作, 即变量可以被其他树结构替换。实际上, 上面的树到树翻译规则就是一种**同步树替换文法** (Synchronous Tree-substitution Grammar) 规则。不论是源语言端还是目标语言端, 都可以通过这种替换操作不断生成更大的树结构, 也就是通过树片段的组合得到更大的树片段。图8.20就展示了树替换操作的一个实例。

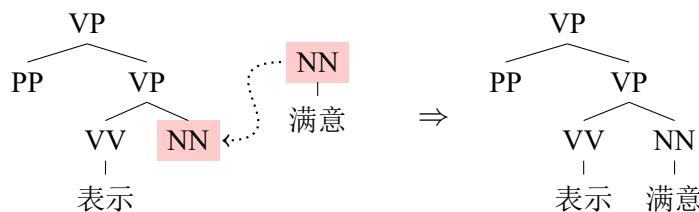


图 8.20 树替换操作 (将 NN 替换为一个树结构)

这种方法也可以被扩展到双语的情况。图8.21给出了一个使用基于树结构的同步文法生成双语句对的实例。其中, 每条规则都同时对应源语言和目标语言的一个树片段 (用矩形表示)。变量部分可以被替换, 这个过程不断执行。最后, 四条规则

组合在一起形成了源语言和目标语言的句法树。这个过程也被称作规则的推导。

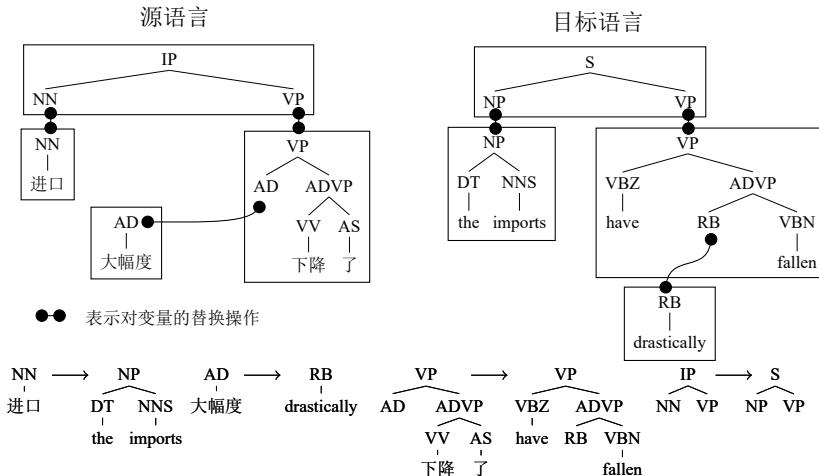


图 8.21 使用基于树结构的同步文法生成汉英句对

规则的推导对应了一种源语言和目标语言树结构的同步生成过程。比如，使用下面的规则集：

- $r_3 : AD(\text{大幅度}) \rightarrow RB(\text{drastically})$
- $r_4 : VV(\text{减少}) \rightarrow VBN(\text{fallen})$
- $r_6 : AS(\text{了}) \rightarrow VBP(\text{have})$
- $r_7 : NN(\text{进口}) \rightarrow NP(DT(\text{the})) NNS(\text{imports})$
- $r_8 : VP(AD_1 VP(VV_2 AS_3)) \rightarrow VP(VBP_3 ADVP(RB_1 VBN_2))$
- $r_9 : IP(NN_1 VP_2) \rightarrow S(NP_1 VP_2)$

可以得到一个翻译推导：

$$\begin{aligned}
 & \langle IP^{[1]}, S^{[1]} \rangle \\
 \xrightarrow[r_9]{IP^{[1]} \Leftrightarrow S^{[1]}} & \langle IP(NN^{[2]} VP^{[3]}), S(NP^{[2]} VP^{[3]}) \rangle \\
 \xrightarrow[r_7]{NN^{[2]} \Leftrightarrow NP^{[2]}} & \langle IP(NN(\text{进口}) VP^{[3]}), S(NP(DT(\text{the})) NNS(\text{imports})) VP^{[3]} \rangle \\
 \xrightarrow[r_8]{VP^{[3]} \Leftrightarrow VP^{[3]}} & \langle IP(NN(\text{进口}) VP(AD^{[4]} VP(VV^{[5]} AS^{[6]}))), \\
 & S(NP(DT(\text{the})) NNS(\text{imports})) VP(VBP^{[6]} ADVP(RB^{[4]} VBN^{[5]}))) \rangle \\
 \xrightarrow[r_3]{AD^{[4]} \Leftrightarrow RB^{[4]}} & \langle IP(NN(\text{进口}) VP(AD(\text{大幅度}) VP(VV^{[5]} AS^{[6]}))), \\
 & S(NP(DT(\text{the})) NNS(\text{imports})) VP(VBP^{[6]} ADVP(RB(drastically) VBN^{[5]}))) \rangle
 \end{aligned}$$

$$\begin{array}{l}
 \xrightarrow[r_4]{\text{VV}^{[5]} \Leftrightarrow \text{VBN}^{[5]}} \langle \text{IP}(\text{NN(进口)} \text{ VP}(\text{AD(大幅度)} \text{ VP}(\text{VV(减少)} \text{ AS}^{[6]}))), \\
 \quad \text{S}(\text{NP}(\text{DT(the)} \text{ NNS(imports)}) \text{ VP}(\text{VBP}^{[6]}) \\
 \quad \text{ADVP}(\text{RB(drastically)} \text{ VBN(fallen)))) \rangle \\
 \xrightarrow[r_6]{\text{AS}^{[6]} \Leftrightarrow \text{VBP}^{[6]}} \langle \text{IP}(\text{NN(进口)} \text{ VP}(\text{AD(大幅度)} \text{ VP}(\text{VV(减少)} \text{ AS(了)))), \\
 \quad \text{S}(\text{NP}(\text{DT(the)} \text{ NNS(imports)}) \text{ VP}(\text{VBP(have)}) \\
 \quad \text{ADVP}(\text{RB(drastically)} \text{ VBN(fallen)))) \rangle
 \end{array}$$

其中，箭头  $\rightarrow$  表示推导之意。显然，可以把翻译看作是基于树结构的推导过程（记为  $d$ ）。因此，与层次短语模型一样，基于语言学句法的机器翻译也是要找到最佳的推导  $\hat{d} = \arg \max_d P(d)$ 。

### 3. 树到串翻译规则

基于树结构的文法可以很好地表示两个树片段之间的对应关系，即树到树翻译规则。那树到串翻译规则该如何表示呢？实际上，基于树结构的文法也同样适用于树到串模型。比如，图8.22是一个树片段到串的映射，它可以被看作是树到串规则的一种表示。

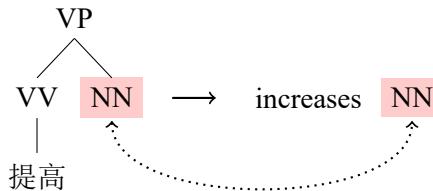


图 8.22 树片段到串的映射

在图8.22中，源语言树片段中的叶子结点 NN 表示变量，它与右手端的变量 NN 对应。这里仍然可以使用基于树结构的规则对上面这个树到串的映射进行表示。参照规则形式  $\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ ，有：

$$\begin{aligned}
 \alpha_h &= \text{VP} \\
 \beta_h &= \text{VP} \\
 \alpha_r &= \text{VP}(\text{VV(提高)} \text{ NN}:x) \\
 \beta_r &= \text{VP}(\text{increases} \text{ NN}:x) \\
 \sim &= \{1 - 1\}
 \end{aligned}$$

这里，源语言部分是一个树片段，因此  $\alpha_h$  和  $\alpha_r$  很容易确定。对于目标语部分，可以把这个符号串当作是一个单层的树片段，根结点直接共享源语言树片段的根结

点，叶子结点就是符号串本身。这样，也可以得到  $\beta_h$  和  $\beta_r$ 。从某种意义上说，树到串翻译仍然体现了一种双语的树结构，只是目标语部分不是语言学句法驱动的，而是一种借用了源语言句法标记所形成的层次结构。

这里也可以把变量的对齐信息用下标表示。同时，由于  $\alpha_h$  和  $\beta_h$  是一样的，可以将左部两个相同的非终结符合并，于是规则可以被写作：

$$\text{VP} \rightarrow \langle \text{VP}(\text{VV(提高)} \text{ NN}_1), \text{ increases } \text{NN}_1 \rangle$$

另外，在机器翻译领域，大家习惯把规则看作源语言结构（树/串）到目标语言结构（树/串）的一种映射，因此常常会把上面的规则记为：

$$\text{VP(VV(提高)} \text{ NN}_1) \rightarrow \text{increases } \text{NN}_1$$

在后面的内容中也会使用这种形式来表示基于句法的翻译规则。

### 8.3.3 树到串翻译规则抽取

基于句法的机器翻译包括两个步骤：文法归纳和解码。其中，文法归纳是指从双语平行数据中自动学习翻译规则及规则所对应的特征；解码是指利用得到的文法对新的句子进行分析，并获取概率最高的翻译推导。

本节首先介绍树到串文法归纳的经典方法——GHKM 方法<sup>[87, 343]</sup>。所谓 GHKM 是四位作者名字的首字母。GHKM 方法的输入包括：

- 源语言句子及其句法树；
- 目标语言句子；
- 源语言句子和目标语言句子之间的词对齐。

它的输出是这个双语句对上的树到串翻译规则。GHKM 不是一套单一的算法，它还包括很多技术手段用于增加规则的覆盖度和准确性。下面就具体看看 GHKM 是如何工作的。

#### 1. 树的切割与最小规则

获取树到串规则就是要找到源语言树片段与目标语言词串之间的对应关系。一棵句法树会有多个树片段，那么哪些树片段可以和目标语言词串产生对应关系呢？

在 GHKM 方法中，源语言树片段和目标语言词串的对应是由词对齐决定的。GHKM 假设：一个合法的树到串翻译规则，不应该违反词对齐。这个假设和双语短语抽取中的词对齐一致性约束是一样的（见第七章短语抽取小节）。简单来说，规则中两种语言互相对应的部分不应包含对齐到外部的词对齐连接。

为了说明这个问题，来看一个例子。图8.23包含了一棵句法树、一个词串和它们

之间的词对齐结果。图中包含如下规则：

$$\text{PP}(\text{P(对)} \text{ NP}(\text{NN(回答)})) \rightarrow \text{with the answer}$$

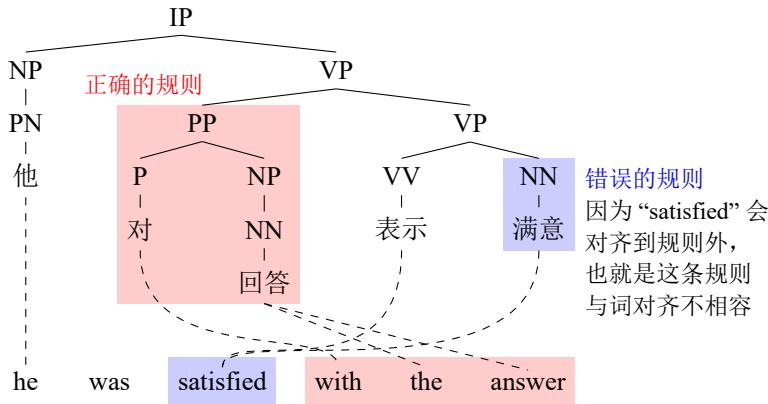


图 8.23 树到串规则与词对齐兼容性示例

该规则是一条满足词对齐约束的规则（对应于图8.23中红色部分），因为不存在从规则的源语言或目标语言部分对齐到规则外部的情况。但是，如下的规则却是一条不合法的规则：

$$\text{NN(满意)} \rightarrow \text{satisfied}$$

这是因为，“satisfied”除了对齐到“满意”，还对齐到“表示”。也就是，这条规则会产生歧义，因为“satisfied”不应该只由“满意”生成。

为了能够获得与词对齐相兼容的规则，GHKM 引入了几个概念。首先，GHKM 方法中定义了可达范围（Span）和补充范围（Complement Span）：

### 定义 8.3.2 可达范围（Span）

对于一个源语言句法树节点，它的可达范围是这个节点所对应到的目标语言第一个单词和最后一个单词所构成的索引范围。

### 定义 8.3.3 补充范围（Complement Span）

对于一个源语言句法树节点，它的补充范围是除了它的祖先和子孙节点外的其他节点可达范围的并集。

可达范围定义了每个节点覆盖的源语言片段所对应的目标语言片段。实际上，它表示了目标语言句子上的一个跨度，这个跨度代表了这个源语言句法树节点所能达到的最大范围。因此可达范围实际上是一个目标语单词索引的范围。补充范围是与可达范围相对应的一个概念，它定义了句法树中一个节点之外的部分对应到目标语

的范围，但是这个范围并不必须是连续的。

有了可达范围和补充范围的定义之后，可以进一步定义：

#### 定义 8.3.4 可信节点 (Admissible Node)

对于源语言树节点  $node$ ，如果它的可达范围和补充范围不相交，节点  $node$  就是一个可信节点，否则是一个不可信节点。

可信节点表示这个树节点  $node$  和树中的其他部分（不包括  $node$  的祖先和孩子）没有任何词对齐上的歧义。也就是说，这个节点可以完整地对应到目标语言句子的一个连续范围，不会出现在这个范围中的词对应到其他节点的情况。如果节点不是可信节点，则表示它会引起词对齐的歧义，因此不能作为树到串规则中源语言树片段的根节点或者变量部分。图8.24给出了一个可信节点的实例。

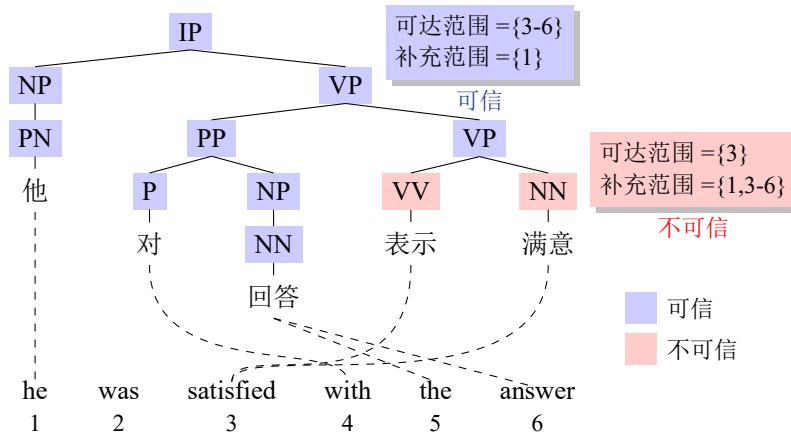


图 8.24 标注了可信节点信息的句法树

进一步，可以定义树到串模型中合法的树片段：

#### 定义 8.3.5 合法的树片段

如果一个树片段的根节点是可信节点，同时它的叶子节点中的非终结符节点也是可信节点，那么这个树片段就是不产生词对齐歧义的树片段，也被称为合法的树片段。

图8.25是一个基于可信节点得到的树到串规则：

$$VP(PP(P(对) NP(NN(回答)))) \text{ VP}_1 \rightarrow \text{VP}_1 \text{ with the answer}$$

其中，蓝色部分表示可以抽取到的规则，显然它的根节点和叶子非终结符节点都是可信节点。由于源语言树片段中包含一个变量（VP），因此需要对 VP 节点的可达范围进行泛化（红色方框部分）。

至此，对于任何一个树片段都能够使用上述方法判断它是否合法。如果合法，就可以抽取相应的树到串规则。但是，枚举句子中的所有树片段并不是一个很高效的

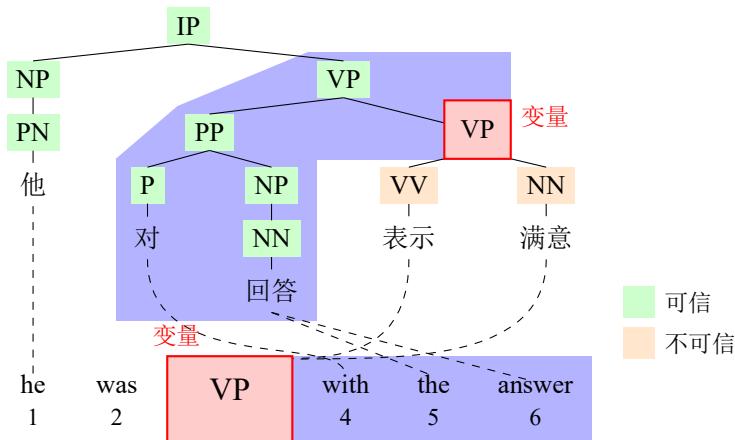
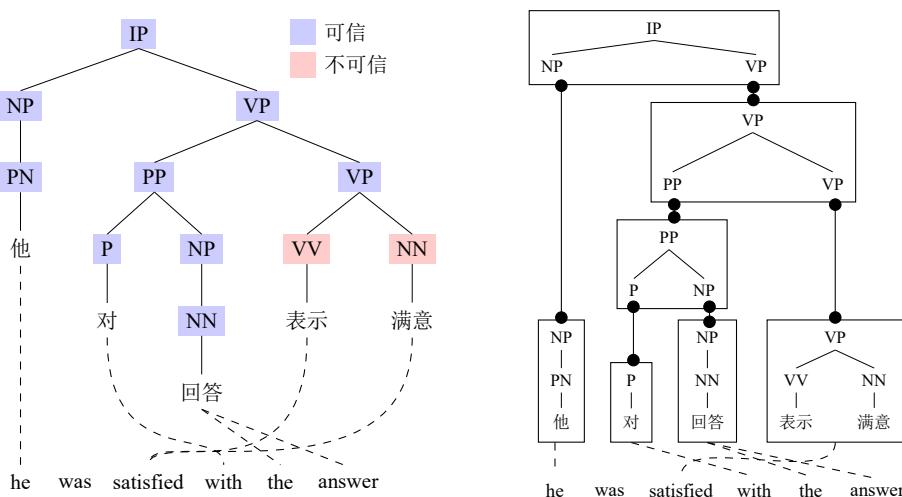


图 8.25 根据可信结点得到的树到串翻译规则

方法，因为对于任何一个节点，以它为根的树片段数量随着其深度和宽度的增加呈指数增长。在 GHKM 方法中，为了避免低效的枚举操作，可以使用另一种方法抽取规则。

实际上，可信节点确定了哪些地方可以作为规则的边界（合法树片段的根节点或者叶子节点），可以把所有的可信节点看作是一个边缘集合（Frontier Set）。所谓边缘集合就是定义了哪些地方可以被“切割”，通过这种切割可以得到一个个合法的树片段，这些树片段无法再被切割为更小的合法树片段。图8.26给出了一个通过边缘集合定义的树切割。图右侧中的矩形框表示切割得到的树片段。



(a) 标有可信节点信息的句法树

(b) 通过边缘集合定义切割得到的句法树

图 8.26 根据边缘节点定义的树切割

需要注意的是，因为“ $NP \rightarrow PN \rightarrow$  他”对应着一个单目生成的过程，所以这里“ $NP(PN(他))$ ”被看作是一个最小的树片段。当然，也可以把它当作两个树片段“ $NP(PN)$ ”和“ $PN(他)$ ”，不过这种单目产生式往往会导致解码时推导数量的膨胀。因此，这里约定把连续的单目生成看作是一个生成过程，它对应一个树片段，而不是多个。

将树进行切割之后，可以得到若干树片段，每个树片段都可以对应一个树到串规则。由于这些树片段不能被进一步切割，因此这样得到的规则也被称作**最小规则**（Minimal Rules）。它们构成了树到串模型中最基本的翻译单元。图8.27展示了基于树切割得到的最小规则。其中左侧的每条规则都对应着右侧相同编号的树片段。

- $r_1 \quad NP(PN(他)) \rightarrow he$
- $r_2 \quad P(\text{对}) \rightarrow with$
- $r_3 \quad NP(NN(\text{回答})) \rightarrow the answer$
- $r_4 \quad VP(VV(\text{表示})) NN(\text{满意}) \rightarrow satisfied$
- $r_5 \quad PP(P_1 NP_2) \rightarrow P_1 NP_2$
- $r_6 \quad VP(PP_1 VP_2) \rightarrow VP_2 PP_1$
- $r_7 \quad IP(NP_1 VP_2) \rightarrow NP_1 VP_2$

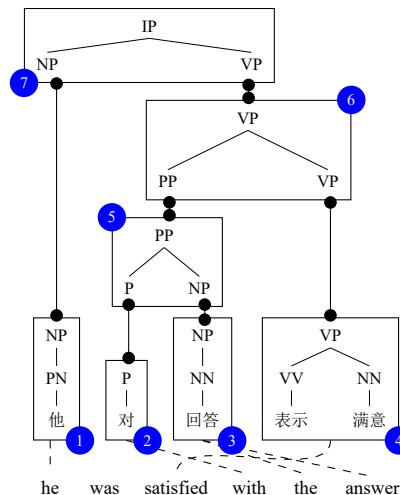


图 8.27 基于树切割得到的最小规则实例

## 2. 空对齐处理

空对齐是翻译中的常见现象。比如，一些虚词经常找不到在另一种语言中的对应，因此不会被翻译，这种情况也被称作空对齐。在图8.27中目标语中的“was”就是一个空对齐单词。空对齐的使用可以大大增加翻译的灵活度。具体到树到串规则抽取任务，需要把空对齐考虑进来，这样能够覆盖更多的语言现象。

处理空对齐单词的手段非常简单。只需要把空对齐单词附着在它周围的规则上即可。也就是，检查每条最小规则，如果空对齐单词能够作为规则的一部分进行扩展，就可以生成一条新的规则。

图8.28展示了前面例子中“was”被附着在周围的规则上的结果。其中，含有红色“was”的规则是通过附着空对齐单词得到的新规则。比如，对于规则：

$$NP(PN(他)) \rightarrow he$$

“was”紧挨着这个规则目标端的单词“he”，因此可以把“was”包含在规则的

- $r_1 \quad \text{NP}(\text{PN(他)}) \rightarrow \text{he}$   
 $r_4 \quad \text{VP}(\text{VV(表示)} \text{ NN(满意)}) \rightarrow$   
 satisfied  
 $r_6 \quad \text{VP}(\text{PP}_1 \text{ VP}_2) \rightarrow \text{VP}_2 \text{ PP}_1$   
 $r_7 \quad \text{IP}(\text{NP}_1 \text{ VP}_2) \rightarrow \text{NP}_1 \text{ VP}_2$   
 $r_8 \quad \text{NP}(\text{PN(他)}) \rightarrow \text{he was}$   
 $r_9 \quad \text{VP}(\text{VV(表示)} \text{ NN(满意)}) \rightarrow$   
**was** satisfied  
 $r_{10} \quad \text{VP}(\text{PP}_1 \text{ VP}_2) \rightarrow$   
**was** VP<sub>2</sub> PP<sub>1</sub>  
 $r_{11} \quad \text{IP}(\text{NP}_1 \text{ VP}_2) \rightarrow$   
 NP<sub>1</sub> **was** VP<sub>2</sub>

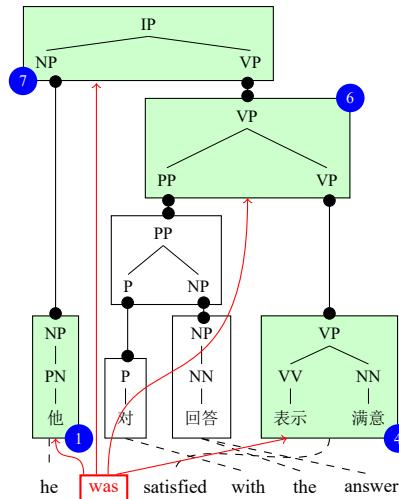


图 8.28 树到串规则抽取中空对齐单词的处理（绿色矩形）

目标端，形成新的规则：

$$\text{NP}(\text{PN(他)}) \rightarrow \text{he was}$$

通常，在规则抽取中考虑空对齐可以大大增加规则的覆盖度。

### 3. 组合规则

最小规则是基于句法的翻译模型中最小的翻译单元。但是，在翻译复杂句子的时候，往往需要更大范围的上下文信息，比如，本节开始图8.15中的例子，需要一条规则同时处理多个变量的调序，而这种规则很可能不是最小规则。为了得到“更大”的规则，一种方法是对最小规则进行组合。得到的规则称为 *composed-m* 规则，其中  $m$  表示这个规则是由  $m$  条最小规则组合而成。

规则的组合非常简单。只需要在得到最小规则之后，对相邻的规则进行拼装。也就是说，如果某个树片段的根节点出现在另一个树片段的叶子节点处，就可以把它们组合成更大的树片段。图8.29给了规则组合的实例。其中，规则 1、5、6、7 可以组合成一条 composed-4 规则，这个规则可以进行非常复杂的调序。

在真实系统开发中，组合规则一般会带来明显的性能提升。不过随着组合规则数量的增加，规则集也会膨胀。因此往往需要在翻译性能和文法大小之间找到一种平衡。

### 4. SPMT 规则

组合规则固然有效，但并不是所有组合规则都非常好用。比如，在机器翻译中已经发现，如果一个规则含有连续词串（短语），这种规则往往会比较可靠。但是由

- $r_1 \quad NP(PN(\text{他})) \rightarrow he$   
 $r_5 \quad PP(P_1 NP_2) \rightarrow P_1 NP_2$   
 $r_6 \quad VP(PP_1 VP_2) \rightarrow VP_2 PP_1$   
 $r_7 \quad IP(NP_1 VP_2) \rightarrow NP_1 VP_2$   
 $r_{1,7} \quad IP(NP(PN(\text{他}))) VP_1 \rightarrow he VP_1$   
 $r_{1,6,7} \quad IP(NP(PN(\text{他}))) VP(PP_1 VP_2) \rightarrow he VP_2 PP_1$   
 $r_{1,5,6,7} \quad IP(NP(PN(\text{他}))) VP(P_1 NP_2 VP_3) \rightarrow he VP_3 P_1 NP_2$

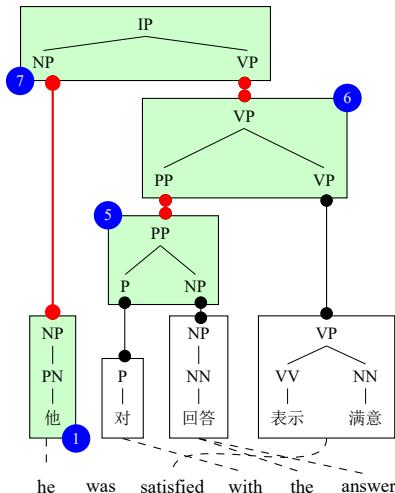


图 8.29 对最小规则进行组合 (绿色矩形)

于句法树结构复杂，获取这样的规则可能会需要很多次规则的组合，规则抽取的效率很低。

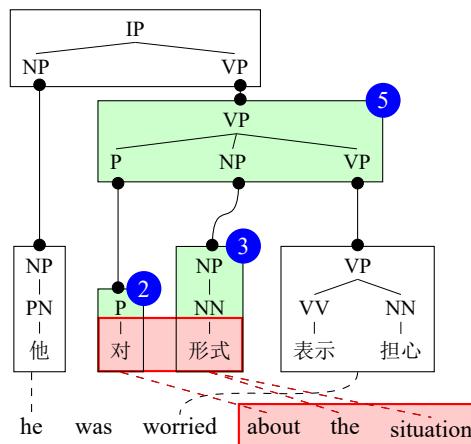


图 8.30 短语 (红色) 所对应的树片段 (绿色)

针对这个问题，一种解决办法是直接从词串出发进行规则抽取。这种方法被称为 SPMT 方法<sup>[346]</sup>。它的思想是：对于任意一个与词对齐兼容的短语，可以找到包含它的“最小”翻译规则，即 SPMT 规则。如图 8.30 所示，可以得到短语翻译：

对 形式  $\rightarrow$  about the situation

然后，从这个短语出发向上搜索，找到覆盖这个短语的最小树片段，之后生成

规则即可。在这个例子中可以得到 SPMT 规则：

$$\text{VP}(\text{P(对)} \text{ NP}(\text{NN(形式)})) \text{ VP}_1 \rightarrow \text{VP}_1 \text{ about the situation}$$

而这条规则需要组合三条最小规则才能得到，但是在 SPMT 中可以直接得到。相比规则组合的方法，SPMT 方法可以更有效地抽取包含短语的规则。

## 5. 句法树二叉化

句法树是使用人类语言学知识归纳出来的一种解释句子结构的工具。比如，CTB<sup>[347]</sup>、PTB<sup>[348]</sup> 等语料就是常用的训练句法分析器的数据。

但是，这些数据的标注中会含有大量的扁平结构，如图8.31所示，多个分句可能会导致一个根节点下有很多个分支。这种扁平的结构会给规则抽取带来麻烦。

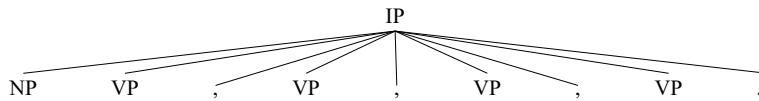


图 8.31 CTB 中含有多个分句的句法树结构

图8.32给出了一个实例，其中的名词短语（NP），包含四个词，都在同一层树结构中。由于“乔治 华盛顿”并不是一个独立的句法结构，因此无法抽取类似于下面这样的规则：

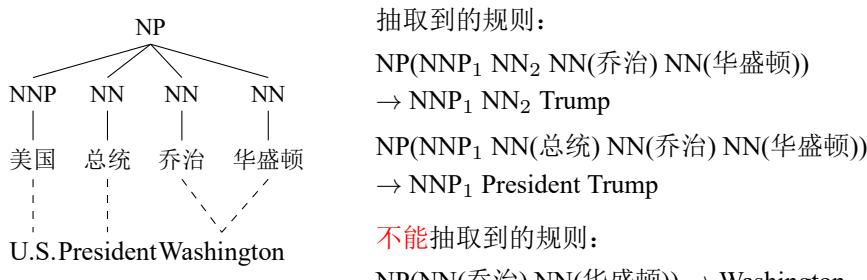
$$\text{NP}(\text{NN(乔治)}) \text{ NN(华盛顿)} \rightarrow \text{Washington}$$


图 8.32 一个扁平的句法结构所对应的规则抽取结果

对于这个问题，一种解决办法是把句法树变得更深，使局部的翻译片段更容易被抽取出来。常用的手段是树**二叉化**（Binarization）。比如，图8.33就是一个树二叉化的实例。二叉化生成了一些新的节点（记为 X-BAR），其中“乔治 华盛顿”被作为一个独立的结构体现出来。这样，就能够抽取到规则：

$$\text{NP-BAR}(\text{NN}(\text{乔治})) \text{ NN}(\text{华盛顿}) \rightarrow \text{Washington}$$

$$\text{NP-BAR}(\text{NN}_1 \text{ NP-BAR}_2) \rightarrow \text{NN}_1 \text{ NP-BAR}_2$$

由于树二叉化可以帮助规则抽取得到更细颗粒度的规则，提高规则抽取的召回率，因此成为了基于句法的机器翻译中的常用方法。二叉化方法也有很多不同的实现策略<sup>[349, 350, 351]</sup>，比如：左二叉化、右二叉化、基于中心词的二叉化等。具体实现时可以根据实际情况进行选择。

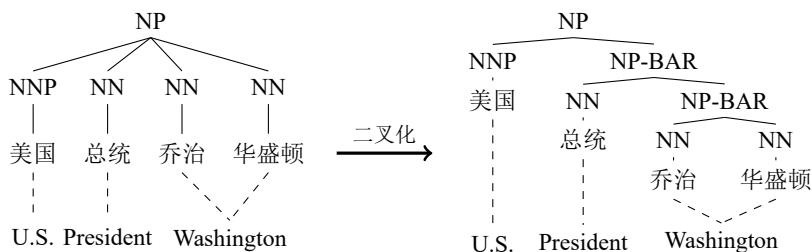


图 8.33 句法树二叉化结果

#### 8.3.4 树到树翻译规则抽取

树到串/串到树模型只在一个语言端使用句法树，而树到树模型可以同时利用源语言和目标语言句法信息，因此可以更细致地刻画两种语言结构的对应关系，进而更好地完成句法结构的调序和生成。树到树翻译中，需要两端都有树结构的规则，比如：

$$\langle \text{VP}, \text{VP} \rangle \rightarrow \langle \text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)), \\ \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1)) \rangle$$

也可以把它写为如下形式：

$$\text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)) \rightarrow \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1))$$

其中，规则的左部是源语言句法树结构，右部是目标语言句法树结构，变量的下标表示对应关系。为了获取这样的规则，需要进行树到树规则抽取。最直接的办法是把GHKM方法推广到树到树翻译的情况。比如，可以利用双语结构的约束和词对齐，定义树的切割点，之后找到两种语言树结构的映射关系<sup>[352]</sup>。

## 1. 基于节点对齐的规则抽取

不过，GHKM 方法的问题在于过于依赖词对齐结果。在树到树翻译中，真正需要的是树结构（节点）之间的对应关系，而不是词对齐。特别是在两端都加入句法树结构约束的情况下，词对齐的错误可能会导致较为严重的规则抽取错误。图8.34就给出了一个实例，其中，中文的“了”被错误的对齐到了英文的“the”，导致很多高质量的规则无法被抽取出来。

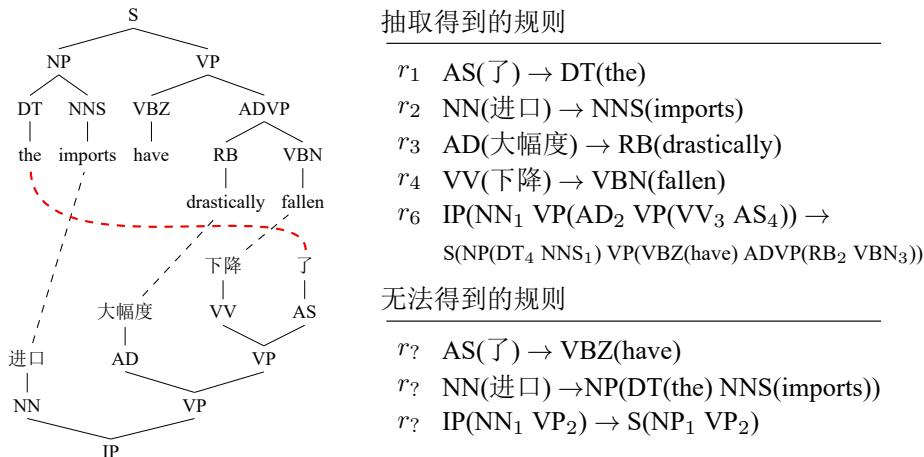


图 8.34 基于词对齐的树到树规则抽取

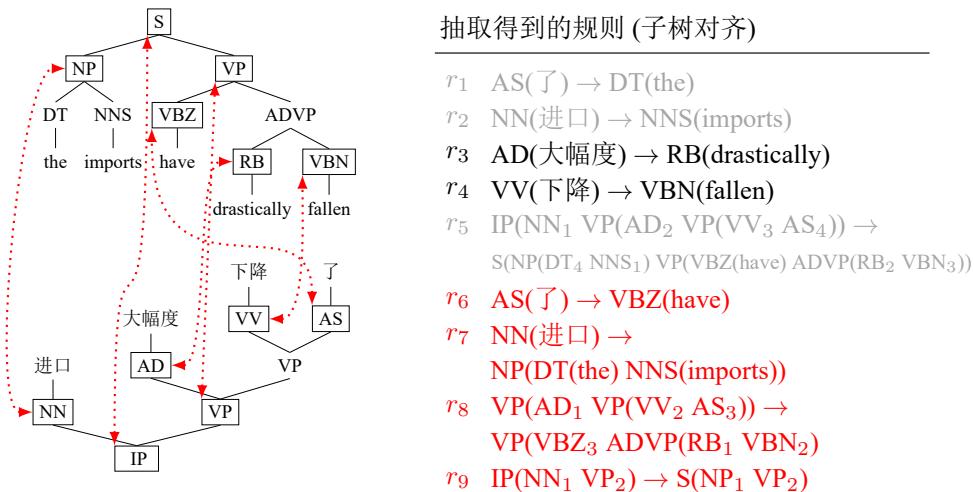


图 8.35 基于节点对齐的树到树规则抽取

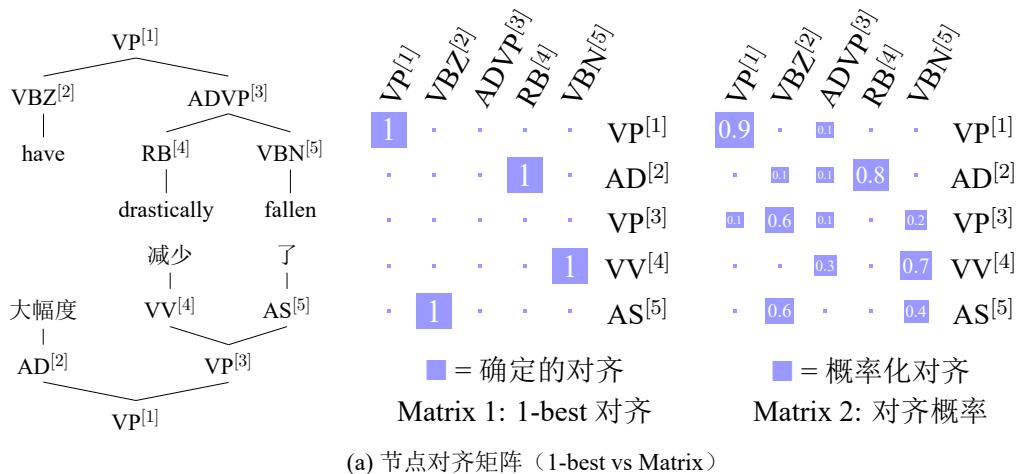
换一个角度来看，词对齐实际上只是帮助模型找到两种语言句法树中节点的对应关系。如果能够直接得到句法树节点的对应，就可以避免掉词对齐的错误。也就是，可以直接使用节点对齐来进行树到树规则的抽取。首先，利用外部的节点对齐工具获得两棵句法树节点之间的对齐关系。之后，将每个对齐的节点看作是树片段的根节点，再进行规则抽取。图8.35展示了基于节点对齐的规则抽取结果。

可以看到，节点对齐可以避免词对齐错误造成的影响。不过，节点对齐需要开发

额外的工具，有很多方法可以参考，比如可以基于启发性规则<sup>[353]</sup>、基于分类模型<sup>[354]</sup>、基于无指导的方法<sup>[248]</sup>等。

## 2. 基于对齐矩阵的规则抽取

同词对齐一样，节点对齐也会存在错误，这样就不可避免地造成规则抽取的错误。既然单一的对齐中含有错误，那能否让系统看到更多样的对齐结果，进而提高正确规则被抽取到的几率呢？答案是肯定的。实际上，在基于短语的模型中就有基于多个词对齐（如  $n$ -best 词对齐）进行规则抽取的方法<sup>[355]</sup>，这种方法可以在一定程度上提高短语的召回率。在树到树规则抽取中也可以使用多个节点对齐结果进行规则抽取。但是，简单使用多个对齐结果会使系统运行代价线性增长，而且即使是  $n$ -best 对齐，也无法保证涵盖到正确的对齐结果。对于这个问题，另一种思路是使用对齐矩阵进行规则的“软”抽取。



### 最小规则

#### Matrix 1 (基于 1-best 对齐)

- $r_3 \quad AD(\text{大幅度}) \rightarrow RB(\text{drastically})$
- $r_4 \quad VV(\text{减少}) \rightarrow VBN(\text{fallen})$
- $r_6 \quad AS(\text{了}) \rightarrow VBZ(\text{have})$
- $r_8 \quad VP(AD_1 VP(VV_2 AS_3)) \rightarrow VP(VBZ_3 ADVP(RB_1 VBN_2))$

### 最小规则

#### Matrix 2 (基于对齐概率)

- $r_3 \quad AD(\text{大幅度}) \rightarrow RB(\text{drastically})$
- $r_4 \quad VV(\text{减少}) \rightarrow VBN(\text{fallen})$
- $r_6 \quad AS(\text{了}) \rightarrow VBZ(\text{have})$
- $r_8 \quad VP(AD_1 VP(VV_2 AS_3)) \rightarrow VP(VBZ_3 ADVP(RB_1 VBN_2))$
- $r_{10} \quad VP(VV(\text{减少}) AS(\text{了})) \rightarrow VBN(\text{fallen})$
- $r_{11} \quad VP(AD_1 VP_2) \rightarrow VP(VBZ_1 ADVP_2)$
- ...

(b) 抽取得到的树到树翻译规则

图 8.36 使用 1-best 节点对齐和概率化节点对齐矩阵的树到树规则抽取<sup>[248]</sup>

所谓对齐矩阵，是描述两个句法树节点之间对应强度的数据结构。矩阵的每个单元中都是一个 0 到 1 之间的数字。规则抽取时，可以认为所有节点之间都存在对齐，这样可以抽出很多  $n$ -best 对齐中无法覆盖的规则。图 8.36 展示了一个用对齐矩

阵的进行规则抽取的实例。其中矩阵 1 (Matrix 1) 表示的是标准的 1-best 节点对齐, 矩阵 2 (Matrix 2) 表示的是一种概率化的对齐矩阵。可以看到使用矩阵 2 可以抽取到更多样的规则。另外, 值得注意的是, 基于对齐矩阵的方法也同样适用于短语和层次短语规则的抽取。关于对齐矩阵的生成可以参考相关论文的内容<sup>[248, 354, 355, 356]</sup>。

此外, 在基于句法的规则抽取中, 一般会对规则进行一些限制, 以避免规则数量过大, 系统无法处理。比如, 可以限制树片段的深度、变量个数、规则组合的次数等等。这些限制往往需要根据具体任务进行设计和调整。

### 8.3.5 句法翻译模型的特征

基于语言学句法的翻译模型使用判别式模型对翻译推导进行建模 (第七章数学建模小节)。给定双语句对  $(s, t)$ , 由  $M$  个特征经过线性加权, 得到每个翻译推导  $d$  的得分, 记为  $\text{score}(d, t, s) = \sum_{i=1}^M \lambda_i \cdot h_i(d, t, s)$ , 其中  $\lambda_i$  表示特征权重,  $h_i(d, t, s)$  表示特征函数。翻译的目标就是要找到使  $\text{score}(d, t, s)$  达到最高的推导  $d$ 。

这里, 可以使用最小错误率训练对特征权重进行调优 (第七章最小错误率训练小节)。而特征函数可参考如下定义:

**基于短语的特征** (对应于每条规则  $r : \langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ )

- ( $h_{1-2}$ ) 短语翻译概率 (取对数), 即规则源语言和目标语言树覆盖的序列翻译概率。令函数  $\tau(\cdot)$  返回一个树片段的叶子节点序列。对于规则:

$$\text{VP}(\text{PP}_1 \text{ VP}(\text{VV(表示)} \text{ NN}_2)) \rightarrow \text{VP}(\text{VBZ(was)} \text{ VP}(\text{VBN}_2 \text{ PP}_1))$$

可以得到:

$$\begin{aligned}\tau(\alpha_r) &= \text{PP 表示 NN} \\ \tau(\beta_r) &= \text{was VBN PP}\end{aligned}$$

于是, 可以定义短语翻译概率特征为  $\log(P(\tau(\alpha_r)|\tau(\beta_r)))$  和  $\log(P(\tau(\beta_r)|\tau(\alpha_r)))$ 。它们的计算方法与基于短语的系统是完全一样的<sup>9</sup>:

- ( $h_{3-4}$ ) 词汇化翻译概率 (取对数), 即  $\log(P_{\text{lex}}(\tau(\alpha_r)|\tau(\beta_r)))$  和  $\log(P_{\text{lex}}(\tau(\beta_r)|\tau(\alpha_r)))$ 。这两个特征的计算方法与基于短语的系统也是一样的。

**基于句法的特征** (对应于每条规则  $r : \langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ )

- ( $h_5$ ) 基于根节点句法标签的规则生成概率 (取对数), 即  $\log(P(r|\text{root}(r)))$ 。这里,  $\text{root}(r)$  是规则所对应的双语根节点  $(\alpha_h, \beta_h)$ ;
- ( $h_6$ ) 基于源语言端的规则生成概率 (取对数), 即  $\log(P(r|\alpha_r))$ , 给定源语言端

<sup>9</sup>对于树到串规则,  $\tau(\beta_r)$  就是规则目标语言端的符号串。

生成整个规则的概率；

- ( $h_7$ ) 基于目标语言端的规则生成概率（取对数），即  $\log(P(r|\beta_r))$ ，给定目标语言端生成整个规则的概率。

### 其他特征（对应于整个推导 $d$ ）

- ( $h_8$ ) 语言模型得分（取对数），即  $\log(P_{lm}(t))$ ，用于度量译文的流畅度；
- ( $h_9$ ) 译文长度，即  $|t|$ ，用于避免模型过于倾向生成短译文（因为短译文语言模型分数高）；
- ( $h_{10}$ ) 翻译规则数量，学习对使用规则数量的偏好。比如，如果这个特征的权重较高，则表明系统更喜欢使用数量多的规则；
- ( $h_{11}$ ) 组合规则的数量，学习对组合规则的偏好；
- ( $h_{12}$ ) 词汇化规则的数量，学习对含有终结符规则的偏好；
- ( $h_{13}$ ) 低频规则的数量，学习对训练数据中出现频次低于 3 的规则的偏好。低频规则大多不可靠，设计这个特征的目的也是为了区分不同质量的规则。

## 8.3.6 基于超图的推导空间表示

在完成建模后，剩下的问题是：如何组织这些翻译推导，高效地完成模型所需的计算？本质上，基于句法的机器翻译与句法分析是一样的，因此关于翻译推导的组织可以借用句法分析中的一些概念。

在句法分析中，上下文无关文法（CFG）的分析过程可以被组织成一个叫**有向超图**（Directed Hyper-graph）的结构，或者简称为**超图**<sup>[357]</sup>：

### 定义 8.3.6 有向超图

一个有向超图  $G$  包含一个节点集合  $N$  和一个有向**超边**（Hyper-edge）集合  $E$ 。每个有向超边包含一个头（Head）和一个尾（Tail），头指向  $N$  中的一个节点，尾是若干个  $N$  中的节点所构成的集合。

与传统的有向图不同，超图中的每一个边（超边）的尾可以包含多个节点。也就是说，每个超边从若干个节点出发最后指向同一个节点。这种定义完美契合了 CFG 的要求。比如，如果把节点看作是一个推导所对应树结构的根节点（含有句法标记），那么每个超边就可以表示一条 CFG 规则。

图8.37就展示了一个简单的超图。其中每个节点都有一个句法标记，句法标记下面记录了这个节点的跨度。超边 edge1 和 edge2 分别对应了两条 CFG 规则：

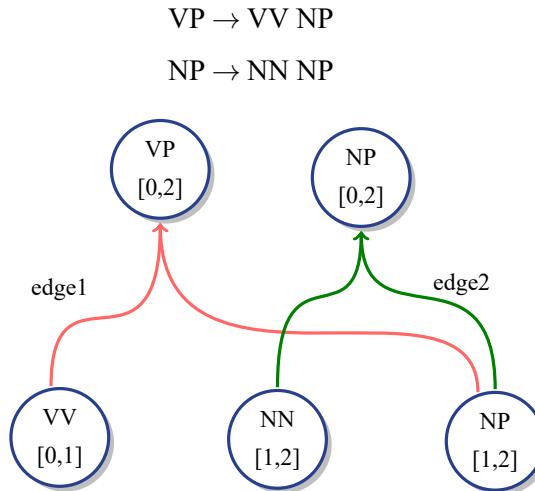


图 8.37 超图实例

对于规则“ $VP \rightarrow VV\ NP$ ”，超边的头指向  $VP$ ，超边的尾表示规则右部的两个变量  $VV$  和  $NP$ 。规则“ $NP \rightarrow NN\ NP$ ”也可以进行类似的解释。

不难发现，超图提供了一种非常紧凑的数据结构来表示多个推导，因为不同推导之间可以共享节点。如果把图8.37中的蓝色和红色部分看作是两个推导，那么它们就共享了同一个节点  $NN[1,2]$ ，其中  $NN$  是句法标记， $[1,2]$  是跨度。能够想象，简单枚举一个句子所有的推导几乎是不可能的，但是用超图的方式却可以很有效地对指数级数量的推导进行表示。另一方面，超图上的运算常常被看作是一种基于半环的代数系统，而且人们发现许多句法分析和机器翻译问题本质上都是**半环分析**（Semi-ring Parsing）。不过，由于篇幅有限，这里不会对半环等结构展开讨论。感兴趣的读者可以查阅相关文献<sup>[358, 359]</sup>。

从句法分析的角度看，超图最大程度地复用了局部的分析结果，使得分析可以“结构化”。比如，有两个推导：

$$d_1 = r_1 \circ r_2 \circ r_3 \circ r_4 \quad (8.9)$$

$$d_2 = r_1 \circ r_2 \circ r_3 \circ r_5 \quad (8.10)$$

其中， $r_1 - r_5$  分别表示不同的规则。 $r_1 \circ r_2 \circ r_3$  是两个推导的公共部分。在超图表示中， $r_1 \circ r_2 \circ r_3$  可以对应一个子图，显然这个子图也是一个推导，记为  $d' = r_1 \circ r_2 \circ r_3$ 。这样， $d_1$  和  $d_2$  不需要重复记录  $r_1 \circ r_2 \circ r_3$ ，重新写作：

$$d_1 = d' \circ r_4 \quad (8.11)$$

$$d_1 = d' \circ r_5 \quad (8.12)$$

引入  $d'$  的意义在于，整个分析过程具有了递归性。从超图上看， $d'$  可以对应以一个（或几个）节点为“根”的子图，因此只需要在这个（或这些）子图上增加新的超边就可以得到更大的推导。这个过程不断执行，最终完成对整个句子的分析。

在句法分析中，超图的结构往往被组织为一种**表格**（Chart）结构。表格的每个单元代表了一个跨度，因此可以把所有覆盖这个跨度的推导都放入相应的**表格单元**（Chart Cell）。对于上下文无关文法，表格里的每一项还会增加一个句法标记，用来区分不同句法功能的推导。

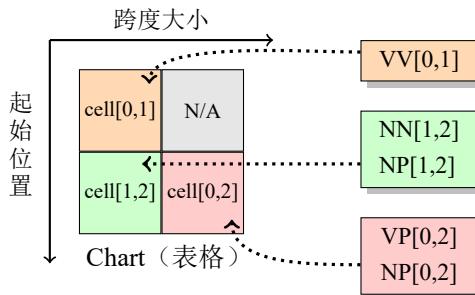


图 8.38 句法分析 Chart（表格）结构实例

如图8.38所示，覆盖相同跨度的节点会被放入同一个表格单元，但是不同句法标记的节点会被看作是不同的项（Item）。这种组织方式建立了一个索引，通过索引可以很容易地访问同一个跨度下的所有推导。比如，如果采用自下而上的分析，可以从小跨度的表格单元开始，构建推导，并填写表格单元。这个过程中，可以访问之前的表格单元来获得所需的局部推导（类似于前面提到的  $d'$ ）。该过程重复执行，直到处理完最大跨度的表格单元。而最后一个表格单元就保存了完整推导的根节点。通过回溯的方式，能够把所有推导都生成出来。

基于句法的机器翻译仍然可以使用超图进行翻译推导的表示。和句法分析一样，超图的每条边可以对应一个基于树结构的文法，超边的头代表文法的左部，超边的尾代表规则中变量所对应的超图中的节点<sup>10</sup>。图8.39给出了一个使用超图来表示机器翻译推导的实例。可以看到，超图的结构是按源语言组织的，但是每个规则（超边）会包含目标语言的信息。由于同步翻译文法可以确保规则的源语言端和目标语言端都覆盖连续的词串，因此超图中的每个节点都对应一个源语言跨度，同时对应一个目标语的连续译文。这样，每个节点实际上代表了一个局部的翻译结果。

不过，机器翻译与句法分析也有不同之处。最主要的区别在于机器翻译使用了语言模型作为一个特征，比如  $n$ -gram 语言模型。因为语言模型并不是上下文无关的，因此机器翻译中计算最优推导的方法和句法分析会有不同。常用的方法是，直接在每个表格单元中融合语言模型的分数，保留前  $k$  个结果；或者，在构建超图时不计算语言模型得分，等到构建完整个超图之后对最好的若干个推导用语言模型重新排序；

<sup>10</sup> 也可以把每个终结符看作是一个节点，这样一个超边的尾就对应规则的树片段中所有的叶子。

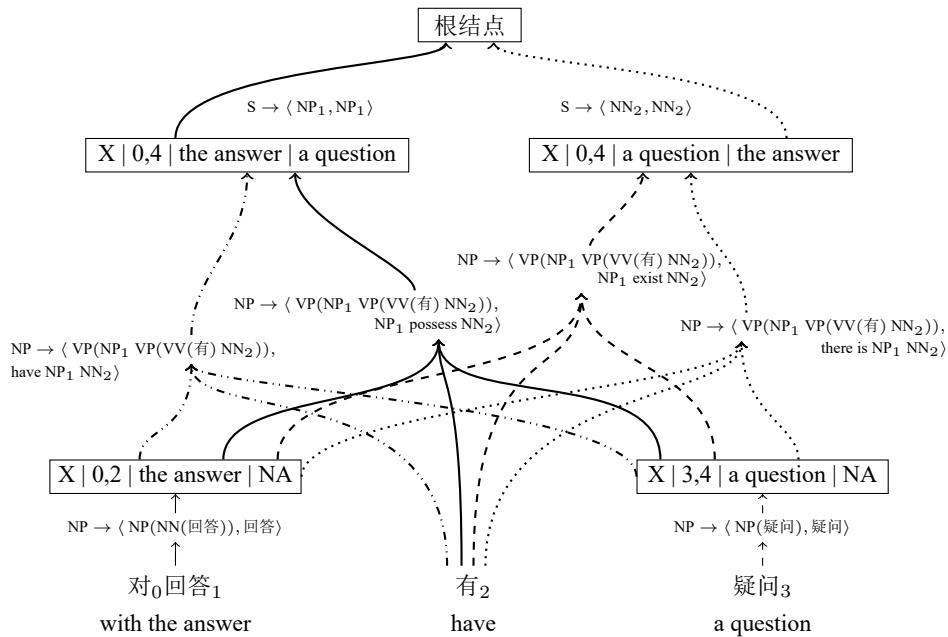


图 8.39 机器翻译推导的超图表示

再或者，将译文和语言模型都转化为加权有限状态自动机，之后直接对两个自动机做**组合**（Composition）得到新的自动机，最后得到融合语言模型得分的译文表示。

基于超图的推导表示方法有着很广泛的应用。比如，8.2节介绍的层次短语系统也可以使用超图进行建模，因为它也使用了同步文法。从这个角度说，基于层次短语的模型和基于语言学句法的模型本质上是一样的。它们的主要区别在于规则中的句法标记和抽取规则的方法不同。

### 8.3.7 基于树的解码 vs 基于串的解码

解码的目标是找到得分  $\text{score}(d)$  最高的推导  $d$ 。这个过程通常被描述为：

$$\hat{d} = \arg \max_d \text{score}(d, s, t) \quad (8.13)$$

这也是一种标准的**基于串的解码**（String-based Decoding），即通过句法模型对输入的源语言句子进行翻译得到译文串。不过，搜索所有的推导会导致巨大的解码空间。对于树到串和树到树翻译来说，源语言句法树是可见的，因此可以使用另一种解码方法——**基于树的解码**（Tree-based Decoding），即把输入的源语句法树翻译为目标语串。

表8.4对比了基于串和基于树的解码方法。可以看到，基于树的解码只考虑了与源语言句法树兼容的推导，因此搜索空间更小，解码速度会更快。

这里需要注意的是，不论是基于串的解码还是基于树的解码都是使用句法模型

表 8.4 基于串的解码 vs 基于树的解码

对比	基于树的解码	基于串的解码
解码方法	$\hat{d} = \arg \max_{d \in D_{\text{tree}}} \text{score}(d)$	$\hat{d} = \arg \max_{d \in D} \text{score}(d)$
搜索空间	与输入的源语句法树兼容的推导 $D_{\text{tree}}$	所有的推导 $D$
适用模型	树到串、树到树	所有的基于句法的模型
解码算法	Chart 解码	CKY + 规则二叉化
速度	快	一般较慢

的方法，在翻译过程中都会生成翻译推导和树结构。二者的本质区别在于，基于树的解码把句法树作为显性的输入，而基于串的解码把句法树看作是翻译过程中的隐含变量。图8.40进一步解释了这个观点。

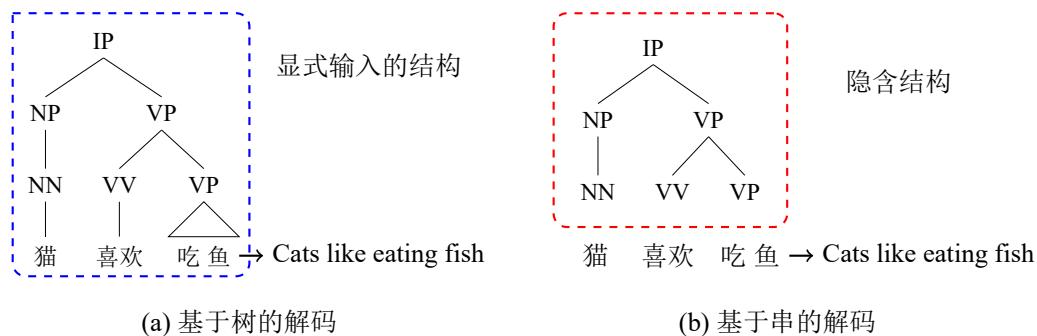


图 8.40 句法树在不同解码方法中的角色

## 1. 基于树的解码

基于树和基于串的解码都可以使用前面的超图结构进行推导的表示。基于树的解码方法相对简单。直接使用表格结构组织解码空间即可。这里采用自底向上的策略，具体步骤如下：

- 从源语言句法树的叶子节点开始，自下而上访问输入句法树的节点；
- 对于每个树节点，匹配相应的规则；
- 从树的根节点可以得到翻译推导，最终生成最优推导所对应的译文。

这个过程如图8.41所示，可以看到，不同的表格单元对应不同跨度，每个表格单元会保存相应的句法标记（还有译文的信息）。

这里的问题在于规则匹配。对于每个树节点，需要知道以它为根可以匹配的规则有哪些。比较直接的解决方法是遍历这个节点下一定深度的句法树片段，用每个树片段在文法中找出相应的匹配规则，如图8.42所示。不过这种匹配是一种严格匹

The diagram illustrates the Chart and its corresponding parse tree for the sentence '猫喜欢吃什么鱼'. The Chart is a grid where columns represent tokens and rows represent spans. The grid has 4 columns labeled l=1 to l=4 and 4 rows labeled 1 to 4. The first row contains '猫' at l=1. The second row contains '喜欢' at l=1 and l=2. The third row contains '吃' at l=1 and l=2, and '鱼' at l=3 and l=4. The fourth row is empty. Below the grid, the tokens are listed as '猫 喜欢 吃 鱼' with indices 0, 1, 2, 3, 4 below them. To the right is a table mapping spans to sequence numbers and their corresponding labels.

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,3]	VV	吃
4	[3,4]	NN & NP	鱼
5	[0,2]	N/A	猫喜欢
6	[1,3]	N/A	喜欢吃
7	[2,4]	VP	吃鱼
8	[0,3]	N/A	猫喜欢吃
9	[1,4]	VP	喜欢吃鱼
10	[0,4]	IP (root)	猫喜欢吃鱼

图 8.41 基于树的解码中 Chart 的内容

配，因为它要求句法树片段内的所有内容都要与规则的源语言部分严格对应。有时，句法结构中的细微差别都会导致规则匹配不成功。因此，也可以考虑采用模糊匹配的方式提高规则的命中率，进而增加可以生成推导的数量<sup>[360]</sup>。

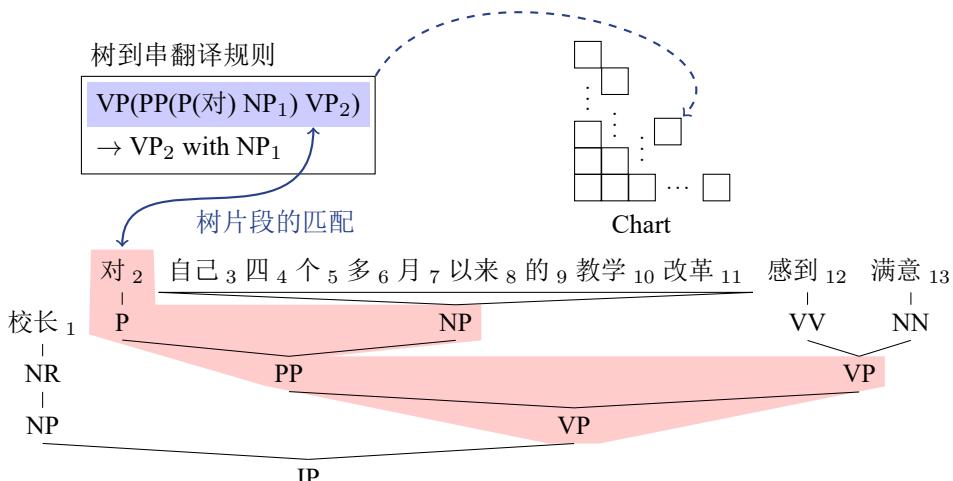


图 8.42 基于树的规则匹配

## 2. 基于串的解码

基于串的解码过程和句法分析几乎一样。对于输入的源语言句子，基于串的解码需要找到这个句子上的最优推导。唯一不同的地方在于，机器翻译需要考虑译文的生成（语言模型的引入会使问题稍微复杂一些），但是源语言部分的处理和句法分析是一样的。因为不要求用户输入句法树，所以这种方法同时适用于树到串、串到树、树到树等多种模型。本质上，基于串的解码可以探索更多潜在的树结构，并增大搜索空间（相比基于树的解码），因此该方法更有可能找到高质量翻译结果。

基于串的解码仍然可以用表格结构来组织翻译推导。不过，一个比较有挑战的

问题是如何找到每个规则能够匹配的源语言跨度。也就是，对于每个表格单元，需要知道哪些规则可以被填入其中。因为，没有用户输入的句法树做指导，理论上输入句子的所有子串要与所有规则进行匹配。匹配时，需要考虑规则中源语言端的符号串（或者树结构的叶子序列）与输入词串匹配的全部可能性。

在跨度 [0,13] 上匹配“NP 对 NP VP”

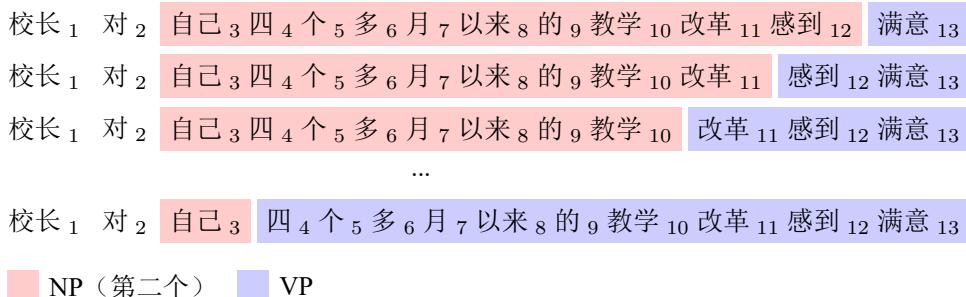


图 8.43 在一个词串上匹配“NP 对 NP VP”。连续变量的匹配对应了对词串不同位置的切割。

图8.43展示了规则匹配输入句子（包含 13 个词）的所有可能。可以看到，规则源语言端的连续变量会使得匹配情况变得复杂。对于长度为  $n$  的词串，匹配含有  $m$  个连续变量的规则的时间复杂度是  $O(n^{m-1})$ 。显然当变量个数增加时规则匹配是相当耗时的操作，甚至当变量个数过多时解码无法在可接受的时间内完成。

对于这个问题，有两种常用的解决办法：

- 对文法进行限制。比如，可以限制规则中变量的数量；或者不允许连续的变量，这样的规则也被称作满足**词汇化标准形式**（Lexicalized Norm Form）（LNF）的规则。比如，层次短语规则就是 LNF 规则。由于 LNF 中单词（终结符）可以作为锚点，因此规则匹配时所有变量的匹配范围是固定的；
- 对规则进行二叉化，使用 CKY 方法进行分析。这个方法也是句法分析中常用的策略。所谓规则二叉化是把规则转化为最多只含两个变量或连续词串的规则（串到树规则）。比如，对于如下的规则：

$$\text{喜欢 } VP_1 \ NP_2 \rightarrow VP(VBZ(\text{likes})) \ VP_1 \ NP_2$$

二叉化的结果为：

$$\text{喜欢 } V103 \rightarrow VP(VBZ(\text{likes})) \ V103$$

$$VP_1 \ NP_2 \rightarrow V103( VP_1 \ NP_2 )$$

可以看到，这两条新的规则中源语言端只有两个部分，代表两个分叉。V103 是

一个新的标签，它没有任何句法含义。不过，为了保证二叉化后规则目标语部分的连续性，需要考虑源语言和目标语二叉化的同步性<sup>[349, 350]</sup>。这样的规则与 CKY 方法一起使用完成解码，具体内容可以参考 8.2.4 节的内容。

总的来说，基于句法的解码器较为复杂，无论是算法的设计还是工程技巧的运用，对开发者的能力都有一定要求。因此开发一个优秀的基于句法的机器翻译系统是一项有挑战的工作。

## 8.4 小结及拓展阅读

自基于规则的方法开始，如何使用句法信息就是机器翻译研究人员关注的热点。在统计机器翻译时代，句法信息与机器翻译的结合成为了最具时代特色的研究方向之一。句法结构具有高度的抽象性，因此可以缓解基于词串方法不善于处理句子上层结构的问题。

本章对基于句法的机器翻译模型进行了介绍，并重点讨论了相关的建模、翻译规则抽取以及解码问题。从某种意义上说，基于句法的模型与基于短语的模型都同属一类模型，因为二者都假设：两种语言间存在由短语或者规则构成的翻译推导，而机器翻译的目标就是找到最优的翻译推导。但是，由于句法信息有其独特的性质，因此也给机器翻译带来了新的问题。有几方面问题值得关注：

- 从建模的角度看，早期的统计机器翻译模型已经涉及到了树结构的表示问题<sup>[281, 361]</sup>。

不过，基于句法的翻译模型的真正崛起是在同步文法提出之后。初期的工作大多集中在反向转录文法和括号转录文法方面<sup>[341, 362, 363]</sup>，这类方法也被用于短语获取<sup>[364, 365]</sup>。进一步，研究者提出了更加通用的层次模型来描述翻译过程<sup>[88, 366, 367]</sup>，本章介绍的层次短语模型就是其中典型的代表。之后，使用语言学句法的模型也逐渐兴起。最具代表性的是在单语言端使用语言学句法信息的模型<sup>[86, 87, 346, 368, 369, 370, 371]</sup>，即：树到串翻译模型和串到树翻译模型。值得注意的是，除了直接用句法信息定义翻译规则，也有研究者将句法信息作为软约束改进层次短语模型<sup>[372, 373]</sup>。这类方法具有很大的灵活性，既保留了层次短语模型比较健壮的特点，同时也兼顾了语言学句法对翻译的指导作用。在同一时期，也有研究者提出同时使用双语两端的语言学句法树对翻译进行建模，比较有代表性的工作是使用同步树插入文法（Synchronous Tree-Insertion Grammars）和同步树替换文法（Synchronous Tree-Substitution Grammars）进行树到树翻译的建模<sup>[352, 374, 375]</sup>。不过，树到树翻译假设两种语言间的句法结构能够相互转换，而这个假设并不总是成立。因此树到树翻译系统往往要配合一些技术，如树二叉化，来提升系统的健壮性。

- 在基于句法的模型中，常常会使用句法分析器完成句法分析树的生成。由于句法分析器会产生错误，因此这些错误会对机器翻译系统产生影响。对于这个问题，一种解决办法是同时考虑更多的句法树，从而增加正确句法分析结果被使用到的概率。其中，比较典型的方式基于句法森林的方法<sup>[376, 377]</sup>，比如，在规

则抽取或者解码阶段使用句法森林，而不是仅仅使用一棵单独的句法树。另一种思路是，对句法结构进行松弛操作，即在翻译的过程中并不严格遵循句法结构<sup>[360, 378]</sup>。实际上，前面提到的基于句法软约束的模型也是这类方法的一种体现<sup>[372, 373]</sup>。事实上，机器翻译领域长期存在一个问题：使用什么样的句法结构最适合机器翻译？因此，有研究者尝试对比不同的句法分析结果对机器翻译系统的影响<sup>[379, 380]</sup>。也有研究者面向机器翻译任务自动归纳句法结构<sup>[381]</sup>，而不是直接使用从单语小规模树库学习到的句法分析器，这样可以提高系统的健壮性。

- 本章所讨论的模型大多基于短语结构树。另一个重要的方向是使用依存树进行翻译建模<sup>[382, 383, 384]</sup>。依存树比短语结构树有更简单的结构，而且依存关系本身也是对“语义”的表征，因此也可以捕捉到短语结构树所无法涵盖的信息。同其它基于句法的模型类似，基于依存树的模型大多也需要进行规则抽取、解码等步骤，因此这方面的研究工作大多涉及翻译规则的抽取、基于依存树的解码等<sup>[385, 386, 387, 388, 389]</sup>。此外，基于依存树的模型也可以与句法森林结构相结合，对系统性能进行进一步提升<sup>[390, 391]</sup>。
- 不同模型往往有不同的优点，为了融合这些优点，系统融合是很受关注的研究方向。某种意义上说，系统融合的兴起源于本世纪初各种机器翻译比赛。因为当时提升翻译性能的主要方法之一就是将多个翻译引擎进行融合。系统融合的出发点是：多样的翻译候选有助于生成更好的译文。系统融合的思路很多，比较简单的方法是假设选择（Hypothesis Selection），即从多个翻译系统的输出中直接选择一个译文<sup>[392, 393, 394]</sup>；另一种方法是用多个系统的输出构建解码格（Decoding Lattice）或者混淆网络（Confusion Networks），这样可以生成新的翻译结果<sup>[395, 396, 397]</sup>；此外，还可以在解码过程中动态融合不同模型<sup>[398, 399]</sup>。另一方面，也有研究者探讨如何在一个翻译系统中让不同的模型进行互补，而不是简单的融合。比如，可以控制句法在机器翻译中使用的程度，让句法模型和层次短语模型处理各自擅长的问题<sup>[400]</sup>。
- 语言模型是统计机器翻译系统所使用的重要特征。但是，即使引入  $n$ -gram 语言模型，机器翻译系统仍然会产生语法上不正确的译文，甚至会生成结构完全错误的译文。对于这个问题，研究者尝试使用基于句法的语言模型。早期的探索有 Charniak 等人<sup>[401]</sup> 和 Och 等人<sup>[311]</sup> 的工作，不过当时的结果并没有显示出基于句法的语言模型可以显著提升机器翻译的品质。后来，BBN 的研究团队提出了基于依存树的语言模型<sup>[402]</sup>，这个模型可以显著提升层次短语模型的性能。除此之外，也有研究工作探索基于树替换文法等结构的语言模型<sup>[403]</sup>。实际上，树到树、串到树模型也可以被看作是一种对目标语言句法合理性的度量，只不过目标语言的句法信息被隐含在翻译规则中。这时，可以在翻译规则上设计相应的特征，以达到引入目标语句法语言模型的目的。



# 神经机器翻译

<b>9</b>	<b>人工神经网络和神经语言建模</b>	<b>269</b>
9.1	深度学习与人工神经网络	
9.2	神经网络基础	
9.3	神经网络的张量实现	
9.4	神经网络的参数训练	
9.5	神经语言模型	
9.6	小结及拓展阅读	
<b>10</b>	<b>基于循环神经网络的模型</b>	<b>329</b>
10.1	神经机器翻译的发展简史	
10.2	编码器-解码器框架	
10.3	基于循环神经网络的翻译建模	
10.4	注意力机制	
10.5	训练及推断	
10.6	小结及拓展阅读	
<b>11</b>	<b>基于卷积神经网络的模型</b>	<b>371</b>
11.1	卷积神经网络	
11.2	基于卷积神经网络的翻译建模	
11.3	局部模型的改进	
11.4	小结及拓展阅读	
<b>12</b>	<b>基于自注意力的模型</b>	<b>391</b>
12.1	自注意力机制	
12.2	Transformer 架构	
12.3	位置编码	
12.4	基于点乘的多头注意力机制	
12.5	残差网络和层标准化	
12.6	前馈全连接网络子层	
12.7	训练	
12.8	推断	
12.9	小结及拓展阅读	





## 9. 人工神经网络和神经语言建模

**人工神经网络**（Artificial Neural Networks）或**神经网络**（Neural Networks）是描述客观世界的一种数学模型。这种模型和生物学上的神经系统在行为上有一些相似之处，但是人们更多的是把它作为一种计算工具，而非一个生物学模型。近些年，随着机器学习领域的快速发展，人工神经网络被大量使用在对图像和自然语言的处理上。特别是，研究人员发现深层神经网络可以被成功训练后，学术界也逐渐形成了一种新的机器学习范式——**深度学习**（Deep Learning）。可以说，深度学习是近几年最受瞩目的研究领域之一，其应用也十分广泛。比如，图像识别的很多重要进展都来自深度学习模型的使用。包括机器翻译在内的很多自然语言处理任务中，深度学习也已经成为了一种标准模型。基于深度学习的表示学习方法也为自然语言处理开辟了新的思路。

本章将对深度学习的概念和技术进行介绍，目的是为本书后面神经机器翻译的内容进行铺垫。此外，本章也会对深度学习在语言建模方面的应用进行介绍，以便读者可以初步了解如何使用深度学习方法描述自然语言处理问题。

### 9.1 深度学习与人工神经网络

深度学习是机器学习研究中一个非常重要的分支，其概念来源于对人工神经网络的研究：通过人工神经元之间的连接建立一种数学模型，使计算机可以像人一样进行分析、学习和推理。

近几年来，随着深度学习技术的广泛传播与使用，“人工智能”这个名词在有些

场合下甚至与“深度学习”划上了等号。这种理解非常片面，比较准确地说，“深度学习”是实现“人工智能”的一种技术手段。但从这种现象中，深度学习的火爆情况可见一斑。深度学习的技术浪潮以惊人的速度席卷世界，也改变了很多领域的现状，在数据挖掘、自然语言处理、语音识别、图像识别等各个领域随处可见深度学习的身影。自然语言处理领域中，深度学习在很多任务中已经取得令人震撼的效果。特别是，基于深度学习的表示学习方法已经成为自然语言处理的新范式，在机器翻译任务中更是衍生出了“神经机器翻译”这样全新的模型。

### 9.1.1 发展简史

神经网络最早出现在控制论中，随后更多地在连接主义中被提及。神经网络被提出的初衷并不是做一个简单的计算模型，而是希望将神经网络应用到一些自动控制相关的场景中。然而随着神经网络技术的持续发展，神经网络方法已经被广泛应用到各行各业的研究和实践工作中。

人工神经网络诞生至今，经历了多次高潮和低谷，这是任何一种技术都无法绕开的命运。然而，好的技术和方法终究不会被埋没，直到今天，神经网络和深度学习迎来了最好的时代。

#### 1. 早期的人工神经网络和第一次寒冬

最初，神经网络设计的初衷是用计算模型来模拟生物大脑中神经元的运行机理，这种想法哪怕是现在看来也是十分超前的。例如，目前很多机构关注的概念——“类脑计算”就是希望研究人脑的运行机制及相关的计算机实现方法。然而模拟大脑这件事并没有想象中的那么简单，众所周知，生物学中对人脑机制的研究是十分困难的。因此，神经网络技术一直在摸索着前行，发展到现在，其计算过程与人脑的运行机制已经大相径庭。

人工神经网络的第一个发展阶段是在二十世纪 40 年代到 70 年代，这个时期的人工神经网络还停留在利用线性模型模拟生物神经元的阶段。虽然，线性模型在现在看来可能比较“简陋”，但是这类模型对后来的随机梯度下降等经典方法产生了深远影响。不过，显而易见的是，这种结构也存在着非常明显的缺陷，单层结构限制了它的学习能力，使它无法描述非线性问题，如著名的异或函数（XOR）学习问题。此后，神经网络的研究陷入了很长一段时间的低迷期。

#### 2. 神经网络的第二次高潮和第二次寒冬

虽然第一代神经网络受到了打击，但是在 20 世纪 80 年代，第二代人工神经网络开始萌发新的生机。在这个发展阶段，生物属性已经不再是神经网络的唯一灵感来源，在**连接主义**（Connectionism）和分布式表示两种思潮的影响下，神经网络方法再次走入了人们的视线。

##### 1) 符号主义与连接主义

人工智能领域始终存在着符号主义和连接主义之争。早期的人工智能研究在认知学中被称为**符号主义** (Symbolicism)，符号主义认为人工智能源于数理逻辑，希望将世界万物的所有运转方式归纳成像文法一样符合逻辑规律的推导过程。符号主义的支持者们坚信基于物理符号系统（即符号操作系统）假设和有限合理性原理，就能通过逻辑推理来模拟智能。但被他们忽略的一点是，模拟智能的推理过程需要大量的先验知识支持，哪怕是在现代，生物学界也很难准确解释大脑中神经元的工作原理，因此也很难用符号系统刻画人脑逻辑。另一方面，连接主义则侧重于利用人工神经网络中神经元的连接去探索并模拟输入与输出之间存在的某种关系，这个过程不需要任何先验知识，其核心思想是“大量简单的计算单元连接到一起可以实现智能行为”，这种思想也推动了反向传播等多种神经网络方法的应用，并发展了包括长短时记忆模型在内的经典建模方法。2019年3月27日，ACM正式宣布将图灵奖授予Yoshua Bengio, Geoffrey Hinton和Yann LeCun，以表彰他们提出的概念和工作使得深度学习神经网络有了重大突破，这三位获奖人均是人工智能连接主义学派的主要代表，从这件事中也可以看出连接主义对当代人工智能和深度学习的巨大影响。

## 2) 分布式表示

分布式表示的主要思想是“一个复杂系统的任何部分的输入都应该是多个特征共同表示的结果”，这种思想在自然语言处理领域的影响尤其深刻，它改变了刻画语言世界的角度，将语言文字从离散空间映射到多维连续空间。例如，在现实世界中，“张三”这个代号就代表着一个人。如果想要知道这个人亲属都有谁，因为有“如果A和B姓氏相同且在同一个家谱中，那么A和B是本家”这个先验知识在，在知道代号“张三”的情况下，可以得知“张三”的亲属是谁。但是如果不能依靠这个先验知识，就无法得知“张三”的亲属是谁。但在分布式表示中，可以用一个实数向量，如 $(0.1, 0.3, 0.4)$ 来表示“张三”这个人，这个人的所有特征信息都包含在这个实数向量中，通过在向量空间中的一些操作（如计算距离等），哪怕没有任何先验知识的存在，也完全可以找到这个人的所有亲属。在自然语言处理中，一个单词也用一个实数向量（词向量或词嵌入）表示，通过这种方式将语义空间重新刻画，将这个离散空间转化成了一个连续空间，这时单词就不再是一个简单的词条，而是由成百上千个特征共同描述出来的，其中每个特征分别代表这个词的某个“方面”。

随着第二代人工神经网络的“脱胎换骨”，学者们又对神经网络方法燃起了希望之火，这也导致有些时候过分夸大了神经网络的能力。20世纪90年代后期，由于在语音识别、自然语言处理等应用中，人们对神经网络方法期望过高，但是结果并没有达到预期，这也让很多人丧失了对神经网络方法的信任。相反，核方法、图模型等机器学习方法取得了很好的效果，这导致神经网络研究又一次进入低谷。

## 3. 深度学习和神经网络方法的崛起

21世纪初，随着深度学习浪潮席卷世界，人工神经网络又一次出现在人们的视野中。深度学习的流行源于2006年Hinton等人成功训练了一个深度信念网络（Deep

Belief Network)，在深度神经网络方法完全不受重视的情况下，大家突然发现深度神经网络完全是一个魔鬼般的存在，可以解决很多当时其他方法无法解决的问题。神经网络方法终于在一次又一次的被否定后，迎来了它的春天。随之针对神经网络和深度学习的一系列研究前赴后继地展开了，延续至今。

回过头来看，现代深度学习的成功主要有三方面的原因：

- 第一，模型和算法的不断完善和改进。这是现代深度学习能够获得成功的最主要的原因；
- 第二，并行计算能力的提升使大规模的实践成为了可能。早期的计算机设备根本无法支撑深度神经网络训练所需要的计算量，导致实践变得十分困难。而设备的进步、计算能力的提升则彻底改变了这种窘境；
- 第三，以 Geoffrey Hinton 等人为代表的学者的坚持和持续努力。

另外，从应用的角度来看，数据量的快速提升和模型容量的增加也为深度学习的成功提供了条件，数据量的增加使得深度学习有了用武之地，例如，2000 年以来，无论在学术研究还是在工业实践中，双语数据的使用数量都在逐年上升（如图9.1所示）。现在的深度学习模型参数量都十分巨大，因此需要大规模数据才能保证模型学习的充分性，而大数据时代的到来为训练这样的模型提供了数据基础。

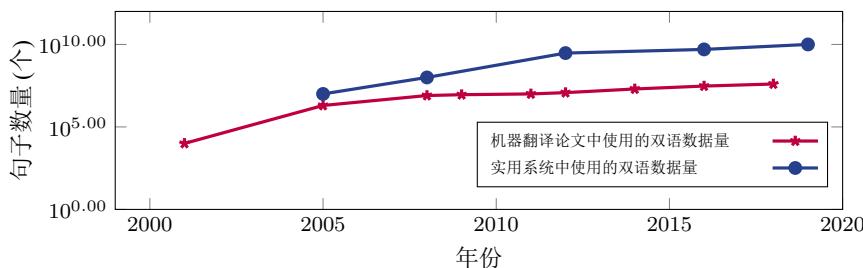


图 9.1 机器翻译系统所使用的双语数据量变化趋势

### 9.1.2 为什么需要深度学习

深度神经网络提供了一种简单的学习机制，即直接学习输入与输出的关系，通常把这种机制称为**端到端学习**（End-to-End Learning）。与传统方法不同，端到端学习并不需要人工定义特征或者进行过多的先验性假设，所有的学习过程都是由一个模型完成。从外面看这个模型只是建立了一种输入到输出的映射，而这种映射具体是如何形成的完全由模型的结构和参数决定。这样做的最大好处是，模型可以更加“自由”的进行学习。此外，端到端学习也引发了一个新的思考——如何表示问题？这也就是所谓的**表示学习**（Representation Learning）问题。在深度学习时代，问题输入和输出的表示已经不再是人类通过简单地总结得到的规律，而是可以让计算机自己进行描述的一种可计算“量”，比如一个实数向量。由于这种表示可以被自动学习，

因此也大大促进了计算机对语言文字等复杂现象的处理能力。

## 1. 端到端学习和表示学习

端到端学习使机器学习不再依赖传统的特征工程方法，因此也不需要繁琐的数据预处理、特征选择、降维等过程，而是直接利用人工神经网络自动从输入数据中提取、组合更复杂的特征，大大提升了模型能力和工程效率。以图9.2中的图像分类为例，在传统方法中，图像分类需要很多阶段的处理。首先，需要提取一些手工设计的图像特征，在将其降维之后，需要利用SVM等分类算法对其进行分类。与这种多阶段的流水线似的处理流程相比，端到端深度学习只训练一个神经网络，输入就是图片的像素表示，输出直接是分类类别。

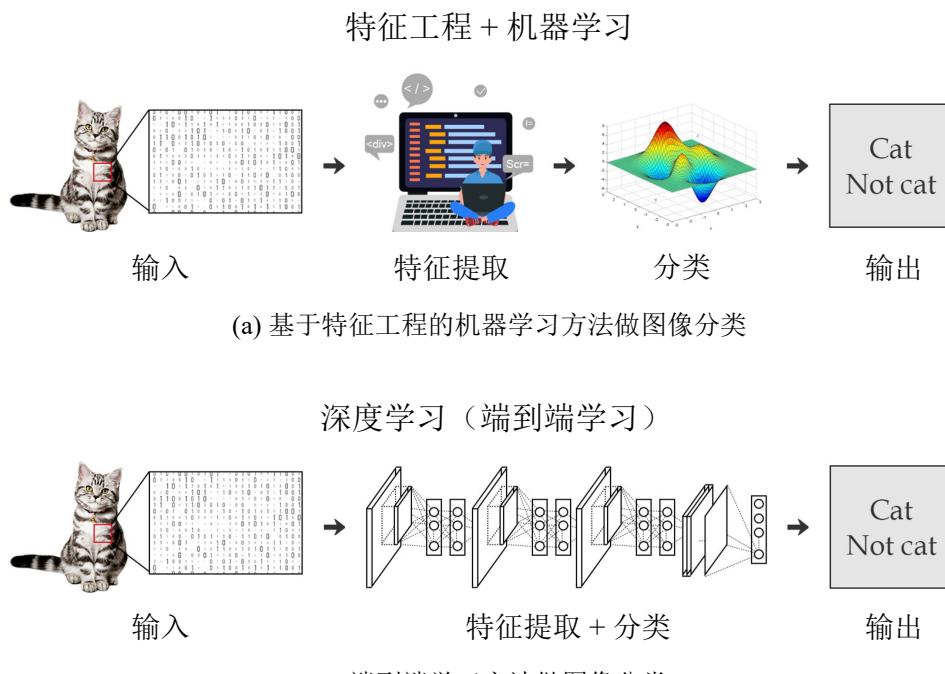


图 9.2 特征工程 **vs** 端到端学习

传统的机器学习需要人工定义特征，这个过程往往需要对问题的隐含假设。这种方法存在三方面的问题：

- **特征的构造需要耗费大量的时间和精力。**在传统机器学习的特征工程方法中，特征提取都是基于人力完成的，该过程往往依赖于大量的先验假设，会导致相关系统的研发周期也大大增加；
- **最终的系统性能强弱非常依赖特征的选择。**有一句话在业界广泛流传：“数据和特征决定了机器学习的上限”，但是人的智力和认知是有限的，因此人工设计的特征的准确性和覆盖度会存在瓶颈；

- **通用性差。**针对不同的任务，传统机器学习的特征工程方法需要选择出不同的特征，在某个任务上表现很好的特征在其他任务上可能没有效果。

端到端学习将人们从大量的特征提取工作之中解放出来，可以不需要太多人的先验知识。从某种意义上讲，对问题的特征提取完全是自动完成的，这也意味着即使系统开发者不是该任务的“专家”也可以完成相关系统的开发。此外，端到端学习实际上也隐含了一种新的对问题的表示形式——分布式表示。在这种框架下，模型的输入可以被描述为分布式的实数向量，这样模型可以有更多的维度描述一个事物，同时避免传统符号系统对客观事物离散化的刻画。比如，在自然语言处理中，表示学习重新定义了什么是词，什么是句子。在本章后面的内容中也会看到，表示学习可以让计算机对语言文字的描述更加准确和充分。

## 2. 深度学习的效果

相比于传统的基于特征工程的方法，基于深度学习的模型更加方便、通用，在系统性能上也普遍更优。这里以语言建模任务为例。语言建模的目的是开发一个模型来描述词串出现的可能性（见第二章）。这个任务已经有着很长时间的历史。表9.1给出了不同方法在常用的PTB数据集上的困惑度结果<sup>1</sup>。传统的n-gram语言模型由于面临维度灾难和数据稀疏问题，最终语言模型的性能并不是很好。而在深度学习模型中，通过引入循环神经网络等结构，所得到的语言模型可以更好地描述序列生成的问题。而最新的基于Transformer架构的语言模型将PPL从最初的178.0下降到了惊人的35.7。可见深度学习为这个任务带来的进步是巨大的。

表9.1 不同方法在PTB语言建模任务上的困惑度（PPL）

模型	作者	年份	PPL
3-gram LM <sup>[404]</sup>	Brown et al.	1992	178.0
Feed-forward Neural LM <sup>[72]</sup>	Bengio et al.	2003	162.2
Recurrent NN-based LM <sup>[73]</sup>	Mikolov et al.	2010	124.7
Recurrent NN-LDA <sup>[405]</sup>	Mikolov et al.	2012	92.0
LSTM <sup>[406]</sup>	Zaremba et al.	2014	78.4
RHN <sup>[407]</sup>	Zilly et al.	2016	65.4
AWD-LSTM <sup>[408]</sup>	Merity et al.	2018	58.8
GPT-2 (Transformer) <sup>[409]</sup>	Radford et al.	2019	35.7

<sup>1</sup> 困惑度越低表明语言建模的效果越好。

## 9.2 神经网络基础

神经网络是一种由大量的节点（或称神经元）之间相互连接构成的计算模型。那么什么是神经元？神经元之间又是如何连接的？神经网络的数学描述又是什么样的？这一节将围绕这些问题系统地对神经网络的基础知识进行介绍。

### 9.2.1 线性代数基础

线性代数作为一个数学分支，广泛应用于科学和工程中，神经网络的数学描述中也大量使用了线性代数工具。因此，这里对线性代数的一些概念进行简要介绍，以方便后续对神经网络进行数学描述。

#### 1. 标量、向量和矩阵

**标量** (Scalar)：标量亦称“无向量”，是一种只具有数值大小而没有方向的量，通俗地说，一个标量就是一个单独的数，这里特指实数<sup>2</sup>。比如，对于  $a = 5$ ,  $a$  就是一个标量。

**向量** (Vector)：向量是由一组实数组成的有序数组。与标量不同，向量既有大小也有方向。可以把向量看作空间中的点，每个元素是不同坐标轴上的坐标。公式(9.1)和公式(9.2)分别展示了一个行向量和一个列向量：

$$\mathbf{a} = \begin{pmatrix} 1 & 2 & 5 & 7 \end{pmatrix} \quad (9.1)$$

$$\mathbf{a}^T = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 7 \end{pmatrix} \quad (9.2)$$

本章默认使用行向量，如  $\mathbf{a} = (a_1, a_2, a_3)$ ,  $\mathbf{a}$  对应的列向量记为  $\mathbf{a}^T$ 。

**矩阵** (Matrix)：矩阵是一个按照长方阵列排列的实数集合，最早来自于方程组的系数及常数所构成的方阵。在计算机领域，通常将矩阵看作二维数组。这里用符号  $\mathbf{A}$  表示一个矩阵，如果该矩阵有  $m$  行  $n$  列，那么有  $\mathbf{A} \in \mathbb{R}^{m \times n}$ 。矩阵中的每个元素都被一个行索引和一个列索引所确定，例如， $a_{ij}$  表示第  $i$  行、第  $j$  列的矩阵元素。如下，下式中的  $\mathbf{A}$  定义了一个 2 行 2 列的矩阵。

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \end{aligned} \quad (9.3)$$

<sup>2</sup>严格意义上，标量可以是复数等其他形式。这里为了方便讨论，仅以实数为对象。

## 2. 矩阵的转置

**转置** (Transpose) 是矩阵的重要操作之一。矩阵的转置可以看作是将矩阵以对角线为镜像进行翻转：假设  $A$  为  $m \times n$  列的矩阵，第  $i$  行、第  $j$  列的元素是  $a_{ij}$ ，即： $A = (a_{ij})_{m \times n}$ ，把  $m \times n$  矩阵  $A$  的行换成同序数的列得到一个  $n \times m$  矩阵，则得到  $A$  的转置矩阵，记为  $A^T$ ，且  $A^T = (a_{ji})_{n \times m}$ 。例如，对于下式中的矩阵，

$$A = \begin{pmatrix} 1 & 3 & 2 & 6 \\ 5 & 4 & 8 & 2 \end{pmatrix} \quad (9.4)$$

它转置的结果如下：

$$A^T = \begin{pmatrix} 1 & 5 \\ 3 & 4 \\ 2 & 8 \\ 6 & 2 \end{pmatrix} \quad (9.5)$$

向量可以看作只有一行（列）的矩阵。对应地，向量的转置可以看作是只有一列（行）的矩阵。标量可以看作是只有一个元素的矩阵。因此，标量的转置等于它本身，即  $a^T = a$ 。

## 3. 矩阵加法和数乘

矩阵加法又被称作**按元素加法** (Element-wise Addition)。它是指两个矩阵把其相对应元素加在一起的运算，通常的矩阵加法被定义在两个形状相同的矩阵上。两个  $m \times n$  矩阵  $A$  和  $B$  的和，标记为  $A + B$ ，它也是个  $m \times n$  矩阵，其内的各元素为其相对应元素相加后的值，即如果矩阵  $C = A + B$ ，则  $c_{ij} = a_{ij} + b_{ij}$ 。下式展示了矩阵之间进行加法的计算过程。

$$\begin{pmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{pmatrix} \quad (9.6)$$

矩阵加法满足以下运算规律：

- 交换律： $A + B = B + A$ 。
- 结合律： $(A + B) + C = A + (B + C)$ 。
- $A + \mathbf{0} = A$ ，其中  $\mathbf{0}$  指的是零矩阵，即元素皆为 0 的矩阵。
- $A + (-A) = \mathbf{0}$ ，其中  $-A$  是矩阵  $A$  的负矩阵，即将矩阵  $A$  的每个元素取负得到

的矩阵。

矩阵的**数乘** (Scalar Multiplication) 是指标量 (实数) 与矩阵的乘法运算, 计算过程是将标量与矩阵的每个元素相乘, 最终得到与原矩阵形状相同的矩阵。例如, 矩阵  $\mathbf{A} = (a_{ij})_{m \times n}$  与标量  $k$  进行数乘运算, 其结果矩阵  $\mathbf{B} = (ka_{ij})_{m \times n}$ , 即  $k(a_{ij})_{m \times n} = (ka_{ij})_{m \times n}$ 。公式(9.7)和(9.8)展示了矩阵数乘的计算过程:

$$\mathbf{A} = \begin{pmatrix} 3 & 2 & 7 \\ 5 & 8 & 1 \end{pmatrix} \quad (9.7)$$

$$2\mathbf{A} = \begin{pmatrix} 6 & 4 & 14 \\ 10 & 16 & 2 \end{pmatrix} \quad (9.8)$$

矩阵的数乘满足以下运算规律, 其中  $k$  和  $l$  是实数,  $\mathbf{A}$  和  $\mathbf{B}$  是形状相同的矩阵:

- 右分配律:  $k(\mathbf{A} + \mathbf{B}) = k\mathbf{A} + k\mathbf{B}$ 。
- 左分配律:  $(k + l)\mathbf{A} = k\mathbf{A} + l\mathbf{A}$ 。
- 结合律:  $(kl)\mathbf{A} = k(l\mathbf{A})$ 。

#### 4. 矩阵乘法和矩阵点乘

矩阵乘法是矩阵运算中最重要的操作之一, 为了与矩阵点乘区分, 通常也把矩阵乘法叫做矩阵叉乘。假设  $\mathbf{A}$  为  $m \times p$  的矩阵,  $\mathbf{B}$  为  $p \times n$  的矩阵, 对  $\mathbf{A}$  和  $\mathbf{B}$  作矩阵乘法的结果是一个  $m \times n$  的矩阵  $\mathbf{C}$ , 其中矩阵  $\mathbf{C}$  中第  $i$  行、第  $j$  列的元素可以表示为:

$$(\mathbf{AB})_{ij} = \sum_{k=1}^p a_{ik}b_{kj} \quad (9.9)$$

只有当第一个矩阵的列数与第二个矩阵的行数相等时, 两个矩阵才可以作矩阵乘法。公式(9.10)展示了矩阵乘法的运算过程, 若  $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$ ,  $\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$ , 则有:

$$\begin{aligned} \mathbf{C} &= \mathbf{AB} \\ &= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix} \end{aligned} \quad (9.10)$$

矩阵乘法满足以下运算规律:

- 结合律: 若  $\mathbf{A} \in \mathbb{R}^{m \times n}$ ,  $\mathbf{B} \in \mathbb{R}^{n \times p}$ ,  $\mathbf{C} \in \mathbb{R}^{p \times q}$ , 则  $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$ 。

- 左分配律：若  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{m \times n}$ ,  $C \in \mathbb{R}^{n \times p}$ , 则  $(A+B)C = AC+BC$ 。
- 右分配律：若  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{n \times p}$ ,  $C \in \mathbb{R}^{n \times p}$ , 则  $A(B+C) = AB+AC$ 。

可以将线性方程组用矩阵乘法表示，如对于线性方程组  $\begin{cases} 5x_1 + 2x_2 = y_1 \\ 3x_1 + x_2 = y_2 \end{cases}$ ，可以表示为  $Ax^T = y^T$ ，其中  $A = \begin{pmatrix} 5 & 2 \\ 3 & 1 \end{pmatrix}$ ,  $x^T = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ ,  $y^T = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ 。

矩阵的点乘就是两个形状相同的矩阵各个对应元素相乘，矩阵点乘也被称为**按元素乘积**（Element-wise Product）或 Hadamard 乘积，记为  $A \odot B$ 。例如，对于公式(9.11)和公式(9.12)所示的两个矩阵，

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 3 \end{pmatrix} \quad (9.11)$$

$$B = \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix} \quad (9.12)$$

矩阵点乘的计算方式如下：

$$\begin{aligned} C &= A \odot B \\ &= \begin{pmatrix} 1 \times 3 & 0 \times 1 \\ -1 \times 2 & 3 \times 1 \end{pmatrix} \end{aligned} \quad (9.13)$$

## 5. 线性映射

**线性映射**（Linear Mapping）或**线性变换**（Linear Transformation）是一个向量空间 V 到另一个向量空间 W 的映射函数  $f: v \rightarrow w$ ，且该映射函数保持加法运算和数量乘法运算，即对于空间 V 中任何两个向量  $u$  和  $v$  以及任何标量  $c$ ，始终符合公式(9.14)和公式(9.15)：

$$f(u+v) = f(u)+f(v) \quad (9.14)$$

$$f(cx) = cf(v) \quad (9.15)$$

利用矩阵  $A \in \mathbb{R}^{m \times n}$ ，可以实现两个有限维欧氏空间的映射函数  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ 。例如  $n$  维列向量  $x^T$  与  $m \times n$  的矩阵  $A$ ，向量  $x^T$  左乘矩阵  $A$ ，可将向量  $x^T$  映射为  $m$

列向量。公式(9.16)(9.18)(9.19)展示了一个具体的例子，

$$\mathbf{x}^T = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad (9.16)$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{m1} & \dots & \dots & a_{mn} \end{pmatrix} \quad (9.18)$$

可以得到：

$$\begin{aligned} \mathbf{y}^T &= \mathbf{A}\mathbf{x}^T \\ &= \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{pmatrix} \end{aligned} \quad (9.19)$$

上例中矩阵  $\mathbf{A}$  定义了一个从  $\mathbb{R}^n$  到  $\mathbb{R}^m$  的线性映射：向量  $\mathbf{x}^T \in \mathbb{R}^n$  和  $\mathbf{y}^T \in \mathbb{R}^m$  分别为两个空间中的列向量，即大小为  $n \times 1$  和  $m \times 1$  的矩阵。

## 6. 范数

工程领域，经常會使用被称为**范数**（Norm）的函数衡量向量大小，范数为向量空间内的所有向量赋予非零的正长度或大小。对于一个  $n$  维向量  $\mathbf{x}$ ，一个常见的范数函数为  $l_p$  范数，通常表示为  $\|\mathbf{x}\|_p$ ，其中  $p \geq 0$ ，是一个标量形式的参数。常用的  $p$  的取值有 1、2、 $\infty$  等。范数的计算方式如下：

$$\begin{aligned} l_p(\mathbf{x}) &= \|\mathbf{x}\|_p \\ &= \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}} \end{aligned} \quad (9.20)$$

$l_1$  范数为向量的各个元素的绝对值之和：

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i| \quad (9.21)$$

$l_2$  范数为向量的各个元素平方和的二分之一次方：

$$\begin{aligned}\|\mathbf{x}\|_2 &= \sqrt{\sum_{i=1}^n x_i^2} \\ &= \sqrt{\mathbf{x}^T \mathbf{x}}\end{aligned}\quad (9.22)$$

$l_2$  范数被称为**欧几里得范数**（Euclidean Norm）。从几何角度，向量也可以表示为从原点出发的一个带箭头的有向线段，其  $l_2$  范数为线段的长度，也常被称为向量的模。 $l_2$  范数在机器学习中非常常用。向量  $\mathbf{x}$  的  $l_2$  范数经常简化表示为  $\|\mathbf{x}\|$ ，可以通过点积  $\mathbf{x}^T \mathbf{x}$  进行计算。

$l_\infty$  范数为向量的各个元素的最大绝对值：

$$\|\mathbf{x}\|_\infty = \max\{x_1, x_2, \dots, x_n\} \quad (9.23)$$

广义上讲，范数是将向量映射到非负值的函数，其作用是衡量向量  $\mathbf{x}$  到坐标原点的距离。更严格的说，范数并不拘于  $l_p$  范数，任何一个同时满足下列性质的函数都可以作为范数：

- 若  $f(\mathbf{x}) = 0$ ，则  $\mathbf{x} = \mathbf{0}$ 。
- 三角不等式： $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ 。
- 任意实数  $\alpha$ ， $f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$ 。

在深度学习中，有时候希望衡量矩阵的大小，这时可以考虑使用**Frobenius 范数**（Frobenius Norm），其计算方式如下：

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{i,j}^2} \quad (9.24)$$

## 9.2.2 人工神经元和感知机

生物学中，神经元是神经系统的基本组成单元。同样，人工神经元是人工神经网络的基本单元。在人们的想象中，人工神经元应该与生物神经元类似。但事实上，二者在形态上是有明显差别的。如图9.3 是一个典型的人工神经元，其本质是一个形似  $y = f(\mathbf{x} \cdot \mathbf{w} + b)$  的函数。显而易见，一个神经元主要由  $\mathbf{x}$ ,  $\mathbf{w}$ ,  $b$ ,  $f$  四个部分构成。其中  $\mathbf{x}$  是一个形如  $(x_1, x_2, \dots, x_n)$  的实数向量，在一个神经元中担任“输入”的角色。 $\mathbf{w}$  通常被理解为神经元连接的**权重**（Weight）（对于一个人工神经元，权重是一个向量，表示为  $\mathbf{w}$ ；对于由多个神经元组成的神经网络，权重是一个矩阵，表示为  $\mathbf{W}$ ），其中的每一个元素都对应着一个输入和一个输出，代表着“某输入对某输出的贡献程度”。 $b$  被称作偏置（对于一个人工神经元，偏置是一个实数，表示为  $b$ ；对于神经网络中的某一层，偏置是一个向量，表示为  $\mathbf{b}$ ）。 $f$  被称作激活函数，用于对输入向

量各项加权和后进行某种变换。可见，一个人工神经元的功能是将输入向量与权重矩阵右乘（做内积）后，加上偏置量，经过一个激活函数得到一个标量结果。

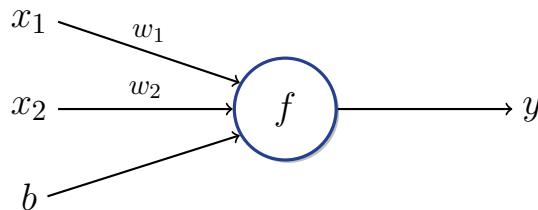


图 9.3 人工神经元

### 1. 感知机——最简单的人工神经元模型

感知机是人工神经元的一种实例，在上世纪 50-60 年代被提出后，对神经网络研究产生了深远的影响。感知机模型如图9.4所示，其输入是一个  $n$  维二值向量  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ，其中  $x_i = 0$  或  $1$ 。权重  $\mathbf{w} = (w_1, w_2, \dots, w_n)$ ，每个输入变量对应一个权重  $w_i$ 。偏置  $b$  是一个实数变量  $(-\sigma)$ 。输出也是一个二值结果，即  $y = 0$  或  $1$ 。 $y$  值的判定由输入的加权和是否大于（或小于）一个阈值  $\sigma$  决定：

$$y = \begin{cases} 0 & \sum_i x_i \cdot w_i - \sigma < 0 \\ 1 & \sum_i x_i \cdot w_i - \sigma \geq 0 \end{cases} \quad (9.25)$$

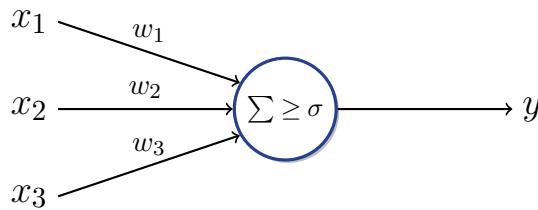


图 9.4 感知机模型

感知机可以做一些简单的决策。举一个非常简单的例子，有一场音乐会，你正在纠结是否去参加，有三个因素会影响你的决定：

- $x_1$ : 剧场是否离你足够近（是，则  $x_1 = 1$ ；否则  $x_1 = 0$ ）；
- $x_2$ : 票价是否低于 300 元（是，则  $x_2 = 1$ ；否则  $x_2 = 0$ ）；
- $x_3$ : 女朋友是否喜欢音乐会（是，则  $x_3 = 1$ ；否则  $x_3 = 0$ ）。

在这种情况下应该如何做出决定呢？比如，女朋友很希望和你一起去看音乐会，但是剧场很远而且票价 500 元，如果这些因素对你都是同等重要的（即  $w_1 = w_2 = w_3$ ，

假设这三个权重都设置为 1) 那么会得到一个综合得分:

$$\begin{aligned} x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 &= 0 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 \\ &= 1 \end{aligned} \quad (9.26)$$

如果你不是十分纠结的人，能够接受不完美的事情，你可能会把  $\sigma$  设置为 1，于是  $\sum w_i \cdot x_i - \sigma \geq 0$ ，那么你会去音乐会。可以看出，上面的例子的本质就是一个如图9.5所示的感知机:

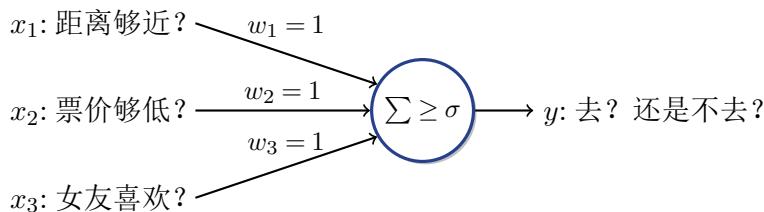


图 9.5 预测是否去剧场的感知机（权重相同）

## 2. 神经元内部权重

在上面的例子中，连接权重代表着每个输入因素对最终输出结果的重要程度，为了得到令人满意的决策，需要不断调整权重。如果你是守财奴，则会对票价看得更重一些，这样你会用不均匀的权重计算每个因素的影响，比如:  $w_1 = 0.5$ ,  $w_2 = 2$ ,  $w_3 = 0.5$ ，此时感知机模型如图9.6所示。在这种情况下，女友很希望和你一起去看音乐会，但是剧场很远而且票价 500 元，会导致你不去看音乐会，该决策过程如下:

$$\begin{aligned} \sum_i x_i \cdot w_i &= 0 \cdot 0.5 + 0 \cdot 2 + 1 \cdot 0.5 \\ &= 0.5 \\ &< \sigma = 1 \end{aligned} \quad (9.27)$$

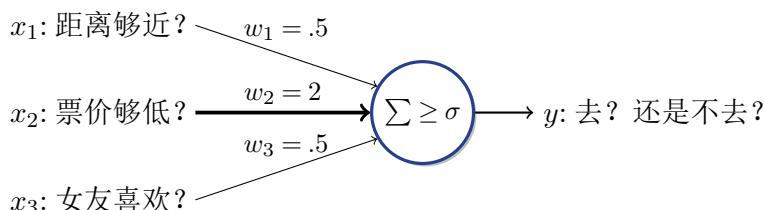


图 9.6 预测是否去剧场的感知机（权重不同）

当然，结果是女友对这个决定非常不满意。

### 3. 神经元的输入 —— 离散 vs 连续

在受到了女友“批评教育”之后，你意识到决策考虑的因素（即输入）不应该只是非 0 即 1，而应该把“程度”考虑进来，于是你改变了三个输入的形式：

$x_1$ : 10/距离 (km)

$x_2$ : 150/票价 (元)

$x_3$ : 女朋友是否喜欢

在新修改的模型中， $x_1$  和  $x_2$  变成了连续变量， $x_3$  仍然是离散变量，如图9.7所示。

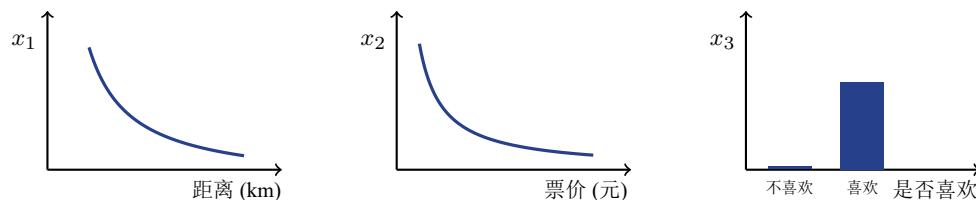


图 9.7 神经元输入的不同形式

使用修改后的模型做决策：女朋友很希望和你一起，但是剧场有 20km 远而且票价有 500 元。于是有  $x_1 = 10/20$ ,  $x_2 = 150/500$ ,  $x_3 = 1$ 。此时决策过程如下：

$$\begin{aligned} \sum_i x_i \cdot w_i &= 0.5 \cdot 0.5 + 0.3 \cdot 2 + 1 \cdot 0.5 \\ &= 1.35 \\ &> \sigma = 1 \end{aligned} \tag{9.28}$$

虽然剧场很远，价格有点贵，但是女友很满意，你还是很高兴。

### 4. 神经元内部的参数学习

一次成功的音乐会之后，你似乎掌握了一个真理：其他什么都不重要，女友的喜好最重要，所以你又将决策模型的权重做出了调整：最简单的方式就是  $w_1 = w_2 = 0$ ，同时令  $w_3 > 0$ ，相当于只考虑  $x_3$  的影响而忽略其他因素，于是你得到了如图9.8所示的决策模型：

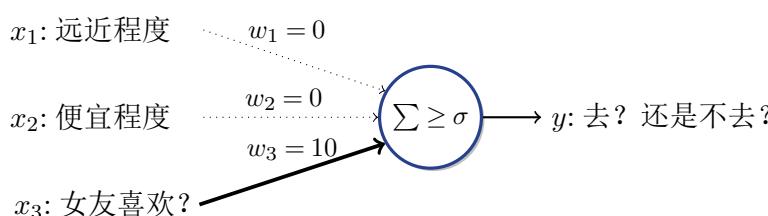


图 9.8 预测是否去剧场的决策模型（只考虑女友喜好）

很快又有了一场音乐会，距你 1000 公里，票价 3000 元，当然女友是一直喜欢音乐会的。根据新的决策模型，你义无反顾地选择去看音乐会。然而，女友又不高兴了，喜欢浪漫的女友觉得去看这场音乐会太奢侈了。在这几次看音乐会的经历中，你发现每个因素的权重需要准确地设置才能达到最好的决策效果。

那么如何确定最好的权重的？方法其实很简单，不断地尝试，根据结果不断地调整权重。在经过成百上千次的尝试之后，终于找到了一组合适的权重，使每次决策的正确率都很高。上面这个过程就类似于参数训练的过程，利用大量的数据来模拟成百上千次的尝试，根据输出的结果来不断地调整权重。

可以看到，在“是否参加音乐会”这个实际问题中，主要涉及到三方面的问题：

- **对问题建模**，即定义输入  $\{x_i\}$  的形式；
- **设计有效的决策模型**，即定义  $y$ ；
- **得到模型参数**（如权重  $\{w_i\}$ ）的最优值。

上面的例子对这三个问题都简要地做出了回答。下面的内容将继续对它们进行详细阐述。

### 9.2.3 多层神经网络

感知机是一种最简单的单层神经网络。一个很自然的问题是：能否把多个这样的网络叠加在一起，获得建模更复杂问题的能力？如果可以，那么在多层神经网络的每一层，神经元之间是怎么组织、工作的呢？单层网络又是通过什么方式构造成多层的呢？

#### 1. 线性变换和激活函数

为了建立多层神经网络，首先需要把前面提到的简单的神经元进行扩展，把多个神经元组成一“层”神经元。比如，很多实际问题需要同时有多个输出，这时可以把多个相同的神经元并列起来，每个神经元都会有一个单独的输出，这就构成一“层”，形成了单层神经网络。单层神经网络中的每一个神经元都对应着一组权重和一个输出，可以把单层神经网络中的不同输出看作一个事物不同角度的描述。

举个简单的例子，预报天气时，往往需要预测温度、湿度和风力，这就意味着如果使用单层神经网络进行预测，需要设置 3 个神经元。如图9.9所示，此时权重矩阵如下：

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \quad (9.29)$$

它的第一列元素  $\begin{pmatrix} w_{11} \\ w_{21} \end{pmatrix}$  是输入相对第一个输出  $y_1$  的权重，参数向量  $\mathbf{b} = (b_1, b_2, b_3)$  的第一个元素  $b_1$  是对应于第一个输出  $y_1$  的偏置量；类似的，可以得到  $y_2$  和  $y_3$ 。预

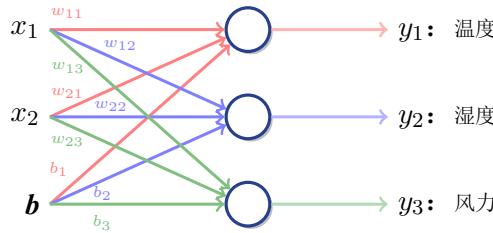


图 9.9 权重矩阵中的元素与输出的对应关系

测天气的单层模型如图9.10所示（在本例中，假设输入  $\mathbf{x} = (x_1, x_2)$ ）。

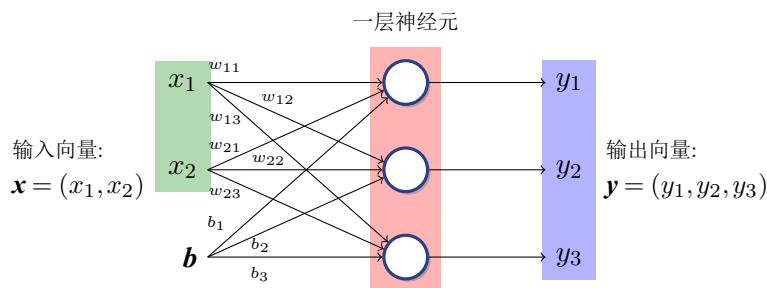


图 9.10 预测天气的单层神经网络

在神经网络中，对于输入向量  $\mathbf{x} \in \mathbb{R}^m$ ，一层神经网络首先将其经过线性变换映射到  $\mathbb{R}^n$ ，再经过激活函数变成  $\mathbf{y} \in \mathbb{R}^n$ 。还是上面天气预测的例子，每个神经元获得相同的输入，权重矩阵  $\mathbf{W}$  是一个  $2 \times 3$  矩阵，矩阵中每个元素  $w_{ij}$  代表第  $j$  个神经元中  $x_i$  对应的权重值，假设编号为 1 的神经元负责预测温度，则  $w_{i1}$  的含义为预测温度时输入  $x_i$  对其影响程度。此外所有神经元的偏置  $b_1, b_2, b_3$  组成了最终的偏置向量  $\mathbf{b}$ 。在该例中则有，权重矩阵  $\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$ ，偏置向量  $\mathbf{b} = (b_1, b_2, b_3)$ 。

那么，线性变换的本质是什么？图9.11正是线性变换的简单示意。

- 从代数角度看，对于线性空间  $V$ ，任意  $\mathbf{a}, \mathbf{a} \in V$  和数域中的任意  $\alpha$ ，线性变换  $T(\cdot)$  需满足： $T(\mathbf{a} + \mathbf{b}) = T(\mathbf{a}) + T(\mathbf{b})$ ，且  $T(\alpha\mathbf{a}) = \alpha T(\mathbf{a})$ ；
- 从几何角度看，公式中的  $\mathbf{x} \cdot \mathbf{W} + \mathbf{b}$  将  $\mathbf{x}$  右乘  $\mathbf{W}$  相当于对  $\mathbf{x}$  进行旋转变换。例如，对三个点  $(0,0), (0,1), (1,0)$  及其围成的矩形区域右乘如下矩阵：

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (9.30)$$

这样，矩形区域由第一象限旋转 90 度到了第四象限，如图9.11第一步所示。公式  $\mathbf{x} \cdot \mathbf{W} + \mathbf{b}$  中的公式中的  $\mathbf{b}$  相当于对其进行平移变换。其过程如图9.11 第二步

所示，偏置矩阵  $b = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$  将矩形区域沿  $x$  轴向右平移了一段距离。

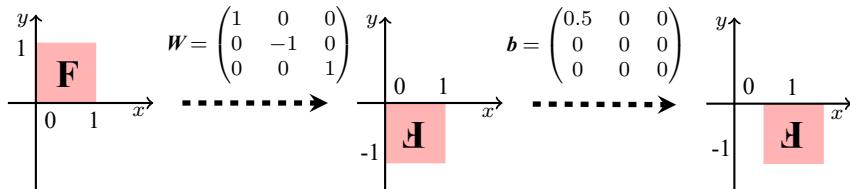


图 9.11 线性变换示意图

线性变换提供了对输入数据进行空间中旋转、平移的能力。线性变换也适用于更加复杂的情况，这也为神经网络提供了拟合不同函数的能力。比如可以利用线性变换将三维图形投影到二维平面上，或者将二维平面上的图形映射到三维空间。如图9.12所示，通过一个简单的线性变换，可以将三维图形投影到二维平面上。

$$\underbrace{\left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdots \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\}}_5 \times \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \underbrace{\left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdots \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}}_5$$

图 9.12 线性变换 3 维  $\rightarrow$  2 维数学示意

那激活函数又是什么？一个神经元在接收到经过线性变换的结果后，通过激活函数的处理，得到最终的输出  $y$ 。激活函数的目的是解决实际问题中的非线性变换，线性变换只能拟合直线，而激活函数的加入，使神经网络具有了拟合曲线的能力。特别是在实际问题中，很多现象都无法用简单的线性关系描述，这时可以使用非线性激活函数来描述更加复杂的问题。常见的非线性激活函数有 Sigmoid、ReLU、Tanh 等。图9.13和9.14中列举了几种激活函数的形式。

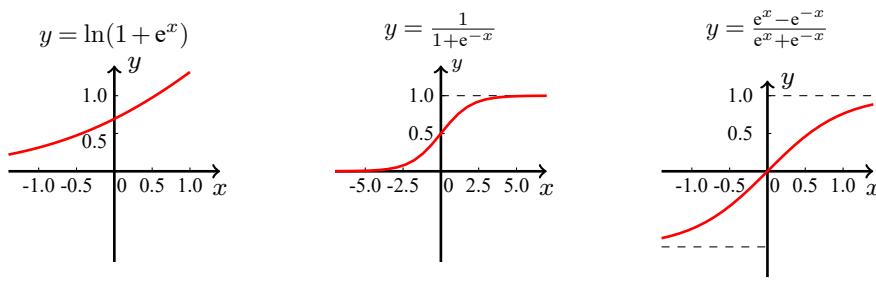


图 9.13 几种常见的激活函数

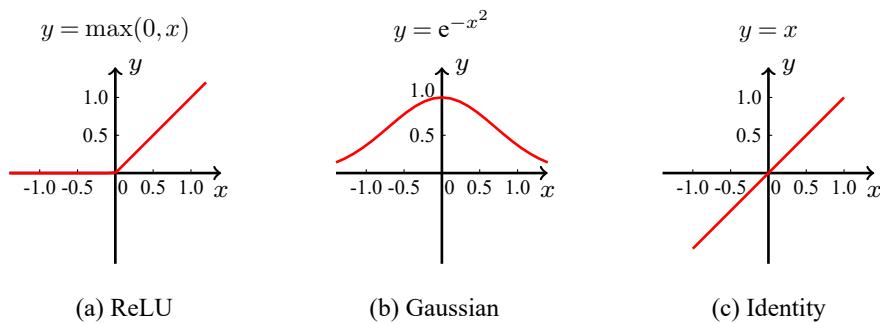


图 9.14 几种常见的激活函数（补）

## 2. 单层神经网络 → 多层神经网络

单层神经网络由线性变换和激活函数两部分构成，但在实际问题中，单层网络并不能很好地拟合复杂函数。因此很自然地想到将单层网络扩展到多层神经网络，即深层神经网络。将一层神经网络的最终输出向量作为另一层神经网络的输入向量，通过这种方式可以将多个单层神经网络连接在一起。

在多层神经网络中，通常包括输入层、输出层和至少一个隐藏层。图9.15展示了三个神经网络，包括输入层<sup>3</sup>、输出层和两个隐藏层。

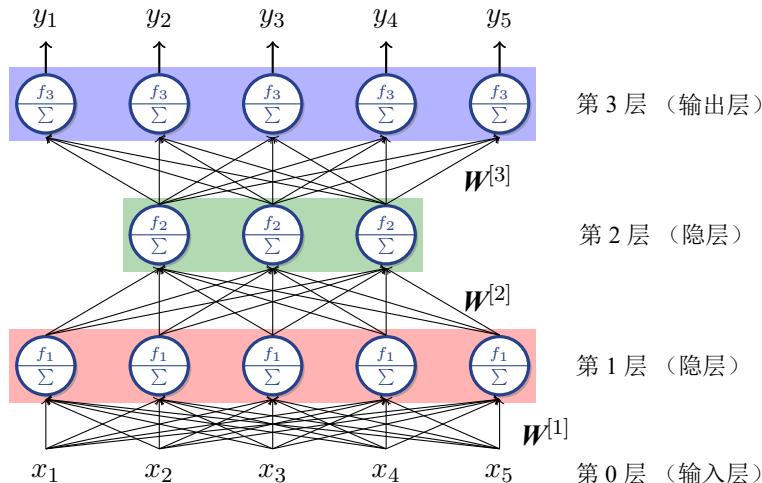


图 9.15 三层神经网络

### 9.2.4 函数拟合能力

神经网络方法之所以受到青睐一方面是由于它提供了端到端学习的模式，另一方面是由于它强大的函数拟合能力。理论上说，神经网络可以拟合任何形状的函数。下面就来看一下为什么神经网络会有这样的能力。

<sup>3</sup>由于输入层不存在神经元，因此在计算神经网络层数时不将其包括在内。

众所周知，单层神经网络无法解决线性不可分问题，比如经典的异或问题。但是具有一个隐藏层的两层神经网络在理论上就可以拟合所有的函数了。接下来我们分析一下为什么仅仅是多了一层，神经网络就能变得如此强大。对于二维空间（平面），“拟合”是指：把平面上一系列的点，用一条光滑的曲线连接起来，并用函数来表示这条拟合的曲线。这个概念可以推广到更高维空间上。在用神经网络解决问题时，可以通过拟合训练数据中的“数据点”来获得输入与输出之间的函数关系，并利用其对未知数据做出判断。可以假设输入与输出之间存在一种函数关系，而神经网络的“拟合”是要尽可能地逼近原函数输出值，与原函数输出值越逼近，则意味着拟合得越好。

如图9.16是一个以 Sigmoid 作为隐藏层激活函数的两层神经网络。通过调整参数  $\mathbf{W}^{[1]} = (w_{11}, w_{12})$ ,  $\mathbf{b} = (b_1, b_2)$  和  $\mathbf{W}^{[2]} = (w'_{11}, w'_{21})^T$  的值，可以不断地改变目标函数的形状。

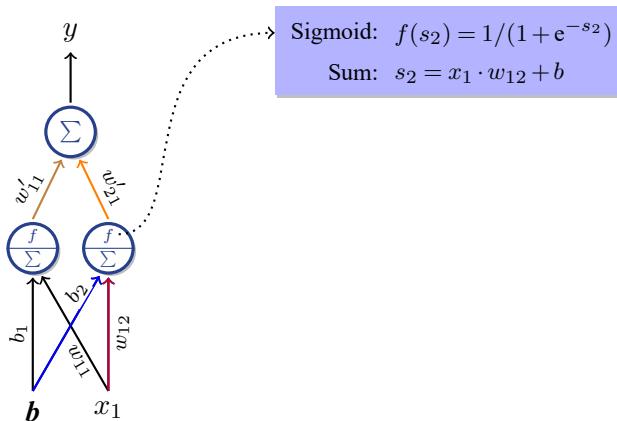


图 9.16 以 Sigmoid 作为隐藏层激活函数的两层神经网络

设置  $w'_{11} = 1$ ,  $w_{11} = 1$ ,  $b_1 = 0$ , 其他参数设置为 0。可以得到如图9.17(a)所示的目标函数，此时目标函数还是比较平缓的。通过调大  $w_{11}$ ，可以将图9.17(a)中函数的坡度调得更陡：当  $w_{11} = 10$  时，如图9.17(b)所示，目标函数的坡度与图9.17(a)相比变得更陡了；当  $w_{11} = 100$  时，如图9.17(c)所示，目标函数的坡度变得更陡、更尖锐，已经逼近一个阶梯函数。

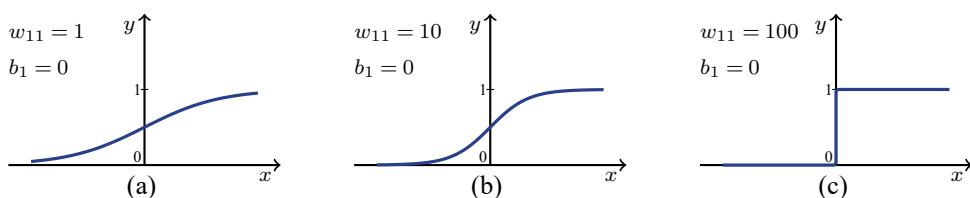


图 9.17 通过调整权重  $w_{11}$  改变目标函数平滑程度

设置  $w'_{11} = 1$ ,  $w_{11} = 100$ ,  $b_1 = 0$ , 其他参数设置为 0。可以得到如图9.18(a)所

示的目标函数，此时目标函数是一个阶梯函数，其“阶梯”恰好与  $y$  轴重合。通过改变  $b_1$ ，可以将整个函数沿  $x$  轴向左右平移：当  $b_1 = -2$  时，如图9.18(b)所示，与图9.18(a)相比目标函数的形状没有发生改变，但其位置沿  $x$  轴向右平移；当  $b_1 = -4$  时，如图9.18(c)所示，目标函数的位置继续沿  $x$  轴向右平移。

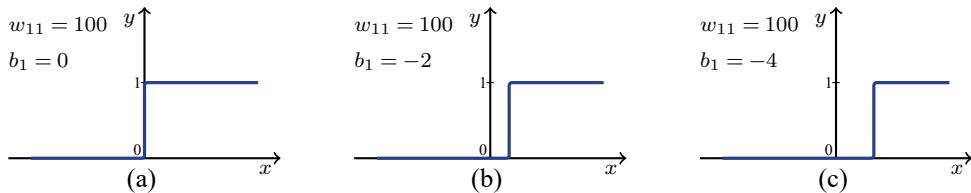


图 9.18 通过调整偏置量  $b_1$  改变目标函数位置

设置  $w'_{11} = 1$ ,  $w_{11} = 100$ ,  $b_1 = -4$ , 其他参数设置为 0。可以得到如图9.19(a)所示的目标函数，此时目标函数是一个阶梯函数，该阶梯函数取得最大值的分段处为  $y = 1$ 。通过改变  $w'_{11}$ ，可以将目标函数“拉高”或是“压扁”。如图9.19(b) 和 (c) 所示，目标函数变得“扁”了。最终，该阶梯函数取得最大值的分段处约为  $y = 0.7$ 。

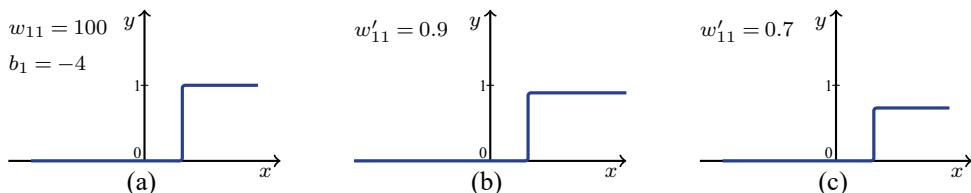


图 9.19 通过改变权重  $w'_{11}$  将目标函数“拉高”或“压扁”

设置  $w'_{11} = 0.7$ ,  $w_{11} = 100$ ,  $b_1 = -4$ , 其他参数设置为 0。可以得到如图9.20(a)所示的目标函数，此时目标函数是一个阶梯函数。若是将其他参数设置为  $w'_{21} = 0.7$ ,  $w'_{11} = 100$ ,  $b_2 = 16$ ，由图9.20(b)可以看出，原来目标函数的“阶梯”由一级变成了两级，由此可以推测，将第二组参数进行设置，可以使目标函数分段数增多；若将第二组参数中的  $w'_{21}$  由原来的 0.7 设置为 -0.7，可得到如图9.20(c)所示的目标函数，与图9.20(b)相比，原目标函数的“第二级阶梯”向下翻转，由此可见  $\mathbf{W}^{[2]}$  的符号决定了目标函数的翻转方向。

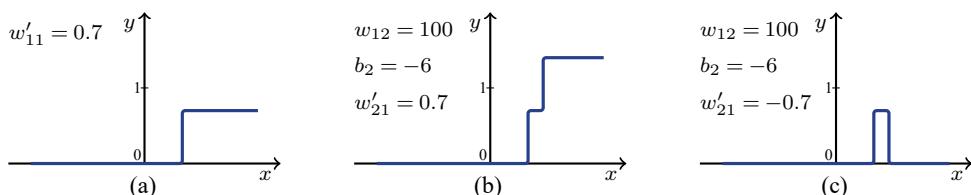


图 9.20 通过设置第二组参数 ( $b_2$  和  $w'_{21}$ ) 将目标函数分段数增加

由上面的内容，已经看到通过设置神经元中的参数将目标函数的形状做各种变换，但是看起来目标函数的类型还是比较单一的。而在实际问题中，输入与输出之

间的函数关系甚至复杂到无法人为构造或是书写，神经网络又是如何拟合这种复杂的函数关系的呢？



图 9.21 将目标函数作分段处理

以如图9.21(a)所示的目标函数为例，为了拟合该函数，可以将其看成分成无数小段的分段函数，如图9.21(b)所示。

如图9.22(a)所示，上例中两层神经网络的函数便可以拟合出目标函数的一小段。为了使两层神经网络可以拟合出目标函数更多的一小段，需要增加隐层神经元的个数。如图9.22(b)，将原本的两层神经网络神经元个数增多一倍，由2个神经元扩展到4个神经元，其函数的分段数也增加一倍，而此时的函数恰好可以拟合目标函数中的两个小段。以此类推，理论上，该两层神经网络便可以通过不断地增加隐层神经元数量去拟合任意函数。

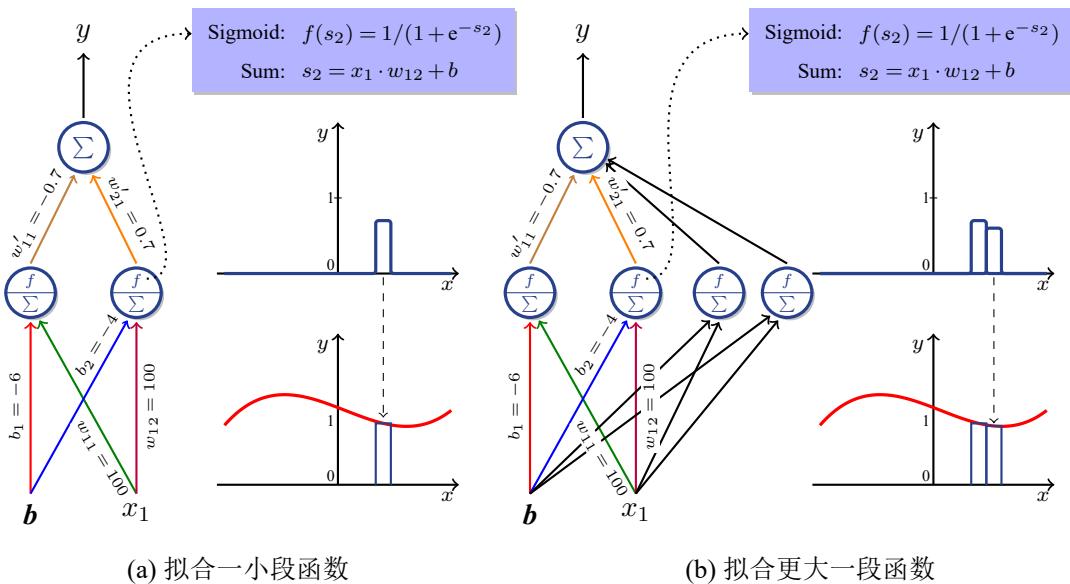


图 9.22 扩展隐层神经元个数去拟合目标函数更多的“一小段”

两层神经元的神经网络在理论上可以拟合所有函数了，但是在实际问题中所使用的神经网络都远远超过了两层，这也是对深度学习这个概念中“深度”的一种体现。使用深层神经网络主要有以下几方面的原因：

- 使用较浅的神经网络去拟合一个比较复杂的函数关系，需要数量极其庞大的神

经元和参数，训练难度大。在上面的例子中可以看出，两层神经元仅仅拟合目标函数的两小段，其隐层就需要 4 个神经元。从另一个角度说，加深网络也可能达到与宽网络（更多神经元）类似的效果。

- 更多层的网络可以提供更多的线性变换和激活函数，对输入的抽象程度更好，因而可以更好的表示数据的特征。

在本书后面的内容中还会看到，深层网络在机器翻译中可以带来明显的性能提升。

## 9.3 神经网络的张量实现

在神经网络内部，输入经过若干次变换，最终得到输出的结果。这个过程类似于一种逐层的数据“流动”。不禁会产生这样的疑问：在神经网络中，数据是以哪种形式“流动”的？如何去编程实现这种数据“流动”呢？

为了解决上面的问题，本节将介绍人工神经网络更加通用的描述形式——张量计算。随后也会看到，使用基于张量的数学工具，可以方便的搭建神经网络。

### 9.3.1 张量及其计算

#### 1. 张量

对于神经网络中的某层神经元  $y = f(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})$ ，其中  $\mathbf{W}$  是权重矩阵，例如  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ ， $\mathbf{b}$  是偏置向量，例如  $(1, 3)$ 。在这里，输入  $\mathbf{x}$  和输出  $\mathbf{y}$ ，可以不是简单的向量或是矩阵形式，而是深度学习中更加通用的数学量——**张量**（Tensor），比如公式(9.31)中的几种情况都可以看作是深度学习中定义数据的张量：

$$\mathbf{x} = \begin{pmatrix} -1 & 3 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} -1 & 3 \\ 0.2 & 2 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} \begin{pmatrix} -1 & 3 \\ 0.2 & 2 \end{pmatrix} \\ \begin{pmatrix} -1 & 3 \\ 0.2 & 2 \end{pmatrix} \end{pmatrix} \quad (9.31)$$

简单来说，张量是一种通用的工具，用于描述由多个数据构成的量。比如，输入的量有三个维度在变化，用矩阵不容易描述，但是用张量却很容易。

从计算机实现的角度来看，现在所有深度学习框架都把张量定义为“多维数组”。张量有一个非常重要的属性——**阶**（Rank）。可以将多维数组中“维”的属性与张量的“阶”的属性作类比，这两个属性都表示多维数组（张量）有多少个独立的方向。例如，3 是一个标量，相当于一个 0 维数组或 0 阶张量； $(2 \quad -3 \quad 0.8 \quad 0.2)^T$  是一个向量，相当于一个 1 维数组或 1 阶张量； $\begin{pmatrix} -1 & 3 & 7 \\ 0.2 & 2 & 9 \end{pmatrix}$  是一个矩阵，相当于一

一个 2 维数组或 2 阶张量；如图9.23所示，这是一个 3 维数组或 3 阶张量，其中，每个  $4 \times 4$  的方形代表一个 2 阶张量，这样的方形有 4 个，最终形成 3 阶张量。

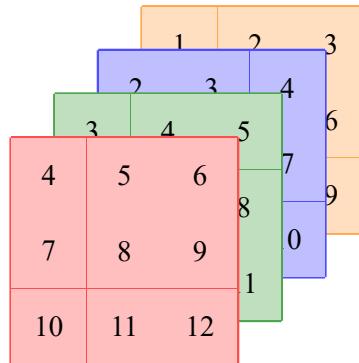


图 9.23 3 阶张量示例 ( $4 \times 4 \times 4$ )

虽然这里所使用的张量是出于编程实现的视角，但是数学中张量有严格的规定。从数学上看“张量并不是向量和矩阵的简单扩展，多维数组也并不是张量所必须的表达形式”。从某种意义上说，矩阵才是张量的扩展。当然，这个逻辑可能和人们在深度学习中的认知是不一致的。但是，本书仍然遵循深度学习中常用的概念，把张量理解为多维数组。在保证数学表达的简洁性的同时，使程序实现接口更加统一。

## 2. 张量的矩阵乘法

对于一个单层神经网络， $\mathbf{y} = f(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})$  中的  $\mathbf{x} \cdot \mathbf{W}$  表示对输入  $\mathbf{x}$  进行线性变换，其中  $\mathbf{x}$  是输入张量， $\mathbf{W}$  是权重矩阵。 $\mathbf{x} \cdot \mathbf{W}$  表示的是矩阵乘法，需要注意的是这里是矩阵乘法而不是张量乘法。

张量乘以矩阵是怎样计算呢？可以先回忆一下9.2.1节的线性代数的知识。假设  $\mathbf{A}$  为  $m \times p$  的矩阵， $\mathbf{B}$  为  $p \times n$  的矩阵，对  $\mathbf{A}$  和  $\mathbf{B}$  作矩阵乘积的结果是一个  $m \times n$  的矩阵  $\mathbf{C}$ ，其中矩阵  $\mathbf{C}$  中第  $i$  行、第  $j$  列的元素可以表示为：

$$(\mathbf{AB})_{ij} = \sum_{k=1}^p a_{ik} b_{kj} \quad (9.32)$$

例如  $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$ ,  $\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$ , 两矩阵做乘法运算的过程如下：

$$\begin{aligned} \mathbf{C} &= \mathbf{AB} \\ &= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix} \end{aligned} \quad (9.33)$$

将矩阵乘法扩展到高阶张量中：一个张量  $\mathbf{x}$  若要与矩阵  $\mathbf{W}$  做矩阵乘法，则  $\mathbf{x}$  的

最后一维度需要与  $\mathbf{W}$  的行数大小相等，即：若张量  $\mathbf{x}$  的形状为  $\cdot \times n$ ， $\mathbf{W}$  须为  $n \times \cdot$  的矩阵。下式是一个例子：

$$\mathbf{x}(1:4, 1:4, 1:4) \times \mathbf{W}(1:4, 1:2) = \mathbf{s}(1:4, 1:4, 1:2) \quad (9.34)$$

其中，张量  $\mathbf{x}$  沿第 1 阶所在的方向与矩阵  $\mathbf{W}$  进行矩阵运算（张量  $\mathbf{x}$  第 1 阶的每个维度都可以看做一个  $4 \times 4$  的矩阵）。图9.24演示了这个计算过程。张量  $\mathbf{x}$  中编号为①的子张量（可看作矩阵）与矩阵  $\mathbf{W}$  进行矩阵乘法，其结果对应张量  $\mathbf{s}$  中编号为①的子张量。这个过程会循环四次，因为有四个这样的矩阵（子张量）。最终，图9.24给出了结果张量的形式  $(4 \times 4 \times 2)$ 。

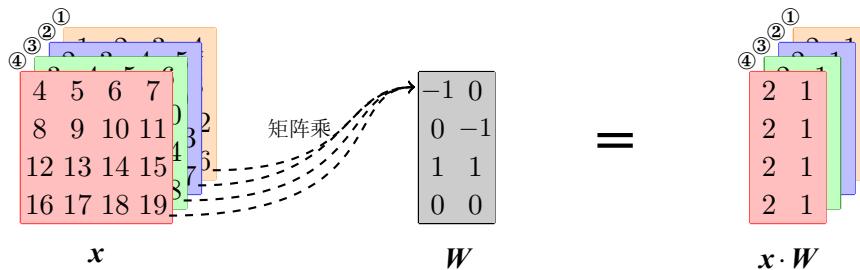


图 9.24 张量与矩阵的矩阵乘法

### 3. 张量的单元操作

对于神经网络中的某层神经元  $\mathbf{y} = f(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})$ ，也包含有其他张量单元操作：1) 加法： $\mathbf{s} + \mathbf{b}$ ，其中张量  $\mathbf{s} = \mathbf{x} \cdot \mathbf{W}$ ；2) 激活函数： $f(\cdot)$ 。具体来说：

- $\mathbf{s} + \mathbf{b}$  中的单元加就是对张量中的每个位置都进行加法。在上例中  $\mathbf{s}$  是形状为  $(1:4, 1:4, 1:2)$  的 3 阶张量，而  $\mathbf{b}$  是含有 4 个元素的向量，在形状不同的情况下是怎样进行单元加的呢？在这里需要引入**广播机制**（Broadcast Mechanism）：如果两个数组的后缘维度（即从末尾开始算起的维度）的轴长度相符或其中一方的长度为 1，则认为它们是广播兼容的。广播会在缺失或长度为 1 的维度上进行，它是深度学习框架中常用的计算方式。来看一个具体的例子，如图9.25所示， $\mathbf{s}$  是一个  $2 \times 4$  的矩阵而  $\mathbf{b}$  是一个长度为 4 的向量，这两者进行单元加运算时，广播机制会将  $\mathbf{b}$  沿第一个维度复制后，再与  $\mathbf{s}$  做加法运算。
- 除了单位加之外，张量之间也可以使用减法操作、乘法操作。此外也可以对张量作激活操作，这里将其称作为函数的**向量化**（Vectorization）。例如，对向量（1 阶张量）作 ReLU 激活，ReLU 激活函数表达式如下：

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \quad (9.35)$$

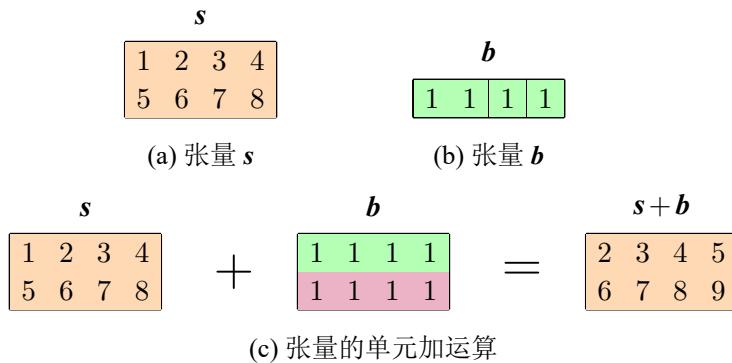


图 9.25 广播机制

例如  $\text{ReLU} \begin{pmatrix} 2 \\ -.3 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ 。

### 9.3.2 张量的物理存储形式

在深度学习世界中，张量就是多维数组。因此，张量的物理存储方式也与多维数组相同。如下就是一些实例：

- 张量  $t(1:3)$  表示一个含有三个元素的向量（1阶张量），其物理存储如图9.26(a)所示。
  - 张量  $t(1:2, 1:3)$  表示一个  $2 \times 3$  的矩阵（2阶张量），其物理存储如图9.26(b)所示。
  - 张量  $t(1:2, 1:2, 1:3)$  表示一个大小  $2 \times 2 \times 3$  的3阶张量，其物理存储如图9.26(c)所示。

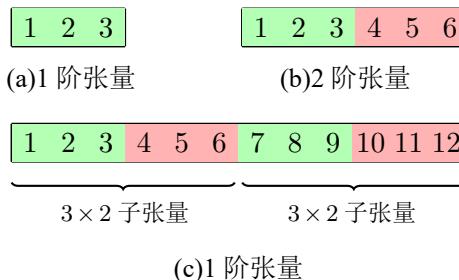


图 9.26 不同阶的张量的物理存储方式

实际上，高阶张量的物理存储方式也与多维数组在 C++、Python 中的物理存储方式相同。

### 9.3.3 张量的实现手段

实现神经网络的开源系统有很多，比如，使用经典的 Python 工具包 Numpy。也可以使用成熟的深度学习框架，比如，Tensorflow 和 Pytorch 就是非常受欢迎的深度学习工具包，除此之外还有很多其他优秀的框架：CNTK、MXNet、PaddlePaddle、Keras、Chainer、dl4j、NiuTensor 等。开发者可以根据自身的喜好和开发项目的要求选择所采用的框架。

这里以 NiuTensor 为例对张量计算库进行简单介绍。这类库需要提供张量计算接口，如张量的声明、定义和张量的各种代数运算，各种单元算子，如 +、-、\*、/、Log（取对数）、Exp（指数运算）、Power（幂方运算）、Absolute（绝对值）等，还有 Sigmoid、Softmax 等激活函数。除了上述单元算子外，张量计算库还支持张量之间的高阶运算，其中最常用的是矩阵乘法。表9.2 展示了一些常用的函数。

表 9.2 NiuTensor 支持的部分函数

函数	描述
a.Reshape(o,s)	把张量 $a$ 变换成阶为 o、形状为 s 的张量
a.Get(pos)	取张量 $a$ 中位置为 pos 的元素
a.Set(v,pos)	把张量 $a$ 中位置为 pos 的元素值设为 v
a.Dump(file)	把张量 $a$ 存到 file 中，file 为文件句柄
a.Read(file)	从 file 中读取张量 $a$ ，file 为文件句柄
Power(a,p)	计算指数 $a^p$
Linear(a,s,b)	计算 $a*s+b$ ，s 和 b 都是一个实数
CopyValue(a)	构建张量 $a$ 的一个拷贝
ReduceMax(a,d)	对张量 $a$ 沿着方向 d 进行规约，得到最大值
ReduceSum(a,d)	对张量 $a$ 沿着方向 d 进行规约，得到和
Concatenate(a,b,d)	把两个张量 $a$ 和 $b$ 沿 d 方向级联
Merge(a,d)	对张量 $a$ 沿 d 方向合并
Split(a,d,n)	对张量 $a$ 沿 d 方向分裂成 n 份
Sigmoid(a)	对张量 $a$ 进行 Sigmoid 变换
Softmax(a)	对张量 $a$ 进行 Softmax 变换，沿最后一个方向
HardTanh(a)	对张量 $a$ 进行 HardTanh 变换（双曲正切的近似）
Rectify(a)	对张量 $a$ 进行 ReLU 变换

### 9.3.4 前向传播与计算图

有了张量这个工具，可以很容易地实现任意的神经网络。反过来，神经网络都可以被看作是张量的函数。一种经典的神经网络计算模型是：给定输入张量，通过各个神经网络层所对应的张量计算之后，最后得到输出张量。这个过程也被称作**前向传播**（Forward Propagation），它常常被应用在使用神经网络对新的样本进行推断中。

来看一个具体的例子，图9.27展示了一个根据天气情况判断穿衣指数（穿衣指数是人们穿衣薄厚的依据）的过程，将当天的天空状况、低空气温、水平气压作为输入，通过一层神经元在输入数据中提取温度、风速两方面的特征，并根据这两方面的特征判断穿衣指数。需要注意的是，在实际的神经网络中，并不能准确地知道神经元究竟可以提取到哪方面的特征，以上表述是为了让读者更好地理解神经网络的建模过程和前向传播过程。这里将上述过程建模为如图9.27所示的两层神经网络。

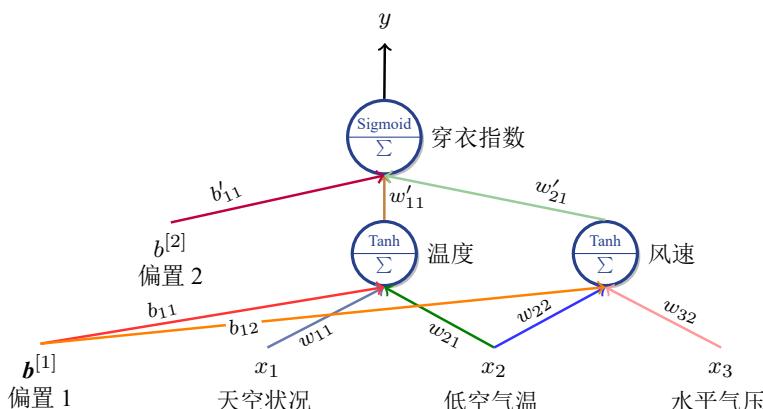


图 9.27 判断穿衣指数问题的神经网络过程

它可以被描述为公式(9.36)，其中隐藏层的激活函数是 Tanh 函数，输出层的激活函数是 Sigmoid 函数， $\mathbf{W}^{[1]}$  和  $\mathbf{b}^{[1]}$  分别表示第一层的权重矩阵和偏置， $\mathbf{W}^{[2]}$  和  $\mathbf{b}^{[2]}$  分别表示第二层的权重矩阵和偏置<sup>4</sup>：

$$y = \text{Sigmoid}(\text{Tanh}(\mathbf{x} \cdot \mathbf{W}^{[1]} + \mathbf{b}^{[1]}) \cdot \mathbf{W}^{[2]} + \mathbf{b}^{[2]}) \quad (9.36)$$

前向计算实现如图9.28所示，图中对各张量和其他参数的形状做了详细说明。输入  $\mathbf{x} = (x_1, x_2, x_3)$  是一个  $1 \times 3$  的张量，其三个维度分别对应天空状况、低空气温、水平气压三个方面的数据。输入数据经过隐藏层的线性变换  $\mathbf{x} \cdot \mathbf{W}^{[1]} + \mathbf{b}^{[1]}$  和 Tanh 函数的激活，得到新的张量  $\mathbf{a} = (a_1, a_2)$ ，其中  $a_1, a_2$  分别对应着从输入数据中提取出的温度和风速两方面特征；神经网络在获取到天气情况的特征  $\mathbf{a}$  后，继续对其进行线性变换  $\mathbf{a} \cdot \mathbf{W}^{[2]} + \mathbf{b}^{[2]}$  和 Sigmoid 函数的激活操作，得到神经网络的最终输出  $y$ ，即神经网络此时预测的穿衣指数。

<sup>4</sup>注意这里  $\mathbf{b}^{[1]}$  是向量而  $b^{[2]}$  是标量，因而前者加粗后者未加粗

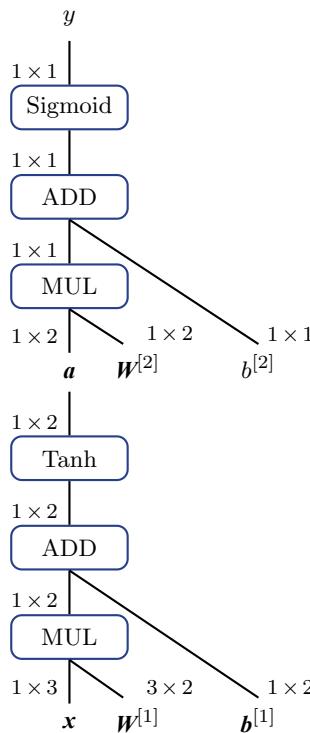


图 9.28 前向计算示例（计算图）

图9.28实际上是神经网络的一种**计算图**（Computation Graph）表示。现在很多深度学习框架都是把神经网络转化为计算图，这样可以把复杂的运算分解为简单的运算，称为**算子**（Calculus）。通过对计算图中节点的遍历，可以方便地完成神经网络的计算。比如，可以对图中节点进行拓扑排序（由输入到输出），之后依次访问每个节点，同时完成相应的计算，这也就实现了一个前向计算的过程。

使用计算图的另一个优点在于，这种方式易于参数梯度的计算。在后面的内容中会看到，计算神经网络中参数的梯度是模型训练的重要步骤。在计算图中，可以使用**反向传播**（Backward Propagation）的方式逐层计算不同节点上的梯度信息。在9.4.2节会看到使用计算图这种结构可以非常方便、高效地计算反向传播中所需的梯度信息。

## 9.4 神经网络的参数训练

简单来说，神经网络可以被看作是由变量和函数组成的表达式，例如： $y = \mathbf{x} + \mathbf{b}$ 、 $y = \text{ReLU}(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})$ 、 $y = \text{Sigmoid}(\text{ReLU}(\mathbf{x} \cdot \mathbf{W}^{[1]} + \mathbf{b}^{[1]}) \cdot \mathbf{W}^{[2]} + \mathbf{b}^{[2]})$  等等，其中的  $\mathbf{x}$  和  $\mathbf{y}$  作为输入和输出向量， $\mathbf{W}$ 、 $\mathbf{b}$  等其他变量作为**模型参数**（Model Parameters）。确定了函数表达式和模型参数，也就确定了神经网络模型。通常，表达式的形式需要系统开发者设计，而模型参数的数量有时会非常巨大，因此需要自动学习，这个过程

也被称为模型学习或训练。为了实现这个目标，通常会准备一定量的带有标准答案的数据，称之为有标注数据。这些数据会用于对模型参数的学习，这也对应了统计模型中的参数估计过程。在机器学习中，一般把这种使用有标注数据进行统计模型参数训练的过程称为**有指导的训练或有监督的训练**（Supervised Training）。在本章中，如果没有特殊说明，模型训练都是指有监督的训练。那么神经网络内部是怎样利用有标注数据对参数进行训练的呢？

为了回答这个问题，可以把模型参数的学习过程看作是一个优化问题，即找到一组参数，使得模型达到某种最优的状态。这个问题又可以被转化为两个新的问题：

- 优化的目标是什么？
- 如何调整参数以达到优化目标？

下面会围绕这两个问题对神经网络的参数学习方法展开介绍。

#### 9.4.1 损失函数

在神经网络的有监督学习中，训练模型的数据是由输入和正确答案所组成的样本构成的。假设有多个输入样本  $\{x^{[1]}, \dots, x^{[n]}\}$ ，每一个  $x^{[i]}$  都对应一个正确答案  $y^{[i]}$ ， $\{x^{[i]}, y^{[i]}\}$  就构成一个优化神经网络的**训练数据集合**（Training Data Set）。对于一个神经网络模型  $y = f(x)$ ，每个  $x^{[i]}$  也会有一个输出  $\hat{y}^{[i]}$ 。如果可以度量正确答案  $y^{[i]}$  和神经网络输出  $\hat{y}^{[i]}$  之间的偏差，进而通过调整网络参数减小这种偏差，就可以得到更好的模型。

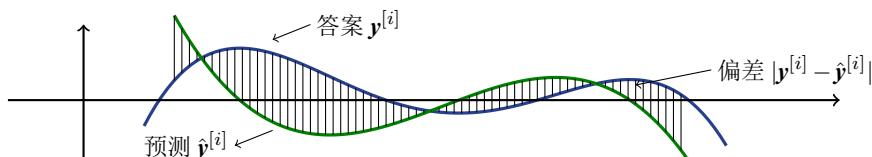


图 9.29 正确答案与神经网络输出之间的偏差

通常，可以通过设计**损失函数**（Loss Function）来度量正确答案  $y^{[i]}$  和神经网络输出  $\hat{y}^{[i]}$  之间的偏差。而这个损失函数往往充当训练的**目标函数**（Objective Function），神经网络训练就是通过不断调整神经网络内部的参数而使损失函数最小化。图9.29展示了一个绝对值损失函数的实例。

这里用  $Loss(y^{[i]}, \hat{y}^{[i]})$  表示网络输出  $\hat{y}^{[i]}$  相对于答案  $y^{[i]}$  的损失，简记为  $L$ 。表9.3是几种常见损失函数的定义。需要注意的是，没有一种损失函数可以适用于所有的问题。损失函数的选择取决于许多因素，包括：数据中是否有离群点、模型结构的选择、是否易于找到函数的导数以及预测结果的置信度等。对于相同的神经网络，不同的损失函数会对训练得到的模型产生不同的影响。对于新的问题，如果无法找到已有的、适合于该问题的损失函数，研究人员也可以自定义损失函数。因此设计新的损失函数也是神经网络中有趣的研究方向。

表 9.3 常见的损失函数

名称	定义	应用
0-1 损失	$L = \begin{cases} 0 & \mathbf{y}^{[i]} = \hat{\mathbf{y}}^{[i]} \\ 1 & \mathbf{y}^{[i]} \neq \hat{\mathbf{y}}^{[i]} \end{cases}$	感知机
Hinge 损失	$L = \max(0, 1 - \mathbf{y}^{[i]} \cdot \hat{\mathbf{y}}^{[i]})$	SVM
绝对值损失	$L =  \mathbf{y}^{[i]} - \hat{\mathbf{y}}^{[i]} $	回归
Logistic 损失	$L = \log(1 + \mathbf{y}^{[i]} \cdot \hat{\mathbf{y}}^{[i]})$	回归
平方损失	$L = (\mathbf{y}^{[i]} - \hat{\mathbf{y}}^{[i]})^2$	回归
指数损失	$L = \exp(-\mathbf{y}^{[i]} \cdot \hat{\mathbf{y}}^{[i]})$	AdaBoost
交叉熵损失	$L = -\sum_k \hat{\mathbf{y}}_k^{[i]} \log \mathbf{y}_k^{[i]}$ 其中, $\mathbf{y}_k^{[i]}$ 表示 $\mathbf{y}^{[i]}$ 的第 $k$ 维	多分类

在实际系统开发中, 损失函数中除了损失项 (即用来度量正确答案  $\mathbf{y}^{[i]}$  和神经网络输出  $\hat{\mathbf{y}}^{[i]}$  之间的偏差的部分) 之外, 还可以包括正则项, 比如 L1 正则和 L2 正则。设置正则项本质上是要加入一些偏置, 使模型在优化的过程中偏向某个方向多一些。关于正则项的内容将在 9.4.5 节介绍。

#### 9.4.2 基于梯度的参数优化

对于第  $i$  个样本  $(\mathbf{x}^{[i]}, \mathbf{y}^{[i]})$ , 把损失函数  $L(\mathbf{y}^{[i]}, \hat{\mathbf{y}}^{[i]})$  看作是参数  $\boldsymbol{\theta}$  的函数<sup>5</sup>, 因为模型输出  $\hat{\mathbf{y}}^{[i]}$  是由输入  $\mathbf{x}^{[i]}$  和模型参数  $\boldsymbol{\theta}$  决定, 因此也把损失函数写为  $L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta})$ 。下式描述了参数学习的过程:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta}) \quad (9.37)$$

其中,  $\hat{\boldsymbol{\theta}}$  表示在训练数据上使损失的平均值达到最小的参数,  $n$  为训练数据总量。 $\frac{1}{n} \sum_{i=1}^n L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta})$  也被称作**代价函数** (Cost Function), 它是损失函数均值期望的估计, 记为  $J(\boldsymbol{\theta})$ 。

参数优化的核心问题是: 找到使代价函数  $J(\boldsymbol{\theta})$  达到最小的  $\boldsymbol{\theta}$ 。然而  $J(\boldsymbol{\theta})$  可能会包含大量的参数, 比如, 基于神经网络的机器翻译模型的参数量可能会超过一亿个。这时不可能用手动方法进行调参。为了实现高效的参数优化, 比较常用的手段是使用**梯度下降方法** (The Gradient Descent Method)。

<sup>5</sup>为了简化描述, 可以用  $\boldsymbol{\theta}$  表示神经网络中的所有参数, 包括各层的权重矩阵  $\mathbf{W}^{[1]} \dots \mathbf{W}^{[n]}$  和偏置向量  $\mathbf{b}^{[1]} \dots \mathbf{b}^{[n]}$  等。

## 1. 梯度下降

梯度下降法是一种常用的优化方法，非常适用于目标函数可微分的问题。它的基本思想是：给定函数上的第一个点，找到使函数值变化最大的方向，然后前进一“步”，这样模型就可以朝着更大（或更小）的函数值以最快的速度移动<sup>6</sup>。具体来说，梯度下降通过迭代更新参数  $\theta$ ，不断沿着梯度的反方向让参数  $\theta$  朝着损失函数更小的方向移动：如果  $J(\theta)$  对  $\theta$  可微分，则  $\frac{\partial J(\theta)}{\partial \theta}$  将指向  $J(\theta)$  在  $\theta$  处变化最大的方向，这里将其称之为梯度方向。 $\theta$  沿着梯度方向更新，新的  $\theta$  可以使函数更接近极值，其过程如图9.30所示<sup>7</sup>。

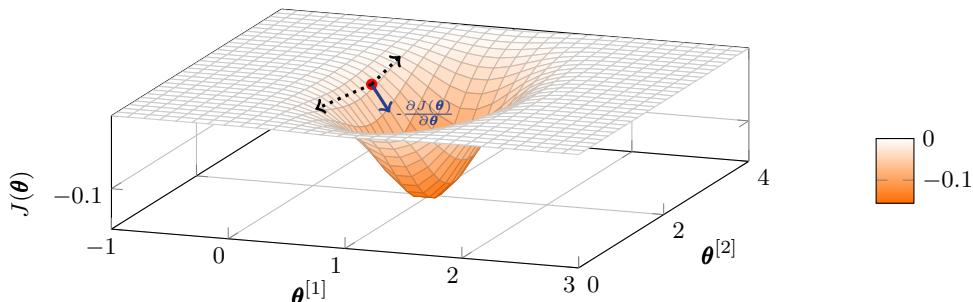


图 9.30 函数上一个点沿着不同方向移动的示例

应用梯度下降算法时，首先需要初始化参数  $\theta$ 。一般情况下深度学习中的参数应该初始化为一个不太大的随机数。一旦初始化  $\theta$  后，就开始对模型进行不断的更新，**参数更新的规则**（Update Rule）如下：

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta} \quad (9.38)$$

其中  $t$  表示更新的步数， $\alpha$  是一个超参数，被称作**学习率**（Learning Rate），表示更新步幅的大小。 $\alpha$  的设置需要根据任务进行调整。

从优化的角度看，梯度下降是一种典型的**基于梯度的方法**（The Gradient-based Method），属于基于一阶导数的方法。其他类似的方法还有牛顿法、共轭方向法、拟牛顿法等。在具体实现时，公式(9.38)可以有以下不同的形式。

### 1) 批量梯度下降 (Batch Gradient Descent)

批量梯度下降是梯度下降方法中最原始的形式，这种梯度下降方法在每一次迭

<sup>6</sup>梯度下降的一种实现是**最速下降**（Steepest Descent）。该方法的每一步移动都选取合适的步长，进而使目标函数能得到最大程度的增长（或下降）。

<sup>7</sup>图中的  $\theta^{[1]}$  和  $\theta^{[2]}$  分别是参数  $\theta$  的不同变化方向

代时使用所有的样本进行参数更新。参数优化的目标函数如下：

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta}) \quad (9.39)$$

公式(9.39)是公式(9.38)的严格实现，也就是将全部训练样本的平均损失作为目标函数。由全数据集确定的方向能够更好地代表样本总体，从而朝着模型在数据上整体优化所在的方向更新参数。

不过，这种方法的缺点也十分明显，因为要在全部训练数据上最小化损失，每一次参数更新都需要计算在所有样本上的损失。在使用海量数据进行训练的情况下，这种计算是非常消耗时间的。当训练数据规模很大时，很少使用这种方法。

## 2) 随机梯度下降 (Stochastic Gradient Descent)

随机梯度下降（简称 SGD）不同于批量梯度下降，每次迭代只使用一个样本对参数进行更新。SGD 的目标函数如下：

$$J(\boldsymbol{\theta}) = L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta}) \quad (9.40)$$

由于每次只随机选取一个样本  $(\mathbf{x}^{[i]}, \mathbf{y}^{[i]})$  进行优化，这样更新的计算代价低，参数更新的速度大大加快，而且也适用于利用少量样本进行在线学习的情况<sup>8</sup>。

因为随机梯度下降算法每次优化的只是某一个样本上的损失，所以它的问题也非常明显：单个样本上的损失无法代表在全部样本上的损失，因此参数更新的效率低，方法收敛速度极慢。即使在目标函数为强凸函数的情况下，SGD 仍旧无法做到线性收敛。

## 3) 小批量梯度下降 (Mini-batch Gradient Descent)

为了综合批量梯度下降和随机梯度下降的优缺点，在实际应用中一般采用这两个算法的折中——小批量梯度下降。其思想是：每次迭代计算一小部分训练数据的损失函数，并对参数进行更新。这一小部分数据被称为一个批次（mini-batch 或者 batch）。小批量梯度下降的参数优化的目标函数如下：

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=j}^{j+m-1} L(\mathbf{x}^{[i]}, \mathbf{y}^{[i]}; \boldsymbol{\theta}) \quad (9.41)$$

其中， $m$  表示一个批次中的样本的数量， $j$  表示这个批次在全体训练数据的起始位置。这种方法可以更充分的利用 GPU 设备，因为批次中的样本可以一起计算。而且每次使用多个样本可以大大减小使模型收敛所需要的参数更新次数。但是需要注意

---

<sup>8</sup>比如，训练数据不是一次给定的，而是随着模型的使用不断追加的。这时，需要不断地用新的训练样本更新模型，这种模式也被称作**在线学习**（Online Learning）。

的是批次大小的选择对模型的最终性能是存在一定影响的。

## 2. 梯度获取

梯度下降算法的一个核心是要得到目标函数相对于参数的梯度。下面将介绍三种常见的求梯度方法：数值微分、符号微分和自动微分，深度学习实现过程中多是采用自动微分方法计算梯度<sup>[410]</sup>。

### 1) 数值微分 (Numerical Differentiation)

数学上，梯度的求解其实就是求函数偏导的问题。导数是用极限来定义的，如公式(9.42)所示：

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta} = \lim_{\Delta\theta \rightarrow 0} \frac{L(\boldsymbol{\theta} + \Delta\theta) - L(\boldsymbol{\theta} - \Delta\theta)}{2\Delta\theta} \quad (9.42)$$

其中， $\theta$  表示参数的一个很小的变化值。公式(9.42)也被称作导数的双边定义。如果一个函数是初等函数，可以用求导法则来求得其导函数。如果不知道函数导数的解析式，则必须利用数值方法来求解该函数在某个点上的导数，这种方法就是数值微分。

数值微分根据导数的原始定义完成，根据公式可知，要得到损失函数在某个参数状态  $\theta$  下的梯度，可以将  $\theta$  增大或减小一点 ( $\Delta\theta$ )，例如，取  $|\Delta\theta| = 0.0001$ ，之后观测损失函数的变化与  $\Delta\theta$  的比值。 $\Delta\theta$  的取值越小计算的结果越接近导数的真实值，但是对计算的精度要求越高。

这种求梯度的方法很简单，但是计算量很大，求解速度非常慢，而且这种方法会造成截断误差 (Truncation Error) 和舍入误差 (Round-off Error)。在网络比较复杂、参数量稍微有点大的模型上一般不会使用这种方法。

截断误差和舍入误差是如何造成的呢？数值微分方法求梯度时，需用极限或无穷过程来求得。然而计算机需要将求解过程化为一系列有限的算术运算和逻辑运算。这样就要对某种无穷过程进行“截断”，即仅保留无穷过程的前段有限序列而舍弃它的后段。这就带来截断误差；舍入误差，是指运算得到的近似值和精确值之间的差异。由于数值微分方法计算复杂函数的梯度问题时，经过无数次的近似，每一次近似都产生了舍入误差，在这样的情况下，误差会随着运算次数增加而积累得很大，最终得出没有意义的运算结果。实际上，截断误差和舍入误差在训练复杂神经网络中，特别是使用低精度计算时，也会出现，因此是实际系统研发中需要注意的问题。

尽管数值微分不适用于大模型中的梯度求解，但是由于其非常简单，因此经常被用于检验其他梯度计算方法的正确性。比如在实现反向传播的时候（详见9.4.6节），可以检验求导是否正确 (Gradient Check)，这个过程就是利用数值微分实现的。

### 2) 符号微分 (Symbolic Differentiation)

顾名思义，符号微分就是通过建立符号表达式求解微分的方法：借助符号表达式

和求导公式，推导出目标函数关于自变量的微分表达式，最后再带入具体数值得到微分结果。例如，对于表达式  $L(\theta) = x \cdot \theta + 2\theta^2$ ，可以手动推导出微分表达式  $\frac{\partial L(\theta)}{\partial \theta} = x + 4\theta$ ，最后将具体数值  $x = (2 -3)$  和  $\theta = (-1 1)$  带入后，得到微分结果  $\frac{\partial L(\theta)}{\partial \theta} = (2 -3) + 4(-1 1) = (-2 1)$ 。

使用这种求梯度的方法，要求必须将目标函数转化成一种完整的数学表达式，这个过程中存在**表达式膨胀** (Expression Swell) 的问题，很容易导致符号微分求解的表达式急速“膨胀”，大大增加系统存储和处理表达式的负担。关于这个问题的一个实例请看表9.4。在深层的神经网络中，神经元数量和参数量极大，损失函数的表达式会非常冗长，不易存储和管理，而且，仅仅写出损失函数的微分表达式就是一个很庞大的工作量。从另一方面来说，这里真正需要的是微分的结果值，而不是微分表达式，推导微分表达式仅仅是求解过程中的中间产物。

表 9.4 符号微分的表达式随函数的规模增加而膨胀

函数	微分表达式	化简的微分表达式
$x$	1	1
$x \cdot (x + 1)$	$(x + 1) + x$	$2x + 1$
$x \cdot (x + 1) \cdot (x^2 + x + 1)$	$(x + 1) \cdot (x^2 + x + 1) + x \cdot (x^2 + x + 1) + x \cdot (x + 1) \cdot (2x + 1)$	$4x^3 + 6x^2 + 4x + 1$
$(x^2 + x) \cdot (x^2 + x + 1) \cdot (x^4 + 2x^3 + 2x^2 + x + 1) \cdot (x^4 + 2x^3 + 2x^2 + x + 1) + 2x^2 + x + 1$	$(2x + 1) \cdot (x^2 + x + 1) \cdot (x^4 + 2x^3 + 2x^2 + x + 1) + (2x + 1) \cdot (x^2 + x) \cdot (x^4 + 2x^3 + 2x^2 + x + 1) + (x^2 + x) \cdot (x^2 + x + 1) \cdot (4x^3 + 6x^2 + 4x + 1)$	$8x^7 + 28x^6 + 48x^5 + 50x^4 + 36x^3 + 18x^2 + 6x + 1$

### 3) 自动微分 (Automatic Differentiation)

自动微分是一种介于数值微分和符号微分的方法：将符号微分应用于最基本的算子，如常数、幂函数、指数函数、对数函数、三角函数等，然后代入数值，保留中间结果，最后再应用于整个函数。通过这种方式，将复杂的微分变成了简单的步骤，这些步骤完全自动化，而且容易进行存储和计算。

由于它只对基本函数或常数运用符号微分法则，所以它非常适合嵌入编程语言的循环条件等结构中，形成一种程序化的微分过程。在具体实现时，自动微分往往被当做是一种基于图的计算，相关的理论和技术方法相对成熟，因此是深度学习中使用最广泛的一种方法。不同于一般的编程模式，图计算先生成计算图，然后按照计算图执行计算过程。

自动微分可以用一种**反向模式** (Reverse Mode/Backward Mode) 即反向传播思想

进行描述<sup>[410]</sup>。令  $\mathbf{h}_i$  是神经网络的计算图中第  $i$  个节点的输出。反向模式的自动微分是要计算：

$$\bar{\mathbf{h}}_i = \frac{\partial L}{\partial \mathbf{h}_i} \quad (9.43)$$

这里， $\bar{\mathbf{h}}_i$  表示损失函数  $L$  相对于  $\mathbf{h}_i$  的梯度信息，它会被保存在节点  $i$  处。为了计算  $\bar{\mathbf{h}}_i$ ，需要从网络的输出反向计算每一个节点处的梯度。具体实现时，这个过程由一个包括前向计算和反向计算的两阶段方法实现。

首先，从神经网络的输入，逐层计算每层网络的输出值。如图9.31所示，第  $i$  层的输出  $\mathbf{h}_i$  作为第  $i+1$  层的输入，数据流在神经网络内部逐层传递。

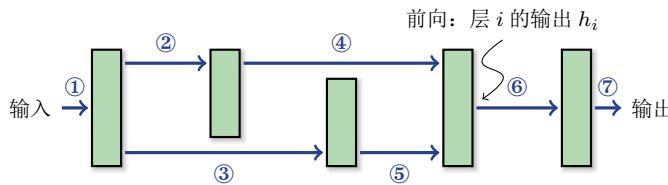


图 9.31 前向计算示意图

前向计算实际上就是网络构建的过程，所有的计算都会被转化为计算图上的节点，前向计算和反向计算都依赖计算图来完成。构建计算图有以下两种实现方式：

- 动态图：前向计算与计算图的搭建同时进行，函数表达式写完即能得到前向计算的结果，有着灵活、易于调试的优点。
- 静态图：先搭建计算图，后执行运算，函数表达式完成后，并不能得到前向计算结果，需要显性调用一个 `Forward` 函数。但是计算图可以进行深度优化，执行效率较高。

对于反向计算的实现，一般从神经网络的输出开始，逆向逐层计算每层网络输入所对应的微分结果。如图9.32所示，在第  $i$  层计算此处的梯度  $\frac{\partial L}{\partial \mathbf{h}_i}$ ，并将微分值向前一层传递，根据链式法则继续计算梯度。

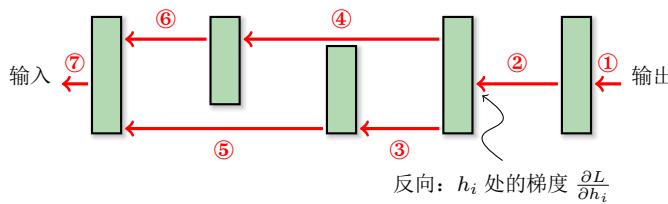


图 9.32 反向计算示意图

反向计算也是深度学习中反向传播方法的基础。其实现的内部细节将在9.4.6节详细阐述，所以在这里不再赘述。

### 3. 基于梯度的方法的变种和改进

参数优化通常基于梯度下降算法，即在每个更新步骤  $t$ ，沿梯度反方向更新参数，该过程如下：

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\partial J(\theta_t)}{\partial \theta_t} \quad (9.44)$$

其中， $\alpha$  是一个超参数，表示更新步幅的大小，称作学习率。当然，这是一种最基本的梯度下降方法。如果函数的形状非均向，比如呈延伸状，搜索最优点的路径就会非常低效，因为这时梯度的方向并没有指向最小值的方向，并且随着参数的更新，梯度方向往往呈锯齿状，这将是一条相当低效的路径；此外这种梯度下降算法并不是总能到达最优点，而是在其附近徘徊；还有一个最令人苦恼的问题——设置学习率，如果学习率设置的比较小，会导致训练收敛速度慢，如果学习率设置的比较大，会导致训练过程中因为优化幅度过大而频频跳过最优点。我们希望网络在优化的时候损失函数有一个很好的收敛速度同时又不至于摆动幅度太大。

针对以上问题，很多学者尝试对梯度下降方法做出改进，如 Momentum<sup>[411]</sup>，AdaGrad<sup>[412]</sup>，Adadelta<sup>[413]</sup>，RMSProp<sup>[414]</sup>，Adam<sup>[415]</sup>，AdaMax<sup>[415]</sup>，Nadam<sup>[416]</sup>，AMSGrad<sup>[417]</sup> 等等，在这里将介绍 Momentum、AdaGrad、RMSProp、Adam 这 4 种方法。

#### 1) Momentum

Momentum 梯度下降算法的参数更新方式如公式(9.45)和(9.46)所示<sup>9</sup>：

$$v_t = \beta v_{t-1} + (1 - \beta) \frac{\partial J}{\partial \theta_t} \quad (9.45)$$

$$\theta_{t+1} = \theta_t - \alpha v_t \quad (9.46)$$

该算法引入了一个“动量”的理念<sup>[411]</sup>，它是基于梯度的移动指数加权平均。公式中的  $v_t$  是损失函数在前  $t - 1$  次更新中累积的梯度动量， $\beta$  是梯度累积的一个指数，这里一般设置值为 0.9。所以 Momentum 梯度下降算法的主要思想就是对网络的参数进行平滑处理，让梯度的摆动幅度变得更小。

这里的“梯度”不再只是现在的损失函数的梯度，而是之前的梯度的加权和。在原始的梯度下降算法中，如果在某个参数状态下，梯度方向变化特别大，甚至与上一次参数更新中梯度方向成 90 度夹角，下一次参数更新中梯度方向可能又一次 90 度的改变，这时参数优化路径将会成“锯齿”状（如图9.33所示），优化效率极慢。而 Momentum 梯度下降算法不会让梯度发生 90 度的变化，而是让梯度慢慢发生改变：如果当前的梯度方向与之前的梯度方向相同，在原梯度方向上加速更新参数；如果当前的梯度方向与之前的梯度方向相反，并不会产生一个急转弯，而是尽量把优化路径平滑地进行改变。这样做的优点也非常明显，一方面杜绝了“锯齿”状优化路

<sup>9</sup>在梯度下降算法的几种改进方法的公式中，其更新对象是某个具体参数而非参数矩阵，因此不再使用加粗样式。

径的出现，另一方面将优化幅度变得更加平滑，不会导致频频跳过最优点。

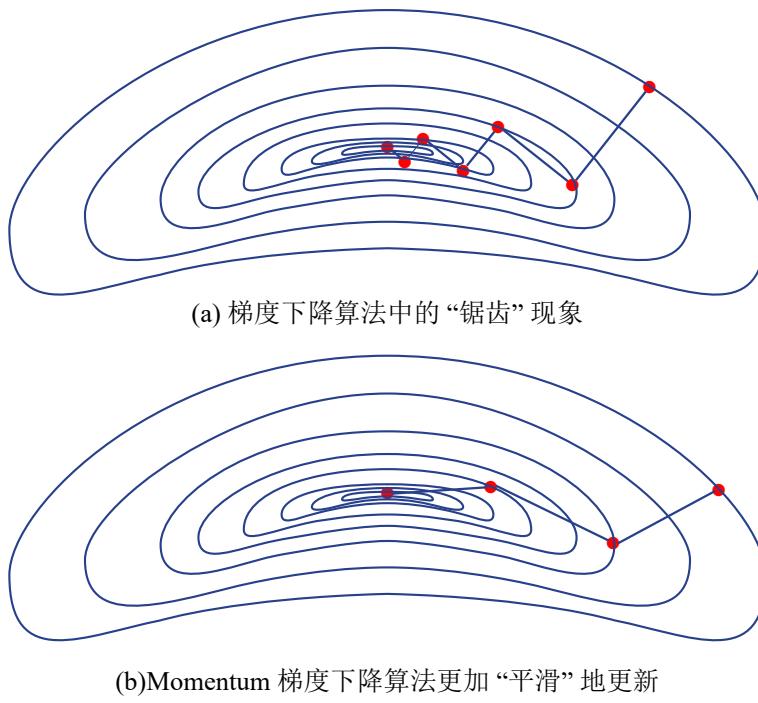


图 9.33 Momentum 梯度下降 vs 普通梯度下降

## 2) AdaGrad

在神经网络的学习中，学习率的设置很重要。学习率过小，会导致学习花费过多时间；反过来，学习率过大，则会导致学习发散，甚至造成模型的“跑偏”。在深度学习实现过程中，有一种被称为学习率**衰减**（Decay）的方法，即最初设置较大的学习率，随着学习的进行，使学习率逐渐减小，这种方法相当于将“全体”参数的学习率值一起降低。AdaGrad 梯度下降算法进一步发展了这个思想<sup>[412]</sup>。

AdaGrad 会为参数的每个元素适当地调整学习率，与此同时进行学习。其参数更新方式如公式(9.47)和(9.48)所示：

$$z_t = z_{t-1} + \frac{\partial J}{\partial \theta_t} \cdot \frac{\partial J}{\partial \theta_t} \quad (9.47)$$

$$\theta_{t+1} = \theta_t - \eta \frac{1}{\sqrt{z_t}} \cdot \frac{\partial J}{\partial \theta_t} \quad (9.48)$$

这里新出现了变量  $z$ ，它保存了以前的所有梯度值的平方和。如公式(9.48)所示，在更新参数时，通过除以  $\sqrt{z_t}$ ，就可以调整学习的尺度。这意味着，变动较大（被大幅度更新）的参数的学习率将变小。也就是说，可以按参数的元素进行学习率衰减，使变动大的参数的学习率逐渐减小。

### 3) RMSProp

RMSProp 算法是一种自适应学习率的方法<sup>[414]</sup>，它是对 AdaGrad 算法的一种改进，可以避免 AdaGrad 算法中学习率不断单调下降以至于过早衰减的缺点。

RMSProp 算法沿袭了 Momentum 梯度下降算法中指数加权平均的思路，不过 Momentum 算法中加权平均的对象是梯度（即  $\frac{\partial J}{\partial \theta}$ ），而 RMSProp 算法加权平均的对象是梯度的平方（即  $\frac{\partial J}{\partial \theta} \cdot \frac{\partial J}{\partial \theta}$ ）。RMSProp 算法的参数更新方式如公式(9.49)和(9.50)所示：

$$z_t = \gamma z_{t-1} + (1 - \gamma) \frac{\partial J}{\partial \theta_t} \cdot \frac{\partial J}{\partial \theta_t} \quad (9.49)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{z_t + \epsilon}} \cdot \frac{\partial J}{\partial \theta_t} \quad (9.50)$$

公式中的  $\epsilon$  是为了维持数值稳定性而添加的常数，一般可设为  $10^{-8}$ 。和 AdaGrad 的想法类似，模型参数中每个元素都拥有各自的学习率。

RMSProp 与 AdaGrad 相比，学习率的分母部分（即两种梯度下降算法迭代公式中的  $z$ ）的计算由累积方式变成了指数衰减移动平均。于是，每个参数的学习率并不是呈衰减趋势，而是既可以变小也可以变大，从而避免 AdaGrad 算法中学习率不断单调下降以至于过早衰减的缺点。

### 4) Adam

Adam 梯度下降算法是在 RMSProp 算法的基础上进行改进的，可以将其看成是带有动量项的 RMSProp 算法<sup>[415]</sup>。该算法在自然语言处理领域非常流行。Adam 算法的参数更新方式如公式(9.51)(9.52)(9.53)所示：

$$v_t = \beta v_{t-1} + (1 - \beta) \frac{\partial J}{\partial \theta_t} \quad (9.51)$$

$$z_t = \gamma z_{t-1} + (1 - \gamma) \frac{\partial J}{\partial \theta_t} \cdot \frac{\partial J}{\partial \theta_t} \quad (9.52)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{z_t + \epsilon}} v_t \quad (9.53)$$

可以看到 Adam 算法相当于在 RMSProp 算法中引入了 Momentum 算法中的动量项，这样做使得 Adam 算法兼具了 Momentum 算法和 RMSProp 算法的优点：既能使梯度更为“平滑”地更新，同时可以为神经网络中的每个参数设置不同的学习率。

需要注意的是包括 Adam 在内的很多参数更新算法中的学习率都需要人为设置。而且模型学习的效果与学习率的设置关系极大，甚至在研发实际系统时工程师需要进行大量的实验，才能得到最佳的模型。

### 9.4.3 参数更新的并行化策略

当神经网络较为复杂时，模型训练还是需要几天甚至几周的时间。如果希望尽可能缩短一次学习所需的时间，最直接的想法就是把不同的训练样本分配给多个GPU或CPU，然后在这些设备上同时进行训练，即实现并行化训练。这种方法也被称作**数据并行**。具体实现时，有两种常用的并行化策略：（参数）同步更新和（参数）异步更新。

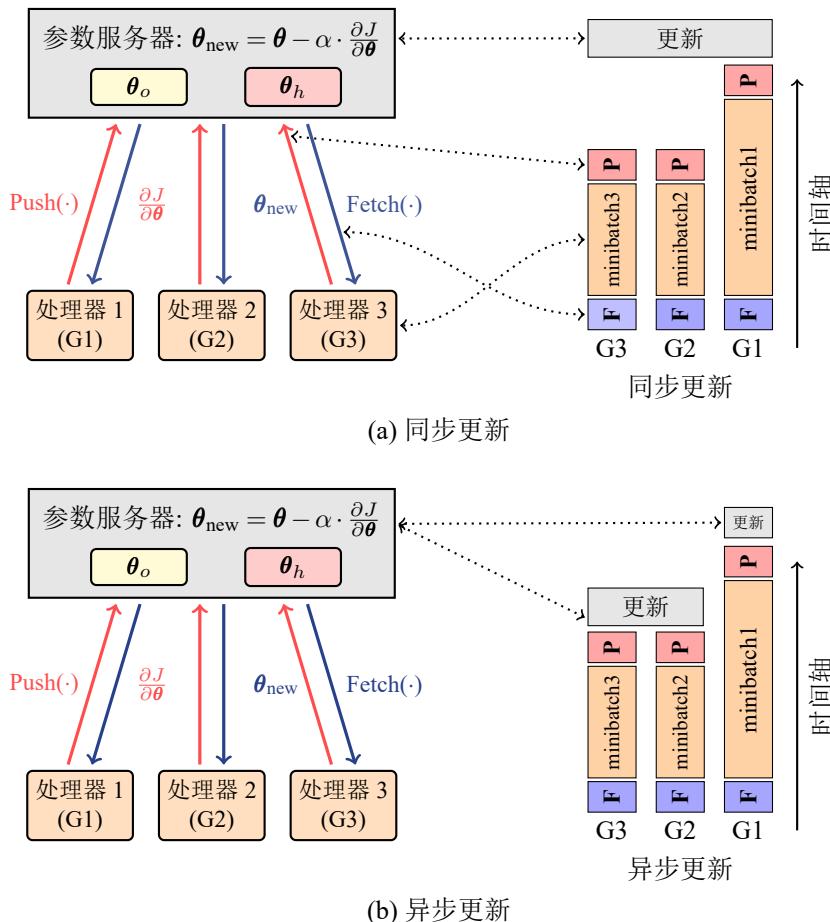


图 9.34 同步更新与异步更新对比

- **同步更新** (Synchronous Update) 是指所有计算设备完成计算后，统一汇总并更新参数。当所有设备的反向传播算法完成之后同步更新参数，不会出现单个设备单独对参数进行更新的情况。这种方法效果稳定，但是效率比较低，在同步更新时，每一次参数更新都需要所有设备统一开始、统一结束，如果设备的运行速度不一致，那么每一次参数更新都需要等待最慢的设备结束才能开始。
- **异步更新** (Asynchronous Update) 是指每个计算设备可以随时更新参数。不同设备可以随时读取参数的最新值，然后根据当前参数值和分配的训练样本，各

自执行反向传播过程并独立更新参数。由于设备间不需要相互等待，这种方法并行度高。但是不同设备读取参数的时间可能不同，会造成不同设备上的参数不同步，导致这种方法不十分稳定，有可能无法达到较好的训练结果。

图9.34对比了同步更新和异步更新的区别，在这个例子中，使用4台设备对一个两层神经网络中的参数进行更新，其中使用了一个**参数服务器**（Parameter Server）来保存最新的参数，不同设备（图中的G1、G2、G3）可以通过同步或者异步的方式访问参数服务器。图中的 $\theta_o$ 和 $\theta_h$ 分别代表输出层和隐藏层的全部参数，操作Push( $\cdot$ )表示设备向参数服务器传送梯度，操作Fetch( $\cdot$ )表示参数服务器向设备传送更新后的参数。

此外，在使用多个设备进行并行训练的时候，由于设备间带宽的限制，大量的数据传输会有较高的延时。对于复杂神经网络来说，设备间参数和梯度传递的时间消耗也会成为一个不得不考虑的因素。有时候，设备间数据传输的时间甚至比模型计算的时间都长，大大降低了并行度<sup>[418]</sup>。对于这种问题，可以考虑对数据进行压缩或者减少传输的次数来缓解问题。

#### 9.4.4 梯度消失、梯度爆炸和稳定性训练

深度学习中随着神经网络层数的增加，导数可能会出现指数级的下降或者指数级的增加，这种现象分别称为**梯度消失**（Gradient Vanishing）和**梯度爆炸**（Gradient Explosion）。出现这两种现象的本质原因是反向传播过程中链式法则导致梯度矩阵的多次相乘。这类问题很容易导致训练的不稳定。

##### 1. 易于优化的激活函数

网络训练过程中，如果每层网络的梯度都小于1，各层梯度的偏导数会与后面层传递而来的梯度相乘得到本层的梯度，并向前一层传递。该过程循环进行，最后导致梯度指数级地减小，这就产生了梯度消失现象。这种情况会导致神经网络层数较浅的部分梯度接近0。一般来说，产生很小梯度的原因是使用了类似于Sigmoid这样的激活函数，当输入的值过大或者过小的时候这类函数曲线会趋于直线，梯度近似为零。针对这个问题，主要的解决办法是使用更加易于优化的激活函数，比如，使用ReLU代替Sigmoid和Tanh作为激活函数。

##### 2. 梯度裁剪

网络训练过程中，如果参数的初始值过大，而且每层网络的梯度都大于1，反向传播过程中，各层梯度的偏导数都会比较大，会导致梯度指数级地增长直至超出浮点数表示的范围，这就产生了梯度爆炸现象。如果发生这种情况，模型中离输入近的部分比离输入远的部分参数更新得更快，使网络变得非常不稳定。在极端情况下，模型的参数值变得非常大，甚至于溢出。针对梯度爆炸的问题，常用的解决办法为**梯度裁剪**（Gradient Clipping）。

梯度裁剪的思想是设置一个梯度剪切阈值。在更新梯度的时候，如果梯度超过这个阈值，就将其强制限制在这个范围之内。假设梯度为  $\mathbf{g}$ ，梯度剪切阈值为  $\sigma$ ，梯度裁剪过程可描述为下式：

$$\mathbf{g} = \min\left(\frac{\sigma}{\|\mathbf{g}\|}, 1\right)\mathbf{g} \quad (9.54)$$

其中， $\|\cdot\|$  表示  $l_2$  范数。梯度裁剪经常被使用在层数较多的模型中，如循环神经网络。

### 3. 稳定性训练

为了使神经网络模型训练更加稳定，通常还会考虑其他策略。

- **批量标准化**（Batch Normalization）。批量标准化，顾名思义，是以进行学习时的小批量样本为单位进行标准化<sup>[419]</sup>。具体而言，就是对神经网络隐层输出的每一个维度，沿着批次的方向进行均值为 0、方差为 1 的标准化。在深层神经网络中，每一层网络都可以使用批量标准化操作。这样使神经网络任意一层的输入不至于过大或过小，从而防止隐层中异常值导致模型状态的巨大改变。
- **层标准化**（Layer Normalization）。类似的，层标准化更多是针对自然语言处理这种序列处理任务<sup>[420]</sup>，它和批量标准化的原理是一样的，只是标准化操作是在序列上同一层网络的输出结果上进行的，也就是标准化操作沿着序列方向进行。这种方法可以很好的避免序列上不同位置神经网络输出结果的不可比性。同时由于标准化后所有的结果都转化到一个可比的范围，使得隐层状态可以在不同层之间进行自由组合。
- **残差网络**（Residual Networks）。最初，残差网络是为了解决神经网络持续加深时的模型退化问题<sup>[421]</sup>，但是残差结构对解决梯度消失和梯度爆炸问题也有所帮助。有了残差结构，可以很轻松的构建几十甚至上百层的神经网络，而不用担心层数过深造成梯度消失问题。残差网络的结构如图9.35所示。图9.35中右侧的曲线叫做**跳接**（Skip Connection），通过跳接在激活函数前，将上一层（或几层）之前的输出与本层计算的输出相加，将求和的结果输入到激活函数中作为本层的输出。假设残差结构的输入为  $\mathbf{x}_l$ ，输出为  $\mathbf{x}_{l+1}$ ，则有

$$\mathbf{x}_{l+1} = F(\mathbf{x}_l) + \mathbf{x}_l \quad (9.55)$$

相比较于简单的多层堆叠的结构，残差网络提供了跨层连接结构。这种结构在反向传播中有很大的好处，比如，对于一个训练样本，损失函数为  $L$ ， $\mathbf{x}_l$  处的梯度的计算方式如公式(9.56)所示。残差网络可以将后一层的梯度  $\frac{\partial L}{\partial \mathbf{x}_{l+1}}$  不经过任何乘法项直接传递到  $\frac{\partial L}{\partial \mathbf{x}_l}$ ，从而缓解了梯度经过每一层后多次累乘造成的梯度消失问题。在第十二章中还会看到，在机器翻译中残差结构可以和层标准化

一起使用，而且这种组合可以取得很好的效果。

$$\begin{aligned}
 \frac{\partial L}{\partial \mathbf{x}_l} &= \frac{\partial L}{\partial \mathbf{x}_{l+1}} \cdot \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} \\
 &= \frac{\partial L}{\partial \mathbf{x}_{l+1}} \cdot \left( 1 + \frac{\partial F(\mathbf{x}_l)}{\partial \mathbf{x}_l} \right) \\
 &= \frac{\partial L}{\partial \mathbf{x}_{l+1}} + \frac{\partial L}{\partial \mathbf{x}_{l+1}} \cdot \frac{\partial F(\mathbf{x}_l)}{\partial \mathbf{x}_l}
 \end{aligned} \tag{9.56}$$

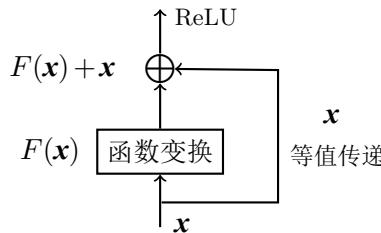


图 9.35 残差网络的结构

#### 9.4.5 过拟合

理想中，我们总是希望尽可能地拟合输入和输出之间的函数关系，即让模型尽量模拟训练数据中由输入预测答案的行为。然而，在实际应用中，模型在训练数据上的表现不一定代表了其在未见数据上的表现。如果模型训练过程中过度拟合训练数据，最终可能无法对未见数据做出准确的判断，这种现象叫做**过拟合**（Overfitting）。随着模型复杂度增加，特别在神经网络变得更深、更宽时，过拟合问题会表现得更为突出。如果训练数据量较小，而模型又很复杂，可以“完美”地拟合这些数据，这时过拟合也很容易发生。所以在模型训练时，往往不希望去完美拟合训练数据中的每一个样本。

**正则化**（Regularization）是常见的缓解过拟合问题的手段，通过在损失函数中加上用来刻画模型复杂程度的正则项来惩罚过度复杂的模型，从而避免神经网络过度学习造成过拟合。引入正则化处理之后目标函数变为  $J(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$ ，其中  $J(\boldsymbol{\theta})$  是原来的代价函数， $R(\boldsymbol{\theta})$  即为正则项， $\lambda$  用来调节正则项对结果影响的程度。

过拟合的模型通常会表现为部分非零参数过多或者参数的值过大。这种参数产生的原因在于模型需要复杂的参数才能匹配样本中的个别现象甚至噪声。基于此，常见的正则化方法有 L1 正则化和 L2 正则化，其命名方式是由  $R(\boldsymbol{\theta})$  的计算形式来决定的。在 L1 正则化中， $R(\boldsymbol{\theta})$  即为参数  $\boldsymbol{\theta}$  的  $l_1$  范数，即  $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_{i=1}^n |\theta_i|$ ；在 L2 正则化中， $R(\boldsymbol{\theta})$  即为参数  $\boldsymbol{\theta}$  的  $l_2$  范数的平方，即  $R(\boldsymbol{\theta}) = (\|\boldsymbol{\theta}\|_2)^2 = \sum_{i=1}^n \theta_i^2$ 。L1 正则化中的正则项衡量了模型中参数的绝对值大小，倾向于生成值为 0 的参数，从而让

参数变得更加稀疏；而 L2 正则化由于平方的加入，当参数中的某一项小到一定程度，比如 0.001 的时候，参数的平方结果已经可以忽略不计了，因此 L2 正则化会倾向生成很小的参数，在这种情况下，即便训练数据中含有少量随机噪音，模型也不太容易通过增加个别参数的值来对噪声进行过度拟合，即提高了模型的抗扰动能力。

此外，在第十二章即将介绍的 Dropout 和标签平滑方法也可以被看作是一种正则化操作。它们都可以提高模型在未见数据上的泛化能力。

#### 9.4.6 反向传播

为了获取梯度，最常用的做法是使用自动微分技术，通常通过反向传播来实现。该方法分为两个计算过程：前向计算和反向计算。前向计算的目的是从输入开始，逐层计算，得到网络的输出，并记录计算图中每个节点的局部输出。反向计算过程从输出端反向计算梯度，这个过程可以被看作是一种梯度的“传播”，最终计算图中所有节点都会得到相应的梯度结果。

这里，首先对反向传播算法中涉及到的符号进行统一说明。图9.36是一个多层神经网络，其中层  $k-1$ 、层  $k$ 、层  $k+1$  均为神经网络中的隐藏层，层  $K$  为神经网络中的输出层。为了化简问题，这里每层网络没有使用偏置项。

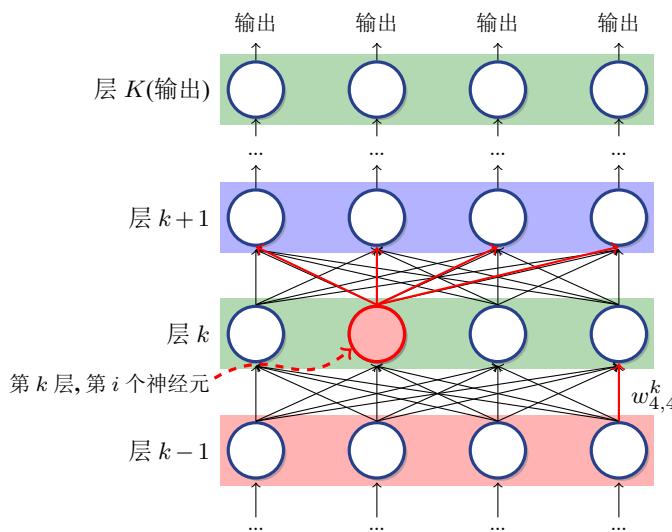


图 9.36 多层神经网络实例

下面是一些符号的定义：

- $h_i^k$ : 第  $k$  层第  $i$  个神经元的输出；
- $\mathbf{h}^k$ : 第  $k$  层的输出。若第  $k$  层有  $n$  个神经元，则：

$$\mathbf{h}^k = (h_1^k, h_2^k, \dots, h_n^k) \quad (9.57)$$

- $w_{j,i}^k$ : 第  $k-1$  层神经元  $j$  与第  $k$  层神经元  $i$  的连接权重;
- $\mathbf{W}^k$ : 第  $k-1$  层与第  $k$  层的连接权重。若第  $k-1$  层有  $m$  个神经元, 第  $k$  层有  $n$  个神经元, 则:

$$\mathbf{W}^k = \begin{pmatrix} w_{1,1}^k & w_{1,2}^k & \dots & w_{1,n}^k \\ w_{2,1}^k & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ w_{m,1}^k & \dots & \dots & w_{m,n}^k \end{pmatrix} \quad (9.58)$$

- $\mathbf{h}^K$ : 整个网络的输出;
- $\mathbf{s}^k$ : 第  $k$  层的线性变换结果, 其计算方式如下:

$$\begin{aligned} \mathbf{s}^k &= \mathbf{h}^{k-1} \mathbf{W}^k \\ &= \sum h_j^{k-1} w_{j,i}^k \end{aligned} \quad (9.59)$$

- $f^k$ : 第  $k$  层的激活函数,  $\mathbf{h}^k = f^k(\mathbf{s}^k)$ 。

于是, 在神经网络的第  $k$  层, 前向计算过程可以描述为:

$$\begin{aligned} \mathbf{h}^k &= f^k(\mathbf{s}^k) \\ &= f^k(\mathbf{h}^{k-1} \mathbf{W}^k) \end{aligned} \quad (9.60)$$

## 1. 输出层的反向传播

反向传播是由输出层开始计算梯度, 之后逆向传播到每一层网络, 直至到达输入层。这里首先讨论输出层的反向传播机制。输出层(即第  $K$  层)可以被描述为公式(9.61)(9.62):

$$\mathbf{h}^K = f^K(\mathbf{s}^K) \quad (9.61)$$

$$\mathbf{s}^K = \mathbf{h}^{K-1} \mathbf{W}^K \quad (9.62)$$

也就是, 输出层(第  $K$  层)的输入  $\mathbf{h}^{K-1}$  先经过线性变换右乘  $\mathbf{W}^K$  转换为中间状态  $\mathbf{s}^K$ , 之后  $\mathbf{s}^K$  再经过激活函数  $f^K(\cdot)$  变为  $\mathbf{h}^K$ ,  $\mathbf{h}^K$  即为第  $K$  层(输出层)的输出。最后,  $\mathbf{h}^K$  和标准答案一起计算得到损失函数的值<sup>10</sup>, 记为  $L$ 。以上过程如图9.37所示。

<sup>10</sup>在反向传播算法部分我们以某一个训练样本为例进行讲解, 因而这里不再计算代价函数  $J$ , 而是损失函数  $L$ 。

这里将输出层的前向计算过程细化为两个阶段：线性变换阶段和激活函数 + 损失函数阶段。

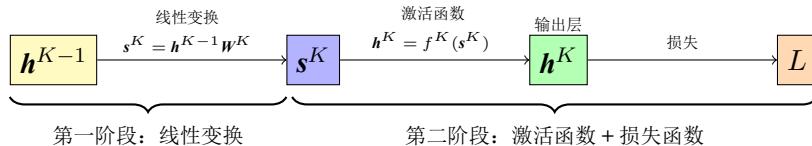


图 9.37 输出层的前向计算过程

在前向过程中，计算次序为  $\mathbf{h}^{K-1} \rightarrow \mathbf{s}^K \rightarrow \mathbf{h}^K \rightarrow L$ 。而反向计算中节点访问的次序与之相反：

- 第一步，获取  $\frac{\partial L}{\partial \mathbf{h}^K}$ ，即计算损失函数  $L$  关于网络输出结果  $\mathbf{h}^K$  的梯度，并将梯度向前传递；
- 第二步，获取  $\frac{\partial L}{\partial \mathbf{s}^K}$ ，即计算损失函数  $L$  关于中间状态  $\mathbf{s}^K$  的梯度，并将梯度向前传递；
- 第三步，获取  $\frac{\partial L}{\partial \mathbf{h}^{K-1}}$  和  $\frac{\partial L}{\partial \mathbf{w}^K}$ ，即计算损失函数  $L$  关于第  $K-1$  层输出结果  $\mathbf{h}^{K-1}$  的梯度，并将梯度向前传递；同时计算损失函数  $L$  关于第  $K$  层参数  $\mathbf{w}^K$  的梯度，并用于参数更新。

对于前两个步骤，如图9.38所示：

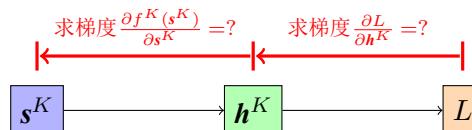


图 9.38 从损失到中间状态的反向传播（输出层）

在第一阶段，计算的目标是得到损失函数  $L$  关于第  $K$  层中间状态  $\mathbf{s}^K$  的梯度，这里令  $\boldsymbol{\pi}^K = \frac{\partial L}{\partial \mathbf{s}^K}$ ，利用链式法则有：

$$\begin{aligned}
 \boldsymbol{\pi}^K &= \frac{\partial L}{\partial \mathbf{s}^K} \\
 &= \frac{\partial L}{\partial \mathbf{h}^K} \cdot \frac{\partial \mathbf{h}^K}{\partial \mathbf{s}^K} \\
 &= \frac{\partial L}{\partial \mathbf{h}^K} \cdot \frac{\partial f^K(\mathbf{s}^K)}{\partial \mathbf{s}^K}
 \end{aligned} \tag{9.63}$$

其中：

- $\frac{\partial L}{\partial \mathbf{h}^K}$  表示损失函数  $L$  相对网络输出  $\mathbf{h}^K$  的梯度。比如，对于平方损失  $L = \frac{1}{2} \|\mathbf{y} - \mathbf{h}^K\|^2$ ，有  $\frac{\partial L}{\partial \mathbf{h}^K} = \mathbf{y} - \mathbf{h}^K$ 。计算结束后，将  $\frac{\partial L}{\partial \mathbf{h}^K}$  向前传递。

- $\frac{\partial f^T(s^K)}{\partial s^K}$  表示激活函数相对于其输入  $s^K$  的梯度。比如，对于 Sigmoid 函数  $f(s) = \frac{1}{1+e^{-s}}$ ，有  $\frac{\partial f(s)}{\partial s} = f(s)(1-f(s))$

这个过程可以得到  $s^K$  节点处的梯度  $\pi^K = \frac{\partial L}{\partial s^K}$ ，在后续的过程中可以直接使用其作为前一层提供的梯度计算结果，而不需要从  $h^K$  节点处重新计算。这也体现了自动微分与符号微分的差别，对于计算图的每一个阶段，并不需要得到完成的微分表达式，而是通过前一层提供的梯度，直接计算当前的梯度即可，这样避免了大量的重复计算。

在得到  $\pi^K = \frac{\partial L}{\partial s^K}$  之后，下一步的目标是：1) 计算损失函数  $L$  相对于第  $K-1$  层与输出层之间连接权重  $W^K$  的梯度；2) 计算损失函数  $L$  相对于神经网络第  $K-1$  层输出结果  $h^{K-1}$  的梯度。这部分内容如图9.39所示。

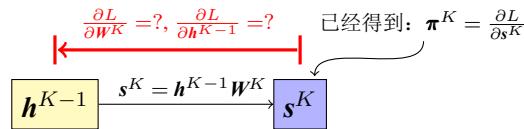


图 9.39 从中间状态到输入的反向传播（输出层）

具体来说：

- 计算  $\frac{\partial L}{\partial W^K}$ ：由于  $s^K = h^{K-1}W^K$ ，且损失函数  $L$  关于  $s^K$  的梯度  $\pi^K = \frac{\partial L}{\partial s^K}$  已经得到，于是有

$$\frac{\partial L}{\partial W^K} = [h^{K-1}]^T \pi^K \quad (9.64)$$

其中  $[.]^T$  表示转置操作<sup>11</sup>。

- 计算  $\frac{\partial L}{\partial h^{K-1}}$ ：与求解  $\frac{\partial L}{\partial W^K}$  类似，可以得到

$$\frac{\partial L}{\partial h^{K-1}} = \pi^K [W^K]^T \quad (9.65)$$

梯度  $\frac{\partial L}{\partial h^{K-1}}$  需要继续向前一层传递，用于计算网络中间层的梯度。 $\frac{\partial L}{\partial W^K}$  会作为参数  $W^K$  的梯度计算结果，用于模型参数的更新<sup>12</sup>。

<sup>11</sup>如果  $h^{K-1}$  是一个向量， $[h^{K-1}]^T$  表示向量的转置，比如，行向量变成列向量；如果  $h^{K-1}$  是一个高阶张量， $[h^{K-1}]^T$  表示沿着张量最后两个方向的转置。

<sup>12</sup> $W^K$  可能会在同一个网络中被多次使用（类似于网络不同部分共享同一个参数），这时需要累加相关计算节点处得到的  $\frac{\partial L}{\partial W^K}$ 。

## 2. 隐藏层的反向传播

对于第  $k$  个隐藏层，有：

$$\mathbf{h}^k = f^k(\mathbf{s}^k) \quad (9.66)$$

$$\mathbf{s}^k = \mathbf{h}^{k-1} \mathbf{W}^k \quad (9.67)$$

其中， $\mathbf{h}^k$ 、 $\mathbf{s}^k$ 、 $\mathbf{h}^{k-1}$ 、 $\mathbf{W}^k$  分别表示隐藏层的输出、中间状态、隐藏层的输入和参数矩阵。隐藏层的前向计算过程如图9.40所示，第  $k-1$  层神经元的输出  $\mathbf{h}^{k-1}$  经过线性变换和激活函数后，将计算结果  $\mathbf{h}^k$  向后一层传递。

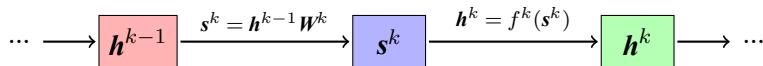


图 9.40 隐藏层的前向计算过程

与输出层类似，隐藏层的反向传播也是逐层逆向计算。

- 第一步，获取  $\frac{\partial L}{\partial \mathbf{s}^k}$ ，即计算损失函数  $L$  关于第  $k$  层中间状态  $\mathbf{s}^k$  的梯度，并将梯度向前传递；
- 第二步，获取  $\frac{\partial L}{\partial \mathbf{h}^{k-1}}$  和  $\frac{\partial L}{\partial \mathbf{W}^k}$ ，即计算损失函数  $L$  关于第  $k-1$  层输出结果  $\mathbf{h}^{k-1}$  的梯度，并将梯度向前传递。同时计算损失函数  $L$  关于参数  $\mathbf{W}^k$  的梯度，并用于参数更新。

这两步和输出层的反向传播十分类似。可以利用链式法则得到：

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{s}^k} &= \frac{\partial L}{\partial \mathbf{h}^k} \cdot \frac{\partial \mathbf{h}^k}{\partial \mathbf{s}^k} \\ &= \frac{\partial L}{\partial \mathbf{h}^k} \cdot \frac{\partial f^k(\mathbf{s}^k)}{\partial \mathbf{s}^k} \end{aligned} \quad (9.68)$$

其中  $\frac{\partial L}{\partial \mathbf{h}^k}$  表示损失函数  $L$  相对该隐藏层输出  $\mathbf{h}^k$  的梯度。进一步，由于  $\mathbf{s}^k = \mathbf{h}^{k-1} \mathbf{W}^k$ ，可以得到

$$\frac{\partial L}{\partial \mathbf{W}^k} = [\mathbf{h}^{k-1}]^T \cdot \frac{\partial L}{\partial \mathbf{s}^k} \quad (9.69)$$

$$\frac{\partial L}{\partial \mathbf{h}^{k-1}} = \frac{\partial L}{\partial \mathbf{s}^k} \cdot [\mathbf{W}^k]^T \quad (9.70)$$

$\frac{\partial L}{\partial \mathbf{h}^{k-1}}$  需要继续向第  $k-1$  隐藏层传递。 $\frac{\partial L}{\partial \mathbf{W}^k}$  会作为参数的梯度用于参数更新。图9.41展示了隐藏层反向传播的计算过程。

综合输出层和隐藏层的反向传播方法，可以得到神经网络中任意位置和任意参数的梯度信息。只需要根据网络的拓扑结构，逆向访问每一个节点，并执行上述反

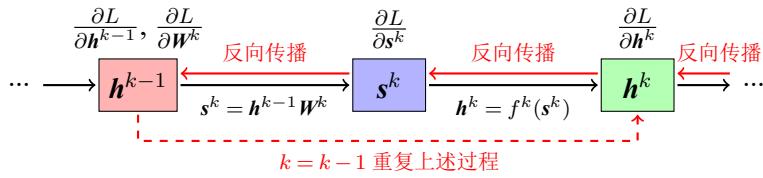


图 9.41 隐藏层的反向传播

向计算过程。

## 9.5 神经语言模型

神经网络提供了一种工具，只要将问题的输入和输出定义好，就可以学习输入和输出之间的对应关系。显然，很多自然语言处理任务都可以用神经网络进行实现。比如，在机器翻译中，可以把输入的源语言句子和输出的目标语言句子用神经网络建模；在文本分类中，可以把输入的文本内容和输出的类别标签进行神经网络建模，等等。

为了更好地理解神经网络和深度学习在自然语言处理中的应用。这里介绍一种基于神经网络的语言建模方法——**神经语言模型**（Neural Language Model）。可以说，神经语言模型是深度学习时代下自然语言处理的标志性成果，它所涉及的许多概念至今仍是研究的热点，比如：词嵌入、表示学习、预训练等。此外，神经语言模型也为机器翻译的建模提供了很好的思路。从某种意义上说，机器翻译的深度学习建模的很多灵感均来自神经语言模型，二者在一定程度上是统一的。

### 9.5.1 基于前馈神经网络的语言模型

回顾一下第二章的内容，语言建模的问题被定义为：对于一个词序列  $w_1 w_2 \dots w_m$ ，如何计算该词序列的可能性？词序列出现的概率可以通过链式法则得到：

$$P(w_1 w_2 \dots w_m) = P(w_1) P(w_2 | w_1) P(w_3 | w_1 w_2) \dots P(w_m | w_1 \dots w_{m-1}) \quad (9.71)$$

由于  $P(w_m | w_1 \dots w_{m-1})$  需要建模  $m-1$  个词构成的历史信息，这个模型仍然很复杂。于是就有了基于局部历史的  $n$ -gram 语言模型：

$$P(w_m | w_1 \dots w_{m-1}) = P(w_m | w_{m-n+1} \dots w_{m-1}) \quad (9.72)$$

其中， $P(w_m | w_{m-n+1} \dots w_{m-1})$  可以通过相对频次估计进行计算，如公式(9.73)所示，其中  $\text{count}(\cdot)$  表示在训练数据上的频次：

$$P(w_m | w_{m-n+1} \dots w_{m-1}) = \frac{\text{count}(w_{m-n+1} \dots w_m)}{\text{count}(w_{m-n+1} \dots w_{m-1})} \quad (9.73)$$

这里,  $w_{m-n+1} \dots w_m$  也被称作  $n$ -gram, 即  $n$  元语法单元。 $n$ -gram 语言模型是一种典型的基于离散表示的模型。在这个模型中, 所有的词都被看作是离散的符号。因此, 不同单词之间是“完全”不同的。另一方面, 语言现象是十分多样的, 即使在很大的语料库上也无法得到所有  $n$ -gram 的准确统计。甚至很多  $n$ -gram 在训练数据中从未出现过。由于不同  $n$ -gram 间没有建立直接的联系,  $n$ -gram 语言模型往往面临数据稀疏的问题。比如, 虽然在训练数据中见过“景色”这个词, 但是测试数据中却出现了“风景”这个词, 恰巧“风景”在训练数据中没有出现过。即使“风景”和“景色”表达的是相同的意思,  $n$ -gram 语言模型仍然会把“风景”看作未登录词, 赋予一个很低的概率值。

上面这个问题的本质是  $n$ -gram 语言模型对词使用了离散化表示, 即每个单词都孤立的对应词表中的一个索引, 词与词之间在语义上没有任何“重叠”。神经语言模型重新定义了这个问题。这里并不需要显性地通过统计离散的  $n$ -gram 的频度, 而是直接设计一个神经网络模型  $g(\cdot)$  来估计单词生成的概率, 如下所示:

$$P(w_m | w_1 \dots w_{m-1}) = g(w_1 \dots w_m) \quad (9.74)$$

$g(w_1 \dots w_m)$  实际上是一个多层次神经网络。与  $n$ -gram 语言模型不同的是  $g(w_1 \dots w_m)$  并不包含对  $w_1 \dots w_m$  的任何假设, 比如, 在神经网络模型中, 单词不再是离散的符号, 而是连续空间上的点。这样两个单词之间也不再是简单的非 0 即 1 的关系, 而是具有可计算的距离。此外, 由于没有对  $w_1 \dots w_m$  进行任何结构性的假设, 神经语言模型对问题进行端到端学习。通过设计不同的神经网络  $g(\cdot)$ , 可以从不同的角度“定义”序列的表示问题。当然, 这么说可能还有一些抽象, 下面就一起看看神经语言模型究竟是什么样子的。

## 1. 模型结构

最具代表性的神经语言模型是**前馈神经网络语言模型** (Feed-forward Neural Network Language Model, FNNLM)。这种语言模型的目标是用神经网络计算  $P(w_m | w_{m-n+1} \dots w_{m-1})$ , 之后将多个  $n$ -gram 的概率相乘得到整个序列的概率<sup>[72]</sup>。

为了有一个直观的认识, 这里以 4-gram 的 FNNLM 为例, 即根据前三个单词  $w_{i-3}$ 、 $w_{i-2}$ 、 $w_{i-1}$  预测当前单词  $w_i$  的概率。模型结构如图9.42所示。从结构上看, FNNLM 是一个典型的多层次神经网络结构。主要有三层:

- **输入层** (词的分布式表示层), 即把输入的离散的单词变为分布式表示对应的实数向量;
- **隐藏层**, 即将得到的词的分布式表示进行线性和非线性变换;
- **输出层** (Softmax 层), 根据隐藏层的输出预测单词的概率分布。

这三层堆叠在一起构成了整个网络, 而且也可以加入从词的分布式表示直接到输出层的连接 (红色虚线箭头)。

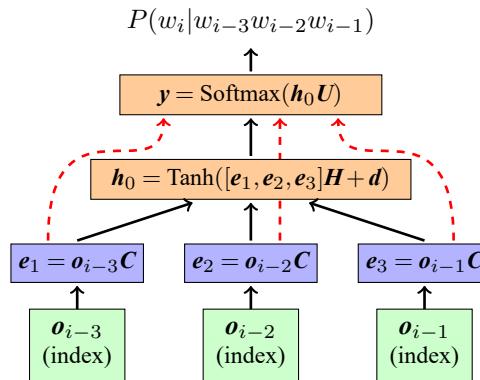


图 9.42 4-gram 前馈神经网络语言架构

## 2. 输入层

$\mathbf{o}_{i-3}$ 、 $\mathbf{o}_{i-2}$ 、 $\mathbf{o}_{i-1}$  为该语言模型的输入（绿色方框），输入为每个词（如上文的  $w_{i-1}$ 、 $w_{i-2}$  等）的 One-hot 向量表示（维度大小与词表大小一致），每个 One-hot 向量仅一维为 1，其余为 0，比如： $(0, 0, 1, \dots, 0)$  表示词表中第三个单词。之后把 One-hot 向量乘以一个矩阵  $\mathbf{C}$  得到单词的分布式表示（紫色方框）。令  $\mathbf{o}_i$  为第  $i$  个词的 One-hot 表示， $\mathbf{e}_i$  为第  $i$  个词的分布式表示，则分布式表示  $\mathbf{e}_i$  的计算方式如下：

$$\mathbf{e}_i = \mathbf{o}_i \cdot \mathbf{C} \quad (9.75)$$

这里的  $\mathbf{C}$  可以被理解为一个查询表，根据  $\mathbf{o}_i$  中为 1 的那一维，在  $\mathbf{C}$  中索引到相应的行进行输出（结果是一个行向量）。通常，把  $\mathbf{e}_i$  这种单词的实数向量表示称为词嵌入，把  $\mathbf{C}$  称为词嵌入矩阵。

## 3. 隐藏层和输出层

把得到的  $\mathbf{e}_0$ 、 $\mathbf{e}_1$ 、 $\mathbf{e}_2$  三个向量级联在一起，经过两层网络，最后通过 Softmax 函数（橙色方框）得到输出，具体过程为：

$$\mathbf{y} = \text{Softmax}(\mathbf{h}_0 \mathbf{U}) \quad (9.76)$$

$$\mathbf{h}_0 = \text{Tanh}([\mathbf{e}_{i-3}, \mathbf{e}_{i-2}, \mathbf{e}_{i-1}] \mathbf{H} + \mathbf{d}) \quad (9.77)$$

这里，输出  $\mathbf{y}$  是词表  $V$  上的一个分布，来表示  $P(w_i | w_{i-1}, w_{i-2}, w_{i-3})$ 。 $\mathbf{U}$ 、 $\mathbf{H}$  和  $\mathbf{d}$  是模型的参数。这样，对于给定的单词  $w_i$  可以用  $y_i$  得到其概率，其中  $y_i$  表示向量  $\mathbf{y}$  的第  $i$  维。

Softmax( $\cdot$ ) 的作用是根据输入的  $|V|$  维向量（即  $\mathbf{h}_0 \mathbf{U}$ ），得到一个  $|V|$  维的分布。

令  $\tau$  表示  $\text{Softmax}(\cdot)$  的输入向量,  $\text{Softmax}$  函数可以被定义为:

$$\text{Softmax}(\tau_i) = \frac{\exp(\tau_i)}{\sum_{i'=1}^{|V|} \exp(\tau_{i'})} \quad (9.78)$$

这里,  $\exp(\cdot)$  表示指数函数。 $\text{Softmax}$  函数是一个典型的归一化函数, 它可以将输入的向量的每一维都转化为 0-1 之间的数, 同时保证所有维的和等于 1。 $\text{Softmax}$  的另一个优点是, 它本身 (对于输出的每一维) 都是可微的 (如图9.43所示), 因此可以直接使用基于梯度的方法进行优化。实际上,  $\text{Softmax}$  经常被用于分类任务。也可以把机器翻译中目标语单词的生成看作一个分类问题, 它的类别数是  $|V|$ 。

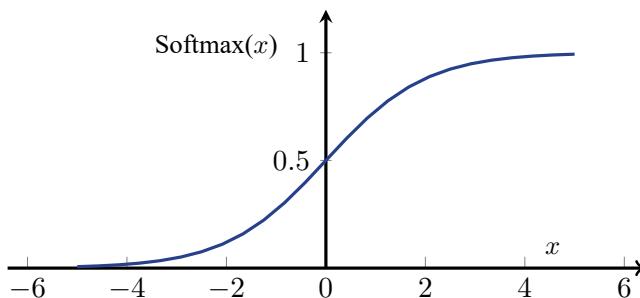


图 9.43 Softmax 函数 (一维) 所对应的曲线

#### 4. 连续空间表示能力

值得注意的是, 在 FNNLM 中, 单词已经不再是一个孤立的符号串, 而是被表示为一个实数向量。这样, 两个单词之间可以通过向量计算某种相似度或距离。这导致相似的单词会具有相似的分布, 进而缓解  $n$ -gram 语言模型的问题——明明意思很相近的两个词但是概率估计的结果差异性却很大。

在 FNNLM 中, 所有的参数、输入、输出都是连续变量, 因此 FNNLM 也是一个典型的连续空间模型。通过使用交叉熵等损失函数, 可以很容易地对 FNNLM 进行优化。比如, 可以使用梯度下降方法对 FNNLM 的模型参数进行训练。

虽然 FNNLM 形式简单, 却为处理自然语言提供了一个全新的视角。首先, 该模型重新定义了“词是什么”——它并非词典的一项, 而是可以用一个连续实数向量进行表示的可计算的“量”。此外, 由于  $n$ -gram 不再是离散的符号序列, 模型不需要记录  $n$ -gram, 所以很好的缓解了上面所提到的数据稀疏问题, 模型体积也大大减小。

当然, FNNLM 也引发后人的许多思考, 比如: 神经网络每一层都学到了什么? 是词法、句法, 还是一些其他知识? 如何理解词的分布式表示? 等等。在随后的内容中也会看到, 随着近几年深度学习和自然语言处理的发展, 部分问题已经得到了很好的解答, 但是仍有许多问题需要进一步探索。

### 9.5.2 对于长序列的建模

FNNLM 固然有效，但是和传统的  $n$ -gram 语言模型一样需要依赖有限上下文假设，也就是  $w_i$  的生成概率只依赖于之前的  $n - 1$  个单词。很自然的一个想法是引入更大范围的历史信息，这样可以捕捉单词间的长距离依赖。

#### 1. 基于循环神经网络的语言模型

对于长距离依赖问题，可以通过**循环神经网络**（Recurrent Neural Network，或 RNN）进行求解。通过引入循环单元这种特殊的结构，循环神经网络可以对任意长度的历史进行建模，因此在一定程度上解决了传统  $n$ -gram 语言模型有限历史的问题。正是基于这个优点，**循环神经网络语言模型**（RNNLM）应运而生<sup>[73]</sup>。

在循环神经网络中，输入和输出都是一个序列，分别记为  $(x_1, \dots, x_m)$  和  $(y_1, \dots, y_m)$ 。它们都可以被看作是时序序列，其中每个时刻  $t$  都对应一个输入  $x_t$  和输出  $y_t$ 。循环神经网络的核心是**循环单元**（RNN Cell），它读入前一个时刻循环单元的输出和当前时刻的输入，生成当前时刻循环单元的输出。图9.44展示了一个简单的循环单元结构，对于时刻  $t$ ，循环单元的输出被定义为：

$$\mathbf{h}_t = \text{Tanh}(\mathbf{x}_t \mathbf{U} + \mathbf{h}_{t-1} \mathbf{W}) \quad (9.79)$$

其中， $\mathbf{h}_t$  表示  $t$  时刻循环单元的输出， $\mathbf{h}_{t-1}$  表示  $t - 1$  时刻循环单元的输出， $\mathbf{U}$  和  $\mathbf{W}$  是模型的参数。可以看出，循环单元的结构其实很简单，只是一个对  $\mathbf{h}_{t-1}$  和  $\mathbf{x}_t$  的线性变换再加上一个 Tanh 函数。通过读入上一时刻的输出，当前时刻可以访问以前的历史信息。这个过程可以循环执行，这样就完成了对所有历史信息的建模。 $\mathbf{h}_t$  可以被看作是序列在  $t$  时刻的一种表示，也可以被看作是网络的一个隐藏层。进一步， $\mathbf{h}_t$  可以被送入输出层，得到  $t$  时刻的输出：

$$\mathbf{y}_t = \text{Softmax}(\mathbf{h}_t \mathbf{V}) \quad (9.80)$$

其中， $\mathbf{V}$  是输出层的模型参数。

图9.44展示了一个基于循环神经网络的语言模型结构。首先，所有输入的单词会被转换成分布式表示（红色部分），这个过程和 FNNLM 是一样的。之后，该模型堆叠了两层循环神经网络（绿色部分）。最后通过 Softmax 层（紫色部分）得到每个时刻的预测结果  $\mathbf{y}_t = P(w_t | w_1 \dots w_{t-1})$ 。

RNNLM 体现了一种“记忆”的能力。对于每一个时刻，循环单元都会保留一部分“以前”的信息，并加入“现在”的信息。从这个角度说，RNNLM 本质上是一种记忆模型。在简单的循环单元结构的基础上，也有很多改进工作，如 LSTM、GRU 等模型，这部分内容将会在第十章进行介绍。

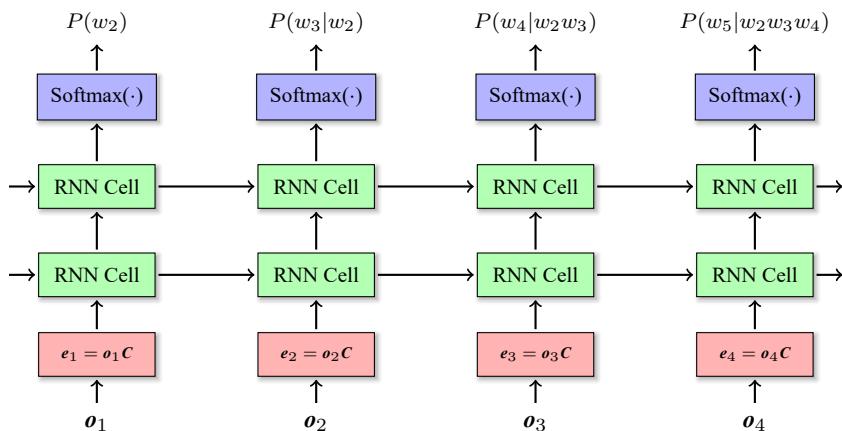


图 9.44 基于循环神经网络的语言模型结构

## 2. 其他类型的语言模型

通过引入记忆历史的能力，RNNLM 缓解了  $n$ -gram 模型中有限上下文的局限性，但依旧存在一些问题。随着序列变长，不同单词之间信息传递路径变长，信息传递的效率变低。对于长序列，很难通过很多次的循环单元操作保留很长的历史信息。过长的序列还容易引起梯度消失和梯度爆炸问题（详见 9.4.4 节），增加模型训练的难度。

针对这个问题，一种解决方法是使用卷积神经网络<sup>[422]</sup>。卷积神经网络的特点是可以对一定窗口大小内的连续单词进行统一建模，这样非常易于捕捉窗口内单词之间的依赖，同时对它们进行整体的表示。进一步，卷积操作可以被多次叠加使用，通过更多层的卷积神经网络可以捕捉更大范围的依赖关系。关于卷积神经网络及其在机器翻译中的应用，第十一章会有详细论述。

此外，研究者也提出了另一种新的结构——**自注意力机制**（Self-attention Mechanism）。自注意力是一种特殊的神经网络结构，它可以对序列上任意两个词的相互作用直接进行建模，这样也就避免了循环神经网络中随着距离变长信息传递步骤增多的缺陷。在自然语言处理领域，自注意力机制被成功地应用在机器翻译任务上，著名的 Transformer 模型<sup>[23]</sup> 就是基于该原理工作的。第十二章会系统地介绍自注意力机制和 Transformer 模型。

### 9.5.3 单词表示模型

在神经语言建模中，每个单词都会被表示为一个实数向量。这对应了一种单词的表示模型。下面就来看看传统的单词表示模型和这种基于实数向量的单词表示模型有何不同。

#### 1. One-hot 编码

**One-hot 编码**（也称**独热编码**）是传统的单词表示方法。One-hot 编码把单词表示为词汇表大小的 0-1 向量，其中只有该词所对应的那一项是 1，而其余所有项都是 0。

举个简单的例子，假如有一个词典，里面包含 10k 个单词，并进行编号。那么每个单词都可以表示为一个 10k 维的 One-hot 向量，它仅在对应编号那个维度为 1，其他维度都为 0，如图9.45所示。

$\cos(\text{‘桌子’}, \text{‘椅子’}) = 0$		
	桌子	椅子
你 <sub>1</sub>	[0]	[0]
桌子 <sub>2</sub>	[1]	[0]
他 <sub>3</sub>	[0]	[0]
椅子 <sub>4</sub>	[0]	[1]
我们 <sub>5</sub>	[0]	[0]
...	[...]	[...]
你好 <sub>10k</sub>	[0]	[0]

图 9.45 单词的 One-hot 表示

One-hot 编码的优点是形式简单、易于计算，而且这种表示与词典具有很好的对应关系，因此每个编码都可以进行解释。但是，One-hot 编码把单词都看作是相互正交的向量。这导致所有单词之间没有任何的相关性。只要是不同的单词，在 One-hot 编码下都是完全不同的东西。比如，大家可能会期望诸如“桌子”和“椅子”之类的词具有一些相似性，但是 One-hot 编码把它们看作相似度为 0 的两个单词。

## 2. 分布式表示

神经语言模型中使用的是一种分布式表示。在神经语言模型里，每个单词不再是完全正交的 0-1 向量，而是在多维实数空间中的一个点，具体表现为一个实数向量。很多时候，也会把单词的这种分布式表示叫做词嵌入。

单词的分布式表示可以被看作是欧式空间中的一个点，因此单词之间的关系也可以通过空间的几何性质进行刻画。如图9.46所示，可以在一个 512 维空间上表示不同的单词。在这种表示下，“桌子”与“椅子”之间是具有一定的联系的。

$\cos(\text{‘桌子’}, \text{‘椅子’}) = 0.5$		
	桌子	椅子
属性 <sub>1</sub>	[0.1]	[1]
属性 <sub>2</sub>	[-1]	[2]
属性 <sub>3</sub>	[2]	[0.2]
...	[...]	[...]
属性 <sub>512</sub>	[0]	[-1]

图 9.46 单词的分布式表示(词嵌入)

那么，分布式表示中每个维度的含义是什么？可以把每一维度都理解为一种属性，比如一个人的身高、体重等。但是，神经网络模型更多的是把每个维度看作是单词的一种抽象“刻画”，是一种统计意义上的“语义”，而非简单的人工归纳的事物的

一个个属性。使用这种连续空间的表示的好处在于，表示的内容（实数向量）可以进行计算和学习，因此可以通过模型训练得到更适用于自然语言处理的单词表示结果。

为了方便理解，看一个简单的例子。假如现在有个“预测下一个单词”的任务：有这样一个句子“屋里/要/摆放/一个/\_\_\_\_\_”，其中下划线的部分表示需要预测的下一个单词。如果模型在训练数据中看到过类似于“摆放一个桌子”这样的片段，那么就可以很自信的预测出“桌子”。另一方面，很容易知道，实际上与“桌子”相近的单词，如“椅子”，也是可以预测的单词的。但是，“椅子”恰巧没有出现在训练数据中，这时如果用 One-hot 编码来表示单词，显然无法把“椅子”填到下划线处；而如果使用单词的分布式表示，很容易就知道“桌子”与“椅子”是相似的，因此预测“椅子”在一定程度上也是合理的。

- 实例 9.1** 屋里/要/摆放/一个/\_\_\_\_\_ 预测下个词  
 屋里/要/摆放/一个/**桌子** 见过  
 屋里/要/摆放/一个/**椅子** 没见过，但是仍然是合理预测

关于单词的分布式表示还有一个经典的例子：通过词嵌入可以得到如下关系：“国王” = “女王” – “女人” + “男人”。从这个例子可以看出，词嵌入也具有一些代数性质，比如，词的分布式表示可以通过加、减等代数运算相互转换。图9.47展示了词嵌入在一个二维平面上的投影，不难发现，含义相近的单词分布比较临近。



图 9.47 分布式表示的可视化

语言模型的词嵌入是通过词嵌入矩阵进行存储的，矩阵中的每一行对应了一个词的分布式表示结果。图9.48展示了一个词嵌入矩阵的实例。

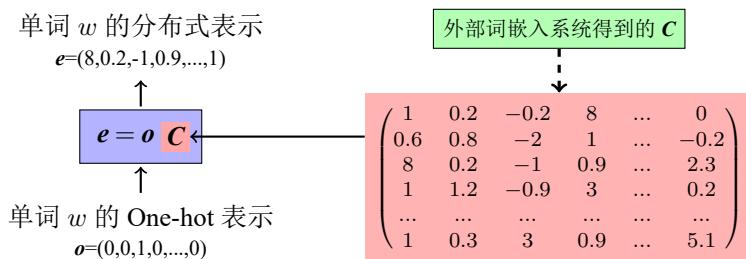


图 9.48 词嵌入矩阵  $\mathbf{C}$

通常，有两种方法得到词嵌入矩阵。一种方法是把词嵌入作为语言模型的一部

分进行训练，不过由于语言模型往往较复杂，这种方法非常耗时；另一种方法使用更加轻便的外部训练方法，如 word2vec<sup>[423]</sup>、Glove<sup>[166]</sup>等。由于这些方法的效率较高，因此可以使用更大规模的数据得到更好的词嵌入结果。

#### 9.5.4 句子表示模型

目前，词嵌入已经成为诸多自然语言处理系统的标配，也衍生出很多有趣的研究方向。但是，冷静地看，词嵌入依旧存在一些问题：每个词都对应唯一的向量表示，那么对于一词多义现象，词义需要通过上下文进行区分，这时使用简单的词嵌入式是无法处理的。有一个著名的例子：

**实例 9.2** Aaron is an employee of apple.

He finally ate the apple.

这两句中“apple”的语义显然是不同的，第一句中的上下文“Jobs”和“CEO”可以帮助我们判断“apple”是一个公司名字，而不是水果。但是词嵌入只有一个结果，因此无法区分这两种情况。这个例子给我们一个启发：在一个句子中，不能孤立的看待单词，应同时考虑其上下文的信息。也就是需要一个能包含句子中上下文信息的表示模型。

回忆一下神经语言模型的结构，它需要在每个位置预测单词生成的概率。这个概率是由若干层神经网络进行计算后，通过输出层得到的。实际上，在送入输出层之前，系统已经得到了这个位置的一个向量（隐藏层的输出），因此可以把它看作是含有一部分上下文信息的表示结果。

以 RNNLM 为例，图9.49展示了一个由四个词组成的句子，这里使用了一个两层循环神经网络对其进行建模。可以看到，对于第三个位置，RNNLM 已经积累了从第 1 个单词到第 3 个单词的信息，因此可以看作是单词 1-3（“乔布斯就职于”）的一种表示；另一方面，第 4 个单词的词嵌入可以看作是“苹果”自身的表示。这样，可以把第 3 个位置 RNNLM 的输出和第 4 个位置的词嵌入进行合并，就得到了第 4 个位置上含有上下文信息的表示结果。从另一个角度说，这里得到了“苹果”的一种新的表示，它不仅包含苹果这个词自身的信息，也包含它前文的信息。

在自然语言处理中，**句子表示模型**是指把输入的句子进行分布式表示。不过表示的形式不一定是一个单独的向量。现在广泛使用的句子表示模型可以被描述为：给定一个输入的句子  $\{w_1, \dots, w_m\}$ ，得到一个表示序列  $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$ ，其中  $\mathbf{h}_i$  是句子在第  $i$  个位置的表示结果。 $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$  就被看作是**句子的表示**，它可以被送入下游模块。比如，在机器翻译任务中，可以用这种模型表示源语言句子，然后通过这种表示结果进行目标语译文的生成；在序列标注（如词性标注）任务中，可以对输入的句子进行表示，然后在这个表示之上构建标签预测模块。很多自然语言处理任务都可以用句子表示模型进行建模，因此句子的表示模型也是应用最广泛的深度学习模型之一。而学习这种表示的过程也被称作表示学习。

句子表示模型有两种训练方法。最简单的方法是把它作为目标系统中的一个模

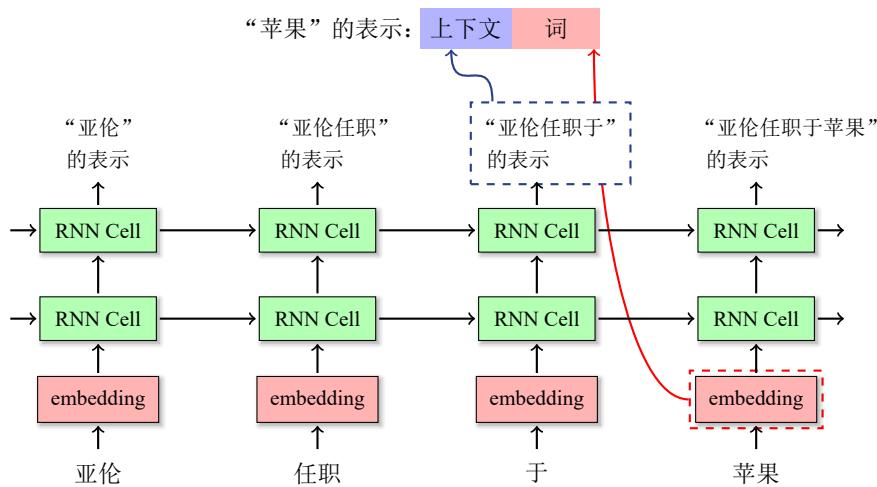


图 9.49 基于 RNN 的表示模型（词 + 上下文）

块进行训练，比如把句子表示模型作为机器翻译系统的一部分。也就是，并不单独训练句子表示模型，而是把它作为一个内部模块放到其他系统中。另一种方法是把句子表示作为独立的模块，用外部系统进行训练，之后把训练好的表示模型放入目标系统中，再进行微调。这种方法构成了一种新的范式：预训练 + 微调（pre-training + fine-tuning）。图9.50对比了这两种不同的方法。

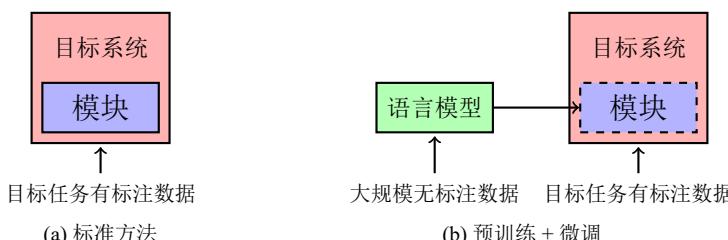


图 9.50 表示模型的训练方法（与目标任务联合训练 vs 用外部任务预训练）

目前，句子表示模型的预训练方法在多项自然语言处理任务上取得了很好的效果。预训练模型也成为了当今自然语言处理中的热点方向，相关系统也在很多评测任务上刷榜。不过，上面介绍的模型是一种最简单句子表示模型，在第十六章会对一些前沿的预训练方法和句子表示模型进行介绍。

## 9.6 小结及拓展阅读

神经网络为解决自然语言处理问题提供了全新的思路。而所谓深度学习也是建立在多层神经网络结构之上的一系列模型和方法。本章从神经网络的基本概念到其在语言建模中的应用进行了概述。由于篇幅所限，这里无法覆盖所有神经网络和深度学习的相关内容，感兴趣的读者可以进一步阅读 *Neural Network Methods in Natural Language Processing*。

*Language Processing*<sup>[31]</sup> 和 *Deep Learning*<sup>[30]</sup>。此外，也有一些研究方向值得关注：

- 端到端学习是神经网络方法的特点之一。这样，系统开发者不需要设计输入和输出的隐含结构，甚至连特征工程都不再需要。但是，另一方面，由于这种端到端学习完全由神经网络自行完成，整个学习过程没有人的先验知识做指导，导致学习的结构和参数很难进行解释。针对这个问题也有很多研究者进行**可解释机器学习**（Explainable Machine Learning）的研究<sup>[424, 425, 426]</sup>。对于自然语言处理，方法的可解释性是十分必要的。从另一个角度说，如何使用先验知识改善端到端学习也是很多人关注的方向<sup>[427, 428]</sup>，比如，如何使用句法知识改善自然语言处理模型<sup>[429, 430, 431, 432, 433]</sup>。
- 为了进一步提高神经语言模型性能，除了改进模型，还可以在模型中引入新的结构或是其他有效信息，该领域也有很多典型工作值得关注。例如在神经语言模型中引入除了词嵌入以外的单词特征，如语言特征（形态、语法、语义特征等）<sup>[434, 435]</sup>、上下文信息<sup>[405, 436]</sup>、知识图谱等外部知识<sup>[437]</sup>；或是在神经语言模型中引入字符级信息，将其作为字符特征单独<sup>[438, 439]</sup> 或与单词特征一起<sup>[440, 441]</sup> 送入模型中；在神经语言模型中引入双向模型也是一种十分有效的尝试，在单词预测时可以同时利用来自过去和未来的文本信息<sup>[22, 165, 442]</sup>。
- 词嵌入是自然语言处理近些年的重要进展。所谓“嵌入”是一类方法，理论上，把一个事物进行分布式表示的过程都可以被看作是广义上的“嵌入”。基于这种思想的表示学习也成为了自然语言处理中的前沿方法。比如，如何对树结构，甚至图结构进行分布式表示成为了分析自然语言的重要方法<sup>[443, 444, 445, 446, 447]</sup>。此外，除了语言建模，还有很多方式可以进行词嵌入的学习，比如，SENNNA<sup>[100]</sup>、word2vec<sup>[162, 423]</sup>、Glove<sup>[166]</sup>、CoVe<sup>[448]</sup> 等。





## 10. 基于循环神经网络的模型

**神经机器翻译**（Neural Machine Translation）是机器翻译的前沿方法。近几年，随着深度学习技术的发展和在各领域中的深入应用，基于端到端表示学习的方法正在改变着我们处理自然语言的方式，神经机器翻译在这种趋势下应运而生。一方面，神经机器翻译仍然延续着统计建模和基于数据驱动的思想，因此在基本问题的定义上与前人的研究是一致的；另一方面，神经机器翻译脱离了统计机器翻译中对隐含翻译结构的假设，同时使用分布式表示来对文字序列进行建模，这使得它可以从一个全新的视角看待翻译问题。现在，神经机器翻译已经成为了机器翻译研究及应用的热点，译文质量得到了巨大的提升。

本章将介绍神经机器翻译中的一种基础模型——基于循环神经网络的模型。该模型是神经机器翻译中最早被成功应用的模型之一。基于这个模型框架，研究人员进行了大量的探索和改进工作，包括使用 LSTM 等循环单元结构、引入注意力机制等。这些内容都会在本章进行讨论。

### 10.1 神经机器翻译的发展简史

纵观机器翻译的发展历程，神经机器翻译诞生较晚。无论是早期的基于规则的方法，还是逐渐发展起来的基于实例的方法，再或是上世纪末的统计方法，每次机器翻译框架级的创新都需要很长时期的酝酿，而技术走向成熟甚至需要更长的时间。但是，神经机器翻译的出现和后来的发展速度多少有些“出人意料”。神经机器翻译的概念出现在 2013-2014 年间，当时机器翻译领域的主流方法仍然是统计机器翻译。

虽然那个时期深度学习已经在图像、语音等领域取得令人瞩目的效果，但是对于自然语言处理来说深度学习仍然不是主流。

不过，研究人员也意识到了神经机器翻译在表示学习等方面的优势。这一时期，很多研究团队对包括机器翻译在内的序列到序列问题进行了广泛而深入的研究，注意力机制等新的方法不断被推出。这使得神经机器翻译系统在翻译品质上逐渐体现出优势，甚至超越了当时的统计机器翻译系统。正当大家在讨论神经机器翻译是否能取代统计机器翻译成为下一代机器翻译范式的时候，一些互联网企业推出了以神经机器翻译技术为内核的在线机器翻译服务，在很多场景下的翻译品质显著超越了当时最好的统计机器翻译系统。这也引发了学术界和产业界对神经机器翻译的讨论。随着关注度的不断升高，神经机器翻译的研究吸引了更多的科研机构和企业的投入，神经机器翻译系统的翻译品质得到进一步提升。

在短短 5-6 年间，神经机器翻译从一个新生的概念已经成长为机器翻译领域的最前沿技术之一，在各种机器翻译评测和应用中呈全面替代统计机器翻译之势。比如，从近几年 WMT、CCMT 等评测的结果来看，神经机器翻译已经处于绝对的统治地位，在不同语种和领域的翻译任务中，成为各参赛系统的标配。此外，从 ACL 等自然语言处理顶级会议的发表论文看，神经机器翻译在论文数量上呈明显的增长趋势，这也体现了学术界对该方法的热情。至今，国内外的很多机构都推出了自己研发的神经机器翻译系统，整个研究和产业生态欣欣向荣。图10.1展示了包含神经机器翻译在内的机器翻译发展简史。

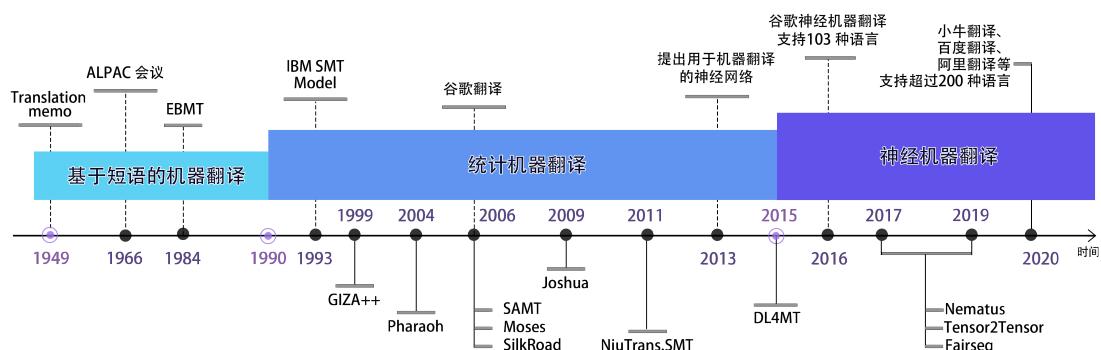


图 10.1 机器翻译发展简史

神经机器翻译的迅速崛起确实让所有研究人员都有些措手不及，甚至有一种一觉醒来天翻地覆的感觉。也有研究人员评价，神经机器翻译的出现给整个机器翻译领域带来了前所未有的发展机遇。不过，客观地看，机器翻译达到今天这样的状态也是一种历史必然，其中有几方面原因：

- 自上世纪末所发展起来的基于数据驱动的方法为神经机器翻译提供了很好的基础。本质上，神经机器翻译仍然是一种基于统计建模的数据驱动的方法，因此无论是对问题的基本建模方式，还是训练统计模型所使用到的带标注数据，

都可以复用机器翻译领域以前的研究成果。特别是机器翻译长期的发展已经积累了大量的双语、单语数据，这些数据在统计机器翻译时代就发挥了很大作用。随着时间的推移，数据规模和质量又得到进一步提升，包括一些评测基准、任务设置都已经非常完备，研究人员可以直接在数据条件全部具备的情况下开展神经机器翻译的研究工作，这些都节省了大量的时间成本。从这个角度说，神经机器翻译是站在巨人的肩膀上才发展起来的。

- 深度学习经过长时间的酝酿终于爆发，为机器翻译等自然语言处理任务提供了新的思路和技术手段。神经机器翻译的不断壮大伴随着深度学习技术的发展。在深度学习的视角下，语言文字可以被表示成抽象的实数向量，这种文字的表示结果可以被自动学习，为机器翻译建模提供了更大的灵活性。相对于神经机器翻译，深度学习的发展更加曲折。虽然深度学习经过了漫长的起伏过程，但是神经机器翻译恰好出现在深度学习逐渐走向成熟的阶段。反过来说，受到深度学习及相关技术空前发展的影响，自然语言处理的范式也发生了变化，神经机器翻译的出现只是这种趋势下的一种必然。
- 此外，计算机算力的提升也为神经机器翻译提供了很好的支撑。与很多神经网络方法一样，神经机器翻译也依赖大量基于浮点数的矩阵运算。甚至在本世纪初，大规模的矩阵运算仍然依赖非常昂贵的 CPU 集群系统，但是随着 GPU 等相关技术的发展，在相对低成本的设备上已经可以完成非常复杂的浮点并行运算。这使得包括神经机器翻译在内的很多基于深度学习的系统可以进行大规模实验，随着实验周期的缩短，相关研究和系统的迭代周期也大大缩短。实际上，计算机硬件运算能力一直是稳定提升的，神经机器翻译只是受益于运算能力的阶段性突破。
- 还有，翻译需求的不断增加也为机器翻译技术提供了新的机会。在近几年，无论是翻译品质，还是翻译语种数量，甚至不同的翻译场景，都对机器翻译有了更高的要求。人们迫切需要一种品质更高、翻译效果稳定的机器翻译方法，神经机器翻译恰好满足了这些要求。当然，应用端需求的增加也会反推机器翻译技术的发展，二者相互促进。

至今，神经机器翻译已经成为带有时代特征的标志性方法。当然，机器翻译的发展也远没有达到终点。下面将介绍神经机器翻译的起源和优势，以便读者在正式了解神经机器翻译的技术方法前对其现状有一个充分的认识。

### 10.1.1 神经机器翻译的起源

从广义上讲，神经机器翻译是一种基于人工神经网络的方法，它把翻译过程描述为可以用人工神经网络表示的函数，所有的训练和推断都在这些函数上进行。由于神经机器翻译中的神经网络可以用连续可微函数表示，因此这类方法也可以用基于梯度的方法进行优化，相关技术非常成熟。更为重要的是，在神经网络的设计中，研究人员引入了分布式表示的概念，这也是近些年自然语言处理领域的重要成果之一。

传统统计机器翻译仍然把词序列看作离散空间里的由多个特征函数描述的点，类似于  $n$ -gram 语言模型，这类模型对数据稀疏问题非常敏感。此外，人工设计特征也在一定程度上限制了模型对问题的表示能力。神经机器翻译把文字序列表示为实数向量，一方面避免了特征工程繁重的工作，另一方面使得系统可以对文字序列的“表示”进行学习。可以说，神经机器翻译的成功很大程度上源自“表示学习”这种自然语言处理的新范式的出现。在表示学习的基础上，注意力机制、深度神经网络等技术都被应用于神经机器翻译，使其得以进一步发展。

虽然神经机器翻译中大量地使用了人工神经网络方法，但是它并不是最早在机器翻译中使用人工神经网络的框架。实际上，人工神经网络在机器翻译中应用的历史要远早于现在的神经机器翻译。在统计机器翻译时代，也有很多研究人员利用人工神经网络进行机器翻译系统模块的构建<sup>[449, 450]</sup>，比如，研究人员成功地在统计机器翻译系统中使用了基于神经网络的联合表示模型，取得了很好的效果<sup>[449]</sup>。

不过，以上这些工作大多都是在系统的局部模块中使用人工神经网络和深度学习方法。与之不同的是，神经机器翻译是用人工神经网络完成整个翻译过程的建模，这样做的一个好处是，整个系统可以进行端到端学习，无需引入对任何翻译的隐含结构假设。这种利用端到端学习对机器翻译进行神经网络建模的方式也就成为了现在大家所熟知的神经机器翻译。这里简单列出部分代表性的工作：

- 早在 2013 年，Nal Kalchbrenner 和 Phil Blunsom 提出了一个基于编码器-解码器结构的新模型<sup>[451]</sup>。该模型用卷积神经网络（CNN）将源语言编码成实数向量，之后用循环神经网络（RNN）将连续向量转换成目标语言。这使得模型不需要进行词对齐、特征提取等工作，就能够自动学习源语言的信息。这也是一种端到端学习的方法。不过，这项工作的实现较复杂，而且方法存在梯度消失/爆炸等问题<sup>[452, 453]</sup>，因此并没有成为后来神经机器翻译的基础框架。
- 2014 年，Ilya Sutskever 等人提出了序列到序列（seq2seq）学习的方法，同时将长短时记忆结构（LSTM）引入到神经机器翻译中，这个方法缓解了梯度消失/爆炸的问题，并且通过遗忘门的设计让网络选择性地记忆信息，缓解了序列中长距离依赖的问题<sup>[21]</sup>。但是该模型在进行编码的过程中，将不同长度的源语言句子压缩成了一个固定长度的向量，句子越长，损失的信息越多，同时该模型无法对输入和输出序列之间的对齐进行建模，因此并不能有效的保证翻译质量。
- 同年 Dzmitry Bahdanau 等人首次将**注意力机制**（Attention Mechanism）应用到机器翻译领域，在机器翻译任务上对翻译和局部翻译单元之间的对应关系同时建模<sup>[22]</sup>。Bahdanau 等人工作的意义在于，使用了更加有效的模型来表示源语言的信息，同时使用注意力机制对两种语言不同部分之间的相互联系进行建模。这种方法可以有效地处理长句子的翻译，而且注意力的中间结果具有一定的可解释性<sup>1</sup>。然而相比于前人的神经机器翻译模型，注意力模型也引入了额外的成

<sup>1</sup> 比如，目标语言和源语言句子不同单词之间的注意力强度能够在一定程度上反应单词之间的互译

本，计算量较大。

- 2016年谷歌公司发布了基于多层循环神经网络方法的GNMT系统。该系统集成了当时的神经机器翻译技术，并进行了诸多的改进。它的性能显著优于基于短语的机器翻译系统<sup>[454]</sup>，引起了研究人员的广泛关注。在之后不到一年的时间里，脸书公司采用卷积神经网络（CNN）研发了新的神经机器翻译系统<sup>[24]</sup>，实现了比基于循环神经网络（RNN）系统更高的翻译水平，并大幅提升翻译速度。
- 2017年，Ashish Vaswani等人提出了新的翻译模型Transformer。其完全摒弃了循环神经网络和卷积神经网络，仅仅通过多头注意力机制和前馈神经网络，不需要使用序列对齐的循环框架就展示出强大的性能，并且巧妙地解决了翻译中长距离依赖问题<sup>[23]</sup>。Transformer是第一个完全基于注意力机制搭建的模型，不仅训练速度更快，在翻译任务上也获得了更好的结果，一跃成为目前最主流的神经机器翻译框架。

当然，神经机器翻译的工作远不止以上这些内容<sup>[455]</sup>。随着本书内容的逐渐深入，很多经典的模型和方法都会被讨论到。

### 10.1.2 神经机器翻译的品质

图10.2展示了用机器翻译把一段汉语翻译为英语的结果。其中译文1是统计机器翻译系统的结果，译文2是神经机器翻译系统的结果。为了保证公平性，两个系统使用完全相同的数据进行训练。

原文：This has happened for a whole range of reasons, not least because we live in a culture where people are encouraged to think of sleep as a luxury - something you can easily cut back on. After all, that's what caffeine is for - to jolt you back into life. But while the average amount of sleep we are getting has fallen, rates of obesity and diabetes have soared. Could the two be connected?

译文1：这已经发生了一系列的原因，不仅仅是因为我们生活在一个文化鼓励人们认为睡眠是一种奢侈的东西，你可以很容易地削减。毕竟，这就是咖啡因是—你回到生命的震动。但是，尽管我们得到的平均睡眠量下降，肥胖和糖尿病率飙升。可以两个连接？

译文2：这种情况的发生有各种各样的原因，特别是因为我们生活在一种鼓励人们把睡眠看作是一种奢侈的东西—你可以很容易地减少睡眠的文化中。毕竟，这就是咖啡因的作用—让你重新回到生活中。但是，当我们的平均睡眠时间减少时，肥胖症和糖尿病的发病率却猛增。这两者有联系吗？

图 10.2 机器翻译实例对比

可以明显地看到译文2更加通顺，意思的表达更加准确，翻译质量明显高于译文1。这个例子基本反应出统计机器翻译和神经机器翻译的差异性。当然，这里并不程度。

是要讨论统计机器翻译和神经机器翻译孰优孰劣，只是很多场景中发现神经机器翻译系统可以生成非常流畅的译文，易于人工阅读和修改。

在很多量化的评价中也可以看到神经机器翻译的优势。回忆一下第四章提到的机器翻译质量的自动评估指标中，使用最广泛的一种指标是 BLEU。2010 年前，在由美国国家标准和科技机构（NIST）举办的汉英机器翻译评测中（比如汉英 MT08 数据集），30% 以上的 BLEU 值对于基于统计方法的翻译系统来说就已经是当时最顶尖的结果了。而现在的神经机器翻译系统，则可以轻松地将 BLEU 提高至 45% 以上。

同样，在机器翻译领域中著名评测比赛 WMT（Workshop of Machine Translation）中，使用统计机器翻译方法的参赛系统也在逐年减少。而现在获得比赛冠军的系统中几乎没有只使用纯统计机器翻译模型的系统<sup>2</sup>。图 10.3 展示了近年来 WMT 比赛冠军系统中神经机器翻译系统的占比，可见神经机器翻译系统的占比在逐年提高。

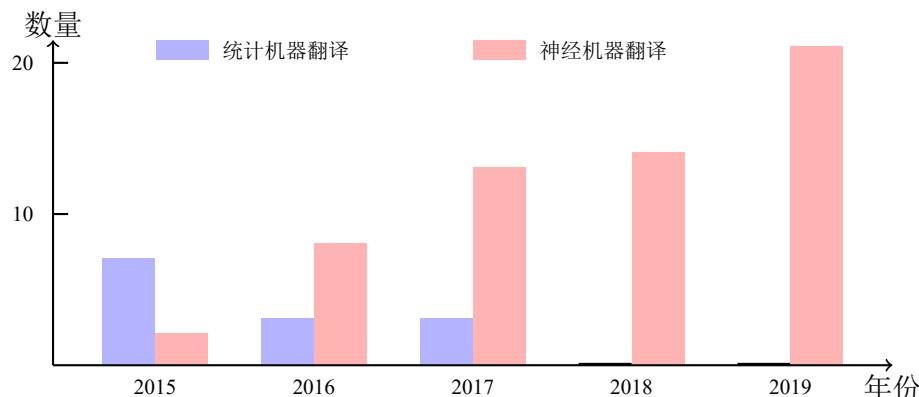


图 10.3 WMT 冠军系统的数量

神经机器翻译在其他评价指标上的表现也全面超越统计机器翻译。比如，在 IWSLT 2015 英语-德语任务中，研究人员搭建了四个较为先进的机器翻译系统<sup>[456]</sup>：

- PBSY：基于短语和串到树模型的混合系统，其中也使用了一些稀疏的词汇化特征；
- HPB：层次短语系统，其中使用了基于句法的预调序和基于神经语言模型的重排序模块；
- SPB：标准的基于短语的模型，其中使用了基于神经语言模型的重排序模块；
- NMT：神经机器翻译系统，其中使用了长短时记忆模型、注意力机制、稀有词处理机制等。

与这些系统相比，神经机器翻译系统的 mTER 得分在不同长度句子上都有明显的下降，如图 10.4<sup>3</sup>。其次，神经机器翻译的单词形态错误率和单词词义错误率（用

<sup>2</sup>但是，仍然有大量的统计机器翻译和神经机器翻译融合的方法。比如，在无指导机器翻译中，统计机器翻译仍然被作为初始模型。

<sup>3</sup>mTER、HTER 等都是错误率度量，值越低表明译文越好。

HTER 度量) 都远低于统计机器翻译系统(表10.1)。

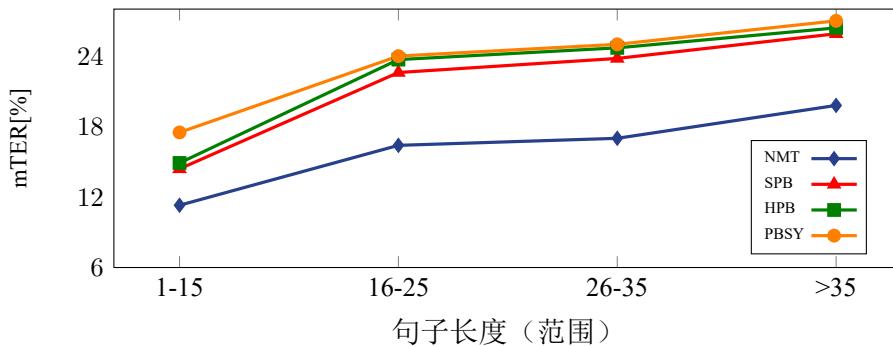


图 10.4 不同系统在不同长度句子上的 mTER[%] 分值 (得分越低越好)<sup>[456]</sup>

表 10.1 神经机器翻译与统计机器翻译系统的译文错误率 HTER[%] (忽略编辑距离中的移动操作)  
<sup>[456]</sup>

系统	单词	词根	$\Delta$
PBSY	27.1	22.5	-16.9
HPB	28.7	23.5	-18.4
SPB	28.3	23.2	-18.0
NMT	21.7	18.7	-13.7

此外, 神经机器翻译在某些任务上的结果已经相当优秀, 比如, 在一些汉英新闻翻译任务中, 神经机器翻译就取得了至少和专业翻译人员相媲美的效果<sup>[457]</sup>。在该任务中, 神经机器系统 (Combo-4、Combo-5 和 Combo-6) 的人工评价得分与 Reference-HT (专业翻译人员翻译) 的得分无显著差别, 且远超 Reference-WMT (WMT 的参考译文, 也是由人类翻译) 得分 (表10.2)。

表 10.2 不同机器翻译系统人类评价结果<sup>[457]</sup>

#	Ave%	系统
	(平均原始分数)	
1	69.0	Combo-6
	68.5	Reference-HT
	68.9	Combo-5
	68.6	Combo-4
2	62.1	Reference-WMT

在最近两年, 神经机器翻译的发展更加迅速, 新的模型及方法层出不穷。表10.3给出了到 2020 年为止, 一些主流的神经机器翻译模型在 WMT14 英德数据集上的表现。可以看到, 相比 2017 年, 2018-2020 年中机器翻译仍然有明显的进步。

表 10.3 WMT14 英德数据集上不同神经机器翻译系统的表现

模型	作者	年份	BLEU[%]
ConvS2S <sup>[24]</sup>	Gehring 等	2017	25.2
Transformer-Base <sup>[23]</sup>	Vaswani 等	2017	27.3
Transformer-Big <sup>[23]</sup>	Vaswani 等	2017	28.4
RNMT+ <sup>[458]</sup>	Chen 等	2018	28.5
Layer-Wise Coordination <sup>[459]</sup>	He 等	2018	29.0
Transformer-RPR <sup>[460]</sup>	Shaw 等	2018	29.2
Transformer-DLCL <sup>[461]</sup>	Wang 等	2019	29.3
SDT <sup>[462]</sup>	Li 等	2020	30.4
MSC <sup>[463]</sup>	Wei 等	2020	30.5

### 10.1.3 神经机器翻译的优势

既然神经机器翻译如此强大，它的优势在哪里呢？为了回答这个问题，表10.4给出了神经机器翻译与统计机器翻译的简单对比。具体来说，神经机器翻译有如下特点：

表 10.4 统计机器翻译 vs 神经机器翻译

统计机器翻译	神经机器翻译
基于离散空间的表示模型	基于连续空间的表示模型
NLP 问题的隐含结构假设	无隐含结构假设，端到端学习
特征工程为主	无显性特征，但需要设计网络
特征、规则的存储耗资源	模型存储相对小，但计算量大

- 分布式连续空间表示模型，能捕获更多隐藏信息。神经机器翻译与统计机器翻译最大的区别在于对语言文字串的表示方法。在统计机器翻译中，所有词串本质上都是由更小的词串（短语、规则）组合而成，也就是统计机器翻译模型利用了词串之间的组合性来表示更大的词串。统计机器翻译使用多个特征描述翻译结果，但是其仍然对应着离散的字符串的组合，因此可以把模型对问题的表示空间看做是由一个离散结构组成的集合。在神经机器翻译中，词串的表示已经被神经网络转化为多维实数向量，而且也不依赖任何的可组合性假设等其他假设来刻画离散的语言结构，从这个角度说，所有的词串分别对应了一个连续空间上的点（比如，对应多维实数空间中一个点）。这样，模型可以更好地进行优化，而且对未见样本有更好的泛化能力。此外，基于连续可微函数的机器学习算法已经相对完备，可以很容易地对问题进行建模和优化。
- 不含隐含结构假设，端到端学习对问题建模更加直接。传统的自然语言处理任务会对问题进行隐含结构假设。比如，进行翻译时，统计机器翻译会假设翻译

过程由短语的拼装完成。这些假设可以大大简化问题的复杂度，但是另一方面也带来了各种各样的约束条件，并且错误的隐含假设往往会导致建模错误。神经机器翻译是一种端到端模型，它并不依赖任何隐含结构假设。这样，模型并不会受到错误的隐含结构的引导。从某种意义上说，端到端学习可以让模型更加“自由”地进行学习，因此往往可以学到很多传统认知上不容易理解或者不容易观测到的现象。

- 不需要特征工程，特征学习更加全面。经典的统计机器翻译可以通过判别式模型引入任意特征，不过这些特征需要人工设计，因此这个过程也被称为特征工程。特征工程依赖大量的人工，特别是对不同语种、不同场景的翻译任务，所采用的特征可能不尽相同，这也使得设计有效的特征成为了统计机器翻译时代最主要的工作之一。但是，由于人类自身的思维和认知水平的限制，人工设计的特征可能不全面，甚至会遗漏一些重要的翻译现象。神经机器翻译并不依赖任何人工特征的设计，或者说它的特征都隐含在分布式表示中。这些“特征”都是自动学习得到的，因此神经机器翻译并不会受到人工思维的限制，学习到的特征对问题描述更加全面。
- 模型结构统一，存储消耗更小。统计机器翻译系统依赖于很多模块，比如词对齐、短语（规则）表和目标语言模型等等，因为所有的信息（如  $n$ -gram）都是离散化表示的，因此模型需要消耗大量的存储资源。同时，由于系统模块较多，开发的难度也较大。神经机器翻译的模型都是用神经网络进行表示，模型参数大多是实数矩阵，因此存储资源的消耗很小。而且神经网络可以作为一个整体进行开发和调试，系统搭建的代价相对较低。实际上，由于模型体积小，神经机器翻译也非常适合于离线小设备上的翻译任务。

当然，神经机器翻译也并不完美，很多问题有待解决。首先，神经机器翻译需要大规模浮点运算的支持，模型的推断速度较低。为了获得优质的翻译结果，往往需要大量 GPU 设备的支持，计算资源成本很高；其次，由于缺乏人类的先验知识对翻译过程的指导，神经机器翻译的运行过程缺乏可解释性，系统的可干预性也较差；此外，虽然脱离了繁重的特征工程，神经机器翻译仍然需要人工设计网络结构，在模型的各种超参数的设置、训练策略的选择等方面，仍然需要大量的人工参与。这也导致很多实验结果不容易复现。显然，完全不依赖人工的机器翻译还很遥远。不过，随着研究人员的不断攻关，很多问题也得到了解决。

## 10.2 编码器-解码器框架

说到神经机器翻译就不得不提**编码器-解码器模型**，或**编码器-解码器框架**（Encoder-Decoder Paradigm）。本质上，编码器-解码器模型是描述输入-输出之间关系的一种方式。编码器-解码器这个概念在日常生活中并不少见。例如，在电视系统上为了便于视频的传播，会使用各种编码器将视频编码成数字信号，在客户端，相应的解码

器组件会把收到的数字信号解码为视频。另外一个更贴近生活的例子是电话，它通过对声波和电信号进行相互转换，达到传递声音的目的。这种“先编码，再解码”的思想被应用到密码学、信息论等多个领域。

不难看出，机器翻译问题也完美的贴合编码器-解码器结构的特点。可以将源语言编码为类似信息传输中的数字信号，然后利用解码器对其进行转换，生成目标语言。下面就来看一下神经机器翻译是如何在编码器-解码器框架下进行工作的。

### 10.2.1 框架结构

编码器-解码器框架是一种典型的基于“表示”的模型。编码器的作用是将输入的文字序列通过某种转换变为一种新的“表示”形式，这种“表示”包含了输入序列的所有信息。之后，解码器把这种“表示”重新转换为输出的文字序列。这其中的一个核心问题是表示学习，即：如何定义对输入文字序列的表示形式，并自动学习这种表示，同时应用它生成输出序列。一般来说，不同的表示学习方法可以对应不同的机器翻译模型，比如，在最初的神经机器翻译模型中，源语言句子都被表示为一个独立的向量，这时表示结果是静态的；而在注意力机制中，源语言句子的表示是动态的，也就是翻译目标语言的每个单词时都会使用不同的表示结果。

图10.5是一个应用编码器-解码器结构来解决机器翻译问题的简单实例。给定一个中文句子“我/对/你/感到/满意”，编码器会将这句话编码成一个实数向量 $(0.2, -1, 6, 5, 0.7, -2)$ ，这个向量就是源语言句子的“表示”结果。虽然有些不可思议，但是神经机器翻译模型把这个向量等同于输入序列。向量中的数字并没有实际的意义，然而解码器却能从中提取到源语言句子中所包含的信息。也有研究人员把向量的每一个维度看作是一个“特征”，这样源语言句子就被表示成多个“特征”的联合，而且这些特征可以被自动学习。有了这样的源语言句子的“表示”，解码器可以把这个实数向量作为输入，然后逐词生成目标语言句子“*I am satisfied with you*”。

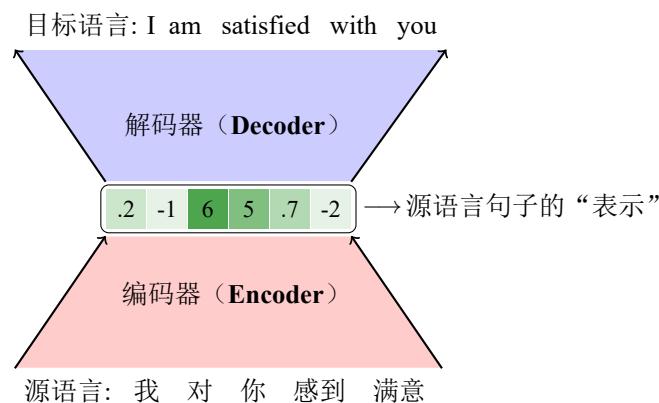


图 10.5 使用编码器-解码器架构处理汉英翻译的过程

在源语言句子的表示形式确定之后，需要设计相应的编码器和解码器结构。在

当今主流的神经机器翻译系统中，编码器由词嵌入层和中间网络层组成。当输入一串单词序列时，词嵌入层会将每个单词映射到多维实数表示空间，这个过程也被称为词嵌入。之后中间层会对词嵌入向量进行更深层的抽象，得到输入单词序列的中间表示。中间层的实现方式有很多，比如：循环神经网络、卷积神经网络、自注意力机制等都是模型常用的结构。解码器的结构基本上和编码器是一致的，在基于循环神经网络的翻译模型中，解码器只比编码器多了输出层，用于输出每个目标语言位置的单词生成概率，而在基于自注意力机制的翻译模型中，除了输出层，解码器还比编码器多一个编码-解码注意力子层，用于帮助模型更好地利用源语言信息。

现在，编码器-解码器框架已经成为了神经机器翻译系统的标准架构。当然，也有一些研究工作在探索编码器-解码器框架之外的结构<sup>[464]</sup>，但是还没有太多颠覆性的进展。因此，本章仍然以编码器-解码器框架为基础对相关模型和方法进行介绍。

### 10.2.2 表示学习

编码器-解码器框架的创新之处在于，将传统基于符号的离散型知识转化为分布式的连续型知识。比如，对于一个句子，它可以由离散的符号所构成的文法规则来生成，也可以直接被表示为一个实数向量记录句子的各个“属性”。这种分布式的实数向量可以不依赖任何离散化的符号系统，简单来说，它就是一个函数，把输入的词串转化为实数向量。更为重要的是，这种分布式表示可以被自动学习。或者从某种意义上说，编码器-解码器框架的作用之一就是学习输入序列的表示。表示结果学习的好与坏很大程度上会影响神经机器翻译系统的性能。

图10.6对比了统计机器翻译和神经机器翻译的表示模型的区别。传统的统计机器翻译（a）通过短语或者规则组合来获得更大的翻译片段，直至覆盖整个句子。这本质上是在一个离散的结构空间中不断组合的过程。神经机器翻译（b）与之不同，它并没有所谓的“组合”的过程，整个句子的处理是直接在连续空间上进行计算得到的。这二者的区别也体现了符号系统与神经网络系统的区别。前者更适合处理离散化的结构表示，后者更适合处理连续化的表示。

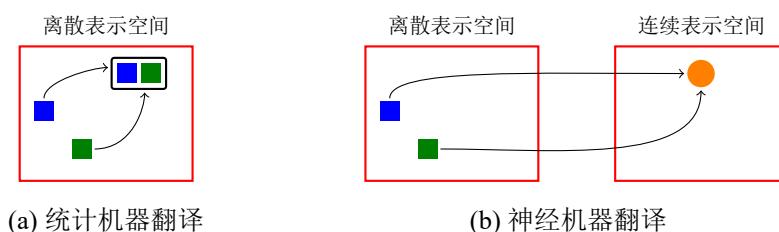


图 10.6 统计机器翻译和神经机器翻译的表示空间

实际上，编码器-解码器模型也并不是表示学习实现的唯一途径。比如，在第九章提到的神经语言模型实际上也是一种有效的学习句子表示的方法，它所衍生出的预训练模型可以从大规模单语数据上学习句子的表示形式。这种学习会比使用少量

的双语数据进行编码器和解码器的学习更加充分。相比机器翻译任务，语言模型相当于一个编码器的学习<sup>4</sup>，可以无缝嵌入到神经机器翻译模型中。不过，值得注意的是，机器翻译的目的是解决双语字符串之间的映射问题，因此它所使用的句子表示是为了更好地进行翻译。从这个角度说，机器翻译中的表示学习又和语言模型中的表示学习有不同。不过，这里不会深入讨论神经语言模型和预训练与神经机器翻译之间的异同，在后续章节会有相关讨论。

还有一点，在神经机器翻译中，句子的表示形式可以有很多选择。使用单个向量表示一个句子是一种最简单的方法。当然，也可以用矩阵、高阶张量完成表示。甚至，在解码时动态地生成源语言的表示结果。

### 10.2.3 简单的运行实例

为了对编码器-解码器框架和神经机器翻译的运行过程有一个直观的认识，这里采用标准的循环神经网络作为编码器和解码器的结构演示一个简单的翻译实例。假设系统的输入和输出为：

输入（汉语）：我 很 好 <eos>

输出（英语）：I am fine <eos>

这里令 <eos> (End of Sequence) 表示序列的终止，<sos> (Start of Sequence) 表示序列的开始。

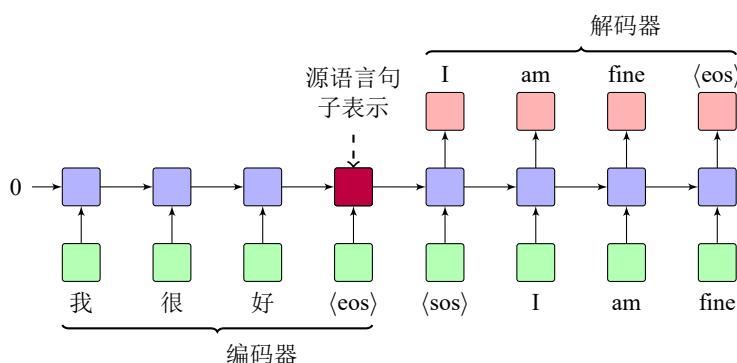


图 10.7 神经机器翻译的运行实例

翻译过程的神经网络结构如图10.7所示，其中左边是编码器，右边是解码器。编码器会顺序处理源语言单词，将每个单词都表示成一个实数向量，也就是每个单词的词嵌入结果（绿色方框）。在词嵌入的基础上运行循环神经网络（蓝色方框）。在编码下一个时间步状态的时候，上一个时间步的隐藏状态会作为历史信息传入循环神经网络。这样，句子中每个位置的信息都被向后传递，最后一个时间步的隐藏状态

<sup>4</sup>相比神经机器翻译的编码器，神经语言模型会多出一个输出层，这时可以直接把神经语言模型的中间层的输出作为编码器的输出。

(红色方框)就包含了整个源语言句子的信息，也就得到了编码器的编码结果——源语言句子的分布式表示。

解码器直接把源语言句子的分布式表示作为输入的隐层状态，之后像编码器一样依次读入目标语言单词，这是一个标准的循环神经网络的执行过程。与编码器不同的是，解码器会有一个输出层，用于根据当前时间步的隐层状态生成目标语言单词及其概率分布。可以看到，解码器当前时刻的输出单词与下一个时刻的输入单词是一样的。从这个角度说，解码器也是一种神经语言模型，只不过它会从另外一种语言(源语言)获得一些信息，而不是仅仅做单语句子的生成。具体来说，当生成第一个单词“`I`”时，解码器利用了源语言句子表示(红色方框)和目标语言的起始词“`<sos>`”。在生成第二个单词“`am`”时，解码器利用了上一个时间步的隐藏状态和已经生成的“`I`”的信息。这个过程会循环执行，直到生成完整的目标语言句子。

从这个例子可以看出，神经机器翻译的流程其实并不复杂：首先通过编码器神经网络将源语言句子编码成实数向量，然后解码器神经网络利用这个向量逐词生成译文。现在几乎所有的神经机器翻译系统都采用类似的架构。

#### 10.2.4 机器翻译范式的对比

对于不同类型的机器翻译方法，人类所扮演的作用是不同的。在统计机器翻译时代，往往需要人工定义所需要的特征和翻译单元，翻译中的每一个步骤对于人来说都是透明的，翻译过程具有一定的可解释性。而在神经机器翻译时代，神经机器翻译将所有的工作都交给神经网络，翻译的过程完全由神经网络计算得到。在整个神经网络的运行过程中并不需要人工先验知识，其中所生成的中间表示也只有神经网络自身才可以理解。有时候也会把神经机器翻译系统看作“黑盒”。所谓“黑盒”并不是指神经网络计算的过程不可见，而是这种复杂的计算过程无法控制也很难解释。那么是神经机器翻译会魔法吗，不需要任何人为的干预就可以进行翻译吗？其实不然，相对于统计机器翻译，真正变化的是人类使用知识的形式。

在机器翻译的不同时期，人类参与到机器翻译中的形式并不相同，如表10.5所述。具体来说，

- 在早期基于规则的方法中，规则的编写、维护均需要人来完成，也就是人类直接提供了计算机可读的知识形式；
- 在统计机器翻译方法中，则需要人为的设计翻译特征，并且定义基本翻译单元的形式，然后剩下的事情(比如翻译过程)交由统计机器翻译算法完成，也就是人类间接的提供了翻译所需要的知识；
- 在神经机器翻译方法中，特征的设计完全不需要人的参与，但是进行特征提取的网络结构仍然需要人为地设计，训练网络所需要的参数也需要工程师的不断调整，才能发挥神经机器翻译的强大性能。

可见，不管是基于规则的机器翻译方法，还是统计机器翻译方法，甚至最新的神经机器翻译方法，人类的作用是不可替代的。虽然神经机器翻译很强大，但是它

表 10.5 不同机器翻译范式中人类的作用

机器翻译方法	人类参与方式
基于规则的方法	设计翻译规则
传统统计方法	设计翻译特征
神经网络方法	设计网络架构

的成功仍然依赖人工设计网络结构、调参。纵然，也有一些研究工作通过结构搜索的方法自动获得神经网络结构，但是搜索的算法和模型仍然需要人工设计。道理很简单：机器翻译是人类设计的，脱离了人的工作，机器翻译是不可能成功的。

### 10.3 基于循环神经网络的翻译建模

早期神经机器翻译的进展主要来自两个方面：1) 使用循环神经网络对单词序列进行建模；2) 注意力机制的使用。表10.6列出了 2013-2015 年间有代表性的部分研究工作。从这些工作的内容上看，当时的研究重点还是如何有效地使用循环神经网络进行翻译建模以及使用注意力机制捕捉双语单词序列间的对应关系。

表 10.6 2013-2015 期间神经机器翻译方面的部分论文

时间	作者	论文
2013	Kalchbrenner 和 Blunsom	Recurrent Continuous Translation Models <sup>[451]</sup>
2014	Sutskever 等	Sequence to Sequence Learning with neural networks <sup>[21]</sup>
2014	Bahdanau 等	Neural Machine Translation by Jointly Learning to Align and Translate <sup>[22]</sup>
2014	Cho 等	On the Properties of Neural Machine Translation <sup>[465]</sup>
2015	Jean 等	On Using Very Large Target Vocabulary for Neural Machine Translation <sup>[466]</sup>
2015	Luong 等	Effective Approches to Attention-based Neural Machine Translation <sup>[25]</sup>

可以说循环神经网络和注意力机制构成了当时神经机器翻译的标准框架。例如，2016 年出现的 GNMT (Google's Neural Machine Translation) 系统就是由多层循环神经网络（长短时记忆模型）以及注意力机制搭建，且在当时展示出很出色的性能<sup>[454]</sup>。其中的很多技术也都为其它神经机器翻译系统的研发提供了很好的依据。

下面将会从基于循环神经网络的翻译模型入手，介绍神经机器翻译的基本方法。之后，会对注意力机制进行介绍，同时也会介绍其在 GNMT 系统中的应用。

### 10.3.1 建模

同大多数自然语言处理任务一样，神经机器翻译要解决的一个基本问题是描述文字序列，称为序列表示问题。例如，处理语音数据、文本数据都可以被看作是典型的序列表示问题。如果把一个序列看作一个时序上的一系列变量，不同时刻的变量之间往往是存在相关性的。也就是说，一个时序中某个时刻变量的状态会依赖其他时刻变量的状态，即上下文的语境信息。下面是一个简单的例子，假设有一个句子，但是最后的单词被擦掉了，如何猜测被擦掉的单词是什么？

中午没吃饭，又刚打了一下午篮球，我现在很饿，我想\_\_\_\_\_。

显然，根据上下文中提到的“没/吃饭”、“很/饿”，最佳的答案是“吃饭”或者“吃东西”。也就是说，对序列中某个位置的答案进行预测时需要记忆当前时刻之前的序列信息，因此，循环神经网络应运而生。实际上循环神经网络有着极为广泛的应用，例如语音识别、语言建模以及即将要介绍的神经机器翻译。

第九章已经对循环神经网络的基本知识进行过介绍，这里再回顾一下。简单来说，循环神经网络由循环单元组成。对于序列中的任意时刻，都有一个循环单元与之对应，它会融合当前时刻的输入和上一时刻循环单元的输出，生成当前时刻的输出。这样每个时刻的信息都会被传递到下一时刻，这也间接达到了记录历史信息的目的。比如，对于序列  $x = \{x_1, \dots, x_m\}$ ，循环神经网络会按顺序输出一个序列  $h = \{h_1, \dots, h_m\}$ ，其中  $h_i$  表示  $i$  时刻循环神经网络的输出（通常为一个向量）。

图10.8展示了一个循环神经网络处理序列问题的实例。当前时刻循环单元的输入由上一个时刻的输出和当前时刻的输入组成，因此也可以理解为，网络当前时刻计算得到的输出是由之前的序列共同决定的，即网络在不断地传递信息的过程中记忆了历史信息。以最后一个时刻的循环单元为例，它在对“开始”这个单词的信息进行处理时，参考了之前所有词（“<sos> 让 我们”）的信息。

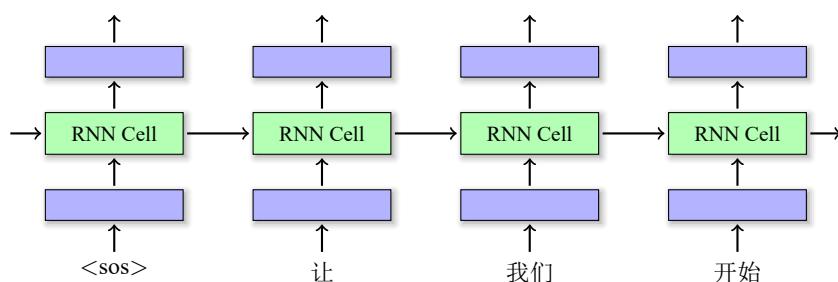


图 10.8 循环神经网络处理序列的实例

在神经机器翻译里使用循环神经网络也很简单。只需要把源语言句子和目标语言句子分别看作两个序列，之后使用两个循环神经网络分别对其进行建模。这个过程如图10.9所示。图中，下半部分是编码器，上半部分是解码器。编码器利用循环神经网络对源语言序列逐词进行编码处理，同时利用循环单元的记忆能力，不断累积

序列信息，遇到终止符  $\langle\text{eos}\rangle$  后便得到了包含源语言句子全部信息的表示结果。解码器利用编码器的输出和起始符  $\langle\text{sos}\rangle$  开始逐词地进行解码，即逐词翻译，每得到一个译文单词，便将其作为当前时刻解码器端循环单元的输入，这也是一个典型的神经语言模型的序列生成过程。解码器通过循环神经网络不断地累积已经得到的译文的信息，并继续生成下一个单词，直到遇到结束符  $\langle\text{eos}\rangle$ ，便得到了最终完整的译文。

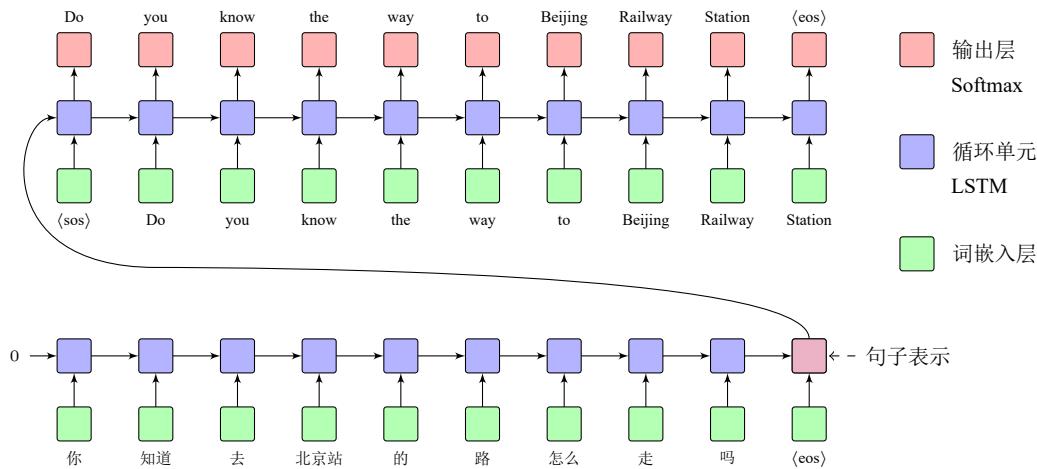


图 10.9 基于循环神经网络翻译的模型结构

从数学模型上看，神经机器翻译模型与统计机器翻译的目标是一样的：在给定源语言句子  $x$  的情况下，找出翻译概率最大的目标语言译文  $\hat{y}$ ，其计算如下式：

$$\hat{y} = \arg \max_y P(y|x) \quad (10.1)$$

这里，用  $x = \{x_1, \dots, x_m\}$  表示输入的源语言单词序列， $y = \{y_1, \dots, y_n\}$  表示生成的目标语言单词序列。由于神经机器翻译在生成译文时采用的是自左向右逐词生成的方式，并在翻译每个单词时考虑已经生成的翻译结果，因此对  $P(y|x)$  的求解可以转换为下式：

$$P(y|x) = \prod_{j=1}^n P(y_j|y_{<j}, x) \quad (10.2)$$

其中， $y_{<j}$  表示目标语言第  $j$  个位置之前已经生成的译文单词序列。 $P(y_j|y_{<j}, x)$  可以被解释为：根据源语言句子  $x$  和已生成的目标语言译文片段  $y_{<j} = \{y_1, \dots, y_{j-1}\}$ ，生成第  $j$  个目标语言单词  $y_j$  的概率。

求解  $P(y_j|y_{<j}, x)$  有三个关键问题（图10.10）：

- 如何对  $x$  和  $y_{<j}$  进行分布式表示，即词嵌入。首先，将由 One-hot 向量表示的源语言单词，即由 0 和 1 构成的离散化向量表示，转化为实数向量。可以把这

个过程记为  $e_x(\cdot)$ 。类似地，可以把目标语言序列  $y_{<j}$  中的每个单词用同样的方式进行表示，记为  $e_y(\cdot)$ 。

- 如何在词嵌入的基础上获取整个序列的表示，即句子的表示学习。可以把词嵌入的序列作为循环神经网络的输入，循环神经网络最后一个时刻的输出向量便是整个句子的表示结果。如图10.10中，编码器最后一个循环单元的输出  $\mathbf{h}_m$  被看作是一种包含了源语言句子信息的表示结果，记为  $\mathbf{C}$ 。
- 如何得到每个目标语言单词的概率，即译文单词的生成（Generation）。与神经语言模型一样，可以用一个 Softmax 输出层来获取当前时刻所有单词的分布，即利用 Softmax 函数计算目标语言词表中每个单词的概率。令目标语言序列  $j$  时刻的循环神经网络的输出向量（或状态）为  $\mathbf{s}_j$ 。根据循环神经网络的性质， $y_j$  的生成只依赖前一个状态  $\mathbf{s}_{j-1}$  和当前时刻的输入（即词嵌入  $e_y(y_{j-1})$ ）。同时考虑源语言信息  $\mathbf{C}$ ， $P(y_j|y_{<j}, x)$  可以被重新定义为：

$$P(y_j|y_{<j}, x) = P(y_j|\mathbf{s}_{j-1}, y_{j-1}, \mathbf{C}) \quad (10.3)$$

$P(y_j|\mathbf{s}_{j-1}, y_{j-1}, \mathbf{C})$  由 Softmax 实现，Softmax 的输入是循环神经网络  $j$  时刻的输出。在具体实现时， $\mathbf{C}$  可以被简单地作为第一个时刻循环单元的输入，即，当  $j=1$  时，解码器的循环神经网络会读入编码器最后一个隐层状态  $\mathbf{h}_m$ （也就是  $\mathbf{C}$ ），而其他时刻的隐层状态不直接与  $\mathbf{C}$  相关。最终， $P(y_j|y_{<j}, x)$  被表示为：

$$P(y_j|y_{<j}, x) = \begin{cases} P(y_j|\mathbf{C}, y_{j-1}) & j=1 \\ P(y_j|\mathbf{s}_{j-1}, y_{j-1}) & j>1 \end{cases} \quad (10.4)$$

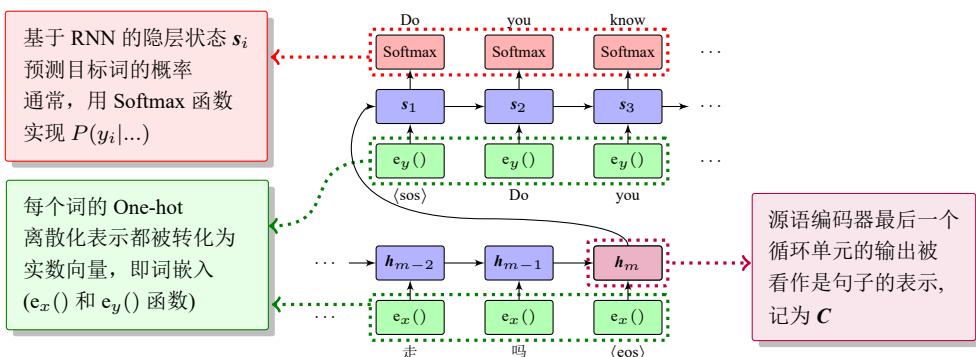


图 10.10 求解  $P(y_j|y_{<j}, x)$  的三个基本问题

输入层（词嵌入）和输出层（Softmax）的内容已在第九章进行了介绍，因此这里的核心内容是设计循环神经网络结构，即设计循环单元的结构。至今，研究人员已经提出了很多优秀的循环单元结构。其中循环神经网络（RNN）是最原始的循环单元结构。在 RNN 中，对于序列  $x = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ ，每个时刻  $t$  都对应一个循环单元，

它的输出是一个向量  $\mathbf{h}_t$ , 可以被描述为:

$$\mathbf{h}_t = f(\mathbf{x}_t \mathbf{U} + \mathbf{h}_{t-1} \mathbf{W} + \mathbf{b}) \quad (10.5)$$

其中  $\mathbf{x}_t$  是当前时刻的输入,  $\mathbf{h}_{t-1}$  是上一时刻循环单元的输出,  $f(\cdot)$  是激活函数,  $\mathbf{U}$  和  $\mathbf{W}$  是参数矩阵,  $\mathbf{b}$  是偏置。

虽然 RNN 的结构很简单, 但是已经具有了对序列信息进行记忆的能力。实际上, 基于 RNN 结构的神经语言模型已经能够取得比传统  $n$ -gram 语言模型更优异的性能。在机器翻译中, RNN 也可以做为入门或者快速原型所使用的神经网络结构。后面会进一步介绍更加先进的循环单元结构, 以及搭建循环神经网络中的常用技术。

### 10.3.2 长短时记忆网络

RNN 结构使得当前时刻循环单元的状态包含了之前时间步的状态信息。但是这种对历史信息的记忆并不是无损的, 随着序列变长, RNN 的记忆信息的损失越来越严重。在很多长序列处理任务中 (如长文本生成) 都观测到了类似现象。对于这个问题, 研究人员提出了**长短时记忆** (Long Short-term Memory) 模型, 也就是常说的 LSTM 模型<sup>[467]</sup>。

LSTM 模型是 RNN 模型的一种改进。相比 RNN 仅传递前一时刻的状态  $\mathbf{h}_{t-1}$ , LSTM 会同时传递两部分信息: 状态信息  $\mathbf{h}_{t-1}$  和记忆信息  $\mathbf{c}_{t-1}$ 。这里,  $\mathbf{c}_{t-1}$  是新引入的变量, 它也是循环单元的一部分, 用于显性地记录需要记录的历史内容,  $\mathbf{h}_{t-1}$  和  $\mathbf{c}_{t-1}$  在循环单元中会相互作用。LSTM 通过“门”单元来动态地选择遗忘多少以前的信息和记忆多少当前的信息。LSTM 中所使用的门单元结构如图10.11所示, 包括遗忘门, 输入门和输出门。图中  $\sigma$  代表 Sigmoid 函数, 它将函数输入映射为 0-1 范围内的实数, 用来充当门控信号。

LSTM 的结构主要分为三个部分:

- **遗忘。**顾名思义, 遗忘的目的是忘记一些历史, 在 LSTM 中通过遗忘门实现, 其结构如图10.11(a) 所示。 $\mathbf{x}_t$  表示时刻  $t$  的输入向量,  $\mathbf{h}_{t-1}$  是时刻  $t-1$  的循环单元的输出,  $\mathbf{x}_t$  和  $\mathbf{h}_{t-1}$  都作为  $t$  时刻循环单元的输入。 $\sigma$  将对  $\mathbf{x}_t$  和  $\mathbf{h}_{t-1}$  进行筛选, 以决定遗忘的信息, 其计算如下:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad (10.6)$$

这里,  $\mathbf{W}_f$  是权值,  $\mathbf{b}_f$  是偏置,  $[\mathbf{h}_{t-1}, \mathbf{x}_t]$  表示两个向量的拼接。该公式可以解释为, 对  $[\mathbf{h}_{t-1}, \mathbf{x}_t]$  进行变换, 并得到一个实数向量  $\mathbf{f}_t$ 。 $\mathbf{f}_t$  的每一维都可以被理解为一个“门”, 它决定可以有多少信息被留下 (或遗忘)。

- **记忆更新。**首先, 要生成当前时刻需要新增加的信息, 该部分由输入门完成, 其结构如图10.11(b) 红色线部分, 图中“ $\otimes$ ”表示进行点乘操作。输入门的计算分为两部分, 首先利用  $\sigma$  决定门控参数  $i_t$ , 如公式(10.7), 然后通过 Tanh 函数

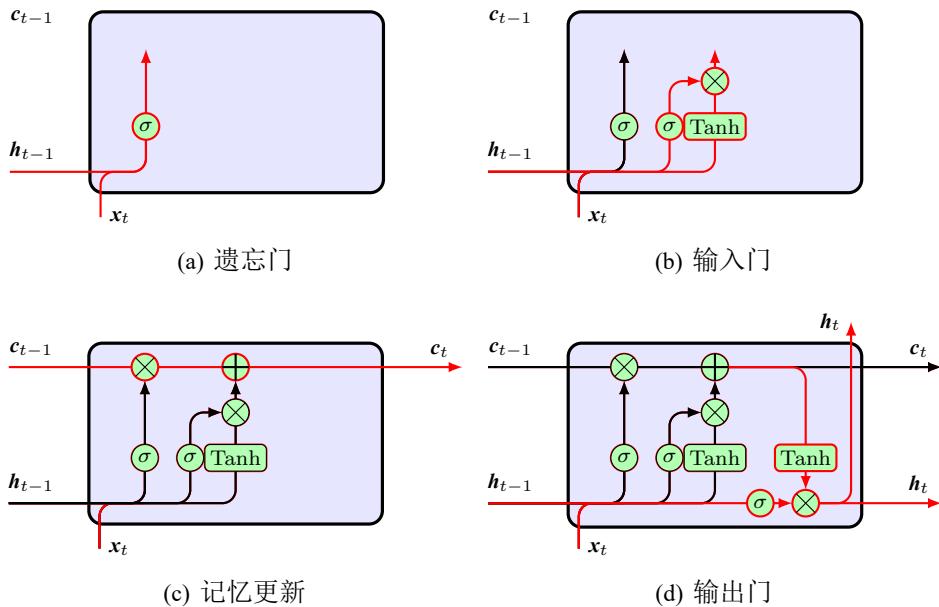


图 10.11 LSTM 中的门控结构

得到新的信息  $\hat{c}_t$ , 如公式(10.8):

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad (10.7)$$

$$\hat{c}_t = \text{Tanh}(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \quad (10.8)$$

之后, 用  $\mathbf{i}_t$  点乘  $\hat{c}_t$ , 得到当前需要记忆的信息, 记为  $\mathbf{i}_t \cdot \hat{c}_t$ 。接下来需要更新旧的信息  $\mathbf{c}_{t-1}$ , 得到新的记忆信息  $\mathbf{c}_t$ , 更新的操作如图10.11(c) 红色线部分所示, “ $\oplus$ ” 表示相加。具体规则是通过遗忘门选择忘记一部分上文信息  $\mathbf{f}_t$ , 通过输入门计算新增的信息  $\mathbf{i}_t \cdot \hat{c}_t$ , 然后根据“ $\otimes$ ”门与“ $\oplus$ ”门进行相应的乘法和加法计算, 如公式(10.9):

$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \cdot \hat{c}_t \quad (10.9)$$

- 输出。**该部分使用输出门计算最终的输出信息  $\mathbf{h}_t$ , 其结构如图10.11(d) 红色线部分所示。在输出门中, 首先将  $\mathbf{x}_t$  和  $\mathbf{h}_{t-1}$  通过  $\sigma$  函数变换得到  $\mathbf{o}_t$ , 如公式(10.10)。其次, 将上一步得到的新记忆信息  $\mathbf{c}_t$  通过  $\text{Tanh}$  函数进行变换, 得到值在  $[-1, 1]$  范围的向量。最后将这两部分进行点乘, 具体如公式(10.11):

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad (10.10)$$

$$\mathbf{h}_t = \mathbf{o}_t \cdot \text{Tanh}(\mathbf{c}_t) \quad (10.11)$$

LSTM 的完整结构如图10.12所示，模型的参数包括：参数矩阵  $\mathbf{W}_f$ 、 $\mathbf{W}_i$ 、 $\mathbf{W}_c$ 、 $\mathbf{W}_o$  和偏置  $\mathbf{b}_f$ 、 $\mathbf{b}_i$ 、 $\mathbf{b}_c$ 、 $\mathbf{b}_o$ 。可以看出， $\mathbf{h}_t$  是由  $\mathbf{c}_{t-1}$ 、 $\mathbf{h}_{t-1}$  与  $\mathbf{x}_t$  共同决定的。此外，本节公式中激活函数的选择是根据函数各自的特点决定的。

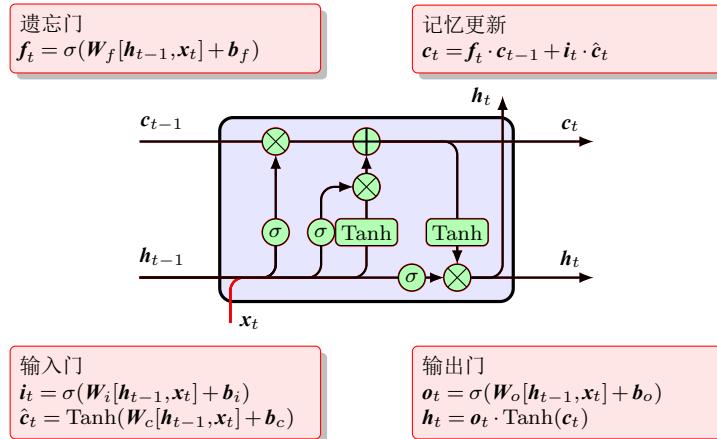


图 10.12 LSTM 的整体结构

### 10.3.3 门控循环单元

LSTM 通过门控单元控制传递状态，忘记不重要的信息，记住必要的历史信息，在长序列上取得了很好的效果，但是其进行了许多门信号的计算，较为繁琐。**门循环单元** (Gated Recurrent Unit, GRU) 作为一个 LSTM 的变种，继承了 LSTM 中利用门控单元控制信息传递的思想，并对 LSTM 进行了简化<sup>[468]</sup>。它把循环单元状态  $\mathbf{h}_t$  和记忆  $\mathbf{c}_t$  合并成一个状态  $\mathbf{h}_t$ ，同时使用了更少的门控单元，大大提升了计算效率。

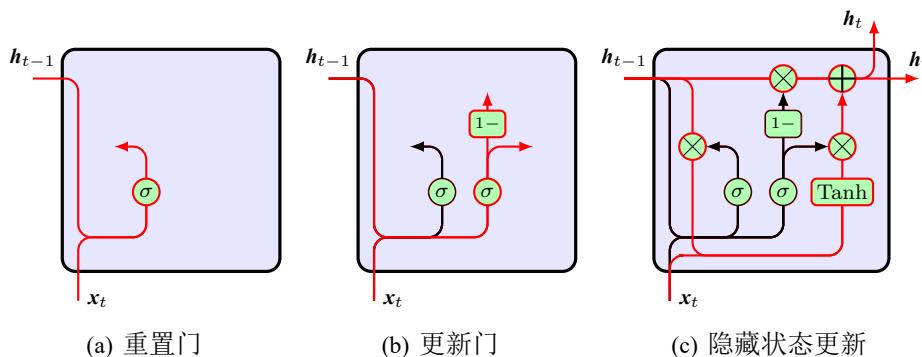


图 10.13 GRU 中的门控结构

GRU 的输入和 RNN 是一样的，由输入  $\mathbf{x}_t$  和  $t-1$  时刻的状态  $\mathbf{h}_{t-1}$  组成。GRU 只有两个门信号，分别是重置门和更新门。重置门  $r_t$  用来控制前一时刻隐藏状态的

记忆程度，其结构如图10.13(a)，其计算如公式(10.12)。更新门用来更新记忆，使用一个门同时完成遗忘和记忆两种操作，其结构如图10.13(b)，其计算如公式(10.13)。

$$\mathbf{r}_t = \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}, \mathbf{x}_t]) \quad (10.12)$$

$$\mathbf{u}_t = \sigma(\mathbf{W}_u[\mathbf{h}_{t-1}, \mathbf{x}_t]) \quad (10.13)$$

当完成了重置门和更新门计算后，就需要更新当前隐藏状态，如图10.13(c)所示。在计算得到了重置门的权重  $\mathbf{r}_t$  后，使用其对前一时刻的状态  $\mathbf{h}_{t-1}$  进行重置 ( $\mathbf{r}_t \cdot \mathbf{h}_{t-1}$ )，将重置后的结果与  $\mathbf{x}_t$  拼接，通过 Tanh 激活函数将数据变换到 [-1,1] 范围内，具体计算为：

$$\hat{\mathbf{h}}_t = \text{Tanh}(\mathbf{W}_h[\mathbf{r}_t \cdot \mathbf{h}_{t-1}, \mathbf{x}_t]) \quad (10.14)$$

$\hat{\mathbf{h}}_t$  在包含了输入信息  $\mathbf{x}_t$  的同时，引入了  $\mathbf{h}_{t-1}$  的信息，可以理解为，记忆了当前时刻的状态。下一步是计算更新后的隐藏状态也就是更新记忆，如下：

$$\mathbf{h}_t = (1 - \mathbf{u}_t) \cdot \mathbf{h}_{t-1} + \mathbf{u}_t \cdot \hat{\mathbf{h}}_t \quad (10.15)$$

这里， $\mathbf{u}_t$  是更新门中得到的权重，将  $\mathbf{u}_t$  作用于  $\hat{\mathbf{h}}_t$  表示对当前时刻的状态进行“遗忘”，舍弃一些不重要的信息，将  $(1 - \mathbf{u}_t)$  作用于  $\mathbf{h}_{t-1}$ ，用于对上一时刻隐藏状态进行选择性记忆。

GRU 的输入输出和 RNN 类似，其采用与 LSTM 类似的门控思想，达到捕获长距离依赖信息的目的。此外，GRU 比 LSTM 少了一个门结构，而且参数只有  $\mathbf{W}_r$ 、 $\mathbf{W}_u$  和  $\mathbf{W}_h$ 。因此，GRU 具有比 LSTM 高的运算效率，在系统研发中也经常被使用。

#### 10.3.4 双向模型

前面提到的循环神经网络都是自左向右运行的，也就是说在处理一个单词的时候只能访问它前面的序列信息。但是，只根据句子的前文来生成一个序列的表示是不全面的，因为从最后一个词来看，第一个词的信息可能已经很微弱了。为了同时考虑前文和后文的信息，一种解决办法是使用双向循环网络，其结构如图10.14所示。这里，编码器可以看作由两个循环神经网络构成，第一个网络，即红色虚线框里的网络，从句子的右边进行处理，第二个网络从句子左边开始处理，最终将正向和反向得到的结果都融合后传递给解码器。

双向模型是自然语言处理领域的常用模型，包括前几章提到的词对齐对称化、语言模型等中都大量地使用了类似的思路。实际上，这里也体现了建模时的非对称思想。也就是，建模时如果设计一个对称模型可能会导致问题复杂度增加，因此往往先对问题进行化简，从某一个角度解决问题。之后再融合多个模型，从不同角度得到相对合理的最终方案。

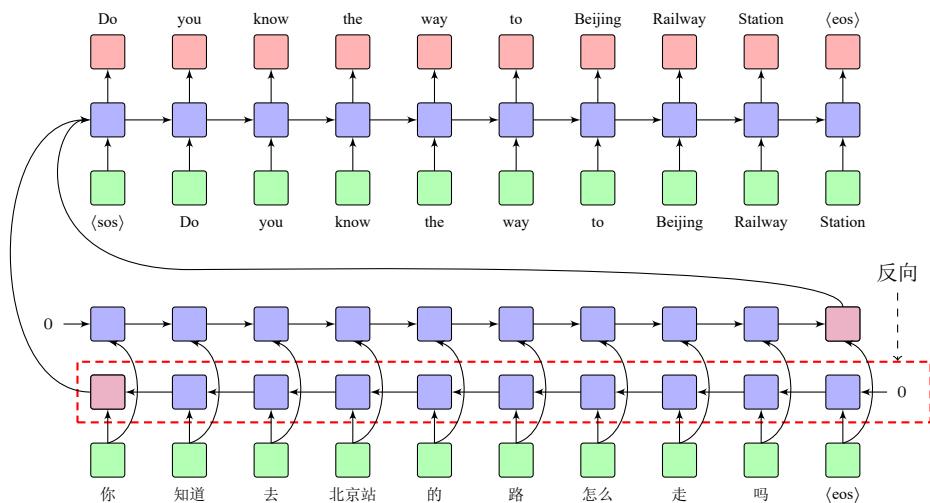


图 10.14 基于双向循环神经网络的机器翻译模型结构

### 10.3.5 多层神经网络

实际上，对于单词序列所使用的循环神经网络是一种很“深”的网络，因为从第一个单词到最后一个单词需要经过至少句子长度相当层数的神经元。比如，一个包含几十个词的句子也会对应几十个神经元层。但是，在很多深度学习应用中，更习惯把对输入序列的同一种处理作为“一层”。比如，对于输入序列，构建一个 RNN，那么这些循环单元就构成了网络的“一层”。当然，这里并不是要混淆概念。只是要明确，在随后的讨论中，“层”并不是指一组神经元的全连接，它一般指的是网络结构中逻辑上的一层。

单层循环神经网络对输入序列进行了抽象，为了得到更深入的抽象能力，可以把多个循环神经网络叠在一起，构成多层循环神经网络。比如，图10.15就展示了基于两层循环神经网络的解码器和编码器结构。通常来说，层数越多模型的表示能力越强，因此在很多基于循环神经网络的机器翻译系统中一般会使用 4~8 层的网络。但是，过多的层也会增加模型训练的难度，甚至导致模型无法进行训练。第十三章还会对这个问题进行深入讨论。

## 10.4 注意力机制

前面提到的 GNMT 系统就使用了注意力机制，那么注意力机制究竟是什么？回顾一下第二章提到的一个观点：世界上不同事物之间的相关性是不一样的，有些事物之间的联系会很强，而其他的联系可能很弱。自然语言也完美地契合了这个观点。比如，再重新看一下前面提到的根据上下文补全缺失单词的例子，

中午没吃饭，又刚打了一下午篮球，我现在很饿，我想\_\_\_\_\_。

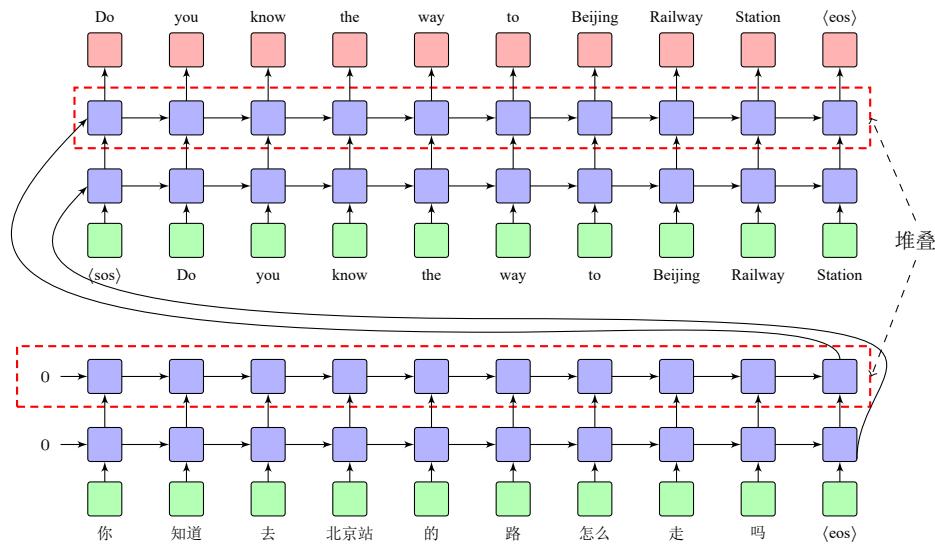


图 10.15 基于双层循环神经网络的机器翻译模型结构

之所以能想到在横线处填“吃饭”、“吃东西”很有可能是因为看到了“没/吃饭”、“很/饿”等关键信息。也就是这些关键的片段对预测缺失的单词起着关键性作用。而预测“吃饭”与前文中的“中午”、“又”之间的联系似乎不那么紧密。也就是说，在形成“吃饭”的逻辑时，在潜意识里会更注意“没/吃饭”、“很/饿”等关键信息。也就是我们的关注度并不是均匀地分布在整個句子上的。

这个现象可以用注意力机制进行解释。注意力机制的概念来源于生物学的一些现象：当待接收的信息过多时，人类会选择性地关注部分信息而忽略其他信息。它在人类的视觉、听觉、嗅觉等方面均有体现，当我们在感受事物时，大脑会自动过滤或衰减部分信息，仅关注其中少数几个部分。例如，当看到图10.16时，往往不是“均匀地”看图像中的所有区域，可能最先注意到的是小狗的嘴，然后才会关注图片中其他的部分。那注意力机制是如何解决神经机器翻译的问题呢？下面就一起来看一看。

#### 10.4.1 翻译中的注意力机制

早期的神经机器翻译只使用循环神经网络最后一个单元的输出作为整个序列的表示，这种方式有两个明显的缺陷：

- 首先，虽然编码器把一个源语言句子的表示传递给解码器，但是一个维度固定的向量所能包含的信息是有限的，随着源语言序列的增长，将整个句子的信息编码到一个固定维度的向量中可能会造成源语言句子信息的丢失。显然，在翻译较长的句子时，解码器可能无法获取完整的源语言信息，降低翻译性能；
- 此外，当生成某一个目标语言单词时，并不是均匀地使用源语言句子中的单词



图 10.16 戴帽子的狗

信息。更普遍的情况是，系统会参考与这个目标语言单词相对应的源语言单词进行翻译。这有些类似于词对齐的作用，即翻译是基于单词之间的某种对应关系。但是，使用单一的源语言表示根本无法区分源语言句子的不同部分，更不用说对源语言单词和目标语言单词之间的联系进行建模了。

更直观的，如图10.17，目标语言中的“very long”仅依赖于源语言中的“很长”。这时如果将所有源语言编码成一个固定的实数向量，“很长”的信息就很可能被其他词的信息淹没掉。

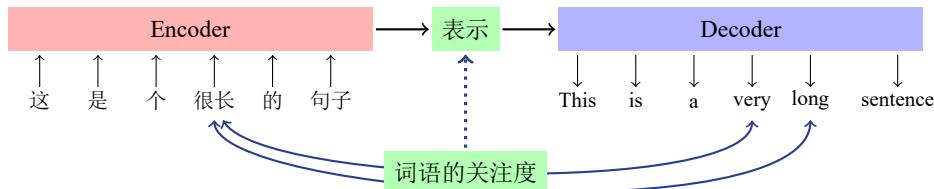


图 10.17 源语言词和目标语言词的关注度

显然，以上问题的根本原因在于所使用的表示模型还比较“弱”。因此需要一个更强大的表示模型，在生成目标语言单词时能够有选择地获取源语言句子中更有用的部分。更准确的说，对于要生成的目标语言单词，相关性更高的源语言片段应该在源语言句子的表示中体现出来，而不是将所有的源语言单词一视同仁。在神经机器翻译中引入注意力机制正是为了达到这个目的<sup>[22, 25]</sup>。实际上，除了机器翻译，注意力机制也被成功地应用于图像处理、语音识别、自然语言处理等其他任务。也正是注意力机制的引入，使得包括机器翻译在内很多自然语言处理系统得到了飞跃发展。

神经机器翻译中的注意力机制并不复杂。对于每个目标语言单词  $y_j$ ，系统生成一个源语言表示向量  $C_j$  与之对应， $C_j$  会包含生成  $y_j$  所需的源语言的信息，或者说  $C_j$  是一种包含目标语言单词与源语言单词对应关系的源语言表示。相比用一个静态的表示  $C$ ，注意机制使用的是动态的表示  $C_j$ 。 $C_j$  也被称作对于目标语言位置  $j$  的上文向量（Context Vector）。图10.18对比了未引入注意力机制和引入了注意力机制的

编码器-解码器结构。可以看出，在注意力模型中，对于每一个目标语言单词的生成，都会额外引入一个单独的上下文向量参与运算。

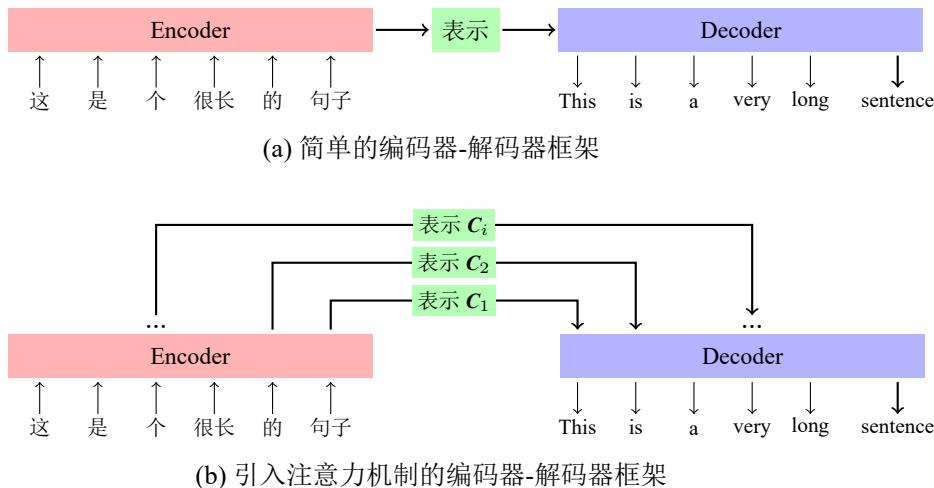


图 10.18 不使用 (a) 和使用 (b) 注意力机制的翻译模型对比

## 10.4.2 上下文向量的计算

神经机器翻译中，注意力机制的核心是：针对不同目标语言单词生成不同的上下文向量。这里，可以将注意力机制看做是一种对接收到的信息的加权处理。对于更重要的信息赋予更高的权重即更高的关注度，对于贡献度较低的信息分配较低的权重，弱化其对结果的影响。这样， $\mathbf{C}_j$  可以包含更多对当前目标语言位置有贡献的源语言片段的信息。

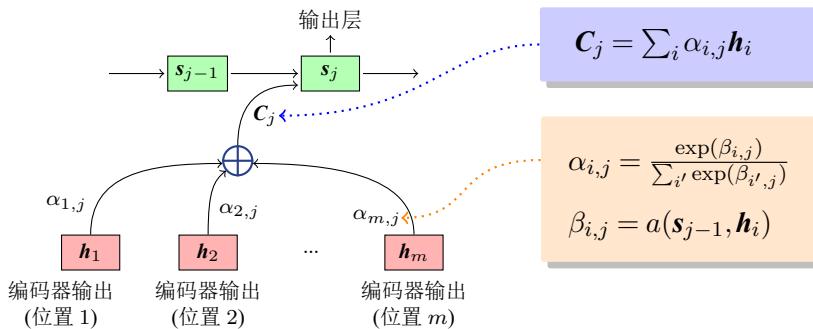
根据这种思想，上下文向量  $\mathbf{C}_j$  被定义为对不同时间步编码器输出的状态序列  $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$  进行加权求和，如下式：

$$\mathbf{C}_j = \sum_i \alpha_{i,j} \mathbf{h}_i \quad (10.16)$$

其中， $\alpha_{i,j}$  是**注意力权重**（Attention Weight），它表示目标语言第  $j$  个位置与源语言第  $i$  个位置之间的相关性大小。这里，将每个时间步编码器的输出  $\mathbf{h}_i$  看作源语言位置  $i$  的表示结果。进行翻译时，解码器可以根据当前的位置  $j$ ，通过控制不同  $\mathbf{h}_i$  的权重得到  $\mathbf{C}_j$ ，使得对目标语言位置  $j$  贡献大的  $\mathbf{h}_i$  对  $\mathbf{C}_j$  的影响增大。也就是说， $\mathbf{C}_j$  实际上就是  $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$  的一种组合，只不过不同的  $\mathbf{h}_i$  会根据对目标端的贡献给予不同的权重。图10.19展示了上下文向量  $\mathbf{C}_j$  的计算过程。

如图10.19所示，注意力权重  $\alpha_{i,j}$  的计算分为两步：

- 使用目标语言上一时刻循环单元的输出  $s_{j-1}$  与源语言第  $i$  个位置的表示  $\mathbf{h}_i$  之间的相关性，其用来表示目标语言位置  $j$  对源语言位置  $i$  的关注度，记为  $\beta_{i,j}$ ，

图 10.19 上下文向量  $\mathbf{C}_j$  的计算过程

由函数  $a(\cdot)$  实现，其具体计算如下：

$$\beta_{i,j} = a(\mathbf{s}_{j-1}, \mathbf{h}_i) \quad (10.17)$$

$a(\cdot)$  可以被看作是目标语言表示和源语言表示的一种“统一化”，即把源语言和目标语言表示映射在同一个语义空间，进而语义相近的内容有更大的相似性。该函数有多种计算方式，比如，向量乘、向量夹角和单层神经网络等，具体数学表达如公式(10.18)：

$$a(\mathbf{s}, \mathbf{h}) = \begin{cases} \mathbf{s}\mathbf{h}^T & \text{向量乘} \\ \cos(\mathbf{s}, \mathbf{h}) & \text{向量夹角} \\ \mathbf{sW}\mathbf{h}^T & \text{线性模型} \\ \text{Tanh}(\mathbf{W}[\mathbf{s}, \mathbf{h}])\mathbf{v}^T & \text{拼接}[\mathbf{s}, \mathbf{h}] + \text{单层网络} \end{cases} \quad (10.18)$$

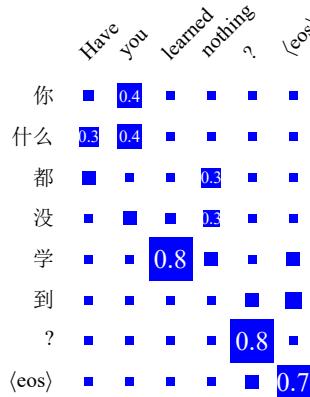
其中  $\mathbf{W}$  和  $\mathbf{v}$  是可学习的参数。

- 进一步，利用 Softmax 函数，将相关性系数  $\beta_{i,j}$  进行指数归一化处理，得到注意力权重  $\alpha_{i,j}$ ，具体计算如下：

$$\alpha_{i,j} = \frac{\exp(\beta_{i,j})}{\sum_{i'} \exp(\beta_{i',j})} \quad (10.19)$$

最终， $\{\alpha_{i,j}\}$  可以被看作是一个矩阵，它的长为目标语言句子长度，宽为源语言句子长度，矩阵中的每一项对应一个  $\alpha_{i,j}$ 。图10.20给出了 $\{\alpha_{i,j}\}$ 的一个矩阵表示。图中蓝色方框的大小表示不同的注意力权重  $\alpha_{i,j}$  的大小，方框越大，源语言位置  $i$  和目标语言位置  $j$  的相关性越高。能够看到，对于互译的中英文句子， $\{\alpha_{i,j}\}$  可以较好的反应两种语言之间不同位置的对应关系。

图10.21展示了一个上下文向量的计算过程实例。首先，计算目标语言第一个单

图 10.20 一个汉英句对之间的注意力权重  $\alpha_{i,j}$  的矩阵表示

词“Have”与源语言中的所有单词的相关性，即注意力权重，对应图中第一列  $\alpha_{i,1}$ ，则当前时刻所使用的上下文向量  $\mathbf{C}_1 = \sum_{i=1}^8 \alpha_{i,1} \mathbf{h}_i$ ；然后，计算第二个单词“you”的注意力权重对应第二列  $\alpha_{i,2}$ ，其上下文向量  $\mathbf{C}_2 = \sum_{i=1}^8 \alpha_{i,2} \mathbf{h}_i$ ，以此类推，可以得到任意目标语言位置  $j$  的上下文向量  $\mathbf{C}_j$ 。很容易看出，不同目标语言单词的上下文向量对应的源语言词的权重  $\alpha_{i,j}$  是不同的，不同的注意力权重为不同位置赋予了不同的重要性。

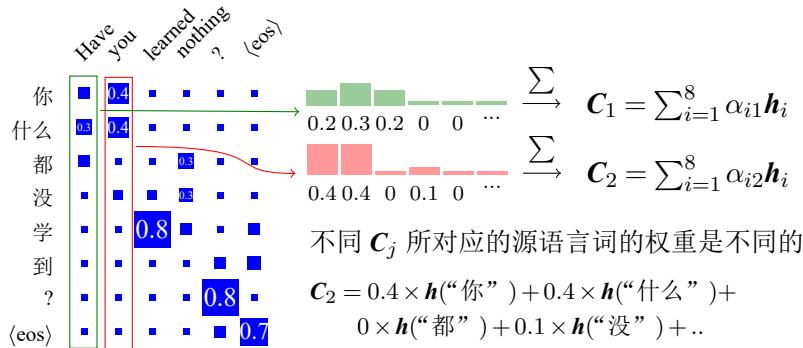


图 10.21 上下文向量计算过程实例

在 10.3.1 节中，公式(10.4)描述了目标语言单词生成概率  $P(y_j | \mathbf{y}_{<j}, \mathbf{x})$ 。在引入注意力机制后，不同时刻的上下文向量  $\mathbf{C}_j$  替换了传统模型中固定的句子表示  $\mathbf{C}$ 。描述如下：

$$P(y_j | \mathbf{y}_{<j}, \mathbf{x}) = P(y_j | \mathbf{s}_{j-1}, y_{j-1}, \mathbf{C}_j) \quad (10.20)$$

这样，可以在生成每个  $y_j$  时动态的使用不同的源语言表示  $\mathbf{C}_j$ ，并更准确地捕捉源语言和目标语言不同位置之间的相关性。表 10.7 展示了引入注意力机制前后译文单词生成公式的对比。

表 10.7 引入注意力机制前后译文单词生成公式

引入注意力之前	引入注意力之后
$\text{have} = \arg \max_{y_1} P(y_1   \mathbf{C}, y_0)$	$\text{have} = \arg \max_{y_1} P(y_1   \mathbf{C}_1, y_0)$
$\text{you} = \arg \max_{y_2} P(y_2   \mathbf{s}_1, y_1)$	$\text{you} = \arg \max_{y_2} P(y_2   \mathbf{s}_1, \mathbf{C}_2, y_1)$

### 10.4.3 注意力机制的解读

从前面的描述可以看出，注意力机制在机器翻译中就是要回答一个问题：给定一个目标语言位置  $j$  和一系列源语言的不同位置上的表示  $\{\mathbf{h}_i\}$ ，如何得到一个新的表示  $\hat{\mathbf{h}}$ ，使得它与目标语言位置  $j$  对应得最好？

那么，如何理解这个过程？注意力机制的本质又是什么呢？换一个角度来看，实际上，目标语言位置  $j$  可以被看作是一个查询，我们希望从源语言端找到与之最匹配的源语言位置，并返回相应的表示结果。为了描述这个问题，可以建立一个查询系统。假设有一个库，里面包含若干个 key-value 单元，其中 key 代表这个单元的索引关键字，value 代表这个单元的值。比如，对于学生信息系统，key 可以是学号，value 可以是学生的身高。当输入一个查询 query，我们希望这个系统返回与之最匹配的结果。也就是，希望找到匹配的 key，并输出其对应的 value。比如，当查询某个学生的身高信息时，可以输入学生的学号，之后在库中查询与这个学号相匹配的记录，并把这个记录中的 value（即身高）作为结果返回。

图10.22展示了一个这样的查询系统。里面包含四个 key-value 单元，当输入查询 query，就把 query 与这四个 key 逐个进行匹配，如果完全匹配就返回相应的 value。在图中的例子中，query 和 key<sub>3</sub> 是完全匹配的（因为都是横纹），因此系统返回第三个单元的值，即 value<sub>3</sub>。当然，如果库中没有与 query 匹配的 key，则返回一个空结果。

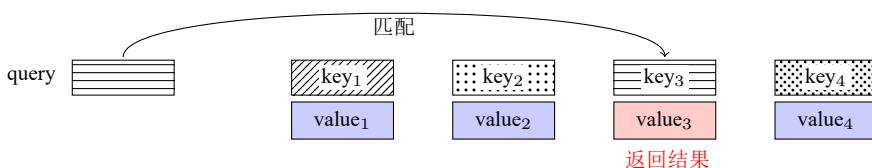


图 10.22 传统查询模型

也可以用这个系统描述翻译中的注意力问题，其中，query 即目标语言位置  $j$  的某种表示，key 和 value 即源语言每个位置  $i$  上的  $\mathbf{h}_i$ （这里 key 和 value 是相同的）。但是，这样的系统在机器翻译问题上并不好用，因为目标语言的表示和源语言的表示都在多维实数空间上，所以无法要求两个实数向量像字符串一样进行严格匹配，或者说这种严格匹配的模型可能会导致 query 几乎不会命中任何的 key。既然无法严格精确匹配，注意力机制就采用了一个“模糊”匹配的方法。这里定义每个 key<sub>i</sub> 和

query 都有一个 0~1 之间的匹配度，这个匹配度描述了  $\text{key}_i$  和 query 之间的相关程度，记为  $\alpha_i$ 。而查询的结果（记为  $\overline{\text{value}}$ ）也不再是某一个单元的 value，而是所有单元 value 用  $\alpha_i$  的加权和，具体计算如下：

$$\overline{\text{value}} = \sum_i \alpha_i \cdot \text{value}_i \quad (10.21)$$

也就是说所有的  $\text{value}_i$  都会对查询结果有贡献，只是贡献度不同罢了。可以通过设计  $\alpha_i$  来捕捉 key 和 query 之间的相关性，以达到相关度越大的 key 所对应的 value 对结果的贡献越大。

重新回到神经机器翻译问题上来。这种基于模糊匹配的查询模型可以很好的满足对注意力建模的要求。实际上，公式(10.21)中的  $\alpha_i$  就是前面提到的注意力权重，它可以由注意力函数  $a(\cdot)$  计算得到。这样， $\overline{\text{value}}$  就是得到的上下文向量，它包含了所有  $\{\mathbf{h}_i\}$  的信息，只是不同  $\mathbf{h}_i$  的贡献度不同罢了。图10.23展示了将基于模糊匹配的查询模型应用于注意力机制的实例。

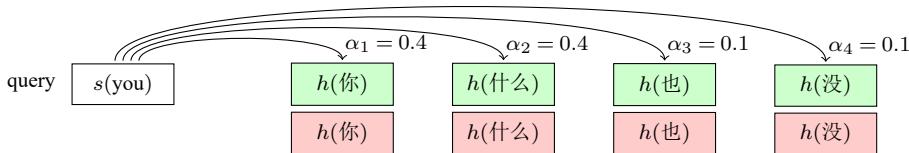


图 10.23 注意力机制所对应的查询模型

最后，从统计学的角度，如果把  $\alpha_i$  作为每个  $\text{value}_i$  出现的概率的某种估计，即： $P(\text{value}_i) = \alpha_i$ ，于是可以把公式(10.21)重写为：

$$\overline{\text{value}} = \sum_i P(\text{value}_i) \cdot \text{value}_i \quad (10.22)$$

显然， $\overline{\text{value}}$  就是  $\text{value}_i$  在分布  $P(\text{value}_i)$  下的期望，即：

$$\mathbb{E}_{\sim P(\text{value}_i)}(\text{value}_i) = \sum_i P(\text{value}_i) \cdot \text{value}_i \quad (10.23)$$

从这个观点看，注意力机制实际上是得到了变量  $\text{value}$  的期望。当然，严格意义上说， $\alpha_i$  并不是从概率角度定义的，在实际应用中也并不必须追求严格的统计学意义。

#### 10.4.4 实例 - GNMT

循环神经网络在机器翻译中有很多成功的应用，比如：RNNSearch<sup>[22]</sup>、Nematus<sup>[469]</sup>等系统就被很多研究人员作为实验系统。在众多基于循环神经网络的系统中，GNMT

系统是非常成功的一个<sup>[454]</sup>。GNMT 是谷歌 2016 年发布的神经机器翻译系统。

GNMT 使用了编码器-解码器结构，构建了一个 8 层的深度网络，每层网络均由 LSTM 组成，且在编码器-解码器之间使用了多层注意力连接。其结构如图 10.24，编码器只有最下面 2 层为双向 LSTM。GNMT 在束搜索中也加入了长度惩罚和覆盖度因子来确保输出高质量的翻译结果。

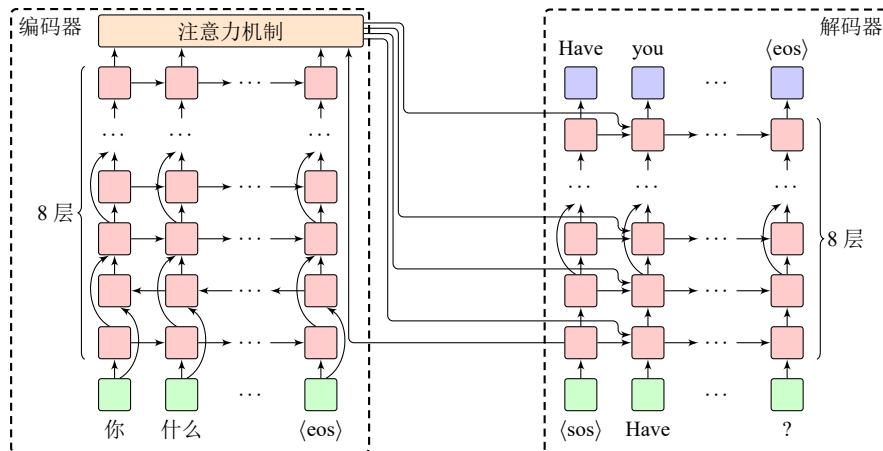


图 10.24 GNMT 结构

实际上，GNMT 的主要贡献在于集成了多种优秀的技术，而且在大规模数据上证明了神经机器翻译的有效性。在引入注意力机制之前，神经机器翻译在较大规模的任务上的性能弱于统计机器翻译。加入注意力机制和深层网络后，神经机器翻译性能有了很大的提升。在英德和英法的任务中，GNMT 的 BLEU 值不仅超过了当时优秀的神经机器翻译系统 RNNSearch 和 LSTM（6 层），还超过了当时处于领导地位的基于短语的统计机器翻译系统（PBMT）（表 10.8）。相比基于短语的统计机器翻译系统，在人工评价中，GNMT 能将翻译错误平均减少 60%。这一结果也充分表明了神经机器翻译带来的巨大性能提升。

表 10.8 GNMT 与其他翻译模型对比<sup>[454]</sup>

	BLEU[%]	
	英德	英法
	EN-DE	EN-FR
PBMT	20.7	37.0
RNNSearch	16.5	-
LSTM(6 layers)	-	31.5
Deep-Att	20.6	37.7
GNMT	24.6	39.0

## 10.5 训练及推断

神经机器翻译模型的训练大多使用基于梯度的方法（见第九章），本节将介绍这种方法训练循环神经网络的应用细节。进一步，会介绍神经机器翻译模型的推断方法。

### 10.5.1 训练

在基于梯度的方法中，模型参数可以通过损失函数  $L$  对参数的梯度进行不断更新。对于第  $\text{step}$  步参数更新，首先进行神经网络的前向计算，之后进行反向计算，并得到所有参数的梯度信息，再使用下面的规则进行参数更新：

$$\mathbf{w}_{\text{step}+1} = \mathbf{w}_{\text{step}} - \alpha \cdot \frac{\partial L(\mathbf{w}_{\text{step}})}{\partial \mathbf{w}_{\text{step}}} \quad (10.24)$$

其中， $\mathbf{w}_{\text{step}}$  表示更新前的模型参数， $\mathbf{w}_{\text{step}+1}$  表示更新后的模型参数， $L(\mathbf{w}_{\text{step}})$  表示模型相对于  $\mathbf{w}_{\text{step}}$  的损失， $\frac{\partial L(\mathbf{w}_{\text{step}})}{\partial \mathbf{w}_{\text{step}}}$  表示损失函数的梯度， $\alpha$  是更新的步长。也就是说，给定一定量的训练数据，不断执行公式(10.24)的过程。反复使用训练数据，直至模型参数达到收敛或者损失函数不再变化。通常，把公式的一次执行称为“一步”更新/训练，把访问完所有样本的训练称为“一轮”训练。

将公式(10.24)应用于神经机器翻译有几个基本问题需要考虑：1) 损失函数的选择；2) 参数初始化的策略，也就是如何设置  $\mathbf{w}_0$ ；3) 优化策略和学习率调整策略；4) 训练加速。下面对这些问题进行讨论。

#### 1. 损失函数

神经机器翻译在目标端的每个位置都会输出一个概率分布，表示这个位置上不同单词出现的可能性。设计损失函数时，需要知道当前位置输出的分布相比于标准答案的“差异”。对于这个问题，常用的是交叉熵损失函数。令  $\hat{\mathbf{y}}$  表示机器翻译模型输出的分布， $\mathbf{y}$  表示标准答案，则交叉熵损失可以被定义为：

$$L_{\text{ce}}(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{k=1}^{|V|} \hat{\mathbf{y}}[k] \log(\mathbf{y}[k]) \quad (10.25)$$

其中  $\mathbf{y}[k]$  和  $\hat{\mathbf{y}}[k]$  分别表示向量  $\mathbf{y}$  和  $\hat{\mathbf{y}}$  的第  $k$  维， $|V|$  表示输出向量的维度（等于词表大小）。假设有  $n$  个训练样本，模型输出的概率分布为  $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_n\}$ ，标准答案的分布  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ 。这个训练样本集合上的损失函数可以被定义为：

$$L(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{j=1}^n L_{\text{ce}}(\hat{\mathbf{y}}_j, \mathbf{y}_j) \quad (10.26)$$

公式(10.26)是一种非常通用的损失函数形式，除了交叉熵，也可以使用其他的

损失函数，这时只需要替换  $L_{\text{ce}}(\cdot)$  即可。这里使用交叉熵损失函数的好处在于，它非常容易优化，特别是与 Softmax 组合，其反向传播的实现非常高效。此外，交叉熵损失（在一定条件下）也对应了极大似然的思想，这种方法在自然语言处理中已经被证明是非常有效的。

除了交叉熵，很多系统也使用了面向评价的损失函数，比如，直接利用评价指标 BLEU 定义损失函数<sup>[233]</sup>。不过这类损失函数往往不可微分，因此无法直接获取梯度。这时可以引入强化学习技术，通过策略梯度等方法进行优化。不过这类方法需要采样等手段，这里不做重点讨论，相关内容会在第十三章进行介绍。

## 2. 参数初始化

神经网络的参数主要是各层中的线性变换矩阵和偏置。在训练开始时，需要对参数进行初始化。但是，由于神经机器翻译的网络结构复杂，因此损失函数往往不是凸函数，不同初始化会导致不同的优化结果。而且在大量实践中已经发现，神经机器翻译模型对初始化方式非常敏感，性能优异的系统往往需要特定的初始化方式。

因为 LSTM 是神经机器翻译中常用的一种模型，下面以 LSTM 模型为例（见 10.3.2 节），介绍机器翻译模型的初始化方法，这些方法也可以推广到 GRU 等结构。具体内容如下：

- LSTM 遗忘门偏置初始化为 1，也就是始终选择遗忘记忆  $c$ ，这样可以有效防止初始化时  $c$  里包含的错误信号传播到后面的时刻。
- 网络中的其他偏置一般都初始化为 0，可以有效防止加入过大或过小的偏置后使得激活函数的输出跑到“饱和区”，也就是梯度接近 0 的区域，防止训练一开始就无法跳出局部极小的区域。
- 网络的权重矩阵  $w$  一般使用 Xavier 参数初始化方法<sup>[470]</sup>，可以有效稳定训练过程，特别是对于比较“深”的网络。令  $d_{\text{in}}$  和  $d_{\text{out}}$  分别表示  $w$  的输入和输出的维度大小<sup>5</sup>，则该方法的具体实现如下：

$$w \sim U\left(-\sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}, \sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}\right) \quad (10.27)$$

其中  $U(a, b)$  表示以  $[a, b]$  为范围的均匀分布。

## 3. 优化策略

公式(10.24)展示了最基本的优化策略，也被称为标准的 SGD 优化器。实际上，训练神经机器翻译模型时，还有非常多的优化器可以选择，在第九章也有详细介绍，本章介绍的循环神经网络考虑使用 Adam 优化器<sup>[415]</sup>。Adam 通过对梯度的一阶矩估计（First Moment Estimation）和二阶矩估计（Second Moment Estimation）进行综合考虑，计算出更新步长。

<sup>5</sup>对于变换  $y = xw$ ， $w$  的列数为  $d_{\text{in}}$ ，行数为  $d_{\text{out}}$ 。

通常，Adam 收敛地比较快，不同任务基本上可以使用一套配置进行优化，虽性能不算差，但很难达到最优效果。相反，SGD 虽能通过在不同的数据集上进行调整，来达到最优的结果，但是收敛速度慢。因此需要根据不同的需求来选择合适的优化器。若需要快速得到模型的初步结果，选择 Adam 较为合适，若是需要在一个任务上得到最优的结果，选择 SGD 更为合适。

#### 4. 梯度裁剪

需要注意的是，训练循环神经网络时，反向传播使得网络层之间的梯度相乘。在网络层数过深时，如果连乘因子小于 1 可能造成梯度指数级的减少，甚至趋近于 0，导致网络无法优化，也就是梯度消失问题。当连乘因子大于 1 时，可能会导致梯度的乘积变得异常大，造成梯度爆炸的问题。在这种情况下需要使用“梯度裁剪”来防止梯度超过阈值。梯度裁剪在第九章已经介绍过，这里简单回顾一下。梯度裁剪的具体公式如下：

$$\mathbf{w}' = \mathbf{w} \cdot \frac{\gamma}{\max(\gamma, \|\mathbf{w}\|_2)} \quad (10.28)$$

其中， $\gamma$  是手工设定的梯度大小阈值， $\|\cdot\|_2$  是  $l_2$  范数， $\mathbf{w}'$  表示梯度裁剪后的参数。这个公式的含义在于只要梯度大小超过阈值，就按照阈值与当前梯度大小的比例进行放缩。

#### 5. 学习率策略

在公式(10.24)中， $\alpha$  决定了每次参数更新时更新的步幅大小，称之为学习率。学习率作为基于梯度方法中的重要超参数，它决定目标函数能否收敛到较好的局部最优点以及收敛的速度。合理的学习率能够使模型快速、稳定地达到较好的状态。但是，如果学习率太小，收敛过程会很慢；而学习率太大，则模型的状态可能会出现震荡，很难达到稳定，甚至使模型无法收敛。图10.25 对比了不同学习率对优化过程的影响。

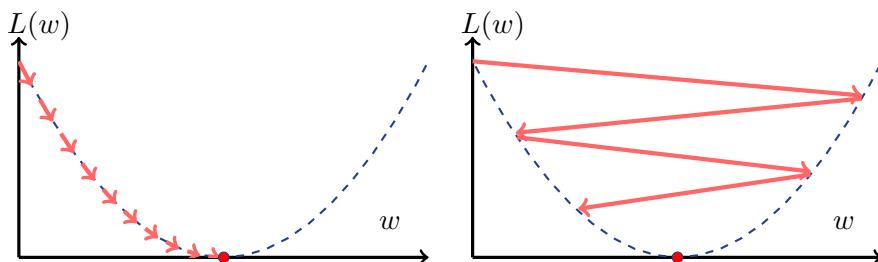


图 10.25 学习率过小（左）vs 学习率过大（右）

不同优化器需要的学习率不同，比如 Adam 一般使用 0.001 或 0.0001，而 SGD

则在 0.1~1 之间进行挑选。在梯度下降法中，都是给定的统一的学习率，整个优化过程中都以确定的步长进行更新。因此无论使用哪个优化器，为了保证训练又快又好，通常都需要根据当前的更新次数来动态调整学习率的大小。

图10.26展示了一种常用的学习率调整策略。它分为两个阶段：预热阶段和衰减阶段。模型训练初期梯度通常很大，如果直接使用较大的学习率很容易让模型陷入局部最优。学习率的预热阶段便是通过在训练初期使学习率从小到大逐渐增加来减缓在初始阶段模型“跑偏”的现象。一般来说，初始学习率太高会使得模型进入一种损失函数曲面非常不平滑的区域，进而使得模型进入一种混乱状态，后续的优化过程很难取得很好的效果。一个常用的学习率预热方法是**逐渐预热**（Gradual Warmup）。假设预热的更新次数为  $N$ ，初始学习率为  $\alpha_0$ ，则预热阶段第  $step$  次更新的学习率计算为：

$$\alpha_t = \frac{step}{N} \alpha_0, \quad 1 \leq t \leq T' \quad (10.29)$$

另一方面，当模型训练逐渐接近收敛的时候，使用太大学习率会很容易让模型在局部最优解附近震荡，从而错过局部极小，因此需要通过减小学习率来调整更新的步长，以此来不断地逼近局部最优，这一阶段也称为学习率的衰减阶段。学习率衰减的方法有很多，比如指数衰减以及余弦衰减等，图10.26右侧下降部分的曲线展示了**分段常数衰减**（Piecewise Constant Decay），即每经过  $m$  次更新，学习率衰减为原来的  $\beta_m$  ( $\beta_m < 1$ ) 倍，其中  $m$  和  $\beta_m$  为经验设置的超参。

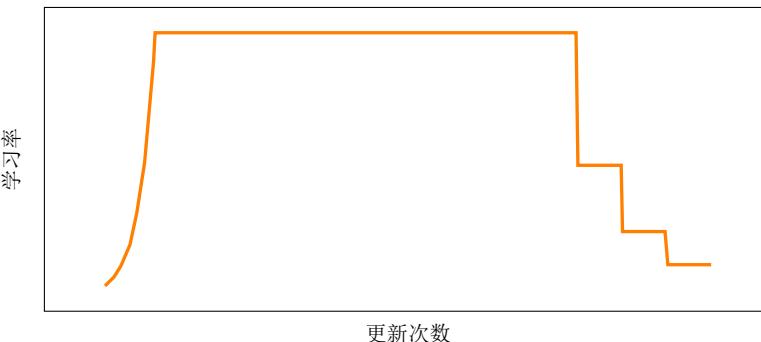


图 10.26 学习率与更新次数的变化关系

## 6. 并行训练

机器翻译是自然语言处理中很“重”的任务。因为数据量巨大而且模型较为复杂，模型训练的时间往往很长。比如，使用一千万句的训练数据，性能优异的系统往往需要几天甚至一周的时间。更大规模的数据会导致训练时间更长。特别是使用多层网络同时增加模型容量时（比如增加隐层宽度时），神经机器翻译的训练会更加缓慢。对于这个问题，一个思路是从模型训练算法上进行改进。比如前面提到的 Adam

就是一种高效的训练策略。另一种思路是利用多设备进行加速，也称作分布式训练。

常用的多设备并行化加速方法有数据并行和模型并行，其优缺点的简单对比如表10.9所示。数据并行是指把同一个批次的不同样本分到不同设备上进行并行计算。其优点是并行度高，理论上有多大的批次就可以有多少个设备并行计算，但模型体积不能大于单个设备容量的极限。而模型并行是指把“模型”切分成若干模块后分配到不同设备上并行计算。其优点是可以对很大的模型进行运算，但只能有限并行，比如，如果按层对模型进行分割，那么有多少层就需要多少个设备，同时这两种方法可以一起使用进一步提高神经网络的训练速度。具体来说：

表 10.9 数据并行与模型并行优缺点对比

	优点	缺点
数据并行	并行度高，理论上有多大的批次（Batch）就可以有多少个设备并行计算	模型不能大于单个设备的极限
模型并行	可以对很大的模型进行运算	只能有限并行，比如有多少层就有多少个设备

- **数据并行**。如果一台设备能完整放下一个神经机器翻译模型，那么数据并行可以把一个大批次均匀切成  $n$  个小批次，然后分发到  $n$  个设备上并行计算，最后把结果汇总，相当于把运算时间变为原来的  $1/n$ ，数据并行的过程如图10.27所示。不过，需要注意的是，多设备并行需要对数据在不同设备间传输，特别是多个 GPU 的情况，设备间传输的带宽十分有限，设备间传输数据往往会造成额外的时间消耗<sup>[418]</sup>。通常，数据并行的训练速度无法随着设备数量增加呈线性增长。不过这个问题也有很多优秀的解决方案，比如采用多个设备的异步训练，但是这些内容已经超出本章的内容，因此这里不做过多讨论。

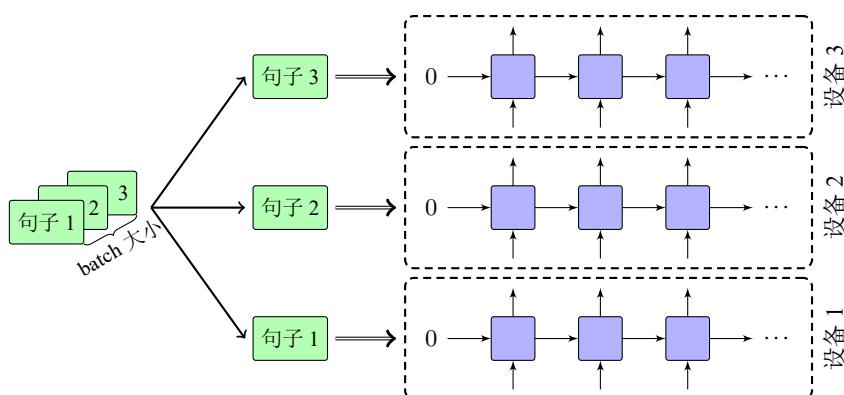


图 10.27 数据并行过程

- **模型并行**。另一种思路是，把较大的模型分成若干小模型，之后在不同设备上

训练小模型。对于循环神经网络，不同层的网络天然就是一个相对独立的模型，因此非常适合使用这种方法。比如，对于  $l$  层的循环神经网络，把每层都看做一个小模型，然后分发到  $l$  个设备上并行计算。在序列较长的时候，该方法使其运算时间变为原来的  $1/l$ 。图10.28以三层循环网络为例展示了对句子“你很不错。”进行模型并行的过程。其中，每一层网络都被放到了一个设备上。当模型根据已经生成的第一个词“你”，并预测下一个词时（图10.28(a)），同层的下一个时刻的计算和对“你”的第二层的计算就可以同时开展（图10.28(b)）。以此类推，就完成了模型的并行计算。

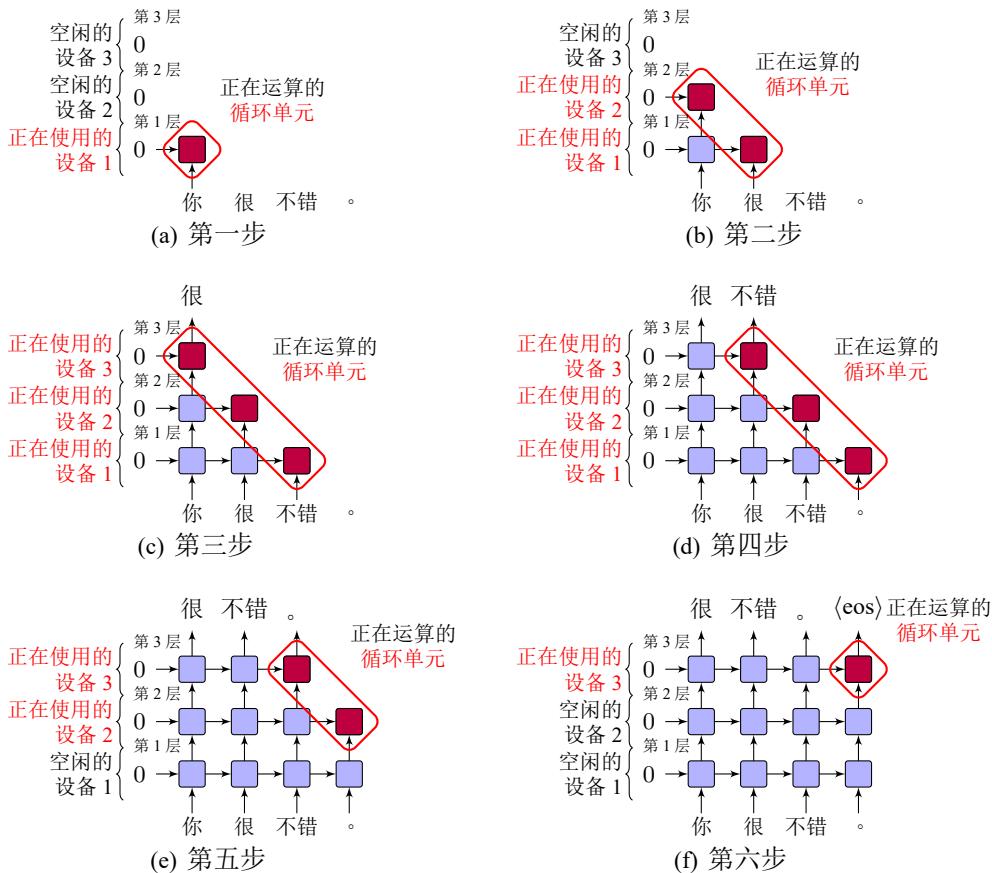


图 10.28 一个三层循环神经网络的模型并行过程

## 10.5.2 推断

神经机器翻译的推断是一个典型的搜索问题（见第二章）。这个过程是指：利用已经训练好的模型对新的源语言句子进行翻译的过程。具体来说，首先利用编码器生成源语言句子的表示，之后利用解码器预测目标语言译文。也就是，对于源语言句子  $x$ ，生成一个使翻译概率  $P(y|x)$  最大的目标语言译文  $\hat{y}$ ，具体计算如下（详细

过程见10.3.1节):

$$\begin{aligned}\hat{y} &= \arg \max_y P(y|x) \\ &= \arg \max_y \prod_{j=1}^n P(y_j|y_{<j}, x)\end{aligned}\quad (10.30)$$

在具体实现时,由于当前目标语言单词的生成需要依赖前面单词的生成,因此无法同时生成所有的目标语言单词。理论上,可以枚举所有的 $y$ ,之后利用 $P(y|x)$ 的定义对每个 $y$ 进行评价,然后找出最好的 $y$ 。这也被称作**全搜索**(Full Search)。但是,枚举所有的译文单词序列显然是不现实的。因此,在具体实现时,并不会访问所有可能的译文单词序列,而是用某种策略进行有效的搜索。常用的做法是自左向右逐词生成。比如,对于每一个目标语言位置 $j$ ,可以执行:

$$\hat{y}_j = \arg \max_{y_j} P(y_j|\hat{y}_{<j}, x) \quad (10.31)$$

其中,  $\hat{y}_j$  表示位置 $j$ 概率最高的单词,  $\hat{y}_{<j} = \{\hat{y}_1, \dots, \hat{y}_{j-1}\}$  表示已经生成的最优译文单词序列。也就是,把最优的译文看作是所有位置上最优单词的组合。显然,这是一种贪婪搜索,因为无法保证 $\{\hat{y}_1, \dots, \hat{y}_n\}$ 是全局最优解。一种缓解这个问题的方法是在每步中引入更多的候选。这里定义 $\hat{y}_{jk}$ 表示在目标语言第 $j$ 个位置排名在第 $k$ 位的单词。在每一个位置 $j$ ,可以生成 $k$ 个最可能的单词,而不是1个,这个过程可以被描述为:

$$\{\hat{y}_{j1}, \dots, \hat{y}_{jk}\} = \arg \max_{\{\hat{y}_{j1}, \dots, \hat{y}_{jk}\}} P(y_j|\{\hat{y}_{<j*}\}, x) \quad (10.32)$$

这里,  $\{\hat{y}_{j1}, \dots, \hat{y}_{jk}\}$  表示对于位置 $j$ 翻译概率最大的前 $k$ 个单词,  $\{\hat{y}_{<j*}\}$  表示前 $j-1$ 步 top- $k$  单词组成的所有历史。 $\hat{y}_{<j*}$  可以被看作是一个集合,里面每一个元素都是一个目标语言单词序列,这个序列是前面生成的一系列 top- $k$  单词的某种组成。 $P(y_j|\{\hat{y}_{<j*}\}, x)$  表示基于 $\{\hat{y}_{<j*}\}$ 的某一条路径生成 $y_j$ 的概率<sup>6</sup>。这种方法也被称为束搜索,意思是搜索时始终考虑一个集束内的候选。

不论是贪婪搜索还是束搜索都是一个自左向右的过程,也就是每个位置的处理需要等前面位置处理完才能执行。这是一种典型的**自回归模型**(Autoregressive Model),它通常用来描述时序上的随机过程,其中每一个时刻的结果对时序上其他部分的结果有依赖<sup>[471]</sup>。相对应的,也有**非自回归模型**(Non-autoregressive Model),它消除了不同时刻结果之间的直接依赖<sup>[271]</sup>。由于自回归模型是当今神经机器翻译主流的推断方法,这里仍以自回归的贪婪搜索和束搜索为基础进行讨论。

<sup>6</sup>严格来说, $P(y_j|\hat{y}_{<j*})$ 不是一个准确的数学表达,这里通过这种写法强调 $y_j$ 是由 $\{\hat{y}_{<j*}\}$ 中的某个译文单词序列作为条件生成的。

## 1. 贪婪搜索

图10.29展示了一个基于贪婪方法的神经机器翻译解码过程。每一个时间步的单词预测都依赖于其前一步单词的生成。在解码第一个单词时，由于没有之前的单词信息，会用`<sos>`进行填充作为起始的单词，且会用一个零向量（可以理解为没有之前时间步的信息）表示第0步的中间层状态。

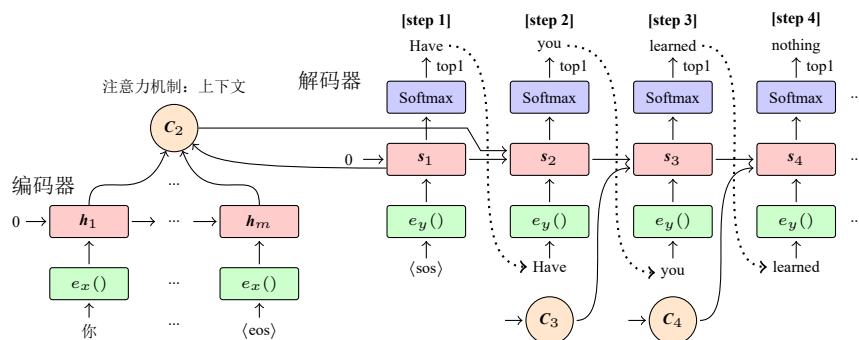


图 10.29 基于贪婪方法的解码过程

解码器的每一步 Softmax 层会输出所有单词的概率，由于是基于贪心的方法，这里会选择概率最大（top-1）的单词作为输出。这个过程可以参考图10.30的内容。选择分布中概率最大的单词“Have”作为得到的第一个单词，并再次送入解码器，作为第二步的输入同时预测下一个单词。以此类推，直到生成句子的终止符为止，就得到了完整的译文。

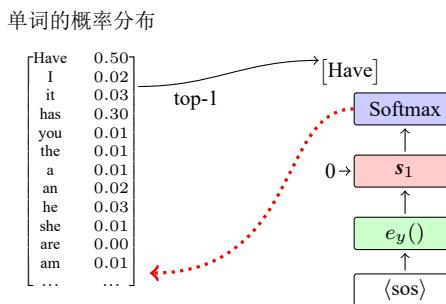


图 10.30 解码第一个位置输出的单词概率分布（“Have”的概率最高）

贪婪搜索的优点在于速度快。在对翻译速度有较高要求的场景中，贪婪搜索是一种十分有效的系统加速方法。而且贪婪搜索的原理非常简单，易于快速实现。不过，由于每一步只保留一个最好的局部结果，贪婪搜索往往会造成翻译品质上的损失。

## 2. 束搜索

束搜索是一种启发式图搜索算法。相比于全搜索，它可以减少搜索所占用的空间和时间，在每一步扩展的时候，剪掉一些质量比较差的结点，保留下一些质量较高的结点。具体到机器翻译任务，对于每一个目标语言位置，束搜索选择了概率最大的前  $k$  个单词进行扩展（其中  $k$  叫做束宽度，或简称为束宽）。如图10.31所示，假设  $\{y_1, \dots, y_n\}$  表示生成的目标语言序列，且  $k = 3$ ，则束搜索的具体过程为：在预测第一个位置时，可以通过模型得到  $y_1$  的概率分布，选取概率最大的前 3 个单词作为候选结果（假设分别为“have”，“has”，“it”）。在预测第二个位置的单词时，模型针对已经得到的三个候选结果（“have”，“has”，“it”）计算第二个单词的概率分布。因为  $y_2$  对应  $|V|$  种可能，总共可以得到  $3 \times |V|$  种结果。然后从中选取使序列概率  $P(y_2, y_1|x)$  最大的前三个  $y_2$  作为新的输出结果，这样便得到了前两个位置的 top-3 译文。在预测其他位置时也是如此，不断重复此过程直到推断结束。可以看到，束搜索的搜索空间大小与束宽度有关，也就是：束宽度越大，搜索空间越大，更有可能搜索到质量更高的译文，但同时搜索会更慢。束宽度等于 3，意味着每次只考虑三个最有可能的结果，贪婪搜索实际上便是束宽度为 1 的情况。在神经机器翻译系统实现中，一般束宽度设置在 4~8 之间。

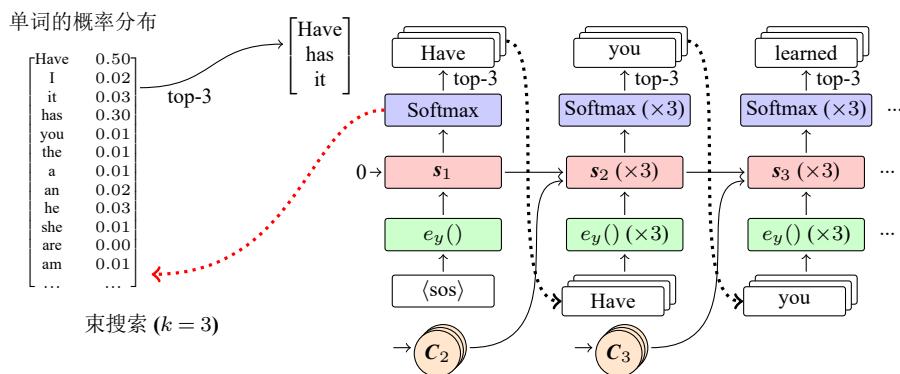


图 10.31 束搜索过程

## 3. 长度惩罚

这里用  $P(y|x) = \prod_{j=1}^n P(y_j|y_{<j}, x)$  作为翻译模型。直接实现这个公式有一个明显的缺点：当句子过长时乘法运算容易产生溢出，也就是多个数相乘可能会产生浮点数无法表示的运算结果。为了解决这个问题，可以利用对数操作将乘法转换为加法，得到新的计算方式： $\log P(y|x) = \sum_{j=1}^n \log P(y_j|y_{<j}, x)$ 。对数函数不会改变函数的单调性，因此在具体实现时，通常用  $\log P(y|x)$  表示句子的得分，而不用  $P(y|x)$ 。

不管是使用  $P(y|x)$  还是  $\log P(y|x)$  计算句子得分，还面临两个问题：

- $P(y|x)$  的范围是 [0,1]，如果句子过长，那么句子的得分就是很多个小于 1 的数

相乘，或者说取  $\log$  之后很多个小于 0 的数相加。这也就是说，句子的得分会随着长度的增加而变小，即模型倾向于生成短句子。

- 模型本身并没有考虑每个源语言单词被使用的程度，比如一个单词可能会被翻译很多“次”。这个问题在统计机器翻译中并不存在，因为所有词在翻译中必须被“覆盖”到。但是早期的神经机器翻译模型没有所谓覆盖度的概念，因此也无法保证每个单词被翻译的“程度”是合理的<sup>[472, 473]</sup>。

为了解决上面提到的问题，可以使用其他特征与  $\log P(y|x)$  一起组成新的模型得分  $\text{score}(y,x)$ 。针对模型倾向于生成短句子的问题，常用的做法是引入惩罚机制。比如，可以定义一个惩罚因子，形式如下：

$$\text{lp}(y) = \frac{(5 + |y|)^\alpha}{(5 + 1)^\alpha} \quad (10.33)$$

其中， $|y|$  代表已经得到的译文长度， $\alpha$  是一个固定的常数，用于控制惩罚的程度。同时在计算句子得分时，额外引入表示覆盖度的因子，如下：

$$\text{cp}(y,x) = \beta \cdot \sum_{i=1}^{|x|} \log \left( \min \left( \sum_j^{|y|} \alpha_{ij}, 1 \right) \right) \quad (10.34)$$

$\text{cp}(\cdot)$  会惩罚把某些源语言单词对应到很多目标语言单词的情况（覆盖度），被覆盖的程度用  $\sum_j^{|y|} \alpha_{ij}$  度量。 $\beta$  也是需要经验性设置的超参数，用于对覆盖度惩罚的程度进行控制。

最终，模型得分定义如下：

$$\text{score}(y,x) = \frac{\log P(y|x)}{\text{lp}(y)} + \text{cp}(y,x) \quad (10.35)$$

显然，当目标语言  $y$  越短时， $\text{lp}(y)$  的值越小，因为  $\log P(y|x)$  是负数，所以句子得分  $\text{score}(y,x)$  越小。也就是说，模型会惩罚译文过短的结果。当覆盖度较高时，同样会使得分变低。通过这样的惩罚机制，使模型得分更为合理，从而帮助模型选择出质量更高的译文。

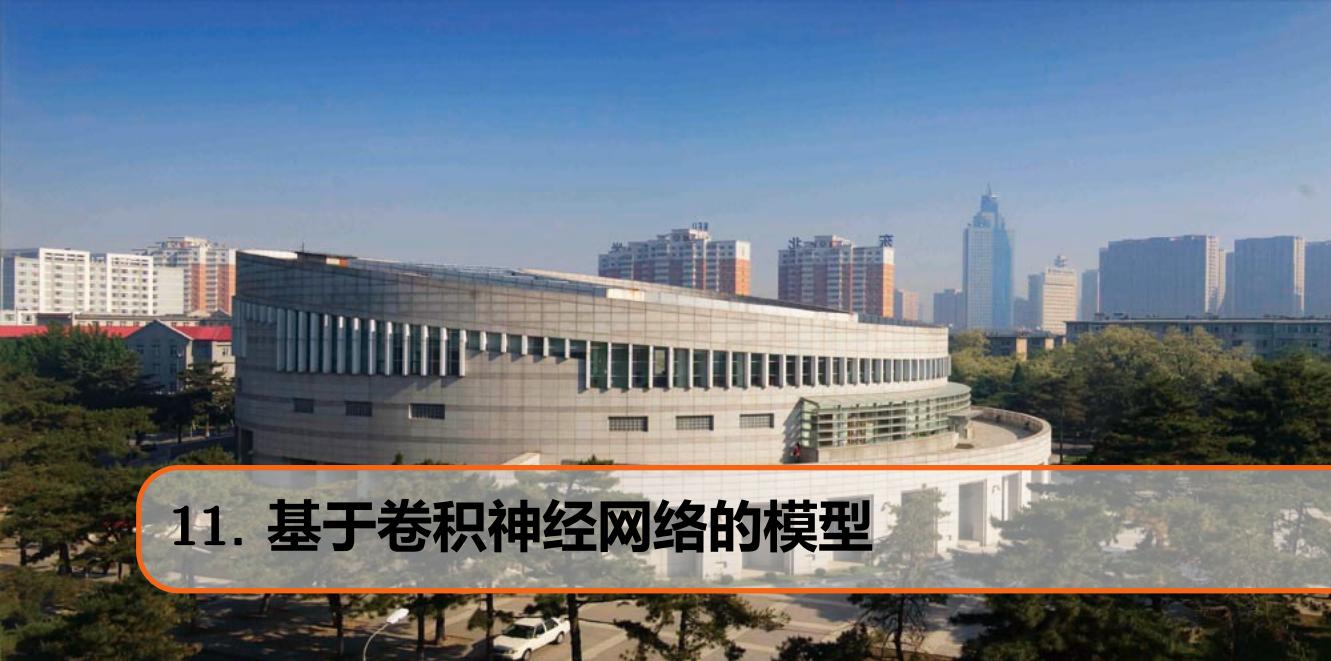
## 10.6 小结及拓展阅读

神经机器翻译是近几年的热门方向。无论是前沿性的技术探索，还是面向应用落地的系统研发，神经机器翻译已经成为当下最好的选择之一。研究人员对神经机器翻译的热情使得这个领域得到了快速的发展。本章作为神经机器翻译的入门章节，对神经机器翻译的建模思想和基础框架进行了描述。同时，对常用的神经机器翻译架构——循环神经网络进行了讨论与分析。

经过几年的积累，神经机器翻译的细分方向已经十分多样，由于篇幅所限，这里也无法覆盖所有内容（虽然笔者尽所能全面介绍相关的基础知识，但是难免会有疏漏）。很多神经机器翻译的模型和方法值得进一步学习和探讨：

- 循环神经网络有很多变种结构。比如，除了 RNN、LSTM、GRU，还有其他改进的循环单元结构，如 LRN<sup>[474]</sup>、SRU<sup>[475]</sup>、ATR<sup>[476]</sup>。
- 注意力机制的使用是机器翻译乃至整个自然语言处理近几年获得成功的重要因素之一<sup>[22, 25]</sup>。早期，有研究人员尝试将注意力机制和统计机器翻译的词对齐进行统一<sup>[477, 478, 479]</sup>。最近，也有大量的研究工作对注意力机制进行改进，比如，使用自注意力机制构建翻译模型等<sup>[23]</sup>。而对注意力模型的改进也成为了自然语言处理中的热点问题之一。在第十五章会对机器翻译中不同注意力模型进行进一步讨论。
- 一般来说，神经机器翻译的计算过程是没有人工干预的，翻译流程也无法用人类的知识直接进行解释，因此一个有趣的方向是在神经机器翻译中引入先验知识，使得机器翻译的行为更“像”人。比如，可以使用句法树来引入人类的语言学知识<sup>[431, 480]</sup>，基于句法的神经机器翻译也包含大量的树结构的神经网络建模<sup>[443, 481]</sup>。此外，也可以把用户定义的词典或者翻译记忆加入到翻译过程中<sup>[428, 482, 483, 484]</sup>，使得用户的约束可以直接反映到机器翻译的结果上来。先验知识的种类还有很多，包括词对齐<sup>[479, 485, 486]</sup>、篇章信息<sup>[487, 488, 489]</sup>等等，都是神经机器翻译中能够使用的信息。





## 11. 基于卷积神经网络的模型

卷积神经网络是一种经典的神经计算模型，在计算机视觉等领域已经得到广泛应用。通过卷积、池化等一系列操作，卷积神经网络可以很好地对输入数据进行特征提取。这个过程也与图像和语言加工中局部输入信号的处理有着天然的联系。卷积操作还可以被多次执行，形成多层卷积神经网络，进而进行更高层次的特征抽象。

在自然语言处理中，卷积神经网络也是备受关注的模型之一。本章将介绍基于卷积神经网络的机器翻译模型，不仅会重点介绍如何利用卷积神经网络构建端到端翻译模型，也会对一些机器翻译中改进的卷积神经网络结构进行讨论。

### 11.1 卷积神经网络

**卷积神经网络** (Convolutional Neural Network, CNN) 是一种前馈神经网络，由若干的卷积层与池化层组成。早期，卷积神经网络被应用在语音识别任务上<sup>[490]</sup>，之后在图像处理领域取得了很好的效果<sup>[491, 492]</sup>。近年来，卷积神经网络已经成为语音、自然语言处理、图像处理任务的基础框架<sup>[421, 493, 494, 495, 496]</sup>。在自然语言处理领域，卷积神经网络已经得到广泛应用，在文本分类<sup>[495, 496, 497]</sup>、情感分析<sup>[498]</sup>、语言建模<sup>[499, 500]</sup>、机器翻译<sup>[24, 449, 451, 501, 502]</sup>等任务中取得不错的成绩。

图11.1展示了全连接层和卷积层的结构对比，可以看到在全连接层中，模型考虑了所有的输入，层输出中的每一个元素都依赖于所有输入。这种全连接层适用于大多数任务，但是当处理图像这种网格数据的时候，规模过大的数据会导致模型参数量过大，难以处理。其次，在一些网格数据中，通常具有局部不变性的特征，比如图

像中不同位置的相同物体，语言序列中相同的  $n$ -gram 等。而全连接网络很难提取这些局部不变性特征。为此，一些研究人员提出使用卷积层来替换全连接层<sup>[503, 504]</sup>。

相比于全连接网络，卷积神经网络最大的特点在于具有**局部连接**（Locally Connected）和**权值共享**（Weight Sharing）的特性。如图11.1(b)，卷积层中每个神经元只响应周围部分的局部输入特征，大大减少了网络中的连接数和参数量。另一方面，卷积层使用相同的卷积核对不同位置进行特征提取，换句话说，就是采用权值共享来进一步减少参数量，共享的参数对应于图中相同颜色的连接。

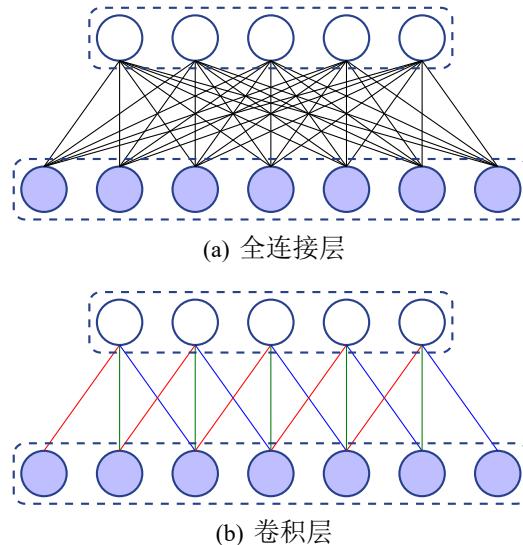


图 11.1 全连接层 (a) 与卷积层 (b) 的结构对比

图11.2展示了标准的卷积神经网络模块，其中包括了卷积层、激活函数和池化层三个部分。本节将对卷积神经网络中的基本结构进行介绍。

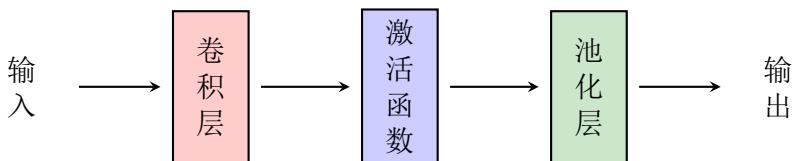


图 11.2 标准的卷积神经网络模块（卷积、激活函数、池化）

### 11.1.1 卷积核与卷积操作

卷积操作作为卷积神经网络的核心部分，其本质是一种特殊的线性运算。区别于全连接的方式，卷积使用一系列**卷积核**（Convolution Kernel，也叫滤波器）对局部输入数据进行特征提取，然后通过在输入数据空间维度上移动卷积核来获取所有位

置的特征信息。卷积的输入可以是任意维度形式的数据。由于其在图像处理领域应用最为广泛，这里以二维图像为例对卷积核和卷积操作进行简单介绍。

在图像卷积中，卷积核是一组  $Q \times U \times O$  的参数（如图11.3）。其中  $Q$  和  $U$  表示卷积核窗口的宽度与长度，分别对应图像中的宽和长两个维度， $Q \times U$  决定了该卷积核窗口的大小。 $O$  是该卷积核的深度，它的取值和输入数据通道数保持一致。在这里，通道可以看作图像不同的特征，比如灰色图像只有灰度信息，通道数为 1；而 RGB 格式的图像有 3 个通道，分别对应红绿蓝三种颜色信息。

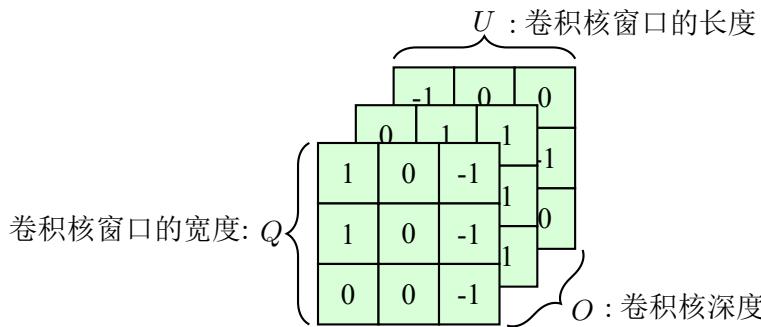


图 11.3 图像卷积中的卷积核

在卷积计算中，不同深度下卷积核不同但是执行操作相同，这里以二维卷积核为例展示具体卷积计算。若设输入矩阵为  $\mathbf{x}$ ，输出矩阵为  $\mathbf{y}$ ，卷积滑动步幅为  $\text{stride}$ ，卷积核为  $\mathbf{w}$ ，且  $\mathbf{w} \in \mathbb{R}^{Q \times U}$ ，那么卷积计算过程如下：

$$\mathbf{y}_{i,j} = \sum \sum (\mathbf{x}_{[j \times \text{stride}:j \times \text{stride}+U-1, i \times \text{stride}:i \times \text{stride}+Q-1]} \odot \mathbf{w}) \quad (11.1)$$

其中  $i$  是输出矩阵的行下标， $j$  是输出矩阵的列下标， $\odot$  表示矩阵点乘，具体见第九章。图11.4展示了一个简单的卷积操作示例，其中  $Q$  为 2， $U$  为 2， $\text{stride}$  为 1，根据公式(11.1)，图中蓝色位置  $\mathbf{y}_{0,0}$  的计算如下：

$$\begin{aligned} \mathbf{y}_{0,0} &= \sum \sum (\mathbf{x}_{[0 \times 1:0 \times 1+2-1, 0 \times 1:0 \times 1+2-1]} \odot \mathbf{w}) \\ &= \sum \sum (\mathbf{x}_{[0:1, 0:1]} \odot \mathbf{w}) \\ &= \sum \sum \begin{pmatrix} 0 \times 0 & 1 \times 1 \\ 3 \times 2 & 4 \times 3 \end{pmatrix} \\ &= 0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 \\ &= 19 \end{aligned} \quad (11.2)$$

卷积计算的作用是提取特征，用不同的卷积核计算可以获取不同的特征，比如图11.5，通过设计的特定卷积核就可以获取图像边缘信息。在卷积神经网络中，不需要手动设计卷积核，只需要指定卷积层中卷积核的数量及大小，模型就可以自己学

习卷积核具体的参数。

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 2 & 3 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 19 & 25 \\ \hline 37 & 43 \\ \hline \end{array}$$

输入:  $3 \times 3$       卷积核:  $2 \times 2$       输出:  $2 \times 2$

图 11.4 图像卷积操作 (\* 表示卷积计算)

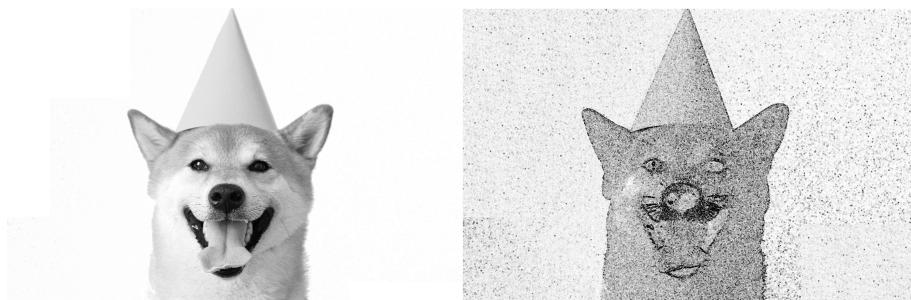


图 11.5 图像通过卷积进行边缘检测

### 11.1.2 步长与填充

在卷积操作中，步长是指卷积核每次滑动的距离，和卷积核的大小共同决定了卷积输出的大小，如图11.6所示。步长越大，对输入数据的压缩程度越高，其输出的维度越小；反之步长越小，对输入数据的压缩程度越低，同时输出的尺寸和输入越接近。比如使用一个  $3 \times 3 \times 1$  的卷积核在  $6 \times 6 \times 1$  的图像上进行卷积，如设置步长为 1，其对应的输出大小就为  $4 \times 4 \times 1$ 。这种做法最为简单，但是会导致两个问题：一是在输入数据中，由于边缘区域的像素只会被计算一次，相比于中心区域来说，这些像素被考虑的次数会更少一些，导致图像边缘信息的丢失；二是在经历多次卷积之后，其输出特征的维度会不断减小，影响模型的泛化能力。

$$\begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 0 & 2 & 1 \\ \hline 0 & 1 & 2 & 4 & 2 & 2 \\ \hline 2 & 0 & 1 & 4 & 0 & 0 \\ \hline 1 & 2 & 3 & 2 & 0 & 0 \\ \hline 2 & 0 & 0 & 1 & 5 & 2 \\ \hline 1 & 1 & 2 & 0 & 2 & 2 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 3 & 8 & 7 \\ \hline 7 & 9 & 9 & 8 \\ \hline 2 & 7 & 8 & 7 \\ \hline 8 & 7 & 2 & 3 \\ \hline \end{array}$$

输入:  $6 \times 6$       卷积核:  $3 \times 3$       输出:  $4 \times 4$

图 11.6 卷积操作的维度变换 (\* 表示卷积计算)

为了解决这两个问题，可以采用填充的操作对图像的边缘进行扩充，填充一些

元素，例如 0。比如在图11.7中，将  $6 \times 6 \times 1$  的图像填充为  $8 \times 8 \times 1$  的图像，然后在  $8 \times 8 \times 1$  的图像上进行卷积操作。这样可以得到与输入数据大小一致的输出结果，同时也缓解了图像边缘信息丢失的问题。

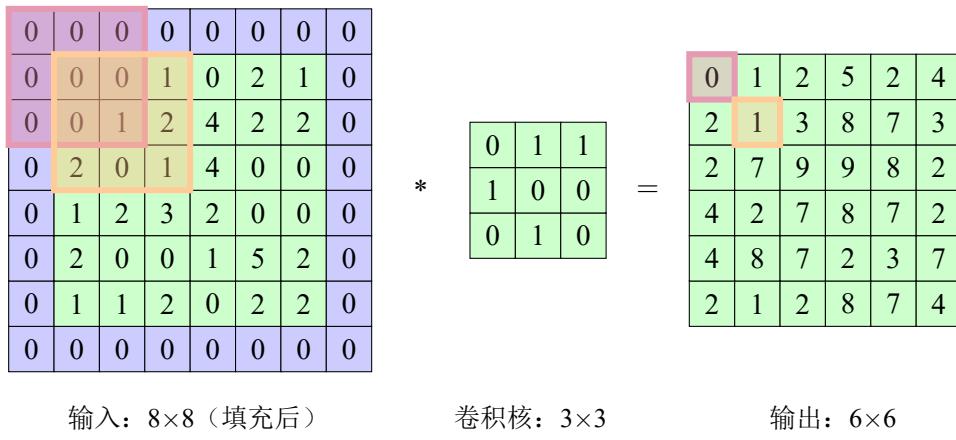


图 11.7 填充和卷积操作 (\* 表示卷积计算)

### 11.1.3 池化

在图11.2所示的网络结构中，卷积层输出会通过一个非线性的激活函数，之后会通过**池化层**（也称为汇聚层）。池化过程和卷积类似，都是根据设定的窗口进行滑动选取局部信息进行计算，不同的是，池化层的计算是无参数化的，不需要额外的权重矩阵。常见的池化操作有**最大池化**（Max Pooling）和**平均池化**（Average Pooling）。前者获取窗口内最大的值，后者则获取窗口内矩阵的平均值。图11.8展示了窗口大小为  $2 \times 2$ 、步长为 2 的两种池化方法的计算过程。

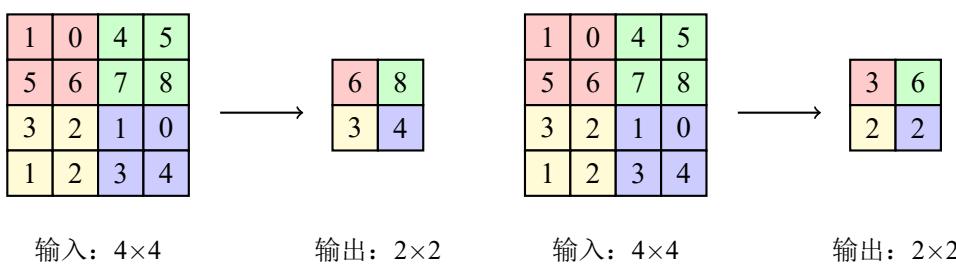


图 11.8 池化操作

池化计算选取每个滑动窗口内最突出的值或平均值作为局部信息，压缩了卷积层输出的维度大小，有效地减少了神经网络的计算量，是卷积神经网络中必不可少的操作。在网络建模时，通常在较低层时会使用最大池化，仅保留特征中最显著的部分。而当网络更深时，特征信息都具有一定意义，比如在自然语言处理任务中，深层网络的特征向量包含的语义信息较多，选取平均池化方法更适合。

### 11.1.4 面向序列的卷积操作

对比于图像处理任务中二维图像数据，自然语言处理任务中主要处理一维序列，如单词序列。由于单词序列长度往往是不固定的，很难使用全连接网络处理它，因为变长序列无法用固定大小的全连接网络进行直接建模，而且过长的序列也会导致全连接网络参数量的急剧增加。

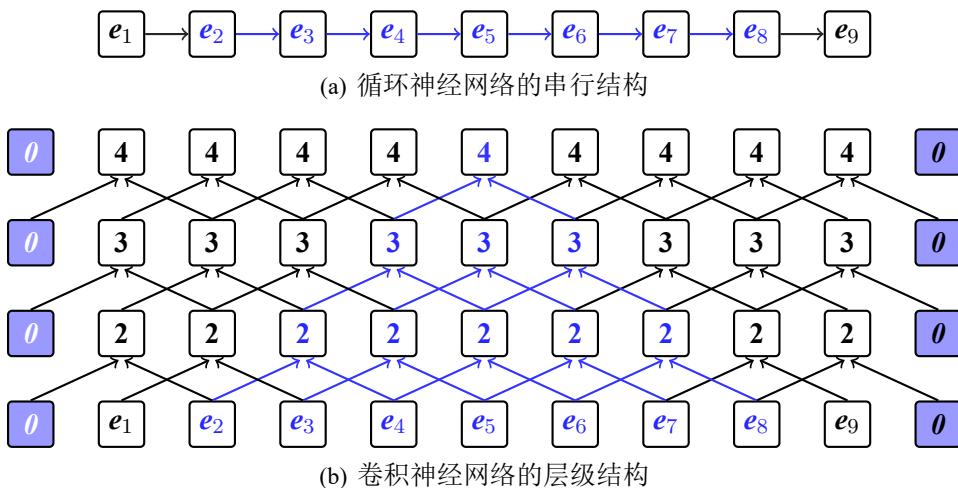


图 11.9 串行及层级结构对比 ( $e_i$  表示词嵌入,  $\theta$  表示  $\theta$  向量, 方框里的 2、3、4 表示层次编号)

针对不定长序列，一种可行的方法是使用之前介绍过的循环神经网络进行信息提取，其本质也是基于权重共享的想法，在不同的时间步复用相同的循环神经网络单元进行处理。但是，循环神经网络最大的弊端在于每一时刻的计算都依赖于上一时刻的结果，因此只能对序列进行串行处理，无法充分利用硬件设备进行并行计算，导致效率相对较低。此外，在处理较长的序列时，这种串行的方式很难捕捉长距离的依赖关系。相比之下，卷积神经网络采用共享参数的方式处理固定大小窗口内的信息，且不同位置的卷积操作之间没有相互依赖，因此可以对序列进行高效地并行处理。同时，针对序列中距离较长的依赖关系，可以通过堆叠多层卷积层来扩大**感受野** (Receptive Field)，这里感受野指能够影响神经元输出的原始输入数据区域的大小。图11.9对比了这两种结构，可以看出，为了捕捉  $e_2$  和  $e_8$  之间的联系，串行结构需要顺序地进行 6 次操作，和序列长度相关。而该卷积神经网络中，卷积操作每次对三个词进行计算，仅需要 4 层卷积计算就能得到  $e_2$  和  $e_8$  之间的联系，其操作数和卷积核的大小相关，相比于串行的方式具有更短的路径和更少的非线性计算，更容易进行训练。因此，也有许多研究人员在许多自然语言处理任务上尝试使用卷积神经网络进行序列建模<sup>[495, 496, 498, 505, 506]</sup>。

区别于传统图像上的卷积操作，在面向序列的卷积操作中，卷积核只在序列这一维度进行移动，用来捕捉连续的多个词之间的特征。需要注意的是，由于单词通常由一个实数向量表示（词嵌入），因此可以将词嵌入的维度看作是卷积操作中的通道数。图11.10就是一个基于序列卷积的文本分类模型，模型的输入是维度大小为

$m \times O$  的句子表示， $m$  表示句子长度， $O$  表示卷积核通道数，其值等于词嵌入维度，模型使用多个不同（对应图中不同的颜色）的卷积核来对序列进行特征提取，得到了多个不同的特征序列。然后使用池化层降低表示维度，得到了一组和序列长度无关的特征表示。最后模型基于这组压缩过的特征表示，使用全连接网络和 Softmax 函数进行类别预测。在这过程中卷积层和池化层分别起到了特征提取和特征压缩的作用，将一个不定长的序列转化为一组固定大小的特征表示。

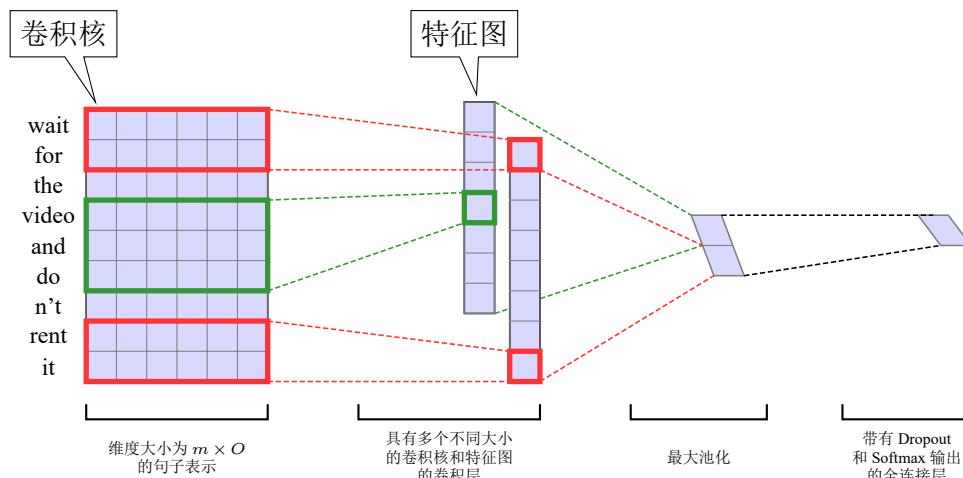


图 11.10 序列卷积在文本分类模型中的应用<sup>[496]</sup>

和其它自然语言处理任务不同的是，机器翻译中需要对序列进行全局表示，换句话说，模型需要捕捉序列中各个位置之间的关系。因此，基于卷积神经网络的神经机器翻译模型需要堆叠多个卷积层进行远距离的依赖关系的建模。同时，为了在多层网络中维持序列的原有长度，需要在卷积操作前对输入序列进行填充。图11.11是一个简单的示例，针对一个长度  $m = 6$  的句子，其隐层表示维度即卷积操作的输入通道数是  $O = 4$ ，卷积核大小为  $K = 3$ 。首先对序列进行填充，得到一个长度为 8 的序列，然后使用这些卷积核在这之上进行特征提取。一共使用了  $N = 4$  个卷积核，整体的参数量为  $K \times O \times N$ ，最后的卷积结果为  $m \times N$  的序列表示。

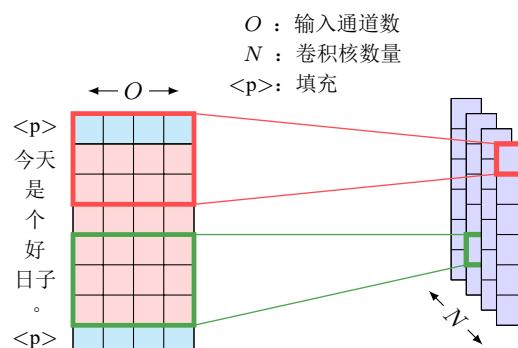


图 11.11 机器翻译中的序列卷积操作

## 11.2 基于卷积神经网络的翻译建模

正如之前所讲，卷积神经网络可以用于序列建模，同时具有并行性高和易于学习的特点，一个很自然的想法就是将其用作神经机器翻译模型中的特征提取器。因此，在神经机器翻译被提出之初，研究人员就已经开始利用卷积神经网络对句子进行特征提取。比较经典的模型是使用卷积神经网络作为源语言句子的编码器，使用循环神经网络作为目标语言译文生成的解码器<sup>[451, 501]</sup>。之后也有研究人员提出完全基于卷积神经网络的翻译模型（ConvS2S）<sup>[24]</sup>，或者针对卷积层进行改进，提出效率更高、性能更好的模型<sup>[502, 507]</sup>。本节将基于 ConvS2S 模型，阐述如何使用卷积神经网络搭建端到端神经机器翻译模型。

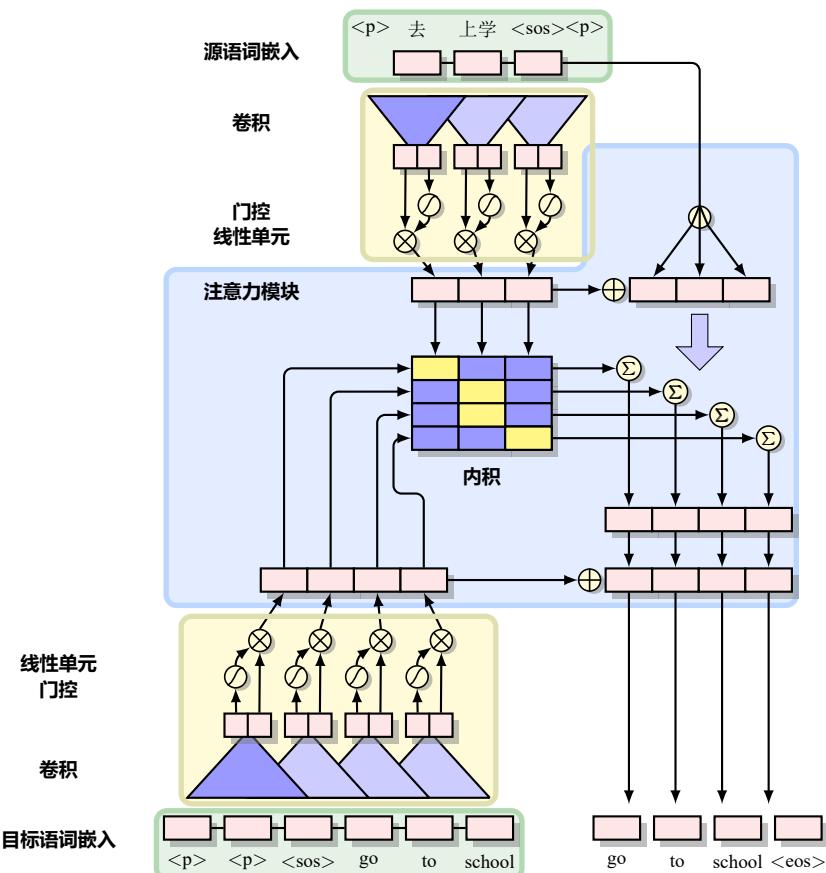


图 11.12 ConvS2S 模型结构

ConvS2S 模型是一种高并行的序列到序列的神经计算模型。该模型利用卷积神经网络分别对源语言端与目标语言端的序列进行特征提取，并使用注意力机制来捕获两个序列之间映射关系。相比于基于多层循环神经网络的 GNMT 模型<sup>[454]</sup>，其主要优势在于每一层的网络计算是完全并行化的，避免了循环神经网络中计算顺序对时序的依赖。同时，利用多层卷积神经网络的层级结构可以有效地捕捉序列不同位置

之间的依赖。即使是远距离依赖，也可以通过若干层卷积单元进行有效的捕捉，而且其信息传递的路径相比循环神经网络更短。除此之外，模型同时使用门控线性单元、残差网络和位置编码等技术来进一步提升模型性能，达到了和 GNMT 模型相媲美的翻译性能，同时大大缩短了训练时间。

图11.12为ConvS2S模型的结构示意图，其内部由若干不同的模块组成，包括：

- **位置编码** (Position Embedding)：图中绿色背景框表示源语言端词嵌入部分。相比于基于循环神经网络的翻译模型中的词嵌入，该模型还引入了位置编码，帮助模型获得词位置信息。位置编码具体实现在图11.12中没有显示，详见11.2.1节。
- **卷积层与门控线性单元** (Gated Linear Units, GLU)：黄色背景框是卷积模块，这里使用门控线性单元作为非线性函数，之前的研究工作<sup>[500]</sup>表明这种非线性函数更适合于序列建模任务。图中为了简化，只展示了一层卷积，但在实际中为了更好地捕获句子信息，通常使用多层卷积的叠加。
- **残差连接** (Residual Connection)：源语言端和目标语言端的卷积层网络之间，都存在一个从输入到输出的额外连接，即跳接<sup>[421]</sup>。该连接方式确保每个隐层输出都能包含输入序列中的更多信息，同时能够有效提高深层网络的信息传递效率（该部分在图11.12中没有显示，具体结构详见11.2.3节）。
- **多跳注意力机制** (Multi-step Attention/Multi-hop Attention)：蓝色框内部展示了基于多跳结构的注意力机制模块<sup>[508]</sup>。ConvS2S模型同样使用注意力机制来捕捉两个序列之间不同位置的对应关系。区别于之前的做法，多跳注意力在解码器端每一个层都会执行注意力操作。下面将以此模型为例对基于卷积神经网络的机器翻译模型进行介绍。

### 11.2.1 位置编码

与基于循环神经网络的翻译模型类似，基于卷积神经网络的翻译模型同样用词嵌入序列来表示输入序列，记为  $w = \{w_1, \dots, w_m\}$ 。序列  $w$  是维度大小为  $m \times d$  的矩阵，第  $i$  个单词  $w_i$  是维度为  $d$  的向量，其中  $m$  为序列长度， $d$  为词嵌入向量维度。和循环神经网络不同的是，基于卷积神经网络的模型需要对每个输入单词位置进行表示。这是由于，在卷积神经网络中，受限于卷积核的大小，单层的卷积神经网络只能捕捉序列局部的相对位置信息。虽然多层的卷积神经网络可以扩大感受野，但是对全局的位置表示并不充分。而相较于基于卷积神经网络的模型，基于循环神经网络的模型按时间步对输入的序列进行建模，这样间接的对位置信息进行了建模。而词序又是自然语言处理任务中重要信息，因此这里需要单独考虑。

为了更好地引入序列的词序信息，该模型引入了位置编码  $p = \{p_1, \dots, p_m\}$ ，其中  $p_i$  的维度大小为  $d$ ，一般和词嵌入维度相等，其中具体数值作为网络可学习的参数。简单来说， $p_i$  是一个可学习的参数向量，对应位置  $i$  的编码。这种编码的作用就是对位置信息进行表示，不同序列中的相同位置都对应一个唯一的位置编码向量。之后将词嵌入矩阵和位置编码进行相加，得到模型的输入序列  $e = \{w_1 + p_1, \dots, w_m + p_m\}$ 。

也有研究人员发现卷积神经网络本身具备一定的编码位置信息的能力<sup>[509]</sup>，而这里额外的位置编码模块可以被看作是对卷积神经网络位置编码能力的一种补充。

### 11.2.2 门控卷积神经网络

单层卷积神经网络的感受野受限于卷积核的大小，因此只能捕捉序列中局部的上下文信息，不能很好地进行长序列建模。为了捕捉更长的上下文信息，最简单的方法就是堆叠多个卷积层。相比于循环神经网络的链式结构，对相同的上下文跨度，多层卷积神经网络的层级结构可以通过更少的非线性计算对其进行建模，缓解了长距离建模中的梯度消失问题。因此，卷积神经网络相对更容易进行训练。

在 ConvS2S 模型中，编码器和解码器分别使用堆叠的门控卷积神经网络对源语言和目标语言序列进行建模，在传统卷积神经网络的基础上引入了门控线性单元<sup>[500]</sup>，通过门控机制对卷积输出进行控制，它在模型中的位置如图11.13黄色方框所示：

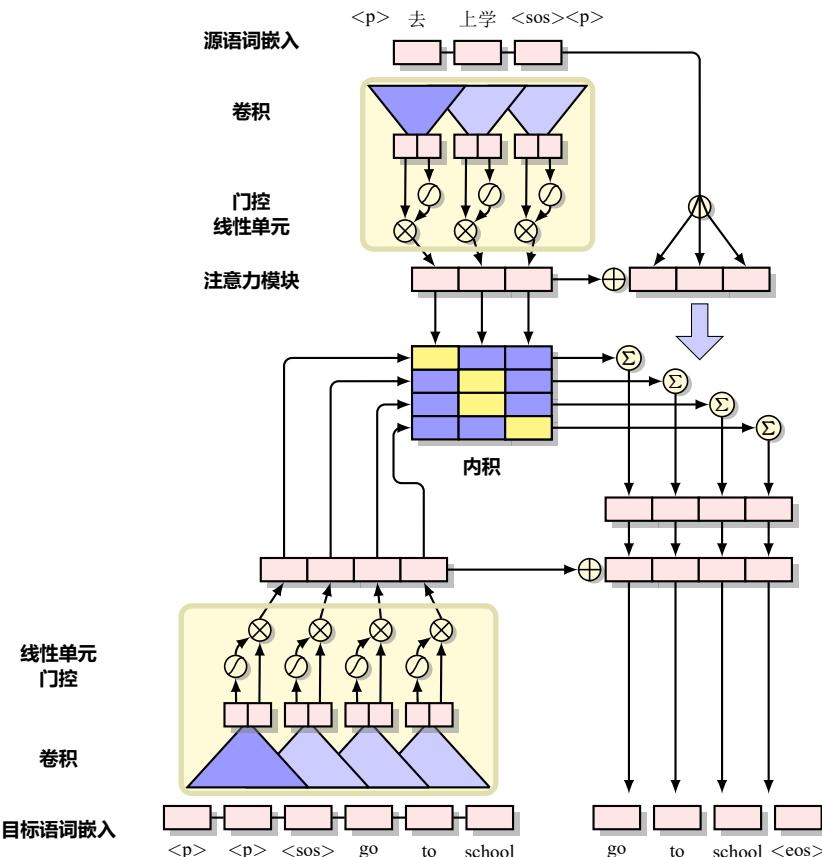


图 11.13 门控卷积神经网络机制在模型中的位置（黄色背景框部分）

门控机制在第十章中介绍 LSTM 模型时已经提到过。在 LSTM 模型中，可以通过引入三个门控单元来控制信息流，使隐层状态能够获得长时间记忆。同时，门控单元的引入简化了不同时间步间状态更新的计算，只包括一些线性计算，缓解了长

距离建模中梯度消失的问题。在多层卷积神经网络中，同样可以通过门控机制来起到相同的作用。

图11.14是单层门控卷积神经网络的基本结构， $x \in \mathbb{R}^{m \times d}$  为单层网络的输入， $y \in \mathbb{R}^{m \times d}$  为单层网络的输出，网络结构主要包括卷积计算和 GLU 非线性单元两部分。

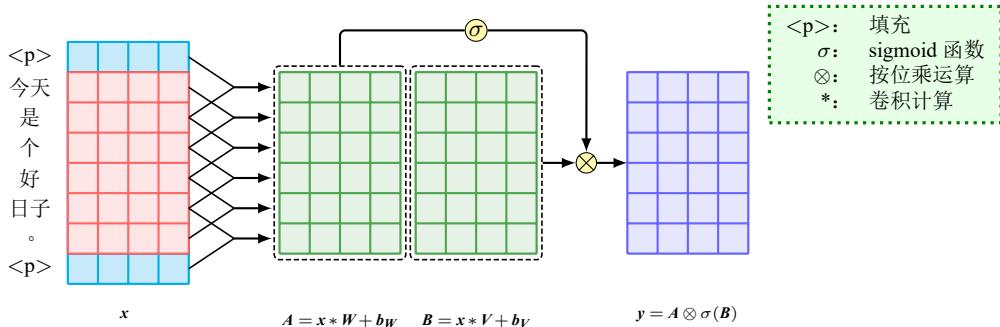


图 11.14 单层门控卷积神经网络结构

如图所示，形式上，卷积操作可以分成两部分，分别使用两个卷积核来得到两个卷积结果，具体计算如公式(11.3)和(11.4)所示：

$$A = x * W + b_W \quad (11.3)$$

$$B = x * V + b_V \quad (11.4)$$

其中， $A, B \in \mathbb{R}^d$ ， $W \in \mathbb{R}^{K \times d \times d}$ 、 $V \in \mathbb{R}^{K \times d \times d}$ 、 $b_W, b_V \in \mathbb{R}^d$ ， $W, V$ 在此表示卷积核， $b_W, b_V$ 为偏置矩阵。在卷积操作之后，引入非线性变换，具体计算如下：

$$y = A \otimes \sigma(B) \quad (11.5)$$

其中， $\sigma$  为 Sigmoid 函数， $\otimes$  为按位乘运算。Sigmoid 将  $B$  映射为 0-1 范围内的实数，用来充当门控。可以看到，门控卷积神经网络中核心部分就是  $\sigma(B)$ ，通过这个门控单元来对卷积输出进行控制，确定保留哪些信息。同时，在梯度反向传播的过程中，这种机制使得不同层之间存在线性的通道，梯度传导更加简单，利于深层网络的训练。这种思想和11.2.3节将要介绍的残差网络也很类似。

在 ConvS2S 模型中，为了保证卷积操作之后的序列长度不变，需要对输入进行填充，这一点已经在之前的章节中讨论过了。因此，在编码器每一次卷积操作前，需要对序列的头部和尾部分别做相应的填充（如图11.14左侧部分）。而在解码器中，由于需要训练和解码的一致性，模型在训练过程中不能使用未来的信息，需要对未来信息进行屏蔽，也就是屏蔽掉当前译文单词右侧的译文信息。从实践角度来看，只需要对解码器输入序列的头部填充  $K - 1$  个空元素，其中  $K$  为卷积核的宽度（图11.15展示了卷积核宽度  $K=3$  时，解码器对输入序列的填充情况，图中三角形表示卷积操作）。

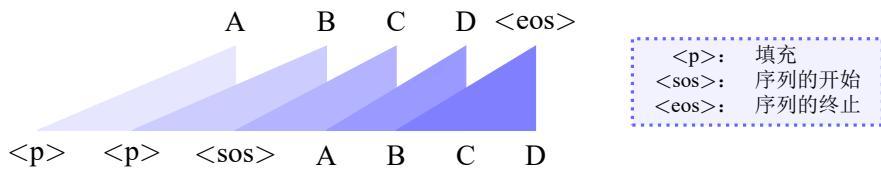


图 11.15 解码器的填充方法

### 11.2.3 残差网络

残差连接是一种训练深层网络的技术，其内容在第九章已经进行了介绍，即在多层神经网络之间通过增加直接连接的方式，从而将底层信息直接传递给上层。通过增加这样的直接连接，可以让不同层之间的信息传递更加高效，有利于深层神经网络的训练，其计算公式为：

$$\mathbf{h}^{l+1} = F(\mathbf{h}^l) + \mathbf{h}^l \quad (11.6)$$

其中， $\mathbf{h}^l$  表示  $l$  层神经网络的输入向量， $F(\mathbf{h}^l)$  是  $l$  层神经网络的运算。如果  $l=2$ ，那么公式(11.6)可以解释为：第 3 层的输入  $\mathbf{h}^3$  等于第 2 层的输出  $F(\mathbf{h}^2)$  加上第 2 层的输入  $\mathbf{h}^2$ 。

在 ConvS2S 中残差连接主要应用于门控卷积神经网络和多跳自注意力机制中，比如在编码器的多层门控卷积神经网络中，在每一层的输入和输出之间增加残差连接，具体的数学描述如下：

$$\mathbf{h}^{l+1} = \mathbf{A}^l \otimes \sigma(\mathbf{B}^l) + \mathbf{h}^l \quad (11.7)$$

### 11.2.4 多跳注意力机制

ConvS2S 模型也采用了注意力机制来获取每个目标语言位置相应的源语言上下文信息。其仍然沿用传统的点乘注意力机制<sup>[25]</sup>，其中图11.16蓝色框代表了多跳自注意力机制在模型中的位置。

在基于循环神经网络的翻译模型中，注意力机制已经被广泛使用<sup>[22]</sup>，并用于避免循环神经网络将源语言序列压缩成一个固定维度的向量表示带来的信息损失。另一方面，注意力同样能够帮助解码器区分源语言中不同位置对当前目标语言位置的贡献度，其具体的计算过程如公式(11.8)和(11.9)所示：

$$\mathbf{C}_j = \sum_i \alpha_{i,j} \mathbf{h}_i \quad (11.8)$$

$$\alpha_{i,j} = \frac{\exp(a(\mathbf{s}_{j-1}, \mathbf{h}_i))}{\sum_{i'} \exp(a(\mathbf{s}_{j-1}, \mathbf{h}_{i'}))} \quad (11.9)$$

其中,  $\mathbf{h}_i$  表示源语言端第  $i$  个位置的隐层状态, 即编码器在第  $i$  个位置的输出。 $\mathbf{s}_j$  表示目标端第  $j$  个位置的隐层状态。给定  $\mathbf{s}_j$  和  $\mathbf{h}_i$ , 注意力机制通过函数  $a(\cdot)$  计算目标语言表示  $\mathbf{s}_j$  与源语言表示  $\mathbf{h}_i$  之间的注意力权重  $\alpha_{i,j}$ , 通过加权平均得到当前目标语言端位置所需的上下文表示  $\mathbf{C}_j$ 。其中  $a(\cdot)$  的具体计算方式在第十章已经详细讨论。

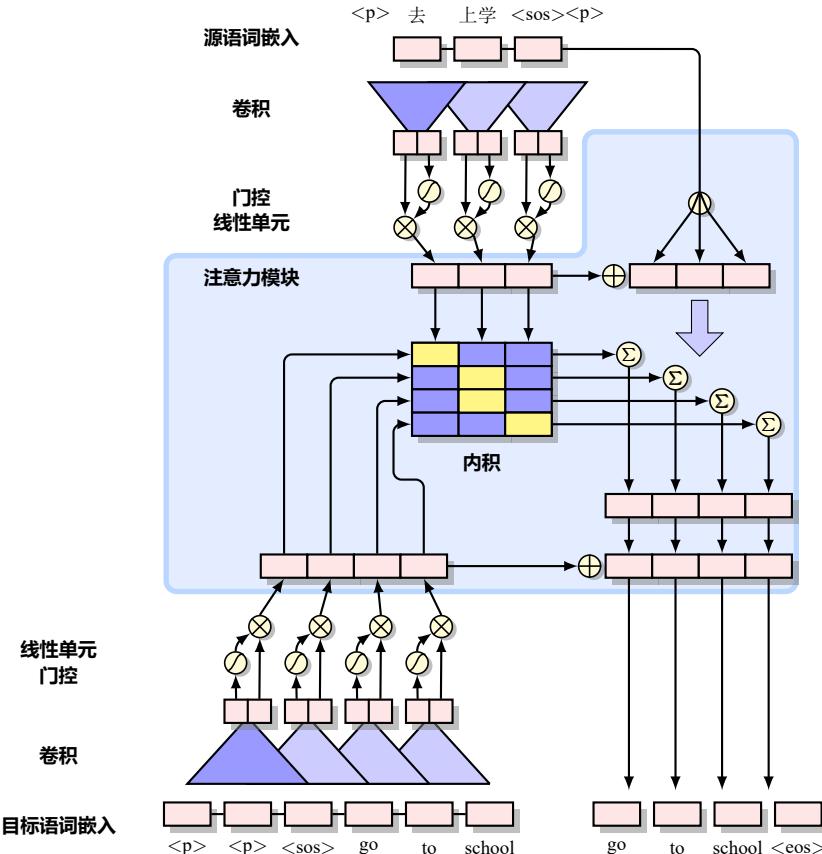


图 11.16 多跳自注意力机制在 ConvS2S 模型中的位置（蓝色背景框部分）

在 ConvS2S 模型中, 解码器同样采用堆叠的多层门控卷积网络来对目标语言进行序列建模。区别于编码器, 解码器在每一层卷积网络之后引入了注意力机制, 用来参考源语言信息。ConvS2S 选用了点乘注意力, 并且通过类似残差连接的方式将注意力操作的输入与输出同时作用于下一层计算, 称为多跳注意力。其具体计算方式如下:

$$\alpha_{ij}^l = \frac{\exp(\mathbf{d}_j^l \mathbf{h}_i)}{\sum_{i'=1}^m \exp(\mathbf{d}_j^l \mathbf{h}_{i'})} \quad (11.10)$$

不同于公式(11.9)中使用的目标语言端隐层表示  $\mathbf{s}_{j-1}$ , 公式(11.10)中的  $\mathbf{d}_j^l$  同时结合了  $\mathbf{s}_j$  的卷积计算结果和目标语言端的词嵌入  $\mathbf{g}_j$ , 其具体计算如公式(11.11)和(11.12)所

示：

$$\mathbf{d}_j^l = \mathbf{W}_d^l \mathbf{z}_j^l + \mathbf{b}_d^l + \mathbf{g}_j \quad (11.11)$$

$$\mathbf{z}_j^l = \text{Conv}(\mathbf{s}_j^l) \quad (11.12)$$

其中， $\mathbf{z}_j^l$  表示第  $l$  层卷积网络输出中第  $j$  个位置的表示， $\mathbf{W}_d^l$  和  $\mathbf{b}_d^l$  是模型可学习的参数， $\text{Conv}(\cdot)$  表示卷积操作。在获得第  $l$  层的注意力权重之后，就可以得到对应的一个上下文表示  $\mathbf{C}_j^l$ ，具体计算如下：

$$\mathbf{C}_j^l = \sum_i \alpha_{ij}^l (\mathbf{h}_i + \mathbf{e}_i) \quad (11.13)$$

模型使用了更全面的源语言信息，同时考虑了源语言端编码表示  $\mathbf{h}_i$  以及词嵌入表示  $\mathbf{e}_i$ 。在获得第  $l$  层的上下文向量  $\mathbf{C}_j^l$  后，模型将其与  $\mathbf{z}_j^l$  相加后送入下一层网络，这个过程可以被描述为：

$$\mathbf{s}_j^{l+1} = \mathbf{C}_j^l + \mathbf{z}_j^l \quad (11.14)$$

与循环网络中的注意力机制相比，该机制能够帮助模型甄别已经考虑了哪些先前的输入。也就是说，多跳的注意力机制会考虑模型之前更关注哪些单词，并且之后层中执行多次注意力的“跳跃”。

### 11.2.5 训练与推断

与基于循环神经网络的翻译模型一致，ConvS2S 模型会计算每个目标语言位置上不同单词的概率，并以交叉熵作为损失函数来衡量模型预测分布与标准分布之间的差异。同时，采用基于梯度的方法对网络中的参数进行更新（见第九章）。

ConvS2S 模型应用了很多工程方面的调整，主要包括：

- ConvS2S 模型使用了 **Nesterov 加速梯度下降法**（Nesterov Accelerated Gradient, NAG），动量累计的系数设置为 0.99，当梯度范数超过 0.1 时重新进行规范化<sup>[510]</sup>；
- ConvS2S 模型中设置学习率为 0.25，每当模型在校验集上的困惑度不再下降时，便在每轮的训练后将学习率降低一个数量级，直至学习率小于一定的阈值（如 0.0004）。

Nesterov 加速梯度下降法和第九章介绍的 Momentum 梯度下降法类似，都使用了历史梯度信息，首先回忆一下 Momentum 梯度下降法，具体计算如公式(11.15)和(11.16)所

示：

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \mathbf{v}_t \quad (11.15)$$

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + (1 - \beta) \frac{\partial J(\mathbf{w}_t)}{\partial \mathbf{w}_t} \quad (11.16)$$

其中， $\mathbf{w}_t$  表示第  $t$  步更新时的模型参数； $J(\mathbf{w}_t)$  表示损失函数均值期望的估计； $\frac{\partial J(\mathbf{w}_t)}{\partial \mathbf{w}_t}$  将指向  $J(\mathbf{w}_t)$  在  $\mathbf{w}_t$  处变化最大的方向，即梯度方向； $\alpha$  为学习率； $\mathbf{v}_t$  为损失函数在前  $t - 1$  步更新中累积的梯度动量，利用超参数  $\beta$  控制累积的范围。

而在 Nesterov 加速梯度下降法中，使用的梯度不是来自于当前参数位置，而是按照之前梯度方向更新一小步的位置，以便于更好地“预测未来”，提前调整更新速率，因此，其动量的更新方式如下：

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + (1 - \beta) \frac{\partial J(\mathbf{w}_t)}{\partial (\mathbf{w}_t - \alpha \beta \mathbf{v}_{t-1})} \quad (11.17)$$

Nesterov 加速梯度下降法利用了二阶导数的信息，可以做到“向前看”，加速收敛过程<sup>[511]</sup>。为了模型的稳定训练。ConvS2S 模型也采用了一些网络正则化和参数初始化的策略，使得模型在前向计算和反向计算过程中方差尽可能保持一致。

此外，ConvS2S 模型为了进一步提升训练效率及性能，还使用了小批量训练，即每次从样本中选择出一小部分数据进行训练。同时，ConvS2S 模型中也使用了 Dropout 方法<sup>[512]</sup>。除了在词嵌入层和解码器输出层应用 Dropout 外，ConvS2S 模型还对卷积块的输入层应用了 Dropout。

ConvS2S 模型的推断过程与第十章中描述的推断过程一样。其基本思想是：依靠源语言句子和前面已经生成的译文单词来预测下一个译文单词。这个过程也可以结合贪婪搜索或者束搜索等解码策略。

## 11.3 局部模型的改进

在序列建模中，卷积神经网络可以通过参数共享，高效地捕捉局部上下文特征，如图11.11所示。但是通过进一步分析可以发现，在标准卷积操作中包括了不同词和不同通道之间两种信息的交互，每个卷积核都是对相邻词的不同通道进行卷积操作，参数量为  $K \times O$ ，其中， $K$  为卷积核大小， $O$  为输入的通道数，即单词表示的维度大小。如果使用  $N$  个卷积核，得到  $N$  个特征（即输出通道数），总共的参数量为  $K \times O \times N$ 。这里涉及卷积核大小、输入通道数和输出通道数三个维度，因此计算复杂度较高。为了进一步提升计算效率，降低参数量，一些研究人员提出**深度可分离卷积**（Depthwise Separable Convolution），将空间维度和通道间的信息交互分离成**深度卷积**（Depthwise Convolution，也叫逐通道卷积）和**逐点卷积**（Pointwise Convolution）两部分<sup>[513, 514]</sup>。除了直接将深度可分离卷积应用到神经机器翻译中<sup>[502]</sup>，研究人员提出

使用更高效的**轻量卷积** (Lightweight Convolution) 和**动态卷积** (Dynamic Convolution) 来进行不同词之间的特征提取<sup>[507]</sup>。本节将主要介绍这些改进的卷积操作。在后续章节中也会看到这些模型在神经机器翻译中的应用。

### 11.3.1 深度可分离卷积

根据前面的介绍，可以看到卷积神经网络容易用于局部检测和处理位置不变的特征。对于特定的表达，比如地点、情绪等，使用卷积神经网络能达到不错的识别效果，因此它常被用在文本分类中<sup>[495, 496, 505, 515]</sup>。不过机器翻译所面临的情况更复杂，除了局部句子片段信息，研究人员还希望模型能够捕获句子结构、语义等信息。虽然单层卷积神经网络在文本分类中已经取得了很好的效果<sup>[496]</sup>，但是神经机器翻译等任务仍然需要有效的卷积神经网络。随着深度可分离卷积在机器翻译中的探索<sup>[502]</sup>，更高效的网络结构被设计出来，获得了比 ConvS2S 模型更好的性能。

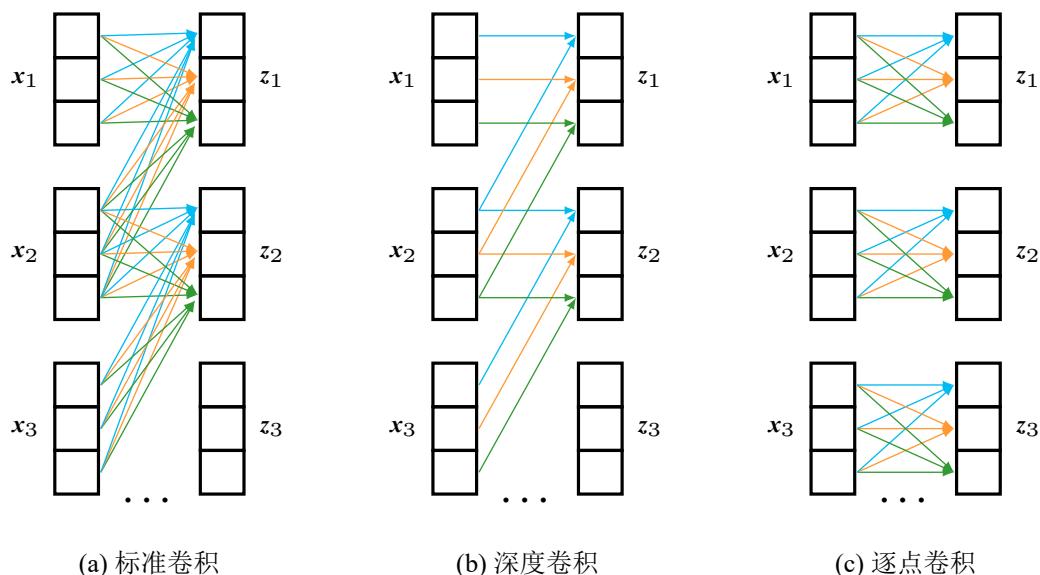


图 11.17 标准卷积、深度卷积和逐点卷积示意图

深度可分离卷积由深度卷积和逐点卷积两部分结合而成<sup>[516]</sup>。图11.17对比了标准卷积、深度卷积和逐点卷积，为了方便显示，图中只画出了部分连接。

给定输入序列表示  $x = \{x_1, \dots, x_m\}$ ，其中  $m$  为序列长度， $x_i \in \mathbb{R}^O$ ， $O$  即输入序列的通道数。为了获得与输入序列长度相同的卷积输出结果，首先需要进行填充。为了方便描述，这里在输入序列尾部填充  $K - 1$  个元素（ $K$  为卷积核窗口的长度），其对应的卷积结果为  $z = \{z_1, \dots, z_m\}$ 。在标准卷积中，若使用  $N$  表示卷积核的个数，也就是标准卷积输出序列的通道数，那么对于第  $i$  个位置的第  $n$  个通道  $z_{i,n}^{\text{std}}$ ，其标准

卷积具体计算如下：

$$z_{i,n}^{\text{std}} = \sum_{o=1}^O \sum_{k=0}^{K-1} W_{k,o,n}^{\text{std}} x_{i+k,o} \quad (11.18)$$

其中， $z^{\text{std}}$  表示标准卷积的输出， $z_i^{\text{std}} \in \mathbb{R}^N$ ， $W^{\text{std}} \in \mathbb{R}^{K \times O \times N}$  为标准卷积的参数。可以看出，标准卷积中每个输出元素需要考虑卷积核尺度内所有词的所有特征，参数量相对较多，对应图11.17中的连接数也最多。

相应的，深度卷积只考虑不同词之间的依赖性，而不考虑不同通道之间的关系，相当于使用  $O$  个卷积核逐个通道对不同的词进行卷积操作。因此深度卷积不改变输出的表示维度，输出序列表示的通道数与输入序列一致，其计算如下：

$$z_{i,o}^{\text{dw}} = \sum_{k=0}^{K-1} W_{k,o}^{\text{dw}} x_{i+k,o} \quad (11.19)$$

其中， $z^{\text{dw}}$  表示深度卷积的输出， $z_i^{\text{dw}} \in \mathbb{R}^O$ ， $W^{\text{dw}} \in \mathbb{R}^{K \times O}$  为深度卷积的参数，参数量只涉及卷积核大小及输入表示维度。

与深度卷积互为补充的是，逐点卷积只考虑不同通道之间的依赖性，而不考虑不同词之间的依赖。换句话说，逐点卷积对每个词表示做了一次线性变换，将输入表示  $x_i$  从  $\mathbb{R}^O$  的空间映射到  $\mathbb{R}^N$  的空间，其具体计算如下：

$$\begin{aligned} z_{i,n}^{\text{pw}} &= \sum_{o=1}^O x_{i,o} W_{o,n}^{\text{pw}} \\ &= x_i W^{\text{pw}} \end{aligned} \quad (11.20)$$

其中  $z^{\text{pw}}$  表示逐点卷积的输出， $z_i^{\text{pw}} \in \mathbb{R}^N$ ， $W^{\text{pw}} \in \mathbb{R}^{O \times N}$  为逐点卷积的参数。

表11.1展示了这几种不同类型卷积的参数量，深度可分离卷积通过将标准卷积进行分解，降低了整体模型的参数量。在相同参数量的情况下，深度可分离卷积可以采用更大的卷积窗口，考虑序列中更大范围的依赖关系。因此相比于标准卷积，深度可分离卷积具有更强的表示能力，在机器翻译任务中也能获得更好的性能。

表 11.1 不同类型卷积的参数量对比（ $K$  表示卷积核大小， $O$  表示输入通道数， $N$  表示输出通道数）  
[455]

卷积类型	参数量
标准卷积	$K \times O \times N$
深度卷积	$K \times O$
逐点卷积	$O \times N$
深度可分离卷积	$K \times O + O \times N$

### 11.3.2 轻量卷积和动态卷积

深度可分离卷积中深度卷积的作用就是用来捕捉相邻词之间的依赖关系，这和第十二章即将介绍的基于自注意力机制的模型类似。基于深度卷积，一些研究人员提出了轻量卷积和动态卷积，用来替换注意力机制，并将其应用于基于自注意力机制的模型中<sup>[507]</sup>。同时，卷积操作的线性复杂度使得它具有较高的运算效率，相比注意力机制的平方复杂度，卷积操作是一种更加“轻量”的方法。接下来分别介绍轻量卷积与动态卷积的思想。

#### 1. 轻量卷积

在序列建模的模型中，一个很重要的模块就是对序列中不同位置信息的提取，如ConvS2S模型中的卷积神经网络等。虽然考虑局部上下文的卷积神经网络只在序列这一维度进行操作，具有线性的复杂度，但是由于标准卷积操作中考虑了不同通道的信息交互，整体复杂度依旧较高。一种简化的策略就是采取通道独立的卷积操作，也就是11.3.1节中介绍的深度卷积。

在神经机器翻译模型中，神经网络不同层的维度通常一致，即  $O = N = d$ 。因此，深度卷积可以使得卷积神经网络参数量从  $Kd^2$  降到  $Kd$ （参考表11.1）。从形式上来看，深度卷积和注意力机制很类似，区别在于注意力机制考虑了序列全局上下文信息，权重来自于当前位置对其他位置的“注意力”，而深度卷积中仅考虑了局部的上下文信息，权重采用了在不同通道上独立的固定参数。为了进一步降低参数量，轻量卷积共享了部分通道的卷积参数。如图11.18所示，深度卷积中4种颜色的连接代表了4个通道上独立的卷积核，而轻量卷积中，第一和第三通道，第二和第四通道采用了共享的卷积核参数。通过共享，可以将参数量压缩到  $Ka$ ，其中压缩比例为  $d/a$  ( $a$  为压缩后保留的共享通道数)。

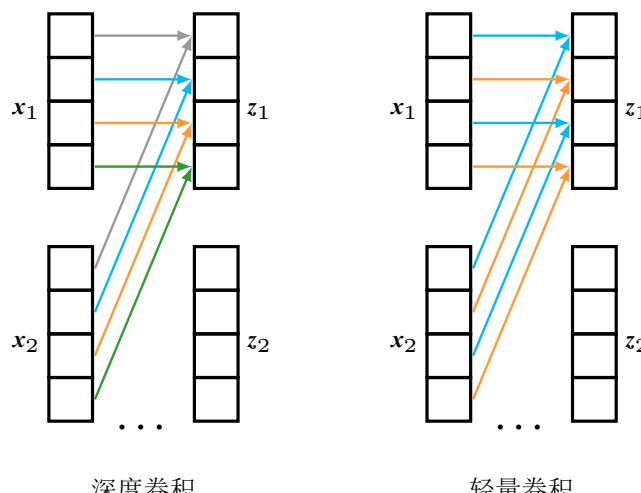


图 11.18 深度卷积 vs 轻量卷积

此外，和标准卷积不同的是，轻量卷积之前需要先对卷积参数进行归一化，具体计算过程为：

$$z_{i,o}^{\text{lw}} = \sum_{k=0}^{K-1} \text{Softmax}(\mathbf{W}^{\text{lw}})_{k,[\frac{oa}{d}]} \mathbf{x}_{i+k,o} \quad (11.21)$$

其中， $z^{\text{lw}}$  表示轻量卷积的输出， $z_i^{\text{lw}} \in \mathbb{R}^d$ ， $\mathbf{W}^{\text{lw}} \in \mathbb{R}^{K \times a}$  为轻量卷积的参数。在这里，轻量卷积用来捕捉相邻词的特征，通过 Softmax 可以在保证关注到不同词的同时，对输出大小进行限制。

## 2. 动态卷积

轻量卷积和动态卷积的概念最早都是在图像领域被提出，大大减少了卷积神经网络模型中的参数和计算量<sup>[492, 517, 518]</sup>。虽然轻量卷积在存储和速度上具有优势，但其参数量的减少也导致了表示能力的下降，损失了一部分模型性能。为此，研究人员提出了动态卷积，旨在不增加网络深度和宽度的情况下增强模型的表示能力，其思想就是根据输入来动态地生成卷积参数<sup>[507, 519]</sup>。

在轻量卷积中，模型使用的卷积参数是静态的，与序列位置无关，维度大小为  $K \times a$ ；而在动态卷积中，为了增强模型的表示能力，卷积参数来自于当前位置输入的变换，具体计算为：

$$f(\mathbf{x}_i) = \sum_{c=1}^d \mathbf{W}_{::,c} \odot \mathbf{x}_{i,c} \quad (11.22)$$

这里采用了最简单的线性变换，其中  $\odot$  表示矩阵的点乘（详见第九章介绍）， $d$  为通道数， $\mathbf{x}_i$  是序列第  $i$  个位置的表示， $c$  表示某个通道， $\mathbf{W} \in \mathbb{R}^{K \times a \times d}$  为变换矩阵， $\mathbf{W}_{::,c}$  表示其只在  $d$  这一维进行计算，最后生成的  $f(\mathbf{x}_i) \in \mathbb{R}^{K \times a}$  就是与输入相关的卷积核参数。通过这种方式，模型可以根据不同位置的表示来确定如何关注其他位置信息的“权重”，更好地提取序列信息。同时，相比于注意力机制中两两位置确定出来的注意力权重，动态卷积线性复杂度的做法具有更高的计算效率。

## 11.4 小结及拓展阅读

卷积是一种高效的神经网络结构，在图像、语音处理等领域取得了令人瞩目的成绩。本章介绍了卷积的概念及其特性，并对池化、填充等操作进行了讨论。本章介绍了具有高并行计算能力的机器翻译范式，即基于卷积神经网络的编码器-解码器框架。其在机器翻译任务上表现出色，并大幅度缩短了模型的训练周期。除了基础部分，本章还针对卷积计算进行了延伸，内容涉及逐通道卷积、逐点卷积、轻量卷积和动态卷积等。除了上述提及的内容，卷积神经网络及其变种在文本分类、命名实体识别、关系分类、事件抽取等其他自然语言处理任务上也有许多应用<sup>[100, 496, 520, 521, 522]</sup>。

和机器翻译任务不同的是，文本分类任务侧重于对序列特征的提取，然后通过压缩后的特征表示做出类别预测。卷积神经网络可以对序列中一些  $n$ -gram 特征进行提取，也可以用在文本分类任务中，其基本结构包括输入层、卷积层、池化层和全连接层。除了在本章介绍过的 TextCNN 模型<sup>[496]</sup>，不少研究工作在此基础上对其进行改进。比如，通过改变输入层来引入更多特征<sup>[523, 524]</sup>，对卷积层的改进<sup>[521, 525]</sup>以及对池化层的改进<sup>[495, 521]</sup>。在命名实体识别任务中，同样可以使用卷积神经网络来进行特征提取<sup>[100, 520]</sup>，或者使用更高效的空洞卷积对更长的上下文进行建模<sup>[526]</sup>。此外，也有一些研究工作尝试使用卷积神经网络来提取字符级特征<sup>[527, 528, 529]</sup>。



## 12. 基于自注意力的模型

循环神经网络和卷积神经网络是两种经典的神经网络结构，在机器翻译中进行应用也是较为自然的想法。但是，这些模型在处理文字序列时也有问题：它们对序列中不同位置之间的依赖关系的建模并不直接。以卷积神经网络为例，如果要对长距离依赖进行描述，需要多层卷积操作，而且不同层之间信息传递也可能有损失，这些都限制了模型的能力。

为了更好地描述文字序列，研究人员提出了一种新的模型 Transformer。Transformer 并不依赖任何循环单元或者卷积单元，而是使用一种被称作自注意力网络的结构来对序列进行表示。自注意力可以非常高效的描述任意距离之间的依赖关系，因此非常适合处理语言文字序列。Transformer 一经提出就受到了广泛关注，现在已经成为了机器翻译中最先进的架构之一。本章将会对 Transformer 的基本结构和实现技术进行介绍。这部分知识也会在本书的前沿部分（第十三章~ 第十八章）中大量使用。

### 12.1 自注意力机制

首先回顾一下循环神经网络处理文字序列的过程。如图12.1所示，对于单词序列  $\{w_1, \dots, w_m\}$ ，处理第  $m$  个单词  $w_m$  时（绿色方框部分），需要输入前一时刻的信息（即处理单词  $w_{m-1}$ ），而  $w_{m-1}$  又依赖于  $w_{m-2}$ ，以此类推。也就是说，如果想建立  $w_m$  和  $w_1$  之间的关系，需要  $m-1$  次信息传递。对于长序列来说，词汇之间信息传递距离过长会导致信息在传递过程中丢失，同时这种按顺序建模的方式也使得系统

对序列的处理十分缓慢。

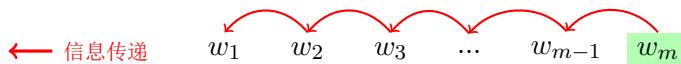


图 12.1 循环神经网络中单词之间的依赖关系

那么能否摆脱这种顺序传递信息的方式，直接对不同位置单词之间的关系进行建模，即将信息传递的距离拉近为 1？自注意力机制的提出便有效解决了这个问题<sup>[530]</sup>。图 12.2 给出了自注意力机制对序列进行建模的示例。对于单词  $w_m$ ，自注意力机制直接建立它与前  $m-1$  个单词之间的关系。也就是说， $w_m$  与序列中所有其他单词的距离都是 1。这种方式很好地解决了长距离依赖问题，同时由于单词之间的联系都是相互独立的，因此也大大提高了模型的并行度。

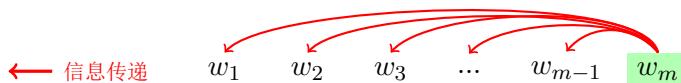


图 12.2 自注意力机制中单词之间的依赖关系

自注意力机制也可以被看作是一个序列表示模型。比如，对于每个目标位置  $j$ ，都生成一个与之对应的源语言句子表示，它的形式如下：

$$\mathbf{c}_j = \sum_i \alpha_{i,j} \mathbf{h}_i \quad (12.1)$$

其中， $\mathbf{h}_i$  为源语言句子每个位置的表示结果， $\alpha_{i,j}$  是目标位置  $j$  对  $\mathbf{h}_i$  的注意力权重。以源语言句子为例，自注意力机制将序列中每个位置的表示  $\mathbf{h}_i$  看作 query（查询），并且将所有位置的表示看作 key（键）和 value（值）。自注意力模型通过计算当前位置与所有位置的匹配程度，也就是在注意力机制中提到的注意力权重，来对各个位置的 value 进行加权求和。得到的结果可以被看作是在这个句子中当前位置的抽象表示。这个过程，可以叠加多次，形成多层注意力模型，对输入序列中各个位置进行更深层的表示。

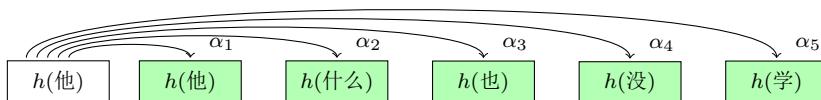


图 12.3 自注意力计算实例

举个例子，如图 12.3 所示，一个汉语句子包含 5 个词。这里，用  $h(\cdot)$  表示“他”当前的表示结果，其中  $h(\cdot)$  是一个函数，用于返回输入单词所在位置对应的表示结果（向量）。如果把“他”看作目标，这时 query 就是  $h(\text{他})$ ，key 和 value 是图中所有位置的表示，即： $h(\text{他})$ 、 $h(\text{什么})$ 、 $h(\text{也})$ 、 $h(\text{没})$ 、 $h(\text{学})$ 。在自注意力模型中，首

先计算 query 和 key 的相关度，这里用  $\alpha_i$  表示  $h(\text{他})$  和位置  $i$  的表示之间的相关性。然后，把  $\alpha_i$  作为权重，对不同位置上的 value 进行加权求和。最终，得到新的表示结果  $\tilde{h}(\text{他})$ ，其具体计算如下：

$$\begin{aligned}\tilde{h}(\text{他}) = & \alpha_1 h(\text{他}) + \alpha_2 h(\text{什么}) + \alpha_3 h(\text{也}) + \\ & \alpha_4 h(\text{没}) + \alpha_5 h(\text{学})\end{aligned}\quad (12.2)$$

同理，也可以用同样的方法处理这个句子中的其他单词。可以看出，在自注意力机制中，并不是使用类似于循环神经网络的记忆能力去访问历史信息。序列中所有单词之间的信息都是通过同一种操作（query 和 key 的相关度）进行处理。这样，表示结果  $\tilde{h}(\text{他})$  在包含“他”这个单词的信息的同时，也包含了序列中其他词的信息。也就是说，序列中每一个位置的表示结果中，都包含了其他位置的信息。从这个角度说， $\tilde{h}(\text{他})$  已经不再是单词“他”自身的表示结果，而是一种在单词“他”的位置上的全局信息的表示。

通常，也把生成  $\tilde{h}(w_i)$  的过程看作特征提取，而实现这个过程的模型被称为特征提取器。循环神经网络、卷积神经网络和自注意力模型都是典型的特征提取器。特征提取是神经机器翻译系统的关键步骤，在随后的内容中可以看到自注意力模型是一个非常适合机器翻译任务的特征提取器。

## 12.2 Transformer 架构

下面对 Transformer 模型的由来以及总体架构进行介绍。

### 12.2.1 Transformer 的优势

首先再来看看一下第十章介绍的循环神经网络，虽然它很强大，但是也存在一些弊端。其中比较突出的问题是，循环神经网络每个循环单元都有向前依赖性，也就是当前时间步的处理依赖前一时间步处理的结果。这个性质可以使序列的“历史”信息不断被传递，但是也造成模型运行效率的下降。特别是对于自然语言处理任务，序列往往较长，无论是传统的 RNN 结构，还是更为复杂的 LSTM 结构，都需要很多次循环单元的处理才能够捕捉到单词之间的长距离依赖。由于需要多个循环单元的处理，距离较远的两个单词之间的信息传递变得很复杂。

针对这些问题，研究人员提出了一种全新的模型——Transformer<sup>[23]</sup>。与循环神经网络等传统模型不同，Transformer 模型仅仅使用自注意力机制和标准的前馈神经网络，完全不依赖任何循环单元或者卷积操作。自注意力机制的优点在于可以直接对序列中任意两个单元之间的关系进行建模，这使得长距离依赖等问题可以更好地被求解。此外，自注意力机制非常适合在 GPU 上进行并行化，因此模型训练的速度

更快。表12.1对比了 RNN、CNN 和 Transformer 的层类型复杂度<sup>1</sup>。

表 12.1 RNN、CNN、Transformer 的层类型复杂度对比<sup>[23]</sup> ( $n$  表示序列长度,  $d$  表示隐层大小,  $k$  表示卷积核大小)

模型	层类型	复杂度	最小顺序操作数	最大路径长度
Transformer	自注意力	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
RNN	循环单元	$O(n \cdot d^2)$	$O(n)$	$O(n)$
CNN	空洞卷积	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

Transformer 在被提出之后，很快就席卷了整个自然语言处理领域。实际上，也可以把 Transformer 当作一种表示模型，因此也被大量地使用在自然语言处理的其他领域，甚至图像处理<sup>[531]</sup> 和语音处理<sup>[532, 533]</sup> 中也能看到它的影子。比如，目前非常流行的 BERT 等预训练模型就是基于 Transformer。表12.2展示了 Transformer 在 WMT 英德和英法机器翻译任务上的性能。它能用更少的计算量 (FLOPs) 达到比其他模型更好的翻译品质<sup>2</sup>。

表 12.2 不同翻译模型性能对比<sup>[23]</sup>

系统	BLEU[%]		模型训练代价 (FLOPs)
	EN-DE	EN-FR	
GNMT+RL	24.6	39.92	$1.4 \times 10^{20}$
ConvS2S	25.16	40.46	$1.5 \times 10^{20}$
MoE	26.03	40.56	$1.2 \times 10^{20}$
Transformer (Base Model)	27.3	38.1	$3.3 \times 10^{18}$
Transformer (Big Model)	<b>28.4</b>	<b>41.8</b>	$2.3 \times 10^{19}$

注意，Transformer 并不简单等同于自注意力机制。Transformer 模型还包含了很多优秀的技术，比如：多头注意力、新的训练学习率调整策略等等。这些因素一起组成了真正的 Transformer。下面就一起看一看自注意力机制和 Transformer 是如何工作的。

## 12.2.2 总体结构

图12.4展示了 Transformer 的结构。编码器由若干层组成（绿色虚线框就代表一层）。每一层 (Layer) 的输入都是一个向量序列，输出是同样大小的向量序列，而 Transformer 层的作用是对输入进行进一步的抽象，得到新的表示结果。不过这里的层并不是指单一的神经网络结构，它里面由若干不同的模块组成，包括：

<sup>1</sup>顺序操作数指模型处理一个序列所需要的操作数，由于 Transformer 和 CNN 都可以并行计算，所以是 1；路径长度指序列中任意两个单词在网络中的距离。

<sup>2</sup>FLOPs = Floating Point Operations，即浮点运算数。它是度量算法/模型复杂度的常用单位

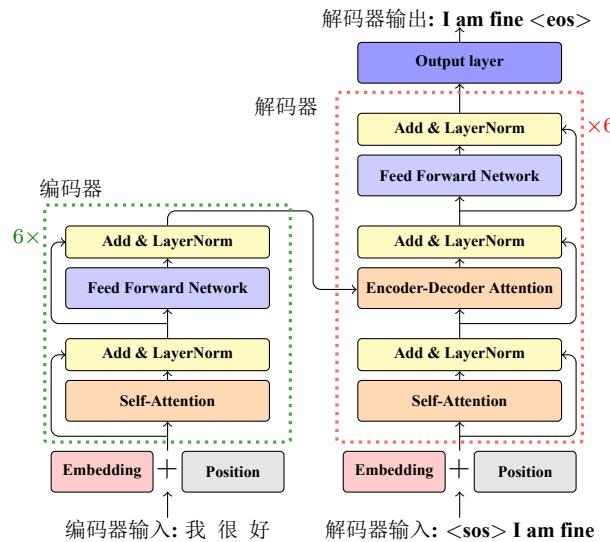


图 12.4 Transformer 结构

- **自注意力子层** (Self-Attention Sub-layer)：使用自注意力机制对输入的序列进行新的表示；
- **前馈神经网络子层** (Feed-Forward Sub-layer)：使用全连接的前馈神经网络对输入向量序列进行进一步变换；
- **残差连接** (标记为“Add”): 对于自注意力子层和前馈神经网络子层，都有一个从输入直接到输出的额外连接，也就是一个跨子层的直连。残差连接可以使深层网络的信息传递更为有效；
- **层标准化** (Layer Normalization)：自注意力子层和前馈神经网络子层进行最终输出之前，会对输出的向量进行层标准化，规范结果向量取值范围，这样易于后面进一步的处理。

以上操作就构成了 Transformer 的一层，各个模块执行的顺序可以简单描述为：Self-Attention → Residual Connection → Layer Normalization → Feed Forward Network → Residual Connection → Layer Normalization。编码器可以包含多个这样的层，比如，可以构建一个六层编码器，每层都执行上面的操作。最上层的结果作为整个编码的结果，会被传入解码器。

解码器的结构与编码器十分类似。它也是由若干层组成，每一层包含编码器中的所有结构，即：自注意力子层、前馈神经网络子层、残差连接和层标准化模块。此外，为了捕捉源语言的信息，解码器又引入了一个额外的**编码-解码注意力子层** (Encoder-Decoder Attention Sub-layer)。这个新的子层，可以帮助模型使用源语言句子的表示信息生成目标语言不同位置的表示。编码-解码注意力子层仍然基于自注意力机制，因此它和自注意力子层的结构是相同的，只是 query、key、value 的定义不同。比如，

在解码器端，自注意力子层的 query、key、value 是相同的，它们都等于解码器每个位置的表示。而在编码-解码注意力子层中，query 是解码器每个位置的表示，此时 key 和 value 是相同的，等于编码器每个位置的表示。图12.5给出了这两种不同注意力子层输入的区别。

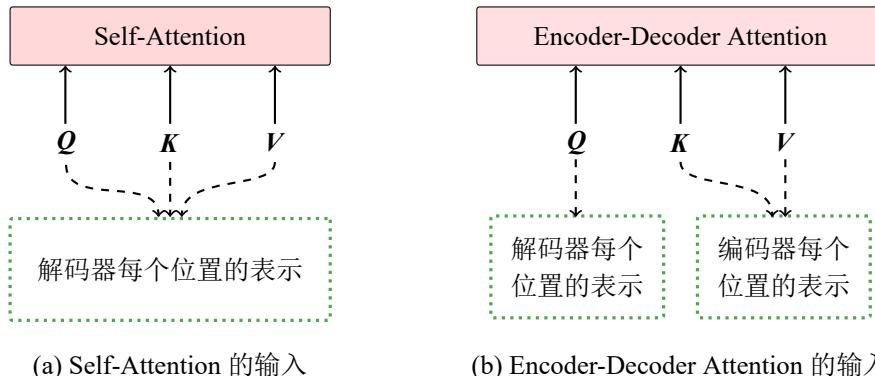


图 12.5 注意力模型的输入（自注意力子层 vs 编码-解码注意力子层）

此外，编码器和解码器都有输入的词序列。编码器的词序列输入是为了对其进行表示，进而解码器能从编码器访问到源语言句子的全部信息。解码器的词序列输入是为了进行目标语言的生成，本质上它和语言模型是一样的，在得到前  $n - 1$  个单词的情况下输出第  $n$  个单词。除了输入词序列的词嵌入，Transformer 中也引入了位置嵌入，以表示每个位置信息。原因是，自注意力机制没有显性地对位置进行表示，因此也无法考虑词序。在输入中引入位置信息可以让自注意力机制间接地感受到每个词的位置，进而保证对序列表示的合理性。最终，整个模型的输出由一个 Softmax 层完成，它和循环神经网络中的输出层是完全一样的。

在进行更详细的介绍前，先利用图12.4简单了解一下 Transformer 模型是如何进行翻译的。首先，Transformer 将源语言句子“我/很好”的词嵌入融合位置编码后作为输入。然后，编码器对输入的源语言句子进行逐层抽象，得到包含丰富的上下文信息的源语言表示并传递给解码器。解码器的每一层，使用自注意力子层对输入解码器的表示进行加工，之后再使用编码-解码注意力子层融合源语言句子的表示信息。就这样逐词生成目标语言译文单词序列。解码器每个位置的输入是当前单词（比如，“I”），而这个位置的输出是下一个单词（比如，“am”），这个设计和标准的神经语言模型是完全一样的。

当然，这里可能还有很多疑惑，比如，什么是位置编码？Transformer 的自注意力机制具体是怎么进行计算的，其结构是怎样的？层标准化又是什么？等等。下面就一一展开介绍。

## 12.3 位置编码

在使用循环神经网络进行序列的信息提取时，每个时刻的运算都要依赖前一个时刻的输出，具有一定的时序性，这也与语言具有顺序的特点相契合。而采用自注意力机制对源语言和目标语言序列进行处理时，直接对当前位置和序列中的任意位置进行建模，忽略了词之间的顺序关系，例如图12.6中两个语义不同的句子，通过自注意力得到的表示  $\tilde{h}$ (机票) 却是相同的。

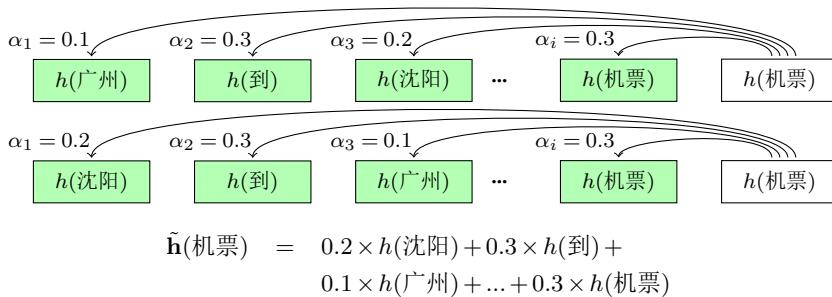


图 12.6 “机票”的更进一步抽象表示  $\tilde{h}$  的计算

为了解决这个问题，Transformer 在原有的词向量输入基础上引入了位置编码，来表示单词之间的顺序关系。位置编码在 Transformer 结构中的位置如图12.7，它是 Transformer 成功的一个重要因素。

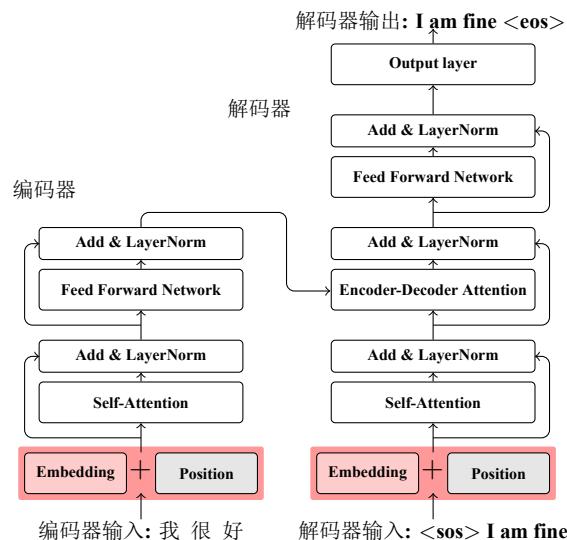


图 12.7 Transformer 输入与位置编码

位置编码的计算方式有很多种，Transformer 使用不同频率的正余弦函数，如公

式(12.3)和(12.4)所示：

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (12.3)$$

$$\text{PE}(\text{pos}, 2i + 1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (12.4)$$

式中  $\text{PE}(\cdot)$  表示位置编码的函数， $\text{pos}$  表示单词的位置， $i$  代表位置编码向量中的第几维， $d_{\text{model}}$  是 Transformer 的一个基础参数，表示每个位置的隐层大小。因为，正余弦函数的编码各占一半，因此当位置编码的维度为 512 时， $i$  的范围是 0-255。在 Transformer 中，位置编码的维度和词嵌入向量的维度相同（均为  $d_{\text{model}}$ ），模型通过将二者相加作为模型输入，如图12.8所示。

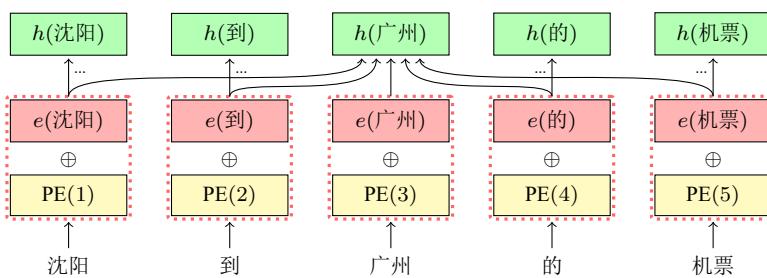


图 12.8 位置编码与词编码的组合

那么为什么通过这种计算方式可以很好的表示位置信息？有几方面原因。首先，正余弦函数是具有上下界的周期函数，用正余弦函数可将长度不同的序列的位置编码的范围都固定到  $[-1, 1]$ ，这样在与词的编码进行相加时，不至于产生太大差距。另外位置编码的不同维度对应不同的正余弦曲线，这为多维的表示空间赋予一定意义。最后，根据三角函数的性质，如公式(12.5)和(12.6)：

$$\sin(\alpha + \beta) = \sin\alpha \cdot \cos\beta + \cos\alpha \cdot \sin\beta \quad (12.5)$$

$$\cos(\alpha + \beta) = \cos\alpha \cdot \cos\beta - \sin\alpha \cdot \sin\beta \quad (12.6)$$

可以得到第  $pos + k$  个位置的编码，如公式(12.7)和(12.8)：

$$\begin{aligned} \text{PE}(\text{pos} + k, 2i) &= \text{PE}(\text{pos}, 2i) \cdot \text{PE}(k, 2i + 1) + \\ &\quad \text{PE}(\text{pos}, 2i + 1) \cdot \text{PE}(k, 2i) \end{aligned} \quad (12.7)$$

$$\begin{aligned} \text{PE}(\text{pos} + k, 2i + 1) &= \text{PE}(\text{pos}, 2i + 1) \cdot \text{PE}(k, 2i + 1) - \\ &\quad \text{PE}(\text{pos}, 2i) \cdot \text{PE}(k, 2i) \end{aligned} \quad (12.8)$$

即对于任意固定的偏移量  $k$ ， $\text{PE}(\text{pos} + k)$  能被表示成  $\text{PE}(\text{pos})$  的线性函数，换句话说，位置编码可以表示词之间的距离。在实践中发现，位置编码对 Transformer 系统

的性能有很大影响。对其进行改进也会带来进一步的性能提升<sup>[460]</sup>。

## 12.4 基于点乘的多头注意力机制

Transformer 模型摒弃了循环单元和卷积等结构，完全基于注意力机制来构造模型，其中包含着大量的注意力计算。比如，可以通过自注意力机制对源语言和目标语言序列进行信息提取，并通过编码-解码注意力对双语句对之间的关系进行建模。图12.9中红色方框部分是 Transformer 中使用注意力机制的模块。而这些模块都是由基于点乘的多头注意力机制实现的。

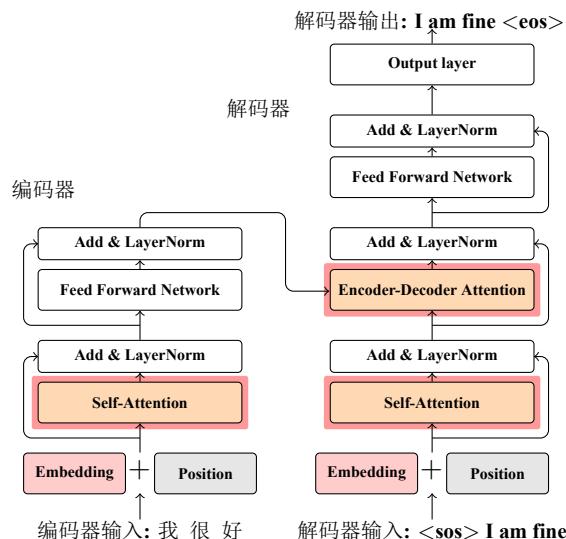


图 12.9 自注意力机制在模型中的位置

### 12.4.1 点乘注意力机制

在12.1节中已经介绍，自注意力机制中至关重要的是获取相关性系数，也就是在融合不同位置的表示向量时各位置的权重。Transformer 模型采用了一种基于点乘的方法来计算相关性系数。这种方法也称为**缩放的点乘注意力**（Scaled Dot-product Attention）机制。它的运算并行度高，同时并不消耗太多的存储空间。

具体来看，在注意力机制的计算过程中，包含三个重要的参数，分别是 query，key 和 value。在下面的描述中，分别用  $\mathbf{Q}$ ,  $\mathbf{K}$ ,  $\mathbf{V}$  对它们进行表示，其中  $\mathbf{Q}$  和  $\mathbf{K}$  的维度为  $L \times d_k$ ,  $\mathbf{V}$  的维度为  $L \times d_v$ 。这里， $L$  为序列的长度， $d_k$  和  $d_v$  分别表示每个 key 和 value 的大小，通常设置为  $d_k = d_v = d_{\text{model}}$ 。

在自注意力机制中， $\mathbf{Q}$ 、 $\mathbf{K}$ 、 $\mathbf{V}$  都是相同的，对应着源语言或目标语言序列的表示。而在编码-解码注意力机制中，由于要对双语之间的信息进行建模，因此，将目标语言每个位置的表示视为编码-解码注意力机制的  $\mathbf{Q}$ ，源语言句子的表示视为  $\mathbf{K}$  和

$V$ 。

在得到  $Q$ ,  $K$  和  $V$  后, 便可以进行注意力的运算, 这个过程可以被形式化为:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{Mask}\right)V \quad (12.9)$$

首先, 通过对  $Q$  和  $K$  的转置进行矩阵乘法操作, 计算得到一个维度大小为  $L \times L$  的相关性矩阵, 即  $QK^T$ , 它表示一个序列上任意两个位置的相关性。再通过系数  $1/\sqrt{d_k}$  进行放缩操作, 放缩可以减少相关性矩阵的方差, 具体体现在运算过程中实数矩阵中的数值不会过大, 有利于模型训练。

在此基础上, 通过对相关性矩阵累加一个掩码矩阵  $\text{Mask}$ , 来屏蔽掉矩阵中的无用信息。比如, 在编码器端, 如果需要对多个句子同时处理, 由于这些句子长度不统一, 需要对句子补齐。再比如, 在解码器端, 训练的时候需要屏蔽掉当前目标语言位置右侧的单词, 因此这些单词在推断的时候是看不到的。

随后, 使用 Softmax 函数对相关性矩阵在行的维度上进行归一化操作, 这可以理解为对第  $i$  行进行归一化, 结果对应了  $V$  中不同位置上向量的注意力权重。对于 value 的加权求和, 可以直接用相关性系数和  $V$  进行矩阵乘法得到, 即  $\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{Mask}\right)$  和  $V$  进行矩阵乘。最终得到自注意力的输出, 它和输入的  $V$  的大小是一模一样的。图12.10展示了点乘注意力计算的全过程。

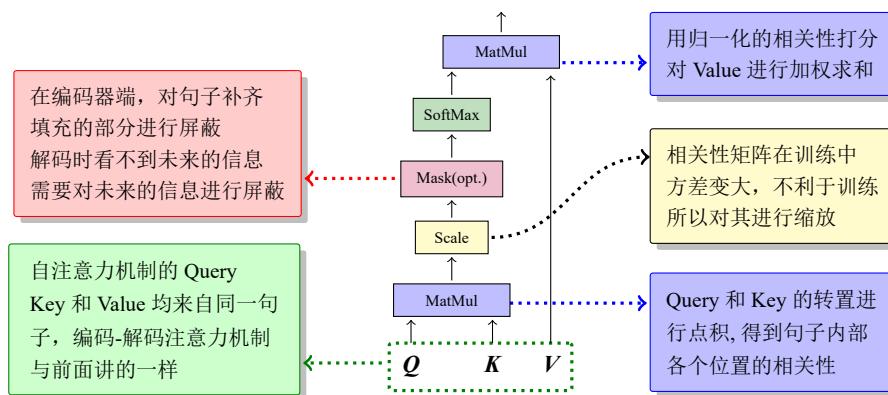


图 12.10 点乘注意力模型

下面举个简单的例子介绍点乘注意力的具体计算过程。如图12.11所示, 用黄色、蓝色和橙色的矩阵分别表示  $Q$ 、 $K$  和  $V$ 。 $Q$ 、 $K$  和  $V$  中的每一个小格都对应一个单词在模型中的表示 (即一个向量)。首先, 通过点乘、放缩、掩码等操作得到相关性矩阵, 即粉色部分。其次, 将得到的中间结果矩阵 (粉色) 的每一行使用 Softmax 激活函数进行归一化操作, 得到最终的权重矩阵, 也就是图中的红色矩阵。红色矩阵中的每一行都对应一个注意力分布。最后, 按行对  $V$  进行加权求和, 便得到了每个单词通过点乘注意力计算得到的表示。这里面, 主要的计算消耗是两次矩阵乘法, 即  $Q$  与  $K^T$  的乘法、相关性矩阵和  $V$  的乘法。这两个操作都可以在 GPU 上高效地完成,

因此可以一次性计算出序列中所有单词之间的注意力权重，并完成所有位置表示的加权求和过程，这样大大提高了模型计算的并行度。

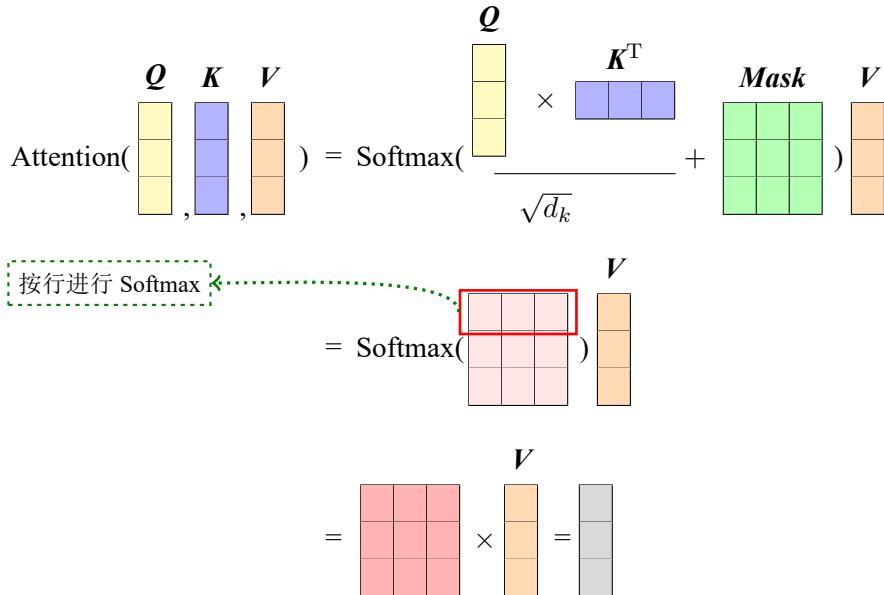


图 12.11 公式(12.9)的执行过程示例

## 12.4.2 多头注意力机制

Transformer 中使用的另一项重要技术是**多头注意力机制** (Multi-head Attention)。“多头”可以理解成将原来的  $Q$ 、 $K$ 、 $V$  按照隐层维度平均切分成多份。假设切分  $h$  份，那么最终会得到  $Q = \{Q_1, \dots, Q_h\}$ ， $K = \{K_1, \dots, K_h\}$ ， $V = \{V_1, \dots, V_h\}$ 。多头注意力就是用每一个切分得到的  $Q$ 、 $K$ 、 $V$  独立的进行注意力计算，即第  $i$  个头的注意力计算结果  $head_i = \text{Attention}(Q_i, K_i, V_i)$ 。

下面根据图12.12详细介绍多头注意力的计算过程：

- 首先，将  $Q$ 、 $K$ 、 $V$  分别通过线性 (Linear) 变换的方式映射为  $h$  个子集。即  $Q_i = QW_i^Q$ 、 $K_i = KW_i^K$ 、 $V_i = VW_i^V$ ，其中  $i$  表示第  $i$  个头， $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ， $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ， $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$  是参数矩阵； $d_k = d_v = d_{\text{model}}/h$ ，对于不同的头采用不同的变换矩阵，这里  $d_{\text{model}}$  表示每个隐层向量的维度；
- 其次，对每个头分别执行点乘注意力操作，并得到每个头的注意力操作的输出  $head_i$ ；
- 最后，将  $h$  个头的注意力输出在最后一维  $d_v$  进行拼接 (Concat) 重新得到维度为  $hd_v$  的输出，并通过对其右乘一个权重矩阵  $W^o$  进行线性变换，从而对多头计算得到的信息进行融合，且将多头注意力输出的维度映射为模型的隐层大小 (即  $d_{\text{model}}$ )，这里参数矩阵  $W^o \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ 。

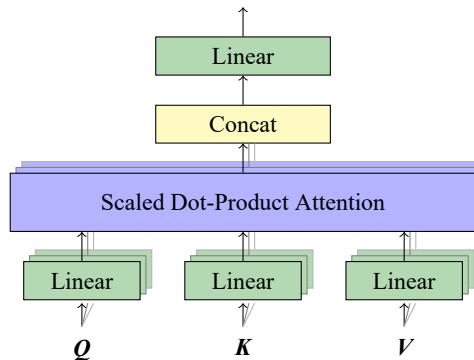


图 12.12 多头注意力模型

多头注意力机制可以被形式化描述为公式(12.10)和(12.11):

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{head}_1, \dots, \mathbf{head}_h) \mathbf{W}^o \quad (12.10)$$

$$\mathbf{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (12.11)$$

多头注意力机制的好处是允许模型在不同的表示子空间里学习。在很多实验中发现，不同表示空间的头捕获的信息是不同的，比如，在使用 Transformer 处理自然语言时，有的头可以捕捉句法信息，有的头可以捕捉词法信息。

### 12.4.3 掩码操作

在公式(12.9)中提到了**掩码** (Mask)，它的目的是对向量中某些值进行掩盖，避免无关位置的数值对运算造成影响。Transformer 中的掩码主要应用在注意力机制中的相关性系数计算，具体方式是在相关性系数矩阵上累加一个掩码矩阵。该矩阵在需要掩码的位置的值为负无穷  $-\infty$  (具体实现时是一个非常小的数，比如  $-1e9$ )，其余位置为 0，这样在进行了 Softmax 归一化操作之后，被掩码掉的位置计算得到的权重便近似为 0，也就是说对无用信息分配的权重为 0，从而避免了其对结果产生影响。Transformer 包含两种掩码：

- **句长补全掩码** (Padding Mask)。在批量处理多个样本时 (训练或解码)，由于要对源语言和目标语言的输入进行批次化处理，而每个批次内序列的长度不一样，为了方便对批次内序列进行矩阵表示，需要进行对齐操作，即在较短的序列后面填充 0 来占位 (padding 操作)。而这些填充的位置没有意义，不参与注意力机制的计算，因此，需要进行掩码操作，屏蔽其影响。
- **未来信息掩码** (Future Mask)。对于解码器来说，由于在预测的时候是自左向右进行的，即第  $t$  时刻解码器的输出只能依赖于  $t$  时刻之前的输出。且为了保证训练解码一致，避免在训练过程中观测到目标语言端每个位置未来的信息，因此需要对未来信息进行屏蔽。具体的做法是：构造一个上三角值全为  $-\infty$  的 Mask

矩阵，也就是说，在解码器计算中，在当前位置，通过未来信息掩码把序列之后的信息屏蔽掉了，避免了  $t$  时刻之后的位置对当前的计算产生影响。图12.13给出了一个具体的实例。

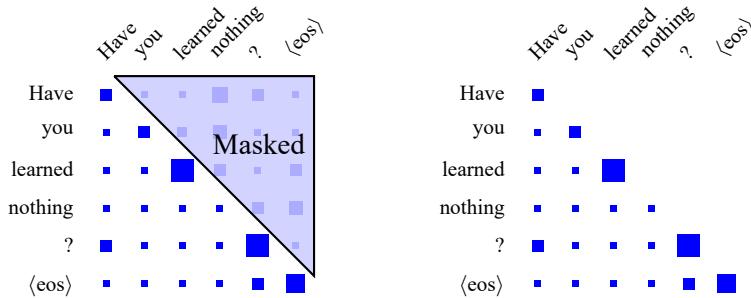


图 12.13 Transformer 中对于未来位置进行的屏蔽的掩码实例

## 12.5 残差网络和层标准化

Transformer 编码器、解码器分别由多层网络组成（通常为 6 层），每层网络又包含多个子层（自注意力网络、前馈神经网络）。因此 Transformer 实际上是一个很深的网络结构。再加上点乘注意力机制中包含很多线性和非线性变换；且注意力函数  $\text{Attention}(\cdot)$  的计算也涉及多层网络，整个网络的信息传递非常复杂。从反向传播的角度来看，每次回传的梯度都会经过若干步骤，容易产生梯度爆炸或者消失。解决这个问题的一种办法就是使用残差连接<sup>[421]</sup>，此部分内容已经在第九章进行了介绍，这里不再赘述。

在 Transformer 的训练过程中，由于引入了残差操作，将前面所有层的输出加到一起，如下：

$$\mathbf{x}^{l+1} = F(\mathbf{x}^l) + \mathbf{x}^l \quad (12.12)$$

其中  $\mathbf{x}^l$  表示第  $l$  层网络的输入向量， $F(\mathbf{x}^l)$  是子层运算，这样会导致不同层（或子层）的结果之间的差异性很大，造成训练过程不稳定、训练时间较长。为了避免这种情况，在每层中加入了层标准化操作<sup>[420]</sup>。图12.14 中的红色方框展示了 Transformer 中残差和层标准化的位置。层标准化的计算如下：

$$\text{LN}(\mathbf{x}) = g \cdot \frac{\mathbf{x} - \mu}{\sigma} + b \quad (12.13)$$

该公式使用均值  $\mu$  和方差  $\sigma$  对样本进行平移缩放，将数据规范化为均值为 0，方差为 1 的标准分布。 $g$  和  $b$  是可学习的参数。

在 Transformer 中经常使用的层标准化操作有两种结构，分别是**后标准化**（Post-

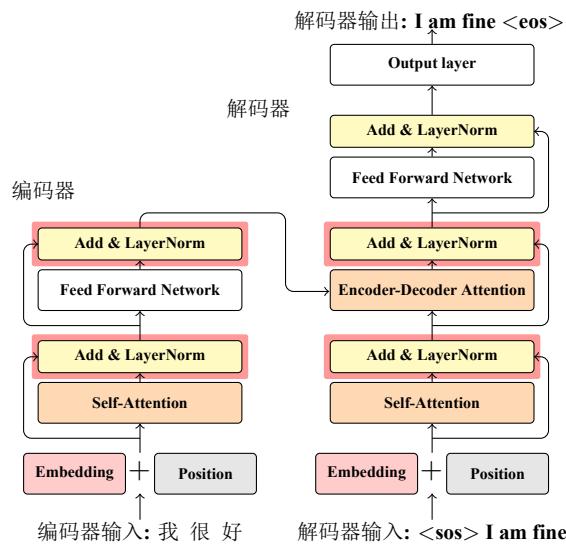


图 12.14 残差和层标准化在模型中的位置

norm) 和 **前标准化** (Pre-norm)，结构如图12.15所示。后标准化中先进行残差连接再进行层标准化，而前标准化则是在子层输入之前进行层标准化操作。在很多实践中已经发现，前标准化的方式更有利于信息传递，因此适合训练深层的 Transformer 模型<sup>[461]</sup>。

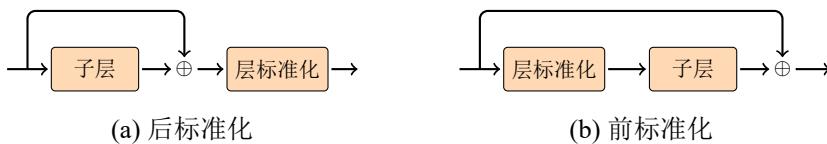


图 12.15 不同标准化方式

## 12.6 前馈全连接网络子层

在 Transformer 的结构中，每一个编码层或者解码层中都包含一个前馈神经网络，它在模型中的位置如图12.16中红色方框所示。

Transformer 使用了全连接网络。全连接网络的作用主要体现在将经过注意力操作之后的表示映射到新的空间中，新的空间会有利于接下来的非线性变换等操作。实验证明，去掉全连接网络会对模型的性能造成很大影响。Transformer 的全连接前馈神经网络包含两次线性变换和一次非线性变换 (ReLU 激活函数:  $\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})$ )，每层的前馈神经网络参数不共享，具体计算如下：

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \quad (12.14)$$

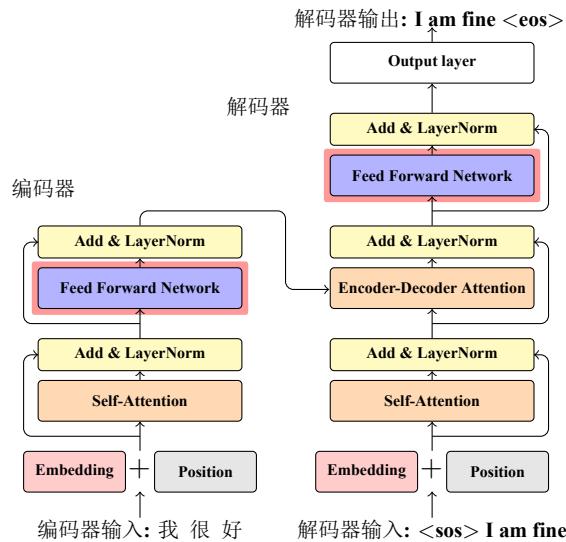


图 12.16 前馈神经网络在模型中的位置

其中， $\mathbf{W}_1$ 、 $\mathbf{W}_2$ 、 $\mathbf{b}_1$  和  $\mathbf{b}_2$  为模型的参数。通常情况下，前馈神经网络的隐层维度要比注意力部分的隐层维度大，而且研究人员发现这种设置对 Transformer 是至关重要的。比如，注意力部分的隐层维度为 512，前馈神经网络部分的隐层维度为 2048。当然，继续增大前馈神经网络的隐层大小，比如设为 4096，甚至 8192，还可以带来性能的增益，但是前馈部分的存储消耗较大，需要更大规模 GPU 设备的支持。因此在具体实现时，往往需要在翻译准确性和存储/速度之间找到一个平衡。

## 12.7 训练

与前面介绍的神经机器翻译模型的训练一样，Transformer 的训练流程为：首先对模型进行初始化，然后在编码器输入包含结束符的源语言单词序列。前面已经介绍过，解码器每个位置单词的预测都要依赖已经生成的序列。在解码器输入包含起始符号的目标语言序列，通过起始符号预测目标语言的第一个单词，用真实的目标语言的第一个单词去预测第二个单词，以此类推，然后用真实的目标语言序列和预测的结果比较，计算它的损失。Transformer 使用了交叉熵损失函数，损失越小说明模型的预测越接近真实输出。然后利用反向传播来调整模型中的参数。由于 Transformer 将任意时刻输入信息之间的距离拉近为 1，摒弃了 RNN 中每一个时刻的计算都要基于前一时刻的计算这种具有时序性的训练方式，因此 Transformer 中训练的不同位置可以并行化训练，大大提高了训练效率。

需要注意的是，Transformer 也包含很多工程方面的技巧。首先，在训练优化器方面，需要注意以下几点：

- Transformer 使用 Adam 优化器优化参数，并设置  $\beta_1 = 0.9$ ,  $\beta_2 = 0.98$ ,  $\epsilon = 10^{-9}$ 。

- Transformer 在学习率中同样应用了学习率预热（Warmup）策略，其计算如下：

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup\_steps}^{-1.5}) \quad (12.15)$$

其中， $\text{step}$  表示更新的次数（或步数）。通常设置网络更新的前 4000 步为预热阶段即  $\text{warmup\_steps} = 4000$ 。Transformer 的学习率曲线如图12.17所示。在训练初期，学习率从一个较小的初始值逐渐增大（线性增长），当到达一定的步数，学习率再逐渐减小。这样做可以减缓在训练初期的不稳定现象，同时在模型达到相对稳定之后，通过逐渐减小的学习率让模型进行更细致的调整。这种学习率的调整方法是 Transformer 系统一个很大的工程贡献。

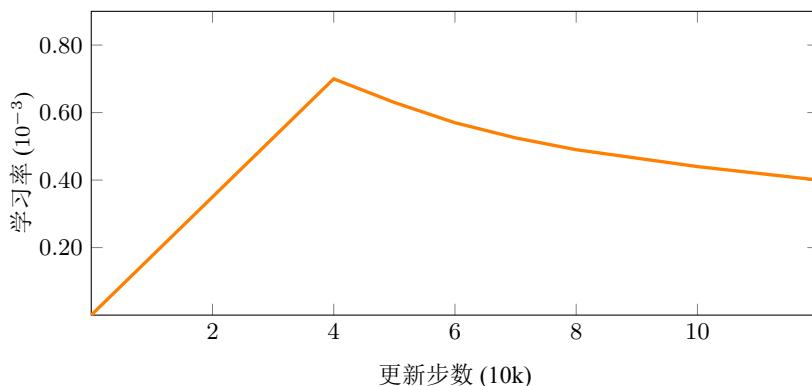


图 12.17 Transformer 模型的学习率曲线

另外，Transformer 为了提高模型训练的效率和性能，还进行了以下几方面的操作：

- 小批量训练**（Mini-batch Training）：每次使用一定数量的样本进行训练，即每次从样本中选择一小部分数据进行训练。这种方法的收敛较快，同时易于提高设备的利用率。批次大小通常设置为 2048 或 4096（token 数即每个批次中的单词个数）。每一个批次中的句子并不是随机选择的，模型通常会根据句子长度进行排序，选取长度相近的句子组成一个批次。这样做可以减少 padding 数量，提高训练效率，如图12.18。



图 12.18 不同批次生成方法对比（白色部分为 padding）

- **Dropout**<sup>[512]</sup>: 由于 Transformer 模型网络结构较为复杂，会导致过度拟合训练数据，从而对未见数据的预测结果变差。这种现象也被称作过拟合。为了避免这种现象，Transformer 加入了 Dropout 操作。Transformer 中这四个地方用到了 Dropout: 词嵌入和位置编码、残差连接、注意力操作和前馈神经网络。Dropout 比例通常设置为 0.1。
- **标签平滑 (Label Smoothing)**<sup>[534]</sup>: 在计算损失的过程中，需要用预测概率去拟合真实概率。在分类任务中，往往使用 One-hot 向量代表真实概率，即真实答案位置那一维对应的概率为 1，其余维为 0，而拟合这种概率分布会造成两个问题：1) 无法保证模型的泛化能力，容易造成过拟合；2) 1 和 0 概率鼓励所属类别和其他类别之间的差距尽可能加大，会造成模型过于相信预测的类别。因此 Transformer 里引入标签平滑来缓解这种现象，简单的说就是给正确答案以外的类别分配一定的概率，而不是采用非 0 即 1 的概率。这样，可以学习一个比较平滑的概率分布，从而提升泛化能力。

不同的 Transformer 可以适应不同的任务，常见的 Transformer 模型有 Transformer Base、Transformer Big 和 Transformer Deep<sup>[23, 461]</sup>，具体设置如下：

- Transformer Base: 标准的 Transformer 结构，解码器编码器均包含 6 层，隐层维度为 512，前馈神经网络维度为 2048，多头注意力机制为 8 头，Dropout 设为 0.1。
- Transformer Big: 为了提升网络的容量，使用更宽的网络。在 Base 的基础上增大隐层维度至 1024，前馈神经网络的维度变为 4096，多头注意力机制为 16 头，Dropout 设为 0.3。
- Transformer Deep: 加深编码器网络层数可以进一步提升网络的性能，它的参数设置与 Transformer Base 基本一致，但是层数增加到 48 层，同时使用 Pre-Norm 作为层标准化的结构。

在 WMT’16 数据上的实验对比如表12.3所示。可以看出，Transformer Base 的 BLEU 得分虽不如另外两种模型，但其参数量是最少的。而 Transformer Deep 的性能整体好于 Transformer Big。

表 12.3 三种 Transformer 模型的对比

系统	BLEU[%]		模型参数量
	EN-DE	EN-FR	
Transformer Base (6 层)	27.3	38.1	$65 \times 10^6$
Transformer Big (6 层)	28.4	41.8	$213 \times 10^6$
Transformer Deep (48 层)	30.2	43.1	$194 \times 10^6$

## 12.8 推断

Transformer 解码器生成译文词序列的过程和其它神经机器翻译系统类似，都是从左往右生成，且下一个单词的预测依赖已经生成的单词。其具体推断过程如图12.19所示，其中  $C_i$  是编码-解码注意力的结果，解码器首先根据“<sos>”和  $C_1$  生成第一个单词“how”，然后根据“how”和  $C_2$  生成第二个单词“are”，以此类推，当解码器生成“<eos>”时结束推断。

但是，Transformer 在推断阶段无法对所有位置进行并行化操作，因为对于每一个目标语言单词都需要对前面所有单词进行注意力操作，因此它推断速度非常慢。可以采用的加速手段有：Cache（缓存需要重复计算的变量）<sup>[535]</sup>、低精度计算<sup>[536, 537]</sup>、共享注意力网络等<sup>[538]</sup>。关于 Transformer 模型的推断加速方法将会在第十四章进一步深入讨论。

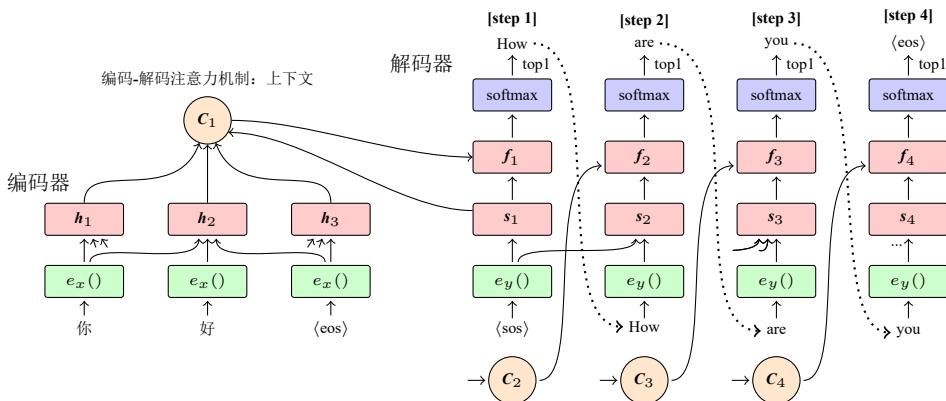


图 12.19 Transformer 模型的推断过程示例

## 12.9 小结及拓展阅读

编码器-解码器框架提供了一个非常灵活的机制，因为开发者只需要设计编码器和解码器的结构就能完成机器翻译。但是，架构的设计是深度学习中最具挑战的工作，优秀的架构往往需要长时间的探索和大量的实验验证，而且还需要一点点“灵感”。前面介绍的基于循环神经网络的翻译模型和注意力机制就是研究人员通过长期的实践发现的神经网络架构。本章介绍了一个全新的模型——Transformer，同时对很多优秀的技术进行了介绍。除了基础知识，关于自注意力机制和模型结构还有很多值得讨论的地方：

- 近两年，有研究已经发现注意力机制可以捕捉一些语言现象<sup>[539]</sup>，比如，在 Transformer 的多头注意力机制中，不同头往往会捕捉到不同的信息，比如，有些头对低频词更加敏感，有些头更适合词意消歧，甚至有些头可以捕捉句法信息。此外，由于注意力机制增加了模型的复杂性，而且随着网络层数的增多，神经

机器翻译中也存在大量的冗余，因此研发轻量的注意力模型也是具有实践意义的方向<sup>[538, 540, 541, 542, 543]</sup>。

- 神经机器翻译依赖成本较高的 GPU 设备，因此对模型的裁剪和加速也是很多系统研发人员所感兴趣的方向。比如，从工程上，可以考虑减少运算强度，比如使用低精度浮点数<sup>[544]</sup> 或者整数<sup>[537, 545]</sup> 进行计算，或者引入缓存机制来加速模型的推断<sup>[535]</sup>；也可以通过对模型参数矩阵的剪枝来减小整个模型的体积<sup>[546]</sup>；另一种方法是知识蒸馏<sup>[547, 548]</sup>。利用大模型训练小模型，这样往往可以得到比单独训练小模型更好的效果<sup>[549]</sup>。
- 自注意力网络作为 Transformer 模型中重要组成部分，近年来受到研究人员的广泛关注，尝试设计更高效地操作来替代它。比如，利用动态卷积网络来替换编码器与解码器的自注意力网络，在保证推断效率的同时取得了和 Transformer 相当甚至略好的翻译性能<sup>[507]</sup>；为了加速 Transformer 处理较长输入文本的效率，利用局部敏感哈希替换自注意力机制的 Reformer 模型也吸引了广泛的关注<sup>[543]</sup>。此外，在自注意力网络引入额外的编码信息能够进一步提高模型的表示能力。比如，引入固定窗口大小的相对位置编码信息<sup>[460, 550]</sup>，或利用动态系统的思想从数据中学习特定的位置编码表示，具有更好的泛化能力<sup>[551]</sup>。通过对 Transformer 模型中各层输出进行可视化分析，研究人员发现 Transformer 自底向上各层网络依次聚焦于词级-语法级-语义级的表示<sup>[462, 552]</sup>，因此在底层的自注意力网络中引入局部编码信息有助于模型对局部特征的抽象<sup>[553, 554]</sup>。
- 除了针对 Transformer 中子层的优化，网络各层之间的连接方式在一定程度上也能影响模型的表示能力。近年来针对网络连接优化的工作如下：在编码器顶部利用平均池化或权重累加等融合手段得到编码器各层的全局表示<sup>[555, 556, 557, 558]</sup>，利用之前各层表示来生成当前层的输入表示<sup>[461, 463, 559]</sup>。



# 机器翻译前沿

13.6	基于样本价值的学习	
13.7	小结及拓展阅读	
<b>14</b>	<b>神经机器翻译模型推断</b>	<b>447</b>
14.1	面临的挑战	
14.2	基本问题	
14.3	轻量模型	
14.4	非自回归翻译	
14.5	多模型集成	
14.6	小结与拓展阅读	
<b>15</b>	<b>神经机器翻译结构优化</b>	<b>475</b>
15.1	注意力机制的改进	
15.2	神经网络连接优化及深层模型	
15.3	基于句法的神经机器翻译模型	
15.4	基于结构搜索的翻译模型优化	
15.5	小结及拓展阅读	
<b>16</b>	<b>低资源神经机器翻译</b>	<b>517</b>
16.1	数据的有效使用	
16.2	双向翻译模型	
16.3	多语言翻译模型	
16.4	无监督机器翻译	
16.5	领域适应	
16.6	小结及拓展阅读	
<b>17</b>	<b>多模态、多层次机器翻译</b>	<b>553</b>
17.1	机器翻译需要更多的上下文	
17.2	语音翻译	
17.3	图像翻译	
17.4	篇章级翻译	
17.5	小结及拓展阅读	
<b>18</b>	<b>机器翻译应用技术</b>	<b>577</b>
18.1	机器翻译的应用并不简单	
18.2	增量式模型优化	
18.3	交互式机器翻译	
18.4	翻译结果的可干预性	
18.5	小设备机器翻译	
18.6	机器翻译系统的部署	
18.7	机器翻译的应用场景	
	<b>随笔</b>	<b>588</b>
	<b>后记</b>	<b>593</b>





## 13. 神经机器翻译模型训练

模型训练是机器翻译领域的重要研究方向，其中的很多发现对其它自然语言处理任务也有很好的借鉴意义。特别是，训练神经机器翻译仍然面临一些挑战，包括：

- 如何对大容量模型进行有效的训练？例如，避免过拟合问题，并让模型更加健壮，同时有效地处理更大的词汇表。
- 如何设计更好的模型训练策略？例如，在训练中更好地利用机器翻译评价指标，同时选择对翻译更有价值的样本进行模型训练。
- 如何让模型学习到的“知识”在模型之间迁移？例如，把一个“强”模型的能力迁移到一个“弱”模型上，而这种能力可能是无法通过直接训练“弱”模型得到的。

本章将就这些问题展开讨论，内容会覆盖开放词表、正则化、对抗样本训练、最小风险训练、知识蒸馏等多个主题。需要注意的是，神经机器翻译模型训练涉及的内容十分广泛。很多情况下，模型训练问题会和建模问题强相关。因此，本章的内容主要集中在相对独立的基础模型训练问题上。在后续章节中，仍然会有模型训练方面的介绍，其主要针对机器翻译的特定主题，如深层神经网络训练、无指导训练等。

### 13.1 开放词表

对于神经机器翻译而言，研究人员通常希望使用更大的词表完成模型训练。因为大词表可以覆盖更多的语言现象，使模型对不同的语言现象有更强的区分能力。但

是，人类的语言表达方式是十分多样的，这也体现在单词的构成上，甚至人们都无法想象数据中存在的不同单词的数量。比如，在 WMT、CCMT 等评测数据上，英语词表大小都会在 100 万以上。如果不加限制，机器翻译的词表将会很“大”。这也会导致模型参数量变大，模型训练变得极为困难。更严重的问题是，测试数据中的一些单词根本就没有在训练数据中出现过，这时会出现未登录词翻译问题（即 OOV 问题），即系统无法对未见单词进行翻译。在神经机器翻译中，通常会考虑使用更小的翻译单元来缓解数据稀疏问题。

### 13.1.1 大词表和未登录词问题

首先来具体看一看神经机器翻译的大词表问题。神经机器翻译模型训练和推断都依赖于源语言和目标语言的词表（见第十章）。在建模中，词表中的每一个单词都会被转换为分布式（向量）表示，即词嵌入。如果每个单词都对应一个向量，那么单词的各种变形（时态、语态等）都会导致词表增大，同时增加词嵌入表示的难度。如果要覆盖更多的翻译现象，词表会不断膨胀，并带来两个问题：

- **数据稀疏。**很多不常见的低频词包含在词表中，而这些低频词的词嵌入表示很难得到充分学习。
- **参数及计算量的增大。**大词表会增加词嵌入矩阵的大小，同时也会显著增加输出层中线性变换和 Softmax 的计算量。

理想情况下，机器翻译应该是一个**开放词表**（Open Vocabulary）的翻译任务。也就是，无论测试数据中包含什么样的词，机器翻译系统都应该能够正常翻译。但是，现实的情况是即使不断扩充词表，也不可能覆盖所有可能的单词。这个问题在使用受限词表时会更加严重，因为低频词和未见过的词都会被看作未登录词。这时会将这些单词用符号 <UNK> 代替。通常，数据中 <UNK> 的数量会直接影响翻译性能，过多的 <UNK> 会造成欠翻译、结构混乱等问题。因此神经机器翻译需要额外的机制解决大词表和未登录词问题。

### 13.1.2 子词

一种解决开放词表翻译问题的思路是改造输出层结构<sup>[466, 560]</sup>，比如，替换原始的 Softmax 层，用更加高效的神经网络结构进行超大规模词表上的预测。不过，模型结构和训练方法的调整使得系统开发与调试的工作量增加，并且这类方法仍然无法解决未登录词问题，因此在实用系统中并不常用。

另一种思路是不改变机器翻译系统，而是从数据处理的角度来缓解未登录词问题。既然使用单词会带来数据稀疏问题，那么自然会想到使用更小的单元，通过更小的单元的多种排列组合来表示更多的单词。比如，把字符作为最小的翻译单元<sup>1</sup>——也就是基于字符的翻译模型<sup>[561]</sup>。以英语为例，只需要构造一个包含 26 个英语

<sup>1</sup>汉语里的字符可以被看作是汉字。

字母、数字和一些特殊符号的字符表，便可以表示所有的单词。

但是字符级翻译也面临着新的问题——使用字符增加了系统捕捉不同语言单元之间搭配的难度。假设平均一个单词由 5 个字符组成，系统所处理的序列长度便增大 5 倍。这使得具有独立意义的不同语言单元需要跨越更远的距离才能产生联系。此外，基于字符的方法也破坏了单词中天然存在的构词规律，或者说破坏了单词内字符的局部依赖。比如，英语单词“telephone”中的“tele”和“phone”都是有具体意义的词缀，但是如果把它们打散为字符就失去了这些含义。

那么有没有一种方式能够兼顾基于单词和基于字符方法的优点呢？常用的手段包括两种，一种是采用字词融合的方式构建词表，将未知单词转换为字符的序列并通过特殊的标记将其与普通的单词区分开来<sup>[562]</sup>。而另一种方式是将单词切分为子词（Sub-word），它是介于单词和字符中间的一种语言单元表示形式。比如，将英语单词“doing”切分为“do” + “ing”。对于形态学丰富的语言来说，子词体现了一种具有独立意义的构词基本单元。如图13.1，子词“do”和“new”可以用于组成其他不同形态的单词。



图 13.1 不同单词共享相同的子词（前缀）

在一些极端的情况下，子词仍然可以包含所有的字母和数字。这样，理论上，所有的单词都可以用子词进行组装。当然，理想的状况是：在子词词表不太大的前提下，使用尽可能少的子词单元拼装出每个单词。在神经机器翻译中，基于子词的切分是很常用的数据处理方法，称为子词切分。主要包括三个步骤：

- 对原始数据进行分词操作；
- 构建符号合并表；
- 根据合并表，将字符合并为子词。

这里面的核心是如何构建符号合并表，下面对一些常用方法进行介绍。

### 13.1.3 双字节编码

字节对编码或双字节编码（BPE）是一种常用的子词词表构建方法。BPE 方法最早用于数据压缩，该方法将数据中常见的连续字符串替换为一个不存在的字符，之后通过构建一个替换关系的对应表，对压缩后的数据进行还原<sup>[563]</sup>。机器翻译借用了这种思想，把子词切分看作是学习对自然语言句子进行压缩编码表示的问题<sup>[89]</sup>。其目的是，保证编码（即子词切分）后的结果占用的字节尽可能少。这样，子词单元会尽可能被不同单词复用，同时又不会因为使用过小的单元造成子词切分后的序列过长。

使用 BPE 算法进行子词切分包含两个步骤。首先，通过统计的方法构造符号合并表（见图13.2），具体的方式为：先对分过词的文本进行统计，得到词表和词频，同时将词表中的单词分割为字符表示；其次统计词表中所有出现的二元组的频次，选择当前频次最高的二元组加入符号合并表，并将所有词表中出现的该二元组合并为一个单元；不断地重复上述过程，直到合并表的大小达到预先设定的大小，或者无法继续合并。图13.3给出了一个使用字符合并表对单词进行子词切分的实例。红色单元为每次合并后得到的新符号，直至无法合并，或遍历结束，得到最终的合并结果。其中每一个单元为一个子词。

使用 BPE 方法后，翻译模型的输出也是子词序列，因此需要对最终得到的翻译结果进行子词还原，即将子词形式表达的单元重新组合为原本的单词。这一步操作也十分简单，只需要不断的将每个子词向后合并，直至遇到表示单词边界的终结符，便得到了一个完整的单词。

使用 BPE 方法的策略有很多。不仅可以单独对源语言和目标语言句子进行子词的切分，也可以联合两种语言，共同进行子词切分，被称作**双字节联合编码**（Joint-BPE）<sup>[89]</sup>。相比于单语 BPE，Joint-BPE 可以增加两种语言子词切分的一致性。对于相似语系中的语言，如英语和德语，常使用 Joint-BPE 的方法联合构建词表。而对于汉语和英语这些差异比较大的语种，则需要独立地进行子词切分。

BPE 还有很多变种方法。比如，可以设计更合理的符号合并优先级。这种方法的出发点在于，在不考虑优先级的情况下，在对一个单词用同一个合并表切分子词时，可能存在多种结果。如 hello，可以被切分为“hell”和“o”，也可以被切分为“h”和“ello”。这种切分方式的多样性可以用来提高神经机器翻译系统的健壮性<sup>[564]</sup>。此外，尽管 BPE 也被命名为双字节编码，但是在实践中该方法一般处理的是 Unicode 编码，而不是字节。相应的，在预训练模型 GPT2 中也探索了字节级别的 BPE，这种方法在机器翻译、自动问答等任务中取得了很好的效果<sup>[409]</sup>。

### 13.1.4 其他方法

与基于统计的 BPE 算法不同，基于 Word Piece 的子词切分方法利用语言模型进行子词词表的构造<sup>[565]</sup>。本质上，基于语言模型的方法和基于 BPE 的方法的思路相同，即通过合并字符和子词不断生成新的子词。它们的区别在于合并子词的方式，基于 BPE 的方法选择出现频次最高的连续字符进行合并，而基于语言模型的方法则是根据语言模型输出的概率选择要合并哪些子词。具体来说，基于 Word Piece 的方法首先将句子切割为字符表示的形式<sup>[565]</sup>，并利用该数据训练一个 1-gram 语言模型，记为  $\log P(\cdot)$ 。假设两个相邻的子词单元  $a$  和  $b$  被合并为新的子词  $c$ ，则整个句子的语言模型得分的变化为  $\Delta = \log P(c) - \log P(a) - \log P(b)$ 。这样，可以不断的选择使  $\Delta$  最大的两个子词单元进行合并，直到达到预设的词表大小或者句子概率的增量低于某个阈值。

目前比较主流的子词切分方法都是作用于分词后的序列，对一些没有明显词边

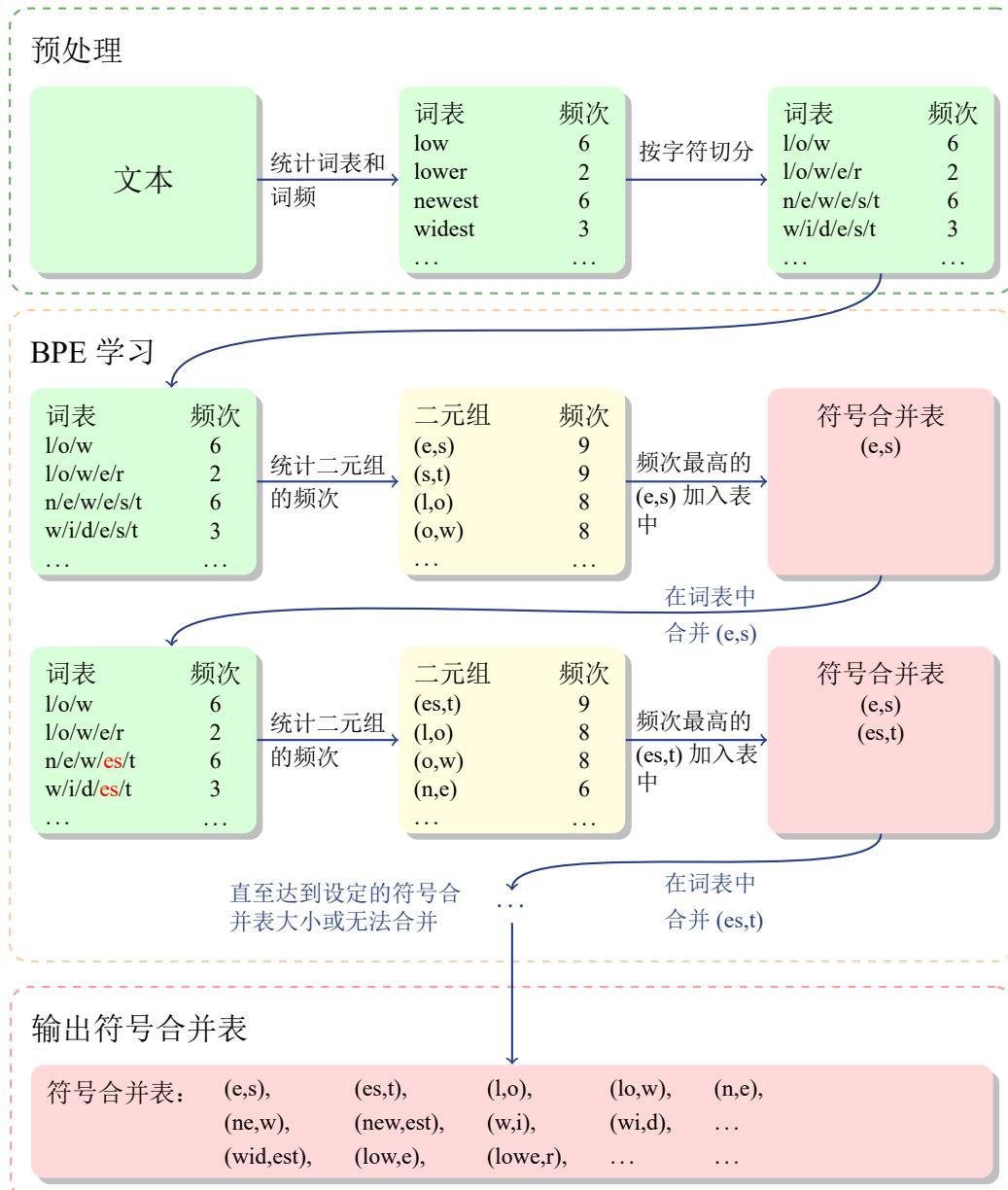


图 13.2 BPE 算法中符号合并表的生成过程

符号合并表					
(r,<e>),	(e,s),	(l,o),	(es,t),	(lo,w),	(est,<e>),

(a) 符号合并表

样例 1: low e r <e> —————→ low er <e>

样例 2: low e st <e> → low es t <e> → lo w es t <e> → lo w est <e> → low est <e> → low est <e>

(b) 合并样例

图 13.3 BPE 中的子词切分过程

界且资源稀缺的语种并不友好。相比之下，Sentence Piece 方法可以作用于未经过分词处理的输入序列<sup>[566]</sup>，同时囊括了双字节编码和语言模型的子词切分方法，更加灵活易用。

不过在以 BPE 为代表的子词切分方法中，每个单词都对应一种唯一的子词切分方式，因此输入的数据经过子词切分后的序列表示也是唯一的。一旦切分出现错误，整句话的翻译效果可能会变得很差。为此，研究人员提出一些规范化方法<sup>[564, 567]</sup>。

- **子词规范化方法**<sup>[564]</sup>。其做法是根据 1-gram 语言模型采样出多种子词切分候选。之后，最大化整个句子的概率为目标来构建词表。
- **BPE-Dropout**<sup>[567]</sup>。在训练时，按照一定概率  $p$  随机丢弃一些可行的合并操作，从而产生不同的子词切分结果。而在推断阶段，将  $p$  设置为 0，等同于标准的 BPE。总的来说，上述方法相当于在子词的粒度上对输入的序列进行扰动，进而达到增加训练健壮性的目的。
- **动态规划编码** (Dynamic Programming Encoding, DPE)<sup>[568]</sup>。引入了混合字符-子词的切分方式，将句子的子词切分看作一种隐含变量。机器翻译解码端的输入是基于字符表示的目标语言序列，推断时将每个时间步的输出映射到预先设定好的子词词表之上，得到当前最可能的子词结果。

## 13.2 正则化

正则化是机器学习中的经典技术，通常用于缓解过拟合问题。正则化的概念源自线性代数和代数几何。在实践中，它更多的是指对**反问题** (The Inverse Problem) 的一种求解方式。假设输入  $x$  和输出  $y$  之间存在一种映射  $f$ :

$$y = f(x) \quad (13.1)$$

反问题是指：当观测到  $y$  时，能否求出  $x$ 。反问题对应了很多实际问题，比如，可以把  $y$  看作经过美化的图片， $x$  看作原始的图片，反问题就对应了图片还原。机器翻译的训练也是一种反问题，因为可以把  $y$  看作是正确的译文， $x$  看作是输入句子或者模型参数<sup>2</sup>。

理想的情况下，研究人员希望反问题的解是**适定的**（Well-posed）。所谓适定解，需要满足三个条件：解是存在的、解是唯一的、解是稳定的（即  $y$  微小的变化会导致  $x$  微小的变化，也被称作解连续）。所有不存在唯一稳定解的问题都被称作**不适定问题**（Ill-posed Problem）。对于机器学习问题，解的存在性比较容易理解。解的唯一性大多由问题决定。比如，如果把描述问题的函数  $f(\cdot)$  看作一个  $n \times n$  矩阵  $A$ ， $x$  和  $y$  都看作是  $n$  维向量。那么  $x$  不唯一的原因在于  $A$  不满秩（非奇异矩阵）。不过，存在性和唯一性并不会对机器学习方法造成太大困扰，因为在实践中往往会找到近似的解。但是，解的稳定性却给神经机器翻译带来了很大的挑战。因为神经机器翻译模型非常复杂，里面存在大量的矩阵乘法和非线性变换。这导致  $f(\cdot)$  往往是不稳定的，也就是说，神经机器翻译中输出  $y$  的微小变化会导致输入  $x$  的巨大变化。比如，在系统研发中经常会发现，即使训练样本发生很小的变化，模型训练得到的参数都会有非常明显的变化。不仅如此，在神经机器翻译模型中，稳定性训练还面临两方面问题：

- **观测数据不充分**。由于语言表达的多样性，训练样本只能覆盖非常有限的翻译现象。从样本的表示空间上看，对于没有观测样本的区域，根本无法知道真实解的样子，因此也很难描述这些样本的性质，更不用说稳定性训练了。
- **数据中存在噪声**。噪声问题是稳定性训练最大的挑战之一。因为，即使是很小的噪声，也可能会导致解的巨大变化。

以上问题体现出来的现象就是过拟合。因为训练数据有限且存在噪声，因此模型参数会过分拟合噪声数据。而且，这样的模型参数又与真实（理想）的模型参数相差很远。正则化正是针对这个问题。有时候，正则化也被称作**降噪**（Denoising），虽然它的出发点并不只是去除噪声的影响。图13.4对比了不同函数对二维空间中一些数据点的拟合情况。在过拟合现象中，函数可以完美的拟合所有的数据点，即使有些数据点是噪声。

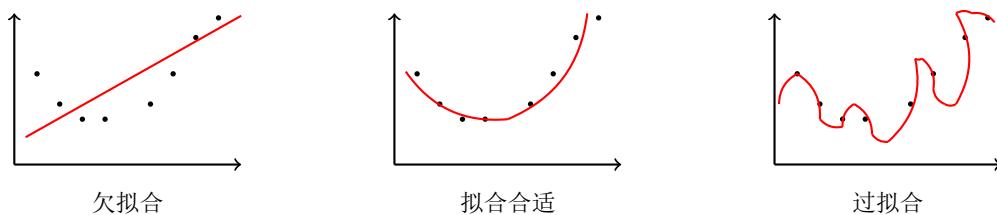


图 13.4 欠拟合 vs 过拟合

<sup>2</sup>在训练中，如果把源语言句子看作是不变的量，这时函数  $f(\cdot)$  的输入只有模型参数。

正则化的一种实现是在训练目标中引入一个正则项。在神经机器翻译中，引入正则项的训练目标为：

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \text{Loss}(\mathbf{w}) + \lambda R(\mathbf{w}) \quad (13.2)$$

其中， $\mathbf{w}$ 是模型参数， $\text{Loss}(\mathbf{w})$ 是损失函数， $R(\mathbf{w})$ 是正则项， $\lambda$ 是正则项的系数，用于控制正则化对训练影响的程度。 $R(\mathbf{w})$ 通常也可以被看作是一种先验，因为在数据不充分且存在噪声的情况下，可以根据一些先验知识让模型偏向正确的方向一些，而不是一味地根据受噪声影响的 $\text{Loss}(\mathbf{w})$ 进行优化。相应的，引入正则化后的模型可以获得更好的**泛化**（Generalization）能力，即模型在新的未见数据上表现会更好。

实践中已经证明，正则化方法有助于使得像神经机器翻译模型这样复杂模型获得稳定的模型参数。甚至有些情况下，如果不引入正则化，训练得到的翻译模型根本无法使用。此外，正则化方法不仅可以用于提高模型的泛化能力，也可以作为干预模型学习的一种手段，比如，可以将一些先验知识作为正则项约束机器翻译模型的学习。类似的手段在本书后续的内容中也会被使用。

### 13.2.1 L1/L2 正则化

L1/L2 正则化是常用的正则化方法，虽然这种方法并不仅针对机器翻译模型。L1/L2 正则化分别对应正则项是  $l_1$  和  $l_2$  范数的情况。具体来说，L1 正则化是指：

$$\begin{aligned} R(\mathbf{w}) &= \|\mathbf{w}\|_1 \\ &= \sum_{w_i} |w_i| \end{aligned} \quad (13.3)$$

L2 正则化是指

$$\begin{aligned} R(\mathbf{w}) &= (\|\mathbf{w}\|_2)^2 \\ &= \sum_{w_i} {w_i}^2 \end{aligned} \quad (13.4)$$

第九章已经介绍了 L1 和 L2 正则化方法，这里做一些展开。从几何的角度看，L1 和 L2 正则项都是有物理意义的。二者都可以被看作是空间上的一个区域，比如，在二维平面上， $l_1$  范数表示一个以 0 点为中心的菱形， $l_2$  范数表示一个以 0 点为中心的圆。此时， $L(\mathbf{w})$  和  $R(\mathbf{w})$  叠加在一起构成了一个新的区域，优化问题可以被看作是在这个新的区域上进行优化。由于 L1 和 L2 正则项都是在 0 点（坐标原点）附近形成的区域，因此优化的过程可以确保参数不会偏离 0 点太多。也就是说，**L1 和 L2 正则项引入了一个先验：模型的解不应该离 0 点太远**。而 L1 和 L2 正则项实际上是在度量这个距离。

那为什么要用 L1 和 L2 正则项惩罚离 0 点远的解呢？这还要从模型复杂度谈起。

实际上，对于神经机器翻译这样的模型来说，模型的容量是足够的。所谓容量可以被简单的理解为独立参数的个数<sup>3</sup>。也就是说，理论上存在一种模型可以完美的描述问题。但是，从目标函数拟合的角度来看，如果一个模型可以拟合很复杂的目标函数，那模型所表示的函数形态也会很复杂。这往往体现在模型中参数的值“偏大”。比如，用一个多项式函数拟合一些空间中的点，如果希望拟合得很好，各个项的系数往往是非零的。而且为了对每个点进行拟合，通常需要多项式中的某些项具有较大的系数，以期望函数在局部有较大的斜率。显然，这样的模型是很复杂的。模型的复杂度可以用函数中参数（比如多项式中各项的系数）的“值”进行度量，这也体现在模型参数的范数上。

因此，L1 和 L2 正则项的目的是防止模型为了匹配少数（噪声）样本而导致模型参数的值过大。反过来说，L1 和 L2 正则项会鼓励那些参数值在 0 点附近的情况。从实践的角度看，这种方法可以很好的对统计模型的训练进行校正，得到泛化能力更强的模型。

### 13.2.2 标签平滑

神经机器翻译在每个目标语言位置  $j$  会输出一个分布  $\hat{\mathbf{y}}_j$ ，这个分布描述了每个目标语言单词出现的可能性。在训练时，每个目标语言位置上的答案是一个单词，也就对应了 One-hot 分布  $\mathbf{y}_j$ ，它仅仅在正确答案那一维为 1，其它维均为 0。模型训练可以被看作是一个调整模型参数让  $\hat{\mathbf{y}}_j$  逼近  $\mathbf{y}_j$  的过程。但是， $\mathbf{y}_j$  的每一个维度是一个非 0 即 1 的目标，这样也就无法考虑类别之间的相关性。具体来说，除非模型在答案那一维输出 1，否则都会得到惩罚。即使模型把一部分概率分配给与答案相近的单词（比如同义词），这个相近的单词仍被视为完全错误的预测。

标签平滑的思想很简单<sup>[534]</sup>：答案所对应的单词不应该“独享”所有的概率，其它单词应该有机会作为答案。这个观点与第二章中语言模型的平滑非常类似。在复杂模型的参数估计中，往往需要给未见或者低频事件分配一些概率，以保证模型具有更好的泛化能力。具体实现时，标签平滑使用了一个额外的分布  $\mathbf{q}$ ，它是在词汇表  $V$  上的一个均匀分布，即  $\mathbf{q}_k = \frac{1}{|V|}$ ，其中  $\mathbf{q}_k$  表示分布的第  $k$  维。然后，标准答案的分布被重新定义为  $\mathbf{y}_j$  和  $\mathbf{q}$  的线性插值：

$$\mathbf{y}_j^{ls} = (1 - \alpha) \cdot \mathbf{y}_j + \alpha \cdot \mathbf{q} \quad (13.5)$$

这里， $\alpha$  表示一个系数，用于控制分布  $\mathbf{q}$  的重要性， $\mathbf{y}_j^{ls}$  表示使用标签平滑后的学习目标。

标签平滑实际上定义了一种“软”标签，使得所有标签都可以分到一些概率。一方面可以缓解数据中噪声的影响，另一方面目标分布会更合理（显然，真实的分布不应该是 One-hot 分布）。图13.5展示了标签平滑前后的损失函数计算结果的对比。

<sup>3</sup>另一种定义是把容量看作神经网络所能表示的假设空间大小<sup>[569]</sup>，也就是神经网络能表示的不同函数所构成的空间。

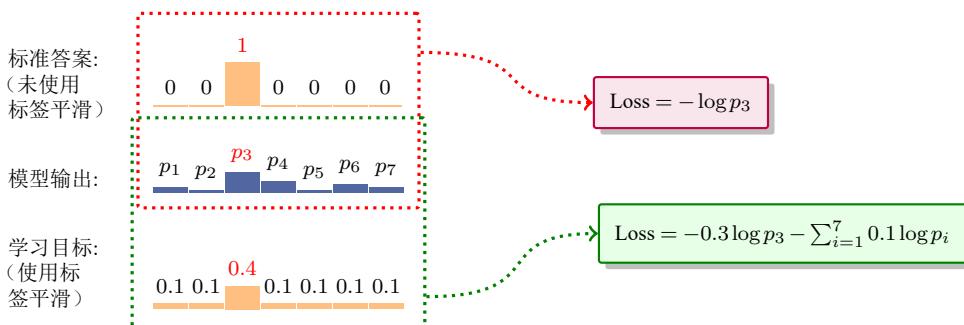


图 13.5 不使用标签平滑 vs 使用标签平滑

标签平滑也可以被看作是对损失函数的一种调整，并引入了额外的先验知识（即与  $q$  相关的部分）。只不过这种先验知识并不是通过公式(13.2)所示的线性插值方式与原始损失函数进行融合。

### 13.2.3 Dropout

神经机器翻译模型是一种典型的多层神经网络模型。每一层都包含若干神经元，负责接收前一层所有神经元的输出，之后进行诸如乘法、加法等变换操作，并有选择地使用非线性的激活函数，最终得到当前层每个神经元的输出。从模型最终预测的角度看，每个神经元都在参与最终的预测。理想的情况下，研究人员希望每个神经元都能相互独立的做出“贡献”。这样的模型会更加健壮，因为即使一部分神经元不能正常工作，其它神经元仍然可以独立做出合理的预测。但是，随着每一层神经元数量的增加以及网络结构的复杂化，神经元之间会出现相互适应（Co-adaptation）的现象。所谓相互适应是指，一个神经元对输出的贡献与同一层其它神经元的行为是相关的，也就是说这个神经元已经适应到它周围的“环境”中。

相互适应的好处在于神经网络可以处理更加复杂的问题，因为联合使用两个神经元要比单独使用每个神经元的表示能力强。这也类似于传统机器学习任务中往往设计一些高阶特征，比如自然语言序列标注中对 2-gram 和 3-gram 的使用。不过另一方面，相互适应会导致模型变得更加“脆弱”。因为相互适应的神经元可以更好的描述训练数据中的现象，但是在测试数据上，由于很多现象是未见的，细微的扰动会导致神经元无法适应。具体体现出来就是过拟合问题。

Dropout 也是解决过拟合问题的一种常用方法<sup>[570]</sup>。该方法很简单，在训练时随机让一部分神经元停止工作，这样每次参数更新中每个神经元周围的环境都在变化，它就不会过分适应到环境中。图13.6中给出了某一次参数更新中使用 Dropout 之前和之后神经网络的状态对比。

具体实现时，可以设置一个参数  $p \in (0, 1)$ 。在每次参数更新所使用的前向和反向计算中，每个神经元都以概率  $p$  停止工作。相当于每层神经网络会有以  $p$  为概率的神经元被“屏蔽”掉。每一次参数更新中会随机屏蔽不同的神经元，图13.7给出了

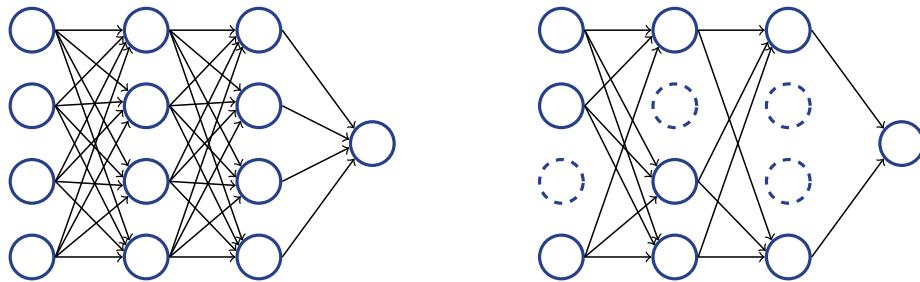


图 13.6 使用 Dropout 之前（左）和之后（右）神经网络状态的对比

Dropout 方法和传统方法计算方式的对比。其中， $x_i^l$  代表第  $l$  层神经网络的第  $i$  个输入， $w_i^l$  为输入所对应的权重， $b^l$  表示第  $l$  层神经网络输入的偏置， $z_i^{l+1}$  表示第  $l$  层神经网络的线性运算的结果， $f(\cdot)$  表示激活函数， $r_i^l$  的值服从于参数为  $1 - p$  的伯努利分布。

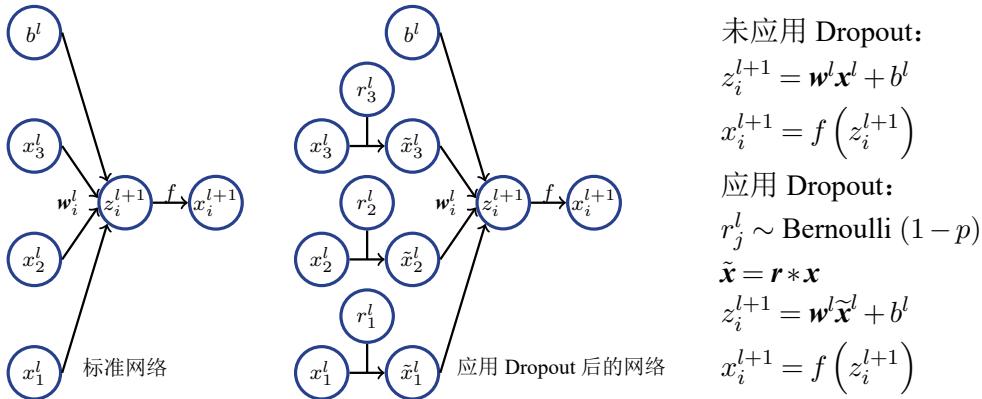


图 13.7 使用 Dropout 之前（左）和之后（右）一层神经网络

对于新的样本，可以使用 Dropout 训练之后的模型对其进行推断，但是每个神经元的输出要乘以  $1 - p$ ，以保证每层神经元输出的期望和训练时是一样的。另一种常用的做法是，在训练时对每个神经元的输出乘以  $\frac{1}{1-p}$ ，然后在推断时神经网络可以不经过任何调整就直接使用。

Dropout 方法的另一种解释是，在训练中屏蔽掉一些神经元相当于从原始的神经网络中抽取出了一个子网络。这样，每次训练都在一个随机生成的子网络上进行，而不同子网络之间的参数是共享的。在推断时，则把所有的子网络集成到一起。这种思想也有一些**集成学习**（Ensemble Learning）的味道，只不过 Dropout 中子模型（或子网络）是在指数级空间中采样出来的。由于 Dropout 可以很好的缓解复杂神经网络模型的过拟合问题，因此也成为了大多数神经机器翻译系统的标配。

随着网络层数的增多，相互适应也会出现在不同层之间，甚至会出现在多头注意力机制的不同头之间。因此，Dropout 方法也可以用于对模型局部结构的屏蔽，比如，对多层神经网络中的层进行屏蔽，即 Layer Dropout。特别是对于深层神经网络，

Layer Dropout 也是一种有效的防止过拟合的方法。关于 Layer Dropout 的内容在第十五章还会有详细讨论。

### 13.3 对抗样本训练

同其它基于神经网络的方法一样，提高**健壮性**（Robustness）也是神经机器翻译研发中需要关注的。比如，大容量模型可以很好地拟合训练数据，但是当测试样本与训练样本差异较大时，会导致很糟糕的翻译结果<sup>[512, 571]</sup>。另一方面，实践中也发现，有些情况下即使输入中有微小的扰动，神经网络模型的输出也会产生巨大变化。或者说，神经网络模型在输入样本上容易受到攻击（Attack）<sup>[572, 573, 574]</sup>。表13.1展示了一个神经机器翻译系统的翻译结果，可以看到，把输入句子中的单词“jumped”换成“sunk”会得到完全不同的译文。这时神经机器翻译系统就存在健壮性问题。

表 13.1 神经机器翻译实例

原始输入	When shot at, the dove <b>jumped</b> into the bushes
原始输出	当鸽子被射中时，它跳进了灌木丛
扰动的输入	When shot at, the dove <b>sunk</b> into the bushes
扰动的输出	当有人开枪射击时，那只鸽子陷进了灌木丛中

决定神经网络**模型健壮性**的因素主要包括训练数据、模型结构、正则化方法等。仅仅从模型的角度来改善健壮性一般是较为困难的，因为如果输入数据是“干净”的，模型就会学习如何在这样的数据上进行预测。无论模型的能力是强还是弱，当推断时的输入数据出现扰动的时候，模型可能就无法适应这种它从未见过的新数据。因此，一种简单直接的方法是从训练样本出发，让模型在学习的过程中能对样本中的扰动进行处理，进而在推断时更加健壮。具体来说，可以在训练过程中构造有噪声的样本，即基于**对抗样本**（Adversarial Examples）进行**对抗训练**（Adversarial Training）。

#### 13.3.1 对抗样本及对抗攻击

在图像识别领域，研究人员就发现，对于输入图像的细小扰动，如像素变化等，会使模型以高置信度给出错误的预测<sup>[575, 576, 577]</sup>，但是这种扰动并不会造成人类的错误判断。也就是说，样本中的微小变化“欺骗”了图像识别系统，但是“欺骗”不了人类。这种现象背后的原因有很多，一种可能的原因是：系统并没有理解图像，而是在拟合数据，因此拟合能力越强，反而对数据中的微小变化更加敏感。从统计学习的角度看，既然新的数据中可能会有扰动，那更好的学习方式就是在训练中显性地把这种扰动建模出来，让模型对输入的细微变化表现得更加健壮。

这种通过在原样本上增加一些难以察觉的扰动，从而使模型得到错误判断的样本被称为对抗样本。对于模型的输入  $\mathbf{x}$  和输出  $\mathbf{y}$ ，对抗样本形式上可以被描述为：

$$C(\mathbf{x}) = \mathbf{y} \quad (13.6)$$

$$C(\mathbf{x}') \neq \mathbf{y} \quad (13.7)$$

$$\text{s.t. } \Psi(\mathbf{x}, \mathbf{x}') < \varepsilon \quad (13.8)$$

其中， $(\mathbf{x}', \mathbf{y})$  为输入中含有扰动的对抗样本，函数  $C(\cdot)$  为模型。公式(13.8)中  $\Psi(\mathbf{x}, \mathbf{x}')$  表示扰动后的输入  $\mathbf{x}'$  和原输入  $\mathbf{x}$  之间的距离， $\varepsilon$  表示扰动的受限范围。当模型对包含噪声的数据容易给出较差的结果时，往往意味着该模型的抗干扰能力差，因此可以利用对抗样本检测现有模型的健壮性<sup>[578]</sup>。同时，采用类似数据增强的方式将对抗样本混合至训练数据中，能够使模型得到稳定的预测能力，这种方式也被称为对抗训练<sup>[577, 579, 580]</sup>。

通过对抗样本训练来提升模型健壮性的首要问题是：如何生成对抗样本。通过当前模型  $C$  和样本  $(\mathbf{x}, \mathbf{y})$ ，生成对抗样本的过程被称为**对抗攻击**（Adversarial Attack）。对抗攻击可以被分为黑盒攻击和白盒攻击。在白盒攻击中，攻击算法可以访问模型的完整信息，包括模型结构、网络参数、损失函数、激活函数、输入和输出数据等。黑盒攻击通常依赖启发式方法来生成对抗样本<sup>[578]</sup>，由于这种攻击方式不需要知道神经网络的详细信息，仅仅通过访问模型的输入和输出就可以达到攻击的目的。并且由于神经网络其本身便是一个黑盒模型，因此在神经网络的相关应用中黑盒攻击方法更加实用。

在神经机器翻译中，训练数据中含有的细微扰动会使得模型比较脆弱<sup>[581]</sup>。研究人员希望借鉴图像任务中的一些对抗攻击方法，并将其应用于自然语言处理任务中。然而，对计算机而言，以像素值等表示的图像数据本身就是连续的<sup>[582]</sup>，而文本中的一个个单词本身离散的，这种图像与文本数据间的差异使得这些方法在自然语言处理上并不适用。比如图像任务中对一幅图片的局部图像进行替换的方法，如果用于自然语言处理中，可能会生成语法错误或者语义错误的句子。而且，简单替换单词产生的扰动过大，模型很容易判别。即使对词嵌入等连续表示的部分进行扰动，也会产生无法与词嵌入空间中的任何词匹配的向量<sup>[583]</sup>。针对这些问题，下面着重介绍神经机器翻译任务中如何有效生成、使用对抗样本。

### 13.3.2 基于黑盒攻击的方法

一个好的对抗样本应该具有这种性质：对文本做最少的修改，并最大程度地保留原文的语义。一种简单的实现方式是对文本加噪声。这里，噪声可以分为自然噪声和人工噪声<sup>[581]</sup>。自然噪声一般是指在语料库中自然出现的错误，如输入错误、拼写错误等。人为噪声是通过人工设计的自动方法修改文本，例如，可以通过规则或是噪声生成器，在干净的数据中以一定的概率引入拼写错误、语法错误等<sup>[584, 585, 586]</sup>；此

外，也可以在文本中加入人为设计过的毫无意义的单词序列。

除了单纯的在文本中引入各种扰动外，还可以通过文本编辑的方式，在不改变语义的情况下尽可能修改文本，从而构建对抗样本<sup>[587, 588]</sup>。文本的编辑方式主要包括交换、插入、替换和删除操作。表13.2给出了通过这几种方式生成对抗样本的例子。

表 13.2 对抗样本实例

原始输入	We are really looking forward to the holiday
替换操作	We are really looking forward to the vacation
插入操作	We are really looking forward to the holiday tomorrow
删除操作	We are really looking forward to the holiday
交换操作	We are really forward looking to the holiday

形式上，可以利用如 FGSM 等算法<sup>[577]</sup>，验证文本中每一个单词的贡献度，同时为每一个单词构建一个候选池，包括该单词的近义词、拼写错误词、同音词等。对于贡献度较低的词，如语气词、副词等，可以使用插入、删除操作进行扰动。对于其他的单词，可以在候选池中选择相应的单词并进行替换。其中，交换操作可以是基于词级别的，比如交换序列中的单词，也可以是基于字符级别的，例如交换单词中的字符<sup>[589]</sup>。重复进行上述的编辑操作，直至编辑出的文本可以误导模型做出错误的判断。

在机器翻译中，常用的回译技术也是生成对抗样本的一种有效方式。回译就是，通过反向模型将目标语言翻译成源语言，并将翻译得到的双语数据用于模型训练（见第十六章）。除了翻译模型，语言模型也可以用于生成对抗样本。第二章已经介绍过，语言模型可以用于检测句子的流畅度，它根据上文预测当前位置可能出现的单词。因此，此时可以使用语言模型预测出当前位置最可能出现的多个单词，并用这些词替换序列中原本的单词。在机器翻译任务中，可以通过与神经机器翻译系统联合训练，共享词向量矩阵的方式得到语言模型<sup>[590]</sup>。

此外，生成对抗网络（Generative Adversarial Networks, GANs）也可以被用来生成对抗样本<sup>[591]</sup>。与回译方法类似，基于生成对抗网络的方法将原始的输入映射为潜在分布  $P$ ，并在其中搜索出服从相同分布的文本构成对抗样本。一些研究也对这种方法进行了优化<sup>[591]</sup>，在稠密的向量空间中进行搜索，也就是说在定义  $P$  的基础稠密向量空间中找到对抗性表示  $z'$ ，然后利用生成模型将其映射回  $x'$ ，使最终生成的对抗样本在语义上接近原始输入。

### 13.3.3 基于白盒攻击的方法

除了在单词级别增加扰动以外，还可以在模型内部增加扰动。一种简单的方法是在每一个词的词嵌入上，累加一个正态分布的变量，之后将其作为模型的最终输入。同时，可以在训练阶段增加额外的训练目标。比如，迫使模型在接收到被扰动的

输入后，编码器能够生成与正常输入类似的表示，解码器输出正确的翻译结果<sup>[592]</sup>。

还可以根据机器翻译的具体问题增加扰动。例如，针对同音字错误问题，可以将单词的发音转换为一个包含  $n$  个发音单元的发音序列，如音素，音节等，并训练相应的嵌入矩阵将每一个发音单元转换为对应的向量表示。对发音序列中发音单元的嵌入表示进行平均后，得到当前单词的发音表示。最后将词嵌入与单词的发音表示进行加权求和，并将结果作为模型的输入<sup>[593]</sup>。通过这种方式可以提高模型对同音异形词的处理能力。除了在词嵌入层增加扰动，也可以在编码器输出中引入额外的噪声，能起到与在层输入中增加扰动相类似的效果<sup>[489]</sup>。

此外，对于训练样本  $(\mathbf{x}, \mathbf{y})$ ，还可以使用基于梯度的方法来生成对抗样本  $(\mathbf{x}', \mathbf{y}')$ 。例如，可以利用替换词与原始单词词向量之间的差值，以及候选词的梯度之间的相似度来生成对抗样本<sup>[574]</sup>。以源语言为例，生成  $\mathbf{x}'$  中第  $i$  个词的过程可以被描述如下：

$$x'_i = \arg \max_{x \in V} \text{sim}(e(x) - e(x_i), \mathbf{g}_{x_i}) \quad (13.9)$$

$$\mathbf{g}_{x_i} = \nabla_{e(x_i)} - \log P(\mathbf{y}|\mathbf{x}; \theta) \quad (13.10)$$

其中， $x_i$  为输入序列中的第  $i$  个词， $e(\cdot)$  用于获取词向量， $\mathbf{g}_{x_i}$  为翻译概率相对于  $e(x_i)$  的梯度， $\text{sim}(\cdot, \cdot)$  是用于评估两个向量之间相似度（距离）的函数， $V$  为源语言的词表。由于对词表中所有单词进行枚举时，计算成本较大。因此可以利用语言模型选择最可能的  $n$  个词作为候选，并从中采样出单词完成替换。同时，为了保护模型不受解码器预测误差的影响，此时需要对模型目标语言端的输入做出同样的调整。与源语言端的操作不同，此时会将公式(13.10)中的损失替换为  $-\log P(\mathbf{y}|\mathbf{x}')$ ，即使用生成的对抗样本  $\mathbf{x}'$  计算翻译概率。

在进行对抗性训练时，可以在原有的训练损失上增加三个额外的损失，最终的损失函数被定义为：

$$\begin{aligned} \text{Loss}(\theta_{\text{mt}}, \theta_{\text{lm}}^{\mathbf{x}}, \theta_{\text{lm}}^{\mathbf{y}}) &= \text{Loss}_{\text{clean}}(\theta_{\text{mt}}) + \text{Loss}_{\text{lm}}(\theta_{\text{lm}}^{\mathbf{x}}) + \\ &\quad \text{Loss}_{\text{robust}}(\theta_{\text{mt}}) + \text{Loss}_{\text{lm}}(\theta_{\text{lm}}^{\mathbf{y}}) \end{aligned} \quad (13.11)$$

其中， $\text{Loss}_{\text{clean}}(\theta_{\text{mt}})$  为正常情况下的损失， $\text{Loss}_{\text{lm}}(\theta_{\text{lm}}^{\mathbf{x}})$  和  $\text{Loss}_{\text{lm}}(\theta_{\text{lm}}^{\mathbf{y}})$  为生成对抗样本所用到的源语言与目标语言的模型的损失， $\text{Loss}_{\text{robust}}(\theta_{\text{mt}})$  是使用修改后得到的对抗样本作为输入，并以原始的译文  $\mathbf{y}$  作为答案时计算得到的损失。假设有  $N$  个样本，则损失函数的具体形式如下：

$$\text{Loss}_{\text{robust}}(\theta_{\text{mt}}) = \frac{1}{N} \sum_{(\mathbf{x}, \mathbf{y})} -\log P(\mathbf{y}|\mathbf{x}', \mathbf{y}'; \theta_{\text{mt}}) \quad (13.12)$$

无论是黑盒方法还是白盒方法，本质上都是通过增加噪声使得模型训练更加健壮。类似的思想在很多机器学习方法中都有体现，比如，在最大熵模型中使用高斯

噪声就是常用的增加模型健壮性的手段之一<sup>[594]</sup>。从噪声信道模型的角度看（见第五章），翻译过程也可以被理解为一种加噪和去噪的过程，不论这种噪声是天然存在于数据中的，还是人为添加的。除了对抗样本训练，机器翻译所使用的降噪自编码方法和基于重构的损失函数<sup>[595, 596]</sup>，也都体现了类似的思想。广义上，这些方法也可以被看作是利用“加噪 + 去噪”进行健壮性训练的方法。

## 13.4 学习策略

尽管极大似然估计在神经机器翻译中取得了巨大的成功，但仍然面临着许多问题。比如，似然函数并不是评价翻译系统性能的指标，这使得即使在训练数据上优化似然函数，但在应用模型时并不一定可以获得更好的翻译结果。本节首先会对极大似然估计的问题进行论述，然后介绍一些解决相关问题的方法。

### 13.4.1 极大似然估计的问题

极大似然估计已成为机器翻译乃至整个自然语言处理领域中使用最广泛的训练用目标函数。但是，使用极大似然估存在**曝光偏置**（Exposure Bias）问题和训练-推断评价指标不一致问题，具体体现在如下两个方面。

- **曝光偏置问题**。在训练过程中，模型使用标注数据进行训练，因此模型在预测下一个单词时，解码器的输入是正确的译文片段。也就是，预测第  $j$  个单词时，系统使用了标准答案  $\{y_1, \dots, y_{j-1}\}$  作为历史信息。但是对新的句子进行翻译时，预测第  $j$  个单词时使用的是模型自己生成的前  $j-1$  个单词，即  $\{\hat{y}_1, \dots, \hat{y}_{j-1}\}$ 。这意味着，训练时使用的输入数据（目标语言端）与真实翻译时的情况不符，如图13.8所示。由于在训练过程中暴露于标注数据，因此模型也适应了标注数据，在推断阶段无法很好地适应模型自动生成的数据，这就是**曝光偏置问题**<sup>[597, 598]</sup>。

- **训练目标函数与任务评价指标不一致问题**。在训练数据上使用极大似然估计，而在新数据上进行推断的时候，通常使用 BLEU 等外部评价指标来评价模型的性能。在机器翻译任务中，这个问题的一种体现是，训练数据上更低的困惑度不一定能带来 BLEU 的提升。更加理想的情况是，模型应该直接最大化性能评价指标，而不是训练集数据上的似然函数<sup>[233]</sup>。但是很多模型性能评价指标不可微分，这使得研究人员无法直接利用基于梯度的方法来优化这些指标。

### 13.4.2 非 Teacher-forcing 方法

所谓 Teacher-forcing 方法，即要求模型预测的结果和标准答案完全对应。Teacher-forcing 是一种深度学习中的训练策略，在序列处理任务上被广泛使用<sup>[569]</sup>。以序列生成任务为例，Teacher-forcing 要求模型在训练时不是使用上一个时刻的模型输出作为下一个时刻的输入，而是使用训练数据中上一时刻的标准答案作为下一个时刻的输

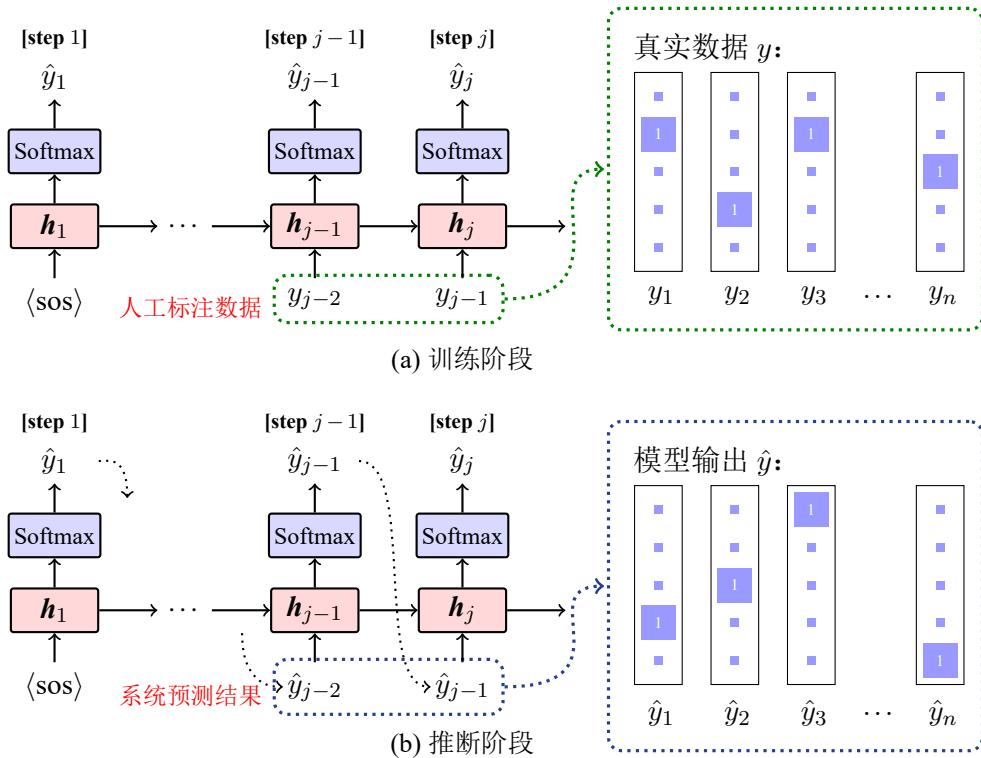


图 13.8 曝光偏置问题（基于循环神经网络的翻译模型）

入。显然，这会导致曝光偏置问题。为了解决这个问题，可以使用非 Teacher-forcing 方法。比如，在训练中使用束搜索，这样可以让训练过程模拟推断时的行为。具体来说，非 Teacher-forcing 方法可以用调度采样和生成对抗网络进行实现。

## 1. 调度采样

对于一个目标语言序列  $y = \{y_1, \dots, y_n\}$ ，在预测第  $j$  个单词时，训练过程与推断过程之间的主要区别在于：训练过程中使用标准答案  $\{y_1, \dots, y_{j-1}\}$ ，而推断过程使用的是来自模型本身的预测结果  $\{\hat{y}_1, \dots, \hat{y}_{j-1}\}$ 。此时可以采取一种**调度采样**（Scheduled Sampling）机制<sup>[597]</sup>。以基于循环神经网络的模型为例，在训练中预测第  $j$  个单词时，随机决定使用  $y_{j-1}$  还是  $\hat{y}_{j-1}$  作为输入。假设训练时使用的是基于小批量的随机梯度下降方法，在第  $i$  个批次中，对序列每一个位置进行预测时以概率  $\epsilon_i$  使用标准答案  $y_{j-1}$ ，或以概率  $1 - \epsilon_i$  使用来自模型本身的预测  $\hat{y}_{j-1}$ 。具体到序列中的一个位置  $j$ ，可以根据模型单词预测的概率进行采样，在  $\epsilon_i$  控制的调度策略下，同  $y_{j-1}$  一起作为输入。此过程如图13.9所示，并且这个过程可以很好地与束搜索融合。

当  $\epsilon_i = 1$  时，模型的训练与原始的训练策略完全相同，而当  $\epsilon_i = 0$  时，模型的训练则与推断时使用的策略完全一样。在这里使用到了一种**课程学习**（Curriculum Learning）策略<sup>[599]</sup>，该策略认为学习应该循序渐进，从一种状态逐渐过渡到另一种状

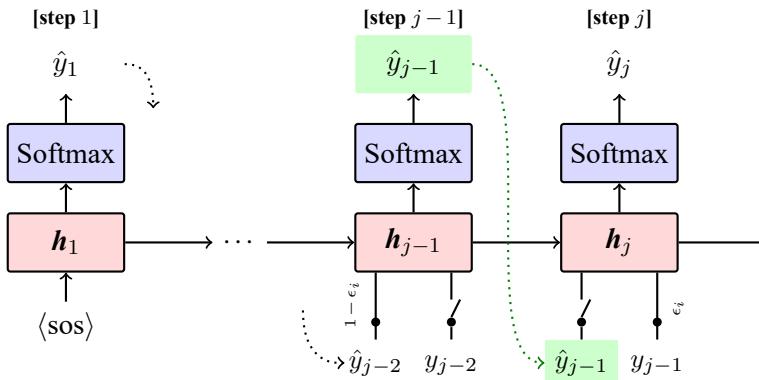


图 13.9 调度采样方法的示意图

态。在训练开始时,由于模型训练不充分,因此如果使用模型预测结果作为输入,会导致收敛速度非常慢。因此,在模型训练的前期,通常会选择使用标准答案  $\{y_1, \dots, y_{j-1}\}$ 。在模型训练的后期,应该更倾向于使用自模型本身的预测  $\{\hat{y}_1, \dots, \hat{y}_{j-1}\}$ 。关于课程学习的内容在 13.6.2 节还会有详细介绍。

在使用调度策略时,需要调整关于训练批次  $i$  的函数来降低  $\epsilon_i$ ,与梯度下降方法中降低学习率的方式相似。调度策略可以采用如下几种方式:

- **线性衰减。**  $\epsilon_i = \max(\epsilon, k - ci)$ , 其中  $\epsilon$  ( $0 \leq \epsilon < 1$ ) 是  $\epsilon_i$  的最小数值,而  $k$  和  $c$  代表衰减的偏移量和斜率,取决于预期的收敛速度。
- **指数衰减。**  $\epsilon_i = k^i$ , 其中  $k$  是一个常数,一般为  $k < 1$ 。
- **反向 Sigmoid 衰减。**  $\epsilon_i = k/(k + \exp(i/k))$ , 其中  $k \geq 1$ 。

## 2. 生成对抗网络

调度采样解决曝光偏置的方法是,把模型前  $j-1$  步的预测结果作为输入,来预测第  $j$  步的输出。但是,如果模型预测的结果中有错误,再使用错误的结果预测未来的序列也会产生问题。解决这个问题就需要知道模型预测的好与坏,并在训练中有效的使用它们。**如果生成好的结果,那么可以使用它进行模型训练,否则就不使用。**生成对抗网络就是这样一种技术,它引入了一个额外的模型(判别器)来对原有模型(生成器)的生成结果进行评价,并根据评价结果同时训练两个模型。

在 13.3 小节已经提到了生成对抗网络,这里稍微进行一些展开。在机器翻译中,基于对抗神经网络的架构被命名为**对抗神经机器翻译**(Adversarial-NMT)<sup>[600]</sup>。这里,令  $(x, y)$  表示一个训练样本,令  $\hat{y}$  表示神经机器翻译系统对源语言句子  $x$  的翻译结果。此时,对抗神经机器翻译的总体框架可以表示为图 13.10,其中。绿色部分表示神经机器翻译模型  $G$ ,该模型将源语言句子  $x$  翻译为目标语言句子  $\hat{y}$ 。红色部分是对抗网络  $D$ ,它的作用是判断目标语言句子是否是源语言句子  $x$  的真实翻译。 $G$  和  $D$  相互对抗,用  $G$  生成的翻译结果  $\hat{y}$  来训练  $D$ ,并生成奖励信号,再使用奖励信号

通过策略梯度训练  $G$ 。

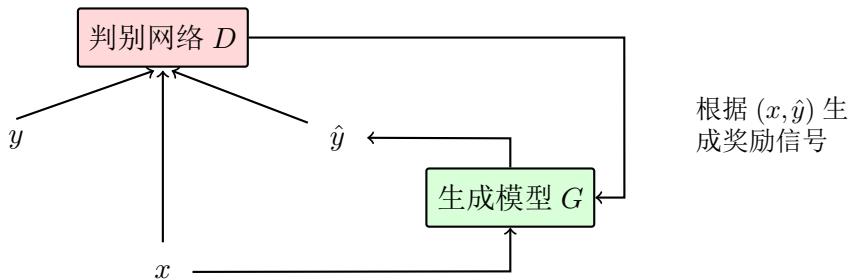


图 13.10 对抗神经机器翻译框架图

实际上，对抗神经机器翻译的训练目标就是强制  $\hat{y}$  与  $y$  相似。在理想情况下， $\hat{y}$  与人类标注的答案  $y$  非常相似，以至于人类也无法分辨  $\hat{y}$  是由机器还是人类产生的。

### 13.4.3 强化学习方法

**强化学习** (Reinforcement Learning, RL) 方法是机器学习中的经典方法，它可以同时解决 13.4.1 节提到的曝光偏置问题和训练-推断评价指标不一致问题。本节主要介绍基于策略的方法和基于演员-评论家的方法<sup>[601]</sup>。

#### 1. 基于策略的方法

**最小风险训练** (Minimum Risk Training, MRT) 可以被看作是一种基于策略的方法。与极大似然估计不同，最小风险训练引入了评价指标作为损失函数，并优化模型将预期风险降至最低<sup>[233]</sup>。

最小风险训练的目标是找到模型参数  $\hat{\theta}_{\text{MRT}}$ ，满足如下公式：

$$\hat{\theta}_{\text{MRT}} = \arg \min_{\theta} \{R(\theta)\} \quad (13.13)$$

其中， $R(\theta)$  表示预期风险，通常用风险函数的期望表示。假设有  $N$  个训练样本  $\{(x^{[1]}, y^{[1]}), \dots, (x^{[N]}, y^{[N]})\}$ ， $R(\theta)$  被定义如下：

$$\begin{aligned} R(\theta) &= \sum_{k=1}^N \mathbb{E}_{\hat{y}|x^{[k]};\theta} [\Delta(\hat{y}, y^{[k]})] \\ &= \sum_{k=1}^N \sum_{\hat{y} \in \chi(x^{[k]})} P(\hat{y}|x^{[k]};\theta) \Delta(\hat{y}, y^{[k]}) \end{aligned} \quad (13.14)$$

这里， $\hat{y}$  是模型预测的译文， $\chi(x^{[k]})$  是  $x^{[k]}$  所对应的所有候选翻译的集合。损失函数  $\Delta(\hat{y}, y^{[k]})$  用来衡量模型预测  $\hat{y}$  与标准答案  $y^{[k]}$  间的差异，损失函数一般用翻译质量

评价指标定义，例如，BLEU，TER 等<sup>4</sup>。在最小风险训练中，对模型参数  $\theta$  的偏导数为：

$$\begin{aligned}\frac{\partial R(\theta)}{\partial \theta} &= \sum_{k=1}^N \mathbb{E}_{\hat{y}|x^{[k]};\theta} [\Delta(\hat{y}, y^{[k]}) \times \frac{\partial P(\hat{y}|x^{[k]};\theta) / \partial \theta}{P(\hat{y}|x^{[k]};\theta)}] \\ &= \sum_{k=1}^N \mathbb{E}_{\hat{y}|x^{[k]};\theta} [\Delta(\hat{y}, y^{[k]}) \times \frac{\partial \log P(\hat{y}|x^{[k]};\theta)}{\partial \theta}]\end{aligned}\quad (13.15)$$

公式(13.15)使用了**策略梯度** (Policy Gradient) 的手段将  $\Delta(\hat{y}, y^{[k]})$  提到微分操作之外<sup>[602, 603]</sup>。这样，就无需对  $\Delta(\hat{y}, y^{[k]})$  进行微分，因此最小风险训练允许任意不可微的损失函数，包括 BLEU 等常用的评价函数。同时，等式右侧将对概率的求导操作转化为了对 log 函数的求导，更易于模型进行优化。因此，使用公式(13.15)就可以求出模型参数相对于风险函数的损失，进而进行基于梯度的优化。

这里需要注意的是，公式(13.15)中求期望的过程是无法直接实现的，因为无法遍历所有的译文句子。通常，会使用采样的方法搜集一定数量的译文，来模拟译文空间。例如，可以使用推断系统生成若干译文。同时，为了保证生成的译文之间具有一定的差异性，也可以对推断过程进行一些“干扰”。从实践的角度看，采样方法是影响强化学习系统的重要因素，因此往往需要对不同的任务设计相适应的采样方法。最简单的方法就是在产生译文的每一个词时候，根据模型产生的下一个词的分布随机选取词当作模型预测，直到选到句子结束符或者达到特定长度的时候停止<sup>[604]</sup>。其他方法还包括随机束搜索，它把束搜索中选取 Top- $k$  的操作替换成随机选取  $k$  个词。这个方法不会采集到重复的样本。还可以使用基于 Gumbel-Top- $k$  的随机束搜索更好地控制了样本里的噪声<sup>[605]</sup>。

相比于极大似然估计，最小风险训练有着以下优点：

- 最小风险训练使用模型自身产生的数据进行训练，从而避免了曝光偏置问题。
- 最小风险训练直接优化 BLEU 等评价指标，从而解决了训练-推断评价指标不一致问题。
- 最小风险训练方法不涉及具体的模型结构，可以应用于任意的机器翻译模型。

## 2. 演员-评论家方法

基于策略的强化学习是要寻找一个策略  $p(a|\hat{y}_{1\dots j-1}, x)$ ，使得该策略选择的行动  $a$  未来可以获得的奖励期望最大化，也被称为**动作价值函数** (Action-value Function) 最

---

<sup>4</sup>对于 BLEU，损失函数可以被定义为 1-BLEU。

大化。这个过程通常用函数  $Q$  来描述：

$$\begin{aligned} Q(a; \hat{y}_{1\dots j-1}, y) &= \mathbb{E}_{\hat{y}_{j+1\dots J} \sim p(\cdot | \hat{y}_{1\dots j-1} a, x)} [r_j(a; \hat{y}_{1\dots j-1}, y) + \\ &\quad \sum_{i=j}^J r_i(\hat{y}_i; \hat{y}_{1\dots j-1} a \hat{y}_{j+1\dots i}, y)] \end{aligned} \quad (13.16)$$

其中,  $r_j(a; \hat{y}_{1\dots j-1}, y)$  是  $j$  时刻做出行动  $a$  获得的奖励,  $r_i(\hat{y}_i; \hat{y}_{1\dots j-1} a \hat{y}_{j+1\dots i}, y)$  是在  $j$  时刻的行动为  $a$  的前提下,  $i$  时刻的做出行动  $\hat{y}_i$  获得的奖励,  $\hat{y}_{j+1\dots J} \sim p(\cdot | \hat{y}_{1\dots j-1} a, x)$  表示序列  $\hat{y}_{j+1\dots J}$  是根据  $p(\cdot | \hat{y}_{1\dots j-1} a, x)$  得到的采样结果, 概率函数  $p$  中的  $\cdot$  表示序列  $\hat{y}_{j+1\dots J}$  服从的随机变量,  $x$  是源语言句子,  $y$  是正确译文,  $\hat{y}_{1\dots j-1}$  是策略  $p$  产生的译文的前  $j-1$  个词,  $J$  是生成译文的长度。特别的, 对于公式13.16中  $\hat{y}_{j+1\dots i}$  来说, 如果  $i < j+1$ , 则  $\hat{y}_{j+1\dots i}$  不存在, 对于源语句子  $x$ , 最优策略  $\hat{p}$  可以被定义为:

$$\hat{p} = \arg \max_p \mathbb{E}_{\hat{y} \sim p(\hat{y}|x)} \sum_{j=1}^J \sum_{a \in A} p(a | \hat{y}_{1\dots j}, x) Q(a; \hat{y}_{1\dots j}, y) \quad (13.17)$$

其中,  $A$  表示所有可能的行动组成的空间, 也就是词表  $V$ 。公式(13.17)的含义是, 最优策略  $\hat{p}$  的选择需要同时考虑当前决策的“信心”(即  $p(a | \hat{y}_{1\dots j}, x)$ ) 和未来可以获得的“价值”(即  $Q(a; \hat{y}_{1\dots j}, y)$ )。

计算动作价值函数  $Q$  需要枚举  $j$  时刻以后所有可能的序列, 而可能的序列数目是随着其长度呈指数级增长, 因此只能采用估计的方法计算  $Q$  的值。基于策略的强化学习方法, 如最小风险训练(风险  $\Delta = -Q$ )等都使用了采样的方法来估计  $Q$ 。尽管采样估计的结果是  $Q$  的无偏估计, 但是它的缺点在于估计的方差比较大。而  $Q$  直接关系到梯度更新的大小, 不稳定的数值会导致模型更新不稳定, 难以优化。

为了避免采样的开销和随机性带来的不稳定, 基于**演员-评论家**(Actor-critic)的强化学习方法引入一个可学习的函数  $\tilde{Q}$ , 通过函数  $\tilde{Q}$  来逼近动作价值函数  $Q$ <sup>[601]</sup>。但是由于  $\tilde{Q}$  是人工设计的一个函数, 该函数有着自身的偏置, 因此  $\tilde{Q}$  不是  $Q$  的一个无偏估计, 所以使用  $\tilde{Q}$  来指导  $p$  的优化无法到达理论上的最优解。尽管如此, 得益于神经网络强大的拟合能力, 基于演员-评论家的强化学习方法在实践中仍然非常流行。

在基于演员-评论家的强化学习方法中, 演员就是策略  $p$ , 而评论家就是动作价值函数  $Q$  的估计  $\tilde{Q}$ 。对于演员, 它的目标是找到最优的决策:

$$\hat{p} = \arg \max_p \mathbb{E}_{\hat{y} \sim p(\hat{y}|x)} \sum_{j=1}^J \sum_{a \in A} p(a | \hat{y}_{1\dots j}, x) \tilde{Q}(a; \hat{y}_{1\dots j}, y) \quad (13.18)$$

与公式(13.17)对比可以发现, 基于演员-评论家的强化学习方法与基于策略的强化学习方法类似, 公式(13.18)对动作价值函数  $Q$  的估计变成了一个可学习的函数  $\tilde{Q}$ 。

对于目标函数里期望的计算，通常使用采样的方式来进行逼近，这与最小风险训练也是十分类似的，例如，选择一定量的  $\hat{y}$  来计算期望，而不是遍历所有的  $\hat{y}$ 。借助与最小风险训练类似的方法，可以计算对  $p$  的梯度来优化演员。

而对于评论家，它的优化目标并不是那么显而易见。尽管可以通过采样的方式来估计  $Q$ ，然后使用该估计作为目标让  $\tilde{Q}$  进行拟合，但是这样会导致非常高的（采样）代价。同时可以想象，既然有了一个无偏估计，为什么还要用有偏估计  $\tilde{Q}$  呢？

回顾动作价值函数的定义，可以对它做适当的展开，可以得到如下等式：

$$\begin{aligned} Q(\hat{y}_j; \hat{y}_{1\dots j-1}, y) &= r_j(\hat{y}_j; \hat{y}_{1\dots j-1}, y) + \\ &\quad \sum_{a \in A} p(a|\hat{y}_{1\dots j}, x) Q(a; \hat{y}_{1\dots j}, y) \end{aligned} \quad (13.19)$$

这个等式也被称为**贝尔曼方程** (Bellman Equation)<sup>[606]</sup>。它表达了  $j-1$  时刻的动作价值函数  $Q(\hat{y}_j; \hat{y}_{1\dots j-1}, y)$  跟下一时刻  $j$  的动作价值函数  $Q(a; \hat{y}_{1\dots j}, y)$  之间的关系。在理想情况下，动作价值函数  $Q$  应该满足上述等式，因此可以使用该等式作为可学习的函数  $\tilde{Q}$  的目标。于是，可以定义  $j$  时刻动作价值函数为：

$$q_j = r_j(\hat{y}_j; \hat{y}_{1\dots j-1}, y) + \sum_{a \in A} p(a|\hat{y}_{1\dots j}, x) \tilde{Q}(a; \hat{y}_{1\dots j}, y) \quad (13.20)$$

相应的，评论家对应的目标定义如下：

$$\hat{\tilde{Q}} = \arg \min_{\tilde{Q}} \sum_{j=1}^J (\tilde{Q}(\hat{y}_j; \hat{y}_{1\dots j-1}, y) - q_j)^2 \quad (13.21)$$

此时，公式13.20与公式13.21共同组成了评论家的学习目标，使得可学习的函数  $\tilde{Q}$  逼近理想的  $Q$ 。最后，通过同时优化演员和评论家直到收敛，获得的演员（也就是策略  $p$ ）就是我们期望的翻译模型。图13.11展示了演员和评论家的关系。

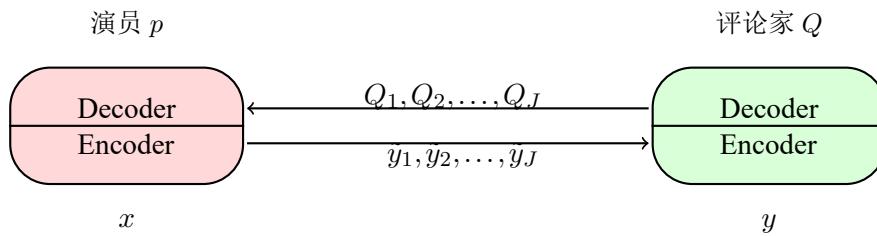


图 13.11 基于演员-评论家的强化学习方法

使用基于演员-评论家的强化学习方法还有许多细节，包括但不限于以下技巧：

- **多目标学习**。演员的优化通常会引入额外的极大似然估计目标函数，同时会使用极大似然估计进行预训练。这样会简化训练，因为随机初始化的演员性能很

差，很难获得有效的奖励。同时极大似然估计也可以被当作一种先验知识，通过正则项的形式约束机器翻译模型的学习，防止模型陷入很差的局部最优，并加速模型收敛。

- **优化目标**。评论家的优化目标是由自身输出所构造的。当模型更新比较快的时候模型的输出变化也会很快，导致构造的优化目标不稳定，影响模型收敛效果。一个解决方案是，在一定更新次数内固定构造优化目标使用的模型，然后再使用比较新的模型来构造后续一定更新次数内的优化目标，如此往复<sup>[607]</sup>。
- **方差惩罚**。在机器翻译中使用强化学习方法的一个问题是动作空间过大，这是由词表过大造成的。因为模型只根据被采样到的结果来进行更新，很多动作很难得到更新，因此对不同动作的动作价值函数估计值会有很大差异。此时，通常会引入一个正则项  $C_j = \sum_{a \in A} (\tilde{Q}(a; \hat{y}_1 \dots j-1, y) - \frac{1}{|A|} \sum_{b \in A} \tilde{Q}(b; \hat{y}_1 \dots j-1, y))^2$  来约束不同动作的动作函数估计值，使其不会偏离均值太远<sup>[608]</sup>。
- **函数塑形**。在机器翻译里面使用强化学习方法的另一个问题就是奖励的稀疏性。评价指标如 BLEU 等只能对完整的句子进行打分，也就是奖励只有在句子结尾有值，而在句子中间为 0。这种情况意味着模型在生成句子的过程中没有任何信号来指导它的行为，从而大大增加了学习难度。常见的解决方案是进行**函数塑形** (Reward Shaping)，使得奖励在生成句子的过程中变得稠密，同时也不会改变模型的最优解<sup>[609]</sup>。

## 13.5 知识蒸馏

理想的机器翻译系统应该是品质好、速度快、存储占用少。不过，为了追求更好的翻译品质，往往需要更大的模型，但是相应的翻译速度会降低，模型的体积会变大。在很多场景下，这样的模型无法直接使用。比如，Transformer-Big 等“大”模型通常在专用 GPU 服务器上运行，在手机等受限环境下仍很难应用。

另一方面，直接训练“小”模型的效果往往并不理想，其翻译品质与“大”模型相比仍有比较明显的差距。既然直接训练小模型无法达到很好的效果，一种有趣的想法是把“大”模型的知识传递给“小”模型。这类似于，教小孩子学习数学，是请一个权威数学家（数据中的标准答案）进行教学，而是会请一个小学数学教师（“大”模型）来教小孩子。这就是知识蒸馏的基本思想。

### 13.5.1 什么是知识蒸馏

通常，知识蒸馏可以被看作是一种知识迁移的手段<sup>[547]</sup>。如果把“大”模型的知识迁移到“小”模型，这种方法的直接结果就是**模型压缩** (Model Compression)。当然，理论上也可以把“小”模型的知识迁移到“大”模型，比如，将迁移后得到的“大”模型作为初始状态，之后继续训练该模型，以期望取得加速收敛的效果。不过，

在实践中更多是使用“大”模型到“小”模型的迁移，这也是本节讨论的重点。

知识蒸馏基于两个假设：

- “知识”在模型间是可迁移的。也就是说，一个模型中蕴含的规律可以被另一个模型使用。最典型的例子就是预训练语言模型（见第九章）。使用单语数据学习到的表示模型，在双语的翻译任务中仍然可以发挥很好的作用。也就是，把单语语言模型学习到的知识迁移到双语翻译中对句子表示的任务中。
- 模型所蕴含的“知识”比原始数据中的“知识”更容易被学习到。比如，机器翻译中大量使用的回译（伪数据）方法，就把模型的输出作为数据让系统进行学习。

这里所说的第二个假设对应了机器学习中的一大类问题——**学习难度**（Learning Difficulty）。所谓难度是指：在给定一个模型的情况下，需要花费多少代价对目标任务进行学习。如果目标任务很简单，同时模型与任务很匹配，那学习难度就会降低。如果目标任务很复杂，同时模型与其匹配程度很低，那学习难度就会很大。在自然语言处理任务中，这个问题的一种表现是：在很好的数据中学习的模型的翻译质量可能仍然很差。即使训练数据是完美的，但是模型仍然无法做到完美的学习。这可能是因为建模的不合理，导致模型无法描述目标任务中复杂的规律。在机器翻译中这个问题体现的尤为明显。比如，在机器翻译系统  $n$ -best 结果中挑选最好的译文（称为 Oracle）作为训练样本让系统重新学习，系统仍然达不到 Oracle 的水平。

知识蒸馏本身也体现了一种“自学习”的思想。即利用模型（自己）的预测来教模型（自己）。这样既保证了知识可以向更轻量的模型迁移，同时也避免了模型从原始数据中学习难度大的问题。虽然“大”模型的预测中也会有错误，但是这种预测是更符合建模的假设的，因此“小”模型反倒更容易从不完美的信息中学习到更多的知识<sup>15</sup>。类似于，刚开始学习围棋的人从职业九段身上可能什么也学不到，但是向一个业余初段的选手学习可能更容易入门。另外，也有研究表明：在机器翻译中，相比于“小”模型，“大”模型更容易进行优化，也更容易找到更好的模型收敛状态<sup>[610]</sup>。因此在需要一个性能优越，存储较小的模型时，也会考虑将大模型压缩得到更轻量模型<sup>[611]</sup>。

通常把“大”模型看作是传授知识的“教师”，被称作**教师模型**（Teacher Model）；把“小”模型看作是接收知识的“学生”，被称作**学生模型**（Student Model）。比如，可以把 Transformer-Big 看作是教师模型，把 Transformer-Base 看作是学生模型。

### 13.5.2 知识蒸馏的基本方法

知识蒸馏的基本思路是让学生模型尽可能去拟合教师模型<sup>[547]</sup>，通常有两种实现方式<sup>[548]</sup>：

<sup>15</sup>很多时候，“大”模型和“小”模型都是基于同一种架构，因此二者对问题的假设和模型结构都是相似的。

- **基于单词的知识蒸馏** (Word-level Knowledge Distillation)。该方法的目标是使得学生模型的预测 (分布) 尽可能逼近教师模型的预测 (分布)。令  $x = \{x_1, \dots, x_m\}$  和  $y = \{y_1, \dots, y_n\}$  分别表示输入和输出 (数据中的答案) 序列,  $V$  表示目标语言词表, 则基于单词的知识蒸馏的损失函数被定义为:

$$L_{\text{word}} = - \sum_{j=1}^n \sum_{y_j \in V} P_t(y_j|x) \log P_s(y_j|x) \quad (13.22)$$

这里,  $P_s(y_j|x)$  和  $P_t(y_j|x)$  分别表示学生模型和教师模型在  $j$  位置输出的概率。公式(13.22)实际上在最小化教师模型和学生模型输出分布之间的交叉熵。

- **基于序列的知识蒸馏** (Sequence-level Knowledge Distillation)。除了单词一级输出的拟合, 基于序列的知识蒸馏希望在序列整体上进行拟合。其损失函数被定义为:

$$L_{\text{seq}} = - \sum_y P_t(y|x) \log P_s(y|x) \quad (13.23)$$

公式(13.23)要求遍历所有可能的译文序列, 并进行求和。当词表大小为  $V$ , 序列长度为  $n$  时, 则序列的数量有  $V^n$  个。因此, 会考虑用教师模型的真实输出序列  $\hat{y}$  来代替整个空间, 即假设  $P_t(\hat{y}|x) = 1$ 。于是, 目标函数变为:

$$L_{\text{seq}} = -\log P_s(\hat{y}|x) \quad (13.24)$$

这样的损失函数最直接的好处是, 知识蒸馏的流程会非常简单。因为只需要利用教师模型将训练数据 (源语言) 翻译一遍, 之后把它的输出替换为训练数据的目标语言部分。之后, 利用新得到的双语数据训练学生模型即可。图13.12对比了词级和序列级知识蒸馏方法。

本质上, 基于单词的知识蒸馏与语言建模等问题的建模方式是一致的。在传统方法中, 训练数据中的答案会被看作是一个 One-hot 分布, 之后让模型去尽可能拟合这种分布。而这里, 答案不再是一个 One-hot 分布, 而是由教师模型生成的真实分布, 但是损失函数的形式是一模一样的。在具体实现时, 一个容易出现的问题是在词级别的知识蒸馏中, 教师模型的 Softmax 可能会生成非常尖锐的分布。这时需要考虑对分布进行平滑, 提高模型的泛化能力, 比如, 可以在 Softmax 函数中加入一个参数  $\alpha$ , 如  $\text{Softmax}(s_i) = \frac{\exp(s_i/\alpha)}{\sum_{i'} \exp(s_{i'}/\alpha)}$ 。这样可以通过  $\alpha$  控制分布的平滑程度。

除了在模型最后输出的分布上进行知识蒸馏, 同样可以使用教师模型对学生模型的中间层输出和注意力分布进行约束。这种方法在第十四章中会有具体应用。

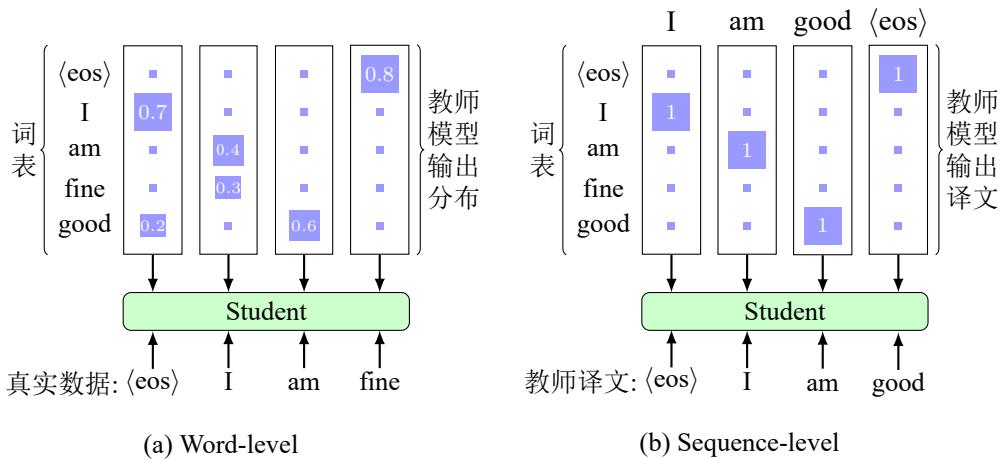


图 13.12 词级和序列级知识蒸馏的差异

### 13.5.3 机器翻译中的知识蒸馏

在神经机器翻译中，通常使用公式(13.24)的方法进行知识蒸馏，即通过教师模型构造伪数据，之后让学生模型从伪数据中学习。这样做的好处在于，系统研发人员不需要对系统进行任何修改，整个过程只需要调用教师模型和学生模型标准的训练和推断模块即可。

另一个问题是如何构造教师模型和学生模型。以 Transformer 为例，通常有两种思路：

- 固定教师模型，通过减少模型容量的方式设计学生模型。比如，可以使用容量较大的模型作为教师模型（如：Transformer-Big 或 Transformer-Deep），然后通过将神经网络变“窄”、变“浅”的方式得到学生模型。例如，可以用 Transformer-Big 做教师模型，然后把 Transformer-Big 的解码器变为一层网络，作为学生模型。
- 固定学生模型，通过模型集成的方式设计教师模型。可以组合多个模型生成更高质量的译文。比如，融合多个 Transformer-Big 模型（不同参数初始化方式），之后学习一个 Transformer-Base 模型。

此外还可以采用迭代式知识蒸馏的方式。首先，通过模型集成得到较强的教师模型，再将知识迁移到不同的学生模型上，随后继续使用这些学生模型集成新的教师模型。不断的重复上述过程可以逐步提升集成模型的性能，如图13.13所示。值得注意的是，随着迭代次数的增加，集成所带来的收益也会随着子模型之间差异性的减小而减少。

如果倾向于使用更少的存储，更快的推理速度，则可以使用更小的学生模型。值得注意的是，对于 Transformer 模型来说，减少解码端的层数会给推理速度带来巨大的提升。特别是对于基于深层编码器的 Transformer-Deep，适当减少解码端层数往往不会带来翻译品质的下降。可以根据不同任务的需求，选择适当大小的学生模型，来

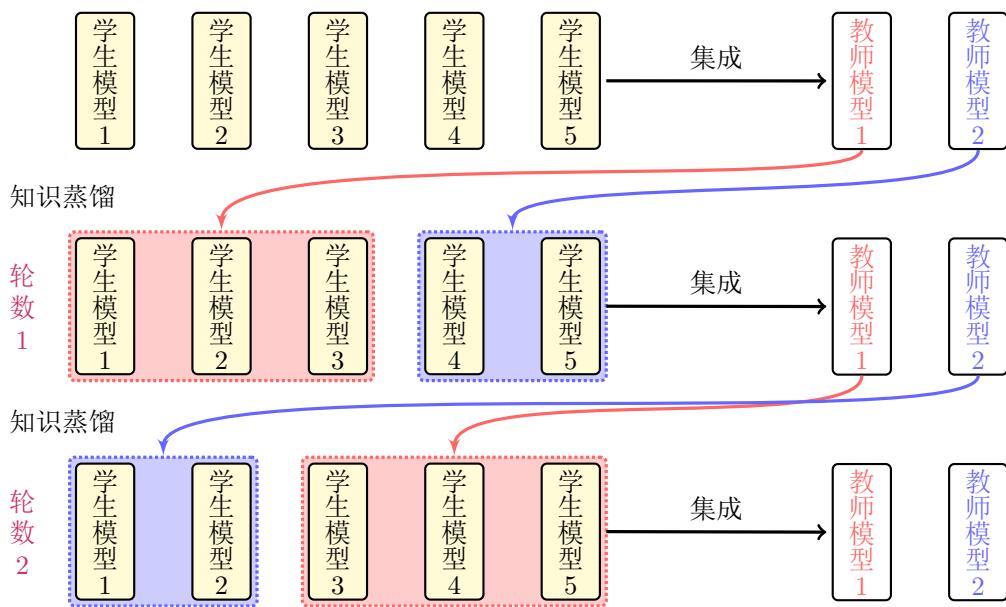


图 13.13 迭代式知识蒸馏

平衡存储空间、推断速度和模型品质之间的关系。

## 13.6 基于样本价值的学习

当人在学习知识时，通常会遵循序渐进、由易到难的原则，这是一种很自然的学习策略。但是，当训练机器翻译模型时，通常是将全部的样本以随机的方式输入模型中进行学习，换句话说，就是让模型来平等地对待所有的训练样本。这种方式忽略了样本对于模型训练的“价值”，显然，更加理想的方式是优先使用价值高的样本对模型进行训练。围绕训练样本的价值差异产生了诸如数据选择、主动学习、课程学习等一系列的样本使用方法，这些学习策略本质上是在不同任务、不同背景、不同假设下，对如何高效地利用训练样本这一问题进行求解，本节即对这些技术进行介绍。

### 13.6.1 数据选择

模型学习的目的就是要学习训练数据中的分布，以期望模型学到的分布和真实的分布越接近越好。然而训练数据是从真实世界中采样得来的，这导致了训练数据无法完整地描述客观世界的真实规律。这种分布的不匹配有许多不同的表现形式，例如，类别不平衡、领域差异、存在标签噪声等，这导致模型在实践中表现不佳。

类别不平衡在分类任务中更为常见，可以通过重采样、代价敏感训练等手段来解决。数据选择则是缓解领域差异和标签噪声等问题的一种有效手段，它的学习策略是让模型有选择地使用样本进行学习。此外，在一些稀缺资源场景下还会面临标

注数据稀少的情况，此时可以利用主动学习选择那些最有价值的样本优先进行人工标注，从而降低标注成本。

显然，上述方法都基于一个假设：在训练过程中，每个样本都是有价值的，且这种价值可以计算。价值在不同任务背景下有不同的含义，这与任务的特性有关。比如，在领域相关数据选择中，样本的价值表示这个样本与领域的相关性；在数据降噪中，价值表示样本的可信度；在主动学习中，价值表示样本的难易程度。

## 1. 领域相关的数据选择

当机器翻译系统应用于不同领域时，训练语料与所应用领域的相关性就显得非常重要<sup>[612, 613]</sup>。不同领域往往具有自己独特的属性，比如语言风格、句子结构、专业术语等，例如，“bank”这个英语单词，在金融领域通常被翻译为“银行”，而在计算机领域，一般被解释为“库”、“存储体”等。这也会导致，使用通用领域数据训练出来的模型在特定领域上的翻译效果往往不理想，这本质上是训练数据和测试数据的领域属性不匹配造成的。

一种解决办法是只使用特定领域的数据进行模型训练，然而这种数据往往比较稀缺。那能不能利用通用领域数据来帮助数据稀少的领域呢？这个研究方向被称为机器翻译的**领域适应**（Domain Adaptation），即从资源丰富的领域（称为**源领域**，Source Domain）向资源稀缺的领域（称为**目标领域**，Target Domain）迁移。这本身也对应着资源稀缺场景下的机器翻译问题，这类问题会在第十六章进行详细讨论。本章更加关注如何有效地利用训练样本以更好地适应目标领域。具体来说，可以使用**数据选择**（Data Selection）从源领域训练数据中选择与目标领域更加相关的样本进行模型训练。这样做的一个好处是，源领域中混有大量与目标领域不相关的样本，数据选择可以有效降低这部分数据的比例，这样可以更加突出与领域相关样本的作用。

数据选择所要解决的核心问题是：给定一个目标领域/任务数据集（如，目标任务的开发集），如何衡量原始训练样本与目标领域/任务的相关性？主要方法可以分为以下几类：

- **基于交叉熵差**（Cross-entropy Difference, CED）**的方法**<sup>[614, 615, 616, 617]</sup>。该方法在目标领域数据和通用数据上分别训练语言模型，然后用两个语言模型来给句子打分并做差，差越小说明句子与目标领域越相关。
- **基于文本分类的方法**<sup>[618, 619, 620, 621]</sup>。将问题转化为文本分类问题，先构造一个领域分类器，之后利用分类器对给定的句子进行领域分类，最后用输出的概率来打分，选择得分高的样本。
- **基于特征衰减算法**（Feature Decay Algorithms, FDA）**的方法**<sup>[622, 623, 624]</sup>。该算法基于特征匹配，试图从源领域中提取出一个句子集合，这些句子能够最大程度覆盖目标领域的语言特征。

上述方法实际上描述了一种静态的学习策略，即首先利用评分函数对源领域的

数据进行打分排序，然后选取一定数量的数据合并到目标领域数据集中，并共同训练模型<sup>[614, 615, 618, 619]</sup>。这个过程其实是扩大了目标领域的数据规模，模型的收益主要来自于数据的增加。但是研究人员也发现静态方法会存在两方面的缺陷：

- 在选定的子集上进行训练会导致词表覆盖率的降低，并加剧单词长尾分布问题<sup>[615, 625]</sup>。
- 静态方法可以看作一种数据过滤技术，它对数据的判定方式是“非黑即白”的，即接收或拒绝。这种方式一方面会受到评分函数的影响，一方面被拒绝的数据可能对于训练模型仍然有用，而且样本的价值可能会随着训练过程的推进而改变<sup>[626]</sup>。

使用动态学习策略可以有效地缓解上述问题。它的基本想法是：不完全抛弃领域相关性低的样本，而只是使模型给予相关性高的样本更高的关注度，使得它更容易参与到训练过程中。具体在实现上，主要有两种方法，一种是将句子的领域相似性表达成概率分布，然后在训练过程中根据该分布对数据进行动态采样<sup>[625, 627]</sup>，另一种是在计算损失函数时根据句子的领域相似性以加权的方式进行训练<sup>[616, 620]</sup>。相比于静态方法的二元选择方式，动态方法是一种“软”选择的方式，这使得模型有机会使用到其它数据，提高了训练数据的多样性，因此性能也更稳定。

## 2. 数据降噪

除了领域差异，训练数据中也存在噪声，比如，机器翻译所使用的数据中经常出现句子未对齐、多种语言文字混合、单词丢失等问题。相关研究表明神经机器翻译对于噪声数据很敏感<sup>[628]</sup>，因此无论是从训练效果还是训练效率出发，数据降噪都是很有意义的。事实上，在统计机器翻译时代，就有很多数据降噪方面的研究工作<sup>[629, 630, 631]</sup>，因此许多方法也可以应用到神经机器翻译中来。

含有噪声的数据通常都具有较为明显的特征，因此可以用诸如句子长度比、词对齐率、最长连续未对齐序列长度等一些特征来对句子进行综合评分<sup>[632, 633, 634]</sup>；也可以将该问题转化为分类任务来对句子进行筛选<sup>[635, 636]</sup>；此外，从某种意义上来说，数据降噪其实也可以算是一种领域数据选择，因为它的目标是选择可信度高的样本，因此也可以人工构建一个可信度高的小数据集，然后利用该数据集和通用数据集之间的差异性进行选择<sup>[626]</sup>。

早期的工作大多在关注过滤噪声样本，但对如何利用噪声样本探讨较少。事实上，噪声是有强度的，有些噪声样本对于模型可能是有价值的，而且它们的价值可能会随着模型的状态而改变<sup>[626]</sup>。对于一个双语句对“我/喜欢/那个/地方/。↔ I love that place. It's very beautiful”。一方面来说，虽然这两个句子都很流畅，但是由于汉语句子中缺少了一部分翻译，因此简单的基于长度或双语词典的方法可以很容易将其过滤掉。从另一方面来说，这个样本对于训练机器翻译模型仍然有用，特别是在数据稀缺的情况下，因为汉语句子和英语句子的前半部分仍然是正确的互译结果。这表

明了噪声数据的微妙之处，它不是对应着简单的二元分类问题：一些训练样本可能部分有用。因此简单的过滤并不一种很好的办法，一种更加理想的学习策略应该是既可以合理的利用这些数据，又不让其对模型产生负面影响。例如，在训练过程中对批量数据的噪声水平进行退火（Anneal），使得模型在越来越干净的数据上进行训练<sup>[626, 637]</sup>。从宏观上看，整个训练过程其实是一个持续微调的过程，这和微调的思想基本一致。这种学习策略一方面充分利用了训练数据，一方面又避免了噪声数据对模型的负面影响，因此取得了不错的效果。

### 3. 主动学习

**主动学习**（Active Learning）也是一种数据选择策略。它最初的应用场景是：标注大量的数据成本过高，因此希望优先标注对模型最有价值的数据，这样可以最大化模型学习的效率，同时降低数据标注的整体代价<sup>[638]</sup>。主动学习主要由五个部分组成，包括：未标注样本池、筛选策略、标注者、标注样本集、目标模型。在主动学习过程中，会根据当前的模型状态找到未标注样本池中最有价值的样本，之后送给标注者。标注结束后，会把标注的样本加入到标注样本集中，之后用这些标注的样本更新模型。之后，重复这个过程，直到到达某种收敛状态。

主动学习的一个核心问题是：如何选择出那些最有价值的未标注样本？通常会假设模型认为最“难”的样本是最有价值的。具体实现有很多思路，例如，基于置信度的方法、基于分类错误的方法等等<sup>[639, 640]</sup>。

在机器翻译中，主动学习可以被用于低资源翻译，以减少人工标注的成本<sup>[641, 642]</sup>。也可以被用于交互式翻译，让模型持续从外界反馈中受益<sup>[643, 644, 645]</sup>。不过，总的来说，主动学习在机器翻译中应用不算广泛。这是由于，机器翻译任务很复杂，设计样本价值的评价函数较为困难。而且，在很多场景中，并不是要简单的选择样本，而是希望训练装置能够考虑样本的价值，以充分发挥所有数据的优势。这也正是即将介绍的课程学习等方法要解决的问题。

#### 13.6.2 课程学习

**课程学习**（Curriculum Learning）的基本思想是：先学习简单的、普适性的知识，然后逐渐增加难度，学习更复杂、更专业化的知识。在统计模型训练中，这种思想可以体现在让模型按照由“易”到“难”的顺序对样本进行学习<sup>[646]</sup>，这本质上是一种样本使用策略。以神经机器翻译使用的随机梯度下降为例，在传统的方法中，所有训练样本都是随机呈现给模型的，换句话说，就是让模型平等地对待所有的训练样本，这忽略了数据样本的各种复杂性和当前模型的学习状态。所以模拟人类由易到难的学习过程就是一种很自然的想法，这样做的好处在于：

- **加速模型训练**。在达到相同的性能条件下，课程学习可以加速训练，减少训练迭代步数。

- 使模型获得更好的泛化性能。即通过对简单样本的学习，让模型不至于过早进入拟合复杂样本的状态。

课程学习是符合直觉的，可以想象，对于一个数学零基础的人来说，如果一开始就同时学习加减乘除和高等数学，效率自然是比较低下的。而如果按照正常的学习顺序，比如先学习加减乘除，然后学习各种函数，最后再学习高等数学，有了前面的基础，再学习后面的知识，效率就可以更高。事实上，课程学习自从被提出就受到了研究人员的极大关注，除了想法本身有趣之外，还因为它作为一种和模型无关的训练策略，具有即插即用的特点。神经机器翻译就是一种很契合课程学习的任务，这是因为神经机器翻译往往需要大规模的平行语料来训练模型，训练成本很高，所以使用课程学习来加快收敛是一个很自然的想法。

那么如何设计课程学习方法呢？有两个核心问题：

- 如何评估每个样本的难度？即设计评估样本学习难易度的准则，简称**难度评估准则**（Difficulty Criteria）
- 以何种策略来规划训练数据？即何时为训练提供更复杂的样本，以及提供多少样本等，称为**课程规划**（Curriculum Schedule）

这里，把这两个问题抽象成两个模块：难度评估器和训练调度器，那么课程学习的一个大致的流程如图13.14所示。首先，难度评估器对训练样本按照由易到难的顺序进行排序，最开始调度器从相对容易的数据块中采样训练样本，发送给模型进行训练，随着训练时间的推移，训练调度器将逐渐从更加困难的数据块中进行采样（至于何时，以及选择何种采样方式则取决于设定的策略），持续这个过程，直到从整个训练集进行均匀采样。

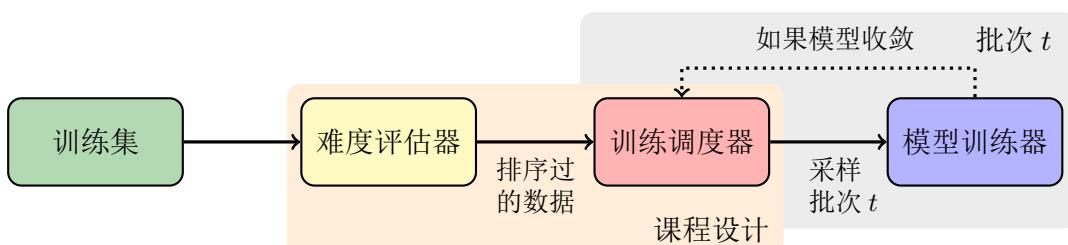


图 13.14 课程学习框架

评估样本的难度和具体的任务相关，在神经机器翻译中，有很多种评估方法，可以利用语言学上的**困难准则**，比如句子长度、句子平均词频、句法树深度等<sup>[647, 648]</sup>。这些准则本质上属于人类的先验知识，符合人类的直觉，但不一定和模型相匹配。对人类来说简单的句子对模型来说可能并不简单，所以研究人员也提出了**基于模型的方法**，比如：语言模型<sup>[637, 649]</sup>，或者神经机器翻译模型<sup>[599, 650]</sup>都可以用于评价样本的难度。值得注意的是，利用神经机器翻译来打分的方法分为静态和动态两种。静态的

方法是利用在小数据集上训练的、更小的翻译模型来打分<sup>[650]</sup>。动态的方法则是利用当前模型的状态来打分，这在广义上也叫作**自步学习**（Self-paced Learning），通常可以利用模型的训练误差或变化率等指标进行样本难度的估计<sup>[599]</sup>。

虽然样本难度的度量在不同任务中有所不同，但课程规划通常与数据和任务无关。在各种场景中，大多数课程学习都利用了类似的调度策略。具体而言，调度策略可以分为预定义的和自动的两种。预定义的调度策略通常将按照难易程度排序好的样本划分为块，每个块中包含一定数量的难度相似的样本。然后按照“先易后难”的原则人工定义一个调度策略，比如，一种较为流行的方法是：在训练早期，模型只在简单块中进行采样，随着训练过程的进行，将下一个块的样本合并到当前训练子集中，继续训练，直到合并了整个数据块，即整个训练集可见为止，之后再继续训练直到收敛。这个过程如图13.15所示。类似的还有一些其他变体，比如，训练到模型可见整个数据集之后，将最难的样本块复制并添加到训练集中，或者是将最容易的数据块逐渐删除，然后再添加回来等，这些方法的基本想法都是想让模型在具备一定的能力之后更多关注于困难样本。

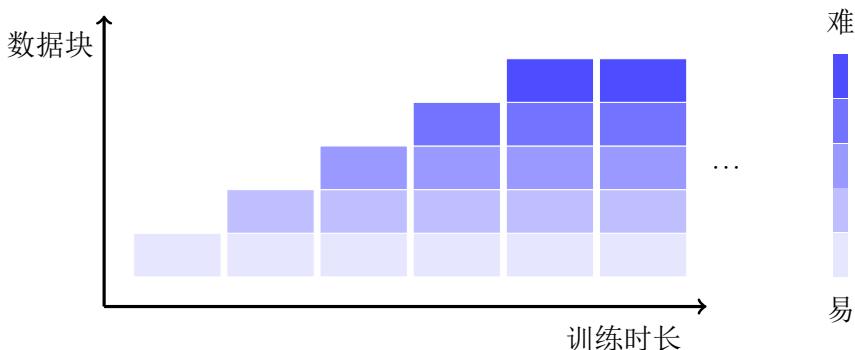


图 13.15 “先易后难” 数据块选择

尽管预定义的调度策略简单有效，但也会面临着方法不够灵活、数据块划分不合理等问题，而且这种策略在一定程度上也忽略了当前模型的反馈。因此另一种方法是自动的方法，根据模型的反馈来动态调整样本的难度或调度策略，模型的反馈可以是模型的不确定性<sup>[651]</sup>、模型的能力<sup>[599, 647]</sup>等。这些方法在一定程度上使得整个训练过程和模型的状态相匹配，同时样本的选择过渡得更加平滑，因此在实践中取得了不错的效果。

### 13.6.3 持续学习

人类具有不断学习、调整和转移知识的能力，这种能力被称为**持续学习**（Continual Learning），也叫**终生学习**（Lifelong Learning）或**增量式学习**（Incremental Learning）。人类学习的新任务时，会很自然的利用以前的知识并将新学习的知识整合到以前的知识中。然而对于机器学习系统来说，尤其在连接主义的范式下（如深度神经网络

模型），这是一个很大的挑战，这是由神经网络的特性所决定的。当前的神经网络模型依赖于标注的训练样本，通过反向传播算法对模型参数进行训练更新，最终达到拟合数据分布的目的。当把模型切换到新的任务时，本质上是数据的分布发生了变化，从这种分布差异过大的数据中不断增量获取可用信息很容易导致**灾难性遗忘**（Catastrophic Forgetting）问题，即用新数据训练模型的时候会干扰先前学习的知识。甚至，这在最坏的情况下会导致旧知识被新知识完全重写。在机器翻译领域，类似的问题经常发生在不断增加数据的场景中，因为当用户使用少量数据对模型进行更新之后，发现在旧数据上的性能下降了（见第十八章）。

为克服灾难性遗忘问题，学习系统一方面必须能连续获取新知识和完善现有知识，另一方面，还应防止新数据输入明显干扰现有的知识，这个问题称作**稳定性-可塑性**（Stability-Plasticity）问题。可塑性指整合新知识的能力，稳定性指保留先前的知识不至于遗忘。要解决这些问题，就需要模型在保留先前任务的知识与学习当前任务的新知识之间取得平衡。目前的解决方法可以分为以下几类：

- **基于正则化的方法**。通过对模型参数的更新施加约束来减轻灾难性的遗忘，通常是在损失函数中引入了一个额外的正则化项，使得模型在学习新数据时巩固先前的知识<sup>[652, 653]</sup>。
- **基于实例的方法**。基于实例的方法。在学习新任务的同时混合训练先前的任务样本以减轻遗忘，这些样本可以是从先前任务的训练数据中精心挑选出的子集，或者利用生成模型生成的伪样本<sup>[654, 655]</sup>。
- **基于动态模型架构的方法**。例如，增加神经元或新的神经网络层进行重新训练，或者是在新任务训练时只更新部分参数<sup>[656, 657]</sup>。

从某种程度上看，多领域、多语言机器翻译等都可以被看做是广义上的持续学习。在多领域神经机器翻译中，研究人员期望一个在通用数据上学习的模型可以继续在新的领域有良好的表现。在多语言神经机器翻译中，研究人员期望一个模型可以支持更多语种的翻译，甚至当新的语言到来时不需要修改模型结构。以上这些问题在第十六章和第十八章中还会有详细讨论。

## 13.7 小结及拓展阅读

本章以不同的角度讨论了神经机器翻译模型的训练问题。一方面，可以作为第九章~第十二章内容的扩展，另一方面，也为本书后续章节的内容进行铺垫。从机器学习的角度看，本章介绍的很多内容并不仅仅使用在机器翻译中，大多数的内容同样适用于其它自然语言处理任务。此外，本章也讨论了许多与机器翻译相关的问题（如大词表），这又使得本章的内容具有机器翻译的特性。总的来说，模型训练是一个非常开放的问题，在后续章节中还会频繁涉及。同时，也有一些方向可以关注：

- 对抗样本除了用于提高模型的健壮性之外，还有很多其他的应用场景，比如评

估模型。通过构建由对抗样本构造的数据集，可以验证模型对于不同类型噪声的健壮性<sup>[658]</sup>。但是在生成对抗样本时常常要考虑很多问题，比如扰动是否足够细微<sup>[573, 575]</sup>，在人类难以察觉的同时做到欺骗模型的目的；对抗样本在不同的模型结构或数据集上是否具有足够的泛化能力<sup>[659, 660]</sup>；生成的方法是否足够高效等等<sup>[578, 661]</sup>。

- 此外，在机器翻译中，强化学习的应用也有很多，比如，MIXER 算法用混合策略梯度和极大似然估计的目标函数来更新模型<sup>[598]</sup>，Dagger<sup>[662]</sup> 以及 DAD<sup>[663]</sup> 等算法在训练过程之中逐渐让模型适应推断阶段的模式。此外，强化学习的效果目前还相当不稳定，研究人员提出了大量的方法来进行改善，比如降对动作价值函数  $Q$  的估计的方差<sup>[601, 664]</sup>、使用单语语料<sup>[665, 666]</sup> 等等。
- 从广义上说，大多数课程学习方法都是遵循由易到难的原则，然而在实践过程中人们逐渐赋予了课程学习更多的内涵，课程学习的含义早已超越了最原始的定义。一方面，课程学习可以与许多任务相结合，此时，评估准则并不一定总是样本的困难度，这取决于具体的任务。或者说，我们更关心的是样本带给模型的“价值”，而非简单的难易标准。另一方面，在一些任务或数据中，由易到难并不总是有效，有时困难优先反而会取得更好的效果<sup>[650, 667]</sup>。实际上这和人类的直觉不太符合，一种合理的解释是课程学习更适合标签噪声、离群值较多或者是目标任务困难的场景，能提高模型的健壮性和收敛速度，而困难优先的策略则更适合数据集干净的场景<sup>[668]</sup>。



## 14. 神经机器翻译模型推断

推断是神经机器翻译中的核心问题。训练时双语句子对模型是可见的，但是在推断阶段，模型需要根据输入的源语言句子预测译文，因此神经机器翻译的推断和训练过程有着很大的不同。特别是，推断系统往往对应着机器翻译实际部署的需要，因此机器翻译推断系统的精度和速度等因素也是实践中需要考虑的。

本章对神经机器翻译模型推断中的若干问题进行讨论。主要涉及三方面内容：

- 神经机器翻译的基本问题，如推断方向、译文长度控制等。
- 神经机器翻译的推断加速方法，如轻量模型、非自回归模型等。
- 多模型集成推断。

### 14.1 面临的挑战

神经机器翻译的推断是指：对于输入的源语言句子  $x$ ，使用已经训练好的模型找到最佳译文  $\hat{y}$  的过程，其中  $\hat{y} = \arg \max_y P(y|x)$ 。这个过程也被称作解码。但是为了避免与神经机器翻译中编码器-解码器造成概念上的混淆，这里统一把翻译新句子的操作称作推断。以上这个过程是一个典型的搜索问题（见第二章），比如，可以使用贪婪搜索或者束搜索完成神经机器翻译的推断（见第十章）。

通用的神经机器翻译推断包括如下几步：

- 对输入的源语言句子进行编码；

- 使用源语言句子的编码结果，在目标语言端自左向右逐词生成译文；
- 在目标语言的每个位置计算模型得分，同时进行剪枝；
- 当满足某种条件时终止搜索。

这个过程与统计机器翻译中自左向右翻译是一样的（见第七章），即在目标语言的每个位置，根据已经生成的部分译文和源语言的信息，生成下一个译文单词<sup>[80, 81]</sup>。它可以由两个模块实现<sup>[669]</sup>：

- **预测模块**，它根据已经生成的部分译文和源语言信息，预测下一个要生成的译文单词的概率分布<sup>1</sup>。因此预测模块实际上就是一个模型打分装置；
- **搜索模块**，它会利用预测结果，对当前的翻译假设进行打分，并根据模型得分对翻译假设进行排序和剪枝。

预测模块是由模型决定的，而搜索模块可以与模型无关。也就是说，不同的模型可以共享同一个搜索模块完成推断。比如，对于基于循环神经网络的模型，预测模块需要读入前一个状态的信息和前一个位置的译文单词，然后预测当前位置单词的概率分布；对于 Transformer，预测模块需要对前面的所有位置做注意力运算，之后预测当前位置单词的概率分布。不过，这两个模型都可以使用同一个搜索模块。图14.1给出了这种架构的示意图。

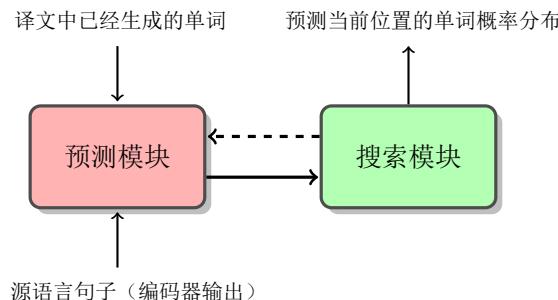


图 14.1 神经机器翻译推断系统结构

这是一个非常通用的框架，同样适用于统计机器翻译模型。因此，神经机器翻译推断中的很多问题与统计机器翻译是一致的，比如：束搜索的宽度、解码终止条件等等。

一般来说，设计机器翻译推断系统需要考虑三个因素：搜索的准确性、搜索的时延、搜索所需要的存储。通常，准确性是研究人员最关心的问题，比如可以通过增大搜索空间来找到模型得分更高的结果。而搜索的时延和存储消耗是实践中必须要考虑的问题，比如可以设计更小的模型和更高效的推断方法来提高系统的可用性。

<sup>1</sup>在统计机器翻译中，翻译的每一步也可以同时预测若干个连续的单词，即短语。在神经机器翻译中也有类似于生成短语的方法，但是主流的方法还是按单词为单位进行生成。

虽然，上述问题在统计机器翻译中均有讨论，但是在神经机器翻译中又面临着新的挑战。

- 搜索的基本问题在神经机器翻译中有着特殊的现象。比如，在统计机器翻译中，减少搜索错误是提升翻译品质的一种手段。但是神经机器翻译中，简单的减少搜索错误可能无法带来性能的提升，甚至会造成翻译品质的下降<sup>[472, 670]</sup>；
- 搜索的时延很高，系统实际部署的成本很高。与统计机器翻译系统不同的是，神经机器翻译依赖大量的浮点运算。这导致神经机器翻译系统的推断会比统计机器翻译系统慢很多。虽然可以使用 GPU 来提高神经机器翻译的推断速度，但是也大大增加了成本；
- 神经机器翻译在优化过程中容易陷入局部最优，单模型的表现并不稳定。由于神经机器翻译优化的目标函数非常不光滑，每次训练得到的模型往往只是一个局部最优解。在新数据上使用这个局部最优模型进行推断时，模型的表现可能不稳定。

研究人员也针对以上问题开展了大量的研究工作。在14.2节中，会对神经机器翻译推断中所涉及的一些基本问题进行讨论。虽然这些问题在统计机器翻译中也有涉及，但是在神经机器翻译中却有着不同的现象和解决思路。在14.3-14.5节中，会围绕如何改进神经机器翻译推断效率和怎样进行多模型融合这两个问题展开讨论。

## 14.2 基本问题

下面将就神经机器翻译推断中的若干基本问题进行讨论，包括：推断方向、译文长度控制、搜索终止条件、译文多样性、搜索错误五个方面。

### 14.2.1 推断方向

机器翻译有两种常用的推断方式——自左向右推断和自右向左推断。自左向右推断符合现实世界中人类的语言使用规律，因为人在翻译一个句子时，总是习惯从句子开始的部分向后生成<sup>2</sup>。不过，有时候人也会使用当前单词后面的译文信息。也就是说，翻译也需要“未来”的文字信息。于是很容易想到使用自右向左的方式对译文进行生成。

以上两种推断方式在神经机器翻译中都有应用，对于源语言句子  $x = \{x_1, \dots, x_m\}$  和目标语言句子  $y = \{y_1, \dots, y_n\}$ ，自左向右推断可以被描述为：

$$P(y|x) = \prod_{j=1}^n P(y_j|y_{<j}, x) \quad (14.1)$$

<sup>2</sup>有些语言中，文字是自右向左书写，这时自右向左推断更符合人类使用这种语言的习惯。

自右向左推断可以被描述为：

$$P(y|x) = \prod_{j=1}^n P(y_{n+1-j}|y_{>n+1-j}, x) \quad (14.2)$$

其中， $y_{<j} = \{y_1, \dots, y_{j-1}\}$ ， $y_{>n+1-j} = \{y_{n+1-j}, \dots, y_n\}$ 。可以看到，自左向右推断和自右向左推断本质上是一样的。第十章到第十二章均使用了自左向右的推断方法。自右向左推断比较简单的实现方式是：在训练过程中直接将双语数据中的目标语言句子进行反转，之后仍然使用原始的模型进行训练即可。在推断的时候，生成的目标语言词串也需要进行反转得到最终的译文。有时候，使用自右向左的推断方式会取得更好的效果<sup>[671]</sup>。不过更多情况下需要同时使用词串左端（历史）和右端（未来）的信息。有多种思路可以融合左右两端信息：

- **重排序** (Reranking)。可以用一个基础模型（比如自左向右的模型）得到每个源语言句子的  $n$ -best 翻译结果，之后同时用基础模型的得分和自右向左模型的得分对  $n$ -best 翻译结果进行重排序<sup>[671, 672, 673]</sup>。也有研究人员利用最小贝叶斯风险的方法进行重排序<sup>[674]</sup>。由于这类方法不会改变基础模型的翻译过程，因此相对“安全”，不会对系统性能造成副作用。
- **双向推断** (Bidirectional Inference)。除了自左向右推断和自右向左推断，另一种方法让自左向右和自右向左模型同步进行，也就是同时考虑译文左侧和右侧的文字信息<sup>[675, 676]</sup>。例如，可以同时对左侧和右侧生成的译文进行注意力计算，得到当前位置的单词预测结果。这种方法能够更加充分地融合双向翻译的优势。
- **多阶段推断** (Multi-stage Inference)。在第一阶段，通过一个基础模型生成一个初步的翻译结果。在第二阶段，同时使用第一阶段生成的翻译结果和源语言句子，进一步生成更好的译文<sup>[677, 678, 679]</sup>。由于第一阶段的结果已经包含了完整的译文信息，因此在第二阶段中，系统实际上已经同时使用了整个译文串的两端信息。上述过程可以扩展为迭代式的译文生成方法，配合掩码等技术，可以在生成每个译文单词时，同时考虑左右两端的上下文信息<sup>[680, 681, 682]</sup>。

不论是自左向右推断还是自右向左推断，本质上都是在对上下文信息进行建模。此外，研究人员也提出了许多新的译文生成策略，比如，从中部向外生成<sup>[683]</sup>、按源语言顺序生成<sup>[684]</sup>、基于插入的方式生成<sup>[685, 686]</sup>等。或者将翻译问题松弛化为一个连续空间模型的优化问题，进而在推断的过程中同时使用译文左右两端的信息<sup>[679]</sup>。

最近，以 BERT 为代表的预训练语言模型已经证明，一个单词的“历史”和“未来”信息对于生成当前单词都是有帮助的<sup>[123]</sup>。类似的观点也在神经机器翻译编码器设计中得到验证。比如，在基于循环神经网络的模型中，经常同时使用自左向右和自右向左的方式对源语言句子进行编码；在 Transformer 模型中，编码器会使用整个句子的信息对每一个源语言位置进行表示。因此，神经机器翻译的推断采用类似的策略是有其合理性的。

## 14.2.2 译文长度控制

机器翻译推断的一个特点是译文长度需要额外的机制进行控制<sup>[687, 688, 689, 690]</sup>。这是因为机器翻译在建模时仅考虑了将训练样本（即标准答案）上的损失最小化，但是推断的时候会看到从未见过的样本，而且这些未见样本占据了样本空间的绝大多数。该问题会导致的一个现象是：直接使用训练好的模型会翻译出长度短得离谱的译文。神经机器翻译模型使用单词概率的乘积表示整个句子的翻译概率，它天然就倾向生成短译文，因为概率为大于 0 小于 1 的常数，短译文会使用更少的概率因式相乘，倾向于得到更高的句子得分，而模型只关心每个目标语言位置是否被正确预测，对于译文长度没有考虑。统计机器翻译模型中也存在译文长度不合理的问题，解决该问题的常见策略是在推断过程中引入译文长度控制机制<sup>[80]</sup>。神经机器翻译也借用了类似的思想来控制译文长度，有以下几种方法：

- **长度惩罚因子。**用译文长度来归一化翻译概率是最常用的方法：对于源语言句子  $x$  和译文句子  $y$ ，模型得分  $\text{score}(x, y)$  的值会随着译文  $y$  的长度增大而减小。为了避免此现象，可以引入一个长度惩罚函数  $\text{lp}(y)$ ，并定义模型得分如公式(14.3)所示：

$$\text{score}(x, y) = \frac{\log P(y|x)}{\text{lp}(y)} \quad (14.3)$$

通常  $\text{lp}(y)$  随译文长度  $|y|$  的增大而增大，因此这种方式相当于对  $\log P(y|x)$  按长度进行归一化<sup>[691]</sup>。 $\text{lp}(y)$  的定义方式有很多，表14.1列出了一些常用的形式，其中  $\alpha$  是需要人为设置的参数。

表 14.1 长度惩罚因子  $\text{lp}(y)$  的定义 ( $|y|$  表示译文长度)

名称	$\text{lp}(y)$
句子长度	$\text{lp}(y) =  y ^\alpha$
GNMT 惩罚因子	$\text{lp}(y) = \frac{(5+ y )^\alpha}{(5+1)^\alpha}$
指数化长度惩罚因子	$\text{lp}(y) = \alpha \cdot \log( y )$

- **译文长度范围约束。**为了让译文的长度落在合理的范围内，神经机器翻译的推断也会设置一个译文长度约束<sup>[535, 692]</sup>。令  $[a, b]$  表示一个长度范围，可以定义：

$$a = \omega_{\text{low}} \cdot |x| \quad (14.4)$$

$$b = \omega_{\text{high}} \cdot |x| \quad (14.5)$$

其中， $\omega_{\text{low}}$  和  $\omega_{\text{high}}$  分别表示译文长度的下限和上限，比如，很多系统中设置为  $\omega_{\text{low}} = 1/2$ ,  $\omega_{\text{high}} = 2$ ，表示译文至少有源语言句子一半长，最多有源语言

句子两倍长。 $\omega_{\text{low}}$  和  $\omega_{\text{high}}$  的设置对推断效率影响很大， $\omega_{\text{high}}$  可以被看作是一个推断的终止条件，最理想的情况是  $\omega_{\text{high}} \cdot |x|$  恰巧就等于最佳译文的长度，这时没有浪费任何计算资源。反过来的一种情况， $\omega_{\text{high}} \cdot |x|$  远大于最佳译文的长度，这时很多计算都是无用的。为了找到长度预测的准确率和召回率之间的平衡，一般需要大量的实验最终确定  $\omega_{\text{low}}$  和  $\omega_{\text{high}}$ 。当然，利用统计模型预测  $\omega_{\text{low}}$  和  $\omega_{\text{high}}$  也是非常值得探索的方向，比如基于繁衍率的模型<sup>[271, 693]</sup>。

- **覆盖度模型**。译文长度过长或过短的问题，本质上对应着**过翻译**(Over Translation) 和**欠翻译**(Under Translation) 的问题<sup>[694]</sup>。这两种问题出现的原因主要在于：神经机器翻译没有对过翻译和欠翻译建模，即机器**翻译覆盖度**问题<sup>[473]</sup>。针对此问题，最常用的方法是在推断的过程中引入一个度量覆盖度的模型。比如，使用GNMT 覆盖度模型定义模型得分<sup>[454]</sup>，如下：

$$\text{score}(x, y) = \frac{\log P(y|x)}{\text{lp}(y)} + \text{cp}(x, y) \quad (14.6)$$

$$\text{cp}(x, y) = \beta \cdot \sum_{i=1}^{|x|} \log(\min(\sum_j^{|y|} a_{ij}, 1)) \quad (14.7)$$

其中， $\text{cp}(x, y)$  表示覆盖度模型，它度量了**译文对源语言每个单词的覆盖程度**。 $\text{cp}(x, y)$  的定义中， $\beta$  是一需要自行设置的超参数， $a_{ij}$  表示源语言第  $i$  个位置与译文第  $j$  个位置的注意力权重，这样  $\sum_j^{|y|} a_{ij}$  就可以用来衡量源语言第  $i$  个单词中的信息被翻译的程度，如果它大于 1，表明翻译多了；如果小于 1，表明翻译少了。公式(14.7)会惩罚那些欠翻译的翻译假设。对覆盖度模型的一种改进形式是<sup>[472]</sup>：

$$\text{cp}(x, y) = \sum_{i=1}^{|x|} \log(\max(\sum_j^{|y|} a_{ij}, \beta)) \quad (14.8)$$

公式(14.8)将公式(14.7)中的向下截断方式改为了向上截断。这样，模型可以对过翻译（或重复翻译）有更好的建模能力。不过，这个模型需要在开发集上细致地调整  $\beta$ ，也带来了一定的额外工作量。此外，也可以将这种覆盖度单独建模并进行参数化，与翻译模型一同训练<sup>[473, 695, 696]</sup>。这样可以得到更加精细的覆盖度模型。

### 14.2.3 搜索终止条件

在机器翻译推断中，何时终止搜索是一个非常基础的问题。如第二章所述，系统研发者一方面希望尽可能遍历更大的搜索空间，找到更好的结果，另一方面也希望在尽可能短的时间内得到结果。这时搜索的终止条件就是一个非常关键的指标。在

束搜索中有很多终止条件可以使用，比如，在生成一定数量的译文之后就终止搜索，或者当最佳译文与排名第二的译文之间的分数差距超过一个阈值时就终止搜索等。

在统计机器翻译中，搜索的终止条件相对容易设计。因为所有的翻译结果都可以用相同步骤的搜索过程生成，比如，在 CYK 推断中搜索的步骤仅与构建的分析表大小有关。在神经机器翻译中，这个问题要更加复杂。当系统找到一个完整的译文之后，可能还有很多译文没有被生成完，这时就面临着一个问题——如何决定是否继续搜索。

针对这些问题，研究人员设计了很多新的方法。比如，可以在束搜索中使用启发性信息让搜索尽可能早地停止，同时保证搜索结果是“最优的”<sup>[57]</sup>。也可以将束搜索建模为优化问题<sup>[58, 697]</sup>，进而设计出新的终止条件<sup>[698]</sup>。很多开源机器翻译系统也都使用了简单有效的终止条件，比如，在 OpenNMT 系统中当搜索束中当前最好的假设生成了完整的译文搜索就会停止<sup>[692]</sup>，在 RNNSearch 系统中当找到预设数量的译文时搜索就会停止，同时在这个过程中会不断减小搜索束的大小<sup>[22]</sup>。

实际上，设计搜索终止条件反映了搜索时延和搜索精度的一种折中<sup>[699, 700]</sup>。在很多应用中，这个问题会非常关键。比如，在同声传译中，对于输入的长文本，何时开始翻译、何时结束翻译都是十分重要的<sup>[701, 702]</sup>。在很多线上翻译应用中，翻译结果的响应不能超过一定的时间，这时就需要一种时间受限的搜索 (Time-constrained Search) 策略<sup>[669]</sup>。

#### 14.2.4 译文多样性

机器翻译系统的输出并不仅限于单个译文。很多情况下，需要多个译文。比如，译文重排序中通常就需要系统的  $n$ -best 输出，在交互式机器翻译中也往往需要提供多个译文供用户选择<sup>[643, 703]</sup>。但是，无论是统计机器翻译还是神经机器翻译，都面临一个同样的问题： $n$ -best 输出中的译文十分相似。实例14.1就展示了一个神经机器翻译输出的多个翻译结果，可以看到这些译文的区别很小。这个问题也被看做是机器翻译缺乏译文多样性的问题<sup>[394, 704, 705, 706, 707]</sup>。

**实例 14.1** 源语言句子：我们/期待/安理会/尽早/就此/作出/决定/。

机器译文 1：We look forward to the Security Council making a decision on this as soon as possible .

机器译文 2：We look forward to the Security Council making a decision on this issue as soon as possible .

机器译文 3：We hope that the Security Council will make a decision on this issue as soon as possible .

机器翻译输出缺乏多样性会带来很多问题。一个直接的问题是在重排序时很难选择到更好的译文，因为所有候选都没有太大的差别。此外，当需要利用  $n$ -best 输出来表示翻译假设空间时，缺乏多样性的译文也会使得翻译后验概率的估计不够准确，造成建模的偏差。在一些模型训练方法中，这种后验概率估计的偏差也会造成

较大的影响<sup>[233]</sup>。从人工翻译的角度，同一个源语言句子的译文应该是多样的，因此过于相似的译文也无法反映足够多的翻译现象。

因此增加译文多样性成为了机器翻译中一个有价值的研究方向。在统计机器翻译中就有很多尝试<sup>[394, 706, 707]</sup>。主要思路是通过加入一些“扰动”让翻译模型的行为发生变化，进而得到区别更大的译文。类似的方法也同样适用于神经机器翻译。例如，可以在推断过程中引入额外的模型，用于惩罚出现相似译文的情况<sup>[705, 708]</sup>。也可以在翻译模型中引入新的隐含变量或者加入新的干扰，进而控制多样性译文的输出<sup>[709, 710, 711]</sup>。类似地，也可以利用模型中局部结构的多样性来生成多样的译文<sup>[712]</sup>。除了考虑每个译文之间的多样性，也可以对译文进行分组，之后增加不同组之间的多样性<sup>[713]</sup>。

#### 14.2.5 搜索错误

机器翻译的错误分为两类：搜索错误和模型错误。搜索错误是指由于搜索算法的限制，即使潜在的搜索空间中有更好的解，模型也无法找到。比较典型的例子是，在对搜索结果进行剪枝的时候，如果剪枝过多，找到的结果很有可能不是最优的，这时就出现了搜索错误。而模型错误则是指由于模型学习能力的限制，即使搜索空间中存在最优解，模型也无法将该解排序在前面。

在统计机器翻译中，搜索错误可以通过减少剪枝进行缓解。比较简单的方式是增加搜索束宽度，这往往会有一定的性能提升<sup>[714]</sup>。也可以对搜索问题进行单独建模，以保证学到的模型出现更少的搜索错误<sup>[715, 716]</sup>。但是，在神经机器翻译中，这个问题却表现出不同的现象：在很多神经机器翻译系统中，随着搜索束的增大，系统的BLEU不升反降。图14.2展示了神经机器翻译系统中BLEU随搜索束大小的变化曲线，这里为了使该图更加规整直观，横坐标处将束大小进行了取对数操作。这个现象与传统的常识是相违背的，因此也有一些研究尝试解释这个现象<sup>[670, 717]</sup>。

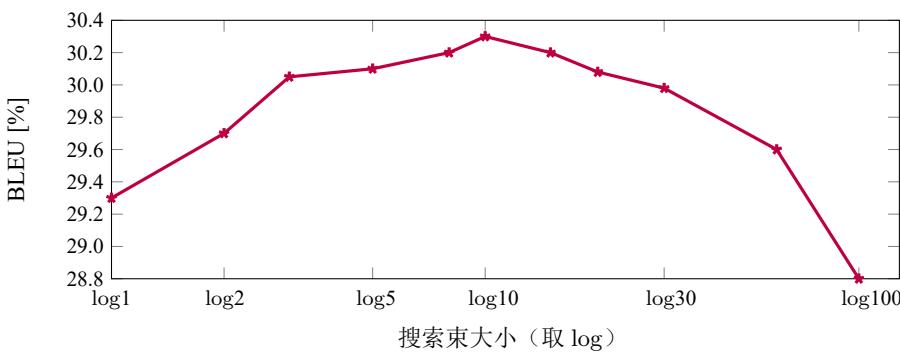


图 14.2 一个神经机器翻译系统中搜索束大小对 BLEU 的影响<sup>[718]</sup>

在实验中，研究人员发现增加搜索束的大小会导致翻译生成的结果变得更短。他们将这个现象归因于：神经机器翻译的建模基于局部归一的最大似然估计，增加搜索束的大小，会导致更多的模型错误<sup>[455, 689, 690]</sup>。此外，也有研究人员把这种翻译过短的现象归因于搜索错误<sup>[670]</sup>：由于搜索时所面临的搜索空间是十分巨大的，因此搜索

时可能无法找到模型定义的“最好”的译文，在某种意义上，这也体现了训练和推断不一致的问题（见第十三章）。一种解决该问题的思路是从“训练和推断行为不一致”的角度切入。比如，为了解决曝光偏置问题<sup>[598]</sup>，可以让系统使用前面步骤的预测结果作为预测下一个词所需要的历史信息，而不是依赖于标准答案<sup>[597, 719]</sup>。为了解决训练和推断目标不一致的问题，可以在训练的时候模拟推断的行为，同时让模型训练的目标与评价系统的标准尽可能一致<sup>[233]</sup>。

此外，还有其它方法解决增大搜索束造成翻译品质下降的问题。比如，通过对结果重排序来缓解这个问题<sup>[58]</sup>，也可以通过设计更好的覆盖度模型来生成长度更加合理的译文<sup>[472]</sup>。从这个角度说，上述问题的成因也较为复杂，因此需要同时考虑模型错误和搜索错误。

## 14.3 轻量模型

翻译速度和翻译精度之间的平衡是机器翻译系统研发中的常见问题。即使是以提升翻译品质为目标的任务（如用 BLEU 进行评价），也不得不考虑翻译速度的影响。比如，在很多任务中会构造伪数据，该过程涉及对大规模单语数据的翻译；无监督机器翻译中也会频繁地使用神经机器翻译系统构造训练数据。在这些情况下，如果翻译速度过慢会增大实验的周期。从应用的角度看，在很多场景下翻译速度甚至比翻译品质更重要。比如，在线翻译和一些小设备上的机器翻译系统都需要保证相对低的翻译时延，以满足用户体验的最基本要求。虽然，我们希望能有一套又好又快的翻译系统，但是现实的情况是：往往需要通过牺牲一些翻译品质来换取翻译速度的提升。下面就列举一些常用的神经机器翻译轻量模型和加速方法。这些方法通常应用在神经机器翻译的解码器上，因为相比编码器，解码器是推断过程中最耗时的部分。

### 14.3.1 输出层的词汇选择

神经机器翻译需要对输入和输出的单词进行分布式表示。但是，由于真实的词表通常很大，因此计算并保存这些单词的向量表示会消耗较多的计算和存储资源。特别是对于基于 Softmax 的输出层，大词表的计算十分耗时。虽然可以通过 BPE 和限制词汇表规模的方法降低输出层计算的负担<sup>[89]</sup>，但是为了获得可接受的翻译品质，词汇表也不能过小，因此输出层的计算代价仍然很高。

通过改变输出层的结构，可以一定程度上缓解这个问题<sup>[466]</sup>。一种比较简单的方法是对可能输出的单词进行筛选，即词汇选择。这里，可以利用类似于统计机器翻译的翻译表，获得每个源语言单词最可能的译文。在翻译过程中，利用注意力机制找到每个目标语言位置对应的源语言位置，之后获得这些源语言单词最可能的翻译候选。之后，Softmax 只需要在这个有限的翻译候选单词集合上进行计算，大大降低了输出层的计算量。尤其对于 CPU 上的系统，这个方法往往会带来明显的速度提升。图14.3对比了标准 Softmax 与词汇选择方法中的 Softmax。

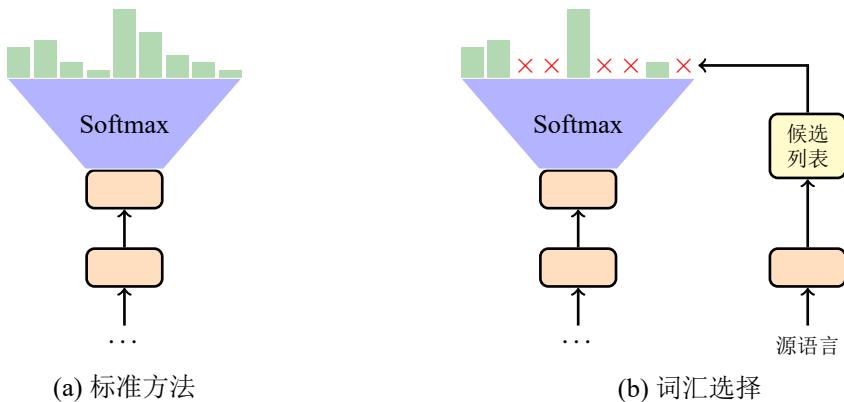


图 14.3 标准 Softmax vs 基于词汇选择的 Softmax

实际上，词汇选择也是一种典型的处理大词表的方法（见第十三章）。这种方法最大的优点在于，它可以与其它方法结合，比如与 BPE 等方法结合。本质上，这种方法与传统基于统计的机器翻译中的短语表剪枝有类似之处<sup>[328, 329, 330]</sup>，当翻译候选过多的时候，可以根据翻译候选对候选集进行剪枝。这种技术已经在统计机器翻译系统中得到成功应用。

### 14.3.2 消除冗余计算

消除不必要的计算是加速机器翻译系统的另一种方法。比如，在统计机器翻译时代，假设重组就是一种典型的避免冗余计算的手段（见第七章）。在神经机器翻译中，消除冗余计算的一种简单有效的方法是对解码器的注意力结果进行缓存。以Transformer为例，在生成每个译文时，Transformer模型会对当前位置之前的所有位置进行自注意力操作，但是这些计算里只有和当前位置相关的计算是“新”的，前面位置之间的注意力结果已经在之前的解码步骤里计算过，因此可以对其进行缓存。

此外，由于 Transformer 模型较为复杂，还存在很多冗余。比如，Transformer 的每一层会包含自注意力机制、层正则化、残差连接、前馈神经网络等多种不同的结构。同时，不同结构之间还会包含一些线性变换。多层 Transformer 模型会更加复杂。但是，这些层可能在做相似的事情，甚至有些计算根本就是重复的。图14.4中展示了解码器自注意力和编码-解码注意力中不同层的注意力权重的相似性，这里的相似性利用 Jensen-Shannon 散度进行度量<sup>[720]</sup>。可以看到，自注意力中，2-6 层之间的注意力权重的分布非常相似。编码-解码注意力也有类似的现象，临近的层之间有非常相似的注意力权重。这个现象说明：在多层神经网络中有些计算是冗余的，因此很自然的想法是消除这些冗余使得机器翻译变得更“轻”。

一种消除冗余计算的方法是将不同层的注意力权重进行共享，这样顶层的注意力权重可以复用底层的注意力权重<sup>[538]</sup>。在编码-解码注意力中，由于注意力机制中输入



图 14.4 自注意力和编码-解码注意力中不同层之间注意力权重的相似性（深色表示相似）

入的 Value 都是一样的<sup>3</sup>，甚至可以直接复用前一层注意力计算的结果。图14.5给出了不同方法的对比，其中  $S$  表示注意力权重， $A$  表示注意力模型的输出。可以看到，使用共享的思想，可以大大减少冗余的计算。

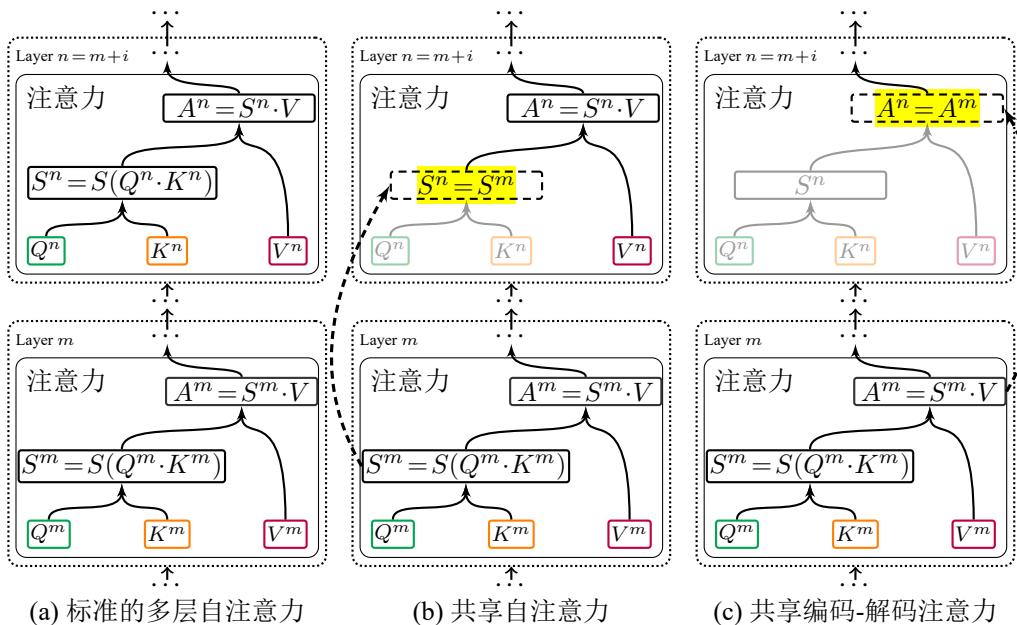


图 14.5 标准的多层自注意力、共享自注意力、共享编码-解码注意力方法的对比<sup>[538]</sup>

另一种方法是对不同层的参数进行共享。这种方法虽然不能带来直接的提速，但是可以大大减小模型的体积。比如，可以重复使用同一层的参数完成多层的计算。极端情况下，六层网络可以只使用一层网络的参数<sup>[721]</sup>。不过，在深层模型中（层数 > 20），浅层部分的差异往往较大，而深层（远离输入）之间的相似度会更高。这时可

<sup>3</sup>在Transformer解码器，编码-解码注意力输入的Value是编码器的输出，因此是相同的（见第十二章）。

以考虑对深层的部分进行更多的共享。

减少冗余计算也代表了一种剪枝的思想。本质上，这类方法利用了模型参数的稀疏性假设<sup>[722, 723]</sup>：一部分参数对模型整体的行为影响不大，因此可以直接被抛弃掉。这类方法也被使用在神经机器翻译模型的不同部分。比如，对于 Transformer 模型，也有研究发现多头注意力中的有些头是有冗余的<sup>[724]</sup>，因此可以直接对其进行剪枝<sup>[539]</sup>。

### 14.3.3 轻量解码器及小模型

在推断时，神经机器翻译的解码器是最耗时的，因为每个目标语言位置需要单独输出单词的分布，同时在搜索过程中每一个翻译假设都要被扩展成多个翻译假设，进一步增加了计算量。因此，提高推断速度的一种思路是使用更加轻量的解码器加快翻译假设的生成速度<sup>[547, 725]</sup>。

比较简单做法是把解码器的网络变得更“浅”、更“窄”。所谓浅网络是指使用更少的层构建神经网络，比如，使用 3 层，甚至 1 层网络的 Transformer 解码器。所谓窄网络是指将网络中某些层中神经元的数量减少。不过，直接训练这样的小模型会造成翻译品质下降。这时会考虑使用知识蒸馏等技术来提升小模型的品质（见第十三章）。

化简 Transformer 解码器的神经网络也可以提高推断速度。比如，可以使用平均注意力机制代替原始 Transformer 中的自注意力机制<sup>[540]</sup>，也可以使用运算更轻的卷积操作代替注意力模块<sup>[507]</sup>。前面提到的基于共享注意力机制的模型也是一种典型的轻量模型<sup>[538]</sup>。这些方法本质上也是对注意力模型结构的优化，这类思想在近几年也受到了很多关注<sup>[543, 726, 727]</sup>，在第十五章也会有进一步讨论。

此外，使用异构神经网络也是一种平衡精度和速度的有效方法。在很多研究中发现，基于 Transformer 的编码器对翻译品质的影响更大，而解码器的作用会小一些。因此，一种想法是使用速度更快的解码器结构，比如，用基于循环神经网络的解码器代替 Transformer 模型中基于注意力机制的解码器<sup>[458]</sup>。这样，既能发挥 Transformer 在编码上的优势，同时也能利用循环神经网络在解码器速度上的优势。使用类似的思想，也可以用卷积神经网络等结构进行解码器的设计。

针对轻量级 Transformer 模型的设计也包括层级的结构剪枝，这类方法试图通过跳过某些操作或者某些层来降低计算量。典型的相关工作是样本自适应神经网络结构，如 FastBERT<sup>[728]</sup>、Depth Adaptive Transformer<sup>[729]</sup> 等，与传统的 Transformer 的解码过程不同，这类神经网络结构在推断时不需要计算全部的解码层，而是根据输入自动选择模型的部分层进行计算，达到加速和减少参数量的目的。

### 14.3.4 批量推断

深度学习时代下，使用 GPU 已经成为大规模使用神经网络方法的前提。特别是对于机器翻译这样的复杂任务，GPU 的并行运算能力会带来明显的速度提升。为了充分利用 GPU 的并行能力，可以同时对多个句子进行翻译，即**批量推断**（Batch

Inference)。

在第十章已经介绍了神经机器翻译中批量处理的基本概念，其实现并不困难，不过有两方面问题需要注意：

- **批次生成策略。**对于源语言文本预先给定的情况，通常是按句子长度组织每个批次，即：把长度相似的句子放到一个批次里。这样做的好处是可以尽可能保证一个批次中的内容是“满”的，否则如果句长差异过大造成批次中有很多位置用占位符填充，产生无用计算。对于实时翻译的情况，批次的组织较为复杂。在机器翻译系统的实际应用中，由于有翻译时延的限制，可能待翻译句子未积累到标准批次数量就要进行翻译。常见的做法是，设置一个等待的时间，在同一个时间段中的句子可以放到一个批次中（或者几个批次中）。对于高并发的情况，也可以考虑使用不同的**桶**（Bucket）保存不同长度范围的句子，之后将同一个桶中的句子进行批量推断。这个问题在第十八章中还会做进一步讨论。
- **批次大小的选择。**一个批次中的句子数量越多，GPU设备的利用率越高，系统吞吐越大。但是，一个批次中所有句子翻译结束后才能拿到翻译结果，因此批次中有些句子即使已经翻译结束也要等待其它没有完成的句子。也就是说，从单个句子来看，批次越大翻译的延时越长，这也导致在翻译实时性要求较高的场景中，不能使用过大的批次。而且，大批次对GPU显存的消耗更大，因此也需要根据具体任务合理选择批次大小。为了说明这些问题，图14.6展示了不同批次大小下的时延和显存消耗。

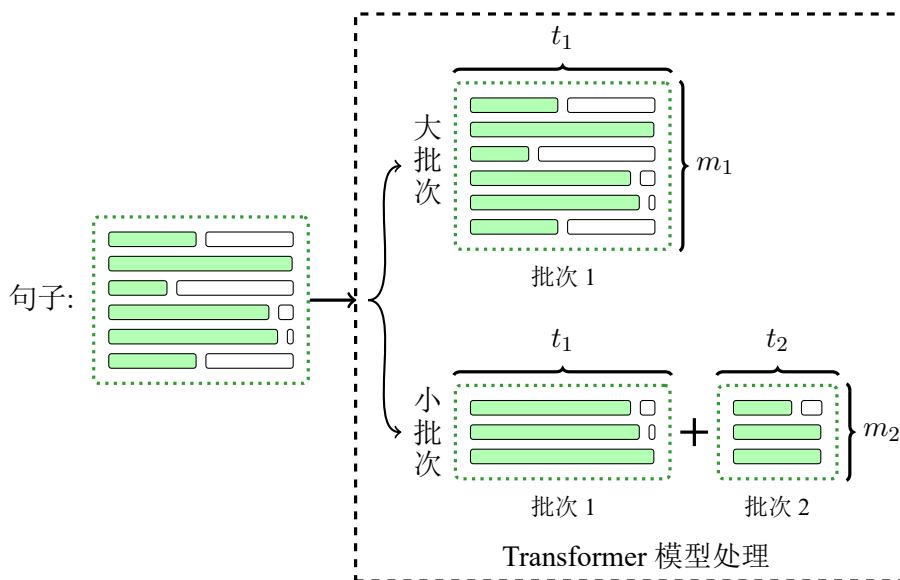


图 14.6 神经机器翻译系统在不同批次大小下的延时和显存消耗

### 14.3.5 低精度运算

降低运算强度也是计算密集型任务的加速手段之一。标准的神经机器翻译系统大多基于单精度浮点运算。从计算机的硬件发展看，单精度浮点运算还是很“重”的。当计算能容忍一些精度损失的时候，可以考虑降低运算精度来达到加速的目的。比如：

- **半精度浮点运算。**半精度浮点运算是随着近几年 GPU 技术发展而逐渐流行的一种运算方式。简单来说，半精度的表示要比单精度需要更少的存储单元，所表示的浮点数范围也相应的变小。不过，实践中已经证明神经机器翻译中的许多运算用半精度计算就可以满足对精度的要求。因此，直接使用半精度运算可以大大加速系统的训练和推断进程，同时对翻译品质的影响很小。不过，需要注意的是，在分布式训练时，由于参数服务器需要对多个计算节点上的梯度进行累加，因此保存参数时仍然会使用单精度浮点以保证多次累加之后不会造成过大的精度损失。
- **整型运算。**整型运算是一种比浮点运算“轻”很多的运算。无论是芯片占用面积、能耗还是处理单次运算的时钟周期数，相比浮点运算，整型运算都有着明显的优势。不过，整数的表示和浮点数有着很大的不同。一个基本问题是，整数是不连续的，因此无法准确地刻画浮点数中很小的小数。对于这个问题，一种解决方法是利用“量化 + 反量化 + 缩放”的策略让整型运算达到与浮点运算近似的效果<sup>[545, 730, 731]</sup>。所谓“量化”就是把一个浮点数离散化为一个整数，“反量化”是这个过程的逆过程。由于浮点数可能超出整数的范围，因此会引入一个缩放因子：在量化前将浮点数缩放到整数可以表示的范围，反量化前再缩放回原始浮点数的表示范围。这种方法在理论上可以带来很好的加速效果。不过由于量化和反量化的操作本身也有时间消耗，而且在不同处理器上的表现差异较大。因此不同实现方式带来的加速效果并不相同，需要通过实验测算。
- **低精度整型运算。**使用更低精度的整型运算是进一步加速的手段之一。比如使用 16 位整数、8 位整数，甚至 4 位整数在理论上都会带来速度的提升，如表14.2 所示。不过，并不是所有处理器都支持低精度整型的运算。开发这样的系统，一般需要硬件和特殊低精度整型计算库的支持。而且相关计算大多是在 CPU 上实现，应用会受到一定的限制。

表 14.2 不同计算精度的运算速度对比<sup>4</sup>

指标	FP32	INT32	INT16	INT8	INT4
速度	1×	3~4×	≈4×	4~6×	≈8×

<sup>4</sup>表中比较了几种通用数据类型的乘法运算速度，不同硬件和架构上不同类型的数据的计算速度略有不同。总体来看整型数据类型和浮点型数据相比具有显著的计算速度优势，INT4 相比于 FP32 数据

实际上，低精度运算的另一个好处是可以减少模型存储的体积。比如，如果要把机器翻译模型作为软件的一部分打包存储，这时可以考虑用低精度的方式保存模型参数，使用时再恢复成原始精度的参数。值得注意的是，参数的离散化表示（比如整型表示）的一个极端例子是**二值网络**（Binarized Neural Networks）<sup>[732]</sup>，即只用 -1 和 +1 表示神经网络的每个参数<sup>5</sup>。二值化可以被看作是一种极端的量化手段。不过，这类方法还没有在机器翻译中得到大规模验证。

## 14.4 非自回归翻译

目前大多数神经机器翻译模型都使用自左向右逐词生成译文的策略，即第  $j$  个目标语言单词的生成依赖于先前生成的  $j-1$  个词。这种翻译方式也被称作**自回归解码**（Autoregressive Decoding）。虽然以 Transformer 为代表的模型使得训练过程高度并行化，加快了训练速度。但由于推断过程自回归的特性，模型无法同时生成译文中的所有单词，导致模型的推断过程非常缓慢，这对于神经机器翻译的实际应用是个很大的挑战。因此，如何设计一个在训练和推断阶段都能够并行化的模型是目前研究的热点之一。

### 14.4.1 自回归 vs 非自回归

目前主流的神经机器翻译的推断是一种**自回归翻译**（Autoregressive Translation）过程。所谓自回归是一种描述时间序列生成的方式：对于目标序列  $y = \{y_1, \dots, y_n\}$ ，如果  $j$  时刻状态  $y_j$  的生成依赖于之前的状态  $\{y_1, \dots, y_{j-1}\}$ ，而且  $y_j$  与  $\{y_1, \dots, y_{j-1}\}$  构成线性关系，那么称目标序列  $y$  的生成过程是自回归的。神经机器翻译借用了这个概念，但是并不要求  $y_j$  与  $\{y_1, \dots, y_{j-1}\}$  构成线性关系，14.2.1 节提到的自左向右翻译模型和自右向左翻译模型都属于自回归翻译模型。自回归模型在机器翻译任务上也有很好的表现，特别是配合束搜索往往能够有效地寻找近似最优译文。但是，由于解码器的每个步骤必须顺序地而不是并行地运行，自回归翻译模型会阻碍不同译文单词生成的并行化。特别是在 GPU 上，翻译的自回归性会大大降低计算的并行度和设备利用率。

对于这个问题，研究人员也考虑移除翻译的自回归性，进行**非自回归翻译**（Non-Autoregressive Translation, NAT）<sup>[271]</sup>。一个简单的**非自回归翻译**模型将问题建模为公式(14.9)：

$$P(y|x) = \prod_{j=1}^n P(y_j|x) \quad (14.9)$$

对比公式(14.1)可以看出，公式(14.9)中位置  $j$  上的输出  $y_j$  只依赖于输入句子  $x$ ，

---

类型的计算最高能达到 8 倍的速度提升。

<sup>5</sup> 也存在使用 0 或 1 表示神经网络参数的二值网络。

与其它位置上的输出无关。于是，可以并行生成所有位置上的  $y_j$ 。理想情况下，这种方式一般可以带来几倍甚至十几倍的速度提升。

#### 14.4.2 非自回归翻译模型的结构

在介绍非自回归模型的具体结构之前，先来看看如何实现一个简单的非自回归翻译模型。这里用标准的 Transformer 来举例。首先为了一次性生成所有的词，需要丢弃解码器对未来信息屏蔽的矩阵，从而去掉模型的自回归性。此外，还要考虑生成译文的长度。在自回归模型中，每步的输入是上一步解码出的结果，当预测到终止符  $\langle \text{eos} \rangle$  时，序列的生成就自动停止了，然而非自回归模型却没有这样的特性，因此还需要一个长度预测器来预测出其长度，之后再用这个长度得到每个位置的表示，将其作为解码器的输入，进而完成整个序列的生成。

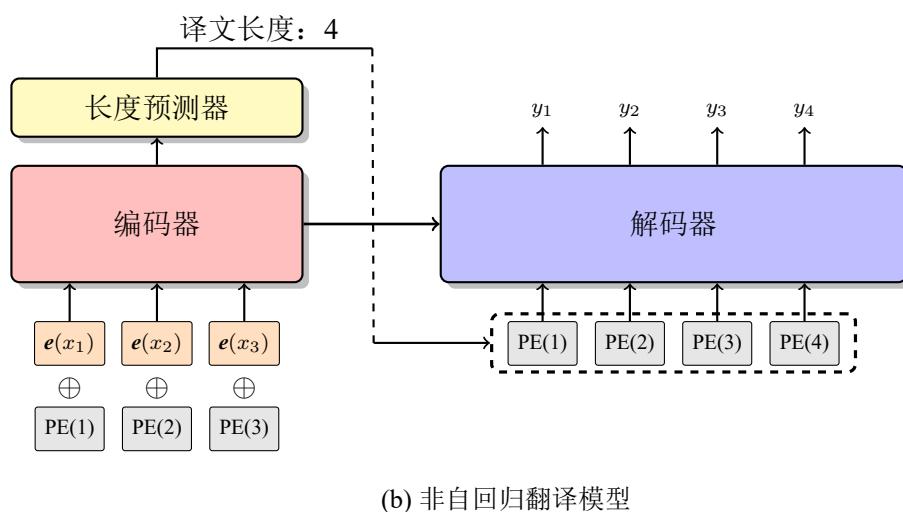
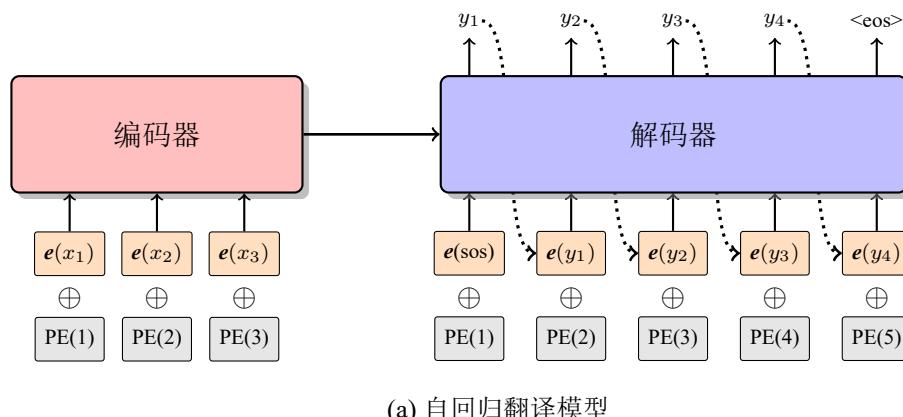


图 14.7 自回归翻译模型 vs 非自回归翻译模型

图14.7对比了自回归翻译模型和简单的非自回归翻译模型。可以看到这种自回归翻译模型可以一次性生成完整的译文。不过，高并行性也带来了翻译品质的下降。

比如，在 IWSLT 英德等数据上的 BLEU[%] 值只有个位数，而现在最好的自回归模型已经能够达到 30 左右的 BLEU 得分。这是因为每个位置词的预测只依赖于源语言句子  $x$ ，使得预测不准确。需要注意的是，图14.7(b) 中将位置编码作为非自回归模型解码器的输入只是一个最简单的例子，在真实的系统中，**非自回归解码器的输入一般是拷贝的源语言句子词嵌入与位置编码的融合**。

完全独立地对每个词建模，会出现什么问题呢？来看一个例子，将汉语句子“干/得/好/！”翻译成英文，可以翻译成“Good job！”或者“Well done！”。假设生成这两种翻译的概率是相等的，即一半的概率是“Good job！”，另一半的概率是“Well done！”。由于非自回归模型的条件独立性假设，推断时第一个词“Good”和“Well”的概率是差不多大的，如果第二个词“job”和“done”的概率也差不多大，会使得模型生成出“Good done！”或者“Well job！”这样错误的翻译，如图14.8所示。这便是影响句子质量的关键问题，称之为**多峰问题**（Multimodality Problem）<sup>[271]</sup>。如何有效处理非自回归模型中的多峰问题是提升非自回归模型质量的关键。

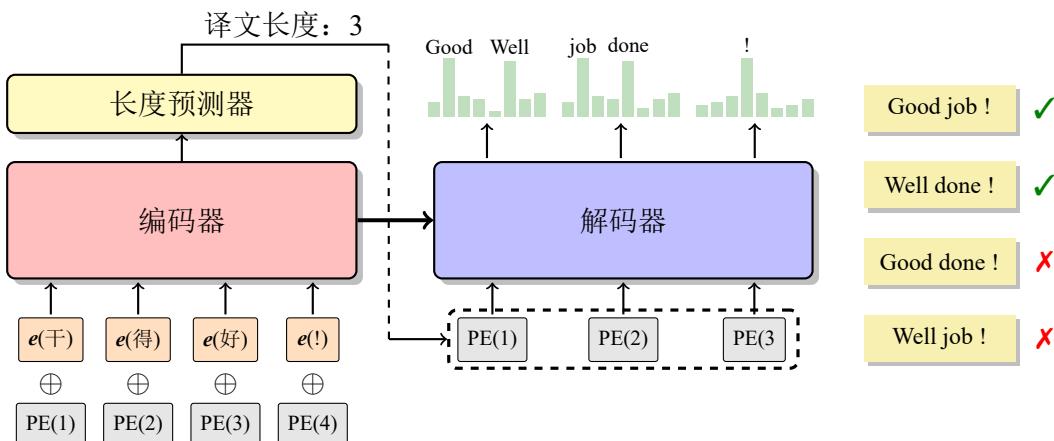


图 14.8 非自回归模型中的多峰问题

因此，非自回归翻译的研究大多集中在针对以上问题的求解。有三个角度：使用繁衍率预测译文长度、使用句子级知识蒸馏来降低学习难度、使用自回归模型进行翻译候选打分。下面将依次对这些方法进行介绍。

## 1. 基于繁衍率的非自回归模型

图14.9给出了基于繁衍率的Transformer非自回归模型的结构<sup>[271]</sup>，由三个模块组成：编码器，解码器，繁衍率预测器。类似于标准的Transformer模型，这里编码器和解码器都完全由前馈神经网络和多头注意力模块组成。唯一的不同是解码器中新增了**位置注意力模块**（图14.9中被红色虚线框住的模块），用于更好的捕捉目标语言端的位置信息。

繁衍率预测器的一个作用是预测整个译文句子的长度，以便并行地生成所有译文单词。通过对每个源语言单词计算繁衍率来估计最终译文的长度。具体来说，

繁衍率指的是：根据每个源语言单词预测出其对应的目标语言单词的个数（见第六章），如图14.9所示，翻译过程中英语单词“*We*”对应一个汉语单词“我们”，其繁衍率为1。于是，可以得到源语言句子对应的繁衍率序列（图14.9中的数字1 1 2 0 1），最终译文长度则由源语言单词的繁衍率之和决定。之后将源语言单词按该繁衍率序列进行拷贝，在图中的例子中，将“*We*”、“totally”、“.”拷贝一次，将“accept”、“it”分别拷贝两次和零次，就得到了最终解码器的输入“*We totally accept accept .*”。在模型训练阶段，繁衍率序列可以通过外部词对齐工具得到，用于之后训练繁衍率预测器。但由于外部词对齐系统会出现错误，因此在模型收敛之后，可以对繁衍率预测器进行额外的微调。

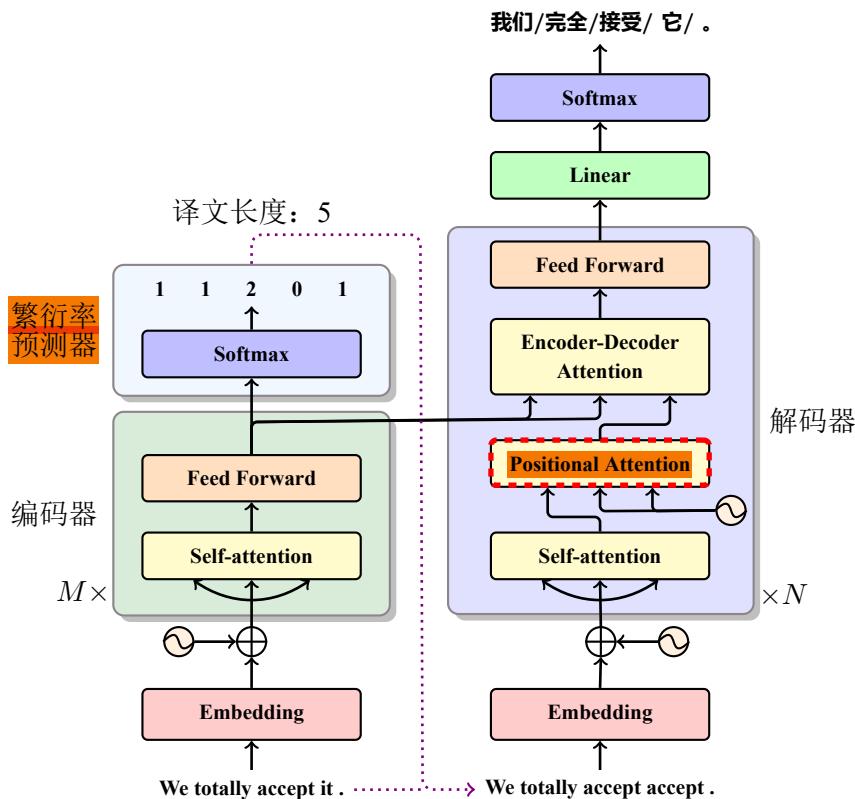


图 14.9 基于繁衍率的非自回归模型

实际上，使用繁衍率的另一个好处在于可以缓解多峰问题。因为，繁衍率本身可以看作是模型的一个隐变量。使用这个隐变量本质上是在对可能的译文空间进行剪枝，因为只有一部分译文满足给定的繁衍率序列。从这个角度说，在翻译率的作用下，不同单词译文组合的情况变少了，因此多峰问题也就被缓解了。

另外，在每个解码器层中还新增了额外的位置注意力模块，该模块与其它部分中使用的多头注意力机制相同。其仍然基于  $Q$ 、 $K$ 、 $V$  之间的计算（见第十二章），只是把位置编码作为  $Q$  和  $K$ ，解码器端前一层的输出作为  $V$ 。这种方法提供了更强的位

置信息。

## 2. 句子级知识蒸馏

知识蒸馏的基本思路是把教师模型的知识传递给学生模型，让学生模型可以更好地学习（见第十三章）。通过这种方法，可以降低非自回归模型的学习难度。具体来说，可以让自回归模型作为“教师”，非自回归模型作为“学生”。把自回归神经机器翻译模型生成的句子作为新的训练样本，送给非自回归机器翻译模型进行学习<sup>[680, 733, 734]</sup>。有研究发现自回归模型生成的结果的“确定性”更高，也就是不同句子中相同源语言片段翻译的多样性相对低一些<sup>[271]</sup>。虽然从人工翻译的角度看，这可能并不是理想的译文，但是使用这样的译文可以在一定程度上缓解多峰问题。因为，经过训练的自回归模型会始终将相同的源语言句子翻译成相同的译文。这样得到的数据集噪声更少，能够降低非自回归模型学习的难度。此外，相比人工标注的译文，自回归模型输出的译文更容易让模型进行学习，这也是句子级知识蒸馏有效的原因之一。

## 3. 自回归模型打分

通过采样不同的繁衍率序列，可以得到多个不同的翻译候选。之后，把这些不同的译文再交给自回归模型来评分，选择一个最好的结果作为最终的翻译结果。通常，这种方法能够很有效地提升非自回归翻译模型的译文质量，并且保证较高的推断速度<sup>[271, 735, 736, 737, 738]</sup>。但是，缺点是需要同时部署自回归和非自回归两套系统。

### 14.4.3 更好的训练目标

虽然非自回归翻译可以显著提升翻译速度，但是很多情况下其翻译质量还是低于传统的自回归翻译<sup>[271, 734, 739]</sup>。因此，很多工作致力于缩小自回归模型和非自回归模型的性能差距<sup>[740, 741, 742]</sup>。

一种直接的方法是层级知识蒸馏<sup>[743]</sup>。由于自回归模型和非自回归模型的结构相差不大，因此可以将翻译质量更高的自回归模型作为“教师”，通过给非自回归模型提供监督信号，使其逐块地学习前者的分布。研究人员发现了两点非常有意思的现象：1) 非自回归模型容易出现“重复翻译”的现象，这些相邻的重复单词所对应的位置的隐藏状态非常相似。2) 非自回归模型的注意力分布比自回归模型的分布更加尖锐。这两点发现启发了研究人员使用自回归模型中的隐层状态和注意力矩阵等中间表示来指导非自回归模型的学习过程。可以计算两个模型隐层状态的距离以及注意力矩阵的 KL 散度<sup>6</sup>，将它们作为额外的损失指导非自回归模型的训练。类似的做法也出现在基于模仿学习的方法中<sup>[735]</sup>，它也可以被看作是对自回归模型不同层行为的模拟。不过，基于模仿学习的方法会使用更复杂的模块来完成自回归模型对非自回归模型的指导，比如，在自回归模型和非自回归模型中都使用一个额外的神经网

<sup>6</sup>KL 散度即相对熵。

络，用于传递自回归模型提供给非自回归模型的层级监督信号。

此外，也可以使用基于正则化因子的方法<sup>[737]</sup>。非自回归模型的翻译结果中存在着两种非常严重的错误：重复翻译和不完整的翻译。重复翻译问题是因为解码器隐层状态中相邻的两个位置过于相似，因此翻译出来的单词也一样。对于不完整翻译，即欠翻译问题，通常是由非自回归模型在翻译的过程中丢失了一些源语言句子的信息。针对这两个问题，可以通过在相邻隐层状态间添加相似度约束来计算一个重构损失。具体实践时，对于翻译  $x \rightarrow y$ ，通过一个反向的自回归模型再将  $y$  翻译成  $x'$ ，最后计算  $x$  与  $x'$  的差异性作为损失。

#### 14.4.4 引入自回归模块

非自回归翻译消除了序列生成过程中不同位置预测结果间的依赖，在每个位置都进行独立的预测，但这反而会导致翻译质量显著下降，因为缺乏不同单词间依赖关系的建模。因此，也有研究聚焦于在非自回归模型中添加一些自回归组件。

一种做法是将句法信息作为目标语言句子的框架<sup>[744]</sup>。具体来说，先自回归地预测出一个目标语言的句法块序列，将句法块作为序列信息的抽象，然后根据句法块序列非自回归地生成所有目标语言单词。如图14.10所示，该模型由一个编码器和两个解码器组成。其中编码器和第一个解码器与标准的 Transformer 模型相同，用来自回归地预测句法树信息；第二个解码器将第一个解码器的句法信息作为输入，之后再非自回归地生成整个译文。在训练过程中，通过使用外部句法分析器获得对句法预测任务的监督信号。虽然可以简单地让模型预测整个句法树，但是这种方法会显著增加自回归步骤的数量，从而增大时间开销。因此，为了维持句法信息与解码时间的平衡，这里预测一些由句法标记和子树大小组成的块标识符（如 VP3）而不是整个句法树。关于基于句法的神经机器翻译模型在第十五章还会有进一步讨论。

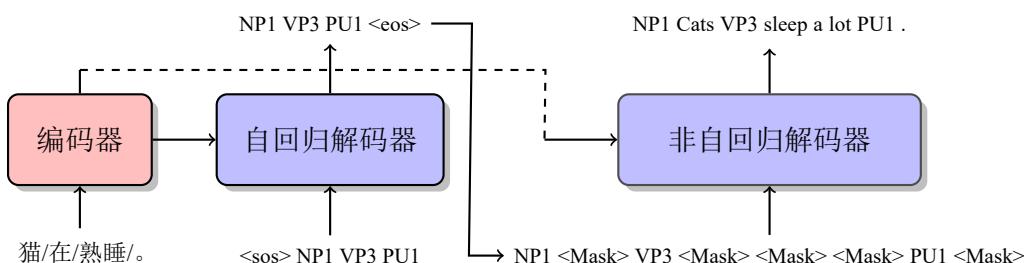


图 14.10 基于句法结构的非自回归模型

另一种做法是半自回归地生成译文<sup>[745]</sup>。如图14.11所示，自回归模型从左到右依次生成译文，具有“最强”的自回归性；而非自回归模型完全独立的生成每个译文单词，具有“最弱”的自回归性；半自回归模型则是将整个译文分成  $k$  个块，在块内执行非自回归解码，在块间则执行自回归的解码，能够在每个时间步并行产生多个连续的单词。通过调整块的大小，半自回归模型可以灵活的调整为自回归翻译（当  $k$  等于 1）和非自回归翻译（当  $k$  大于等于最大的译文长度）。

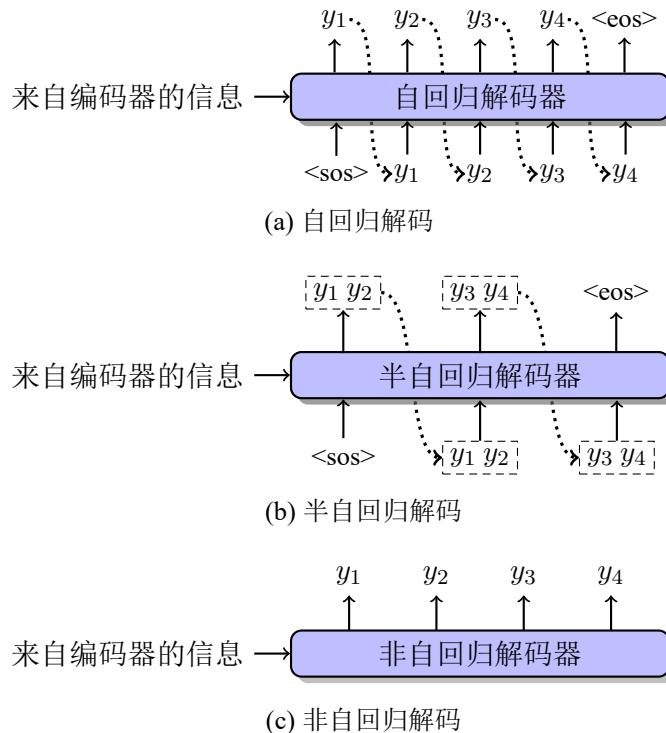


图 14.11 自回归、半自回归和非自回归解码对比<sup>[745]</sup>

还有一种做法引入了轻量级的自回归调序模块<sup>[746]</sup>。为了解决非自回归模型解码搜索空间过大的问题，可以使用调序技术在相对较少的翻译候选上进行自回归模型的计算。如图14.12所示，该方法对源语言句子进行重新排列转换成由源语言单词组成但位于目标语言结构中的伪译文，然后将伪译文进一步转换成目标语言以获得最终的翻译。其中，这个调序模块可以是一个轻量自回归模型，例如，一层的循环神经网络。

#### 14.4.5 基于迭代精化的非自回归翻译模型

如果一次并行生成整个序列，往往很难捕捉单词之间的关系，而且即便生成了错误的译文单词，这类方法也无法修改。针对这些问题，也可以使用迭代式的生成方式<sup>[680, 747, 748]</sup>。这种方法放弃了一次生成最终的译文句子，而是将解码出的译文再重新送给解码器，在每次迭代中来改进之前生成的译文单词，可以理解为句子级的自回归模型。这样做好处在于，每次迭代的过程中可以利用已经生成的部分翻译结果，来指导其它部分的生成。

图14.13展示了这种方法的简单示例。它拥有一个编码器和  $N$  个解码器。编码器首先预测出译文的长度，然后将输入  $x$  按照长度复制出  $x'$  作为第一个解码器的输入，之后生成  $y^{[1]}$  作为第一轮迭代的输出。接下来再把  $y^{[1]}$  输入给第二个解码器，然后输出  $y^{[2]}$ ，以此类推。那么迭代到什么时候结束呢？一种简单的做法是提前制定好迭代

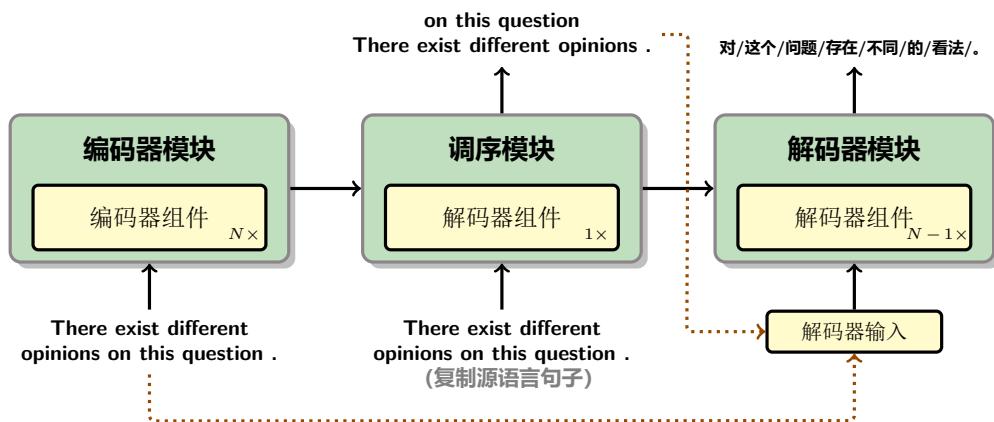


图 14.12 引入调序模块的非自回归模型

次数，这种方法能够自主地对生成句子的质量和效率进行平衡。另一种称之为“自适应”的方法，具体是通过计算当前生成的句子与上一次生成句子之间的变化量来判断是否停止，例如，使用杰卡德相似系数作为变化量函数<sup>7</sup>。另外，需要说明的是，图14.13中是使用多个解码器的一种逻辑示意。真实的系统仅需要一个解码器，并运行多次，就达到了迭代精化的目的。

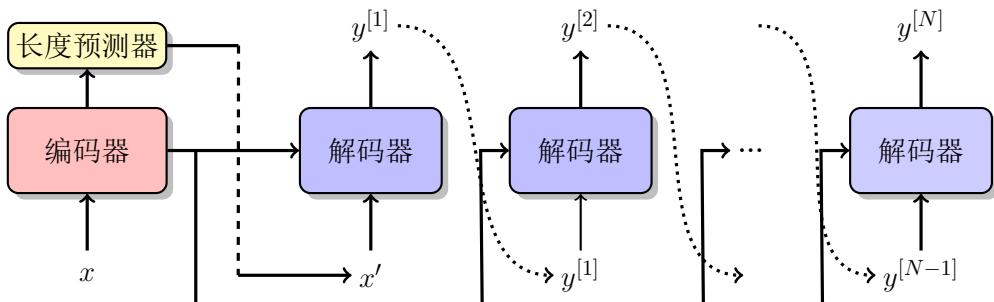


图 14.13 基于迭代精化的非自回归翻译模型运行示例

除了使用上一个步骤的输出，当前解码器的输入还可以使用了添加噪声的正确目标语言句子<sup>[680]</sup>。另外，对于译文长度的预测，也可以使用编码器的输出单独训练一个独立的长度预测模块，这种方法也推广到了目前大多数非自回归模型上。

另一种方法借鉴了 BERT 的思想<sup>[123]</sup>，称为 Mask-Predict<sup>[747]</sup>。类似于 BERT 中的 <CLS> 标记，该方法在源语言句子的最前面加上了一个特殊符号 <LEN> 作为输入，用来预测目标句的长度  $n$ 。之后，将特殊符 <Mask>（与 BERT 中的 <Mask> 有相似的含义）复制  $n$  次作为解码器的输入，然后用非自回归的方式生成所有的译文单词。这样生成的翻译可能是比较差的，因此可以将第一次生成的这些词中不确定（即生

<sup>7</sup> 杰卡德相似系数是衡量有限样本集之间的相似性与差异性的一种指标，杰卡德相似系数值越大，样本相似度越高。

成概率比较低)的一些词“擦”掉,依据剩余的译文单词以及源语言句子重新进行预测,不断迭代,直到满足停止条件为止。图14.14给出了一个示例。

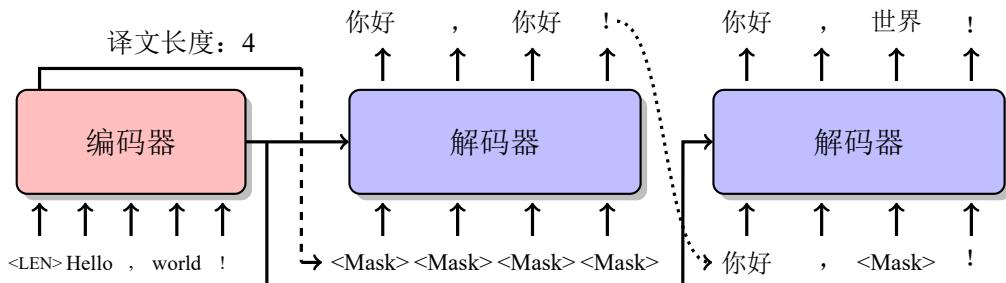


图 14.14 Mask-Predict 方法的运行示例

## 14.5 多模型集成

在机器学习领域,把多个模型融合成一个模型是提升系统性能的一种有效方法。比如,在经典的 AdaBoost 方法中<sup>[749]</sup>,用多个“弱”分类器构建的“强”分类器可以使模型在训练集上的分类错误率无限接近 0。类似的思想也被应用到机器翻译中<sup>[707, 750, 751, 752]</sup>,被称为**系统融合**(System Combination)。在各种机器翻译比赛中,系统融合已经成为经常使用的技术之一。由于许多模型融合方法都是在推断阶段完成,因此此类方法开发的代价较低。

广义上来讲,使用多个特征组合的方式都可以被看作是一种模型的融合。融合多个神经机器翻译系统的方法有很多,可以分为假设选择、局部预测融合、译文重组三类,下面分别进行介绍。

### 14.5.1 假设选择

**假设选择**(Hypothesis Selection)是最简单的系统融合方法<sup>[706]</sup>。其思想是:给定一个翻译假设集合,综合多个模型对每一个翻译假设进行打分,之后选择得分最高的假设作为结果输出。

假设选择中首先需要考虑的问题是假设生成。构建翻译假设集合是假设选择的第一步,也是最重要的一步。理想的情况下,这个集合应该尽可能包含更多高质量的翻译假设,这样后面有更大的几率选出更好的结果。不过,由于单个模型的性能是有上限的,因此无法期望这些翻译假设的品质超越单个模型的上限。研究人员更加关心的是翻译假设的多样性,因为已经证明**多样的翻译假设**非常有助于提升系统融合的性能<sup>[394, 753]</sup>。为了生成多样的翻译假设,通常有两种思路:1) 使用**不同的模型**生成翻译假设;2) 使用**同一个模型的不同参数和设置**生成翻译假设。图14.15展示了二者的区别。比如,可以使用基于循环神经网络的模型和Transformer模型生成不同的翻译假设,之后都放入集合中;也可以只用Transformer模型,但是用不同的模

型参数构建多个系统，之后分别生成翻译假设。在神经机器翻译中，经常采用的是第二种方式，因为系统开发的成本更低。

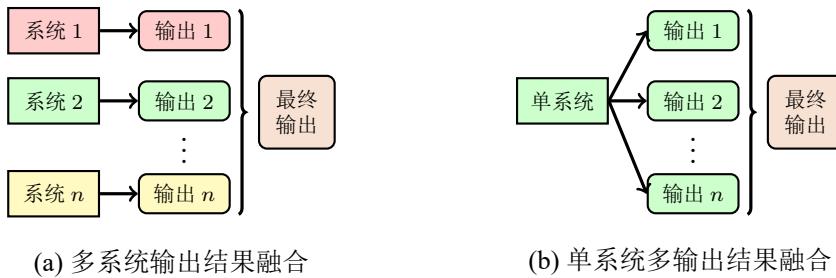


图 14.15 多模型翻译假设生成 vs 单模型翻译假设生成

此外，模型的选择也十分重要。所谓假设选择实际上就是要用一个更强的模型在候选中进行选择。这个“强”模型一般是由更多、更复杂的子模型组合而成。常用的方法是直接使用翻译假设生成时的模型构建“强”模型。比如，使用两个模型生成了翻译假设集合，之后对所有翻译假设都分别用这两个模型进行打分。最后，综合两个模型的打分（如线性插值）得到翻译假设的最终得分，并进行选择。当然，也可以使用更强大的统计模型对多个子模型进行组合，如使用更深、更宽的神经网络。

假设选择也可以被看作是一种简单的投票模型，对所有的候选用多个模型投票，选出最好的结果输出。包括重排序在内的很多方法也是假设选择的一种特例。比如，在重排序中，可以把生成  $n$ -best 列表的过程看作是翻译假设生成过程，而重排序的过程可以被看作是融合多个子模型进行最终结果选择的过程。

### 14.5.2 局部预测融合

神经机器翻译模型对每个目标语言位置  $j$  的单词的概率分布进行预测<sup>8</sup>，假设有  $K$  个神经机器翻译系统，那么每个系统  $k$  都可以独立计算这个概率分布，记为  $P_k(y_j|y_{<j}, x)$ 。于是，可以用如下方式融合这  $K$  个系统的预测：

$$P(y_j|y_{<j}, x) = \sum_{k=1}^K \gamma_k \cdot P_k(y_j|y_{<j}, x) \quad (14.10)$$

其中， $\gamma_k$  表示第  $k$  个系统的权重，且满足  $\sum_{k=1}^K \gamma_k = 1$ 。权重  $\{\gamma_k\}$  可以在开发集上自动调整，比如，使用最小错误率训练得到最优的权重（见第七章）。不过在实践中发现，如果这  $K$  个模型都是由一个基础模型衍生出来的，权重  $\{\gamma_k\}$  对最终结果的影响并不大。因此，有时候也简单的将权重设置为  $\gamma_k = \frac{1}{K}$ 。图14.16展示了对三个模型预测结果的集成。

公式(14.10)是一种典型的线性插值模型，这类模型在语言建模等任务中已经得

<sup>8</sup>即对于目标语言词汇表中的每个单词  $w_r$ ，计算  $P(y_j = w_r | y_{<j}, x)$ 。

$$\begin{array}{c}
 \begin{bmatrix} \text{Have 0.5} \\ \text{Has 0.1} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \text{Have 0.2} \\ \text{Has 0.3} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \text{Have 0.4} \\ \text{Has 0.3} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \\
 P_1 \qquad \qquad P_2 \qquad \qquad P_3 \qquad \qquad \Rightarrow \qquad \qquad \begin{bmatrix} \text{Have 0.37} \\ \text{Has 0.23} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \\
 P = \sum_{i=1}^3 \frac{1}{3} P_i
 \end{array}$$

图 14.16 基于三个模型预测结果的集成

到成功应用。从统计学习的角度，多个模型的插值可以有效地降低经验错误率。不过，多模型集成依赖一个假设：这些模型之间需要有一定的互补性。这种互补性有时也体现在多个模型预测的上限上，称为 Oracle。比如，可以把这  $K$  个模型输出中 BLEU 最高的结果作为 Oracle，也可以选择每个预测结果中使 BLEU 达到最高的译文单词，这样构成的句子作为 Oracle。当然，并不是说 Oracle 提高，模型集成的结果一定会变好。因为 Oracle 是最理想情况下的结果，而实际预测的结果与 Oracle 往往有很大差异。如何使用 Oracle 进行模型优化也是很多研究人员在探索的问题。

此外，如何构建集成用的模型也是非常重要的，甚至说这部分工作会成为模型集成方法中最困难的部分<sup>[671, 673, 754]</sup>。为了增加模型的多样性，常用的方法有：

- 改变模型宽度和深度，即用不同层数或者不同隐藏层大小得到多个模型；
- 使用不同的参数进行初始化，即用不同的随机种子初始化参数训练多个模型；
- 不同模型（局部）架构的调整，比如，使用不同的位置编码模型<sup>[460]</sup>、多层次融合模型<sup>[461]</sup>等；
- 利用不同数量以及不同数据增强方式产生的伪数据训练模型<sup>[755]</sup>；
- 利用多分支多通道的模型，不同分支可能有不同结构，使得模型能有更好的表示能力<sup>[755]</sup>；
- 利用预训练进行参数共享之后微调的模型。

### 14.5.3 译文重组

假设选择是直接从已经生成的译文中进行选择，因此无法产生“新”的译文，也就是它的输出只能是某个单模型的输出。此外，预测融合需要同时使用多个模型进行推断，对计算和内存消耗较大。而且这两种方法有一个共性问题：搜索都是基于一个个字符串，相比指数级的译文空间，所看到的结果还是非常小的一部分。对于这个问题，一种方法是利用更加紧凑的数据结构对指数级的译文串进行表示。比如，可以使用词格（Word Lattice）对多个译文串进行表示<sup>[756]</sup>。图14.17展示了基于  $n$ -best 词

串和基于词格的表示方法的区别。可以看到，词格中从起始状态到结束状态的每一条路径都表示一个译文，不同译文的不同部分可以通过词格中的节点得到共享<sup>9</sup>。理论上，词格可以把指数级数量的词串用线性复杂度的结构表示出来。

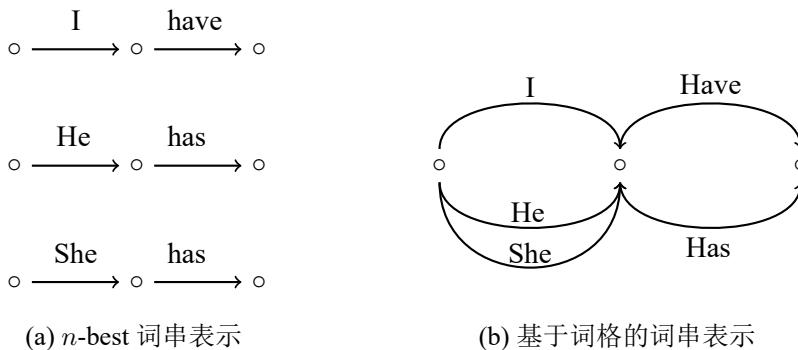


图 14.17 *n*-best 词串表示 vs 基于词格的词串表示

有了词格这样的结构，多模型集成又有了新的思路。首先，可以将多个模型的译文融合为词格。注意，这个词格会包含这些模型无法生成的完整译文句子。之后，用一个更强的模型在词格上搜索最优的结果。这个过程有可能找到一些“新”的译文，即结果可能是从多个模型的结果中重组而来的。词格上的搜索模型可以基于多模型的融合，也可以使用一个简单的模型，这里需要考虑的是将神经机器翻译模型适应到词格上进行推断<sup>[757]</sup>。其过程基本与原始的模型推断没有区别，只是需要把模型预测的结果附着到词格中的每条边上，再进行推断。

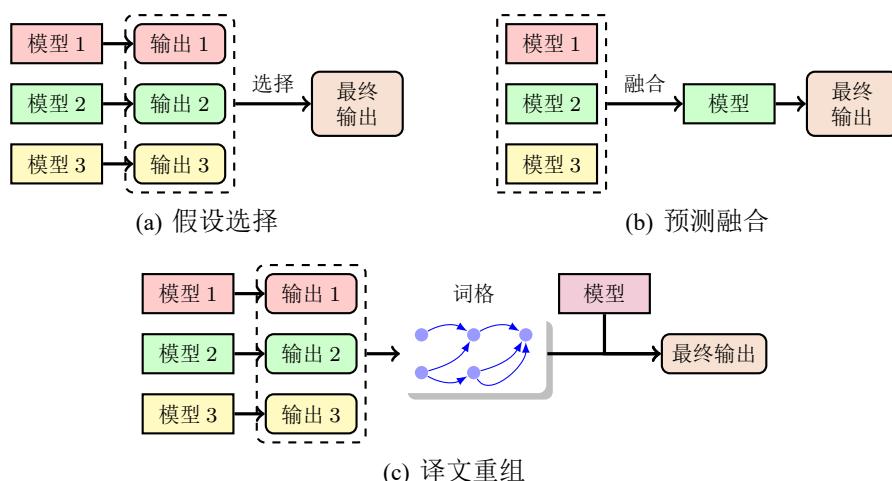


图 14.18 不同的模型集成方法对比

图14.18对比了不同模型集成方法的区别。从系统开发的角度看，假设选择和模

<sup>9</sup>本例中的词格也是一个混淆网络（Confusion Network）。

型预测融合的复杂度较低，适合快速开发原型系统，而且性能稳定。译文重组需要更多的模块，系统调试的复杂度较高，但是由于看到了更大的搜索空间，因此系统性能提升的潜力较大<sup>10</sup>。

## 14.6 小结与拓展阅读

推断系统（或解码系统）是神经机器翻译的重要组成部分。在神经机器翻译研究中，单独针对推断问题开展的讨论并不多见。更多的工作是将其与实践结合，常见于开源系统、评测比赛中。但是，从应用的角度看，研发高效的推断系统是机器翻译能够被大规模使用的前提。本章也从神经机器翻译推断的基本问题出发，重点探讨了推断系统的效率、非自回归翻译、多模型集成等问题。但是，由于推断问题涉及的问题十分广泛，因此本章也无法对其进行全面覆盖。关于神经机器翻译模型推断还有以下若干研究方向值得关注：

- 机器翻译系统中的推断也借用了**统计推断**（Statistical Inference）的概念。传统意义上讲，这类方法都是在利用样本数据去推测总体的趋势和特征。因此，从统计学的角度也有很多不同的思路。例如，贝叶斯学习等方法就在自然语言处理中得到广泛应用<sup>[758, 759]</sup>。其中比较有代表性的是**变分方法**（Variational Methods）。这类方法通过引入新的隐含变量来对样本的分布进行建模，从某种意义上说它是在描述“分布的分布”，因此这种方法对事物的统计规律描述得更加细致<sup>[760]</sup>。这类方法也被成功地用于统计机器翻译<sup>[403, 761]</sup> 和神经机器翻译<sup>[762, 763, 764, 765]</sup>。
- 推断系统也可以受益于更加高效的神经网络结构。这方面工作集中在**结构化剪枝**、**减少模型的冗余计算**、**低秩分解**等方向。结构化剪枝中的代表性工作是LayerDrop<sup>[766, 767, 768]</sup>，这类方法在训练时随机选择部分子结构，在推断时根据输入来选择模型中的部分层进行计算，而跳过其余层，达到加速的目的。有关减少冗余计算的研究主要集中在改进注意力机制上，本章已经有所介绍。低秩分解则针对词向量或者注意力的映射矩阵进行改进，例如词频自适应表示<sup>[769]</sup>，词频越高则对应的向量维度越大，反之则越小，或者层数越高注意力映射矩阵维度越小<sup>[727, 770, 771, 772]</sup>。在实践中比较有效的是较深的编码器与较浅的解码器结合的方式，极端情况下解码器仅使用1层神经网络即可取得与多层神经网络相媲美的翻译品质，从而极大地提升翻译效率<sup>[773, 774, 775]</sup>。在第十五章还会进一步对高效神经机器翻译的模型结构进行讨论。
- 在对机器翻译推断系统进行实际部署时，对存储的消耗也是需要考虑的因素。因此如何让模型变得更小也是研发人员所关注的方向。当前的模型压缩方法主要可以分为几类：剪枝、量化、知识蒸馏和轻量方法，其中轻量方法主要是基于更轻量模型结构的设计，这类方法已经在本章进行了介绍。剪枝主要包括权

<sup>10</sup>一般说来词格上的 Oracle 要比  $n$ -best 译文上的 Oracle 的质量高。

重大小剪枝<sup>[776, 777, 778, 779]</sup>、面向多头注意力的剪枝<sup>[539, 724]</sup>、网络层以及其他结构剪枝等<sup>[780, 781]</sup>，还有一些方法也通过在训练期间采用正则化的方式来提升剪枝能力<sup>[766]</sup>。量化方法主要通过截断浮点数来减少模型的存储大小，使其仅使用几个比特位的数字表示方法便能存储整个模型，虽然会导致舍入误差，但压缩效果显著<sup>[545, 782, 783, 784]</sup>。一些方法利用知识蒸馏手段还将 Transformer 模型蒸馏成如 LSTMs 等其他各种推断速度更快的结构<sup>[547, 725, 785]</sup>。

- 目前的翻译模型使用交叉熵损失作为优化函数，这在自回归模型上取得了非常优秀的性能。交叉熵是一个严格的损失函数，每个预测错误的单词所对应的位置都会受到惩罚，即使是编辑距离很小的输出序列<sup>[786]</sup>。自回归模型会很大程度上避免这种惩罚，因为当前位置的单词是根据先前生成的词得到的，然而非自回归模型无法获得这种信息。如果在预测时漏掉一个单词，就可能会将正确的单词放在错误的位置上。为此，一些研究工作通过改进损失函数来提高非自回归模型的性能。一种做法使用一种新的交叉熵函数<sup>[786]</sup>，它通过忽略绝对位置、关注相对顺序和词汇匹配来为非自回归模型提供更精确的训练信号。另外，也可以使用基于  $n$ -gram 的训练目标<sup>[787]</sup> 来最小化模型与参考译文之间的  $n$ -gram 差异。该训练目标在  $n$ -gram 的层面上评估预测结果，因此能够建模目标序列单词之间的依赖关系。
- 自回归模型解码时，当前位置单词的生成依赖于先前生成的单词，已生成的单词提供了较强的目标端上下文信息。与自回归模型相比，非自回归模型的解码器需要在信息更少的情况下执行翻译任务。一些研究工作通过将条件随机场引入非自回归模型中来对序列依赖进行建模<sup>[738]</sup>。也有工作引入了词嵌入转换矩阵来将源语言端的词嵌入转换为目标语言端的词嵌入来为解码器提供更好的输入<sup>[736]</sup>。此外，研究人员也提出了轻量级的调序模块来显式地建模调序信息，以指导非自回归模型的推断<sup>[746]</sup>。



## 15. 神经机器翻译结构优化

模型结构的设计是机器翻译系统研发中最重要的工作之一。在神经机器翻译时代，虽然系统研发人员脱离了繁琐的特征工程，但是神经网络结构的设计仍然耗时耗力。无论是像循环神经网络、Transformer这样的整体架构的设计，还是注意力机制等局部结构的设计，都对机器翻译性能有着很大的影响。

本章主要讨论神经机器翻译中结构优化的若干研究方向，包括：注意力机制的改进、网络连接优化及深层网络建模、基于树结构的模型、神经网络结构自动搜索等。这些内容可以指导神经机器翻译系统的深入优化，其中涉及的一些模型和方法也可以应用于其他自然语言处理任务。

### 15.1 注意力机制的改进

注意力机制是神经机器翻译成功的关键。以 Transformer 模型为例，由于使用了自注意力机制，该模型展现出较高的训练并行性，同时在机器翻译、语言建模等任务上，该模型也取得了很好的表现。但是 Transformer 模型仍存在许多亟待解决的问题，例如，在处理长文本序列时（假设文本长度为  $N$ ），自注意力机制的时间复杂度为  $O(N^2)$ ，当  $N$  过大时翻译速度很低。此外，尽管 Transformer 模型的输入中包含了绝对位置编码表示，但是现有的自注意力机制仍然无法显性捕获局部窗口下不同位置之间的关系。而且注意力机制也需要更多样的手段进行特征提取，例如，采用多头或者多分支结构对不同空间特征进行提取。针对以上问题，本节将介绍注意力机制的优化策略，并重点讨论 Transformer 模型的若干改进方法。

### 15.1.1 局部信息建模

使用循环神经网络进行序列建模时，每一个时刻的计算都依赖于上一时刻循环单元的状态。这种模式天然具有一定的时序性，同时具有**归纳偏置**（Inductive Bias）的特性<sup>[788]</sup>，即每一时刻的状态仅仅基于当前时刻的输入和前一时刻的状态。这种归纳偏置的好处在于，模型并不需要对绝对位置进行建模，因此模型可以很容易处理任意长度的序列，即使测试样本显著长于训练样本。

但是，Transformer 模型中的自注意力机制本身并不具有这种性质，而且它直接忽略了输入单元之间的位置关系。虽然，Transformer 中引入了基于正余弦函数的绝对位置编码（见第十二章），但是该方法仍然无法显性区分局部依赖与长距离依赖<sup>1</sup>。

针对上述问题，研究人员尝试引入“相对位置”信息，对原有的“绝对位置”信息进行补充，强化了局部依赖<sup>[460, 550]</sup>。此外，由于模型中每一层均存在自注意力机制计算，因此模型捕获位置信息的能力也逐渐减弱，这种现象在深层模型中尤为明显。而利用相对位置编码能够把位置信息显性加入到每一层的注意力机制的计算中，进而强化深层模型的位置表示能力<sup>[462]</sup>。图15.1对比了Transformer 中绝对位置编码和相对位置编码方法。

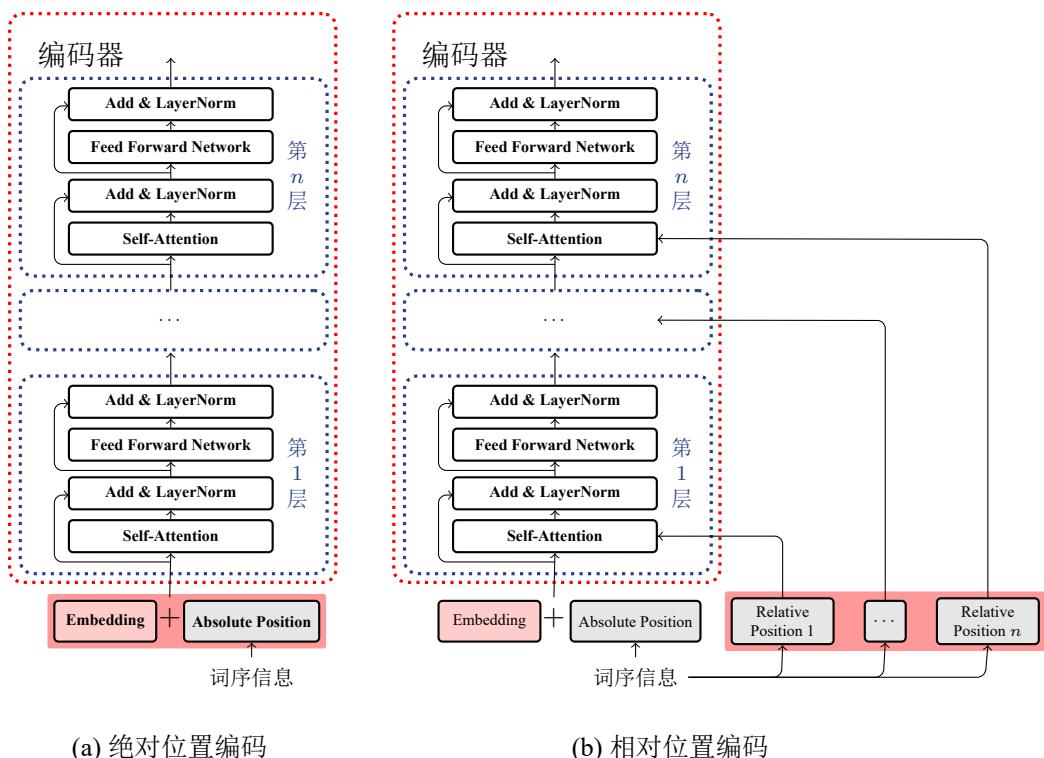


图 15.1 绝对位置编码和相对位置编码

<sup>1</sup>局部依赖指当前位置与局部的相邻位置之间的联系。

## 1. 位置编码

在介绍相对位置编码之前，首先简要回顾一下自注意力机制的计算流程（见第十二章）。对于 Transformer 模型中的某一层神经网络，可以定义：

$$\mathbf{Q} = \mathbf{x}\mathbf{W}_Q \quad (15.1)$$

$$\mathbf{K} = \mathbf{x}\mathbf{W}_K \quad (15.2)$$

$$\mathbf{V} = \mathbf{x}\mathbf{W}_V \quad (15.3)$$

其中， $\mathbf{x}$  为上一层的输出<sup>2</sup>， $\mathbf{W}_Q$ 、 $\mathbf{W}_K$ 、 $\mathbf{W}_V$  为模型参数，它们可以通过自动学习得到。此时，对于整个模型输入的向量序列  $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ ，通过点乘计算，可以得到当前位置  $i$  和序列中所有位置间的关系，记为  $\mathbf{z}_i$ ，计算公式如下：

$$\mathbf{z}_i = \sum_{j=1}^m \alpha_{ij} (\mathbf{x}_j \mathbf{W}_V) \quad (15.4)$$

这里， $\mathbf{z}_i$  可以被看做是输入序列的线性加权表示结果。权重  $\alpha_{ij}$  通过 Softmax 函数得到：

$$\alpha_{ij} = \frac{\exp(\mathbf{e}_{ij})}{\sum_{k=1}^m \exp(\mathbf{e}_{ik})} \quad (15.5)$$

进一步， $\mathbf{e}_{ij}$  被定义为：

$$\mathbf{e}_{ij} = \frac{(\mathbf{x}_i \mathbf{W}_Q)(\mathbf{x}_j \mathbf{W}_K)^T}{\sqrt{d_k}} \quad (15.6)$$

其中， $d_k$  为模型中隐层的维度<sup>3</sup>。 $\mathbf{e}_{ij}$  实际上就是  $\mathbf{Q}$  和  $\mathbf{K}$  的向量积缩放后的一个结果。

基于上述描述，相对位置模型可以按如下方式实现：

- **相对位置编码** (Relative Positional Representation)<sup>[460]</sup>。核心思想是在能够捕获全局依赖的自注意力机制中引入相对位置信息。该方法可以有效补充绝对位置编码的不足，甚至完全取代绝对位置编码。对于 Transformer 模型中的任意一层，假设  $\mathbf{x}_i$  和  $\mathbf{x}_j$  是位置  $i$  和  $j$  的输入向量（也就是来自上一层位置  $i$  和  $j$  的输

---

<sup>2</sup>这里， $\mathbf{K}$ 、 $\mathbf{Q}$ 、 $\mathbf{V}$  的定义与第十二章略有不同，因为在这里的  $\mathbf{K}$ 、 $\mathbf{Q}$ 、 $\mathbf{V}$  是指对注意力模型输入进行线性变换后的结果，而第十二章中  $\mathbf{K}$ 、 $\mathbf{Q}$ 、 $\mathbf{V}$  直接表示输入。但是，这两种描述方式本质上是一样的，区别仅仅在于对输入的线性变化是放在输入自身中描述，还是作为输入之后的一个额外操作。

<sup>3</sup>在多头注意力中， $d_k$  为经过多头分割后每个头的维度。

出向量), 二者的位置关系可以通过向量  $\mathbf{a}_{ij}^V$  和  $\mathbf{a}_{ij}^K$  来表示, 定义如下:

$$\mathbf{a}_{ij}^K = \mathbf{w}_{\text{clip}(j-i,k)}^K \quad (15.7)$$

$$\mathbf{a}_{ij}^V = \mathbf{w}_{\text{clip}(j-i,k)}^V \quad (15.8)$$

$$\text{clip}(x, k) = \max(-k, \min(k, x)) \quad (15.9)$$

其中,  $\mathbf{w}^K \in \mathbb{R}^{d_k}$  和  $\mathbf{w}^V \in \mathbb{R}^{d_k}$  是模型中可学习的参数矩阵;  $\text{clip}(\cdot, \cdot)$  表示截断操作, 由公式(15.9)定义。可以看出,  $\mathbf{a}^K$  与  $\mathbf{a}^V$  是根据输入的相对位置信息 (由  $\text{clip}(j - i, k)$  确定) 对  $\mathbf{w}^K$  和  $\mathbf{w}^V$  进行查表得到的向量, 即相对位置表示, 如图15.2所示。这里通过预先设定的最大相对位置  $k$ , 强化模型对以当前词为中心的左右各  $k$  个词的注意力计算。因此, 最终的窗口大小为  $2k + 1$ 。对于边缘位置窗口大小不足  $2k$  的单词, 采用了裁剪的机制, 即只对有效的临近词进行建模。此时, 注意力模型的计算可以调整为:

$$\mathbf{z}_i = \sum_{j=1}^m \alpha_{ij} (\mathbf{x}_j \mathbf{W}_V + \mathbf{a}_{ij}^V) \quad (15.10)$$

6	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-2}$	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	
5	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-2}$	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	$\mathbf{w}_1$	
4	$\mathbf{w}_{-3}$	$\mathbf{w}_{-3}$	$\mathbf{w}_{-2}$	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	
3	$\mathbf{w}_{-3}$	$\mathbf{w}_{-2}$	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	
2	$\mathbf{w}_{-2}$	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_3$	
1	$\mathbf{w}_{-1}$	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_3$	$\mathbf{w}_3$	
0	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_3$	$\mathbf{w}_3$	$\mathbf{w}_3$	
<i>i</i>	j	0	1	2	3	4	5	6

图 15.2 相对位置权重  $\mathbf{a}_{ij}$ <sup>[789]</sup>

相比于公式(15.4), 公式(15.10)在计算  $\mathbf{z}_i$  时引入了额外的向量  $\mathbf{a}_{ij}^V$ , 用它来表示位置  $i$  与位置  $j$  之间的相对位置信息。同时在计算注意力权重时对  $\mathbf{K}$  进行修改, 同样引入了  $\mathbf{a}_{ij}^K$  向量表示位置  $i$  与位置  $j$  之间的相对位置。在公式(15.6)的基础上, 注意力权重的计算方式调整为:

$$\begin{aligned} \mathbf{e}_{ij} &= \frac{\mathbf{x}_i \mathbf{W}_Q (\mathbf{x}_j \mathbf{W}_K + \mathbf{a}_{ij}^K)^T}{\sqrt{d_k}} \\ &= \frac{\mathbf{x}_i \mathbf{W}_Q (\mathbf{x}_j \mathbf{W}_K)^T + \mathbf{x}_i \mathbf{W}_Q (\mathbf{a}_{ij}^K)^T}{\sqrt{d_k}} \end{aligned} \quad (15.11)$$

可以注意到，公式(15.10)和公式(15.11)将位置编码信息直接暴露给每一层注意力机制的计算，而不是像标准 Transformer 中只将其作为整个模型的输入。

- **Transformer-XL**<sup>[550]</sup>。在 Transformer 中，模型的输入由词嵌入表示与绝对位置编码组成，例如，对于输入层有， $\mathbf{x}_i = \mathbf{E}_{x_i} + \mathbf{U}_i$ ,  $\mathbf{x}_j = \mathbf{E}_{x_j} + \mathbf{U}_j$ ，其中  $\mathbf{E}_{x_i}$  和  $\mathbf{E}_{x_j}$  表示词嵌入， $\mathbf{U}_i$  和  $\mathbf{U}_j$  表示绝对位置编码（正余弦函数）。将  $\mathbf{x}_i$  与  $\mathbf{x}_j$  代入公式(15.6)中可以得到：

$$\mathbf{e}_{ij} = \frac{(\mathbf{E}_{x_i} + \mathbf{U}_i)\mathbf{W}_Q((\mathbf{E}_{x_j} + \mathbf{U}_j)\mathbf{W}_K)^T}{\sqrt{d_k}} \quad (15.12)$$

这里使用  $A_{ij}^{\text{abs}}$  表示公式(15.12)中等式右侧的分子部分，并对其进行展开：

$$\begin{aligned} A_{ij}^{\text{abs}} &= \underbrace{\mathbf{E}_{x_i}\mathbf{W}_Q\mathbf{W}_K^T\mathbf{E}_{x_j}^T}_{(a)} + \underbrace{\mathbf{E}_{x_i}\mathbf{W}_Q\mathbf{W}_K^T\mathbf{U}_j^T}_{(b)} + \\ &\quad \underbrace{\mathbf{U}_i\mathbf{W}_Q\mathbf{W}_K^T\mathbf{E}_{x_j}^T}_{(c)} + \underbrace{\mathbf{U}_i\mathbf{W}_Q\mathbf{W}_K^T\mathbf{U}_j^T}_{(d)} \end{aligned} \quad (15.13)$$

其中，abs 代表使用绝对位置编码计算得到的  $A_{ij}$ ， $\mathbf{W}_Q$  与  $\mathbf{W}_K$  表示线性变换矩阵。为了引入相对位置信息，可以将公式(15.13)修改为如下形式：

$$\begin{aligned} A_{ij}^{\text{rel}} &= \underbrace{\mathbf{E}_{x_i}\mathbf{W}_Q\mathbf{W}_K^T\mathbf{E}_{x_j}^T}_{(a)} + \underbrace{\mathbf{E}_{x_i}\mathbf{W}_Q\mathbf{W}_K^T\mathbf{R}_{i-j}^T}_{(b)} + \\ &\quad \underbrace{\mathbf{u}\mathbf{W}_{K,E}^T\mathbf{E}_{x_j}^T}_{(c)} + \underbrace{\mathbf{v}\mathbf{W}_{K,R}^T\mathbf{R}_{i-j}^T}_{(d)} \end{aligned} \quad (15.14)$$

其中， $A_{ij}^{\text{rel}}$  为使用相对位置编码后位置  $i$  与  $j$  关系的表示结果， $\mathbf{R}$  是一个固定的正弦矩阵。不同于公式(15.13)，公式(15.14)对 (c) 中的  $\mathbf{E}_{x_j}^T$  与 (d) 中的  $\mathbf{R}_{i-j}^T$  采用了不同的映射矩阵，分别为  $\mathbf{W}_{K,E}^T$  和  $\mathbf{W}_{K,R}^T$ ，这两项分别代表了键  $\mathbf{K}$  中的词嵌入表示和相对位置编码表示，并且由于此时只采用了相对位置编码，因此公式(15.14)在 (c) 与 (d) 部分使用了  $\mathbf{u}$  和  $\mathbf{v}$  两个可学习的矩阵代替  $\mathbf{U}_i\mathbf{W}_Q$  与  $\mathbf{U}_i\mathbf{W}_Q$ ，即查询  $\mathbf{Q}$  中的绝对位置编码部分。此时公式中各项的含义为：(a) 表示位置  $i$  与位置  $j$  之间词嵌入的相关性，可以看作是基于内容的表示，(b) 表示基于内容的位置偏置，(c) 表示全局内容的偏置，(d) 表示全局位置的偏置。公式(15.13)中的 (a)、(b) 两项与前面介绍的绝对位置编码一致<sup>[460]</sup>，并针对相对位置编码引入了额外的线性变换矩阵。同时，这种方法兼顾了全局内容偏置和全局位置偏置，可以更好地利用正余弦函数的归纳偏置特性。

- **结构化位置编码** (Structural Position Representations)<sup>[790]</sup>。例如，通过对输入句子进行依存句法分析得到句法树，根据叶子结点在句法树中的深度来表示

其绝对位置，并在此基础上利用相对位置编码的思想计算节点之间的相对位置信息。

- 基于连续动态系统（Continuous Dynamic Model）的位置编码<sup>[551]</sup>。使用神经常微分方程求解器（Solver）来建模位置信息<sup>[791]</sup>，模型具有更好的归纳偏置能力，可以处理变长的输入序列，同时能够从不同的数据中进行自适应学习。

## 2. 注意力分布约束

局部注意力机制一直是机器翻译中受关注的研究方向<sup>[25]</sup>。通过对注意力权重的可视化，可以观测到不同位置的词受关注的程度相对平滑。这样的建模方式利于全局建模，但一定程度上分散了注意力，导致模型忽略了邻近单词之间的关系。为了提高模型对局部信息的感知，有以下几种方法：

- 引入高斯约束<sup>[553]</sup>。如图15.3所示，这类方法的核心思想是引入可学习的高斯分布  $\mathbf{G}$  作为局部约束，与注意力权重进行融合。

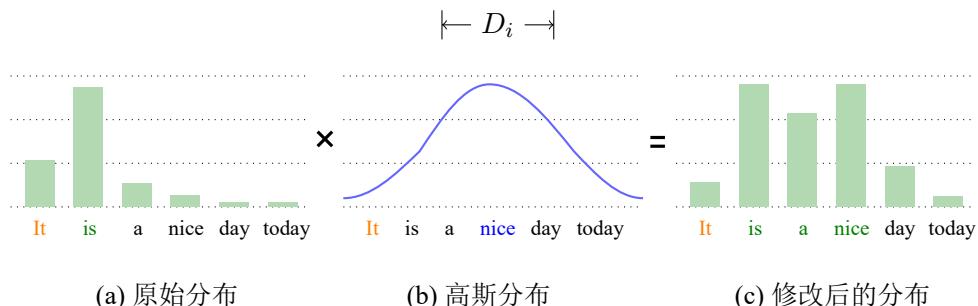


图 15.3 融合高斯分布的注意力分布

具体的形式如下：

$$\mathbf{e}_{ij} = \frac{(\mathbf{x}_i \mathbf{W}_Q)(\mathbf{x}_j \mathbf{W}_K)^T}{\sqrt{d_k}} + G_{ij} \quad (15.15)$$

其中， $G_{ij}$  表示位置  $j$  和预测的中心位置  $P_i$  之间的关联程度， $G_{ij}$  是  $\mathbf{G}$  中的一个元素， $\mathbf{G} \in \mathbb{R}^{m \times m}$ 。计算公式如下：

$$G_{ij} = -\frac{(j - P_i)^2}{2\sigma_i^2} \quad (15.16)$$

其中， $\sigma_i$  表示偏差，被定义为第  $i$  个词的局部建模窗口大小  $D_i$  的一半，即

$\sigma_i = \frac{D_i}{2}$ 。中心位置  $P_i$  和局部建模窗口  $D_i$  的计算方式如下：

$$\begin{pmatrix} P_i \\ D_i \end{pmatrix} = m \cdot \text{Sigmoid}\left(\begin{pmatrix} p_i \\ v_i \end{pmatrix}\right) \quad (15.17)$$

其中， $m$  表示序列长度， $p_i$  和  $v_i$  为计算的中间结果，被定义为：

$$p_i = \mathbf{I}_p^T \text{Tanh}(\mathbf{W}_p \mathbf{Q}_i) \quad (15.18)$$

$$v_i = \mathbf{I}_d^T \text{Tanh}(\mathbf{W}_d \mathbf{Q}_i) \quad (15.19)$$

其中， $\mathbf{W}_p$ 、 $\mathbf{W}_d$ 、 $\mathbf{I}_p$ 、 $\mathbf{I}_d$  均为模型中可学习的参数矩阵。

- **多尺度局部建模**<sup>[792]</sup>。不同于上述方法直接作用于注意力权重，多尺度局部建模通过赋予多头不一样的局部感受野，间接地引入局部约束（图15.4）。

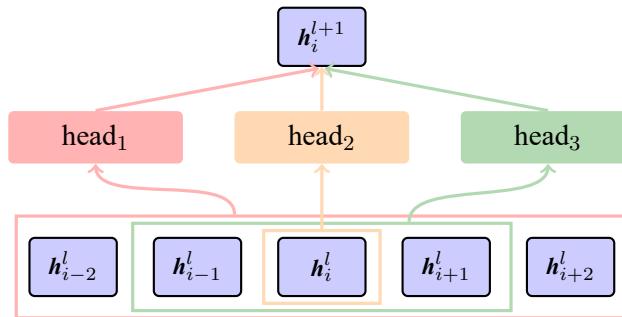


图 15.4 多尺度局部建模<sup>[792]</sup>

于是，在计算第  $i$  个词对第  $j$  个词的相关系数时，通过超参数  $\omega$  控制实际的感受野为  $j - \omega, \dots, j + \omega$ ，注意力计算中  $\mathbf{e}_{ij}$  的计算方式与公式(15.6)相同，权重  $\alpha_{ij}$  的具体计算公式为：

$$\alpha_{ij} = \frac{\exp(\mathbf{e}_{ij})}{\sum_{k=j-\omega}^{j+\omega} \exp(\mathbf{e}_{ik})} \quad (15.20)$$

之后在计算注意力输出时同样利用上述思想进行局部约束：

$$\mathbf{z}_i = \sum_{j=j-\omega}^{j+\omega} \alpha_{ij} (\mathbf{x}_j \mathbf{W}_V) \quad (15.21)$$

其中，约束的具体作用范围会根据实际句长进行一定的裁剪，通过对不同的头设置不同的超参数来控制感受野的大小，最终实现多尺度局部建模。

值得注意的是上述两种添加局部约束的方法都更适用于 Transformer 模型的底层

网络。这是由于模型离输入更近的层更倾向于捕获局部信息<sup>[552, 553]</sup>，伴随着神经网络的加深，模型更倾向于逐渐加强全局建模的能力。类似的结论在针对 BERT 模型的解释性研究工作中也有论述<sup>[552, 793]</sup>。

### 3. 卷积 vs 注意力

第十一章已经提到，卷积神经网络能够很好地捕捉序列中的局部信息。因此，充分地利用卷积神经网络的特性，也是一种进一步优化注意力模型的思路。常见的做法是在注意力模型中引入卷积操作，甚至用卷积操作替换注意力模型，如：

- **使用轻量卷积和动态卷积神经网络**<sup>[507, 533]</sup>。比如，分别在编码器和解码器利用轻量卷积或动态卷积神经网络（见第九章）替换 Transformer 的自注意力机制，同时保留解码器的编码-解码注意力机制，一定程度上加强了模型对局部信息的建模能力，同时提高了计算效率。
- **使用 1 维卷积注意力网络**<sup>[554]</sup>（图15.5(b)）。可以使用一维的卷积自注意力网络（1D-CSAN）将关注的范围限制在相近的元素窗口中。其形式上十分简单，只需预先设定好局部建模的窗口大小  $D$ ，并在进行注意力权重计算和对 Value 值进行加权求和时，将其限制在设定好的窗口范围内。
- **使用 2 维卷积注意力网络**（图15.5(c)）。在一维卷积注意力网络的基础上，对多个注意力头之间的信息进行交互建模，打破了注意力头之间的界限。1D-CSAN 的关注区域为  $1 \times D$ ，当将其扩展为二维矩形  $D \times N$ ，长和宽分别为局部窗口的大小和参与建模的自注意力头的个数。这样，模型可以计算某个头中的第  $i$  个元素和另一个头中的第  $j$  个元素之间的相关性系数，实现了对不同子空间之间关系的建模，所得到的注意力分布表示了头之间的依赖关系。

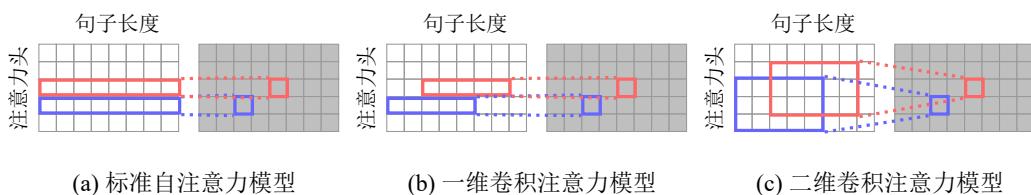


图 15.5 卷积注意力模型示意图<sup>[554]</sup>

#### 15.1.2 多分支结构

在神经网络模型中，可以使用多个平行的组件从不同角度捕捉输入的特征，这种结构被称为**多分支**（Multi-branch）结构。多分支结构在图像处理领域被广泛应用<sup>[794]</sup>，在许多人工设计或者自动搜索获得的神经网络结构中也有它的身影<sup>[795, 796, 797]</sup>。

在自然语言处理领域，多分支结构同样也有很多应用。比如，第十章介绍过，为了更好地对源语言进行表示，编码器可以采用双向循环神经网络。这种模型就可以

被看作一个两分支的结构，分别用来建模正向序列和反向序列的表示，之后将这两种表示进行拼接得到更丰富的序列表示结果。另一个典型的例子是第十二章介绍的多头注意力机制。在 Transformer 模型中，多头注意力将输入向量分割成多个子向量，然后分别进行点乘注意力的计算，最后再将多个输出的子向量拼接后通过线性变换进行不同子空间信息的融合。在这个过程中，多个不同的头对应着不同的特征空间，可以捕捉到不同的特征信息。

近几年，在 Transformer 的结构基础上，研究人员探索了更为丰富的多分支结构。下面介绍几种在 Transformer 模型中引入多分支结构的方法：

- **基于权重的方法**<sup>[798]</sup>。其主要思想是在多头自注意力机制的基础上保留不同表示空间的特征。传统方法使用级联操作并通过线性映射矩阵来融合不同头之间的信息，而基于权重的 Transformer 直接利用线性映射将维度为  $d_k$  的向量表示映射到  $d_{\text{model}}$  维的向量。之后，将这个  $d_{\text{model}}$  维向量分别送入每个分支中的前馈神经网络，最后对不同分支的输出进行线性加权。但是，这种模型的计算复杂度要大于标准的 Transformer 模型。
- **基于多分支注意力的方法**<sup>[797]</sup>。不同于基于权重的 Transformer 模型，多分支注意力模型直接利用每个分支独立地进行自注意力模型的计算（图15.6）。同时为了避免结构相同的多个多头注意力机制之间的协同适应，这种模型使用 Dropout 方法在训练过程中以一定的概率随机地丢弃一些分支。

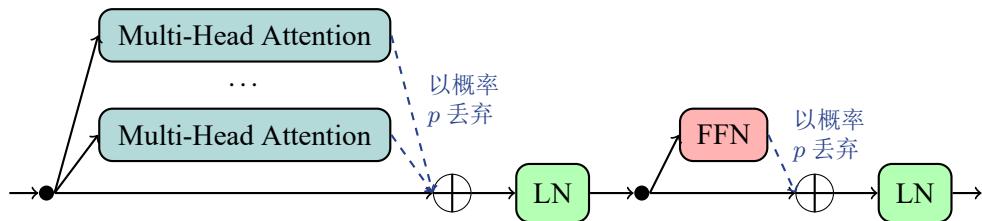


图 15.6 多分支注意力模型

- **基于多单元的方法**。例如，为了进一步加强不同分支的作用，基于多单元的 Transformer 模型进行了序列不同位置表示结果的交换，或使用不同的掩码策略对不同分支的输入进行扰动，保证分支间的多样性与互补性<sup>[796]</sup>。本质上，所谓的多单元思想与集成学习十分相似，类似于在训练过程中同时训练多个编码器。此外，通过增大子单元之间的结构差异性也能够进一步增大分支之间的多样性<sup>[799]</sup>。

此外，在15.1.1节中曾提到过，利用卷积神经网络可以与自注意力机制形成互补。类似的想法在多分支结构中也有体现。如图15.7所示，可以使用自注意力机制和卷积神经网络分别提取全局和局部两种依赖关系<sup>[542]</sup>。具体的做法是将输入的特征向量切分成等同维度的两部分，之后分别送入两个分支进行计算。其中，全局信息使用自

注意力机制进行提取，局部信息使用轻量卷积网络进行提取<sup>[507]</sup>。此外，由于每个分支的维度只有原始的一半，采用并行计算方式可以显著提升系统的运行速度。

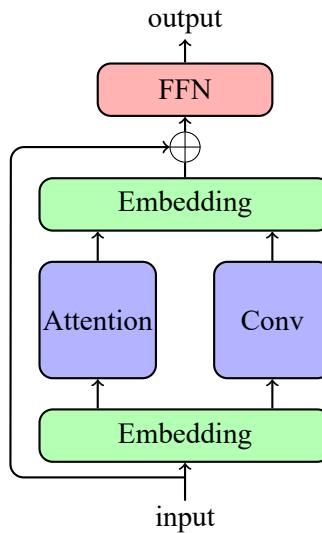


图 15.7 基于自注意力和卷积神经网络的 2 分支结构

### 15.1.3 引入循环机制

虽然 Transformer 模型完全摒弃了循环单元与卷积单元，仅通过位置编码来区分序列中的不同位置。但是，循环神经网络也非常适用于处理序列结构，且其结构成熟、易于优化。因此，有研究人员尝试将其与 Transformer 模型融合。这种方式一方面能够发挥循环神经网络简单高效的特点，另一方面也能够发挥 Transformer 模型在特征提取方面的优势，是一种非常值得探索的思路<sup>[458]</sup>。

在 Transformer 模型中引入循环神经网络的一种方法是，对深层网络的不同层使用循环机制。早在残差网络提出时，研究人员已经开始尝试探讨残差网络成功背后的原因<sup>[800, 801, 802]</sup>。本质上，在卷积神经网络中引入残差连接后，神经网络从深度上隐性地利用了循环的特性。也就是，多层 Transformer 的不同层本身也可以被看作是一个处理序列，只是序列中不同位置（对应不同层）的模型参数独立，而非共享。Transformer 编码器与解码器分别由  $N$  个结构相同但参数独立的层堆叠而成，其中编码器包含 2 个子层，解码器包含 3 个子层。同时，子层之间引入了残差连接保证了网络信息传递的高效性。因此，一个自然的想法是通过共享不同层之间的参数，引入循环神经网络中的归纳偏置<sup>[803]</sup>。其中每层的权重是共享的，并引入了基于时序的编码向量用于显著区分不同深度下的时序信息，如图 15.8 所示。在训练大容量预训练模型时同样也采取了共享层间参数的方式<sup>[804]</sup>。

另一种方法是，利用循环神经网络对输入序列进行编码，之后通过门控机制将

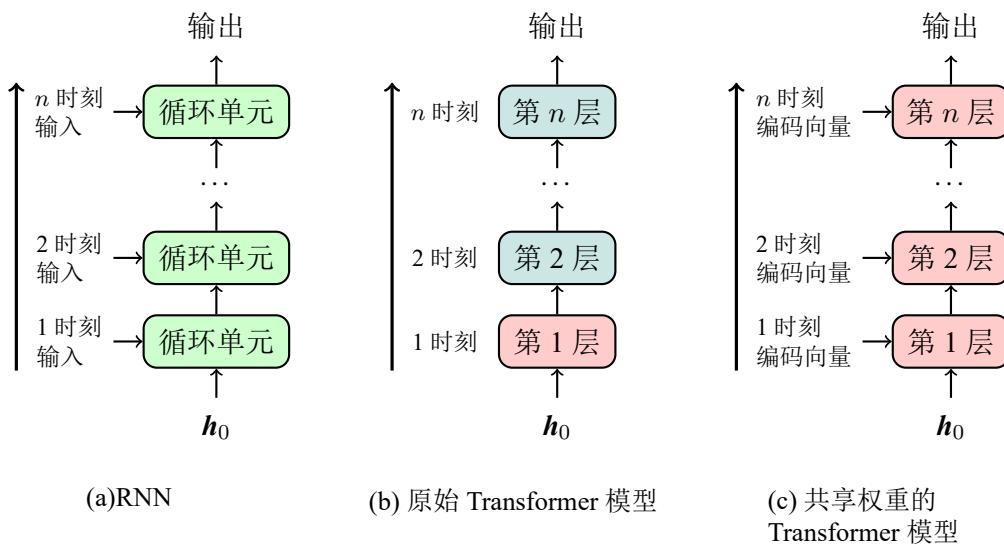


图 15.8 在 Transformer 中引入循环机制

得到的结果与 Transformer 进行融合<sup>[805]</sup>。融合机制可以采用串行计算或并行计算。

### 15.1.4 高效的自注意力模型

除了机器翻译, Transformer 模型同样被广泛应用于自然语言理解、图像处理、语音处理等任务。但是, 自注意力机制的时间复杂度是序列长度  $N$  的平方项, 同时其对内存(显存)的消耗巨大, 尤其当处理较长序列的文本时, 这种问题尤为严重。因此如何提高 Transformer 模型的效率也受到了广泛的关注。第十四章已经从模型推断的角度介绍了 Transformer 系统加速的方法, 这里重点讨论一些高效的 Transformer 变种模型。

自注意力机制的时间复杂度较高，正是因其需要对序列中的每一个位置计算与其他所有位置的相关性。因此一个自然的想法就是限制自注意力机制的作用范围，大体上可以分为如下几种方式：

- **分块注意力**: 顾名思义，就是将序列划分为固定大小的片段，注意力模型只在对应的片段内执行。这样，每一个片段内的注意力计算成本是固定的，可以大大降低处理长序列时的总体计算时间<sup>[806, 807]</sup>。
  - **跨步注意力**: 该模型是一种稀疏的注意力机制，通常会设置一个固定的间隔，也就是说在计算注意力表示时，每隔固定数量的词后将下一个词纳入所需考虑的范围内，参与注意力的计算<sup>[808]</sup>。和分片段进行注意力计算类似，假设最终参与注意力计算的间隔长度为  $N/B$ ，每次参与注意力计算的单词数为  $B$ ，那么注意力的计算复杂度将从  $O(N^2)$  缩减为  $O(N/B \times B^2)$ ，即  $O(NB)$ 。

- **内存压缩注意力**: 这种方式的主要的思想是使用一些操作, 如卷积、池化等对序列进行下采样, 来缩短序列长度。例如, 使用**跨步卷积** (Stride Convolution) 来减少 Key 和 Value 的数量, 即减少表示序列长度的维度的大小, Query 的数量保持不变, 从而减少了注意力权重计算时的复杂度<sup>[807]</sup>。其具体的计算复杂度取决于跨步卷积时步幅的大小  $K$ , 形式上可以理解为每  $K$  个单元做一次特征融合后, 将关注的目标缩减为  $N/K$ , 整体的计算复杂度为  $N^2/K$ 。相比于使用前两种方式对局部进行注意力计算, 该方式仍是对全局的建模。

在不同的任务中, 可以根据不同的需求使用不同的注意力模型, 甚至可以采用多种注意力模型的结合。比如, 对分类任务中的某些特殊标签, 如 BERT 中的 <CLS>, 需要对全局信息进行整合, 因此可以使用全局注意力。而对于其他位置, 则可以使用局部注意力提高计算效率。同样的, 也可以针对多头机制中的不同注意力头采用不同的计算方式, 或者对不同的头设置不同的局部窗口大小, 以此来增大感受野, 在提高模型计算效率的同时使模型保留全局建模能力。

由于上述方法都是基于预先设定好的超参来限制注意力机制的作用范围, 因此可以称这些方法是静态的。除此之外还有以数据驱动的方法, 这类方法通过模型来学习注意力机制的作用范围。比如, 可以将序列分块, 并对序列中的不同单元进行排序或者聚类, 之后采用稀疏注意力的计算。下面对部分相关的模型进行介绍:

- Reformer 模型在计算 Key 和 Value 时使用相同的线性映射, 共享 Key 和 Value 的值<sup>[543]</sup>, 降低了自注意力机制的复杂度。进一步, Reformer 引入了一种**局部哈希敏感注意力机制** (LSH Attention), 其提高效率的方式和固定模式中的局部建模一致, 减少注意力机制的计算范围。对于每一个 Query, 通过局部哈希敏感机制找出和其较为相关的 Key, 并进行注意力的计算。其基本思路就是距离相近的向量以较大的概率被哈希分配到一个桶内, 距离较远的向量被分配到一个桶内的概率则较低。此外, Reformer 中还采用了一种**可逆残差网络结构** (The Reversible Residual Network) 和分块计算前馈神经网络层的机制, 即将前馈层的隐层维度拆分为多个块并独立的进行计算, 最后进行拼接操作, 得到前馈层的输出, 这种方式大幅度减少了内存 (显存) 占用。
- Routing Transformer 通过聚类算法对序列中的不同单元进行分组, 分别在组内进行自注意力机制的计算<sup>[809]</sup>。首先是将 Query 和 Key 映射到聚类矩阵  $\mathbf{S}$ :

$$\mathbf{S} = \mathbf{QW} + \mathbf{KW} \quad (15.22)$$

其中,  $\mathbf{W}$  为映射矩阵。为了保证每个簇内的单词数量一致, 利用聚类算法将  $\mathbf{S}$  中的向量分配到  $\sqrt{N}$  个簇中, 其中  $N$  为序列长度, 即分别计算  $\mathbf{S}$  中每个向量与质心的距离, 并对每个质心取距离最近的若干个节点。

另外, 在注意力机制中对计算效率影响很大的一个因素是 Softmax 函数的计算。

第十二章已经介绍过自注意力机制的计算公式为：

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (15.23)$$

由于 Softmax 函数的存在，因此首先要进行  $\mathbf{Q}\mathbf{K}^T$  的计算得到  $N \times N$  的矩阵，其时间复杂度便是  $O(N^2)$ 。假设能够移除 Softmax 操作，便可以将注意力机制的计算调整为  $\mathbf{Q}\mathbf{K}^T\mathbf{V}$ ，由于矩阵的运算满足结合律，可以先进行  $\mathbf{K}^T\mathbf{V}$  的运算，得到  $d_k \times d_k$  的矩阵，再左乘  $\mathbf{Q}$ 。在长文本处理中，由于多头机制的存在，一般有  $d_k \ll N$ ，所以最终的计算复杂度便可以近似为  $O(N)$ ，从而将注意力机制简化为线性模型<sup>[726, 810]</sup>。

## 15.2 神经网络连接优化及深层模型

除了对 Transformer 模型中的局部组件进行改进，改进不同层之间的连接方式也十分重要。常见的做法是融合编码器/解码器的中间层表示得到信息更丰富的编码/解码输出<sup>[555, 557, 558, 559]</sup>。同时，可以利用稠密连接等更丰富的层间连接方式来强化或替换残差连接。

与此同时，虽然宽网络（如 Transformer-Big）在机器翻译、语言模型等任务上表现十分出色，但伴随而来的是快速增长的参数量与更大的训练代价。并且受限于任务的复杂度与计算设备的算力，进一步探索更宽的神经网络显然不是特别高效的手段。因此研究人员普遍选择增加神经网络的深度来对句子进行更充分地表示。但是，简单地堆叠很多层的 Transformer 模型并不能带来性能上的提升，反而会面临更加严重的梯度消失/梯度爆炸的问题。这是由于伴随神经网络变深，梯度无法有效地从输出层回传到底层神经网络，造成浅层部分的参数无法得到充分训练<sup>[461, 556, 811, 812]</sup>。针对这些问题，可以设计更有利于深层信息传递的神经网络连接和恰当的参数初始化方法等。

但是，如何设计一个足够“深”的机器翻译模型仍然是业界关注的热点问题之一。此外，伴随着神经网络的继续变深，将会面临一些新的问题，例如，如何加速深层神经网络的训练，如何解决深层神经网络的过拟合问题等。下面将会对以上问题展开讨论。首先对 Transformer 模型的内部信息流进行分析，之后分别从模型结构和参数初始化两个角度求解为什么深层网络难以训练，并介绍相应的解决手段。

### 15.2.1 Post-Norm vs Pre-Norm

为了探究为何深层 Transformer 模型很难直接训练，首先对 Transformer 的模型结构进行简单的回顾，详细内容可以参考第十二章。以 Transformer 的编码器为例，在多头自注意力和前馈神经网络中间，Transformer 模型利用残差连接<sup>[421]</sup> 和层标准化操作<sup>[420]</sup> 来提高信息的传递效率。Transformer 模型大致分为图15.9中的两种结构——后作方式的残差单元（Post-Norm）和前作方式的残差单元（Pre-Norm）。

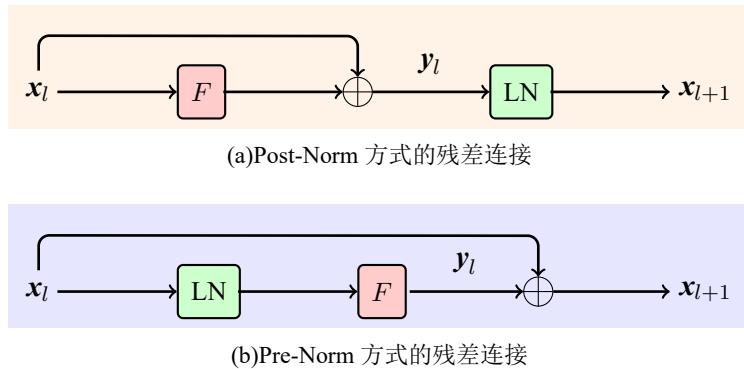


图 15.9 Post-Norm Transformer 与 Pre-Norm Transformer

令  $x_l$  和  $x_{l+1}$  表示第  $l$  个子层的输入和输出<sup>4</sup>,  $y_l$  表示中间的临时输出;  $\text{LN}(\cdot)$  表示层标准化操作, 帮助减小子层输出的方差, 从而让训练变得更稳定;  $F(\cdot)$  表示子层所对应的函数, 比如前馈神经网络、自注意力等。下面分别对 Post-Norm 和 Pre-Norm 进行简单的描述。

- **Post-Norm:** 早期的 Transformer 遵循的是 Post-Norm 结构<sup>[23]</sup>。也就是层标准化作用于每一子层的输入和输出的残差结果上, 如图15.9(a) 所示。可以表示如下:

$$x_{l+1} = \text{LN}(x_l + F(x_l; \theta_l)) \quad (15.24)$$

其中,  $\theta_l$  是子层  $l$  的参数。

- **Pre-Norm:** 通过调整层标准化的位置, 将其放置于每一子层的输入之前, 得到了 Pre-Norm 结构<sup>[813]</sup>, 如图15.9(b) 所示。这种结构也被广泛应用于最新的 Transformer 开源系统中<sup>[535, 692, 814]</sup>, 公式如下:

$$x_{l+1} = x_l + F(\text{LN}(x_l); \theta_l) \quad (15.25)$$

从上述公式中可以发现, 在前向传播过程中, Pre-Norm 结构可以通过残差路径将底层神经网络的输出直接暴露给上层神经网络。此外, 在反向传播过程中, 使用 Pre-Norm 结构也可以使得顶层网络的梯度更容易地反馈到底层网络。这里以一个含有  $L$  个子层的结构为例, 令 Loss 表示整个神经网络输出上的损失,  $x_L$  为顶层的输

<sup>4</sup>这里沿用 Transformer 中的定义, 每一层 (Layer) 包含多个子层 (Sub-layer)。比如, 对于 Transformer 编码器, 每一层包含一个自注意力子层和一个前馈神经网络子层。所有子层都需要进行层标准化和残差连接。

出。对于 Post-Norm 结构，根据链式法则，损失 Loss 相对于  $\mathbf{x}_l$  的梯度可以表示为：

$$\frac{\partial \text{Loss}}{\partial \mathbf{x}_l} = \frac{\partial \text{Loss}}{\partial \mathbf{x}_L} \times \prod_{k=l}^{L-1} \frac{\partial \text{LN}(\mathbf{y}_k)}{\partial \mathbf{y}_k} \times \prod_{k=l}^{L-1} \left(1 + \frac{\partial F(\mathbf{x}_k; \boldsymbol{\theta}_k)}{\partial \mathbf{x}_k}\right) \quad (15.26)$$

其中， $\prod_{k=l}^{L-1} \frac{\partial \text{LN}(\mathbf{y}_k)}{\partial \mathbf{y}_k}$  表示在反向传播过程中，经过层标准化得到的复合函数导数。 $\prod_{k=l}^{L-1} \left(1 + \frac{\partial F(\mathbf{x}_k; \boldsymbol{\theta}_k)}{\partial \mathbf{x}_k}\right)$  代表每个子层间残差连接的导数。

类似的，也能得到 Pre-Norm 结构的梯度计算结果，如下：

$$\frac{\partial \text{Loss}}{\partial \mathbf{x}_l} = \frac{\partial \text{Loss}}{\partial \mathbf{x}_L} \times \left(1 + \sum_{k=l}^{L-1} \frac{\partial F(\text{LN}(\mathbf{x}_k); \boldsymbol{\theta}_k)}{\partial \mathbf{x}_l}\right) \quad (15.27)$$

对比公式(15.26)和公式(15.27)可以看出，Pre-Norm 结构直接把顶层的梯度  $\frac{\partial \text{Loss}}{\partial \mathbf{x}_L}$  传递给下层，并且如果将公式(15.27)右侧展开，可以发现  $\frac{\partial \text{Loss}}{\partial \mathbf{x}_l}$  中直接含有  $\frac{\partial \text{Loss}}{\partial \mathbf{x}_L}$  部分。这个性质弱化了梯度计算对模型深度  $L$  的依赖；而如公式(15.26)右侧所示，Post-Norm 结构则包含一个与  $L$  相关的多项导数的积，伴随着  $L$  的增大更容易发生梯度消失和梯度爆炸问题。因此，Pre-Norm 结构更适于堆叠多层神经网络的情况。比如，使用 Pre-Norm 结构可以很轻松地训练一个 30 层（60 个子层）编码器的 Transformer 网络，并带来可观的 BLEU 提升。这个结果相当于标准 Transformer 编码器深度的 6 倍，相对的，用 Post-Norm 结构训练深层网络的时候，训练结果很不稳定，当编码器深度超过 12 层后很难完成有效训练<sup>[461]</sup>，尤其是在低精度设备环境下损失函数更容易出现发散情况。这里把使用 Pre-Norm 的深层 Transformer 模型称为 Transformer-Deep。

另一个有趣的发现是，**使用深层网络后，网络可以更有效地利用较大的学习率和较大的批量训练，大幅度缩短了模型达到收敛状态的时间**。相比于 Transformer-Big 等宽网络，Transformer-Deep 并不需要太大的隐藏层维度就可以取得更优的翻译品质<sup>[461]</sup>。也就是说，Transformer-Deep 是一个更“窄”更“深”的神经网络。这种结构的参数量比 Transformer-Big 少，系统运行效率更高。

此外研究人员发现当编码器使用深层模型之后，解码器使用更浅的模型依然能够维持很好的翻译品质。这是由于解码器也会对源语言信息进行加工和抽象，当编码器变深之后，解码器对源语言的加工就不那么重要了，因此可以减少解码器的深度。这样做了一个直接好处是：可以通过减少解码器的深度提高翻译速度。对于一些翻译延时敏感的场景，这种架构是极具潜力的<sup>[773, 774, 815]</sup>。

## 15.2.2 高效信息传递

尽管使用 Pre-Norm 结构可以很容易地训练深层 Transformer 模型，但从信息传递的角度看，Transformer 模型中第  $l$  层的输入仅仅依赖于前一层的输出。虽然残差连接可以跨层传递信息，但是对于很深的模型，整个模型的输入和输出之间仍需要经过很多次残差连接。

为了使上层的神经网络可以更加方便地访问下层神经网络的信息，最简单的方法是引入更多的跨层连接。一种方法是直接将所有层的输出都连接到最上层，达到聚合多层信息的目的<sup>[555, 556, 557]</sup>。另一种更加有效的方式是在网络前向计算的过程中建立当前层表示与之前层表示之间的关系，例如**动态线性聚合网络**<sup>[461]</sup>（Dynamic Linear Combination of Layers, DLCL）和**动态层聚合方法**<sup>[559]</sup>。这些方法的共性在于，在每一层的输入中不仅考虑前一层的输出，同时将前面所有层的中间结果（包括词嵌入表示）进行聚合，本质上利用稠密的层间连接提高了网络中信息传递的效率（前向计算和反向梯度计算）。而 DLCL 利用线性的层融合手段来保证计算的时效性，主要应用于深层神经网络的训练，理论上等价于常微分方程中的高阶求解方法<sup>[461]</sup>。此外，为了进一步增强上层神经网络对底层表示的利用，研究人员从多尺度的角度对深层的编码器进行分块，并使用 GRU 来捕获不同块之间的联系，得到更高层次的表示。该方法可以看作是对动态线性聚合网络的延伸。接下来分别对上述几种改进方法展开讨论。

## 1. 使用更多的跨层连接

图15.10描述了引入更多跨层连接的结构的方法。在模型的前向计算过程中，假设编码器总层数为  $L$ ，当完成编码器  $L$  层的逐层计算后，通过线性平均、加权平均等机制对模型的中间层表示进行融合，得到蕴含所有层信息的表示  $\mathbf{g}$ ，作为编码-解码注意力机制的输入，与总共有  $M$  层的解码器共同处理解码信息。

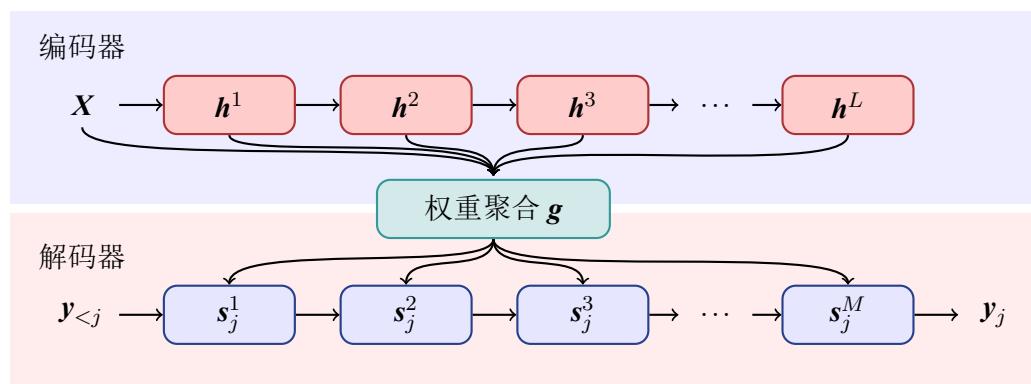


图 15.10 层融合方法

这里，令  $\mathbf{h}^i$  是编码器第  $i$  层的输出， $\mathbf{s}_j^k$  是解码器生成第  $j$  个单词时第  $k$  层的输出。层融合机制可以大致划分为如下几种：

- **线性平均**。即平均池化，通过对各层中间表示进行累加之后取平均值，表示如下：

$$\mathbf{g} = \frac{1}{L} \sum_{l=1}^L \mathbf{h}^l \quad (15.28)$$

- **权重平均。**在线性平均的基础上，为每一个中间层表示赋予一个相应的权重。权重的值通常采用可学习的参数矩阵  $\mathbf{W}_l$  表示。这种方法通常会略优于线性平均方法。可以用如下方式描述：

$$\mathbf{g} = \sum_{l=1}^L \mathbf{W}_l \mathbf{h}^l \quad (15.29)$$

- **前馈神经网络。**将之前中间层的表示进行级联，之后利用前馈神经网络得到融合的表示，如下：

$$\mathbf{g} = \text{FNN}([\mathbf{h}^1, \dots, \mathbf{h}^L]) \quad (15.30)$$

其中， $[.]$  表示级联操作。这种方式具有比权重平均更强的拟合能力。

- **基于多跳的自注意力机制。**如图15.11所示，其做法与前馈神经网络类似，首先将不同层的表示拼接成 2 维的句子级矩阵表示<sup>[530]</sup>。之后利用类似于前馈神经网络的思想将维度为  $\mathbb{R}^{d_{\text{model}} \times L}$  的矩阵映射到维度为  $\mathbb{R}^{d_{\text{model}} \times n_{\text{hop}}}$  的矩阵，如下：

$$\mathbf{o} = \sigma([\mathbf{h}^1, \dots, \mathbf{h}^L]^T \cdot \mathbf{W}_1) \mathbf{W}_2 \quad (15.31)$$

其中， $[\mathbf{h}^1, \dots, \mathbf{h}^L]$  是输入矩阵， $\mathbf{o}$  是输出矩阵， $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_a}$ ， $\mathbf{W}_2 \in \mathbb{R}^{d_a \times n_{\text{hop}}}$ ， $d_a$  表示前馈神经网络隐层大小， $n_{\text{hop}}$  表示跳数。之后使用 Softmax 函数计算不同层沿相同维度上的归一化结果  $\mathbf{u}_l$ ：

$$\mathbf{u}_l = \frac{\exp(\mathbf{o}_l)}{\sum_{i=1}^L \exp(\mathbf{o}_i)} \quad (15.32)$$

通过向量积操作得到维度为  $\mathbb{R}^{d_{\text{model}} \times n_{\text{hop}}}$  的稠密表示  $\mathbf{v}_l$ ：

$$\mathbf{v}_l = [\mathbf{h}^1, \dots, \mathbf{h}^L] \cdot \mathbf{u}_l \quad (15.33)$$

通过单层的前馈神经网络得到最终的融合表示：

$$\mathbf{g} = \text{FNN}([\mathbf{v}_1, \dots, \mathbf{v}_L]) \quad (15.34)$$

上述工作更多应用于浅层的 Transformer 模型，这种仅在编码器顶部使用融合机制的方法并没有在深层 Transformer 模型上得到有效的验证。主要原因是融合机制仅作用于编码器或解码器的顶层，对中间层的信息传递效率并没有显著提升。因此当网络深度较深时，这种方法的信息传递仍然不够高效。但这种“静态”的融合方式也为深层 Transformer 模型的研究奠定了基础。例如，可以使用透明注意力网络<sup>[556]</sup>，

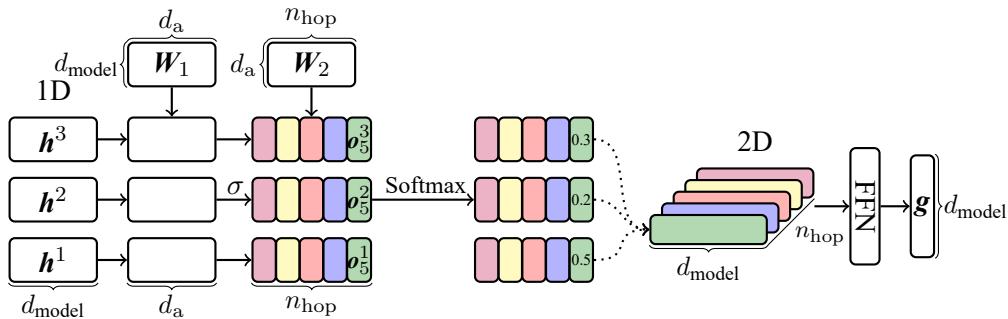


图 15.11 基于多跳注意力机制的层融合

即在权重平均的基础上，引入了一个权重矩阵。其核心思想是，让解码器中每一层的编码-解码注意力模块都接收不同比例的编码信息，而不是使用相同的融合表示。

## 2. 动态层融合

那如何进一步提高信息的传递效率？本节介绍的动态层融合可以更充分地利用之前层的信息，其神经网络连接更加稠密，模型表示能力更强<sup>[461, 462, 556]</sup>。以基于 Pre-Norm 结构的 DLCL 模型中的编码器为例，具体做法如下：

- 对于每一层的输出  $\mathbf{x}_l$ ，对其进行层标准化，得到每一层的信息表示，如下：

$$\mathbf{h}^l = \text{LN}(\mathbf{x}_l) \quad (15.35)$$

这里， $\mathbf{h}^0$  表示词嵌入层的输出  $\mathbf{X}$ ， $\mathbf{h}^l$  ( $l > 0$ ) 代表 Transformer 模型第  $l$  层的隐藏层表示。

- 定义一个维度为  $(L+1) \times (L+1)$  的权值矩阵  $\mathbf{W}$ ，矩阵中每一行表示之前各层对当前层的贡献度。令  $\mathbf{W}_{l,i}$  代表权值矩阵  $\mathbf{W}$  第  $l$  行第  $i$  列的权重，则第  $0 \sim l$  层的聚合结果为  $\mathbf{h}_i$  的线性加权和：

$$\mathbf{g}^l = \sum_{i=0}^l \mathbf{h}^i \times \mathbf{W}_{l,i} \quad (15.36)$$

$\mathbf{g}^l$  会作为输入的一部分送入第  $l+1$  层。其网络的结构如图 15.12 所示。

根据上述描述可以发现，权值矩阵  $\mathbf{W}$  每个位置的值由先前层对应的位置的值计算得到，因此该矩阵是一个下三角矩阵。开始时，对权值矩阵的每行进行平均初始化，即初始化矩阵  $\mathbf{W}_0$  的每一行各个位置的值为  $\frac{1}{\lambda}$ ， $\lambda \in (1, 2, \dots, L+1)$ 。伴随着神经网络的训练，不断更新  $\mathbf{W}$  中每一行不同位置权重的大小。

动态线性层聚合的一个好处是，系统可以自动学习不同层对当前层的贡献度。在实验中也发现，离当前层更近的部分的贡献度（权重）会更大，如图 15.13 所示，在

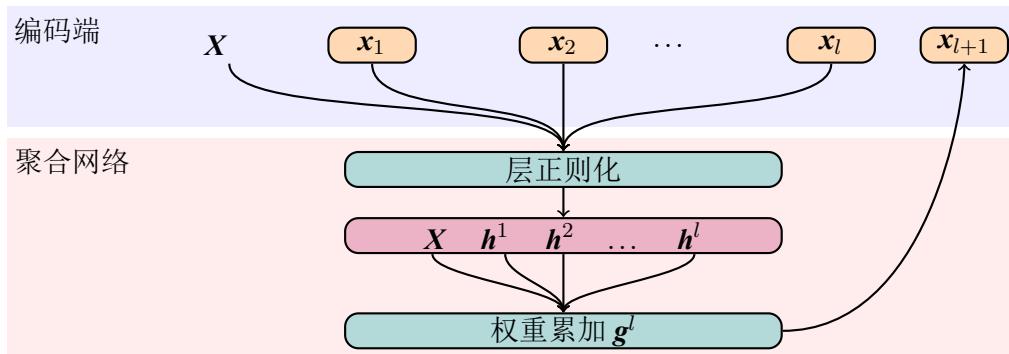
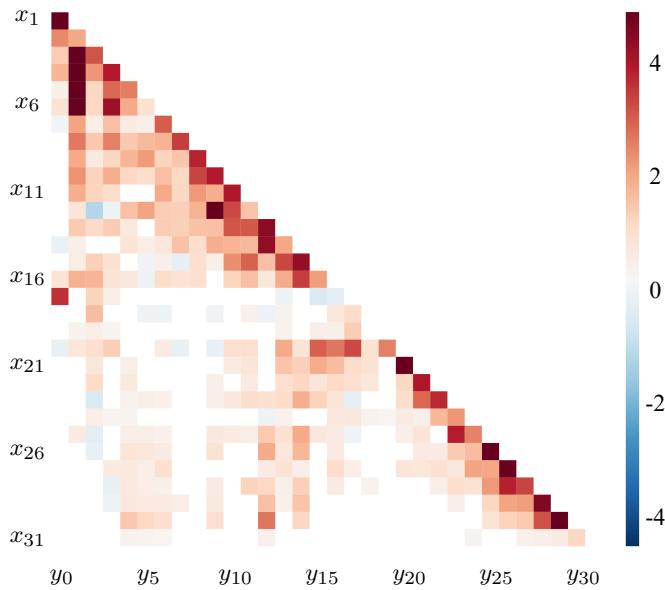


图 15.12 线性层聚合网络

每一行中颜色越深代表对当前层的贡献度越大。

除了动态层线性聚合方法，也可以利用更为复杂的胶囊网络<sup>[559]</sup>、树状层次结构<sup>[557]</sup>、多尺度协同框架<sup>[816]</sup>等作为层间的融合方式。然而，也有研究发现进一步增加模型编码器的深度并不能取得更优的翻译性能。因此如何进一步突破神经网络深度的限制是值得关注的研究方向，类似的话题在图像处理领域也引起了广泛的讨论<sup>[817, 818, 819, 820]</sup>。

图 15.13 对收敛的 DLCL 网络进行权重的可视化<sup>[461]</sup>

### 15.2.3 面向深层模型的参数初始化策略

对于深层神经机器翻译模型，除了神经网络结构的设计，合适的模型参数初始化策略同样十分重要。例如，Transformer 中参数矩阵采用了 Xavier 初始化方法<sup>[470]</sup>。

该方法可以保证在训练过程中各层激活函数的输出和梯度的方差的一致性，即同时保证每层在前向和反向传播时输入和输出的方差相同。但是，这类方法更多地被用于初始化浅层神经网络，在训练深层 Transformer 模型时表现不佳<sup>[470]</sup>。因此，研究人员也针对深层网络的参数初始化问题进行了探索，分为以下几种方法。

## 1. 基于深度缩放的初始化策略

随着神经网络层数的加深，输入的特征要经过很多的线性及非线性变换，受神经网络中激活函数导数值域范围和连乘操作的影响，常常会带来梯度爆炸或梯度消失的问题。这个问题的原因是过多地堆叠网络层数时，无法保证反向传播过程中每层梯度方差的一致性，因此在目前深层模型中采用的很多标准化方式（如层标准化、批次标准化等）都是从方差一致性的角度来解决问题，即将各层输出的取值范围控制在激活函数的梯度敏感区域，从而维持神经网络中梯度传递的稳定性。

为了说明问题，首先来看一看 Xavier 初始化方法如何对参数矩阵  $\mathbf{W}$  进行初始化<sup>[470]</sup>。具体做法为从一个均匀分布中进行随机采样：

$$\mathbf{W} \in \mathbb{R}^{n_i \times n_o} \sim u(-\gamma, \gamma) \quad (15.37)$$

$$\gamma = \sqrt{\frac{6}{n_i + n_o}} \quad (15.38)$$

其中， $u(-\gamma, \gamma)$  表示  $-\gamma$  与  $\gamma$  间的均匀分布， $n_i$  和  $n_o$  分别为线性变换  $\mathbf{W}$  中输入和输出的维度，也就是上一层神经元的数量和下一层神经元的数量。通过这种方式可以维持在前向与反向计算过程中输入与输出方差的一致性<sup>[821]</sup>。

令模型中某层神经元的输出表示为  $\mathbf{Z} = \sum_{j=1}^{n_i} w_j x_j$ 。可以看出， $\mathbf{Z}$  的核心是计算两个变量  $w_j$  和  $x_j$  乘积。两个变量乘积的方差的展开式为：

$$\text{Var}(w_j x_j) = E[w_j]^2 \text{Var}(x_j) + E[x_j]^2 \text{Var}(w_j) + \text{Var}(w_j) \text{Var}(x_j) \quad (15.39)$$

其中  $\text{Var}(\cdot)$  表示求方差操作，由于在大多数情况下，现有模型中的各种标准化方法可以维持  $E[w_j]^2$  和  $E[x_j]^2$  等于或者近似为 0。因此，模型中一层神经元输出的方差可以表示为：

$$\begin{aligned} \text{Var}(\mathbf{Z}) &= \sum_{j=1}^{n_i} \text{Var}(x_j) \text{Var}(w_j) \\ &= n_i \text{Var}(\mathbf{W}) \text{Var}(\mathbf{X}) \end{aligned} \quad (15.40)$$

通过观察公式(15.40)可以发现，在前向传播的过程中，当  $\text{Var}(\mathbf{W}) = \frac{1}{n_i}$  时，可以保证每层的输入和输出的方差一致。类似的，通过相关计算可以得知，为了保证模型中每一层的输入和输出的方差一致，反向传播时应有  $\text{Var}(\mathbf{W}) = \frac{1}{n_o}$ ，通过对两种情况取平均值，控制参数  $\mathbf{W}$  的方差为  $\frac{2}{n_i + n_o}$ ，则可以维持在前向与反向过程中输入与

输出方差的一致性。若将参数初始化为一个服从边界为  $[-a, b]$  的均匀分布，那么其方差为  $\frac{(b+a)^2}{12}$ ，为了达到  $\mathbf{W}$  的取值要求，初始化时应有  $a = b = \sqrt{\frac{6}{n_i + n_o}}$ 。

但是随着神经网络层数的增加，上述初始化方法已经不能很好地约束基于 Post-Norm 的 Transformer 模型的输出方差。当神经网络堆叠很多层时，模型顶层输出的方差较大，同时反向传播时顶层的梯度范数也要大于底层。因此，一个自然的想法是根据网络的深度对不同层的参数矩阵采取不同的初始化方式，进而强化对各层输出方差的约束，可以描述为：

$$\mathbf{W} \in \mathbb{R}^{n_i \times n_o} \sim u\left(-\gamma \frac{\alpha}{\sqrt{l}}, \gamma \frac{\alpha}{\sqrt{l}}\right) \quad (15.41)$$

其中， $l$  为对应的神经网络的深度， $\alpha$  为预先设定的超参数来控制缩放的比例。这样，可以通过缩减顶层神经网络输出与输入之间的差异，让激活函数的输入分布保持在一个稳定状态，以此来尽可能避免它们陷入梯度饱和区。

## 2. Lipschitz 初始化策略

15.2.1 节已经介绍，在 Pre-Norm 结构中每一个子层的输入为  $\mathbf{x}_{l+1}^{\text{pre}} = \mathbf{x}_l + \mathbf{y}_l$ ，其中  $\mathbf{x}_l$  为当前子层的输入， $\mathbf{y}_l$  为  $\mathbf{x}_l$  经过自注意力或前馈神经网络计算后得到的子层输出。在 Post-Norm 结构中，在残差连接之后还要进行层标准化操作，具体的计算流程为：

- **计算均值：**  $\mu = \text{mean}(\mathbf{x}_l + \mathbf{y}_l)$
- **计算方差：**  $\sigma = \text{std}(\mathbf{x}_l + \mathbf{y}_l)$
- **根据均值和方差对输入进行放缩，如下：**

$$\mathbf{x}_{l+1}^{\text{post}} = \frac{\mathbf{x}_l + \mathbf{y}_l - \mu}{\sigma} \cdot \mathbf{w} + \mathbf{b} \quad (15.42)$$

其中， $\mathbf{w}$  和  $\mathbf{b}$  为可学习参数。进一步将公式(15.42)展开后可得：

$$\begin{aligned} \mathbf{x}_{l+1}^{\text{post}} &= \frac{\mathbf{x}_l + \mathbf{y}_l}{\sigma} \cdot \mathbf{w} - \frac{\mu}{\sigma} \cdot \mathbf{w} + \mathbf{b} \\ &= \frac{\mathbf{w}}{\sigma} \cdot \mathbf{x}_{l+1}^{\text{pre}} - \frac{\mathbf{w}}{\sigma} \cdot \mu + \mathbf{b} \end{aligned} \quad (15.43)$$

可以看到相比于 Pre-Norm 的计算方式，基于 Post-Norm 的 Transformer 中子层的输出为 Pre-Norm 形式的  $\frac{\mathbf{w}}{\sigma}$  倍。当  $\frac{\mathbf{w}}{\sigma} < 1$  时， $\mathbf{x}_l$  较小，输入与输出之间差异过大，导致深层 Transformer 系统难以收敛。Lipschitz 初始化策略通过维持条件  $\frac{\mathbf{w}}{\sigma} > 1$ ，保证网络输入与输出范数一致，进而缓解梯度消失的问题<sup>[822]</sup>。一般情况下， $\mathbf{w}$  可以被初始化为 1，因此 Lipschitz 初始化方法最终的约束条件则为：

$$0 < \sigma = \text{std}(\mathbf{x}_l + \mathbf{y}_l) \leq 1 \quad (15.44)$$

### 3. T-Fixup 初始化策略

另外一种初始化方法是从神经网络结构与优化器的计算方式入手。Post-Norm 结构在 Warmup 阶段难以精确地估计参数的二阶动量，这导致了训练不稳定问题<sup>[823]</sup>。也就是，层标准化是导致深层 Transformer 难以优化的主要原因之一<sup>[461]</sup>。Post-Norm 方式下 Transformer 的底层网络，尤其是编码器的词嵌入层面临严重的梯度消失问题。出现该问题的原因在于，在不改变层标准化位置的条件下，Adam 优化器利用滑动平均的方式来估计参数的二阶矩，其方差是无界的。在训练阶段的前期，由于模型只能看到有限数量样本，因此很难有效地估计参数的二阶矩，导致反向更新参数时参数的梯度方差过大。

除了用 Pre-Norm 代替 Post-Norm 结构来训练深层网络，也可以采用去除 Warmup 策略并移除层标准化机制的方式，并对神经网络中不同的参数矩阵制定相应的缩放机制来保证训练的稳定性<sup>[823]</sup>。具体的缩放策略如下：

- 类似于标准的 Transformer 初始化方式，使用 Xavier 初始化方式来初始化除了词嵌入以外的所有参数矩阵。词嵌入矩阵服从  $N(0, d^{-\frac{1}{2}})$  的高斯分布，其中  $d$  代表词嵌入的维度。
- 对编码器中自注意力机制的参数矩阵以及前馈神经网络中所有参数矩阵进行缩放因子为  $0.67L^{-\frac{1}{4}}$  的缩放， $L$  为编码器层数。
- 对解码器中全部注意力机制的参数矩阵以及前馈神经网络中所有参数矩阵进行缩放因子为  $(9M)^{-\frac{1}{4}}$  的缩放，其中  $M$  为解码器层数。

这种初始化方法由于没有 Warmup 策略，学习率会直接从峰值根据参数的更新次数进行退火，大幅度增大了模型收敛的时间。因此，如何进一步解决该初始化方法下的模型收敛速度是比较关键的问题。

### 4. ADMIN 初始化策略

也有研究发现 Post-Norm 结构在训练过程中过度依赖残差支路，在训练初期很容易发生参数梯度方差过大的现象<sup>[812]</sup>。经过分析发现，虽然底层神经网络发生梯度消失是导致训练不稳定的重要因素，但并不是唯一因素。例如，标准 Transformer 模型中梯度消失的原因在于使用了 Post-Norm 结构的解码器。尽管通过调整模型结构解决了梯度消失问题，但是模型训练不稳定的问题仍然没有被很好地解决。研究人员观测到 Post-Norm 结构在训练过程中过于依赖残差支路，而 Pre-Norm 结构在训练过程中逐渐呈现出对残差支路的依赖性，这更易于网络的训练。进一步，从参数更新的角度出发，Pre-Norm 由于参数的改变导致网络输出变化的方差经推导后可以表示为  $O(\log L)$ ，而 Post-Norm 对应的方差为  $O(L)$ 。因此，可以尝试减小 Post-Norm 中由于参数更新导致的输出的方差值，从而达到稳定训练的目的。针对该问题，可以

采用两阶段的初始化方法。这里，可以重新定义子层之间的残差连接如下：

$$\mathbf{x}_{l+1} = \mathbf{x}_l \cdot \boldsymbol{\omega}_{l+1} + F_{l+1}(\mathbf{x}_l) \quad (15.45)$$

其两阶段的初始化方法如下所示：

- **Profiling 阶段:**  $\boldsymbol{\omega}_{l+1} = 1$ , 只进行前向计算, 无需进行梯度计算。在训练样本上计算  $F_{l+1}(\mathbf{x}_l)$  的方差
- **Initialization 阶段:** 通过 Profiling 阶段得到的  $F_{l+1}(\mathbf{x}_l)$  的方差来初始化  $\boldsymbol{\omega}_{l+1}$ :

$$\boldsymbol{\omega}_{l+1} = \sqrt{\sum_{j < l} \text{Var}[F_{l+1}(\mathbf{x}_l)]} \quad (15.46)$$

这种动态的参数初始化方法不受限于具体的模型结构, 有较好的通用性。

#### 15.2.4 深层模型的训练加速

尽管窄而深的神经网络比宽网络有更快的收敛速度<sup>[461]</sup>, 但伴随着训练数据的增加, 以及模型进一步的加深, 训练代价成为不可忽视的问题。例如, 在几千万甚至上亿的双语平行句对上训练一个 48 层的 Transformer 模型需要几周的时间才能达到收敛<sup>5</sup>。因此, 在保证模型性能不变的前提下, 高效地完成深层模型的训练也是至关重要的。

##### 1. 渐进式训练

所谓渐进式训练是指从浅层神经网络开始, 在训练过程中逐渐增加模型的深度。一种比较简单的方法是将模型分为浅层部分和深层部分, 之后分别进行训练, 最终达到提高模型翻译性能的目的<sup>[824]</sup>。

另一种方式是动态构建深层模型, 并尽可能复用浅层部分的训练结果<sup>[462]</sup>。假设开始的时候模型包含  $l$  层神经网络, 然后训练这个模型至收敛。之后, 直接拷贝这  $l$  层神经网络(包括参数), 并堆叠出一个  $2l$  层的模型。之后继续训练, 重复这个过程。进行  $n$  次之后就得到了  $(n+1) \times l$  层的模型。图15.14给出了在编码器上使用渐进式训练的示意图。

渐进式训练的好处在于深层模型并不是从头开始训练。每一次堆叠, 都相当于利用“浅”模型给“深”模型提供了一个很好的初始点, 这样深层模型的训练会更容易。

<sup>5</sup>训练时间的估算是在单台 8 卡 Titan V GPU 服务器上得到的

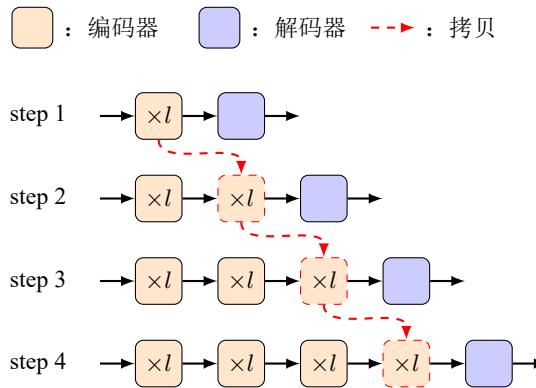


图 15.14 渐进式深层模型训练过程

## 2. 分组稠密连接

很多研究工作已经发现深层模型不同层之间的稠密连接能够很明显地提高信息传递的效率<sup>[461, 557, 824, 825]</sup>。与此同时，对之前层信息的不断复用有助于得到更好的表示，但也带来了计算代价过大的问题。在动态线性层聚合方法（DLCL）中，每一次聚合时都需要重新计算之前每一层表示对当前层输入的贡献度，因此伴随着编码器整体深度的增加，这部分的计算代价变得不可忽略。例如，一个基于动态层聚合的 48 层 Transformer 模型比不使用动态层聚合的模型在进行训练时慢近 2 倍。同时，缓存中间结果也增加了显存的使用量。比如，即使在使用半精度计算的情况下，每张 12G 显存的 GPU 上计算的词也不能超过 2048 个，这导致训练开销急剧增大。

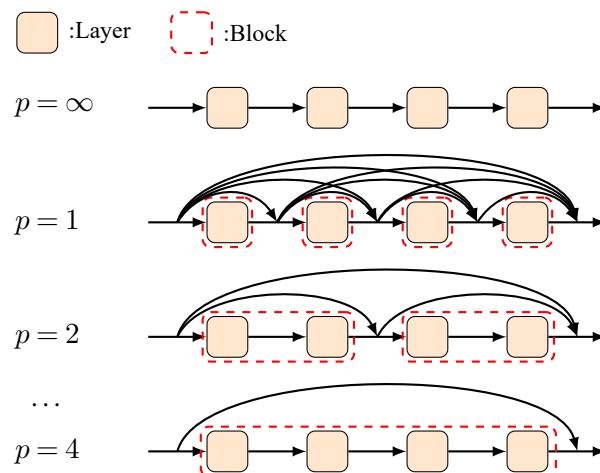


图 15.15 不同组之间的稀疏连接

缓解这个问题的一种方法是使用更稀疏的层间连接方式。其核心思想与动态线性层聚合是类似的，不同点在于可以通过调整层之间连接的稠密程度来降低训练代

价。比如，可以将每  $p$  层分为一组，之后动态线性层聚合只在不同组之间进行。这样，通过调节  $p$  值的大小可以控制神经网络中连接的稠密程度，作为一种训练代价与翻译性能之间的权衡。显然，标准的 Transformer 模型<sup>[23]</sup> 和 DLCL 模型<sup>[461]</sup> 都可以看作是该方法的一种特例。如图15.15所示：当  $p = 1$  时，每一个单独的块被看作一个独立的组，它等价于基于动态层聚合的 DLCL 模型；当  $p = \infty$  时，它等价于正常的 Transformer 模型。值得注意的是，如果配合渐进式训练。在分组稠密连接中可以设置  $p$  等于模型层数。

### 3. 学习率重置

尽管渐进式训练策略与分组稠密连接结构都可以加速深层模型的训练，但使用传统的学习率衰减策略会导致训练深层模型时的学习率较小，因此模型无法快速地达到收敛状态，同时也影响最终的模型性能。

图15.16中的红色曲线描绘了在 WMT 英德翻译任务上标准 Transformer 模型的学习率曲线，可以看到当模型训练到 40k 步时，学习率对比峰值有明显的差距，而此时刚开始训练最终的深层模型，过小的学习率并不利于后期深层网络的充分训练。

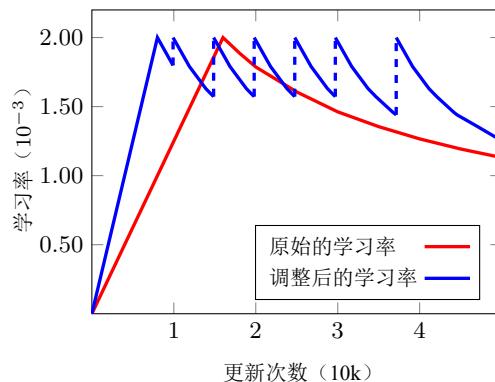


图 15.16 学习率重置 vs 从头训练的学习率曲线

针对该问题的一个解决方案是修改学习率曲线的衰减策略，如图15.16所示。图中蓝色的曲线是修改后的学习率曲线。首先在训练的初期让模型快速地达到学习率的峰值（线性递增），之后的每一次增加  $l$  层神经网络时，都会将当前的学习率值重置到峰值点。之后，根据训练的步数对其进行相应的衰减。具体的步骤如下：

- 在训练的初期，模型会先经历一个学习率预热的过程：

$$lr = d_{\text{model}}^{-0.5} \cdot step\_num \cdot warmup\_steps^{-0.5} \quad (15.47)$$

这里， $step\_num$  表示参数更新的次数， $warmup\_step$  表示预热的更新次数， $d_{\text{model}}$  表示 Transformer 模型的隐层大小， $lr$  是学习率。

- 在之后的训练过程中，每当增加模型深度时，学习率都会重置到峰值，之后进行相应的衰减：

$$lr = d_{\text{model}}^{-0.5} \cdot step\_num^{-0.5} \quad (15.48)$$

这里 *step\_num* 代表学习率重置后更新的步数。

综合使用渐进式训练、分组稠密连接、学习率重置策略可以保证在翻译品质不变的前提下，缩减近 40% 的训练时间<sup>[462]</sup>。同时，伴随着模型的加深与数据集的增大加速比也会进一步地增大。

### 15.2.5 深层模型的健壮性训练

伴随着网络的加深，模型的训练还会面临另外一个比较严峻的问题——过拟合。由于参数量的增大，深层模型的输入与输出分布之间的差异也会越来越大，然而不同子层之间的相互适应也会更加的明显，这将导致任意子层网络对其他子层的依赖过大。这种现象在训练阶段是有帮助的，因为不同子层可以协同工作从而更好地拟合训练数据。然而这种方式也降低了模型的泛化能力，即深层模型更容易陷入过拟合问题。

通常，可以使用 Dropout 手段用来缓解过拟合问题（见第十三章）。不幸的是，尽管目前 Transformer 模型使用了多种 Dropout 手段（如 Residual Dropout、Attention Dropout、ReLU Dropout 等），过拟合问题在深层模型中仍然存在。从图15.17中可以看到，**深层模型**比浅层模型在训练集和校验集的困惑度上都有明显的优势，然而模型**在训练一段时间后出现校验集困惑度上涨的现象**，说明模型已经过拟合于训练数据。

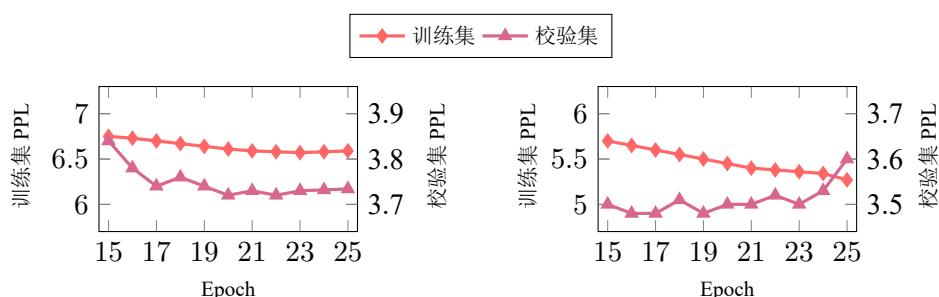


图 15.17 浅层模型（左）与深层模型（右）在 WMT16 英德翻译任务的校验集与训练集的困惑度

第十三章提到的 Layer Dropout 方法可以有效地缓解这个问题。以编码器为例，**Layer Dropout** 的过程可以被描述为：在训练过程中，对自注意力子层或前馈神经网络子层进行随机丢弃，以减少不同子层之间的相互适应。这里选择 Pre-Norm 结构作

为基础架构，它可以被描述为：

$$\mathbf{x}_{l+1} = F(\text{LN}(\mathbf{x}_l)) + \mathbf{x}_l \quad (15.49)$$

其中， $\text{LN}(\cdot)$  表示层标准化函数， $F(\cdot)$  表示自注意力机制或者前馈神经网络， $\mathbf{x}_l$  表示第  $l$  个子层的输入。之后，使用一个掩码 Mask（值为 0 或 1）来控制每个子层是正常计算还是丢弃。于是，该子层的计算公式可以被重写为：

$$\mathbf{x}_{l+1} = \text{Mask} \cdot F(\text{LN}(\mathbf{x}_l)) + \mathbf{x}_l \quad (15.50)$$

$\text{Mask} = 0$  代表该子层被丢弃，而  $\text{Mask} = 1$  代表正常进行当前子层的计算。图15.18展示了这个方法与标准 Pre-Norm 结构之间的区别。

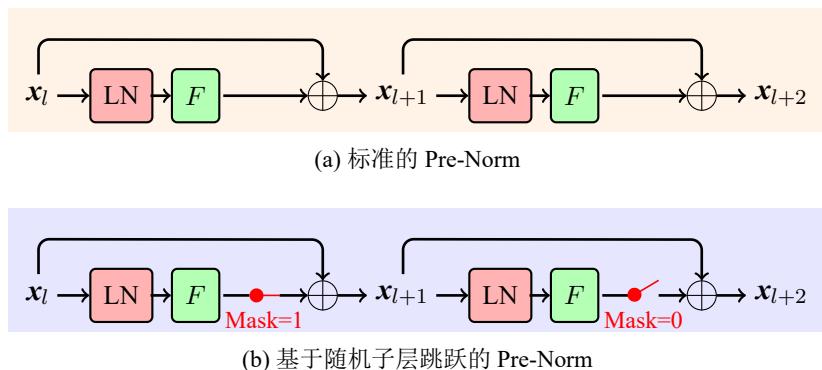


图 15.18 标准的 Pre-Norm 结构与基于随机子层跳跃的 Pre-Norm 结构

除此之外，在残差网络中，研究人员已经发现底层神经网络的作用是对输入进行抽象表示，而上层神经网络会进一步修正这种表示来拟合训练目标，因此底层神经网络对模型最终的输出有很大的影响<sup>[801]</sup>。该结论同样适用于 Transformer 模型，比如，在训练中，残差支路以及底层的梯度范数通常比较大，这也间接表明底层神经网络在整个优化的过程中需要更大的更新。考虑到这个因素，在设计每一个子层被丢弃的概率时，可以采用自底向上线性增大的策略，保证底层的神经网络相比于顶层更容易保留下。

## 15.3 基于句法的神经机器翻译模型

在统计机器翻译时代，使用句法信息是一种非常有效的机器翻译建模手段（见第八章）。由于句法是人类运用语言的高级抽象结果，使用句法信息（如句法树）可以帮助机器翻译系统对句子结构进行建模。例如，利用句法树提升译文语法结构的正确性。在神经机器翻译中，大多数框架均基于词串进行建模，因此在模型中引入

句法树等结构也很有潜力<sup>[826]</sup>。具体来说，由于传统神经机器翻译模型缺少对句子结构的理解，会导致一些翻译问题：

- **过度翻译问题**，如：

“两/个/女孩” → “two girls and two girls”

- **翻译不连贯问题**，如：

“新生/银行/申请/上市” → “new listing bank”

显然，神经机器翻译系统并没有按照合理的句法结构生成译文。也就是说，模型并没有理解句子的结构<sup>[826]</sup>。甚至对于一些语言差异很大的语言对，会出现将介词短语翻译成一个词的情况。虽然可以通过很多手段对上述问题进行求解，但是使用句法树是解决该问题的一种最直接的方法<sup>[827]</sup>。

那么在神经机器翻译中，如何将这种离散化的树结构融入到基于分布式表示的翻译模型中呢？有以下两种策略：

- **将句法信息加入到编码器**，使得编码器更加充分地表示源语言句子。
- **将句法信息加入到解码器**，使得翻译模型能生成更符合句法的译文。

### 15.3.1 编码器使用句法信息

在编码器中使用句法信息有两种思路，一种思路是在编码器中显性使用树结构进行建模，另一种思路是把句法信息作为特征输入到传统的序列编码器中。这两种思路与统计机器翻译中基于句法树结构的模型和基于句法特征的模型十分相似（见第八章）。

#### 1. 基于句法树结构的编码

使用句法信息的一种简单的方法是将源语言句子编码成一个二叉树结构<sup>6</sup>，树节点的信息是由左子树和右子树变换而来，如下所示：

$$\mathbf{h}_p = f_{\text{tree}}(\mathbf{h}_l, \mathbf{h}_r) \quad (15.51)$$

其中， $\mathbf{h}_l$  和  $\mathbf{h}_r$  分别代表了左孩子节点和右孩子节点的神经网络输出（隐层状态），通过一个非线性函数  $f_{\text{tree}}(\cdot, \cdot)$  得到父节点的状态  $\mathbf{h}_p$ 。图15.19 展示了一个基于树结构的循环神经网络编码器<sup>[827]</sup>。这些编码器由下自上组成了一个树型结构，这种树结构的具体连接形式由句法分析决定。其中  $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$  是输入序列所对应的循环神经单

<sup>6</sup>所有句法树都可以通过二义化方法转化为二叉树（见第八章）。

元（绿色部分）， $\{\mathbf{h}_{m+1}, \dots, \mathbf{h}_{2m-1}\}$  对应着树中的节点（红色部分），它的输出由其左右子节点通过公式(15.51)计算得到。对于注意力模型，图中所有的节点都会参与上下文向量的计算，因此仅需要对第十章所描述的计算方式稍加修改，如下：

$$\mathbf{C}_j = \sum_{i=1}^m \alpha_{i,j} \mathbf{h}_i + \sum_{i=m+1}^{2m-1} \alpha_{i,j} \mathbf{h}_i \quad (15.52)$$

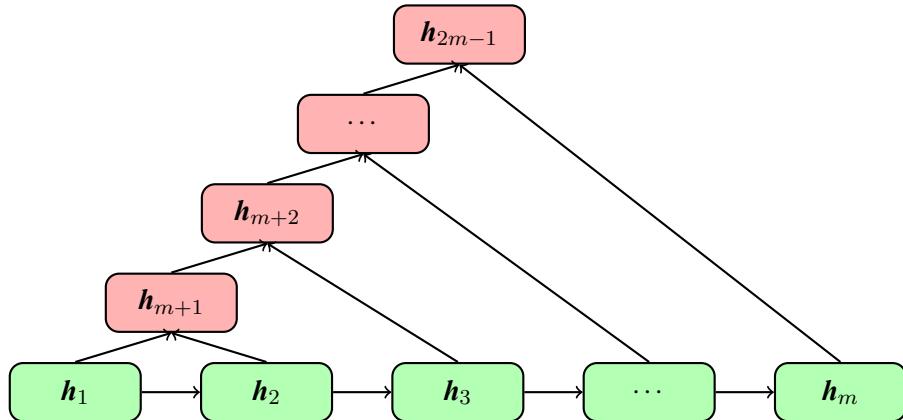


图 15.19 编码器树结构建模

其中， $\mathbf{C}_j$  代表生成第  $j$  个目标语言单词所需的源语言上下文表示。这样做的好处是编码器更容易将一个短语结构表示成一个单元，进而在解码器中映射成一个整体。比如，对于英语句子：

“I am having a cup of green tea.”

可以翻译成：

“私は/緑茶/を/飲んでいます。”

在标准的英语到日语的翻译中，英语短语 “a cup of green tea” 只会被翻译为 “綠茶” 一词。在加入句法树后，“a cup of green tea” 会作为树中一个节点，这样更容易把英语短语作为一个整体进行翻译。

只是，这种自底向上的树结构表示方法也存在问题：每个树节点的状态并不能包含树中其它位置的信息。也就是说，从每个节点上看，其表示结果没有很好地利用句法树中的上下文信息。因此，可以同时使用自下而上和自上而下的信息传递方式进行句法树的表示<sup>[431, 828]</sup>，这样增加了树中每个节点对其覆盖的子树以及周围上下文的建模能力。如图15.20所示， $\mathbf{h}^{\text{up}}$  和  $\mathbf{h}^{\text{down}}$  分别代表向上传输节点和向下传输节点的状态，虚线框代表了  $\mathbf{h}^{\text{up}}$  和  $\mathbf{h}^{\text{down}}$  会拼接到一起，并作为这个节点的整体表示参与注意力模型的计算。显然，自下而上的传递，可以保证句子的浅层信息（如短距离词汇搭配）被传递给上层节点，而自上而下的传递，可以保证句子上层结构的抽象被有效地传递给下层节点。这样，每个节点就同时含有浅层和深层句子表示的信息。

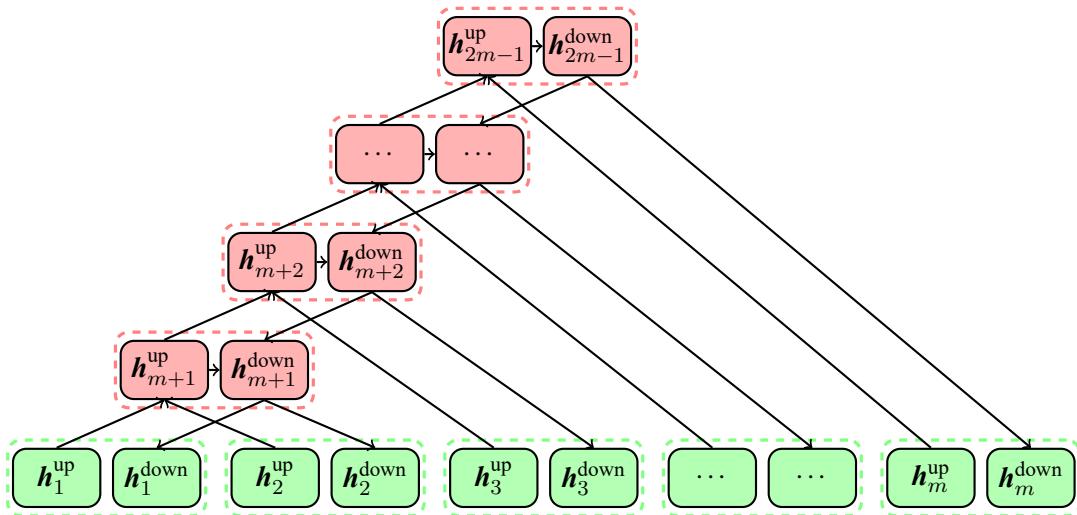


图 15.20 双向树结构编码模型

## 2. 基于句法特征的编码

不同于直接对树结构进行编码，另一种方法是将单词、句法信息等直接转换为特征向量拼接到一起，作为机器翻译系统的输入<sup>[829]</sup>。这种方法的优点在于，句法信息可以无缝融入到现有神经机器翻译框架，对系统结构的修改很小。以基于循环神经网络的翻译模型为例，可以用如下方式计算输入序列第  $i$  个位置的表示结果：

$$\mathbf{h}_i = \tanh(\mathbf{W}(\|_{k=1}^F \mathbf{E}_k x_{ik}) + \mathbf{U} \mathbf{h}_{i-1}) \quad (15.53)$$

其中， $\mathbf{W}$  和  $\mathbf{U}$  是线性变换矩阵， $F$  代表了特征的数量；而  $\mathbf{E}_k$  是一个特征嵌入矩阵，它记录了第  $k$  个特征不同取值对应的分布式表示； $x_{ik}$  代表了第  $i$  个词在第  $k$  个特征上的取值，于是  $\mathbf{E}_k x_{ik}$  就得到所激活特征的嵌入结果。 $\|$  操作为拼接操作，它将所有特征的嵌入结果拼接为一个向量。这种方法十分灵活，可以很容易地融合不同句法特征，例如，词根、子词、形态、词性以及依存关系等。

另一种方式是将句法信息的表示转化为基于序列的编码，之后与原始的词串融合。这样做的好处在于，并不需要使用基于树结构的编码器，而是直接复用基于序列的编码器即可。而句法信息可以在对句法树的序列化表示中学习得到。如图15.21(a)所示，对于英语句子 “I love dogs”，可以得到如图15.21(a)所示的句法树。这里，使用  $w_i$  表示第  $i$  个单词，如图15.21(b)所示。通过对句法树进行先序遍历，可以得到句法树节点的序列  $\{l_1, \dots, l_T\}$ ，其中  $T$  表示句法树中节点的个数， $l_j$  表示树中的第  $j$  个节点，如图15.21(c)所示。

在对句法树的树结构进行序列化的基础之上，可以用句法树节点与原始的词信息

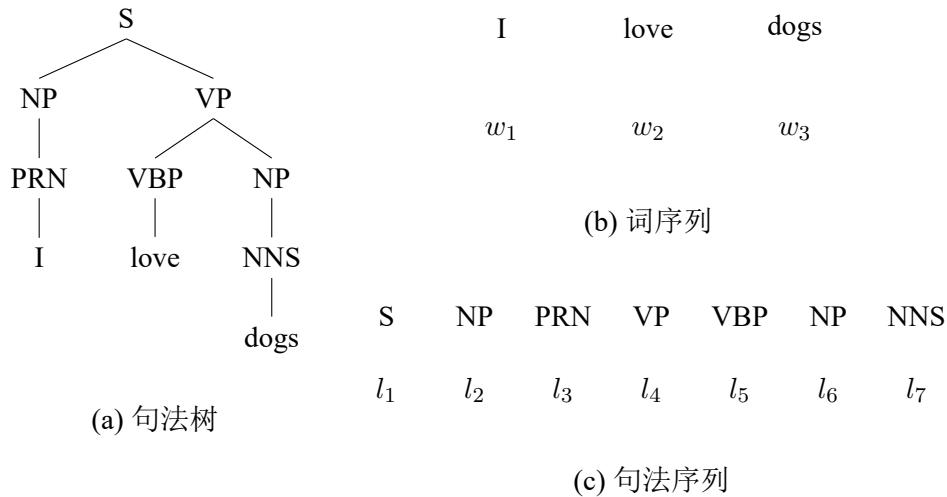


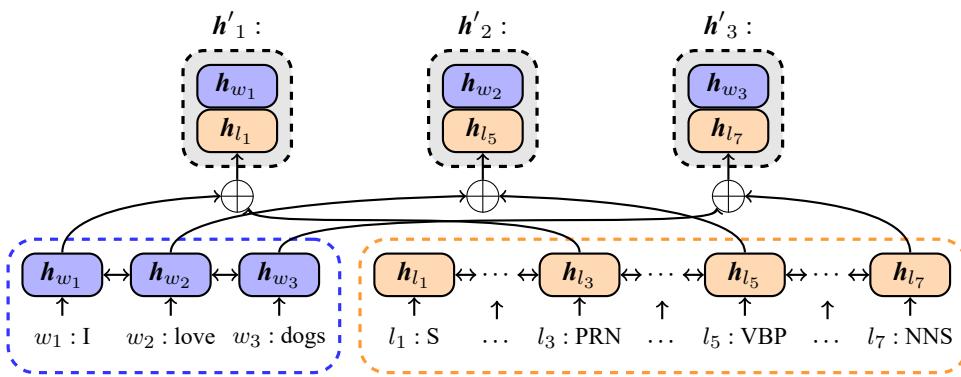
图 15.21 一个句子的句法树、词序列、句法树节点序列

一同构造出新的融合表示  $\mathbf{h}'_i$ , 并使用这种新的表示计算上下文向量, 如下:

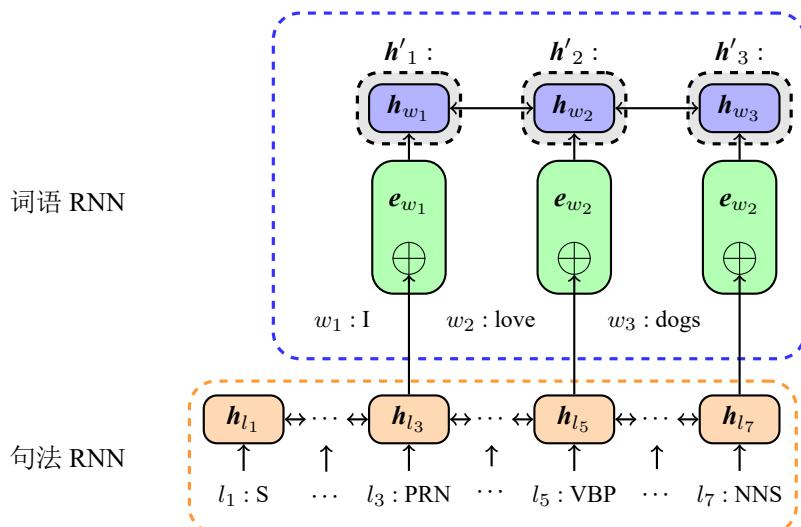
$$\mathbf{C}_j = \sum_{i=1}^m \alpha_{i,j} \mathbf{h}'_i \quad (15.54)$$

其中,  $m$  是源语言句子的长度。新的融合表示  $\mathbf{h}'_i$  有如下几种计算方式<sup>[826]</sup>:

- **平行结构。**利用两个编码器分别对源语言单词序列和线性化的句法树进行建模, 之后在句法树节点序列中寻找每个单词的父节点(或者祖先节点), 将这个单词和它的父节点(或者祖先节点)的状态相融合, 得到新的表示。如图15.22(a)所示, 图中  $\mathbf{h}_{w_i}$  为词  $w_i$  在单词序列中的状态,  $\mathbf{h}_{l_j}$  为树节点  $l_j$  在句法节点序列中的状态。如果单词  $w_i$  是节点  $l_j$  在句法树(图15.21(a))中的子节点, 则将  $\mathbf{h}_{w_i}$  和  $\mathbf{h}_{l_j}$  向量拼接到一起作为这个词的新的融合表示向量  $\mathbf{h}'_i$ ;
- **分层结构。**将句法表示结果与源语言单词的词嵌入向量进行融合, 如图15.22(b)所示, 其中  $\mathbf{e}_{w_i}$  为第  $i$  个词的词嵌入。类似地, 如果单词  $w_i$  是节点  $l_j$  在句法树(图15.21(a))中的子节点, 则将  $\mathbf{e}_{w_i}$  和  $\mathbf{h}_{l_j}$  向量拼接到一起作为原始模型的输入, 这样  $\mathbf{h}'_i$  直接参与注意力计算。注意, 分层结构和平行结构的区别在于, 分层结构最终还是使用了一个编码器, 句法信息只是与词嵌入进行融合, 因此最终的结构和原始的模型是一致的; 平行结构相当于使用了两个编码器, 因此单词和句法信息的融合是在两个编码器的输出上进行的;
- **混合结构。**首先对图15.21(a)中句法树进行先序遍历, 将句法标记和源语言单词融合到同一个序列中, 得到如图15.22(c)所示序列。之后使用传统的序列编码器对这个序列进行编码, 然后使用序列中源语言单词所对应的状态参与注意力



(a) 平行结构

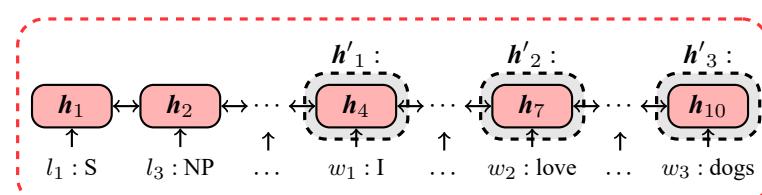


(b) 分层结构

先序遍历句法树，得到序列: S NP PRN I VP VBP love NP NNS dogs

$l_1 \quad l_2 \quad l_3 \quad w_1 \quad l_4 \quad l_5 \quad w_2 \quad l_6 \quad l_7 \quad w_3$

混合 RNN



(c) 混合结构

图 15.22 三种对树结构信息的融合方式

模型的计算。有趣的是，相比于前两种方法，这种方法参数量少而且也十分有效<sup>[826]</sup>。

需要注意的是，句法分析的错误会在很大程度上影响源语言句子的表示结果。如果获得的句法分析结果不够准确，可能会对翻译系统带来负面影响。此外，也有研究发现基于词串的神经机器翻译模型本身就能学习到一些源语言的句法信息<sup>[830]</sup>，这表明了神经机器翻译模型也有一定的归纳句子结构的能力。除了循环神经网络结构，也有研究人员探索了如何在 Transformer 中引入树结构信息。比如，可以将词与词之间的依存关系距离作为额外的语法信息融入到注意力模型中<sup>[831]</sup>。

### 15.3.2 解码器使用句法信息

在解码器中使用句法信息，一种最直接的方式是将目标语言句法树结构进行线性化，然后目标语言句子就变成了一个含有句法标记和单词的混合序列。这样，神经机器翻译系统不需要进行修改，可以直接使用句法树序列化的结果进行训练和推断<sup>[445]</sup>。图15.23展示了一个目标语言句法树经过线性化后的结果。

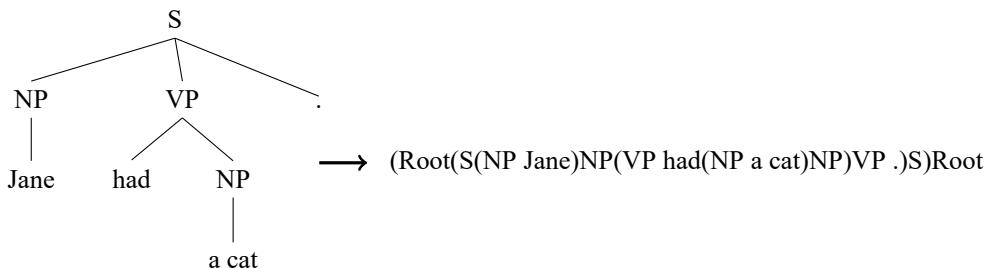


图 15.23 句法树线性化示例

不过，直接使用序列化的句法树也会带来新的问题。比如，在推断时，生成的译文序列可能根本不对应合法的句法树。此时，需要额外的模块对结果进行修正或者调整，以得到合理的译文。

另一种方法是直接在目标语言端使用句法树进行建模。与源语言句法树的建模不同，目标语言句法树的生成伴随着译文的生成，因此无法像源语言端一样将整个句法树一起处理。这样译文生成问题本质上就变成了目标语言树结构的生成，从这个角度说，这个过程与统计机器翻译中串到树的模型是类似的（见第八章）。树结构的生成有很多种策略，但基本的思想类似，可以根据已经生成的局部结构预测新的局部结构，并将这些局部结构拼装成更大的结构，直到得到完整的句法树结构<sup>[832]</sup>。

实现目标语言句法树生成的一种手段是将形式文法扩展，以适应分布式表示学习框架。这样，可以使用形式文法描述句法树的生成过程（见第三章），同时利用分布式表示来进行建模和学习。比如，可以使用基于循环神经网络的文法描述方法，把句法分析过程看作是一个循环神经网络的执行过程<sup>[833]</sup>。此外，也可以从**多任务学习**

(Multitask Learning)出发,用多个解码器共同完成目标语言句子的生成<sup>[834]</sup>。图15.24展示了由一个编码器(汉语)和多个解码器组成的序列生成模型。其中不同解码器分别负责不同的任务:第一个用于预测翻译结果,即翻译任务;第二个用于句法分析任务;第三个用于语言理解任务,生成汉语上下文。其设计思想是各个任务之间能够相互辅助,使得编码器的表示能包含更多的信息,进而让多个任务都获得性能提升。这种方法也可以使用在多个编码器上,其思想是类似的。

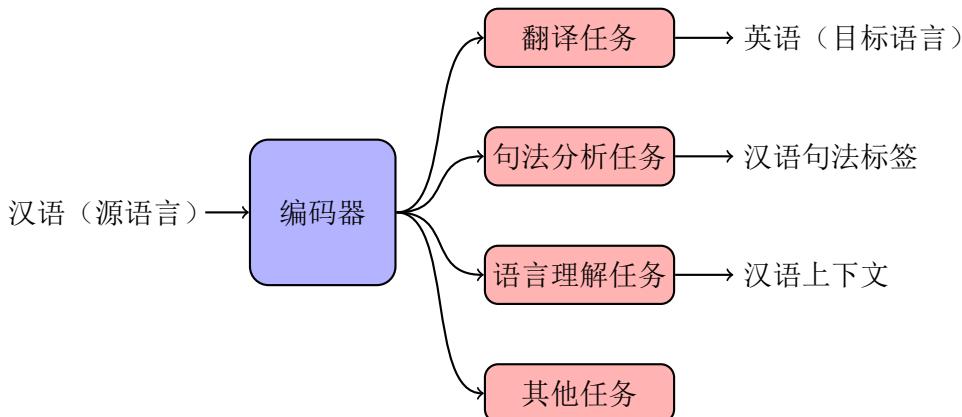


图 15.24 融合句法信息的多任务学习

不过,融合树结构和目标语言词串的方法也存在问题:它会导致目标语言端的序列过长,使得模型难以训练。为了缓解这个问题,可以使用两个模型,一个生成句子,另一个生成树结构<sup>[481, 835]</sup>。以生成目标语言依存树为例,生成依存树的模型仍然是一个移进-规约序列的生成模型,称为动作模型。另一个模型负责预测目标语言词序列,称为词预测模型,它只有在第一个模型进行移位操作的时候才会预测一下词,同时会将当前词的状态送入到第一个模型中。整个过程如图15.25所示,这里使用循环神经网络构建了动作模型和词预测模型。 $h_i^{\text{action}}$  表示动作模型的隐藏层状态, $h_i^{\text{word}}$  表示词预测模型的隐藏层状态。动作模型会结合词预测模型的状态预测出“移位”,“左规约”,“右规约”三种动作,只有当动作模型预测出“移位”操作时,词预测模型才会预测下一时刻的词语;而动作模型预测“左规约”和“右规约”相当于完成了依存关系的预测(依存树见图15.25右侧)。最后词预测模型预测出结束符号<eos>时,整个过程结束。

相较于在编码器中融入句法信息,在解码器中融入句法信息更为困难。由于树结构与单词的生成是一个相互影响的过程,如果先生成树结构,再根据树得到译文单词串,那么一旦树结构有误,翻译结果就会有问题。在统计机器翻译中,句法信息究竟应该使用到什么程度已经有一些讨论<sup>[370, 400]</sup>。而在神经机器翻译中,如何更有效地引入树结构信息以及如何平衡树结构信息与词串的作用还有待确认。如前文所述,基于词串的神经机器翻译模型已经能够捕捉到一些句法结构信息<sup>[830]</sup>,虽然有些信息是不容易通过人的先验知识进行解释的。这时,使用人工总结的句法结构来约束或

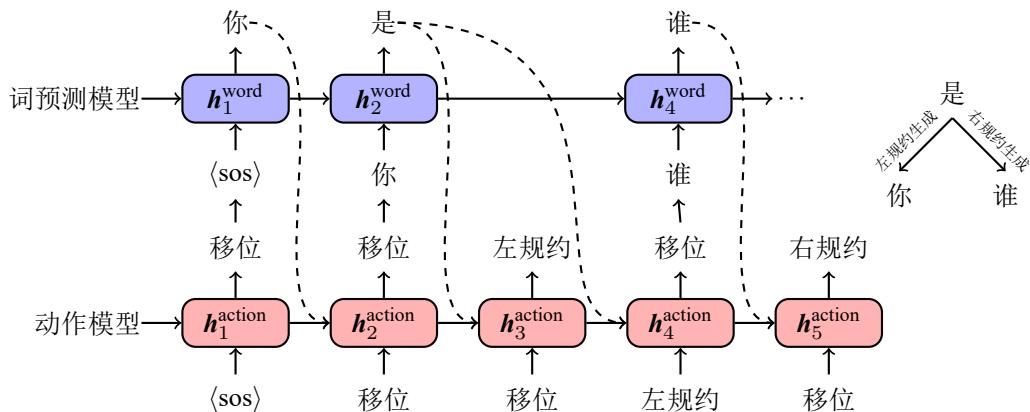


图 15.25 词预测模型和动作模型

者强化翻译模型，是否可以补充模型无法学到的信息，还是需要进一步研究。

## 15.4 基于结构搜索的翻译模型优化

理想中，人们希望计算机能够自动地找到最适用于当前任务的神经网络模型结构。这种方法也被称作**神经架构搜索**（Neural Architecture Search），在神经网络模型中有时也被称作神经网络结构搜索，或简称网络结构搜索<sup>[836, 837, 838]</sup>。

### 15.4.1 神经网络结构搜索

网络结构搜索属于**自动机器学习**（Automated Machine Learning）的范畴，其目的是根据对应任务上的数据找到最合适的数据结构。在这个过程中，模型结构就像神经网络中的参数一样被自动地学习出来。图15.26(a)展示了人工设计的Transformer编码器的局部结构，图15.26(b)给出对该结构使用进化算法优化后得到的结构<sup>[795]</sup>。可以看到，使用网络结构搜索方法得到的模型中，出现了与人工设计的结构不同的跨层连接，同时还搜索到了全新的多分支结构，这种结构也是人工不易设计出来的。

那么网络结构搜索究竟是一种什么样的技术呢？如图15.27所示，在传统机器学习方法中，研究人员需要设计大量的特征来描述待解决的问题，即“特征工程”。在深度学习时代，神经网络模型可以完成特征的抽取和学习，但是却需要人工设计神经网络结构，这项工作仍然十分繁重。因此一些科研人员开始思考，能否将设计模型结构的工作也交由机器自动完成？深度学习方法中模型参数能够通过梯度下降等方式进行自动优化，那么模型结构是否可以也看做是一种特殊的参数，使用搜索算法自动找到最适用于当前任务的模型结构？基于上述想法，网络结构搜索应运而生。

早在上世纪八十年代，研究人员就开始使用进化算法对神经网络结构进行设计<sup>[839]</sup>，也引发了之后的很多探索<sup>[840, 841, 842]</sup>。近些年，随着深度学习技术的发展，网络结构搜索技术在很多任务中受到关注。例如，网络结构搜索就很好地应用在语言

建模上，并取得了很好的结果<sup>[843, 844, 845]</sup>。下面将对网络结构搜索的基本方法和其在机器翻译中的应用进行介绍。

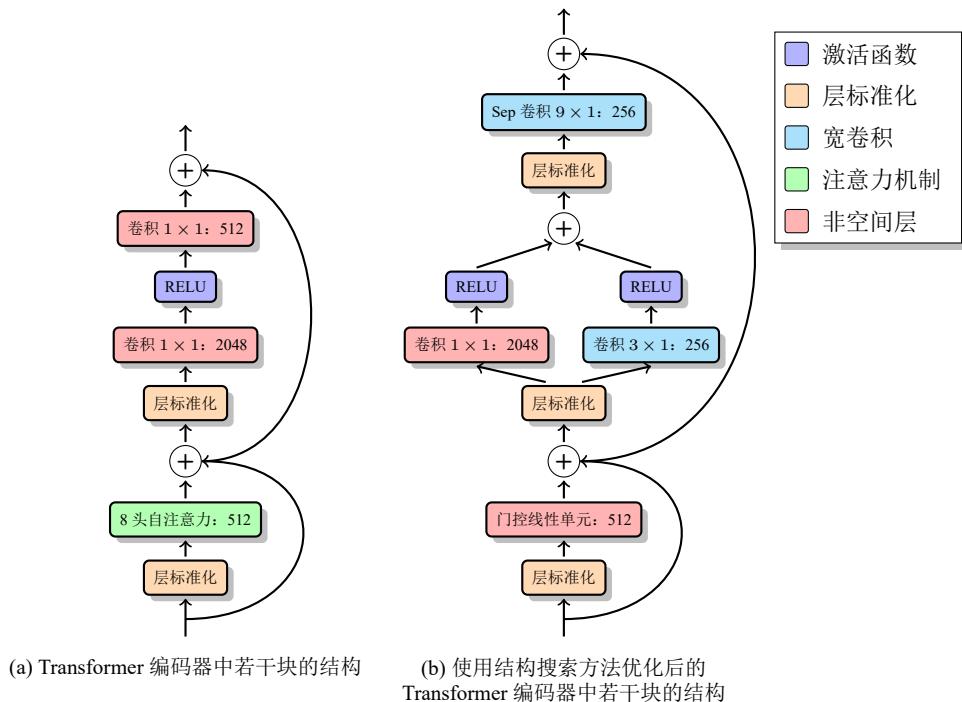


图 15.26 传统 Transformer 和通过网络结构搜索方法优化后的 Transformer<sup>[795]</sup>

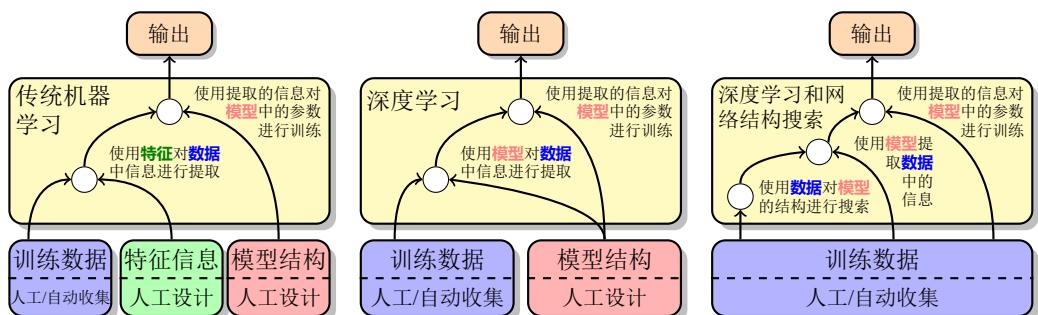


图 15.27 机器学习范式对比

## 15.4.2 结构搜索的基本方法

对于网络结构搜索任务来说，目标是通过数据驱动的方式自动地找到最合适的模型结构。以有监督学习为例，给定训练集合  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ （其中  $x_i$  表示的是第  $i$  个样本的输入， $y_i$  表示该样本的答案，并假设  $x_i$  和  $y_i$  均为向量表示），网络结

构搜索过程可以被建模为根据数据找到最佳模型结构  $\hat{a}$  的过程，如下所示：

$$\hat{a} = \arg \max_a \sum_{i=1}^n P(\mathbf{y}_i | \mathbf{x}_i; a) \quad (15.55)$$

其中， $P(\mathbf{y}_i | \mathbf{x}_i; a)$  为模型  $a$  观察到数据  $\mathbf{x}_i$  后预测  $\mathbf{y}_i$  的概率，而模型结构  $a$  本身可以看作是输入  $\mathbf{x}$  到输出  $\mathbf{y}$  的映射函数。图15.28展示了神经网络结构搜索方法的主要流程，其中包括三个部分：设计搜索空间、选择搜索策略以及进行性能评估，下面将对上述各个部分进行简要介绍。

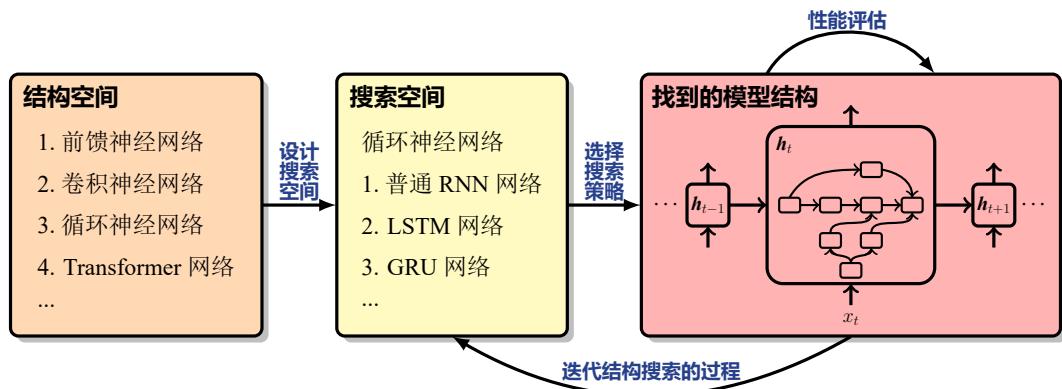


图 15.28 神经网络结构搜索的主要流程

## 1. 搜索空间

对搜索空间建模是结构搜索任务中的基础部分。如图15.29所示，结构空间中包含着所有潜在的模型结构。图15.29以结构之间的相似性为衡量指标对模型结构在搜索空间中的相对位置进行了刻画，同时颜色的深浅表示了该结构在指定任务下的性能情况。可以看到对于特定任务来说，性能较好的模型结构往往会聚集在一起。因此，在研究人员设计搜索空间的时候，为了增加找到最优结构的可能性，往往会根据经验或者实验将易产出高性能模型结构的区域设定为搜索空间。以自然语言处理任务为例，最初的网络结构搜索工作主要对基于循环神经网络构成的搜索空间进行探索<sup>[836, 843, 846]</sup>，而近些年，在 Transformer 模型的基础上进行结构搜索也引起了研究人员的广泛关注<sup>[795, 847, 848]</sup>。

另一个很重要的问题是如何表示一个网络结构。在目前的结构搜索方法中，通常将模型结构分为整体框架和内部结构（元结构）两部分。整体框架将若干内部结构的输出按照特定的方式组织起来，最终得到模型输出。

- **整体框架**。如图15.29所示，整体框架一般基于经验进行设计。比如，对于包括机器翻译在内的自然语言处理任务而言，一般会更倾向于使用循环神经网络或

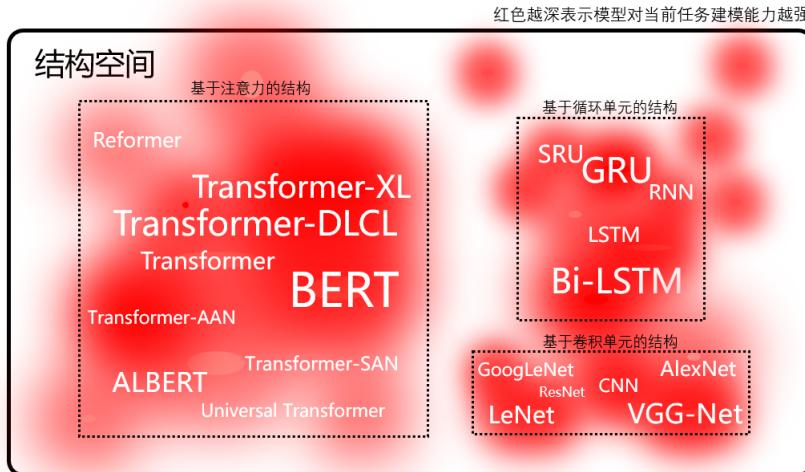


图 15.29 结构空间内结构之间的关系

Transformer 模型的相关结构作为搜索空间<sup>[795, 836, 843]</sup>。

- **内部结构。**对于内部结构的设计需要考虑到搜索过程中的最小搜索单元，以及搜索单元之间的连接方式。最小搜索单元指的是在结构搜索过程中可被选择的最小独立计算单元，在不同搜索空间的设计中，最小搜索单元的颗粒度各有不同，较小的搜索粒度主要包括如矩阵乘法、张量缩放等基本数学运算<sup>[849]</sup>，更大粒度的搜索单元包括常见的激活函数以及一些局部结构，如 ReLU、注意力机制等<sup>[513, 844, 850]</sup>。不过，对于搜索颗粒度的问题，目前还缺乏有效的方法针对不同任务进行自动优化。

## 2. 搜索策略

在定义好搜索空间之后，如何进行网络结构的搜索也同样重要。该过程被称为搜索策略的设计，其主要目的是根据已找到的模型结构计算出下一个最有潜力的模型结构，为保证模型有效性，在一些方法中也会引入外部知识（如经验性的模型结构或张量运算规则）对搜索过程进行剪枝。目前常见的搜索策略一般包括基于进化算法的方法、基于强化学习的方法以及基于梯度的方法等等。

- **进化算法。**进化算法最初被用来对神经网络模型结构、以及其中的权重参数进行优化<sup>[839, 851, 852]</sup>。随着最优化算法的发展，近年来，对于网络参数的学习开始更多地采用梯度下降的方式，但是进化算法依旧被用于对模型结构进行优化<sup>[853, 854, 855]</sup>。从结构优化的角度来说，一般是将模型结构看做遗传算法中种群的个体，使用轮盘赌或锦标赛等抽取方式，对种群中的结构进行取样作为亲本，之后通过亲本模型的突变产生新的模型结构，最终对这些新的模型结构进行适应度评估。根据模型结构在校验集上的性能确定是否将其加入种群。

- **强化学习。**强化学习方法在第十三章已经进行了介绍，这里可以将神经网络结构的设计看做是一种序列生成任务，使用字符序列对网络结构进行表述<sup>[836]</sup>。这种方法的执行过程如图15.30所示。其执行过程为由智能体对模型结构进行生成，之后将生成的结构应用于对应的任务（如机器翻译、语言建模等），根据模型在对应任务中的输出以及表现水平来进一步对智能体进行反馈，促使智能体生成更适用于当前任务的模型结构。

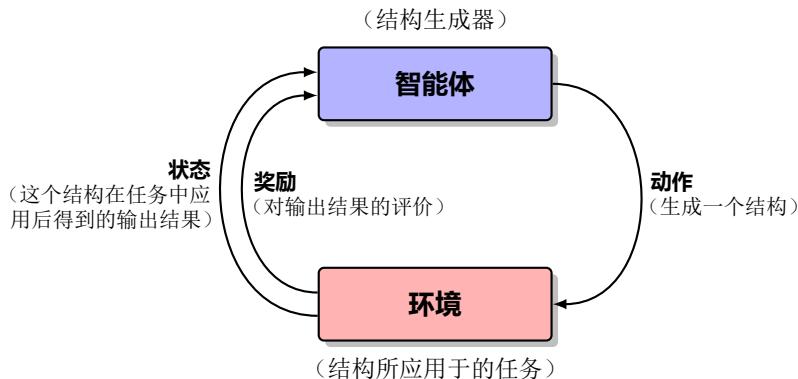


图 15.30 基于强化学习的结构搜索

- **基于梯度的方法。**这种方法的思想是在连续空间中对模型结构进行表示<sup>[843]</sup>，通常将模型结构建模为超网络中的结构参数，接下来使用基于梯度的方法对超网络中的参数进行优化，最终根据其中的结构参数离散出最终的模型结构，达到结构搜索的目的，整体过程如图15.31所示。基于梯度的方法十分高效，因此也受到了很多关注<sup>[844, 856, 857]</sup>。

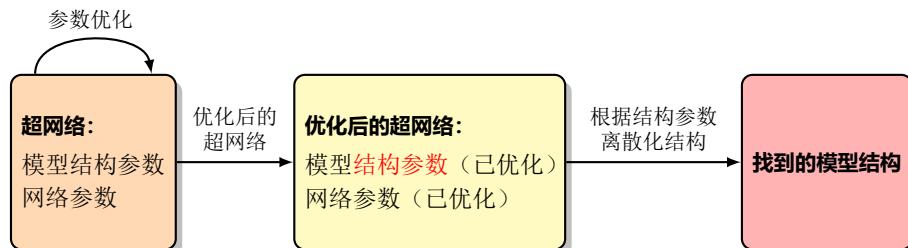


图 15.31 基于梯度方法的结构搜索

### 3. 性能评估

由于结构搜索过程中会产生大量的中间结构，因此需要快速评估这些结构的性能优劣，以保证在搜索中可以有效地挑选高质量的模型结构。对于该问题，可以从以下三个方面来考虑：

- **数据以及超参数的调整。**具体来说，可以用少量的数据训练模型，以便快速评估其性能<sup>[858, 859]</sup>。在超参数的调整方面，也可以通过减少模型训练轮数、减少模型的层数等方式来简化模型参数，达到加速训练、评估的目的<sup>[837, 838, 860]</sup>。
- **现有参数的继承及复用。**通过在现有的模型参数基础上，继续优化中间过程产生的模型结构，来加快待评价模型的收敛进程<sup>[853, 854, 861]</sup>。这种方式无需从头训练搜索过程中产生的中间结构，通过“热启动”的方式对模型参数进行优化，能够大幅减少性能评估过程的时间消耗。
- **模型性能的预测。**这种方式使用训练过程中的性能变化曲线来预估模型是否具有潜力，从而快速终止低性能模型的训练过程<sup>[862, 863, 864]</sup>。

### 15.4.3 机器翻译任务下的结构搜索

对于自然语言处理任务来说，网络结构搜索方法更多是在语言建模、命名实体识别等任务上进行尝试<sup>[844, 845]</sup>。其中，大多数工作是在基于循环神经网络的模型结构上进行探索的，相较于目前在机器翻译领域中广泛使用的Transformer模型结构来说，这些搜索到的结构在性能上并没有体现出绝对的优势。此外，由于机器翻译任务的复杂性，针对基于Transformer的机器翻译模型的结构搜索方法会更少一些。不过仍有部分工作在机器翻译任务上取得了很好的表现。例如，在WMT19机器翻译比赛中，神经网络结构优化方法在多个任务上取得了很好的成绩<sup>[865, 866]</sup>。对于结构搜索在机器翻译领域的应用，目前主要包括两个方面：分别是模型性能的改进以及模型效率的优化。

#### 1. 模型性能改进

结构搜索任务中一个非常重要的目标是找到更加适用于当前任务的模型结构。目前来看，有两种思路：

- **搜索模型中的局部结构。**在机器翻译任务中，一种典型的局部模型结构搜索方法是面向激活函数的搜索<sup>[867]</sup>，该方法将激活函数看作是一元、二元函数的若干次复合。例如，Swish激活函数就是一种被找到的新的激活函数，如下：

$$f(x) = x \cdot \delta(\beta x) \quad (15.56)$$

$$\delta(z) = (1 + \exp(-z))^{-1} \quad (15.57)$$

相比于人工设计的激活函数ReLU而言，Swish函数在多个机器翻译任务取得了不错的效果。

- **搜索模型中局部结构的组合。**在基于Transformer模型的网络结构搜索任务中，对于局部结构的组合方式的学习也受到了很多关注，其中包括基于进化算法的方法和基于梯度对现有Transformer模型结构的改良<sup>[795, 850]</sup>。与前文所述的对局部

结构的改良不同，此处更多地是对现有经验性的局部结构进行组合，找到最佳的整体结构。在模型结构的表示方法上，这些方法会根据先验知识为搜索单元设定一个部分框架，如每当信息传递过来之后先进行层标准化，之后再对候选位置上的操作使用对应的搜索策略进行搜索。另外这类方法也会在 Transformer 结构中引入多分支结构，一个搜索单元的输出可以被多个后续单元所使用，这种方式有效扩大了结构搜索过程中的搜索空间，能够在现有 Transformer 结构的基础上找到更优的模型结构。

此外对模型结构中超参数的自动搜索同样能够有效提升模型的性能<sup>[868]</sup>，这种方法在机器翻译中也有应用<sup>[464]</sup>。

## 2. 模型效率优化

网络结构搜索除了能够提高机器翻译模型性能之外，也能够优化模型的执行效率。从实用的角度出发，可以在进行结构搜索的同时考虑设备的计算能力，希望找到更适合运行设备的模型结构。同时，网络结构搜索也可以用来对大模型进行压缩，增加其在推断过程中的效率，这方面的工作不仅限于在机器翻译模型上，也有部分工作对基于注意力机制的预训练模型进行压缩。

- **面向特定设备的模型结构优化。**可以在结构优化的过程中将设备的算力作为一个约束<sup>[848]</sup>。具体来说，可以将搜索空间中各种结构建模在同一个超网络中，通过权重共享的方式进行训练。使用设备算力约束子模型，并通过进化算法对子模型进行搜索，搜索到适用于目标设备的模型结构。该方法搜索到的模型能够在保证模型性能不变前提下获得较大的效率提升。
- **模型压缩。**此外，在不考虑设备算力的情况下，也可以通过结构搜索方法对基于 Transformer 的预训练模型进行压缩。例如，将 Transformer 模型拆分为若干小组件，然后通过基于采样的结构搜索方法对压缩后的模型结构进行搜索，尝试找到最优且高效的推断模型<sup>[869]</sup>。类似的，也可以在基于 BERT 的预训练模型上通过结构搜索方法进行模型压缩，通过基于梯度的结构搜索方法，针对不同的下游任务将 BERT 模型压缩为小模型<sup>[847]</sup>。

虽然由于算力等条件的限制，目前很多网络结构搜索方法并没有直接在机器翻译任务中进行实验，但是这些方法并没有被限制在特定任务上。例如，可微分结构搜索方法被成功地用于学习更好的循环单元结构，这类方法完全可以应用在机器翻译任务上。

此外，受到预训练模型的启发，网络结构预搜索可能是一个极具潜力的方向。例如，有研究人员在大规模语言模型上进行网络结构搜索<sup>[844]</sup>，然后将搜索到的模型结构应用于更多的自然语言处理任务中，这种方式有效提升了模型结构的可复用性。同时，相较于使用特定任务下受限的数据，从大规模单语数据中可以更充分地学习语言的规律，进而更好地指导模型结构的设计。此外，对于机器翻译任务而言，结构的

预搜索同样是一个值得关注的研究方向。

## 15.5 小结及拓展阅读

模型结构优化一直是机器翻译研究的重要方向。一方面，对于通用框架（如注意力机制）的结构改良可以服务于多种自然语言处理任务，另一方面，针对机器翻译中存在的问题设计相适应的模型结构也是极具价值的。本章节重点介绍了神经机器翻译中几种结构优化方法，内容涉及注意力机制的改进、深层神经网络的构建、句法结构的使用以及自动结构搜索等几个方面。此外，还有若干问题值得关注：

- 多头注意力是近些年神经机器翻译中常用的结构。多头机制可以让模型从更多维度提取特征，也反应了一种多分支建模的思想。研究人员针对 Transformer 编码器的多头机制进行了分析，发现部分头在神经网络的学习过程中扮演了至关重要的角色，并且蕴含语言学解释<sup>[539]</sup>。而另一部分头本身则不具备很好的解释，对模型的帮助也不大，因此可以被剪枝掉。而且也有研究人员发现，在 Transformer 模型中并不是头数越多模型的性能就越强。如果在训练过程中使用多头机制，并在推断过程中去除大部分头，可以在模型性能不变的前提下提高模型在 CPU 上的执行效率<sup>[724]</sup>。
- 此外，也可以利用正则化手段，在训练过程中增大不同头之间的差异<sup>[870]</sup>。或引入多尺度的思想，对输入的特征进行分级表示，并引入短语的信息<sup>[871]</sup>。还可以通过对注意力权重进行调整，来区分序列中的实词与虚词<sup>[872]</sup>。除了上述基于编码器端-解码器端的建模范式，还可以定义隐变量模型来捕获句子中潜在的语言信息<sup>[764, 873]</sup>，或直接对源语言和目标语言序列进行联合表示<sup>[464]</sup>。
- 对 Transformer 等模型来说，处理超长序列是较为困难的。一种比较直接的解决办法是优化自注意力机制，降低模型计算复杂度。例如，采用了基于滑动窗口的局部注意力的 Longformer 模型<sup>[808]</sup>、基于随机特征的 Performer<sup>[727]</sup>、使用低秩分解的 Linformer<sup>[810]</sup> 和应用星型拓扑排序的 Star-Transformer<sup>[874]</sup>。



## 16. 低资源神经机器翻译

神经机器翻译带来的性能提升是显著的，但随之而来的问题是对海量双语训练数据的依赖。不同语言可使用的数据规模是不同的。比如汉语、英语这种使用范围广泛的语言，存在着大量的双语平行句对，这些语言被称为**富资源语言**（High-resource Language）。而对于其它一些使用范围稍小的语言，如斐济语、古吉拉特语等，相关的数据非常稀少，这些语言被称为**低资源语言**（Low-resource Language）。世界上现存语言超过 5000 种，仅有很少一部分为富资源语言，绝大多数均为低资源语言。即使在富资源语言中，对于一些特定的领域，双语平行语料也是十分稀缺的。有时，一些特殊的语种或者领域甚至会面临“零资源”的问题。因此，**低资源机器翻译**（Low-resource Machine Translation）是当下急需解决且颇具挑战的问题。

本章将对低资源神经机器翻译的相关问题、模型和方法展开介绍，内容涉及数据的有效使用、双向翻译模型、多语言翻译模型、无监督机器翻译、领域适应五个方面。

### 16.1 数据的有效使用

数据稀缺是低资源机器翻译所面临的主要问题，充分使用既有数据是一种解决问题的思路。比如，在双语训练不充足的时候，可以对双语数据的部分单词用近义词进行替换，达到丰富双语数据的目的<sup>[875, 876]</sup>，也可以考虑用转述等方式生成更多的双语训练数据<sup>[877, 878]</sup>。

另一种思路是使用更容易获取的单语数据。实际上，在统计机器翻译时代，使

用单语数据训练语言模型是构建机器翻译系统的关键步骤，好的语言模型往往会有带来性能的增益。而这个现象在神经机器翻译中似乎并不明显，因为在大多数神经机器翻译的范式中，并不要求使用大规模单语数据来帮助机器翻译系统。甚至，连语言模型都不会作为一个独立的模块。这一方面是由于神经机器翻译系统的解码端本身就起着语言模型的作用，另一方面是由于双语数据的增多使得翻译模型可以很好地捕捉目标语言的规律。但是，双语数据总是有限的，很多场景下，单语数据的规模会远大于双语数据，如果能够让这些单语数据发挥作用，显然是一种非常好的选择。针对以上问题，下面将从数据增强、基于语言模型的单语数据使用等方面展开讨论。

### 16.1.1 数据增强

**数据增强** (Data Augmentation) 是一种增加训练数据的方法，通常通过对既有数据进行修改或者生成新的伪数据等方式实现。有时候，数据增强也可以被看做是一种防止模型过拟合的手段<sup>[879]</sup>。在机器翻译中，典型的数据增强方法包括回译、修改双语数据、双语句对挖掘等。

#### 1. 回译

**回译** (Back Translation, BT) 是目前机器翻译任务上最常用的一种数据增强方法<sup>[604, 665, 880]</sup>。回译的主要思想是：利用目标语言-源语言翻译模型（反向翻译模型）来生成伪双语句对，用于训练源语言-目标语言翻译模型（正向翻译模型）。假设现在需要训练一个英汉翻译模型。首先，使用双语数据训练汉英翻译模型，即反向翻译模型。然后通过该模型将额外的汉语单语句子翻译为英语句子，从而得到大量的英语-真实汉语伪双语句对。然后，将回译得到的伪双语句对和真实双语句对混合，训练得到最终的英汉翻译模型。回译方法只需要训练一个反向翻译模型，就可以利用单语数据来增加训练数据的数量，因此得到了广泛使用<sup>[457, 881, 882]</sup>。图16.1 给出了回译方法的一个简要流程。

真实双语数据:  伪数据:  额外单语数据: 

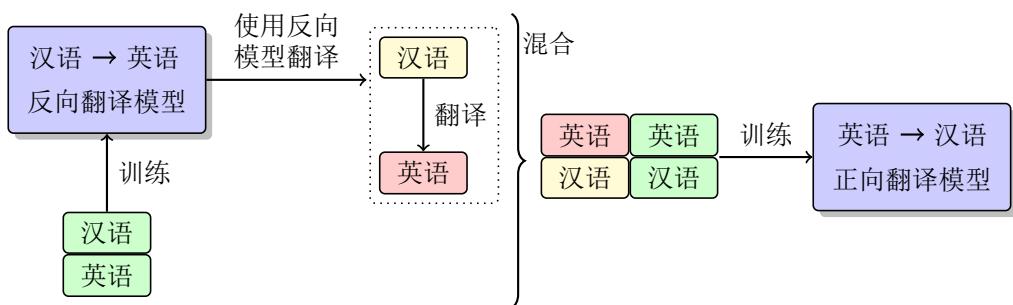


图 16.1 回译方法的简要流程

围绕如何利用回译方法生成伪双语数据这一问题，研究人员进行了详细地分析探讨。一般观点认为，反向翻译模型的性能越好，生成的伪数据质量也就越高，对正

向翻译模型的性能提升也就越大<sup>[665, 880]</sup>。不过，在实践中发现，即使一些简单的策略也能带来性能的增长。比如，对于一些低资源翻译任务，通过将目标语言句子复制到源语言端构造伪数据便能带来增益<sup>[883]</sup>。原因在于，即使构造的双语伪数据是不准确的，其目标语言端仍然是真实数据，可以使解码器训练得更加充分，进而提升神经机器翻译模型生成结果的流畅度。但是，相比这些简单的伪数据生成策略，利用目标语言单语数据进行回译可以带来更大的性能提升<sup>[883]</sup>。一种可能的解释是，双语伪数据的源语言是模型生成的翻译结果，保留了两种语言之间的互译信息，相比真实数据又存在一定的噪声。神经机器翻译模型在伪双语句对上进行训练，可以学习到如何处理带有噪声的输入，提高了模型的健壮性。

在回译方法中，反向翻译模型的训练只依赖于有限的双语数据，因此生成的源语言端伪数据的质量难以保证。为此，可以采用**迭代式回译**（Iterative Back Translation）的方法<sup>[880]</sup>，同时利用源语言端和目标语言端的单语数据，不断通过回译的方式来提升正向和反向翻译模型的性能。图16.2展示了迭代式回译的框架，图中带圈的数字代表了迭代式回译方法执行的顺序。首先，使用双语数据训练一个正向翻译模型，然后利用额外的源语言单语数据通过回译的方式生成伪双语数据，来提升反向翻译模型的性能。之后，再利用反向翻译模型和额外的目标语言单语数据生成伪双语数据，用于提升正向翻译模型的性能。可以看出，迭代式回译的过程是完全闭环的，因此可以一直重复进行，直到正向和反向翻译模型的性能均不再提升。

真实双语数据: ■ 伪数据: ■ 额外单语数据: ■  
训练: → 推断: - - →

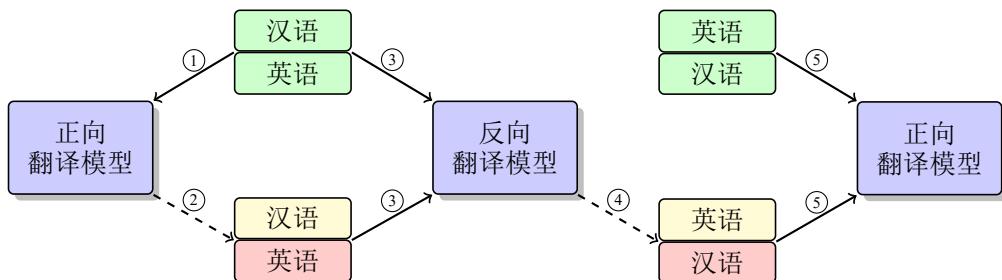


图 16.2 迭代式回译方法的流程

进一步，研究人员发现，在低资源场景中，由于缺乏双语数据，高质量的伪双语数据对于模型来说更有帮助。而在富资源场景中，在回译产生的源语言句子中添加一些噪声，提高翻译结果的多样性，反而可以达到更好的效果，比较常用的方法是使用采样解码、Top- $k$  解码和加噪<sup>[604, 884, 885]</sup>。回译中常用的解码方式为束搜索，在生成每个词的时候只考虑预测概率最高的几个词，因此生成的翻译结果质量更高，但导致的问题是翻译结果主要集中在部分高频词上，生成的伪数据缺乏多样性，也就很难去准确地覆盖真实的数据分布<sup>[886]</sup>。采样解码是指在解码过程中，对词表中所有的词按照预测概率进行随机采样，因此整个词表中的词都有可能被选中，从而使生成

结果多样性更强，但翻译质量和流畅度也会明显下降。Top- $k$  解码是对束搜索和采样解码的一个折中方法。在解码过程中，Top- $k$  解码对词表中预测概率最高的前  $k$  个词进行随机采样，这样在保证翻译结果准确的前提下，提高了结果的多样性。加噪方法在束搜索的解码结果加入一些噪声，如丢掉或屏蔽部分词、打乱句子顺序等。这些方法在生成的源语言句子中引入了噪声，不仅增加了对包含低频词或噪声句子的训练次数，同时也提高了模型的健壮性和泛化能力<sup>[595]</sup>。

与回译方法类似，源语言单语数据也可以通过一个双语数据训练的正向翻译模型获得对应的目标语言翻译结果，从而构造正向翻译的伪数据<sup>[887]</sup>。与回译方法相反，这时的伪数据中源语言句子是真实的，而目标语言句子是自动生成的，构造的伪数据对译文的流畅性并没有太大帮助，其主要作用是提升编码器的特征提取能力。然而，由于伪数据中生成的译文质量很难保证，因此利用正向翻译模型生成伪数据的方法带来的性能提升效果要弱于回译，甚至可能是有害的<sup>[885]</sup>。

## 2. 修改双语数据

回译方法是利用单语数据来生成伪数据，而另外一种数据增强技术是对原始双语数据进行修改来得到伪双语数据，常用的方法包括加噪和转述等。

加噪是自然语言处理任务中广泛使用的一种方法<sup>[123, 595, 881, 888]</sup>。比如，在广泛使用的降噪自编码器（Denoising Autoencoder）中，向原始数据中加入噪声作为模型的输入，模型通过学习如何预测原始数据进行训练。而在神经机器翻译中，利用加噪方法进行数据增强的常用方法是，在保证句子整体语义不变的情况下，对原始的双语数据适当加入一些噪声，从而生成伪双语数据来增加训练数据的规模。常用的加噪方法主要有以下三种：

- **丢掉单词**: 句子中的每个词均有  $P_{Drop}$  的概率被丢弃。
- **掩码单词**: 句子中的每个词均有  $P_{Mask}$  的概率被替换为一个额外的 <Mask> 词。<Mask> 的作用类似于占位符，可以理解为一个句子中的部分词被屏蔽掉，无法得知该位置词的准确含义。
- **打乱顺序**: 将句子中距离较近的某些词的位置进行随机交换。

图16.3展示了三种加噪方法的示例。这里， $P_{Drop}$  和  $P_{Mask}$  均设置为 0.1，表示每个词有 10% 的概率被丢弃或掩码。打乱句子内部顺序的操作略微复杂，一种实现方法是：通过一个数字来表示每个词在句子中的位置，如“我”是第一个词，“你”是第三个词，然后，在每个位置生成一个 1 到  $n$  的随机数， $n$  一般设置为 3，然后将每个词的位置数和对应的随机数相加，即图中的  $S$ 。对  $S$  按照从小到大排序，根据排序后每个位置的索引从原始句子中选择对应的词，从而得到最终打乱顺序后的结果。比如，在计算后，除了  $S_2$  的值小于  $S_1$  外，其余单词的  $S$  值均为递增顺序，则将原句中第一个词和第二个词进行交换，其他词保持不变。

和回译方法相似，加噪方法一般仅在源语言句子上进行操作，既保证了目标语

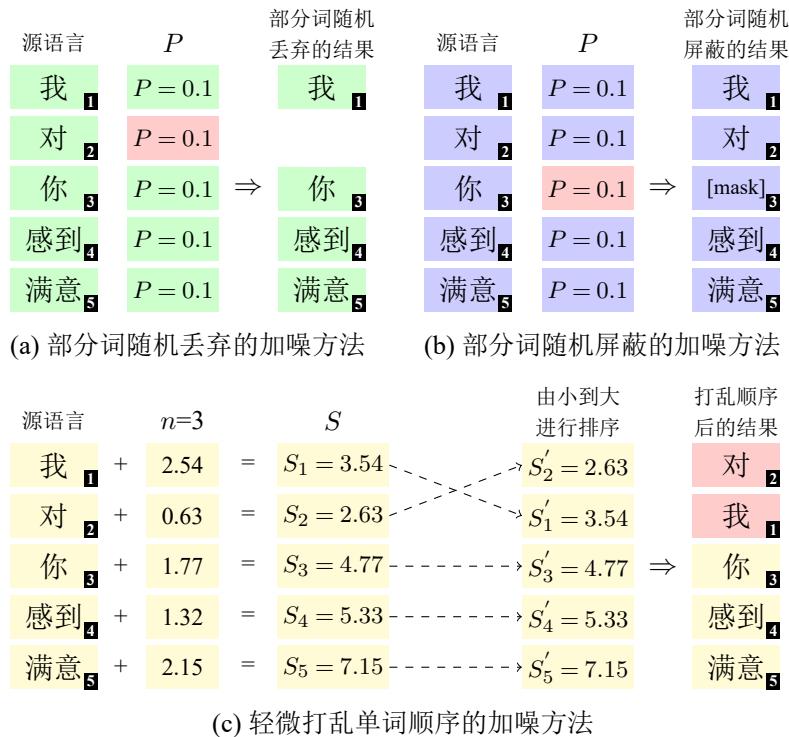


图 16.3 三种数据加噪方法

言句子的流畅度，又可以增加数据的多样性，提高模型的健壮性和泛化能力<sup>[595]</sup>。加噪作为一种简单有效的方法，实际的应用场景很多，比如：

- **对单语数据加噪。**通过一个端到端模型预测源语言句子的调序结果，该模型和神经机器翻译模型的编码器共享参数，从而增强编码器的特征提取能力<sup>[887]</sup>；
- **训练降噪自编码器。**将加噪后的句子作为输入，原始句子作为输出，用来训练降噪自编码器，这一思想在无监督机器翻译中得到了广泛应用，详细方法可以参考16.4.3节；
- **对伪数据进行加噪。**比如在上文中提到的对伪数据加入噪声的方法中，通常也使用上述这三种加噪方法来提高伪数据的多样性；

另外一种加噪方法是进行词替换：将双语数据中的部分词替换为词表中的其他词，在保证句子的语义或语法正确性的前提下，增加了训练数据的多样性。比如，对于“我/出去/玩。”这句话，将“我”替换为“你”、“他”、“我们”。或者，将“玩”替换为“骑车”、“学习”、“吃饭”等，虽然改变了语义，但句子在语法上仍然是合理的。

词替换的另一种策略是将源语言中的稀有词替换为语义相近的词<sup>[875]</sup>。词表中的稀有词由于出现次数较少，很容易导致训练不充分问题<sup>[89]</sup>。通过语言模型将源语言

句子中的某个词替换为满足语法或语义条件的稀有词，再通过词对齐工具找到源语言句子中被替换的词在目标语言句子中对应的位置，借助翻译词典将这个目标语言位置的单词替换为词典中的翻译结果，从而得到伪双语数据。

此外，通过在源语言或目标语言中随机选择某些词，将这些词替换为词表中一个随机词，也可以得到伪双语数据<sup>[876]</sup>。随机选择句子中的某个词，将这个词的词嵌入替换为其他词的词嵌入的加权结果。相比直接替换单词，这种丰富的分布式表示相比直接使用词嵌入可以包含更多的语义信息，同一个词在不同的上下文中也会被替换为不同的上下文表示结果<sup>[590]</sup>。

相比上述两种方法只是对句子做轻微的修改，转述（Paraphrasing）方法考虑到了自然语言表达的多样性：通过对原始句子进行改写，使用不同的句式来传达相同含义的信息<sup>[889, 890]</sup>。比如对于“东北大学的校训是自强不息、知行合一”这句话，可以使用其他的句式来表达同样的含义，例如：“自强不息、知行合一是东北大学的校训”。转述在机器翻译任务上得到了广泛引用<sup>[878, 891, 892]</sup>，通过转述方法对原始的双语数据进行改写，训练数据可以覆盖更多的语言学现象。同时由于每个句子可以对应多个不同的翻译，转述方法可以避免模型过拟合，提高模型的泛化能力。

### 3. 双语句对挖掘

在双语平行语料缺乏的时候，从可比语料中挖掘可用的双语句对也是一种有效的方法<sup>[893, 894, 895]</sup>。可比语料是指源语言和目标语言虽然不是完全互译的文本，但是蕴含了丰富的双语对照知识，可以从中挖掘出可用的双语句对来训练。相比双语平行语料来说，可比语料相对容易获取，比如，多种语言报道的新闻事件、多种语言的维基百科词条和多种语言翻译的书籍等。如图16.4中的维基百科词条所示。

#### WIKIPEDIA

**Machine Translation**, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation interactive translation), is a subfield of computational linguistics that investigates the use of software to translate text or speech from one language to another.

#### 维基百科

**机器翻译**（英语：Machine Translation，经常简写为 MT，简称机译或机翻）属于计算语言学的范畴，其研究借由计算机程序将文字或演说从一种自然语言翻译成另一种自然语言。

图 16.4 维基百科中的可比语料

可比语料大多存在于网页中，内容较为复杂，可能会存在较大比例的噪声，如HTML字符、乱码等。首先需要对内容进行充分的数据清洗，得到干净的可比语料，然

后从中抽取出可用的双语句对。传统的抽取方法一般通过统计模型或双语词典来得到双语句对。比如，通过计算两个不同语言句子之间的单词重叠数或 BLEU 值<sup>[893, 896]</sup>；或者通过排序模型或二分类器判断一个目标语言句子和一个源语言句子互译的可能性<sup>[894, 897]</sup>。

另外一种比较有效的方法是根据两种语言中每个句子的表示向量来抽取数据<sup>[895]</sup>。首先，对于两种语言的每个句子，分别使用词嵌入加权平均等方法计算得到句子的表示向量，然后计算每个源语言句子和目标语言句子之间的余弦相似度，相似度大于一定阈值的句对则认为是可用的双语句对<sup>[895]</sup>。然而，不同语言单独训练得到的词嵌入可能对应不同的表示空间，因此得到的表示向量无法用于衡量两个句子的相似度<sup>[898]</sup>。为了解决这个问题，一般使用在同一表示空间的跨语言词嵌入来表示两种语言的单词<sup>[899]</sup>。在跨语言词嵌入中，不同语言相同意思的词对应的词嵌入具有较高的相似性，因此得到的句子表示向量也就可以用于衡量两个句子是否表示相似的语义<sup>[163]</sup>。关于跨语言词嵌入的具体内容，可以参考16.4.1节的内容。

## 16.1.2 基于语言模型的方法

除了构造双语数据进行数据增强，直接利用单语数据也是机器翻译中的常用方法。通常，单语数据会被用于训练语言模型（见第二章）。对于机器翻译系统，使用语言模型也是一件十分自然的事情，在目标语言端，语言模型可以帮助系统选择更加流畅的译文；在源语言端，语言模型也可以用于句子编码，进而更好地生成句子的表示结果。在传统方法中，语言模型更多地被使用在目标语言端。不过，近些年来随着预训练技术的发展，语言模型也被使用在神经机器翻译的编码器端。下面将从语言模型在解码器端的融合、预训练词嵌入、预训练编码器和多任务学习四方面介绍基于语言模型的单语数据使用方法。

### 1. 语言模型在目标语言端的融合

融合目标语言端的语言模型是一种最直接的使用单语数据的方法<sup>[900, 901, 902]</sup>。实际上，神经机器翻译模型本身也具备了语言模型的作用，因为解码器本质上也是一个语言模型，用于描述生成译文词串的规律。对于一个双语句对  $(x, y)$ ，神经机器翻译模型可以被描述为：

$$\log P(y|x; \theta) = \sum_t \log P(y_j|y_{<j}, x; \theta) \quad (16.1)$$

这里， $\theta$  是神经机器翻译模型的参数， $y_{<j}$  表示第  $j$  个位置前面已经生成的词序列。可以看出，模型的翻译过程与两部分信息有关，分别是源语言句子  $x$  以及前面生成的译文序列  $y_{<j}$ 。语言模型可以与解码过程融合，根据  $y_{<j}$  生成流畅度更高的翻译结果。**常用的融合方法主要分为浅融合和深融合<sup>[900]</sup>**。

浅融合方法独立训练翻译模型和语言模型，在生成每个词的时候，对两个模型的预测概率进行加权求和得到最终的预测概率。浅融合的不足在于，解码过程对每个词均采用相同的语言模型权重，缺乏灵活性。针对这个问题，深融合联合翻译模型和语言模型进行训练，从而在解码过程中动态地计算语言模型的权重，更好地融合翻译模型和语言模型来计算预测概率。

大多数情况下，目标语言端语言模型的使用可以提高译文的流畅度。不过，它并不会增加翻译结果对源语言句子表达的充分性，即源语言句子的信息是否被充分体现到了译文中。也有一些研究发现，神经机器翻译过于关注译文的流畅度，但是充分性的问题没有得到很好考虑，比如，神经机器翻译系统的结果中经常出现漏译等问题。也有一些研究人员提出控制翻译充分性的方法，让译文在流畅度和充分性之间达到平衡<sup>[472, 473, 903]</sup>。

## 2. 预训练词嵌入

神经机器翻译模型所使用的编码器-解码器框架天然就包含了对输入（源语言）和输出（目标语言）进行表示学习的过程。在编码端，需要学习一种分布式表示来表示源语言句子的信息，这种分布式表示可以包含序列中每个位置的表示结果（见第九章）。从结构上看，神经机器翻译所使用的编码器与语言模型无异，或者说神经机器翻译的编码器其实就是一个源语言的语言模型。唯一的区别在于，神经机器翻译的编码器并不直接输出源语言句子的生成概率，而传统语言模型是建立在序列生成任务上的。既然神经机器翻译的编码器可以与解码器一起在双语数据上联合训练，那为什么不使用更大规模的数据单独对编码器进行训练呢？或者说，直接使用一个预先训练好的编码器，与机器翻译的解码器配合完成翻译过程。

实现上述想法的一种手段是预训练（Pre-training）<sup>[123, 124, 165, 904]</sup>。预训练的做法相当于将句子的表示学习任务从目标任务中分离出来，这样可以利用额外的更大规模的数据进行学习。常用的一种方法是使用语言建模等方式在大规模单语数据上进行训练，来得到神经机器翻译模型中的一部分（比如词嵌入和编码器等）的模型参数初始值。然后，神经机器翻译模型在双语数据上进行微调（Fine-tuning），以得到最终的翻译模型。

词嵌入可以被看作是对每个独立单词进行的表示学习的结果，在自然语言处理的众多任务中都扮演着重要角色<sup>[100, 905, 906]</sup>。到目前为止已经有大量的词嵌入学习方法被提出（见第九章），因此可以直接应用这些方法在海量的单语数据上训练得到词嵌入，用来初始化神经机器翻译模型的词嵌入参数矩阵<sup>[907, 908]</sup>。

需要注意的是，在神经机器翻译中使用预训练词嵌入有两种方法。一种方法是直接将词嵌入作为固定的输入，也就是在训练神经机器翻译模型的过程中，并不调整词嵌入的参数。这样做的目的是完全将词嵌入模块独立出来，机器翻译可以被看作是在固定的词嵌入输入上进行的建模，从而降低了机器翻译模型学习的难度。另一种方法是仍然遵循“预训练 + 微调”的策略，将词嵌入作为机器翻译模型部分参数

的初始值。在之后机器翻译训练过程中，词嵌入模型结果会被进一步更新。近些年，在词嵌入预训练的基础上进行微调的方法越来越受到研究者的青睐。因为在实践中发现，完全用单语数据学习的单词表示，与双语数据上的翻译任务并不完全匹配。同时目标语言的信息也会影响源语言的表示学习。

虽然预训练词嵌入在海量的单语数据上学习到了丰富的表示，但词嵌入一个主要的缺点是无法解决一词多义问题。在不同的上下文中，同一个单词经常表示不同的意思，但它的词嵌入是完全相同的，模型需要在编码过程中通过上下文去理解每个词在当前语境下的含义。因此，上下文词向量在近些年得到了广泛的关注<sup>[165, 448, 909]</sup>。上下文词嵌入是指一个词的表示不仅依赖于单词自身，还依赖于上下文语境。由于在不同的上下文中，每个词对应的词嵌入是不同的，因此无法简单地通过词嵌入矩阵来表示。通常的做法是使用海量的单语数据预训练语言模型任务，以期望句子中每个位置对应的表示结果包含了一定的上下文信息<sup>[123, 124, 165]</sup>。这本质上和下面即将介绍的句子级预训练模型是一样。

### 3. 预训练模型

相比固定的词嵌入，上下文词嵌入包含了在当前语境中的语义信息，丰富了模型的输入表示，降低了训练难度。但是，模型仍有大量的参数需要从零学习，来进一步提取整个句子的表示。一种可行的方案是在预训练阶段中直接得到预训练好的模型参数，在下游任务中仅仅通过任务特定的数据对模型参数进行微调，来得到一个较强的模型。基于这个想法，有大量的预训练模型被提出。比如，**生成式预训练** (Generative Pre-training, GPT) 和**来自 Transformer 的双向编码器表示** (Bidirectional Encoder Representations From Transformers, BERT) 就是两种典型的预训练模型。图16.5对比了二者的模型结构。

TRM：标准 Transformer 模块

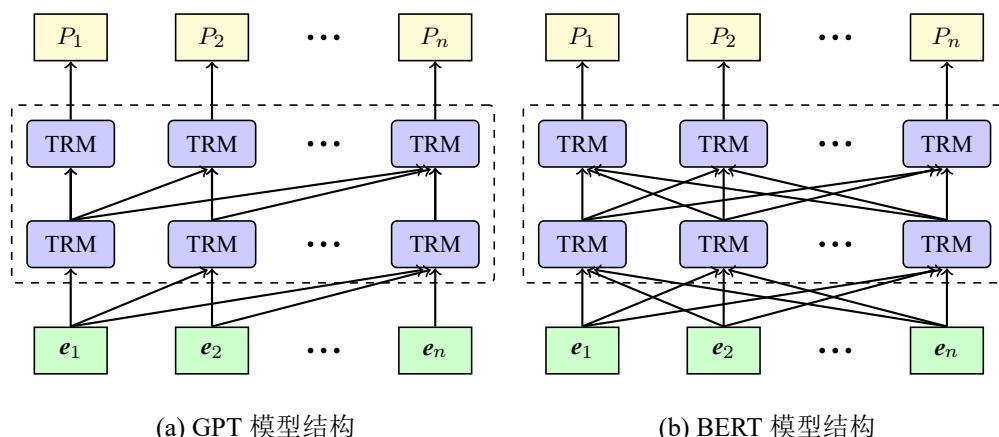


图 16.5 GPT 模型结构和 BERT 模型结构对比示意图

GPT 通过 Transformer 模型自回归地训练单向语言模型<sup>[124]</sup>，类似于神经机器翻译模型的解码器，相比双向 LSTM 等模型，Tranformer 模型的表示能力更强。之后提出的 BERT 模型更是将预训练的作用提升到了新的水平<sup>[123]</sup>。GPT 模型的一个缺陷在于模型只能进行单向编码，也就是前面的文本在建模时无法获取到后面的信息。而 BERT 提出了一种自编码的方式，使模型在预训练阶段可以通过双向编码的方式进行建模，进一步增强了模型的表示能力。

BERT 的核心思想是通过**掩码语言模型**（Masked Language Model, MLM）任务进行预训练。掩码语言模型的思想类似于完形填空，随机选择输入句子中的部分词进行掩码，之后让模型预测这些被掩码的词。掩码的具体做法是将被选中的词替换为一个特殊的词 <Mask>，这样模型在训练过程中，无法得到掩码位置词的信息，需要联合上下文内容进行预测，因此提高了模型对上下文的特征提取能力。而使用掩码的方式进行训练也给神经机器翻译提供了新的思路，在本章的其它部分中也会使用到类似方法。

在神经机器翻译任务中，**预训练模型可以用于初始化编码器的模型参数**<sup>[910, 911, 912]</sup>。之所以用在编码器端而不是解码器端，主要原因是编码器的作用主要在于特征提取，训练难度相对较高，而解码器的作用主要在于生成，和编码器提取到的表示是强依赖的，相对比较脆弱<sup>[913]</sup>。

然而，在实践中发现，**参数初始化的方法在一些富资源语种上提升效果并不明显，甚至会带来性能的下降**<sup>[914]</sup>。原因可能在于，**预训练阶段的训练数据规模是非常大的，因此在下游任务数据量较少的情况下帮助较大。而在一些富资源语种上，双语句对的数据足够充分，因此简单通过预训练模型来初始化模型参数无法带来明显的提升**。此外，**预训练模型的训练目标并没有考虑到序列到序列的生成，与神经机器翻译的训练目标并不完全一致，两者训练得到的模型参数可能存在一些区别**。

因此，一种做法将预训练模型和翻译模型进行融合，把预训练模型作为一个独立的模块来为编码器或者解码器提供句子级表示结果<sup>[914, 915]</sup>。另外一种做法是针对生成任务进行预训练。**机器翻译是一种典型的语言生成任务，不仅包含源语言表示学习的问题，还有序列到序列的映射，以及目标语言端序列生成的问题，这些知识是无法单独通过（源语言）单语数据学习到的。因此，可以使用单语数据对编码器-解码器结构进行预训练**<sup>[916, 917, 918]</sup>。

以**掩码端到端预训练**（Masked Sequence to Sequence Pre-training, MASS）方法为例<sup>[916]</sup>，其思想与 BERT 十分相似，也是在预训练过程中采用掩码的方式，随机选择编码器输入句子中的连续片段替换为特殊词 <Mask>，然后在解码器预测这个连续片段，如图16.6 所示。这种做法可以使得编码器捕捉上下文信息，同时迫使解码器依赖于编码器进行自回归的生成，从而学习到编码器和解码器之间的注意力。为了适配下游的机器翻译任务，使预训练模型可以学习到不同语言的表示，MASS 对不同语言的句子采用共享词汇表和模型参数的方法，利用同一个预训练模型来进行不同语言句子的预训练。通过这种方式，模型既学到了对源语言句子的编码，也学习到了

对目标语言句子的生成方法，之后通过使用双语句对来对预训练模型进行微调，模型可以快速收敛到较好的状态。

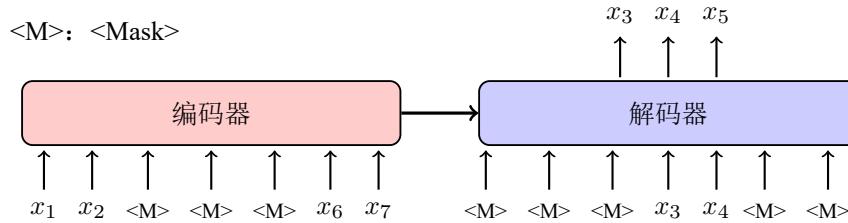


图 16.6 MASS 预训练方法

此外，还有很多问题值得探讨。例如，为何预训练词嵌入在神经机器翻译模型中有效<sup>[908]</sup>；如何在神经机器翻译模型中利用预训练的 BERT 模型<sup>[910, 911, 915, 919]</sup>；如何针对神经机器翻译任务进行预训练<sup>[917, 920, 921]</sup>；如何针对机器翻译中的 Code-switching 问题进行预训练<sup>[922]</sup>；如何在微调过程中避免灾难性遗忘<sup>[923]</sup>。

#### 4. 多任务学习

在训练一个神经网络的时候，如果过分地关注单个训练目标，可能使模型忽略掉其他可能有帮助的信息，这些信息可能来自于一些其他相关的任务<sup>[924]</sup>。通过联合多个独立但相关的任务共同学习，任务之间相互“促进”，就是多任务学习<sup>[924, 925, 926]</sup>。多任务学习的常用做法是，针对多个相关的任务，共享模型的部分参数来学习不同任务之间相似的特征，并通过特定的模块来学习每个任务独立的特征（见第十五章）。常用的策略是对底层的模型参数进行共享，顶层的模型参数用于独立学习各个不同的任务。

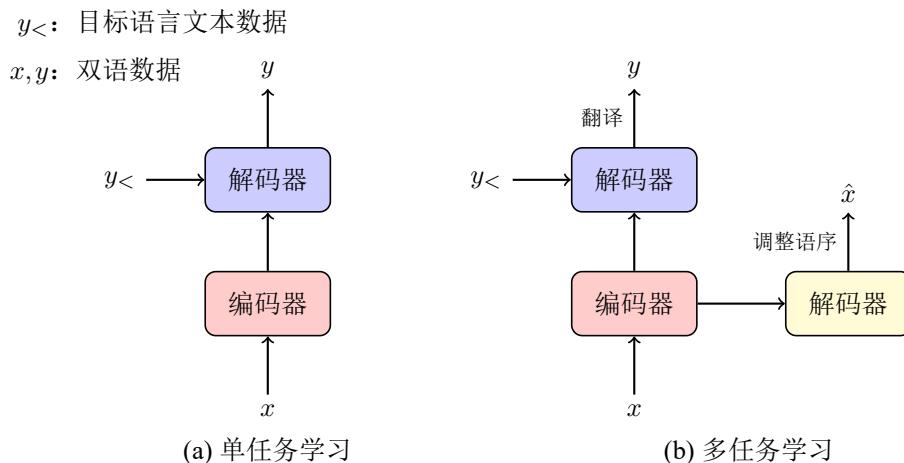


图 16.7 使用源语言单语数据的多任务学习

在神经机器翻译中，应用多任务学习的主要策略是将翻译任务作为主任务，同

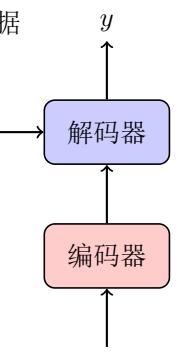
时设置一些仅使用单语数据的子任务，通过这些子任务来捕捉单语数据中的语言知识<sup>[834, 887, 927]</sup>。一种多任务学习的方法是利用源语言单语数据，通过单个编码器对源语言数据进行建模，再分别使用两个解码器来学习源语言排序和翻译任务。源语言排序任务是指利用预排序规则对源语言句子中词的顺序进行调整<sup>[303]</sup>，可以通过单语数据来构造训练数据，从而使编码器被训练得更加充分<sup>[887]</sup>，如图16.7所示，图中  $y_{<}$  表示当前时刻之前的单词序列， $x_{<}$  表示源语言句子中词的顺序调整后的句子。

虽然神经机器翻译模型可以看作一种语言生成模型，但生成过程中却依赖于源语言信息，因此无法直接利用目标语言单语数据进行多任务学习。针对这个问题，可以对原有翻译模型结构进行修改，在解码器底层增加一个语言模型子层，这个子层用于学习语言模型任务，与编码器端是完全独立的，如图16.8所示<sup>[927]</sup>，图中  $y_{<}$  表示当前时刻之前的单词序列， $z_{<}$  表示当前时刻之前的单语数据。在训练过程中，分别将双语数据和单语数据送入翻译模型和语言模型进行计算，双语数据训练产生的梯度用于对整个模型进行参数更新，而单语数据产生的梯度只对语言模型子层进行参数更新。

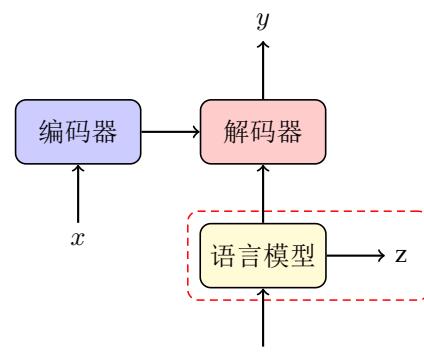
$y_{<}$ : 目标语言文本数据

$x, y$ : 双语数据

$z$ : 单语数据



(a) 单任务学习



(b) 多任务学习

图 16.8 使用语言模型的多任务学习

此外，也可以利用多任务学习的思想来训练多到一模型（多个编码器、单个解码器）、一到多模型（单个编码器、多个解码器）和多到多模型（多个编码器、多个解码器），从而借助单语数据或其他数据来使编码器或解码器训练得更加充分<sup>[834]</sup>，任务的形式包括翻译任务、句法分析任务、图像分类等。另外一种策略是利用多任务学习的思想同时训练多个语言的翻译任务<sup>[928, 929]</sup>，同样包括多到一翻译（多个语种到一个语种）、一到多翻译（一个语种到多个语种）以及多到多翻译（多个语种到多个语种），这种方法可以利用多种语言的训练数据进行学习，具有较大的潜力，逐渐受到了研究人员们的关注，具体内容可以参考16.3节。

## 16.2 双向翻译模型

在机器翻译任务中，对于给定的双语数据，可以同时学习源语言到目标语言和目标语言到源语言的翻译模型，因此机器翻译可被视为一种双向任务。那么，两个方向的翻译模型能否联合起来，相辅相成呢？下面将从双向训练和对偶学习两方面对双向翻译模型进行介绍。这些方法被大量使用在低资源翻译系统中，比如，可以用双向翻译模型反复迭代构造伪数据。

### 16.2.1 双向训练

回顾神经机器翻译系统的建模过程，给定一个互译的句对  $(x, y)$ ，一个从源语言句子  $x$  到目标语言句子  $y$  的翻译被表示为求条件概率  $P(y|x)$  的问题。类似地，一个从目标语言句子  $y$  到源语言句子  $x$  的翻译可以表示为  $P(x|y)$ 。通常来说，神经机器翻译的训练一次只得到一个方向的模型，也就是  $P(y|x)$  或者  $P(x|y)$ 。这意味着  $P(y|x)$  和  $P(x|y)$  之间是互相独立的。但  $P(y|x)$  和  $P(x|y)$  是否真的没有关系呢？这里以最简单的情况为例，假设  $x$  和  $y$  被表示为相同大小的两个向量  $\mathbf{E}_x$  和  $\mathbf{E}_y$ ，且  $\mathbf{E}_x$  到  $\mathbf{E}_y$  的变换是一个线性变换，也就是与一个方阵  $\mathbf{W}$  做矩阵乘法：

$$\mathbf{E}_y = \mathbf{E}_x \cdot \mathbf{W} \quad (16.2)$$

这里， $\mathbf{W}$  应当是一个满秩矩阵，否则对于任意一个  $\mathbf{E}_x$  经过  $\mathbf{W}$  变换得到的  $\mathbf{E}_y$  只落在所有可能的  $\mathbf{E}_y$  的一个子空间内，即在给定  $\mathbf{W}$  的情况下有些  $y$  不能被任何一个  $x$  表达，而这不符合常识，因为不管是什么句子，总能找到它的一种译文。若  $\mathbf{W}$  是满秩矩阵说明  $\mathbf{W}$  可逆，也就是给定  $\mathbf{E}_x$  到  $\mathbf{E}_y$  的变换  $\mathbf{W}$  下， $\mathbf{E}_y$  到  $\mathbf{E}_x$  的变换必然是  $\mathbf{W}$  的逆而不是其他矩阵。

这个例子说明  $P(y|x)$  和  $P(x|y)$  直觉上应当存在联系。当然， $x$  和  $y$  之间是否存在简单的线性变换关系并没有结论，但是上面的例子给出了一种对源语言句子和目标语言句子进行相互转化的思路。实际上，研究人员已经通过一些数学技巧用目标函数把  $P(y|x)$  和  $P(x|y)$  联系起来，这样训练神经机器翻译系统一次就可以同时得到两个方向的翻译模型，使得训练变得更加高效<sup>[457, 930, 931]</sup>。双向联合训练的基本思想是：使用两个方向的翻译模型对单语数据进行推断，之后把翻译结果与原始的单语数据作为训练语料，通过多次迭代更新两个方向上的机器翻译模型。

图16.9给出了一个双向训练的流程，其中  $M_{x \rightarrow y}^k$  表示第  $k$  轮得到的  $x$  到  $y$  的翻译模型， $M_{y \rightarrow x}^k$  表示第  $k$  轮得到的  $y$  到  $x$  的翻译模型。这里只展示了前两轮迭代。在第一次迭代开始之前，首先使用双语数据对两个初始翻译模型进行训练。为了保持一致性，这里称之为第 0 轮迭代。在第一轮迭代中，首先使用这两个翻译模型  $M_{x \rightarrow y}^0$  和  $M_{y \rightarrow x}^0$  翻译单语数据  $X = \{x_i\}$  和  $Y = \{y_i\}$  后得到译文  $\{\hat{y}_i^0\}$  和  $\{\hat{x}_i^0\}$ 。进一步，构建伪训练数据集  $\{x_i, \hat{y}_i^0\}$  与  $\{\hat{x}_i^0, y_i\}$ 。然后使用上面的两个伪训练数据集和原始双语数据混合，训练得到模型  $M_{x \rightarrow y}^1$  和  $M_{y \rightarrow x}^1$  并进行参数更新，即用  $\{\hat{x}_i^0, y_i\} \cup \{x_i, y_i\}$  训

练  $M_{x \rightarrow y}^1$ , 用  $\{\hat{y}_i^0, x_i\} \cup \{y_i, x_i\}$  训练  $M_{y \rightarrow x}^1$ 。第二轮迭代继续重复上述过程, 使用更新参数后的翻译模型  $M_{x \rightarrow y}^1$  和  $M_{y \rightarrow x}^1$  得到新的伪数据集  $\{x_i, \hat{y}_i^1\}$  与  $\{\hat{x}_i^1, y_i\}$ 。然后, 进一步得到翻译模型  $M_{x \rightarrow y}^2$  和  $M_{y \rightarrow x}^2$ 。这种方式本质上也是一种自学习的过程, 逐步生成更好的伪数据, 同时提升模型质量。

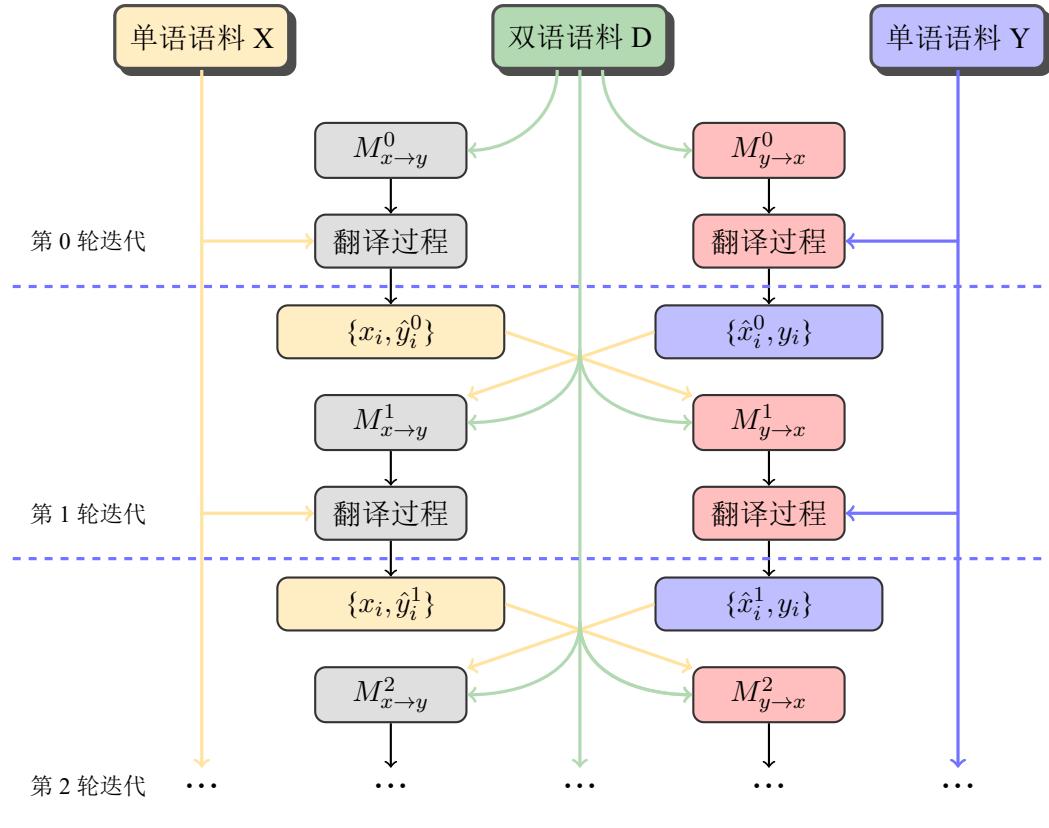


图 16.9 翻译模型的双向训练

### 16.2.2 对偶学习

对称, 也许是人类最喜欢的美, 其始终贯穿在整个人类文明的诞生与发展之中。古语“夫美者, 上下、内外、大小、远近皆无害焉, 故曰美”描述的即是这样的美。在人工智能的任务中, 也存在着这样的对称结构, 比如机器翻译中英译汉和汉译英、图像处理中的图像标注和图像生成, 以及语音处理中的语音识别和语音合成等。利用这些任务的对称性质(也称对偶性), 可以使互为对偶的两个任务获得更有效的反馈, 从而使对应的模型相互学习、相互提高。

目前, 对偶学习的思想已经广泛应用于低资源机器翻译领域, 它不仅能够提升在有限双语资源下的翻译模型性能, 而且能够利用未标注的单语数据来进行学习。下面将针对**有监督对偶学习**(Dual Supervised Learning)<sup>[932, 933]</sup>与**无监督对偶学习**(Dual Unsupervised Learning)<sup>[934, 935, 936]</sup>两方面, 对对偶学习的思想进行介绍。

## 1. 有监督对偶学习

对偶学习涉及两个任务，分别是原始任务和它的对偶任务。在机器翻译任务中，给定一个互译的句对  $(x, y)$ ，原始任务学习一个条件概率  $P(y|x)$  将源语言句子  $x$  翻译成目标语言句子  $y$ ；对偶任务同样学习一个条件概率  $P(x|y)$  将目标语言句子  $y$  翻译成源语言句子  $x$ 。除了使用条件概率建模翻译问题，还可以使用联合分布  $P(x, y)$  进行建模。根据条件概率定义，有：

$$\begin{aligned} P(x, y) &= P(x)P(y|x) \\ &= P(y)P(x|y) \end{aligned} \quad (16.3)$$

公式(16.3)很自然地把两个方向的翻译模型  $P(y|x)$  和  $P(x|y)$  以及两个语言模型  $P(x)$  和  $P(y)$  联系起来： $P(x)P(y|x)$  应该与  $P(y)P(x|y)$  接近，因为它们都表达了同一个联合分布  $P(x, y)$ 。因此，在构建训练两个方向的翻译模型的目标函数时，除了它们单独训练时各自使用的极大似然估计目标函数，可以额外增加一个目标项来鼓励两个方向的翻译模型，例如：

$$L_{\text{dual}} = \left( \log P(x) + \log P(y|x) - \log P(y) - \log P(x|y) \right)^2 \quad (16.4)$$

通过该正则化项，互为对偶的两个任务可以被放在一起学习，通过任务对偶性加强监督学习的过程，就是有监督对偶学习<sup>[932, 934]</sup>。这里， $P(x)$  和  $P(y)$  两个语言模型是预先训练好的，并不参与翻译模型的训练。可以看到，对于单独的一个模型来说，其目标函数增加了与另外一个方向的模型相关的损失项。这样的形式与 L1/L2 正则化非常类似（见第十三章），因此可以把这个方法看作是一种正则化的手段（由翻译任务本身的性质所启发而来）。有监督对偶学习实际上要优化如下的损失函数：

$$L = \log P(y|x) + \log P(x|y) + L_{\text{dual}} \quad (16.5)$$

由于两个方向的翻译模型和语言模型相互影响，这种共同训练、共同提高的方法能得到比基于单个方向训练效果更好的模型。

## 2. 无监督对偶学习

有监督的对偶学习需要使用双语数据来训练两个翻译模型，但是有些低资源语言仅有少量双语数据可以训练。因此，如何使用资源相对丰富的单语数据来提升翻译模型的性能也是一个关键问题。

无监督对偶学习提供了一个解决问题的思路<sup>[934]</sup>。假设目前有两个比较弱的翻译模型，一个原始翻译模型  $f$  将源语言句子  $x$  翻译成目标语言句子  $y$ ，一个对偶任务模型  $g$  将目标语言句子  $y$  翻译成源语言句子  $x$ 。翻译模型可由有限的双语训练，或者使用无监督机器翻译得到（见16.4节）。如图16.10所示，无监督对偶学习的做法是，先

通过原始任务模型  $f$  将一个源语言单语句子  $x$  翻译为目标语言句子  $y$ ，随后，再通过对偶任务模型  $g$  将目标语言句子  $y$  翻译为源语言句子  $x'$ 。如果模型  $f$  和  $g$  的翻译性能较好，那么  $x'$  和  $x$  会十分相似。通过计算二者的**重构损失**（Reconstruction Loss），就可以优化模型  $f$  和  $g$  的参数。这个过程可以多次迭代，从大量的无标注单语数据上不断提升性能。

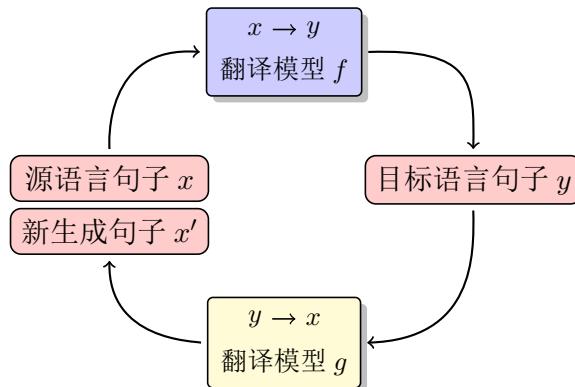


图 16.10 无监督对偶学习流程

这个过程与强化学习的流程非常相似（见第十三章）。在训练过程中，模型无法知道某个状态下正确的行为是什么，只能通过这种试错-反馈的机制来反复调整。训练这两个模型可以用已有的强化学习算法来训练，比如策略梯度方法<sup>[937]</sup>。策略梯度的基本思想如下：如果在执行某个动作之后，获得了一个不错的反馈，那么会调整策略来增加这个状态下执行该动作的概率；反之，如果采取某个动作后获得了一个负反馈，就需要调整策略来降低这个状态下执行该动作的概率。

## 16.3 多语言翻译模型

低资源机器翻译面临的主要挑战是缺乏大规模高质量的双语数据。这个问题往往伴随着多语言的翻译任务<sup>[938]</sup>。也就是，要同时开发多个不同语言之间的机器翻译系统，其中少部分语言是富资源语言，而其它语言是低资源语言。针对低资源语言双语数据稀少或者缺失的情况，一种常见的思路是利用富资源语言的数据或者系统帮助低资源机器翻译系统。这也构成了多语言翻译的思想，并延伸出大量的研究工作，其中有三个典型研究方向：基于枢轴语言的方法<sup>[939]</sup>、基于知识蒸馏的方法<sup>[549]</sup>、基于迁移学习的方法<sup>[929, 940]</sup>，下面进行介绍。

### 16.3.1 基于枢轴语言的方法

传统的多语言翻译中，广泛使用的是**基于枢轴语言的翻译**（Pivot-based Translation）<sup>[939, 940]</sup>。这种方法会使用一种数据丰富的语言作为**枢轴语言**（Pivot Language）。翻译过程分为两个阶段：源语言到枢轴语言的翻译，枢轴语言到目标语言的翻译。这样，

通过资源丰富的枢轴语言将源语言和目标语言桥接在一起，达到解决源语言-目标语言双语数据缺乏的问题。比如，想要得到泰语到波兰语的翻译，可以通过英语做枢轴语言。通过“泰语 → 英语 → 波兰语”的翻译过程完成泰语到波兰语的转换。

在统计机器翻译中，有很多基于枢轴语言的方法<sup>[941, 942, 943, 944]</sup>，这些方法也已经广泛用于低资源翻译任务<sup>[939, 945, 946, 947]</sup>。由于基于枢轴语言的方法与模型结构无关，这些方法也适用于神经机器翻译，并且取得了不错的效果<sup>[940, 948]</sup>。

基于枢轴语言的方法可以被描述为如图16.11所示的过程。这里，使用虚线表示具有双语平行语料库的语言对，并使用带有箭头的实线表示翻译方向，令  $x$ 、 $y$  和  $p$  分别表示源语言、目标语言和枢轴语言，对于输入源语言句子  $x$  和目标语言句子  $y$ ，其翻译过程可以被建模为：

$$P(y|x) = \sum_p P(p|x)P(y|p) \quad (16.6)$$

其中， $p$  表示一个枢轴语言句子。 $P(p|x)$  和  $P(y|p)$  的求解可以直接复用既有的模型和方法。不过，枚举所有的枢轴语言句子  $p$  是不可行的。因此一部分研究工作也探讨了如何选择有效的路径，从  $x$  经过少量  $p$  到达  $y$ <sup>[949]</sup>。

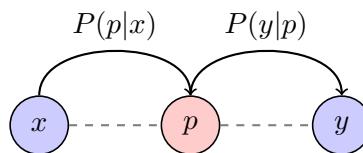


图 16.11 基于枢轴语言的翻译过程

虽然基于枢轴语言的方法简单且易于实现，但该方法也有一些不足。例如，它需要两次翻译，时间开销较大。而且在两次翻译中，翻译错误会进行累积从而产生错误传播问题，导致模型翻译准确性降低。此外，基于枢轴语言的方法仍然假设源语言和枢轴语言（或者目标语言和枢轴语言）之间存在一定规模的双语平行数据，但是这个假设在很多情况下并不成立。比如，对于一些资源极度稀缺的语言，其到英语或者汉语的双语数据仍然十分匮乏，这时使用基于枢轴语言的方法的效果往往也并不理想。虽然存在以上问题，基于枢轴语言的方法仍然受到工业界的青睐，很多在线翻译引擎也在大量使用这种方法进行多语言的翻译。

### 16.3.2 基于知识蒸馏的方法

为了缓解基于枢轴语言的方法中存在的错误传播等问题，可以采用基于知识蒸馏的方法<sup>[549, 950]</sup>。知识蒸馏是一种常用的模型压缩方法<sup>[547]</sup>，基于教师-学生框架，在第十三章已经进行了详细介绍。针对低资源翻译任务，基于教师-学生框架的方法基本思想如图16.12所示。其中，虚线表示具有平行语料库的语言对，带有箭头的实线

表示翻译方向。这里，将枢轴语言 ( $p$ ) 到目标语言 ( $y$ ) 的翻译模型  $P(y|p)$  当作教师模型，源语言 ( $x$ ) 到目标语言 ( $y$ ) 的翻译模型  $P(y|x)$  当作学生模型。然后，用教师模型来指导学生模型的训练，这个过程中学习的目标就是让  $P(y|x)$  尽可能接近  $P(y|p)$ ，这样学生模型就可以学习到源语言到目标语言的翻译知识。举个例子，假设图16.12中  $x$  为源语言德语 “hallo”， $p$  为中间语言英语 “hello”， $y$  为目标语言法语 “bonjour”，则德语 “hallo” 翻译为法语 “bonjour”的概率应该与英语 “hello” 翻译为法语 “bonjour”的概率相近。

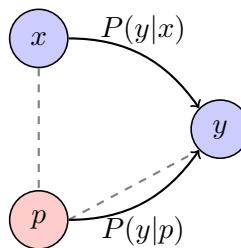


图 16.12 基于知识蒸馏的翻译过程

需要注意的是，基于知识蒸馏的方法基于一个假设：如果源语言句子  $x$ 、枢轴语言句子  $p$  和目标语言句子  $y$  这三者互译，则  $P(y|x)$  应接近  $P(y|p)$ ，即：

$$P(y|x) \approx P(y|p) \quad (16.7)$$

和基于枢轴语言的方法相比，基于知识蒸馏的方法无需训练源语言到枢轴语言的翻译模型，也就无需经历两次翻译过程。不过，基于知识蒸馏的方法仍然需要显性地使用枢轴语言进行桥接，因此仍然面临着“源语言 → 枢轴语言 → 目标语言”转换中信息丢失的问题。比如，当枢轴语言到目标语言翻译效果较差时，由于教师模型无法提供准确的指导，学生模型也无法取得很好的学习效果。

### 16.3.3 基于迁移学习的方法

**迁移学习** (Transfer Learning) 是一种机器学习的方法，指的是一个预训练的模型被重新用在另一个任务中，而并不是从头训练一个新的模型<sup>[547]</sup>。迁移学习的目标是将某个领域或任务上学习到的知识应用到新的领域或问题中。在机器翻译中，可以用富资源语言的知识来改进低资源语言上的机器翻译性能，也就是将富资源语言中的知识迁移到低资源语言中。

基于枢轴语言的方法需要显性地建立“源语言 → 枢轴语言 → 目标语言”的路径。这时，如果路径中某处出现了问题，就会成为整个路径的瓶颈。如果使用多个枢轴语言，这个问题就会更加严重。不同于基于枢轴语言的方法，迁移学习无需进行两次翻译，也就避免了翻译路径中错误累积的问题。如图16.13所示，迁移学习将所

有任务分类为源任务和目标任务，目标是将源任务中的知识迁移到目标任务当中。

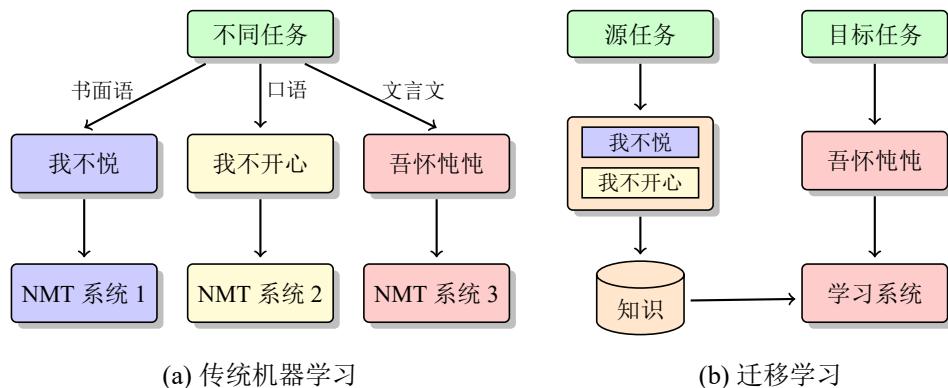


图 16.13 传统机器学习和迁移学习方法对比

## 1. 参数初始化方法

在解决多语言翻译问题时，首先需要在富资源语言上训练一个翻译模型，将其称之为**父模型**（Parent Model）。在对父模型的参数进行初始化的基础上，训练低资源语言的翻译模型，称之为**子模型**（Child Model），这意味着低资源翻译模型将不会从随机初始化的参数开始学习，而是从父模型的参数开始<sup>[951, 952, 953]</sup>。这时，也可以把参数初始化过程看作是迁移学习。在图16.14中，左侧模型为父模型，右侧模型为子模型。这里假设从英语到汉语的翻译为富资源翻译，从英语到西班牙语的翻译为低资源翻译，则首先用英中双语平行语料库训练出一个父模型，之后再用英语到西班牙语的数据在父模型上微调得到子模型，这个子模型即为迁移学习的模型。此过程可以看作是在富资源语言训练模型上使用低资源语言的数据进行微调，将富资源语言中的知识迁移到低资源语言中，从而提升低资源语言的模型性能。

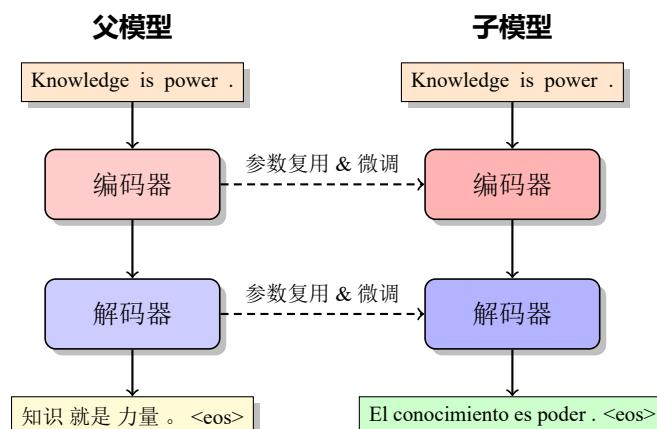


图 16.14 参数初始化方法示意图

这种方法尽管在某些低资源语言上取得了成功，但在资源极度匮乏或零资源的翻译任务中仍然表现不佳<sup>[954]</sup>。具体而言，如果子模型训练数据过少，无法通过训练弥补父模型跟子模型之间的差异，因此微调的结果很差。一种解决方案是先预训练一个多语言的模型，然后固定这个预训练模型的部分参数后训练父模型，最后从父模型中微调子模型<sup>[955]</sup>。这样做好处在于先用预训练提取父模型的任务和子模型的任务之间通用的信息（保存在模型参数里），然后强制在训练父模型的时候保留这些信息（通过固定参数），这样最后微调子模型的时候就可以利用这些通用信息，减少父模型和子模型之间的差异，使得微调的结果得到提升<sup>[956]</sup>。

## 2. 多语言单模型系统

**多语言单模型方法**（Multi-lingual Single Model-based Method）也可以被看做是一种迁移学习。多语言单模型方法尤其适用于翻译方向较多的情况，因为每一个翻译方向单独训练一个模型是不实际的，不仅因为设备资源和时间上的限制，还因为很多翻译方向都没有双语平行数据<sup>[929, 938, 957]</sup>。比如，要翻译 100 个语言之间互译的系统，理论上就需要训练  $100 \times 99$  个翻译模型，代价十分巨大。这时就可以使用多语言单模型方法。

多语言单模型系统是指用单个模型具有多个语言方向翻译的能力。对于源语言集合  $G_x$  和目标语言集合  $G_y$ ，多语言单模型的学习目标是学习一个单一的模型，这个模型可以进行任意源语言到任意目标语言的翻译，即同时支持所有  $\{(l_x, l_y) | x \in G_x, y \in G_y\}$  的翻译。**多语言单模型方法**又可以进一步分为**一对多**<sup>[928]</sup>、**多对一**<sup>[561]</sup>和**多对多**<sup>[958]</sup>的方法。不过这些方法本质上是相同的，因此这里以多对多翻译为例进行介绍。

在模型结构方面，多语言模型与普通的神经机器翻译模型相同，都是标准的编码器-解码器结构。多语言单模型方法的一个假设是：不同语言可以共享同一个表示空间。因此，该方法使用同一个编码器处理所有的源语言句子，使用同一个解码器处理所有的目标语言句子。为了使多个语言共享同一个解码器（或编码器），一种简单的方法是直接在输入句子上加入语言标记，让模型显性地知道当前句子属于哪个语言。如图16.15所示，在此示例中，标记“<spanish>”表示目标句子为西班牙语，标记“<german>”表示目标句子为德语，则模型在进行翻译时便会将句子开头加有“<spanish>”标签的句子翻译为西班牙语<sup>[929]</sup>。假设训练时有英语到西班牙语“<spanish> Hello” → “Hola” 和法语到德语“<german> Bonjour” → “Hallo”的双语句对，则在解码时候输入英语“<german> Hello”时就会得到解码结果“Hallo”。

多语言单模型系统无需显性训练基于枢轴语言的翻译系统，而是共享多个语言的编码器和解码器，因此极大地提升了数据资源的利用效率。其适用的的一个极端场景是零资源翻译，即源语言和目标语言之间没有任何平行数据。以法语到德语的翻译为例，假设此翻译语言方向为零资源，即没有法语到德语的双语平行数据，但是有法语到其他语言（如英语）的双语平行数据，也有其他语言（如英语）到德语

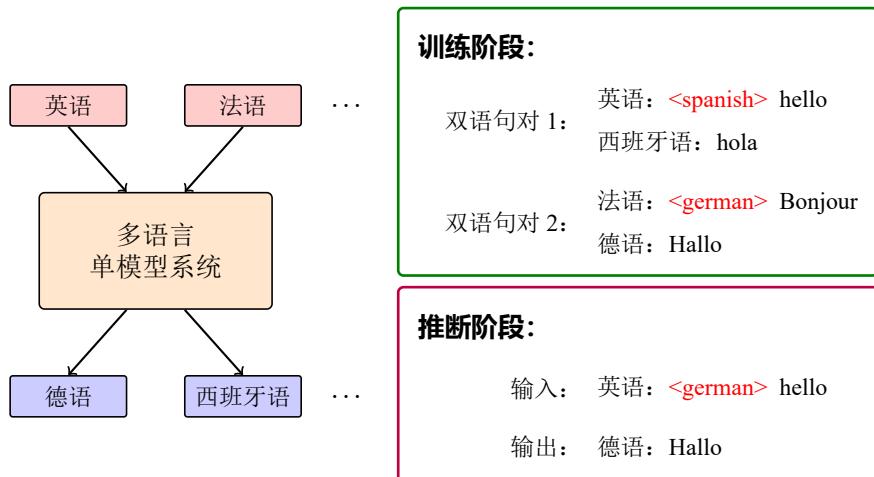


图 16.15 多语言单模型结构示意图

的双语平行数据。这时直接运行图16.15所示模型，可以学习到法语到英语、英语到德语的翻译能力，同时具备了法语到德语的翻译能力，即零资源翻译能力。从这个角度说，零资源神经机器翻译也需要枢轴语言，只是这些枢轴语言数据仅在训练期间使用<sup>[929]</sup>，而无需生成伪并行语料库。这种使用枢轴语言的方式也被称作**隐式桥接**（Implicit Bridging）。

另外，使用多语言单模型系统进行零资源翻译的一个优势在于，它可以最大程度上利用其它语言的数据。还是以上面提到法语到德语的零资源翻译任务为例，除了使用法语到英语、英语到德语的数据之外，所有法语到其它语言、其它语言到德语的数据都是有价值的，这些数据可以强化对法语句子的表示能力，同时强化对德语句子的生成能力。这个优点也是16.3.1节所介绍的传统基于枢轴语言方法所不具备的。

不过，多语言单模型系统经常面临脱靶翻译问题，即把源语言翻译成错误的目标语言，比如要求翻译成英语，结果却是汉语或者英语夹杂其他语言的字符。这是因为多语言单模型系统对所有语言都使用一样的参数，导致不同语言字符混合时不容易让模型进行区分。针对这个问题，可以在原来共享参数的基础上为每种语言添加额外的独立的参数，使得每种语言拥有足够的建模能力，以便于更好地完成特定语言的翻译<sup>[959, 960]</sup>。

## 16.4 无监督机器翻译

低资源机器翻译的一种极端情况是：没有任何可以用于模型训练的双语平行数据。一种思路是借用多语言翻译方面的技术（见16.3节），利用基于枢轴语言或者零资源的方法构建翻译系统。但是，这类方法仍然需要多个语种的平行数据。对于某一个语言对，在只有源语言和目标语言单语数据的前提下，能否训练一个翻译模型呢？

这里称这种不需要双语数据的机器翻译方法为**无监督机器翻译**（Unsupervised Machine Translation）。

直接进行无监督机器翻译是很困难的。一个简单可行的思路是把问题进行分解，然后分别解决各个子问题，最后形成完整的解决方案。放到无监督机器翻译里面，可以首先使用无监督方法寻找词与词之间的翻译，然后在此基础上，进一步得到句子到句子的翻译模型。这种“由小到大”的建模思路十分类似于统计机器翻译中的方法（见第七章）。

### 16.4.1 无监督词典归纳

**词典归纳**（Bilingual Dictionary Induction, BDI）可用于处理不同语言间单词级别的翻译任务。在统计机器翻译中，词典归纳是一项核心的任务，它从双语平行语料中发掘互为翻译的单词，是翻译知识的主要来源<sup>[961]</sup>。在神经机器翻译中，词典归纳通常被用在无监督机器翻译、多语言机器翻译等任务中。这里，单词通过实数向量进行表示，即词嵌入。所有单词分布在一个多维空间中，而且研究人员发现：词嵌入空间在一些语言中显示出类似的结构，这使得直接利用词嵌入来构建双语词典成为可能<sup>[898]</sup>。其基本想法是先将来自不同语言的词嵌入投影到共享嵌入空间中，然后在这个共享空间中归纳出双语词典，原理如图16.16所示。较早的尝试是使用一个包含数千词对的种子词典作为锚点来学习从源语言到目标语词嵌入空间的线性映射，将两个语言的单词投影到共享的嵌入空间之后，执行一些对齐算法即可得到双语词典<sup>[898]</sup>。最近的研究表明，词典归纳可以在更弱的监督信号下完成，这些监督信号来自更小的种子词典<sup>[962]</sup>、相同的字符串<sup>[963]</sup>，甚至仅仅是共享的数字<sup>[964]</sup>。

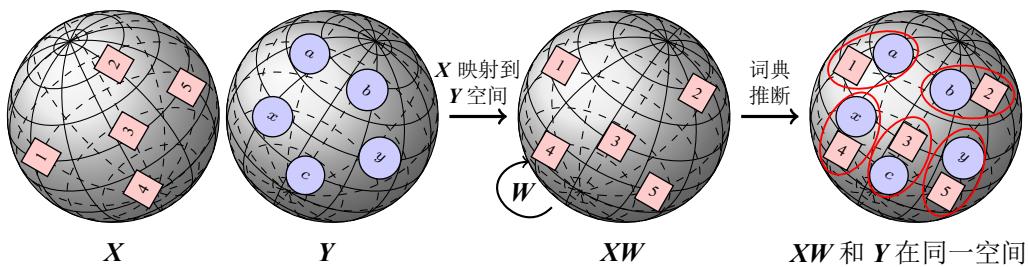


图 16.16 词典归纳原理图

研究人员也提出了完全无监督的词典归纳方法，这类方法不依赖于任何种子词典即可实现词典归纳，下面进行介绍。

#### 1. 基本框架

无监督词典归纳的核心思想是充分利用词嵌入空间近似同构的假设<sup>[965]</sup>，基于一些无监督匹配的方法来得到一个初始化的种子词典，之后利用该种子词典作为起始

监督信号不断进行微调进一步提高性能。总结起来，无监督词典归纳系统通常包括以下两个阶段：

- **基于无监督的分布匹配。**该步骤利用一些无监督的方法来得到一个包含噪声的初始化词典  $D$ 。
- **基于有监督的微调。**利用两个单语词嵌入和第一步中学习到的种子字典执行一些对齐算法来迭代微调，例如，**普氏分析**（Procrustes Analysis）<sup>[966]</sup>。

其具体流程如图16.17所示，主要步骤包括：

- 对于图16.17(a) 中的分布在不同空间中的两个单语词嵌入  $X$  和  $Y$ ，基于两者近似同构的假设，利用无监督匹配的方法来得到一个粗糙的线性映射  $W$ ，使得两个空间能大致对齐，结果如图16.17(b) 所示。
- 在这个共享空间中执行对齐算法从而归纳出一个种子词典，如图16.17(c) 所示。
- 利用种子词典不断迭代微调进一步提高映射  $W$  的性能，最终映射的效果如图16.17(d) 所示，之后即可从中推断出词典，并作为最后的结果。

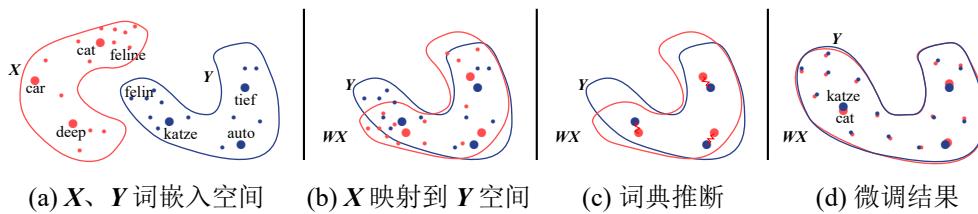


图 16.17 无监督词典归纳流程图<sup>[967]</sup>

不同的无监督方法最大的区别主要在于第一阶段，获得初始种子词典的手段，而第二阶段微调的原理都大同小异。第一阶段的主流方法主要有两大类：

- **基于生成对抗网络的方法**<sup>[965, 967, 968, 969]</sup>。在这个方法中，通过生成器来产生映射  $W$ ，鉴别器负责区分随机抽样的元素  $WX$  和  $Y$ ，两者共同优化收敛后即可得到映射  $W$ 。
- **基于 Gromov-wasserstein 的方法**<sup>[965, 970, 971, 972]</sup>。Wasserstein 距离是度量空间中定义两个概率分布之间距离的函数。在这个任务中，它用来衡量不同语言中单词对之间的相似性，利用空间近似同构的信息可以定义出一些目标函数，之后通过优化该目标函数也可以得到映射  $W$ 。

在得到映射  $W$  之后，对于  $X$  中的任意一个单词  $x_i$ ，通过  $WE(x_i)$  将其映射到空间  $Y$  中（ $E(x_i)$  表示的是单词  $x_i$  的词嵌入向量），然后在  $Y$  中找到该点的最近邻点  $y_j$ ，于是  $y_j$  就是  $x_i$  的翻译词，重复该过程即可归纳出种子词典  $D$ ，第一阶段结束。

事实上，由于第一阶段缺乏监督信号，得到的种子词典  $D$  会包含大量的噪音，因此需要进行进一步的微调。

微调的原理普遍基于普氏分析<sup>[898]</sup>。假设现在有一个种子词典  $D = \{x_i, y_i\}$  其中  $i \in \{1, n\}$ ，和两个单语词嵌入  $\mathbf{X}$  和  $\mathbf{Y}$ ，那么就可以将  $D$  作为映射锚点（Anchor）学习一个转移矩阵  $\mathbf{W}$ ，使得  $\mathbf{WX}$  与  $\mathbf{Y}$  这两个空间尽可能相近，此外通过对  $\mathbf{W}$  施加正交约束可以显著提高性能<sup>[973]</sup>，于是这个优化问题就转变成了普鲁克问题（Procrustes Problem）<sup>[963]</sup>，可以通过奇异值分解（Singular Value Decomposition, SVD）来获得近似解。这里用  $\mathbf{X}'$  和  $\mathbf{Y}'$  表示  $D$  中源语言单词和目标语言单词的词嵌入矩阵，优化  $\mathbf{W}$  的过程可以被描述为：

$$\begin{aligned}\hat{\mathbf{W}} &= \arg \min_{\mathbf{W} \in O_d(\mathbb{R})} \|\mathbf{WX}' - \mathbf{Y}'\|_F \\ &= \mathbf{UV}^T\end{aligned}\quad (16.8)$$

$$\text{s.t. } \mathbf{U}\Sigma\mathbf{V}^T = \text{SVD}(\mathbf{Y}'\mathbf{X}'^T) \quad (16.9)$$

其中， $\|\cdot\|_F$  表示矩阵的 Frobenius 范数，即矩阵元素绝对值的平方和再开方， $d$  是词嵌入的维度， $O_d(\mathbb{R})$  表示  $d \times d$  的实数空间， $\text{SVD}(\cdot)$  表示奇异值分解。利用上式可以获得新的  $\mathbf{W}$ ，通过  $\mathbf{W}$  可以归纳出新的  $D$ ，如此迭代进行微调最后即可以得到收敛的  $D$ 。

较早的无监督方法是基于生成对抗网络的方法<sup>[967, 968, 974]</sup>，这是一个很自然的想法，利用生成器产生单词间的映射，然后用判别器来区别两个空间。然而研究表明生成对抗网络缺乏稳定性，容易在低资源语言对上失败<sup>[975]</sup>，因此有不少改进的工作，比如：利用变分自编码器（Variational Autoencoders, VAEs）来捕获更深层次的语义信息并结合对抗训练的方法<sup>[969, 976]</sup>；通过改进最近邻点的度量函数来提升性能的方法<sup>[977, 978]</sup>；利用多语言信号来提升性能的方法<sup>[972, 979, 980, 981]</sup>；也有一些工作舍弃生成对抗网络，通过直接优化空间距离来进行单词的匹配<sup>[965, 970, 982, 983]</sup>。

## 2. 健壮性问题

目前很多无监督词典归纳方法在相似语言对比如英-法、英-德上已经取得不错的成绩，然而在远距离语言对比如英-中，英-日上的性能仍然很差<sup>[984, 985]</sup>。研发健壮的无监督词典归纳方法仍然存在挑战。这有多个层面的原因：

- 词典归纳依赖于基于大规模单语数据训练出来的词嵌入，而词嵌入会受到单语数据的来源、数量、词向量训练算法、超参数配置等多方面因素的影响，这很容易导致不同情况下词嵌入结果的差异很大。
- 词典归纳强烈依赖于词嵌入空间近似同构的假设，然而许多语言之间天然的差异导致该假设并不成立。因为无监督系统通常是基于两阶段的方法，起始阶段

由于缺乏监督信号很难得到质量较高的种子词典，进而导致后续阶段无法完成准确的词典归纳<sup>[985, 986]</sup>。

- 由于词嵌入这种表示方式的局限性，模型无法实现单词多对多的对齐，而且对于一些相似的词或者实体，模型也很难实现对齐。

无监督方法的健壮性是一个很难解决的问题。对于词典推断这个任务来说，是否有必要进行完全无监督的学习仍然值得商榷。因为其作为一个底层任务，不仅可以利用词嵌入，还可以利用单语、甚至是双语信息。此外，基于弱监督的方法代价也不是很大，只需要数千个词对即可。有了监督信号的引导，健壮性问题就能得到一定的缓解。

## 16.4.2 无监督统计机器翻译

在无监督词典归纳的基础上，可以进一步得到句子间的翻译，实现无监督机器翻译<sup>[987]</sup>。统计机器翻译作为机器翻译的主流方法，对其进行无监督学习可以帮助构建初始的无监督机器翻译系统，从而进一步帮助训练更为先进的无监督神经机器翻译系统。以基于短语的统计机器翻译为例，系统主要包含短语表、语言模型、调序模型以及权重调优等模块（见第七章）。其中短语表和模型调优需要双语数据，而语言模型和（基于距离的）调序模型只依赖于单语数据。因此，如果可以通过无监督的方法完成短语表和权重调优，那么就得到了无监督统计机器翻译系统<sup>[988]</sup>。

### 1. 无监督短语归纳

回顾统计机器翻译中的短语表，它类似于一个词典，对一个源语言短语给出相应的译文<sup>[285]</sup>。只不过词典的基本单元是词，而短语表的基本单元是短语（或  $n$ -gram）。此外短语表还提供短语翻译的得分。既然短语表跟词典如此相似，那么很容易就可以把无监督词典归纳的方法移植到短语上，也就是把里面的词替换成短语，就可以无监督地得到短语表。

如16.4.1节所述，无监督词典归纳的方法依赖于词的分布式表示，也就是词嵌入。因此当把无监督词典归纳拓展到短语上时，首先需要获得短语的分布式表示。比较简单的方法是把词换成短语，然后借助与无监督词典归纳相同的算法得到短语的分布式表示。最后直接应用无监督词典归纳方法，得到源语言短语与目标语言短语之间的对应。

在得到了短语翻译的基础上，需要确定短语翻译的得分。在无监督词典归纳中，在推断词典的时候会为一对源语言单词和目标语言单词打分（词嵌入之间的相似度），再根据打分来决定哪一个目标语言单词更有可能是当前源语言单词的翻译。在无监督短语归纳中，这样一个打分已经提供了对短语对质量的度量，因此经过适当的归一化处理后就可以得到短语翻译的得分。

## 2. 无监督权重调优

有了短语表之后，剩下的问题是如何在没有双语数据的情况下进行模型调优，从而把短语表、语言模型、调序模型等模块融合起来<sup>[232]</sup>。在统计机器翻译系统中，短语表可以提供短语的翻译，而语言模型可以保证从短语的翻译拼装得到的句子的流畅度，因此统计机器翻译模型即使在没有权重调优的基础上也已经具备了一定的翻译能力。所以一个简单而有效的无监督方法就是使用未经过模型调优的统计机器翻译模型进行回译，也就是将目标语言句子翻译成源语言句子后，再将翻译得到的源语句言子当成输入而目标语言句子当成标准答案，完成权重调优。

经过上述的无监督模型调优后，就获得了一个效果更好的翻译模型。这时候，可以使用这个翻译模型去产生质量更高的数据，再用这些数据来继续对翻译模型进行调优，如此反复迭代一定次数后停止。这个方法也被称为**迭代优化**(Iterative Refinement)<sup>[988]</sup>。

迭代优化也会带来另外一个问题：在每一次迭代中都会产生新的模型，应该什么时候停止生成新模型，挑选哪一个模型呢？因为在无监督的场景当中，没有任何真实的双语数据可以使用，所以无法使用监督学习里的校验集来对每个模型进行检验并筛选。另外，即使有很少量的双语数据（比如数百条双语句对），直接在上面挑选模型和调整超参数会导致过拟合问题，使得最后结果越来越差。一个经验上非常高效的模型选择方法是：事先从训练集里挑选一部分句子作为校验集不参与训练，再使用当前的模型把这些句子翻译过去之后再翻译回来（源语言 → 目标语言 → 源语言，或者目标语言 → 源语言 → 目标语言），得到的结果跟原始的结果计算 BLEU，得分越高则效果越好。这种方法已被证明跟使用大规模双语校验集的结果是高度相关的<sup>[882]</sup>。

### 16.4.3 无监督神经机器翻译

既然神经机器翻译已经在很多任务上优于统计机器翻译，为什么不直接做无监督神经机器翻译呢？实际上，由于神经网络的黑盒特性使其无法像统计机器翻译那样进行拆解，并定位问题。因此需要借用其它无监督翻译系统来训练神经机器翻译模型。

#### 1. 基于无监督统计机器翻译的方法

一个简单的方法是，借助已经成功的无监督方法来为神经机器翻译模型提供少量双语监督信号。初始的监督信号可能很少或者包含大量噪声，因此需要逐步优化数据来重新训练出更好的模型。这也是目前绝大多数无监督神经机器翻译方法的核心思路。这个方案最简单最实现就是借助已经构建的无监督统计机器翻译模型，用它产生伪双语数据来训练神经机器翻译模型，然后进行迭代回译来进行数据优化<sup>[989]</sup>。这个方法的优点是直观，并且性能稳定，容易调试（所有模块都互相独立）。缺点是复杂繁琐，涉及许多超参数调整工作，而且训练代价较大。

## 2. 基于无监督词典归纳的方法

另一个思路是直接从无监督词典归纳中得到神经机器翻译模型，从而避免繁琐的无监督统计机器翻译模型的训练，同时也避免神经机器翻译模型继承统计机器翻译模型的错误。这种方法的核心就是把翻译看成一个两阶段的过程：

- 首先，无监督词典归纳通过双语词典把一个源语言句子转换成一个不通顺但是意思完整的目标语言句子。
- 然后，把这样一个不通顺的句子改写成一个流畅的句子，同时保留原来的含义，最后达到翻译的目的。

而第二阶段的改写任务其实也是一个特殊的翻译任务，只不过现在的源语言和目标语言是使用不同的方式表达的同一种语言的句子。因此可以使用神经机器翻译模型来完成这个任务，而且由于这里不涉及双语数据而只需要单语数据，模型的训练也将是无监督的。这样的方法不再需要无监督统计机器翻译，并且适应能力很强。对于新语种，不需要重新训练神经机器翻译模型，只需要训练无监督词典归纳进行词的翻译，再使用相同的模型进行改写。

但是，目前训练数据需要使用其他语种的双语数据来进行构造（把源语言句子里每个词使用双语词典进行翻译作为输入，输出的目标语言句子不变）。虽然可以通过把单语句子根据规则或者随机进行打乱来生成训练数据，但是这些句子与真实的句子差异较大，导致训练-测试不一致的问题。而且这样一个两阶段的过程会产生错误传播的问题，比如无监督词典归纳对一些词进行了错误的翻译，那么这些错误的翻译会被送下一阶段进行改写，因为翻译模型这时候已经无法看到源语言句子来进行修正，所以最终的结果将继承无监督词典归纳的错误<sup>[990]</sup>。

## 3. 更深层的融合

为了获得更好的神经机器翻译模型，可以对训练流程和模型做更深度的整合。第十章已经介绍，神经机器翻译模型的训练包含两个阶段：初始化和优化。而无监督神经机器翻译的核心思路也对应这两个阶段，因此可以考虑在模型的初始化阶段使用无监督方法提供初始的监督信号，然后不但优化模型的参数，还优化训练使用的数据，从而避免流水线带来的错误传播。其中初始的监督信号可以通过两种方法提供给模型。一种是直接使用无监督方法提供最初的伪双语数据，然后训练最初的翻译模型。另一种则是借助无监督方法来初始化模型，得到最初的翻译模型后，直接使用初始化好的翻译模型产生伪双语数据，然后训练自己，如图16.18所示。图16.18(a)的一个简单实现是利用无监督词典归纳得到词典，用这个词典对单语数据进行逐词的翻译，得到最初的伪双语数据，再在这些数据上训练最初的翻译模型，最后不断地交替优化数据和模型，得到更好的翻译模型和质量更好的伪数据<sup>[881]</sup>。这样的做法通过不断优化训练用的双语数据，摆脱了无监督词典归纳在最初的伪双语数据中遗留下来的错误，同时也避免了使用无监督统计机器翻译模型的代价。图16.18(b)的实现

则依赖于具体的翻译模型初始化方法，下一节会讨论翻译模型的不同初始化方法。

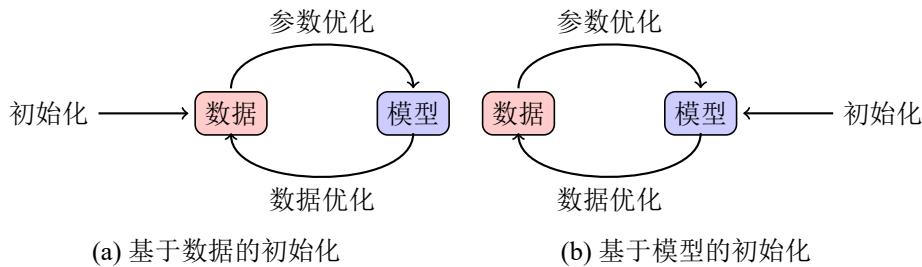


图 16.18 模型参数初始化策略

#### 4. 其它问题

一般可以认为，在生成的伪数据上优化模型会使模型变得更好，这时候对这个更好的模型使用数据增强的手段（如回译等）就可以生成更好的训练数据。这样的一个数据优化过程依赖于一个假设：模型经过优化后会生成比原始数据更好的数据。而在数据优化和参数优化的共同影响下，模型非常容易拟合数据中的简单模式，使得模型倾向产生包含这种简单模式的数据，造成模型对这种类型数据过拟合的现象。一个常见的问题是模型对任何输入都输出相同的译文，这时候翻译模型无法产生任何有意义的结果，也就是，数据优化产生的数据里无论什么目标语言对应的源语言都是同一个句子。这种情况下翻译模型虽然能降低损失，但是它不能学会任何源语言跟目标语言之间的对应关系，也就无法进行正确翻译。这个现象也反映出无监督机器翻译训练的脆弱性。

比较常见的解决方案是在双语数据对应的目标函数外增加一个语言模型的目标函数。因为，在初始阶段，由于数据中存在大量不通顺的句子，额外的语言模型目标函数能把部分句子纠正过来，使得模型逐渐生成更好的数据<sup>[882]</sup>。这个方法在实际应用中非常有效，尽管目前还没有太多理论上的支持。

无监督神经机器翻译还有两个关键的技巧：

- **词表共享：**对于源语言和目标语言里都一样的词使用同一个词嵌入，而不是源语言和目标语言各自对应一个词嵌入，比如，阿拉伯数字或者一些实体名字。这样相当于告诉模型这个词在源语言和目标语言里面表达同一个意思，隐式地引入了单词翻译的监督信号。在无监督神经机器翻译里词表共享搭配子词切分会更加有效，因为子词的覆盖范围广，比如，多个不同的词可以包含同一个子词。
- **模型共享：**与多语言翻译系统类似，使用同一个翻译模型来进行正向翻译（源语言 → 目标语言）和反向翻译（目标语言 → 源语言）。这样做降低了模型的参数量。而且，两个翻译方向可以互相为对方起到正则化的作用，减小了过拟合的风险。

最后图16.19简单总结了无监督神经机器翻译的流程。下面分别讨论：无监督神经机器翻译里面模型的初始化，以及语言模型目标函数的选择。

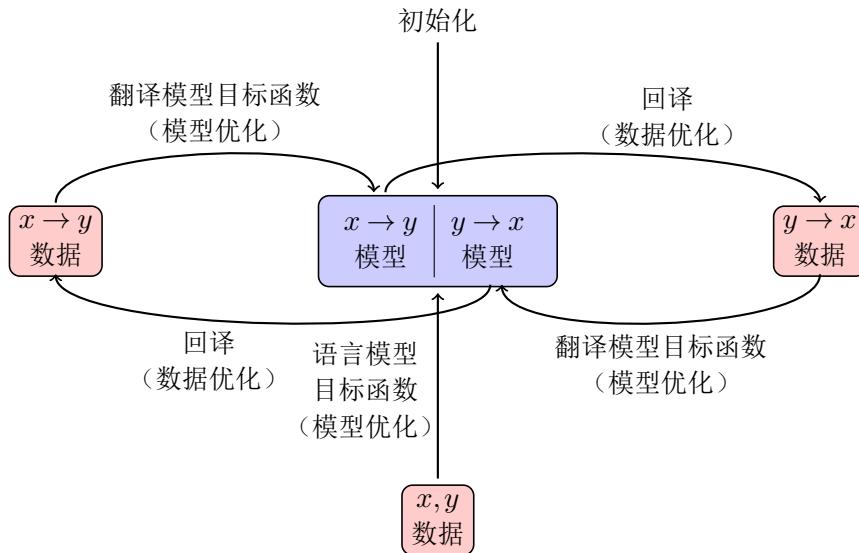


图 16.19 无监督神经机器翻译模型训练流程

### 1) 模型参数初始化

无监督神经机器翻译的关键在于如何提供最开始的监督信号，从而启动后续的迭代流程。无监督词典归纳已经可以提供一些可靠的监督信号，那么如何在模型初始化中融入这些信息？既然神经机器翻译模型都使用词嵌入作为输入，而且无监督词典归纳也是基于两种语言共享的词嵌入空间，那么可以使用共享词嵌入空间的词嵌入结果来初始化模型的词嵌入层，然后在这个基础上训练模型。比如，两个语言里意思相近的词对应的词嵌入会比其他词更靠近对方<sup>[888]</sup>。为了防止机器翻译训练过程中模型参数的更新会破坏词嵌入中蕴含的信息，通常初始化后会固定模型的词嵌入层不让其更新<sup>[988]</sup>。

进一步，无监督神经机器翻译能在提供更少监督信号的情况下启动，也就是可以去除无监督词典归纳这一步骤<sup>[991]</sup>。这时候模型的初始化直接使用共享词表的预训练模型的参数作为起始点。这个预训练模型直接使用前面提到的预训练方法（如 MASS）进行训练，区别在于模型的结构需要严格匹配翻译模型。此外，这个模型不仅仅只在一个语言的单语数据上进行训练，而是同时在两个语言的单语数据上进行训练，并且两个语言的词表进行共享。前面提到，在共享词表特别是共享子词词表的情况下，已经隐式地告诉模型源语言和目标语言里一样的（子）词互为翻译，相当于模型使用了少量的监督信号。在这基础上使用两个语言的单语数据进行预训练，通过模型共享进一步挖掘了语言之间共通的部分。因此，使用预训练模型进行初始化后，无监督神经机器翻译模型已经得到大量的监督信号，从而得以不断通过优化来提升模型性能。

## 2) 语言模型的使用

无监督神经机器翻译的一个重要部分就是来自语言模型的目标函数。因为翻译模型本质上是在完成文本生成任务，所以只有文本生成类型的语言模型建模方法才可以应用到无监督神经机器翻译里。比如，给定前文预测下一词就是一个典型的自回归生成任务（见第二章），因此可以运用到无监督神经机器翻译里。但是，目前在预训练里流行的 BERT 等模型是掩码语言模型<sup>[123]</sup>，不能直接在无监督神经机器翻译里使用。

另外一个在无监督神经机器翻译中比较常见的语言模型目标函数则是降噪自编码器。它也是文本生成类型的语言模型建模方法。对于一个句子  $x$ ，首先使用一个噪声函数  $x' = \text{noise}(x)$  来对  $x$  注入噪声，产生一个质量较差的句子  $x'$ 。然后，让模型学习如何从  $x'$  还原出  $x$ 。这样的目标函数比预测下一词更贴近翻译任务，因为它是一个序列到序列的映射，并且输入、输出两个序列在语义上是等价的。这里之所以采用  $x'$  而不是  $x$  自己来预测  $x$ ，是因为模型可以通过简单的复制输入作为输出来完成从  $x$  预测  $x$  的任务，很难学到有价值的信息。并且在输入中注入噪声会让模型更加健壮，因此模型可以学会如何利用句子中噪声以外的信息来得到正确的输出。通常来说，噪声函数有三种形式，如表16.1所示。

表 16.1 三种噪声加噪方式示例（原句为“我喜欢吃苹果。”）

噪声函数	描述	例子
交换	句子中任意两个词进行交换	“我 喜欢 苹果 吃。”
删除	句子中的词按一定概率被删除	“我 喜欢 吃。”
空白	句子中的词按一定概率被替换成空白符	“我 _____ 吃 苹果。”

实际应用中以上三种形式的噪声函数都会被使用到，其中在交换方法中距离越近的词越容易被交换，并且要保证交换次数有上限，而删除和空白方法里词的删除和替换概率通常都非常低，如 0.1 等。

## 16.5 领域适应

机器翻译常常面临训练时与应用时所处领域不一致的问题，比如，将一个在新闻类数据上训练的翻译系统应用在医学文献翻译任务上。不同领域的句子通常存在着很大的区别，比如，日常用语的结构较为简单，而在化学领域的学术论文中，单词和句子结构较为复杂。此外，不同领域之间存在着较为严重的一词多义问题，即同一个词在不同领域中经常会有不同的含义。实例16.1展示了英语单词 pitch 在不同领域的不同词义。

### 实例 16.1 单词 pitch 在不同领域的不同词义

体育领域：The rugby tour was a disaster both on and off the **pitch**.

这次橄榄球巡回赛在场上、场下都彻底失败。

化学领域: The timbers of similar houses were painted with **pitch**.

类似房屋所用的栋木刷了**沥青**。

声学领域: A basic sense of rhythm and **pitch** is essential in a music teacher.

基本的韵律感和**音高**感是音乐教师的必备素质。

在机器翻译任务中,新闻等领域的双语数据相对容易获取,所以机器翻译在这些领域上表现较佳。然而,即使在富资源语种上,化学、医学等专业领域的双语数据也十分有限。如果直接使用这些低资源领域的数据来训练机器翻译模型,由于数据稀缺问题,会导致模型的性能较差<sup>[992]</sup>。如果混合多个领域的数据增大训练数据规模,不同领域数据量之间的不平衡会导致数据较少的领域训练不充分,使得在低资源领域上的翻译结果不尽人意<sup>[993]</sup>。

领域适应方法是利用源领域的知识来改进目标领域模型效果的方法,该方法可以有效地减少模型对目标领域数据的依赖。领域适应主要有两类方法:

- **基于数据的方法**。利用源领域的双语数据或目标领域单语数据进行**数据选择**或**数据增强**,来增加模型训练的数据量。
- **基于模型的方法**。针对领域适应开发特定的模型结构、训练策略和推断方法。

### 16.5.1 基于数据的方法

在统计机器翻译时代,如何有效利用外部数据来改善目标领域的翻译效果已经备受关注。其中的绝大多数方法与翻译模型无关,因此这些方法也同样适用于神经机器翻译。基于数据的领域适应可以分为基于数据加权的方法、基于数据选择的方法、基于伪数据的方法。图16.20展示了这几种方法的示意图。

#### 1. 基于数据加权/数据选择的方法

一种观点认为,数据量较少的领域数据应该在训练过程中获得更大的权重,从而使这些更有价值的数据发挥出更大的作用<sup>[994, 995]</sup>。实际上,基于数据加权的方法与第十三章中基于样本价值的学习方法是一致的,只是描述的场景略有不同。这类方法**本质上在解决类别不均衡问题**(Class Imbalance Problem)<sup>[996]</sup>。数据加权可以通过修改损失函数,将其缩放 $\alpha$ 倍来实现( $\alpha$ 是样本的权重)。在具体实践中,也可以直接将低资源的领域数据进行复制<sup>1</sup>达到与其相同的效果<sup>[997]</sup>。

数据选择是数据加权的一种特殊情况,它可以被看做是样本权重非零即一的情况。具体来说,可以直接选择与领域相关的数据参与训练<sup>[993]</sup>。这种方法并不需要使用全部数据进行训练,因此模型的训练成本较低。由于第十三章已经对数据加权和数据选择方法进行了详细介绍,这里不再赘述。

<sup>1</sup>相当于对数据进行重采样

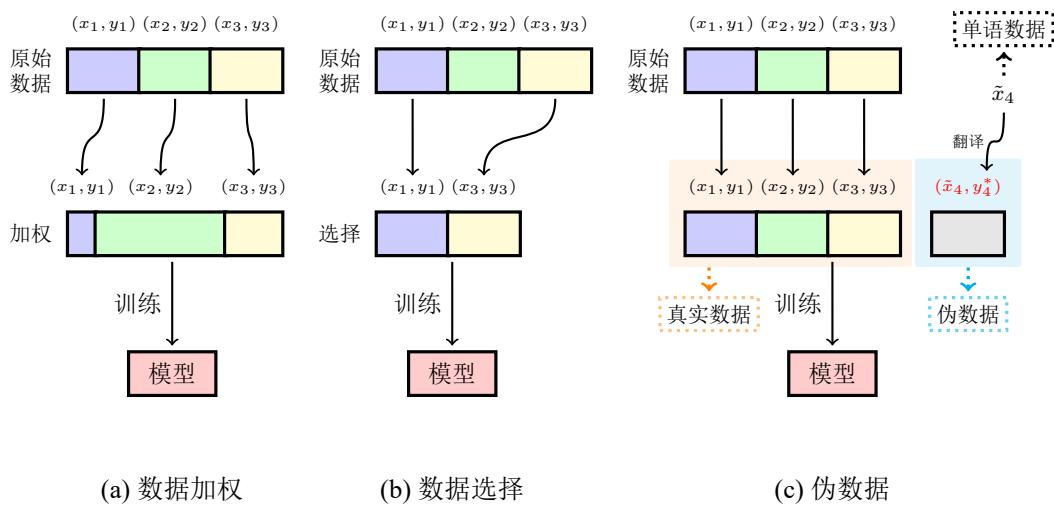


图 16.20 基于数据的领域适应方法

## 2. 基于伪数据的方法

数据选择方法可以从源领域中选择出和目标领域相似的样本用于训练，但可用的数据是较为有限的。因此，另外一种思路是对现有的双语数据进行修改<sup>[998]</sup>（如抽取双语短语对等）或通过单语数据生成伪数据来增加数据量<sup>[999]</sup>。这个问题和16.1小节所面临的场景是基本一致的，因此可以直接复用16.1小节所描述的方法。

## 3. 多领域数据的使用

领域适应中的目标领域往往不止一个，想要同时提升多个目标领域的效果，一种简单的思路是使用前文所述的单领域适应方法对每一个目标领域进行领域适应。不过，与多语言翻译一样，多领域适应也往往伴随着严重的数据稀缺问题，大多数领域的数据量很小，因此无法保证单个领域的领域适应效果。

解决该问题的一种思路是将所有数据混合使用，并训练一个能够同时适应所有领域的模型。同时，为了区分不同领域的数据，可以在样本上增加领域标签<sup>[1000]</sup>。事实上，这种方法与16.3.3节所描述的方法是一样的。它也是一种典型的小样本学习策略，旨在让模型自己从不同类型的样本中寻找联系，进而更加充分地利用数据，改善模型在低资源任务上的表现。

### 16.5.2 基于模型的方法

对于神经机器翻译模型，可以在训练和推断阶段进行领域适应。具体来说，有如下方法：

## 1. 多目标学习

在使用多领域数据时，混合多个相差较大的领域数据进行训练会使单个领域的翻译性能下降<sup>[1001]</sup>。为了解决这一问题，可以对所有训练数据的来源领域进行区分，一个比较典型的做法是在使用多领域数据训练时，在神经机器翻译模型的编码器顶部中添加一个判别器<sup>[613]</sup>，该判别器使用源语言句子  $x$  的编码器表示作为输入，预测句子所属的领域标签  $d$ ，如图16.21所示。为了使预测领域标签  $d$  的正确概率  $P(d|\mathbf{H})$  最大（其中  $\mathbf{H}$  为编码器的隐藏状态），模型在训练过程中最小化如下损失函数  $L_{\text{disc}}$ ：

$$L_{\text{disc}} = -\log P(d|\mathbf{H}) \quad (16.10)$$

在此基础上，加上原始的翻译模型损失函数  $L_{\text{gen}}$ ：

$$L_{\text{gen}} = -\log P(y|x) \quad (16.11)$$

最终可以得到融合后的损失函数，如下：

$$L = L_{\text{disc}} + L_{\text{gen}} \quad (16.12)$$

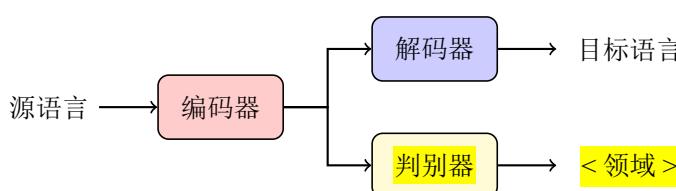


图 16.21 领域判别器示意图

## 2. 训练阶段的领域适应

实际上，16.5.1节所描述的数据加权和数据选择方法本身也是与模型训练相关的，例如，数据选择方法会降低训练数据的数据量。所以在具体实现时，需要对训练策略进行调整。一种方法是在不同的训练轮次动态地改变训练数据集。动态数据选择既可以使得每轮的训练数据均小于全部数据量，从而加快训练进程，又可以缓解训练数据覆盖度不足的问题，具体做法有两种：

- 将完整的数据送入模型，再根据其与目标领域数据的相似度逐次减少每轮的数据量<sup>[625]</sup>。

- 将与目标领域数据相似度最高的句子先送入模型，让模型可以最先学到跟目标领域最相关的知识，逐渐增加数据量<sup>[649]</sup>。

另一种方法是不从随机状态开始训练网络，而是使用翻译性能较好的源领域模型作为初始状态，因为源领域模型中包含着一些通用知识可以被目标领域借鉴。比如，想获得口语的翻译模型，可以使用新闻的翻译模型作为初始状态进行训练。这也可以说是一种预训练-微调方法。

不过这种方法经常会带来灾难性遗忘问题，即在目标领域上过拟合，导致在源领域上的翻译性能大幅度下降（见第十三章）。如果想要保证模型在目标领域和源领域上都有较好的性能，一个比较常用的方法是进行混合微调<sup>[1000]</sup>。具体做法是先在源领域数据上训练一个神经机器翻译模型，然后将目标领域数据复制数倍和源领域数据量相等，之后将数据混合对神经机器翻译模型进行微调。混合微调方法既降低了目标领域数据量小导致的过拟合问题的影响，又带来了更好的微调性能。除了混合微调外，也可以使用知识蒸馏方法缓解灾难性遗忘问题（见16.3节），即对源领域和目标领域进行多次循环知识蒸馏，迭代学习对方领域的知识，可以保证在源领域和目标领域上的翻译性能共同逐步上升<sup>[1002]</sup>。此外，还可以使用L2正则化和Dropout方法来缓解这个问题<sup>[1003]</sup>。

### 3. 推断阶段的领域适应

神经机器翻译中，领域适应的另一种典型思路是优化推断算法<sup>[1004]</sup>。不同领域的数据存在着共性，但是又有各自的风格，因此对于使用多领域数据训练出的模型，分情况进行推断可能会带来更好的效果，例如在统计机器翻译中对疑问句和陈述句分别使用两个模型进行推断可以使翻译效果更好<sup>[1005]</sup>。在神经机器翻译模型中可以采用集成推断（见第十四章）达到同样的效果，即把多个领域的模型融合为一个模型用于推断<sup>[1006]</sup>。集成推断方法的主要优势在于实现简单，多个领域的模型可以独立训练，使训练时间大大缩短。集成推断也可以结合加权的思想，对不同领域的句子，赋予每个模型不同的先验权重进行推断，来获得最佳的推断结果<sup>[1007]</sup>。此外，也可以在推断过程中融入语言模型<sup>[900, 927]</sup>或目标领域的罕见词<sup>[1008]</sup>。

## 16.6 小结及拓展阅读

低资源机器翻译是机器翻译大规模应用所面临的挑战之一，因此也备受关注。一方面，小样本学习技术的发展，使得研究人员可以有更多的手段对问题求解；另一方面，从多语言之间的联系出发，也可以进一步挖掘不同语言背后的知识，并应用于低资源机器翻译任务。本章从多个方面介绍了低资源机器翻译方法，并结合多语言、零资源翻译等问题给出了不同场景下解决问题的思路。除此之外，还有几方面工作值得进一步关注：

- 如何更高效地利用已有双语数据或单语数据进行数据增强始终是一个热点问

题。研究人员分别探索了源语言单语数据和目标语言单语数据的使用方法<sup>[885, 887, 1009]</sup>，以及如何对已有双语数据进行修改的问题<sup>[590, 876]</sup>。经过数据增强得到的伪数据的质量时好时坏，如何提高伪数据的质量，以及更好地利用伪数据进行训练也是十分重要的问题<sup>[1010, 1011, 1012, 1013, 1014]</sup>。此外，还有一些工作对数据增强技术进行了理论分析<sup>[1015, 1016]</sup>。

- 预训练模型也是自然语言处理的重要突破之一，也给低资源机器翻译提供了新的思路。除了基于语言模型或掩码语言模型的方法，也有很多新的架构和模型被提出，如排列语言模型、降噪自编码器等<sup>[917, 1017, 1018, 1019]</sup>。预训练技术也逐渐向多语言领域扩展<sup>[916, 991, 1020]</sup>，甚至不再只局限于文本任务<sup>[1021, 1022, 1023]</sup>。对于如何将预训练模型高效地应用到下游任务中，也进行了很多的经验性对比与分析<sup>[165, 1024, 1025]</sup>。
- 多任务学习是多语言翻译的一种典型方法。通过共享编码器模块或是注意力模块来进行一对多<sup>[928]</sup> 或多对一<sup>[561]</sup> 或多对多<sup>[958]</sup> 的学习，然而这些方法需要为每个翻译语言对设计单独的编码器和解码器，限制了其扩展性。为了解决以上问题，研究人员进一步探索了用于多语言翻译的单个机器翻译模型的方法，也就是本章提到的多语言单模型系统<sup>[929, 1026]</sup>。为了弥补多语言单模型系统中缺乏语言表示多样性的问题，可以重新组织多语言共享模块，设计特定任务相关模块<sup>[1027, 1028, 1029, 1030]</sup>；也可以将多语言单词编码和语言聚类分离，用一种多语言词典编码框架共享词汇级别的信息，有助于语言间的泛化<sup>[1031]</sup>；还可以将语言聚类为不同的组，并为每个聚类单独训练一个多语言模型<sup>[1032]</sup>。
- 零资源翻译也是近几年受到广泛关注的研究方向<sup>[1033, 1034]</sup>。在零资源翻译中，仅使用少量并行语料库（覆盖  $k$  个语言），一个模型就能在任何  $k(k-1)$  个语言对之间进行翻译<sup>[1035]</sup>。但是，零资源翻译的性能通常很不稳定并且明显落后于有监督的翻译方法。为了改善零资源翻译，可以开发新的跨语言正则化方法，例如对齐正则化方法<sup>[1036]</sup>，一致性正则化方法<sup>[1035]</sup>；也可以通过反向翻译或基于枢轴语言的翻译生成伪数据<sup>[1033, 1037, 1038]</sup>。





## 17. 多模态、多层次机器翻译

基于上下文的翻译是机器翻译的一个重要分支。传统方法中，机器翻译通常被定义为对一个句子进行翻译的任务。但是，现实中每句话往往不是独立出现的。比如，人们会使用语音进行表达，或者通过图片来传递信息，这些语音和图片内容都可以伴随着文字一起出现在翻译场景中。此外，句子往往存在于段落或者篇章之中，如果要理解这个句子，也需要整个段落或者篇章的信息，而这些上下文信息都是机器翻译可以利用的。

本章在句子级翻译的基础上将问题扩展为更大的上下文中的翻译，具体包括语音翻译、图像翻译、篇章翻译三个主题。这些问题均为机器翻译应用中的真实需求。同时，使用多模态等信息也是当下自然语言处理的热点研究方向之一。

### 17.1 机器翻译需要更多的上下文

长期以来，机器翻译都是指句子级翻译。主要原因在于，句子级的翻译建模可以大大简化问题，使得机器翻译方法更容易被实践和验证。但是人类使用语言的过程并不是孤立地在一个个句子上进行的。这个问题可以类比于人类学习语言的过程：小孩成长过程中会接受视觉、听觉、触觉等多种信号，这些信号的共同作用使得他们产生对客观世界的“认识”，同时促使他们使用“语言”进行表达。从这个角度说，语言能力并不是由单一因素形成的，它往往伴随着其他信息的相互作用，比如，当人们翻译一句话的时候，会用到看到的画面、听到的语调、甚至前面说过的句子中的信息。

广义上，当前句子以外的信息都可以被看作一种上下文。比如，图17.1中，需要把英语句子“*A girl jumps off a bank.*”翻译为汉语。但是，其中的“bank”有多个含义，因此仅仅使用英语句子本身的信息可能会将其翻译为“银行”，而非正确的译文“河床”。但是，图17.1中也提供了这个英语句子所对应的图片，显然图片中直接展示了河床，这时“bank”是没有歧义的。通常也会把这种使用图片和文字一起进行机器翻译的任务称作**多模态机器翻译**（Multi-Modal Machine Translation）。

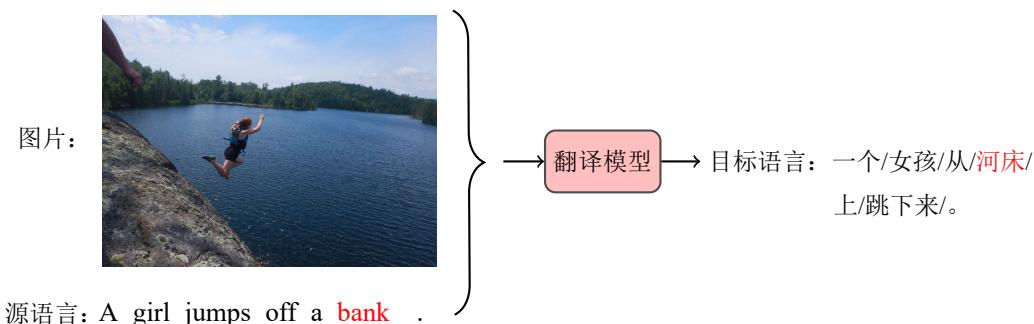


图 17.1 多模态机器翻译实例

**模态**（Modality）是指某一种信息来源。例如，视觉、听觉、嗅觉、味觉都可以被看作是不同的模态。因此视频、语音、文字等都可以被看作是承载这些模态的媒介。在机器翻译中使用多模态这个概念，是为了区分某些不同于文字的信息。除了图像等视觉模态信息，机器翻译也可以利用听觉模态信息。比如，直接对语音进行翻译，甚至直接用语音表达出翻译结果。

除了不同信息源所引入的上下文，机器翻译也可以利用文字本身的上下文。比如，翻译一篇文章中的某个句子时，可以根据整个篇章的内容进行翻译。显然这种篇章的语境是有助于机器翻译的。在本章接下来的内容中，会对机器翻译中使用不同上下文（多模态和篇章信息）的方法展开讨论。

## 17.2 语音翻译

语音，是人类交流中最常用的一种信息载体。从日常聊天、出国旅游，到国际会议、跨国合作，对于语音翻译的需求不断增加。甚至在有些场景下，用语音进行交互要比用文本进行交互频繁得多。因此，**语音翻译**（Speech Translation）也成为了语音处理和机器翻译相结合的重要产物。根据目标语言的载体类型，可以将语音翻译分为**语音到文本翻译**（Speech-to-Text Translation）和**语音到语音翻译**（Speech-to-Speech Translation）；基于翻译的实时性，还可以分为**实时语音翻译**（即同声传译，Simultaneous Translation）和**离线语音翻译**（Offline Speech Translation）。本节主要关注离线语音到文本翻译方法（简称为语音翻译），分别从音频处理、级联语音翻译和端到端语音翻译几个角度开展讨论。

### 17.2.1 音频处理

为了保证对相关内容描述的完整性，这里对语音处理的基本知识作简要介绍。不同于文本，音频本质上是经过若干信号处理之后的**波形**（Waveform）。具体来说，声音是一种空气的震动，因此可以被转换为模拟信号。模拟信号是一段连续的信号，经过采样变为离散的数字信号。采样是每隔固定的时间记录一下声音的振幅，采样率表示每秒的采样点数，单位是赫兹（Hz）。采样率越高，采样的结果与原始的语音越相像。通常来说，采样的标准是能够通过离散化的数字信号重现原始语音。日常生活中使用的手机和电脑设备的采样率一般为16kHz，表示每秒16000个采样点；而音频CD的采样率可以达到44.1kHz。经过进一步的量化，将采样点的值转换为整型数值保存，从而减少占用的存储空间，通常采用的是16位量化。将采样率和量化位数相乘，就可以得到**比特率**（Bits Per Second, BPS），表示音频每秒占用的位数。例如，16kHz采样率和16位量化的音频，比特率为256kb/s。音频处理的整体流程如图17.2所示<sup>[1039, 1040]</sup>。

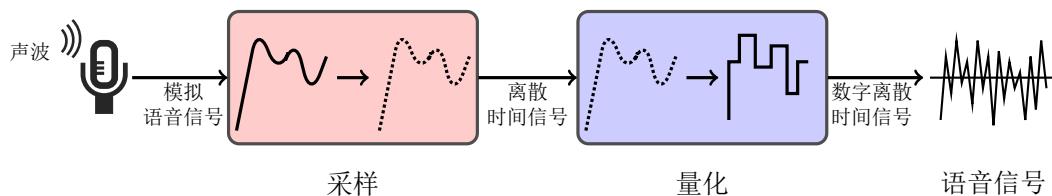


图 17.2 音频处理过程

经过上面的描述可以看出，音频的表示实际上是一个非常长的采样点序列，这导致了直接使用现有的深度学习技术处理音频序列较为困难。并且，原始的音频信号中可能包含着较多的噪声、环境声或冗余信息，也会对模型产生干扰。因此，一般会对音频序列进行处理来提取声学特征，具体为将长序列的采样点序列转换为短序列的特征向量序列，再用于下游系统。虽然已有一些工作不依赖特征提取，直接在原始的采样点序列上进行声学建模和模型训练<sup>[1041]</sup>，但目前的主流方法仍然是基于声学特征进行建模<sup>[1042]</sup>。

声学特征提取的第一步是预处理。其流程主要是对音频进行**预加重**（Pre-emphasis）、**分帧**（Framing）和**加窗**（Windowing）。预加重是通过增强音频信号中的高频部分来减弱语音中对高频信号的抑制，使频谱更加顺滑。分帧（原理如图17.3所示）是基于**短时平稳假设**，即根据生物学特征，语音信号是一个缓慢变化的过程，10ms~30ms的信号片段是相对平稳的。基于这个假设，一般将每25ms作为一帧来提取特征，这个时间称为**帧长**（Frame Length）。同时，为了保证不同帧之间的信号平滑性，使每两个相邻帧之间存在一定的重合部分。一般每隔10ms取一帧，这个时长称为**帧移**（Frame Shift）。为了缓解分帧带来的频谱泄漏问题，需要对每帧的信号进行加窗处理使其幅度在两段渐变到0，一般采用的是**汉明窗**（Hamming）<sup>[1039]</sup>。

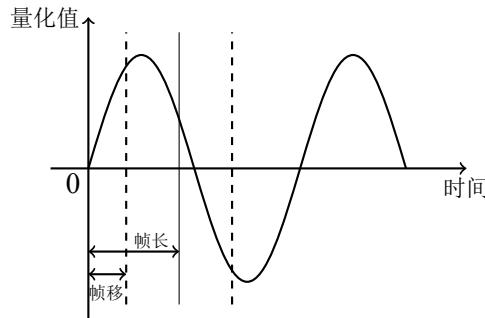


图 17.3 分帧原理图

经过了上述的预处理操作，可以得到音频对应的帧序列，之后通过不同的操作来提取不同类型的声学特征。在语音翻译中，比较常用的声学特征为**滤波器组**（Filterbank, Fbank）和**Mel 频率倒谱系数**（Mel-frequency Cepstral Coefficient, MFCC）<sup>[1039]</sup>。实际上，提取到的声学特征可以类比于计算机视觉中的像素特征，或者自然语言处理中的词嵌入表示。不同之处在于，声学特征更加复杂多变，可能存在较多的噪声和冗余信息。此外，相比对应的文字序列，音频提取到的特征序列长度要大十倍以上。比如，人类正常交流中每秒钟一般可以说 2-3 个字，而每秒钟的语音可以提取得到 100 帧的特征序列。巨大的长度比差异也为声学特征建模带来了挑战。

## 17.2.2 级联式语音翻译

实现语音翻译最简单的思路是基于级联的方式，即：先通过**自动语音识别**（Automatic Speech Recognition, ASR）系统将语音转化为源语言文本，然后利用机器翻译系统将源语言文本翻译为目标语言文本。这种做法的好处在于语音识别和机器翻译模型可以分别进行训练，有很多数据资源以及成熟技术可以分别运用到两个系统中。因此，级联语音翻译是很长时间以来的主流方法，深受工业界的青睐。级联语音翻译主要的流程如图17.4所示。

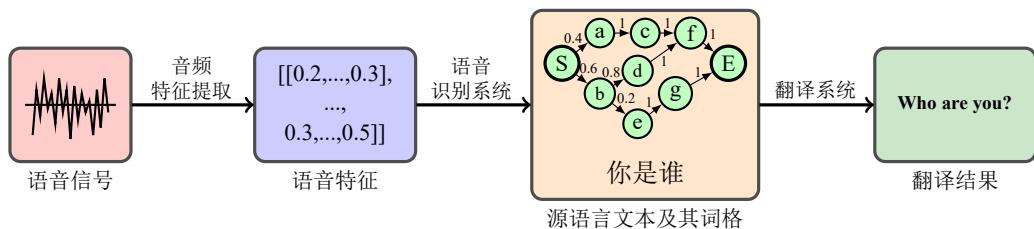


图 17.4 级联式语音翻译流程示例

由于声学特征提取在上一节中已经进行了描述，而且文本翻译可以直接使用本书介绍的统计机器翻译或者神经机器翻译方法。因此下面简要介绍一下语音识别模型，以便读者对级联式语音翻译系统有一个完整的认识。其中的部分概念在后续介绍的端到端语言翻译中也会有所涉及。

传统的语音识别模型和统计机器翻译相似，需要利用声学模型、语言模型和发音词典联合进行识别，系统较为复杂<sup>[1043, 1044, 1045]</sup>。而近些年来，随着神经网络的发展，基于神经网络的端到端语音识别模型逐渐受到关注，训练流程也大大被简化<sup>[1046, 1047]</sup>。目前的端到端语音识别模型主要基于序列到序列结构，编码器根据输入的声学特征进一步提取高级特征，解码器根据编码器提取的特征识别对应的文本。在后文中即将介绍的端到端语音翻译模型也是基于十分相似的结构。因此，从某种意义上说，语音识别和翻译所使用的端到端方法与神经机器翻译是一致的。

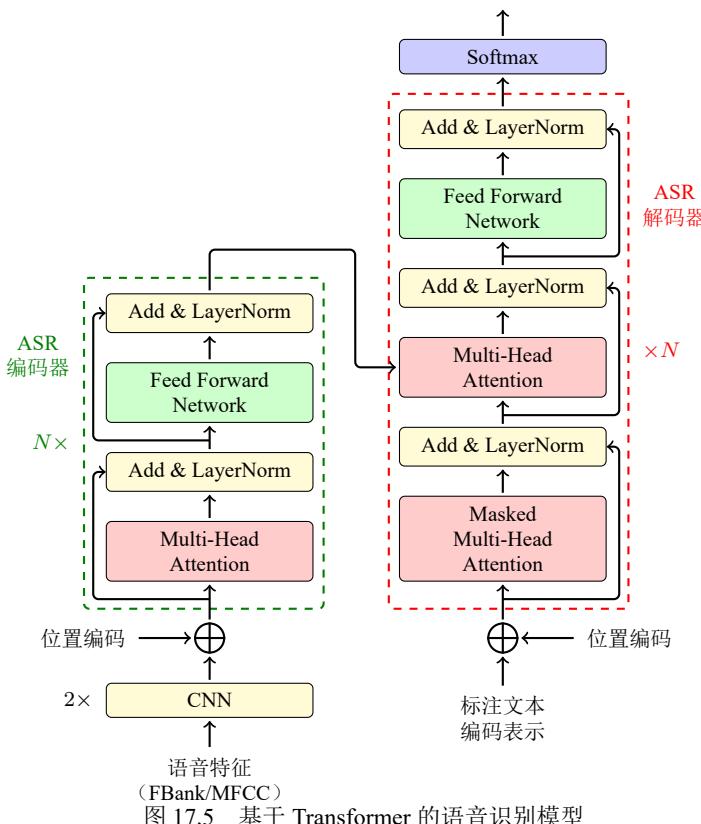


图 17.5 基于 Transformer 的语音识别模型

语音识别目前广泛使用基于 Transformer 的模型结构（见第十二章），如图17.5所示。可以看出，相比文本翻译，模型结构上唯一的区别在于编码器的输入为声学特征，以及编码器底层会使用额外的卷积层来减小输入序列的长度。这是由于语音对应的特征序列过长，在计算注意力模型的时候，会占用大量的内存/显存，并增加训练时间。因此，一个常用的做法是在语音特征上进行两层步长为 2 的卷积操作，从而将输入序列的长度缩小为之前的 1/4。通过使用大量的语音-标注平行数据对模型进行训练，可以得到高质量的语音识别模型。

为了降低语音识别的错误对下游系统的影响，通常也会用词格来取代 One-best 语音识别结果。另一种思路是通过一个后处理模型修正识别结果中的错误，再送给文本翻译模型进行翻译。也可以进一步对文本做**顺滑** (Disfluency Detection) 处理，使得送给翻译系统的文本更加干净、流畅，比如除去一些导致停顿的语气词。这一做

法在工业界得到了广泛应用，但由于每个模型只能串行地计算，也会带来额外的计算代价以及运算时间。另外一种思路是训练更加健壮的文本翻译模型，使其可以处理输入中存在的噪声或误差<sup>[592]</sup>。

### 17.2.3 端到端语音翻译

级联语音翻译模型结构简单、易于实现，但不可避免地存在一些缺陷：

- **错误传播问题**。级联模型导致的一个很严重的问题在于，语音识别模型得到的文本如果存在错误，这些错误很可能在翻译过程中被放大，从而使最后翻译结果出现比较大的偏差。比如识别时在句尾少生成了个“吗”，会导致翻译模型将疑问句翻译为陈述句。
- **翻译效率问题**。由于语音识别模型和文本标注模型只能串行地计算，翻译效率相对较低，而实际很多场景中都需要实现低延时的翻译。
- **语音中的副语言信息丢失**。将语音识别为文本的过程中，语音中包含的语气、情感、音调等信息会丢失，而同一句话在不同的语气中表达的意思很可能是不同的。尤其是在实际应用中，由于语音识别结果通常并不包含标点，还需要额外的后处理模型将**标点还原**，也会带来额外的计算代价。

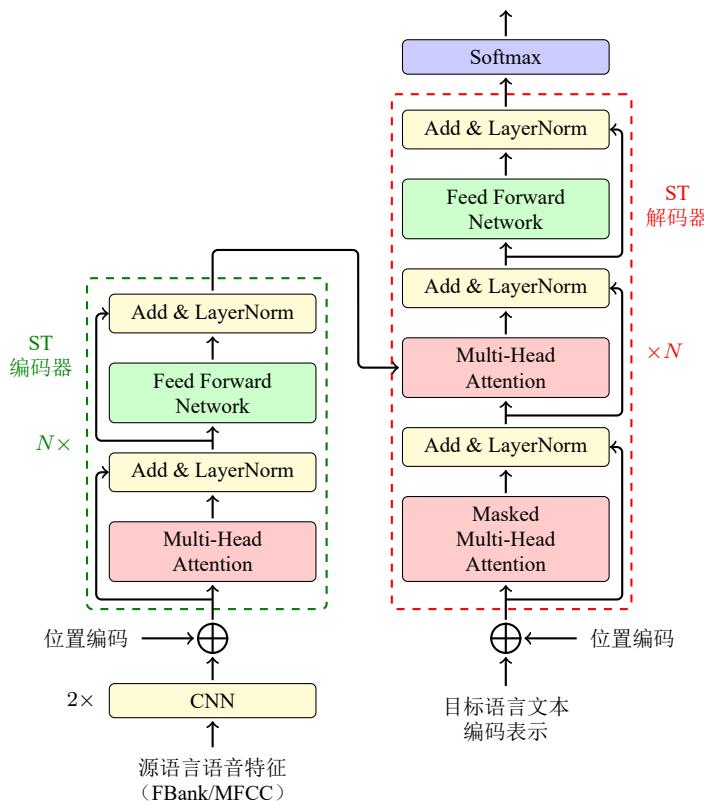


图 17.6 基于 Transformer 的端到端语音翻译模型

针对级联语音翻译模型存在的缺陷，研究人员提出了**端到端的语音翻译模型**（End-to-End Speech Translation, E2E-ST）<sup>[1048, 1049, 1050]</sup>，也就是模型的输入是源语言语音，输出是对应的目标语言文本。相比级联模型，端到端模型有如下优点：

- 端到端模型不需要多阶段的处理，因此避免了错误传播问题。
- 同样地，端到端模型所涉及的模块更少，容易控制模型体积。
- 端到端模型语音信号可以直接作用于翻译过程，因此可以使得副语言信息得以体现。

以Transformer模型为例，图17.6展示了端到端语音翻译的架构（下文中语音翻译模型均指端到端的模型）。该模型采用的也是序列到序列架构，编码器的输入是从语音中提取的特征（比如FBank特征）。编码器底层采用和语音识别模型相同的卷积结构来降低序列的长度（见17.2.2节）。之后的流程和标准的神经机器翻译是完全一致的，编码器对语音特征进行编码，解码器根据编码结果生成目标语言的翻译结果。

虽然端到端语音翻译模型解决了级联模型存在的问题，但同时也面临着两个严峻的问题：

- **训练数据稀缺**。虽然语音识别和文本翻译的训练数据都很多，但是直接由源语言语音到目标语言文本的平行数据十分有限，因此端到端语音翻译天然地就是一种低资源翻译任务。
- **建模复杂度更高**。在语音识别中，模型是学习如何生成语音对应的文字序列，输入和输出的对齐比较简单，不涉及到调序的问题。在文本翻译中，模型要学习如何生成源语言序列对应的目标语言序列，仅需要学习不同语言之间的映射，不涉及到模态的转换。而语音翻译模型需要学习从语音到目标语言文本的生成，任务更加复杂。

针对这两个问题，研究人们也提出了很多方法进行缓解，包括多任务学习、迁移学习等，主要思想都是利用语音识别或文本翻译数据来指导模型的学习。并且，文本翻译的很多方法对语音翻译技术的发展提供了思路。如何将其他领域现有的工作在语音翻译任务上验证，也是语音翻译研究人员当前所关注的<sup>[1051]</sup>。

## 1. 多任务学习

一种思路是进行多任务学习，让模型在训练过程中得到更多的监督信息。使用多个任务强化主任务（机器翻译），在本书的第十五章和第十六章也有所涉及。从这个角度说，机器翻译中很多问题的解决手段都是一致的。

语音翻译中多任务学习主要借助语音对应的标注信息，也就是源语言文本。**连接时序分类**（Connectionist Temporal Classification, CTC）<sup>[1052]</sup>是语音处理中最简单有效的一种多任务学习方法<sup>[1053, 1054]</sup>，被广泛应用于文本识别任务中<sup>[1055]</sup>。CTC可以将输入序列的每一位置都对应到标注文本中，学习语音和文字之间的软对齐关系。如图17.7

，对于下面的音频序列，CTC 可以将每个位置分别对应到同一个词。需要注意的是，CTC 会额外新增一个词  $\epsilon$ ，类似于一个空白词，表示这个位置没有声音或者没有任何对应的预测结果。在对齐完成之后，将相同且连续的词合并，去除  $\epsilon$ ，就可以得到预测结果。

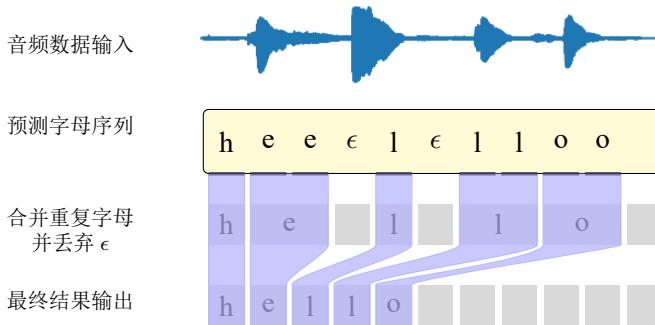


图 17.7 CTC 预测单词序列示例

CTC 的一些特性使其可以很好的完成输入输出之间的对齐，例如：

- **输入和输出之间的对齐是单调的。**对于音频输入序列  $\{s_1, \dots, s_m\}$ ，其对应的预测输出序列为  $\{x_1, \dots, x_n\}$ 。假设  $s_i$  对应的预测输出结果为  $x_j$ ，那么  $s_{i+1}$  相对对应的预测结果只能是  $x_j$ 、 $x_{j+1}$  和  $\epsilon$  三者中的一个。例如对于图17.7中的例子，如果输入的位置  $s_i$  已经对齐了字符“e”，那么  $s_{i+1}$  的对齐结果只能是“e”、“1”和  $\epsilon$  三者中的一个。
- **输入和输出之间是多对一的关系。**也就是多个输入会对应到同一个输出上。这对于语音序列来说是非常自然的一件事情，由于输入的每个位置只包含非常短的语音特征，因此多个输入才可以对应到一个输出字符。

将 CTC 应用到语音翻译中的方法非常简单，只需要在编码器的顶层加上一个额外的输出层即可（图17.8）。通过这种方式，不需要增加过多的参数，就可以给模型加入一个较强的监督信息。

另外一种多任务学习的思想是通过两个解码器，分别预测语音对应的源语言句子和目标语言句子，具体有图17.9展示的三种方式<sup>[1056, 1057]</sup>。图17.9(a) 中采用单编码器-双解码器的方式，两个解码器根据编码器的表示，分别预测源语言句子和目标语言句子，从而使编码器训练地更加充分。这种做法的好处在于源语言的文本生成任务可以辅助翻译过程，相当于为源语言语音提供了额外的“模态”信息。图17.9(b) 则通过使用两个级联的解码器，先利用第一个解码器生成源语言句子，然后再利用第一个解码器的表示，通过第二个解码器生成目标语言句子。这种方法通过增加一个中间输出，降低了模型的训练难度，但同时也会带来额外的解码耗时，因为两个解码器需要串行地进行生成。图17.9(c) 中模型更进一步，第二个编码器联合编码器和第一个解码器的表示进行生成，更充分地利用了已有信息。

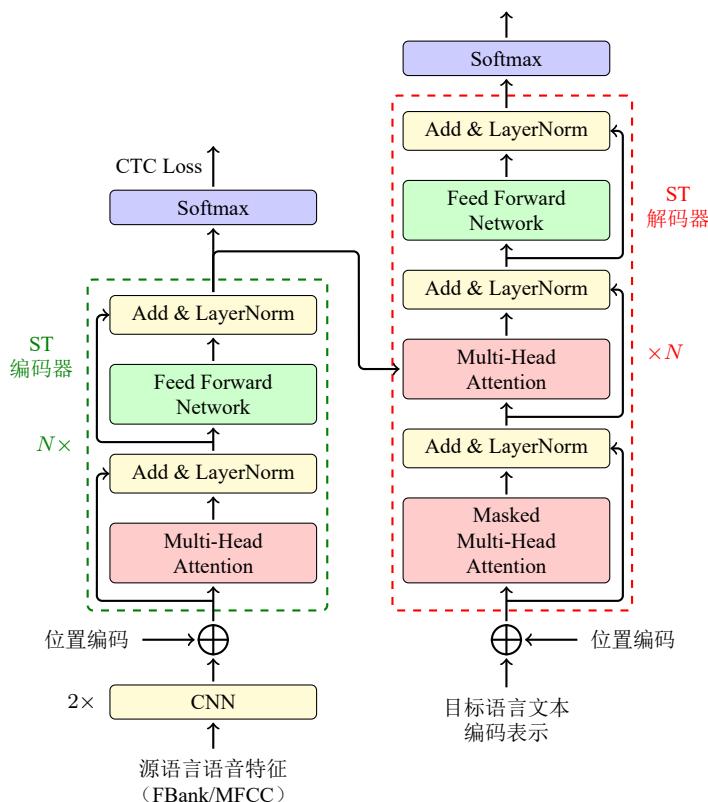


图 17.8 基于 CTC 的语音翻译模型

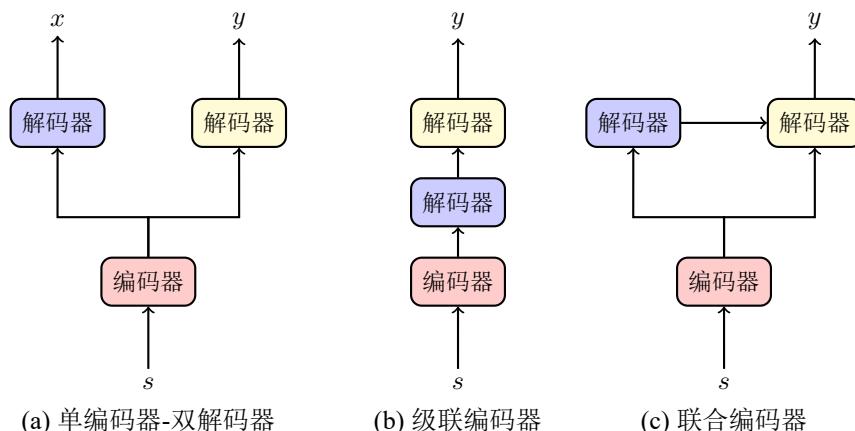
 $x$ : 源语言文本数据 $y$ : 目标语言文本数据 $s$ : 源语言语音数据

图 17.9 双解码器进行语音翻译的三种方式

## 2. 迁移学习

相比语音识别和文本翻译，端到端语音翻译的训练数据量要小很多，因此，如何利用其它数据来增加可用的数据量是语音翻译的一个重要方向。和文本翻译中的方法相似，一种思路是利用迁移学习或预训练，利用其他语言的双语数据预训练模型参数，然后迁移到生成目标语言的任务上<sup>[1058]</sup>，或者是利用语音识别数据或文本翻译数据，分别预训练编码器和解码器的参数，用于初始化语音翻译模型的参数<sup>[1059]</sup>。预训练的编码器对语音翻译模型的学习尤为重要<sup>[1058]</sup>，相比文本数据，语音数据的复杂性更高，仅使用小规模语音翻译数据很难学习充分。此外，模型对声学特征的学习与语言并不是强相关的，其他语种预训练的编码器对模型学习也是有帮助的。

## 3. 数据增强

数据增强是增加训练数据最直接的一种方法。不同于文本翻译的回译等方法（见第十六章），语音翻译并不具有简单的“可逆性”。如果要利用回译的思想，需要通过一个模型，将目标语言文本转化为源语言语音，但实际上这种模型是不能简单得到的。因此，一个简单的思路是通过一个反向翻译模型和语音合成模型级联来生成伪数据<sup>[1060]</sup>。另外，正向翻译模型生成的伪数据在文本翻译中也被验证了对模型训练有一定的帮助，因此同样可以利用语音识别和文本翻译模型，将源语言语音翻译成目标语言文本，得到伪平行语料。

此外，也可以利用在海量的无标注语音数据上预训练的**自监督**（Self-supervised）模型作为一个特征提取器，将从语音中提取的特征作为语音翻译模型的输入，可以有效提高模型的性能<sup>[1061]</sup>。相比语音翻译模型，文本翻译模型任务更加简单，因此一种思想是利用文本翻译模型来指导语音翻译模型，比如，使用知识蒸馏<sup>[1062]</sup>、正则化<sup>[1063]</sup>等方法。为了简化语音翻译模型的学习，也可以使用课程学习方法（见第十三章）。这样，使模型从语音识别任务，逐渐过渡到语音翻译任务，这种由易到难的训练策略可以使模型训练更加充分<sup>[1064, 1065]</sup>。

### 17.3 图像翻译

在人类所接受的信息中，视觉信息的比重往往不亚于语音和文本信息，甚至更多。视觉信息通常以图像的形式存在，近几年，结合图像的多模态机器翻译受到了广泛的关注。多模态机器翻译（图17.10 (a)）简单来说就是结合源语言和其他模态（例如图像等）的信息生成目标语言的过程。这种结合图像的机器翻译还是一种狭义上的“翻译”，它本质上还是从源语言到目标语言或者说从文本到文本的翻译。事实上从图像到文本（图17.10(b)）的转换，即给定图像，生成与图像内容相关的描述，也可以被称为广义上的“翻译”。例如，**图片描述生成**（Image Captioning）就是一种典型的图像到文本的翻译。当然，这种广义上的翻译形式不仅仅包括图像到文本的转换，还可以包括从图像到图像的转换（图17.10(c)），甚至是文本到图像的转换（图17.10(d)）等等。这里将这些与图像相关的翻译任务统称为图像翻译。

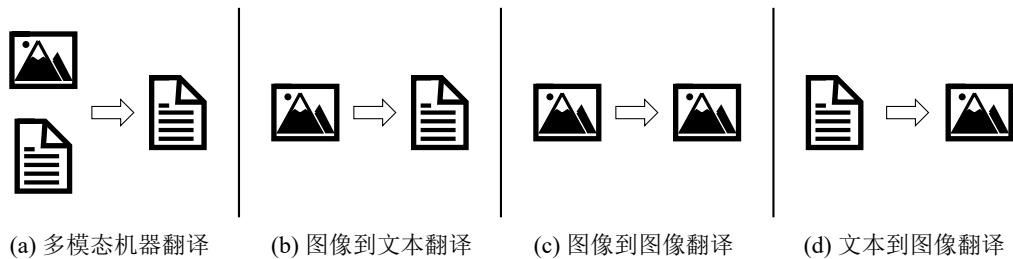


图 17.10 图像翻译任务

### 17.3.1 基于图像增强的文本翻译

在文本翻译中引入图像信息是最典型的多模态机器翻译任务。虽然多模态机器翻译还是一种从源语言文本到目标语言文本的转换，但是在转换的过程中，融入了其他模态的信息减少了歧义的产生。例如前文提到的通过与源语言相关的图像信息，将“*A girl jumps off a bank.*”中“bank”翻译为“河岸”而不是“银行”，因为图像中出现了河岸，因此“bank”的歧义大大降低。换句话说，对于同一图像或者视觉场景的描述，源语言和目标语言描述的信息是一致的，只不过，体现在不同语言上会有表达方法上的差异。那么，图像就会存在一些源语言和目标语言的隐含对齐“约束”，而这种“约束”可以捕捉语言中不易表达的隐含信息。

如何融入视觉信息，更好的理解多模态上下文语义是多模态机器翻译研究的重点<sup>[1066, 1067, 1068]</sup>，主要方向包括基于特征融合的方法<sup>[1069, 1070, 1071]</sup>、基于联合模型的方法<sup>[1072, 1073]</sup>。下面是具体介绍。

#### 1. 基于特征融合的方法

早期，通常将图像信息作为输入句子的一部分<sup>[1069, 1074]</sup>，或者用其对编码器、解码器的状态进行初始化<sup>[1069, 1075, 1076]</sup>。如图17.11所示，图中  $y_{<}$  表示当前时刻之前的单词序列，对图像特征的提取通常是基于卷积神经网络，有关卷积神经网络的内容，可以参考第十一章内容。通过卷积神经网络得到全局图像特征，在进行维度变换后，将其作为源语言输入的一部分或者初始化状态引入到模型当中。但是，这种图像信息的引入方式有以下两个缺点：

- 图像信息不全都是有用，往往存在一些与源语言或目标语言无关的信息，作为全局特征会引入噪音。
- 图像信息作为源语言的一部分或者初始化状态，间接地参与了译文的生成，在神经网络的计算过程中，图像信息会有一定的损失。

说到噪音问题就不得不提到注意力机制的引入，前面章节中提到过这样的一个例子：

中午/没/吃饭/， /又/刚/打/了/ 一/下午/篮球/， /我/现在/很/饿/， /我/想\_\_\_\_\_。

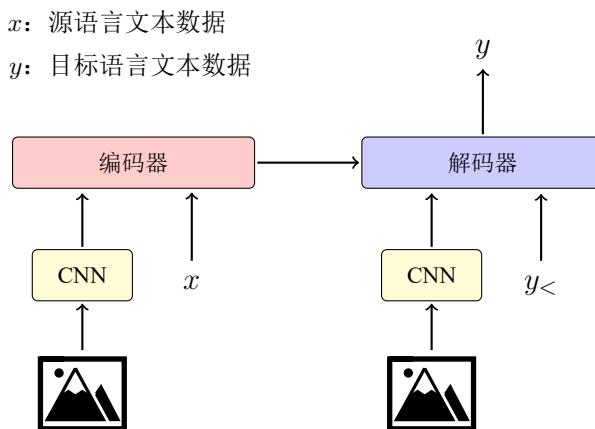


图 17.11 基于全局视觉特征的多模态翻译方法

想在横线处填写“吃饭”，“吃东西”的原因是在读句子的过程中，关注到了“没/吃饭”，“很/饿”等关键信息。这是在语言生成中注意力机制所解决的问题，即对于要生成的目标语言单词，相关性更高的语言片段应该更加“重要”，而不是将所有单词一视同仁。同样的，注意力机制也应用在多模态机器翻译中，即在生成目标单词时，更应该关注与目标单词相关的图像部分，而弱化对其他部分的关注。另外，注意力机制的引入，也使图像信息更加直接地参与目标语言的生成，解决了在不使用注意力机制的方法中图像信息传递损失的问题。



图 17.12 使用注意力机制前后图像中对单词“bank”的关注度对比

那么，多模态机器翻译是如何计算上下文向量的呢？这里仿照第十章的内容给出描述。假设编码器输出的状态序列为  $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$ ，需要注意的是，这里的状态序列不是源语言句子的状态序列，而是通过基于卷积等操作提取到的图像的状态序列。假设图像的特征维度是  $16 \times 16 \times 512$ ，其中前两个维度分别表示图像的高和宽，这里会将图像映射为  $256 \times 512$  的状态序列，其中 512 为每个状态的维度。对于目标语言

位置  $j$ , 上下文向量  $\mathbf{C}_j$  被定义为对序列的编码器输出进行加权求和, 如下:

$$\mathbf{C}_j = \sum_i \alpha_{i,j} \mathbf{h}_i \quad (17.1)$$

其中,  $\alpha_{i,j}$  是注意力权重, 它表示目标语言第  $j$  个位置与图片编码状态序列第  $i$  个位置的相关性大小, 计算方式与第十章描述的注意力函数一致。

这里, 将  $\mathbf{h}_i$  看作图像表示序列位置  $i$  上的表示结果。图17.12给出了模型在生成目标词“bank”时, 图像经过注意力机制对图像区域关注度的可视化效果。可以看到, 经过注意力机制后, 模型更关注与目标词相关的图像部分。当然, 多模态机器翻译的输入还包括源语言文字序列。通常, 源语言文字对于翻译的作用比图像更大<sup>[1077]</sup>。从这个角度说, 在当下的多模态翻译任务中, 图像信息更多的是作为文字信息的补充, 而不是替代。除此之外, 注意力机制在多模态机器翻译中也有很多研究, 比如, 在编码器端将源语言文本与图像信息进行注意力建模, 得到更好的源语言的表示结果<sup>[1070, 1077]</sup>。

## 2. 基于联合模型的方法

基于联合模型的方法通常是把翻译任务与其他视觉任务结合, 进行联合训练。这种方法也可以被看做是一种多任务学习, 只不过这里仅关注翻译和视觉任务。一种常见的方法是共享模型的部分参数来学习不同任务之间相似的部分, 并通过特定的模块来学习每个任务特有的部分。

如图17.13所示, 图中  $y_{<}$  表示当前时刻之前的单词序列, 可以将多模态机器翻译任务分解为两个子任务: 机器翻译和图片生成<sup>[1072]</sup>。其中机器翻译作为主任务, 图片生成作为子任务。这里的图片生成指的是从一个图片描述生成对应图片, 对于图片生成任务在后面还会有描述。通过单个编码器对源语言数据进行建模, 然后通过两个解码器(翻译解码器和图像解码器)来分别学习翻译任务和图像生成任务。顶层学习每个任务的独立特征, 底层共享参数能够学习到更丰富的文本表示。

另外在视觉问答领域有研究表明, 在多模态任务中, 不宜引入过多层的注意力机制, 因为过深的模型会导致多模态模型的过拟合<sup>[1078]</sup>。这一方面是由于深层模型本身对数据的拟合能力较强, 另一方面也是由于多模态任务的数据普遍较少, 容易造成复杂模型的过拟合。从另一角度来说, 利用多任务学习的方式, 提高模型的泛化能力, 也是一种有效防止过拟合现象的方式。类似的思想, 也大量使用在多模态自然语言处理任务中, 例如图像描述生成、视觉问答等<sup>[1079]</sup>。

### 17.3.2 图像到文本的翻译

图像到文本的转换也可以看作是广义上的翻译, 简单来说, 就是把图像作为唯一的输入, 而输出是文本。其中, 图像描述生成是最典型的图像到文本的翻译任务<sup>[1080]</sup>。虽然, 这部分内容并不是本书的重点, 不过为了保证多模态翻译内容的完整性, 这

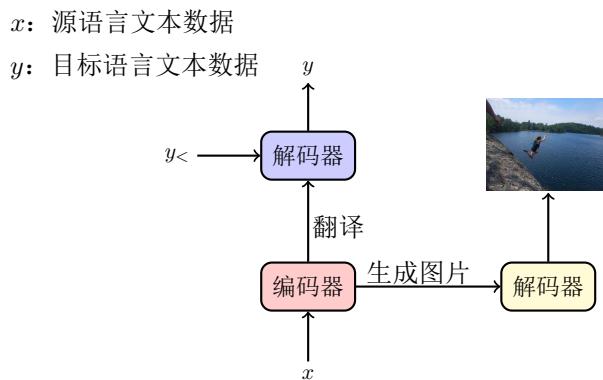


图 17.13 翻译 + 图片生成联合学习模型

里对相关技术进行简要介绍。图像描述有时也被称看图说话、图像字幕生成，它在图像检索、智能导盲、人机交互等领域有着广泛的应用场景。

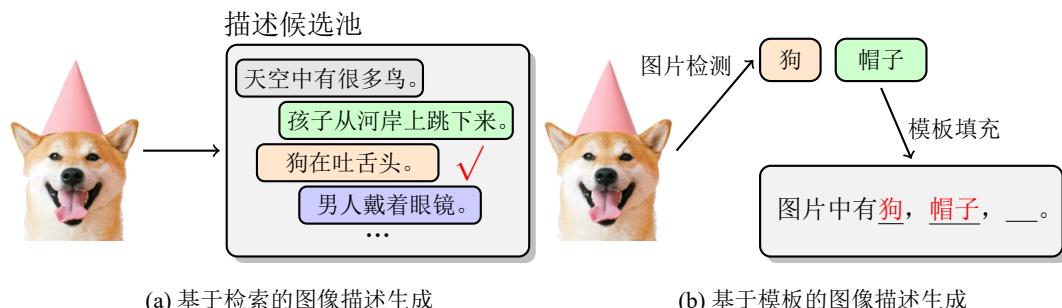


图 17.14 图像描述传统方法

传统图像描述生成有两种范式：基于检索的方法和基于模板的方法。其中图17.14(a)展示了一个基于检索的图像描述生成实例，这种方法在图像描述的候选中选择一个描述输出。但是，弊端是所选择的句子可能会和图像很大程度上不相符。而17.14(b)展示的是一种基于模版的方法，这种方法需要在图像上提取视觉特征，然后把内容填在实现设计好的模版当中，这种方法的缺点是生成的图像描述过于呆板，“像是在一个模子中刻出来的”说的就是这个意思。近几年来，受到机器翻译领域等任务的启发，图像描述生成任务也开始大量使用编码器-解码器框架。这里会从基础的图像描述范式编码器-解码器框架展开<sup>[1081, 1082]</sup>，并从编码器的改进和解码器的改进两个方面进行介绍。

## 1. 基础框架

在编码器-解码器框架中，编码器将输入的图像转换为一种新的“表示”形式，这种“表示”包含了输入图像的所有信息。之后解码器把这种“表示”转换为自然语

言描述。比如，可以通过卷积神经网络提取图像特征为一个向量表示。然后，利用长短时记忆网络（LSTMs）解码生成文字描述，这个过程中与机器翻译的解码过程类似。这种建模方式存在与17.3.1描述一样的问题：生成的描述单词不一定需要所有的图像信息，将全局的图像信息送入模型中，可能会引入噪音。这时可以使用注意力机制来缓解该问题<sup>[1082]</sup>。

## 2. 编码器的改进

要想使编码器-解码器框架在图像描述生成中充分发挥作用，编码器也要更好的表示图像信息。对于编码器的改进，通常体现在向编码器中添加图像的语义信息<sup>[1083, 1084, 1085]</sup>和位置信息<sup>[1084, 1086]</sup>。

图像的语义信息一般是指图像中存在的实体、属性、场景等等。如图17.15所示，从图像中利用属性或实体检测器提取出“jump”、“girl”、“river”、“bank”等属性词和实体词，将他们作为图像的语义信息编码的一部分，再利用注意力机制计算目标语言单词与这些属性词或实体词之间的注意力权重<sup>[1083]</sup>。当然，除了图像中的实体和属性作为语义信息外，也可以将图片的场景信息加入到编码器当中<sup>[1085]</sup>。有关如何做属性、实体和场景的检测，涉及到目标检测任务的工作，例如 Faster-RCNN<sup>[504]</sup>、YOLO<sup>[1087, 1088]</sup>等等，这里不再赘述。

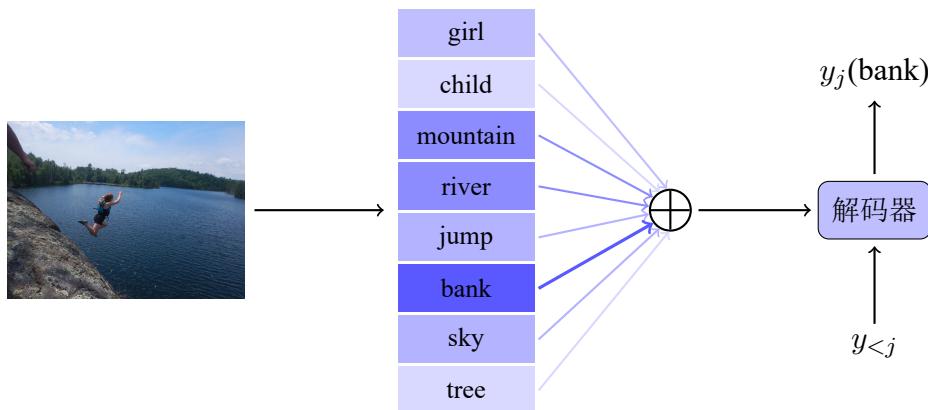


图 17.15 编码器“显式”融入语义信息

以上的方法大都是将图像中的实体、属性、场景等映射到文字上，并把这些信息显式地输入到编码器中。另一种方式，把图像中的语义特征隐式地引入编码器中<sup>[1084]</sup>。例如，图像数据可以分解为三个通道（红、绿、蓝），简单来说，就是将图像的每一个像素点按照红色、绿色、蓝色分成三个部分，这样就将图像分成了三个通道。在很多图像中，不同通道伴随的特征是不一样的，可以将其作用于编码器。另一种方法是基于位置信息的编码增强。位置信息指的是图像中对象（物体）的位置。利用目标检测技术检测系统获得图中的对象和对应的特征，这样就确定了图中的对象位置。显然，这些信息可以加强编码器的表示能力<sup>[1089]</sup>。

### 3. 解码器的改进

由于解码器输出的是语言文字序列，因此需要考虑语言的特点对其进行改进。例如，解码过程中，“the”，“on”，“at”这种介词或者冠词与图像的相关性较低<sup>[1090]</sup>。因此，可以通过门控单元，控制视觉信号作用于文字生成的程度。另外，在解码过程中，生成的每个单词对应着图像的区域可能是不同的。因此也可以设计更为有效的注意力机制来捕捉解码器端对不同图像局部信息的关注程度<sup>[1091]</sup>。

除了更好地使生成文本与图像特征进行相互作用以外，还有一些改进方法。例如，用卷积神经网络或者 Transformer 代替解码器所使用的循环神经网络<sup>[1092]</sup>。或者使用更深层的神经网络学习动词或者形容词等视觉中不易表现出来的单词<sup>[1093]</sup>，其思想与深层神经机器翻译模型有相通之处（见第十五章）。

#### 17.3.3 图像、文本到图像的翻译

当生成的目标对象是图像时，问题就变为了图像生成任务。虽然，这个领域本身并不属于机器翻译，但是其使用的基本方法与机器翻译有类似之处。二者也可以相互借鉴。

在计算机视觉中，图像风格变换、图像超分辨率重建等任务，都可以被视为**图像到图像的翻译**（Image-to-Image Translation）问题。与机器翻译类似，这些问题的共同目标是学习从一个对象到另一个对象的映射，只不过这里的对象是指图像，而非机器翻译中的文字。例如，给定物体的轮廓生成真实物体图片，或者给定白天照片生成夜晚的照片等。图像到图像的翻译有广阔的应用场景，如图片补全、风格迁移等。**文本到图像的翻译**（Text-to-Image Translation）是指给定描述物体颜色和形状等细节的自然语言文字，生成对应的图像。该任务也可以看作是图像描述生成的逆任务。

无论是图像到图像的生成，还是文本到图像的生成，均可直接使用编码器-解码器框架进行实现。比如，在文本到图像生成中，可以使用机器翻译中的编码器对输入文本进行编码，之后用对抗生成网络将编码结果转化为图像<sup>[1094]</sup>。近些年，图像生成类任务也取得了很大的进展，这主要得益于生成对抗网络的使用<sup>[1095, 1096, 1097]</sup>。在第十三章已经介绍了生成对抗网络，而且图像生成也不是本书的重点，感兴趣的读者可以参考第十三章的内容或者自行查阅相关文献进行了解。

## 17.4 篇章级翻译

目前大多数机器翻译系统是句子级的。由于缺少了对篇章上下文信息的建模，在需要依赖上下文的翻译场景中，模型的翻译效果总是不尽人意。篇章级翻译的目的就是对篇章上下文信息进行建模，进而改善机器翻译在整个篇章上的翻译质量。篇章级翻译的概念在很早就已经出现<sup>[1098]</sup>，随着近几年神经机器翻译取得了巨大进展，篇章级神经机器翻译也成为了重要的方向<sup>[1099, 1100]</sup>。基于此，本节将对篇章级神经机器翻译的若干问题展开讨论。

### 17.4.1 篇章级翻译的挑战

“篇章”在这里是指一系列连续的段落或句子所构成的整体，其中各个句子间从形式和内容上都具有一定的连贯性和一致性<sup>[140]</sup>。这些联系主要体现在衔接以及连贯两个方面。其中衔接体现在显性的语言成分和结构上，包括篇章中句子间的语法和词汇的联系，而连贯体现在各个句子之间的逻辑和语义的联系上。因此，篇章级翻译就是要将这些上下文之间的联系考虑在内，从而生成比句子级翻译更连贯和准确的翻译结果。实例17.1就展示了一个使用篇章信息进行机器翻译的实例。

**实例 17.1** 上下文句子：我/上周/针对/这个/问题/做出/解释/并/咨询/了/他的/意见/。

待翻译句子：他/也/同意/我的/看法/。

句子级翻译结果：He also agrees with me .

篇章级翻译结果：And he agreed with me .

不过由于不同语言的特性多种多样，上下文信息在篇章级翻译中的作用也不尽相同。比如，在德语中名词是分词性的，因此在代词翻译的过程中需要根据其先行词的词性进行区分，而这种现象在其它不区分名词词性的语言中是不存在的。这意味着篇章级翻译在不同的语种中可能对应不同的上下文现象。

正是这种上下文现象的多样性，使评价篇章级翻译模型的性能变得相对困难。目前篇章级机器翻译主要针对一些常见的上下文现象进行优化，比如代词翻译、省略、连接和词汇衔接等，而第四章介绍的 BLEU 等通用自动评价指标通常对这些上下文依赖现象不敏感，因此篇章级翻译需要采用一些专用方法来对这些具体现象进行评价。

在统计机器翻译时代就已经有大量的研究工作专注于篇章信息的建模，这些工作大多针对某一具体的上下文现象，比如，篇章结构<sup>[1101, 1102, 1103]</sup>、代词回指<sup>[1104, 1105, 1106]</sup>、词汇衔接<sup>[1107, 1108, 1109, 1110]</sup>和篇章连接词<sup>[1111, 1112]</sup>等。区别于篇章级统计机器翻译，篇章级神经机器翻译不需要针对某一具体的上下文现象构造相应的特征，而是通过翻译模型本身从上下文句子中抽取和融合的上下文信息。通常情况下，篇章级机器翻译可以采用局部建模的手段将前一句或者周围几句作为上下文送入模型。针对需要长距离上下文的情况，也可以使用全局建模的手段直接从篇章的所有句子中提取上下文信息。近几年多数研究工作都在探索更有效的局部建模或全局建模方法，主要包括改进输入<sup>[1113, 1114, 1115, 1116]</sup>、多编码器结构<sup>[488, 1117, 1118]</sup>、层次结构<sup>[1119, 1120, 1121, 1122]</sup>以及基于缓存的方法<sup>[1123, 1124]</sup>等。

此外，篇章级机器翻译面临的另外一个挑战是数据稀缺。篇章级机器翻译所需要的双语数据需要保留篇章边界，数量相比于句子级双语数据要少很多。除了在之前提到的端到端方法中采用预训练或者参数共享的手段（见第十六章），也可以采用新的建模手段来缓解数据稀缺问题。这类方法通常将篇章级翻译流程进行分离：先训练一个句子级的翻译模型，再通过一些额外的模块来引入上下文信息。比如，在句子级翻译模型的推断过程中，通过在目标端结合篇章级语言模型引入上下文信息<sup>[1125, 1126, 1127]</sup>，或者基于句子级的翻译结果，使用两阶段解码等手段引入上下文信息，进而对句子

级翻译结果进行修正<sup>[1128, 1129, 1130]</sup>。

### 17.4.2 篇章级翻译的评价

BLEU 等自动评价指标能够在一定程度上反映译文的整体质量，但是并不能有效地评估篇章级翻译模型的性能。这是由于很多标准测试集中需要篇章上下文的情况比例相对较少。而且，*n*-gram 的匹配很难检测到一些具体的语言现象，这使得研究人员很难通过 BLEU 得分来判断篇章级翻译模型的效果。

为此，研究人员总结了机器翻译任务中存在的上下文现象，并基于此设计了相应的自动评价指标。比如针对篇章中代词的翻译问题，首先借助词对齐工具确定源语言中的代词在译文和参考答案中的对应位置，然后通过计算译文中代词的准确率和召回率等指标对代词翻译质量进行评价<sup>[1104, 1131]</sup>。针对篇章中的词汇衔接，使用**词汇链** (Lexical Chain)<sup>1</sup>等来获取能够反映词汇衔接质量的分数，然后通过加权的方式与常规的 BLEU 或 METEOR 等指标结合在一起<sup>[1132, 1133]</sup>。针对篇章中的连接词，使用候选词典和词对齐工具对源文中连接词的正确翻译结果进行计数，计算其准确率<sup>[1134]</sup>。

除了直接对译文打分，也有一些工作针对特有的上下文现象手工构造了相应的测试套件用于评价翻译质量。测试套件中每一个测试样例都包含一个正确翻译的结果，以及多个错误结果，一个理想的翻译模型应该对正确的翻译结果评价最高，排名在所有错误结果之上，此时就可以根据模型是否能挑选出正确翻译结果来评估其性能。这种方法可以很好地衡量翻译模型在某一特定上下文现象上的处理能力，比如词义消歧<sup>[1135]</sup>、代词翻译<sup>[1114, 1136]</sup>和一些衔接问题<sup>[1129]</sup>等。但是该方法也存在使用范围受限于测试集的语种和规模的缺点，因此扩展性较差。

### 17.4.3 篇章级翻译的建模

在理想情况下，篇章级翻译应该以整个篇章为单位作为模型的输入和输出。然而由于现实中篇章对应的序列过长，因此直接建模整个篇章序列难度很大，这使得主流的序列到序列模型很难直接使用。一种思路是采用能够处理超长序列的模型对篇章序列建模，比如，使用第十五章中提到的处理长序列的 Transformer 模型就是一种的解决方法<sup>[543]</sup>。不过，这类模型并不针对篇章级翻译的具体问题，因此并不是篇章级翻译中的主流方法。

现在常见的端到端做法还是从句子级翻译出发，通过额外的模块来对篇章中的上下文句子进行表示，然后提取相应的上下文信息并融入到当前句子的翻译过程中。形式上，篇章级翻译的建模方式如下：

$$P(Y|X) = \prod_{i=1}^T P(Y_i|X_i, D_i) \quad (17.2)$$

其中，*X* 和 *Y* 分别为源语言篇章和目标语言篇章，*X<sub>i</sub>* 和 *Y<sub>i</sub>* 分别为源语言篇章和目

<sup>1</sup>词汇链指篇章中语义相关的词所构成的序列。

标语言篇章中的第  $i$  个句子， $T$  表示篇章中句子的数目。为了简化问题，这里假设源语言和目标语言具有相同的句子数目  $T$ ，而且两个篇章间句子是顺序对应的。 $D_i$  表示翻译第  $i$  个句子时所对应的上下文句子集合，理想情况下， $D_i$  中包含源语言篇章和目标语言篇章中所有除第  $i$  句之外的句子，但实践中通常仅使用其中的部分句子作为上下文。

上下文范围的选取是篇章级神经机器翻译需要着重考虑的问题，比如上下文句子的多少<sup>[487, 1119, 1137]</sup>，是否考虑目标端上下文句子<sup>[1113, 1137]</sup>等。此外，不同的上下文范围也对应着不同的建模方法，接下来将对一些典型的方法进行介绍，包括改进输入<sup>[1113, 1114, 1115, 1116]</sup>、多编码器模型<sup>[488, 1117, 1118]</sup>、层次结构模型<sup>[487, 1138, 1139]</sup>以及基于缓存的方法<sup>[1123, 1124]</sup>。

## 1. 输入形式

一种简单的方法是直接复用传统的序列到序列模型，将篇章中待翻译句子与其上下文句子拼接后作为模型输入。如实例17.2所示，这种做法不需要改动模型结构，操作简单，适用于大多数神经机器翻译系统<sup>[1113, 1137, 1140]</sup>。但是由于过长的序列会导致模型难以训练，通常只会选取局部的上下文句子进行拼接，比如只拼接源语言端前一句或者周围几句<sup>[1113]</sup>。此外，也可以引入目标语言端的上下文<sup>[1114, 1137, 1140]</sup>，在解码时拼接目标语言端上下文和当前句同样会带来一定的性能提升。但是过大的窗口会造成推断速度的下降<sup>[1137]</sup>，因此通常只考虑前一个目标语言句子。

### 实例 17.2 传统模型训练输入：

源语言：你/看到/了/吗/？

目标语言：Do you see them ?

### 改进后模型训练输入：

源语言：他们/在/哪/? <sep> 你/看到/了/吗/？

目标语言：Do you see them ?

其他改进输入的做法相比于拼接的方法要复杂一些，首先需要对篇章进行处理，得到词汇链<sup>2</sup>或者篇章嵌入等信息<sup>[1115, 1116]</sup>，然后将这些信息与当前句子一起送入模型中。目前，这种预先提取篇章信息的方法是否适合机器翻译还有待论证。

## 2. 多编码器结构

另一种思路是对传统的编码器-解码器框架进行更改，引入额外的编码器来对上下文句子进行编码，该结构被称为多编码器结构<sup>[489, 1141]</sup>。这种结构最早被应用在基于循环神经网络的篇章级翻译模型中<sup>[1114, 1117, 1142, 1143]</sup>，后期证明在 Transformer 模型上同样适用<sup>[488, 1118]</sup>。图17.16展示了一个基于 Transformer 模型的多编码器结构，基于源语言当前待翻译句子的编码表示  $h$  和上下文句子的编码表示  $h^{\text{pre}}$ ，模型首先通过注意

<sup>2</sup>词汇链指篇章中语义相关的词所构成的序列。

力机制提取句子间上下文信息  $d$ :

$$d = \text{Attention}(\mathbf{h}, \mathbf{h}^{\text{pre}}, \mathbf{h}^{\text{pre}}) \quad (17.3)$$

其中,  $\mathbf{h}$  作为 Query (查询),  $\mathbf{h}^{\text{pre}}$  作为 Key (键) 和 Value (值)。然后通过门控机制将待翻译句子中每个位置的编码表示和该位置对应的上下文信息进行融合, 具体方式如下:

$$\lambda_t = \sigma(\mathbf{W}_\lambda[\mathbf{h}_t; d] + \mathbf{b}_\lambda) \quad (17.4)$$

$$\tilde{\mathbf{h}}_t = \lambda_t \mathbf{h}_t + (1 - \lambda_t) \mathbf{d}_t \quad (17.5)$$

其中,  $\tilde{\mathbf{h}}$  为融合了上下文信息的最终序列表示结果,  $\tilde{\mathbf{h}}_t$  为其第  $t$  个位置的表示。 $\mathbf{W}_\lambda$  和  $\mathbf{b}_\lambda$  为模型可学习的参数,  $\sigma$  为 Sigmoid 函数, 用来获取门控权值  $\lambda$ 。除了在编码端融合源语言上下文信息, 也可以直接用类似机制在解码器内完成源语言上下文信息的融合<sup>[1118]</sup>。

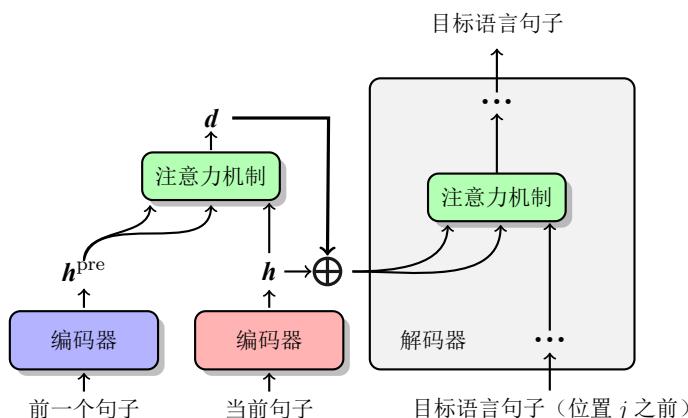


图 17.16 多编码器结构<sup>[489]</sup>

此外, 由于多编码器结构引入了额外的模块, 模型整体参数量大大增加, 同时增加了模型训练的难度。为此, 一些研究人员提出使用句子级模型预训练的方式来初始化模型参数<sup>[1117, 1118]</sup>, 或者将两个编码器的参数进行共享来降低模型复杂度<sup>[488, 1142, 1143]</sup>。

### 3. 层次结构模型

多编码器结构通过额外的编码器对前一句进行编码, 但是当处理更多上下文句子的时候仍然面临效率低下的问题。为了捕捉更大范围的上下文, 可以采用层次结构来对更多的上下文句子进行建模。层次结构是一种有效的序列表示方法, 而且人类语言中天然就具有层次性, 比如, 句法树、篇章结构树等。类似的思想也成功的应用在基于树的句子级翻译模型中 (见第八章和第十五章)。

图17.17描述了一个基于层次注意力的模型结构<sup>[487]</sup>。首先通过翻译模型的编码器

获取前  $K$  个句子的词序列编码表示  $(\mathbf{h}^{\text{pre}1}, \dots, \mathbf{h}^{\text{pre}K})$ , 然后针对前文每个句子的词序列编码表示  $\mathbf{h}^{\text{pre}k}$ , 使用词级注意力提取当前句子内部的注意力信息  $s^k$ , 然后在这  $K$  个句子级上下文信息  $\mathbf{s} = (s^1, \dots, s^K)$  的基础上, 使用句子级注意力提取篇章上下文信息  $\mathbf{d}$ 。由于上下文信息  $\mathbf{d}$  的获取涉及到词级和句子级两个不同层次的注意力操作, 因此将该过程称为层次注意力。实际上, 这种方法并没有使用语言学的篇章层次结构。但是, 句子级注意力在归纳统计意义上的篇章结构, 因此这种方法也可以捕捉不同句子之间的关系。

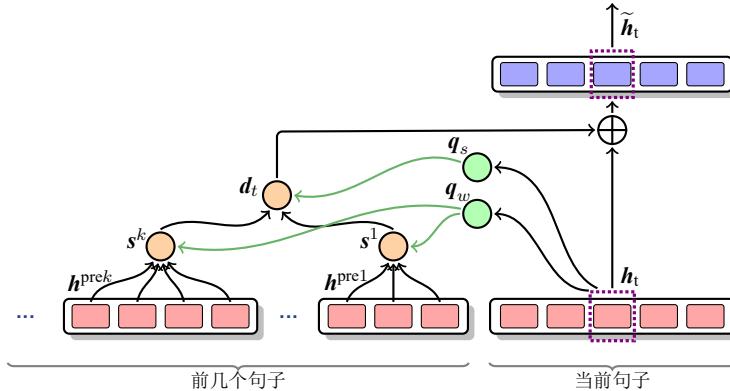


图 17.17 层次注意力结构<sup>[487]</sup>

为了增强模型的表示能力, 层次注意力中并未直接使用当前句子第  $t$  个位置的编码表示  $\mathbf{h}_t$  作为注意力操作的 Query (查询), 而是通过两个线性变换分别获取词级注意力和句子级注意力的查询  $\mathbf{q}_w$  和  $\mathbf{q}_s$ , 定义如公式(17.6)(17.7), 其中  $\mathbf{W}_w$ 、 $\mathbf{W}_s$ 、 $\mathbf{b}_w$ 、 $\mathbf{b}_s$  分别是两个线性变换的权重和偏置。

$$\mathbf{q}_w = \mathbf{W}_w \mathbf{h}_t + \mathbf{b}_w \quad (17.6)$$

$$\mathbf{q}_s = \mathbf{W}_s \mathbf{h}_t + \mathbf{b}_s \quad (17.7)$$

之后, 分别计算词级和句子级注意力模型。需要注意的是句子级注意力添加了一个前馈全连接网络子层 FFN。其具体计算方式如下:

$$s^k = \text{WordAttention}(\mathbf{q}_w, \mathbf{h}^{\text{pre}k}, \mathbf{h}^{\text{pre}k}) \quad (17.8)$$

$$\mathbf{d}_t = \text{FFN}(\text{SentAttention}(\mathbf{q}_s, \mathbf{s}, \mathbf{s})) \quad (17.9)$$

其中,  $\text{WordAttention}(\cdot)$  和  $\text{SentAttention}(\cdot)$  都是标准的自注意力模型。在得到最终的上下文信息  $\mathbf{d}$  后, 模型同样采用门控机制 (如公式(17.5) 和 公式(17.4)) 与  $\mathbf{h}$  进行融合来得到一个上下文相关的当前句子表示  $\tilde{\mathbf{h}}$ 。

通过层次注意力，模型可以在词级和句子级两个维度从多个句子中提取更充分的上下文信息，除了用于编码器，也可以用于解码器来获取目标语言的上下文信息。基于层次注意力，为了进一步编码整个篇章的上下文信息，研究人员提出选择性注意力来对篇章中整体上下文进行有选择的信息提取<sup>[1119]</sup>。此外，也有研究人员使用循环神经网络<sup>[1138]</sup>、记忆网络<sup>[1120]</sup>、胶囊网络<sup>[1121]</sup>和片段级相对注意力<sup>[1122]</sup>等结构来对多个上下文句子进行上下文信息提取。

#### 4. 基于缓存的方法

除了以上提到的建模方法，还有一类基于缓存的方法<sup>[1123, 1124]</sup>。这类方法最大的特点在于将篇章翻译看作一个连续的过程，即依次翻译篇章中的每一个句子，该过程中通过一个额外的缓存来记录一些相关信息，且在每个句子的推断过程中都使用这个缓存来提供上下文信息。图17.18描述了一种基于缓存的篇章级翻译模型结构<sup>[1124]</sup>。在这里，翻译模型基于循环神经网络（见第十章），但是这种方法同样适用于包括Transformer在内的其他神经机器翻译模型。

模型中篇章上下文的建模依赖于缓存的读和写操作。缓存的写操作指的是：按照一定规则，将翻译历史中一些译文单词对应的上下文向量作为键，将其解码器端的隐藏状态作为值，共同写入到缓存中。而缓存的读操作是指将待翻译句子中第 $t$ 个单词的上下文向量 $\mathbf{C}_t$ 作为Query（查询），与缓存中的所有键分别进行匹配，并根据其匹配程度进行带权相加，最后得到当前待翻译句子的篇章上下文信息 $\mathbf{d}$ 。该方法中，解码器端隐藏状态 $\mathbf{s}_t$ 与对应位置的上下文信息 $\mathbf{d}_t$ 的融合也是基于门控机制。事实上，由于该方法中缓存空间是有限的，其内容的更新也存在一定的规则：在当前句子的翻译结束后，如果单词 $y_t$ 的对应信息未曾写入缓存，则写入其中的空槽或者替换最久未使用的键值对；如果 $y_t$ 已作为翻译历史存在于缓存中，则将对应的键值对按照以下规则进行更新：

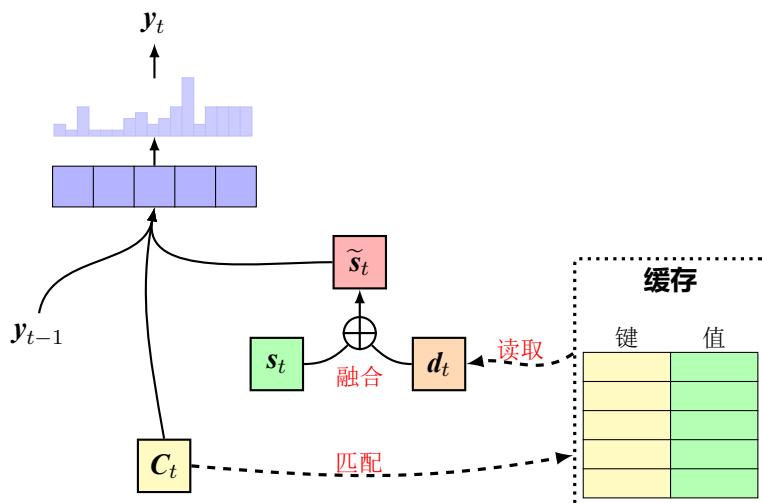
$$\mathbf{k}_i = \frac{\mathbf{k}_i + \mathbf{c}_t}{2} \quad (17.10)$$

$$\mathbf{v}_i = \frac{\mathbf{v}_i + \mathbf{s}_t}{2} \quad (17.11)$$

其中， $i$ 表示 $y_t$ 在缓存中的位置， $\mathbf{k}_i$ 和 $\mathbf{v}_i$ 分别为缓存中对应的键和值。这种方法缓存的都是目标语言历史的词级表示，因此能够解决一些词汇衔接的问题，比如词汇一致性和一些搭配问题，产生更连贯的翻译结果。

##### 17.4.4 在推断阶段结合篇章上下文

前面介绍的方法主要是对篇章中待翻译句子的上下文句子进行建模，通过端到端的方式对上下文信息进行提取和融合。由于篇章级双语数据相对稀缺，这种复杂的篇章级翻译模型很难得到充分训练，通常可以采用两阶段训练或参数共享的方式来缓解这个问题。此外，由于句子级双语数据更为丰富，一个自然的想法是以高质

图 17.18 基于缓存的解码器结构<sup>[1124]</sup>

量的句子级翻译模型为基础，通过在推断过程中结合上下文信息来构造篇章级翻译模型。

在句子级翻译模型中引入目标语言端的篇章级语言模型是一种结合上下文信息的常用手段<sup>[1125, 1126, 1127]</sup>。相比于篇章级双语数据，篇章级单语数据更容易获取。在双语数据稀缺的情况下，通过引入目标语言端的篇章级语言模型可以更充分的利用这些单语数据，比如，可以把这个语言模型与翻译模型做插值，也可以将其作为重排序阶段的一种特征。

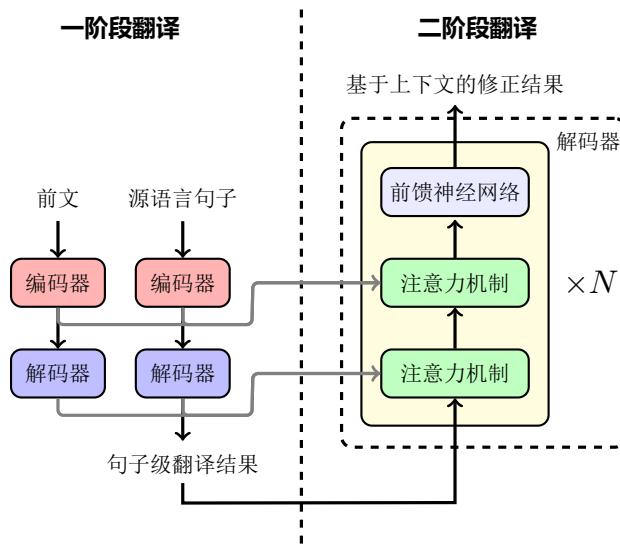


图 17.19 两阶段翻译

另一种方法是两阶段翻译。这种方法不影响句子级翻译模型的推断过程，而是

在完成翻译后使用额外的模块进行第二阶段的翻译<sup>[1128, 1129]</sup>。如图17.19所示，这种两阶段翻译的做法相当于将篇章级翻译的问题进行了分离和简化：在第一阶段翻译中使用句子级翻译模型完成对篇章中某个句子的翻译，为了进一步地引入篇章上下文信息，第二阶段的翻译过程在第一阶段翻译结果的基础上，利用两次注意力操作，融合并引入源语言和目标语言的篇章上下文信息和当前句子信息。该方法适用于篇章级双语数据稀缺的场景。基于类似的思想，也可以使用后编辑的做法对翻译结果进行修正。区别于两阶段翻译的方法，后编辑的方法无需参考源语言信息，只利用目标语言端的上下文信息对译文结果进行修正<sup>[1130]</sup>。

## 17.5 小结及拓展阅读

使用更大上下文进行机器翻译建模是极具潜力的研究方向，包括多模态翻译在内的多个领域也非常活跃。有许多问题值得进一步思考与讨论：

- 本章仅对音频处理和语音识别进行了简单的介绍，具体内容可以参考一些经典书籍，比如关于信号处理的基础知识<sup>[1144, 1145]</sup>，以及语音识别的传统方法<sup>[1146, 1147]</sup>和基于深度学习的最新方法<sup>[1148]</sup>。
- 此外，语音翻译的一个重要应用是机器同声传译。机器同声传译的一个难点在于不同语言的文字顺序不同。目前，同声传译的一种思路是基于目前已经说出的语音进行翻译<sup>[1149]</sup>，比如，等待源语  $k$  个词语，然后再进行翻译，同时改进束搜索方式来预测未来的词序列，从而提升准确度<sup>[1150]</sup>。或者，对当前语音进行翻译，但需要判断翻译的词是否能够作为最终结果，已决定是否根据之后的语音重新进行翻译<sup>[1151, 1152]</sup>。第二种思路是动态预测当前时刻是应该继续等待还是开始翻译，这种方式更符合人类进行同传的行为。但是这种策略的难点在于标注每一时刻的决策状态十分耗时且标准难以统一，目前主流的方式是利用强化学习方法<sup>[1153, 1154]</sup>，对句子进行不同决策方案采样，最终学到最优的决策方案。此外，还有一些工作设计不同的学习策略<sup>[1155, 1156, 1157]</sup>或改进注意力机制<sup>[1158]</sup>以提升机器同声传译的性能。
- 在多模态机器翻译任务和篇章级机器翻译任务中，数据规模往往受限，导致模型训练困难，很难取得较好的性能。比如在篇章级机器翻译中，一些研究工作对这类模型的上下文建模能力进行了探索<sup>[489, 1159]</sup>，发现模型在小数据集上对上下文信息的利用并不能带来明显的性能提升。针对数据稀缺导致的训练问题，一些研究人员通过调整训练策略使得模型更容易捕获上下文信息<sup>[1160, 1161, 1162]</sup>。除了训练策略的调整，也可以使用数据增强的方式（例如，构造伪数据）来提升整体数据量<sup>[1141, 1163, 1164]</sup>，或者使用预训练的手段来利用额外的单语或图像数据<sup>[1165, 1166, 1167]</sup>。



## 18. 机器翻译应用技术

随着机器翻译品质的不断提升，越来越多的应用需求被挖掘出来。但是，一个优秀的机器翻译引擎并不意味着机器翻译可以被成功应用。机器翻译技术落地需要“额外”考虑很多因素，例如，数据处理方式、交互方式、应用的领域等，甚至机器翻译模型也要经过改造才能适应到不同的场景中。

本章将重点介绍机器翻译应用中所面临的一些实际问题，以及解决这些问题可以采用的策略。本章所涉及的内容较为广泛，一方面会大量使用本书前十七章的模型和方法，另一方面也会介绍新的技术手段。最终，本章会结合机器翻译的特点展示一些机器翻译可以应用的场景。

### 18.1 机器翻译的应用并不简单

近几年，无论从评测比赛的结果，还是论文发表数量上看，机器翻译的研究可谓火热。但是，客观的说，我们离机器翻译完美的应用还有相当的距离。这主要是因为，成熟的系统需要很多技术的融合。因此，机器翻译系统研发也是一项复杂的系统工程。而机器翻译研究大多是对局部模型和方法的调整，这也会造成一个现象：很多论文里报道的技术方法可能无法直接应用于真实场景的系统。这里，有几方面挑战：

- 机器翻译模型很脆弱。实验环境下，给定翻译任务，甚至给定训练和测试数据，机器翻译模型可以表现得很好。但是，应用场景是不断变化的。经常会出现训练数据缺乏、应用领域与训练数据不匹配、用户的测试方法与开发者不同等等

一系列问题。特别是，对于不同的任务，神经机器翻译模型需要进行非常细致的调整，理想中“一套包打天下”的模型和设置是不存在的。这些都导致一个结果：直接使用既有机器翻译模型很难满足不断变化的应用需求。

- 机器翻译缺少针对场景的应用技术。目前为止，机器翻译的研究进展已经为我们提供很好的机器翻译基础模型。但是，用户并不是简单的与这些模型“打交道”，他们更加关注如何解决自身的业务需求，例如，机器翻译应用的交互方式、系统是否可以自己预估翻译可信度等等。甚至，在某些场景中，用户对翻译模型占用的存储空间和运行速度都有非常严格的要求。
- 优秀系统的研发需要长时间的打磨。工程打磨也是研发优秀机器翻译系统的必备条件，有些时候甚至是决定性的。从科学研究的角度看，我们需要对更本质的科学问题进行探索，而非简单的工程开发与调试。但是，对一个初级的系统进行研究往往会掩盖掉“真正的问题”，因为很多问题在更优秀的系统中并不存在。

下面本章将重点对机器翻译应用中的若干技术问题展开讨论，旨在给机器翻译应用提供一些可落地的思路。

## 18.2 增量式模型优化

机器翻译的训练数据不是一成不变的。系统研发人员可以使用自有数据训练得到基础的翻译模型（或初始模型）。当应用这个基础模型时，可能会有新的数据出现，例如：

- 应用的目标领域和场景可能是研发系统时无法预见的，但是用户会有一定量的自有数据，可以用于系统优化。
- 系统在应用中会产生新的数据，这些数据经过一些筛选和修改也可以用于模型训练。

这时就产生一个问题，能否使用新的数据让系统变得更好？简单直接的方式是，将新的数据和原始数据混合重新训练系统，但是使用全量数据训练模型的周期很长，这种方法的成本很高。而且，新的数据可能是不断产生的，甚至是流式的。这时就需要一种快速、低成本的方式对模型进行更新。

增量训练就是满足上述需求的一种方法。第十三章已经就增量训练这个概念展开了一些讨论，这里重点介绍一些具体的实践手段。本质上，神经机器翻译中使用的随机梯度下降方法就是典型的增量训练方法，其基本思想是：每次选择一个样本对模型进行更新，这个过程反复不断执行，每次模型更新都是一次增量训练。当多个样本构成了一个新数据集时，可以把这些新样本作为训练数据，把当前的模型作为初始模型，之后正常执行机器翻译的训练过程即可。如果新增加的数据量不大（比如，几万句对），训练的代价非常低。

这里面的一个问题是，新的数据虽然能代表一部分的翻译现象，但是如果仅仅依赖新数据进行更新，会使模型对新数据过分拟合，从而无法很好地处理新数据之外的样本。这也可以被看做是一种灾难性遗忘的问题<sup>[1168]</sup>，即：模型过分注重对新样本的拟合，丧失了旧模型的一部分能力。在实际系统开发中，有几种常用的增量训练方法：

- **数据混合**<sup>[1169]</sup>。在增量训练时，除了使用新的数据，再混合一定量的旧数据，混合的比例可以根据训练的代价进行调整。这样，模型相当于在全量数据的一个采样结果上进行更新。
- **模型插值**<sup>[616]</sup>。在增量训练之后，将新模型与旧模型进行插值。
- **多目标训练**<sup>[1003, 1170, 1171]</sup>。在增量训练时，除了在新数据上定义损失函数之外，可以再定义一个在旧数据上的损失函数，这样确保模型可以在两个数据上都有较好的表现。也可以在损失函数中引入正则化项，使新模型的参数不会偏离旧模型的参数太远。

图18.1给出了上述方法的对比。在实际应用中，还有很多细节会影响增量训练的效果，比如，学习率大小的选择等。另外，新的数据积累到何种规模可以进行增量训练也是实践中需要解决的问题。一般来说，增量训练使用的数据量越大，训练的效果越稳定。但是，这并不是说数据量少就不可以进行增量训练，而是如果数据量过少时，需要考虑训练代价和效果之间的平衡。而且，过于频繁的增量训练也会带来更多的灾难性遗忘的风险，因此合理进行增量训练也是机器翻译应用中需要实践的。

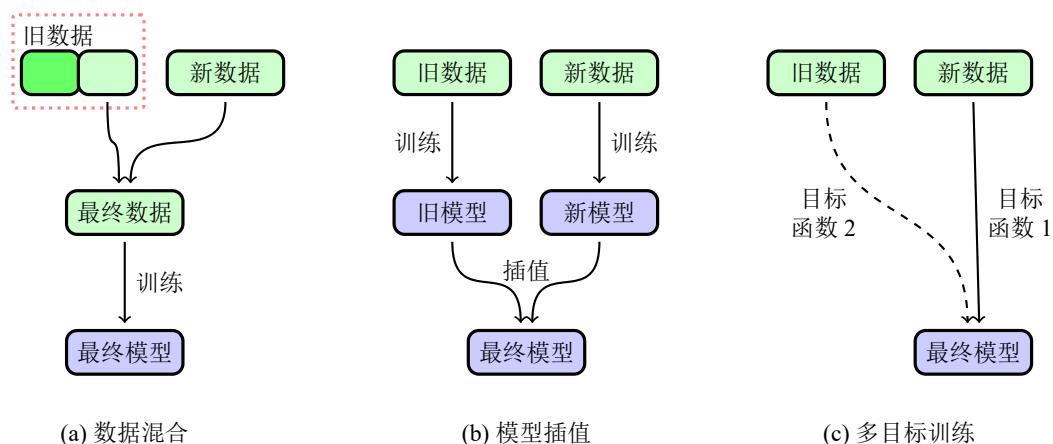


图 18.1 增量式模型优化方法

需要注意的是，理想状态下，系统使用者会希望系统看到少量句子就可以很好地解决一类翻译问题，即：进行真正的小样本学习。但是，现实的情况是，现在的机器翻译系统还无法很好的做到“举一反三”。增量训练也需要专业人士完成才能得到相对较好的效果。

另一个实际的问题是，当应用场景没有双语句对时是否可以优化系统？这个问题在第十六章的低资源翻译部分进行了一些讨论。一般来说，如果目标任务没有双语数据，仍然可以使用单语数据进行优化。常用的方法有数据增强、基于语言模型的方法等。具体方法可以参考第十六章的内容。

### 18.3 交互式机器翻译

机器翻译的结果会存在错误，因此很多时候需要人工的修改才能被使用。例如，在**译后编辑**（Post-editing）中，翻译人员对机器翻译的译文进行修改，最终使译文达到要求。但是，译后编辑的成本仍然很高，因为它需要翻译人员阅读机器翻译的结果，同时做出修改的动作。有时候，由于译文修改的内容较为复杂，译后编辑的时间甚至比人工直接翻译源语言句子的时间都长。因此在机器翻译应用中，需要更高效的方式调整机器翻译的结果，使其达到可用的程度。比如，可以使用质量评估方法（见第四章），选择模型置信度较高的译文进行译后编辑，对置信度低的译文直接进行人工翻译。而另一种思路是，让人的行为直接影响机器翻译生成译文的过程，让人和机器翻译系统进行交互，在不断的修正中生成更好的译文。这种方法也被称作**交互式机器翻译**（Interactive Machine Translation, IMT）。

交互式机器翻译的大致流程如下：机器翻译系统根据用户输入的源语言句子预测出可能的译文交给用户，然后用户在现有翻译的基础上进行接受、修改或者删除等操作，然后翻译系统根据用户的反馈信息再次生成比前一次更好的翻译并提交给用户。以此循环，直到得到最终的译文。

图18.2给出了一个使用 TranSmart 系统进行交互式机器翻译的例子，在这里要将一个汉语句子“疼痛/也/可能/会在/夜间/使/你/醒来。”翻译成英语“Pain may also wake you up during the night .”。在开始交互之前，系统首先推荐一个可能的译文“Pain may also wake you up at night .”。在第一次交互中，用户将单词 at 替换成 during，然后系统根据用户修改后的译文立即给出新的译文候选，提供给用户选择。循环往复，直到用户接受了系统当前推荐的译文。

交互式机器翻译系统主要通过用户的反馈来提升译文的质量，不同类型的反馈信息则影响着系统最终的性能。根据反馈形式的不同，可以将交互式机器翻译分为以下几种：

- **基于前缀的交互式机器翻译。**早期的交互式机器翻译系统都是采用基于前缀的方式。基于翻译系统生成的初始译文，翻译人员从左到右检查翻译的正确性，并在第一个错误的位置进行更正。这为系统提供了一种双重信号：表明该位置上的单词必须是翻译人员修改过后的单词，并且该位置之前的单词都是正确的。之后系统根据已经检查过的前缀再生成后面的译文<sup>[645, 1172, 1173, 1174]</sup>。
- **基于片段的交互式机器翻译。**根据用户提供的反馈来生成更好的翻译结果是交互式翻译系统的关键。而基于前缀的系统则存在一个严重的缺陷，当翻译系统

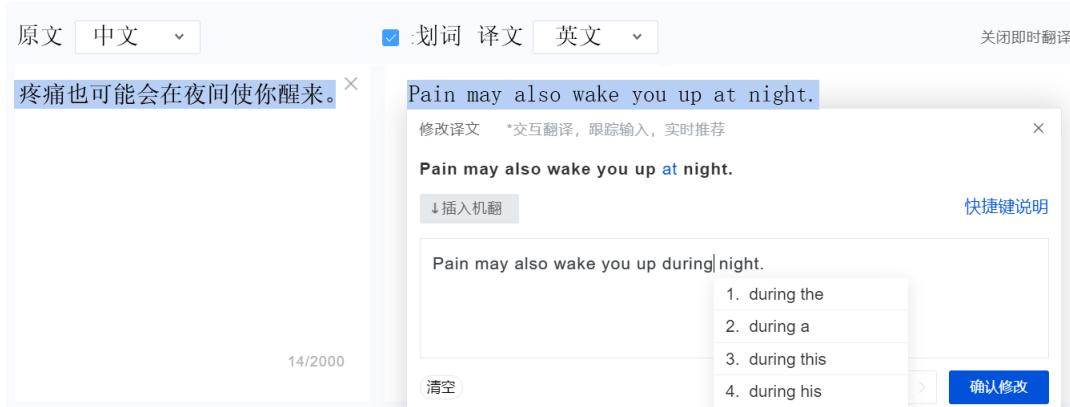


图 18.2 使用 TranSmart 系统进行交互式翻译的实例

获得确定的翻译前缀之后，再重新生成译文时会将原本正确的翻译后缀遗漏了，因此会引入新的错误。在基于片段的交互式机器翻译系统中，翻译人员除了纠正第一个错误的单词，还可以指定在未来迭代中保留的单词序列。之后系统根据这些反馈信号再生成新的译文<sup>[703, 1175]</sup>。

- 基于评分的交互式机器翻译。随着计算机算力的提升，有时会出现“机器等人”的现象，因此需要提升人参与交互的效率也是需要考虑的。与之前的系统不同，基于评分的交互式机器翻译系统不需要翻译人员选择、纠正或删除某个片段，而是使用翻译人员对译文的评分来强化机器翻译的学习<sup>[664, 1176]</sup>。

除此之外，基于在线学习的方法也受到了关注，这类方法也可以被看作是交互式翻译与增量训练的一种结合。用户总是希望翻译系统能从反馈中自动纠正以前的错误。当用户最终确认一个修改过后的译文后，翻译系统将源语言句子与该修正后的译文作为训练语料继续训练<sup>[1177]</sup>。实际上，交互式机器翻译是机器翻译大规模应用的重要途径之一，它为打通翻译人员和机器翻译系统之间的障碍提供了手段。不过，交互式机器翻译也有许多挑战等待解决。一个是如何设计交互方式？理想的方式应该是更加贴近翻译人员输入文字的习惯，比如，利用输入法完成交互；另一个是如何把交互式翻译嵌入到翻译的生产流程里？这本身不完全是一个技术问题，可能需要更多的产品手段来求解。

## 18.4 翻译结果的可干预性

交互式机器翻译体现了一种用户的行为“干预”机器翻译结果的思想。实际上，在机器翻译出现错误时，人们总是希望用一种直接有效的方式“改变”译文，最短时间内达到改善翻译质量的目的。比如，如果机器翻译系统可以输出多个候选译文，用户可以在其中挑选最好的译文进行输出。也就是，人为干预了译文候选的排序过程。另一个例子是翻译记忆（Translation Memory, TM）。翻译记忆记录了高质量的源

语言-目标语言句对，有时也可以被看作是一种先验知识或“记忆”。因此，当进行机器翻译时，使用翻译记忆指导翻译过程也可以被看作是一种干预手段<sup>[1178, 1179]</sup>。

虽然干预机器翻译系统的方式很多，最常用的是对源语言特定片段翻译的干预，以期望最终句子的译文满足某些约束。这个问题也被称作**基于约束的翻译**（Constraint-based Translation）。比如，在翻译网页时，需要保持译文中的网页标签与源文一致。另一个典型例子是**术语翻译**。在实际应用中，经常会遇到公司名称、品牌名称、产品名称等专有名词和行业术语，以及不同含义的缩写，比如，对于“小牛翻译”这个专有名词，不同的机器翻译系统给出的结果不一样：“Maverick translation”、“Calf translation”、“The mavericks translation”等等，而它正确的翻译应该为“NiuTrans”。对于这些类似的特殊词汇，机器翻译引擎很难翻译得准确。一方面，因为模型大多是在通用数据集上训练出来的，并不能保证数据集能涵盖所有的语言现象。另一方面，即使是这些术语在训练数据中出现，它们通常也是低频的，模型不容易捕捉它们的规律。为了保证翻译的准确性，对术语翻译进行干预是十分有必要的，对领域适应等问题的求解也是非常有意义的。

就**词汇约束翻译**（Lexically Constrained Translation）而言，在不干预的情况下让模型直接翻译出正确术语是很难的，因为术语的译文很可能是未登录词，因此必须人为提供额外的术语词典，那么我们的目标就是让模型的翻译输出遵守用户提供的术语约束。这个过程如图18.3所示。



图 18.3 词汇约束翻译过程

在统计机器翻译中，翻译本质上是由短语和规则构成的推导，因此修改译文比较容易，比如，可以在一个源语言片段所对应的翻译候选集中添加希望得到的译文。而神经机器翻译是一个端到端模型，翻译过程本质上是连续空间中元素的一系列映射、组合和代数运算。虽然在模型训练阶段仍然可以通过修改损失函数等手段引入约束，但是在推断阶段进行直接干预并不容易，因为我们无法像修改符号系统那样直接修改模型（如短语翻译表）来影响译文生成。实践中主要有两种解决思路：

- **强制生成**。这种方法并不改变模型，而是在推断过程中按照一定的策略来实施约束，一般是修改束搜索算法以确保输出必须包含指定的词或者短语<sup>[1180, 1181, 1182, 1183]</sup>，例如，在获得译文输出后，利用注意力机制获取词对齐，之后通过词对齐得到源语言和目标语言片段的对应关系，最后对指定译文片段进行强制替换。或者，对包含正确术语的翻译候选进行额外的加分，以确保推断时这样的翻译候选的

排名足够靠前。

- **数据增强**。这类方法通过修改机器翻译模型的训练数据来实现术语约束。通常根据术语词典对训练数据进行一定的修改，例如，将术语的译文添加到源文句子中，之后将原始语料库和合成语料库进行混合训练，期望模型能够学会自动利用术语信息来指导解码，或者是在训练数据中利用占位符来替换术语，待翻译完成后再进行还原<sup>[1184, 1185, 1186, 1187]</sup>。

强制生成的方法是在搜索策略上进行限制，与模型无关，这类方法能保证输出满足约束，但是会影响翻译速度。数据增强的方法是通过构造特定格式的数据让模型训练，从而让模型具有自动适应术语约束的能力，通常不会影响翻译速度，但并不能保证输出能满足约束。

此外，机器翻译在应用时通常还需要进行译前译后的处理，译前处理指的是在翻译前对源语言句子进行修改和规范，从而能生成比较通顺的译文，提高译文的可读性和准确率。在实际应用时，由于用户输入的形式多样，可能会包含比如术语、缩写、数学公式等，有些甚至可能还包含网页标签，因此对源文进行预处理是很有必要的。常见的处理工作包括格式转换、标点符号检查、术语编辑、标签识别等，待翻译完成后，则需要对机器译文进行进一步的编辑和修正，从而使其符合使用规范，比如进行标点、格式检查，术语、标签还原等，这些过程通常都是按照设定的处理策略自动完成的。另外，译文长度的控制、译文多样性的控制等也可以丰富机器翻译系统干预的手段（见第十四章）。

## 18.5 小设备机器翻译

在机器翻译研究中，一般会假设计算资源是充足的。但是，在很多应用场景中，机器翻译使用的计算资源非常有限，比如，一些离线设备上没有 GPU，而且 CPU 的处理能力也很弱，甚至内存也非常有限。这时，让模型变得更小、系统变得更快就成为了重要的需求。

本书中已经讨论了大量的可用于小设备上的机器翻译技术方法，例如：

- 知识蒸馏（第十三章）。这种方法可以有效地将翻译能力从大模型迁移到小模型。
- 低精度存储及计算（第十四章）。可以使用量化的方式将模型压缩，同时整型计算也非常适合在 CPU 等设备上执行。
- 轻量模型结构（第十四章和第十五章）。对机器翻译模型的局部结构进行优化也是非常有效的手段，比如，使用更加轻量的卷积计算模块，或者使用深编码器-浅解码器等高效的结构。
- 面向设备的模型结构学习（第十五章）。可以把设备的存储及延时作为目标函数的一部分，自动搜索高效的翻译模型结构。

- 动态适应性模型<sup>[767, 1188, 1189]</sup>。模型可以动态调整大小或者计算规模，以达到在不同设备上平衡延时和精度的目的。比如，可以根据延时的要求，动态生成合适深度的神经网络进行翻译。

此外，机器翻译系统的工程实现方式也是十分重要的，例如，编译器的选择、底层线性代数库的选择等等。有时候，使用与运行设备相匹配的编译器，会带来明显的性能提升<sup>1</sup>。如果希望追求更加极致的性能，甚至需要对一些热点模块进行修改。例如，在神经机器翻译中，矩阵乘法就是一个非常耗时的部分。但是这部分计算又与设备、矩阵的形状有很大关系。对于不同设备，根据不同的矩阵形状可以设计相应的矩阵乘法算法。不过，这部分工作对系统开发和硬件指令的使用水平要求较高。

另外，在很多系统中，机器翻译模块并不是单独执行，而是与其它的模块并发执行。这时，由于多个计算密集型任务存在竞争，处理器要进行更多的上下文切换，会造成程序变慢。比如，机器翻译和语音识别两个模块一起运行时<sup>2</sup>，机器翻译的速度会有较明显的下降。对于这种情况，需要设计更好的调度机制。因此在一些同时具有 CPU 和 GPU 的设备上，可以考虑合理调度 CPU 和 GPU 的资源，增加两种设备可并行处理的内容，避免在某个处理器上的拥塞。

除了运行速度，模型过大也是限制其在小设备上运行的因素。在模型体积上，神经机器翻译模型具有天然的优势。因此，在对模型规模有苛刻要求的场景中，神经机器翻译是不二的选择。另外通过量化、剪枝、参数共享等方式，可以将模型大幅度压缩。

## 18.6 机器翻译系统的部署

除了在一些离线设备上使用机器翻译，更多时候机器翻译系统会部署在运算能力较强的服务器上。一方面随着神经机器翻译的大规模应用，在 GPU 服务器上部署机器翻译系统已经成为了常态。另一方面，GPU 服务器的成本较高，而且很多应用中需要同时部署多个语言方向的系统。这时如何充分利用设备以满足大规模的翻译需求就成为了不可回避的问题。有几个方向值得尝试：

- 对于多语言翻译的场景，使用多语言单模型翻译系统是一种很好的选择（第十六章）。当多个语种的数据量有限、使用频度不高时，这种方法可以很有效地解决翻译需求中的长尾部分。例如，一些线上机器翻译服务已经支持超过 100 种语言的翻译，其中大部分语言之间的翻译需求是相对低频的，因此使用同一个模型进行翻译可以大大节约部署和运维的成本。
- 使用基于枢轴语言的翻译也可以有效的解决多语言翻译问题（第十六章）。这种方法同时适合统计机器翻译和神经机器翻译，因此很早就使用在大规模机器

<sup>1</sup>以神经机器翻译为例，张量计算部分大多使用 C++ 等语言编写，因此编译器与设备的适配程度对程序的执行效率影响很大。

<sup>2</sup>在一些语音翻译场景中，由于采用了语音识别和翻译异步执行的方式，两个程序可能会并发。

翻译部署中。

- GPU 部署中，由于 GPU 成本较高，因此可以考虑在单个 GPU 设备上部署多套不同的系统。如果这些系统之间的并发不频繁，翻译延时不会有明显增加。这种多个模型共享一个设备的方法比较适合翻译请求相对低频但是翻译任务又很多样的情况。
- 机器翻译大规模 GPU 部署对显存的使用也很严格。由于 GPU 显存较为有限，因此模型运行的显存消耗也是需要考虑的。一般来说，除了模型压缩和结构优化之外（第十四章和第十五章），也需要对模型的显存分配和使用进行单独的优化。例如，使用显存池来缓解频繁申请和释放显存空间造成的延时。另外，也可以尽可能让同一个显存块保存生命周期不重叠的数据，避免重复开辟新的存储空间。图18.4展示了显存复用的一个示例。

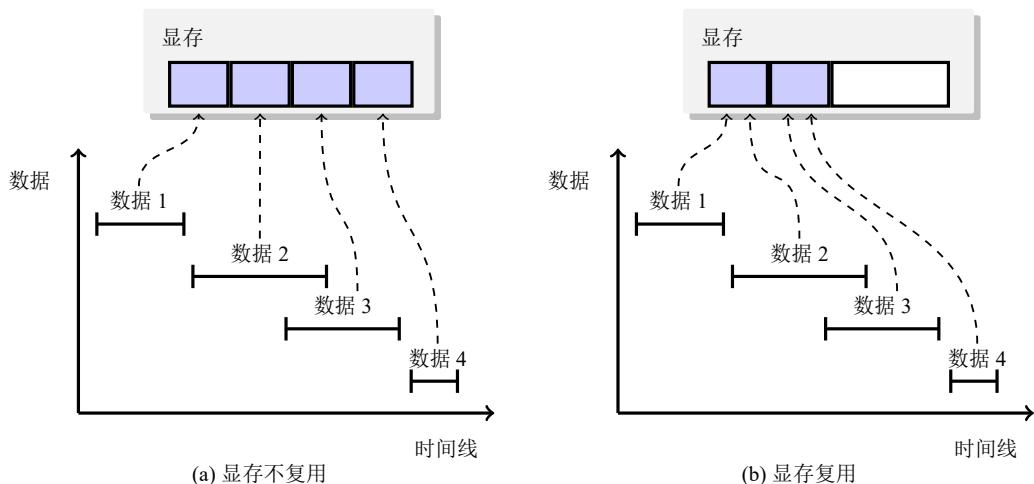


图 18.4 显存不复用与显存复用的示例

- 在翻译请求高并发的场景中，使用批量翻译也是有效利用 GPU 设备的方式。不过，机器翻译是一个处理不定长序列的任务，输入的句子长度差异较大。而且，由于译文长度无法预知，进一步增加了不同长度的句子所消耗计算资源的不确定性。这时，可以让长度相近的句子在一个批次里处理，减小由于句子长度不统一造成的补全过多、设备利用率低的问题。例如，可以按输入句子长度范围分组。也可以设计更加细致的方法对句子进行分组，以最大化批量翻译中设备的利用率<sup>[1190]</sup>。

除了上述问题，如何对多设备环境下进行负载均衡、容灾处理等都是大规模机器翻译系统部署中需要考虑的。有时候，甚至统计机器翻译系统也可以与神经机器翻译系统混合使用。由于统计机器翻译系统对 GPU 资源的要求较低，纯 CPU 部署的方案也相对成熟。因此，可以作为 GPU 机器翻译服务的灾备。此外，在有些任务，

特别是某些低资源翻译任务上，统计机器翻译仍然具有优势。

## 18.7 机器翻译的应用场景

机器翻译有着十分广泛的应用，这里列举了一些常见的应用场景：

- **网页翻译。**进入信息爆炸的时代之后，互联网海量的数据随处可得，然而由于不同国家和地区语言的差异，网络上的数据也呈现出多语言的特性。当人们在遇到包含不熟悉语言的网页时，无法及时有效地获取其中的信息。因此，对不同语言的网页进行翻译是必不可少的一步。由于网络上的网页数不胜数，依靠人工对网页进行翻译是不切实际的，相反，机器翻译十分适合这个任务。目前，市场上有很多浏览器提供网页翻译的服务，极大地降低了人们从网络上获取不同语言信息的难度。
- **科技文献翻译。**在专利等科技文献翻译中，往往需要将文献翻译为英语或者其他语言，比如摘要翻译。以往这种翻译工作通常由人工来完成。由于对翻译结果的质量要求较高，因此要求翻译人员具有相关专业的背景知识，这导致翻译人员资源稀缺。特别是，近几年国内专利申请数不断增加，这给人工翻译带来了很大的负担。相比于人工翻译，机器翻译可以在短时间内完成大量的专利翻译，同时结合术语词典和人工校对等方式，可以保证专利的翻译质量。另外，以专利为代表的科技文献往往具有很强的领域性，针对各类领域文本进行单独优化，机器翻译的品质可以大大提高。因此，机器翻译在专利翻译等行业有十分广泛的应用前景。
- **视频字幕翻译。**随着互联网的普及，人们可以通过互联网接触到大量境外影视作品。由于人们可能没有相应的外语能力，通常需要翻译人员对字幕进行翻译。因此，这些境外视频的传播受限于字幕翻译的速度和准确度。现在的一些视频网站在使用语音识别为视频生成源语言字幕的同时，通过机器翻译技术为各种语言的受众提供质量尚可的目标语言字幕，这种方式为人们提供了极大的便利。
- **社交。**社交是人们的重要社会活动。人们可以通过各种各样的社交软件做到即时通讯，进行协作或者分享自己的观点。然而受限于语言问题，人们的社交范围往往不会超出自己所掌握的语种范围，很难方便地进行跨语言社交。随着机器翻译技术的发展，越来越多的社交软件开始支持自动翻译，用户可以轻易地将各种语言的内容翻译成自己的母语，方便了人们的交流，让语言问题不再成为社交的障碍。
- **同声传译。**在一些国际会议中，与会者来自许多不同的国家，为了保证会议的流畅，通常需要专业翻译人员进行同声传译。同声传译需要在不打断演讲的同时，不间断地将讲话内容进行口译，对翻译人员的要求极高。现在，一些会议开始采用语音识别来将语音转换成文本，同时使用机器翻译技术进行翻译的方式，达到同步翻译的目的。这项技术已经得到了多个企业的关注，并在很多重

要在会议上进行尝试，取得了很好的反响。不过同声传译达到可以真正使用的程度还需一定时间的打磨，特别是会议场景下，准确进行语音识别和翻译仍然具有挑战性。

- **中国传统语言文化的翻译。**中国几千年的历史留下了极为宝贵的文化遗产，而其中，文言文作为古代书面语，具有言文分离、行文简练的特点，易于流传。言文分离的特点使得文言文和现在的标准汉语具有一定的区别。为了更好发扬中国传统文化，需要对文言文进行翻译。而文言文古奥难懂，人们需要具备一定的文言文知识背景才能准确翻译。机器翻译技术也可以帮助人们快速完成文言文的翻译。除此之外，机器翻译技术同样可以用于古诗生成和对联生成等任务。
- **全球化。**在经济全球化的今天，很多企业都有国际化的需求，企业员工或多或少地会遇到一些跨语言阅读和交流的情况，比如阅读进口产品的说明书，跨国公司之间的邮件、说明文件等等。相比于成本较高的人工翻译，机器翻译往往是一种很好的选择。在一些质量要求不高的翻译场景中，机器翻译可以得到应用。
- **翻译机/翻译笔。**出于商务、学术交流或者旅游的目的，人们在出国时会面临着跨语言交流的问题。近几年，随着出境人数的增加，不少企业推出了翻译机产品。通过结合机器翻译、语音识别和图像识别技术，翻译机实现了图像翻译和语音翻译的功能。用户可以很便捷地获取一些外语图像文字和语音信息，同时可以通过翻译机进行对话，降低跨语言交流门槛。类似地，翻译笔等应用产品可以通过划词翻译的方式，对打印材料中的外语文字进行翻译。
- **译后编辑。**翻译结果后编辑是指在机器翻译的结果之上，通过少量的人工编辑来进一步完善机器译文。在传统的人工翻译过程中，翻译人员完全依靠人工的方式进行翻译，这虽然保证了翻译质量，但是时间成本高。相对应地，机器翻译具有速度快和成本低的优势。在一些领域，目前的机器翻译质量已经可以很大程度上减少翻译人员的工作量，翻译人员可以在机器翻译的辅助下，花费相对较小的代价来完成翻译。

## 随 筆

自计算机诞生，机器翻译，即利用计算机软件技术实现不同语言自动翻译，就是人们首先想到的计算机主要应用。很多人说，人工智能时代是得语言者的天下，并将机器翻译当作认知智能的终极梦想之一。接下来，笔者将分享自己对机器翻译技术和应用的思考，有些想法不一定正确，有些想法也许需要十年或更久才能被验证。

简单来说，机器翻译技术至少可以满足三种用户需求。一是实现外文资料辅助阅读，帮助不同母语的人进行无障碍交流；二是通过计算机辅助翻译，帮助人工翻译降本增效；三是通过大数据分析和处理，实现对多语言文字资料（也可以是图像资料或语音资料）的加工处理。仅凭人工，是无法完成海量数据的翻译工作的，而机器翻译是大数据翻译的唯一有效解决方案。从上述三种需求可以看出，机器翻译和人工翻译在本质上不存在冲突，两者可以和谐共存、相互帮助，处于平行轨道上。对机器翻译来说，至少有两个应用场景是其无法独立胜任的。一是对翻译结果的质量要求高的场景，如诗歌、小说的翻译出版；二是不允许出现低级实时翻译错误的场景，如国际会议的发言。因此，对译文准确性要求很高的应用场景不可能只采用机器翻译，必须有高水平的人工翻译参与。

如何构建一套好的机器翻译系统呢？假设我们需要为用户提供一套翻译品质不错的机器翻译系统，至少需要考虑三个方面：有足够的大规模的双语句对集合用于训练、有强大的机器翻译技术和错误驱动的打磨过程。从技术应用和产业化的角度看，对于构建一套好的机器翻译系统来说，上述三个方面缺一不可。仅拥有强大的机器翻译技术是必要条件，但不是充分条件。更具体地：

- 从数据角度来看，大部分语言对的电子化双语句对集合规模非常小，有的甚至只有一个规模双语词典。因此，针对资源稀缺语种的机器翻译技术研究也成了学术界的研究热点，相信这个课题的突破能大大推动机器翻译技术落地。早些年，机器翻译市场的规模较小，其主要原因是数据规模有限，同时机器翻译的品质不够理想。就算采用最先进的神经机器翻译技术，在缺乏足够大规模的双语句对集合作为训练数据的情况下，研究人员也是巧妇难为无米之炊。从技术研究和应用可行性的角度看，解决资源稀缺语种的机器翻译问题非常有价值。解决资源稀缺语种机器翻译问题的思路，已经在第十六章进行了详细的介绍，本部分就不再赘述。
- 从机器翻译技术来看，可实用的机器翻译系统的构建，需要多技术互补融合。做研究可以搞单点突破，但它很难能应对实际问题和改善真实应用中的翻译品质。多技术互补融合有很多研究工作，比如说，有的业内研究人员提出采用知识图谱来改善机器翻译模型性能，并希望用于解决稀缺资源语种机器翻译问题；有的引入语言分析技术来改善机器翻译；有的将基于规则的方法、统计机器翻

译技术与神经机器翻译技术互补性融合；有的引入预训练技术来改善机器翻译品质等等。总体来说，这些思路都具有良好的研究价值，但是从应用角度来说，构建可实用的机器翻译系统，还需要考虑技术落地可行性。比如大规模知识图谱构建的代价和语言分析技术的精度如何，预训练技术对富资源场景下机器翻译的价值等。

- 错误驱动，即根据用户对机器翻译译文的反馈与纠正，完善机器翻译模型的过程。机器翻译一直被诟病：用户不知道如何有效地干预纠错，来帮助机器翻译系统越做越好，毕竟谁都不希望它“屡教不改”。基于规则的方法和统计机器翻译方法相对容易实现人工干预纠错，实现手段也比较丰富，而神经机器翻译方法很多时候被看做是黑箱，其运行机理与离散的符号系统有很大差别，难以用传统方式有效地实现人工干预纠错。目前，有研究人员通过引入外部知识库（用户双语术语库）来实现对未登录词翻译的干预纠错；也有的提出使用增量式训练的方法不断迭代优化模型，取得了一些进展；还有研究人员通过融合不同技术来实现更好的机器翻译效果，如引入基于规则的翻译前处理和后处理，或者引入统计机器翻译技术优化译文选择等。这些方法的代价不低，甚至很高，并且无法保障对机器翻译性能提升的效果，有时可能会降低翻译品质（有点像“跷跷板”现象）。总体来说，这个方向的研究成果还不够丰富，但对用户体验来说非常重要。如果能采用隐性反馈学习方法，在用户不知不觉中不断改善、优化机器翻译品质，就非常酷了，这也许会成为将来的一个研究热点。

除了翻译品质维度以外，机器翻译还可以从以下三个维度来讨论：语种维度、领域维度和应用模式维度。关于语种维度，机器翻译技术应该为全球用户服务，提供所有国家至少一种官方语言到其他国家语言的自动互译功能。该维度面临的最大问题是双语数据稀缺。关于领域维度，通用领域翻译系统的翻译能力，对于垂直领域数据来说是不足的。最典型的问题是不能恰当地翻译垂直领域术语，计算机不能无中生有。比较直接可行的解决方案至少有两个，一是引入垂直领域术语双语词典来改善机器翻译效果；二是收集加工一定规模的垂直领域双语句对来优化翻译模型。这两种工程方法虽然简单，但效果不错，并且两者结合对于翻译模型性能的提升帮助更大。但很多时候垂直领域双语句对的收集代价太高，可行性低，因此垂直领域翻译问题本质上就转换成为垂直领域资源稀缺问题和领域自适应学习问题。除此之外，小样本学习、迁移学习等机器学习技术也被一些研究人员用来解决垂直领域翻译问题。关于应用模式维度，可以从下面几个方面进行讨论：

- 通常，机器翻译的典型应用包括在线翻译公有云服务，用户接入非常简单，只需要联网使用浏览器就可以自由免费使用。在某些行业，用户对数据翻译安全性和保密性的要求非常高，其中可能还会涉及个性化定制，这是在线翻译公有云服务无法满足的，于是，在本地部署机器翻译私有云、离线机器翻译技术和服务成了新的应用模式。在本地部署私有云的问题在于：需要用户自己购买GPU服务器并建机房，对硬件的投入高。也许将来机器翻译领域会出现新的应用模式：类似服务托管模式的在线私有云或专有云，以及混合云服务（公有云、私

有云和专有云的混合体)。

- 离线机器翻译技术可以为更小型的智能翻译终端设备提供服务,如大家熟知的翻译机、翻译笔、翻译耳机等智能翻译设备。在不联网的情况下,这些设备能实现高品质机器翻译功能,这类应用模式具有很大的潜力。但这类应用模式需要解决的问题也很多:首先是模型大小、翻译速度和翻译品质的问题;其次,考虑不同操作系统(如Linux、Android Q和iOS)和不同架构(如x86、MIPS、ARM等)的CPU芯片的智能适配兼容问题。将来,离线翻译系统还可以通过芯片安装到办公设备上,如传真机、打印机和复印机等,辅助人们实现支持多语言的智能办公。目前,人工智能芯片发展的速度非常快,而机器翻译芯片研发面临的最大问题是缺少应用场景和上下游的应用支撑,一旦时机成熟,机器翻译芯片的研发和应用也有可能会爆发。
- 机器翻译可以与文档解析、语音识别、光学字符识别(OCR)和视频字幕提取等技术相结合,丰富机器翻译的应用模式。具体的:
  - 文档解析技术可以帮助实现Word文档翻译、PDF文档翻译、WPS文档翻译、邮件翻译等更多格式文档自动翻译的目标,也可以作为插件嵌入到各种办公平台中,成为智能办公好助手。
  - 语音识别与机器翻译是绝配,语音翻译用途广泛,比如翻译机、语音翻译APP和会议AI同传应用。但目前最大的问题主要体现在两个方面,一是很多实际应用场景中语音识别效果欠佳,造成错误蔓延,导致机器翻译结果不够理想;二是就算小语种的语音识别效果很好,但资源稀缺型小语种翻译性能不够好。
  - OCR技术可以帮助实现扫描笔和翻译笔的应用、出国旅游的拍照翻译功能,将来还可以与穿戴式设备相结合,比如智能眼镜等等。视频字幕翻译能够帮助我们欣赏没有中文字幕的国外电影和电视节目,比如到达任何一个国家,打开电视都能够看到中文字幕,也是非常酷的应用。

上面提到的机器翻译技术大多采用串行流水线,只是简单将两个或者多个不同的技术连接在一起,比如语音翻译过程可以分两步:语音识别和机器翻译。其它翻译模式也大同小异。简单的串行流水线技术框架的最大问题是错误蔓延,一旦某个技术环节的准确率不高,最后的结果就不会太好( $90\% \times 90\% = 81\%$ )。并且,后续的技术环节不一定有能力纠正前面技术环节引入的错误,最终导致用户体验不够好。很多人认为,英中AI会议同传用户体验不够好,问题出在机器翻译技术上。其实,问题主要出在语音识别环节。学术界正在研究的端到端的机器翻译技术,不是采用串行流水线技术架构,而是采用一步到位的方式,这理论上能够缓解错误蔓延的问题,但目前的效果还不够理想,期待学术界取得新的突破。

- 机器翻译技术可以辅助人工翻译。即使双语句对训练集合规模已经非常大、机器翻译技术也在不断优化,但机器翻译的结果仍然不可能完美,出现译文错误是难免的。如果我们想利用机器翻译技术辅助人工翻译,比较常见的方式是译

后编辑，即由人对自动译文进行修改（详见第四章）。这就很自然地产生了两个实际问题：第一个问题是，自动译文是否具有编辑价值？一个简便的计算方法就是编辑距离，即人工需要通过多少次增、删、改动作完成译后编辑。其次数越少，说明机器翻译对人工翻译的帮助越大。编辑距离本质上是一种译文质量评价的方法，可以考虑推荐具有较高译后编辑价值的自动译文给人工译员。第二个问题是，当机器翻译出现错误，且被人工译后编辑修正后，能否通过一种有效的错误反馈机制帮助机器翻译系统提高性能。学术界也有很多人研究这个问题，目前还没有取得令人满意的结果。除此之外，还有一些问题，如人机交互的用户体验，该需求很自然地带起了交互式机器翻译技术（详见第十八章）研究的热潮，希望在最大程度上发挥人机协同合作的效果，这个也是值得研究的课题。

接下来，简单谈谈笔者对第四代机器翻译技术发展趋势的看法。通常，我们分别将基于规则的方法、统计机器翻译和神经机器翻译称为第一、第二和第三代机器翻译技术。有人说，第四代机器翻译技术会是基于知识的机器翻译技术；也有人说，是无监督机器翻译技术或者新的机器翻译范式，等等。在讨论第四代机器翻译技术这个问题之前，我们先思考一个问题：在翻译品质上，新一代机器翻译技术是否应该比目前的好？现在的实验结果显示，比如拿商用的英汉汉英新闻机器翻译系统举例，经过几亿双语句对的训练学习，机器翻译译文准确率的人工评估得分可以达到80%–90%（100%为满分，值越高说明译文准确率越高），那我们需要回答的一个简单问题是：所谓的第四代机器翻译技术准备在新闻领域翻译达到怎样的准确率呢？只比现在高2或3个百分点，达到92%或者93%这一结果，估计无法获得新一代机器翻译技术这一称谓。

从历史发展观的维度考虑，新一代的技术必然存在，换句话说，第四代机器翻译技术一定会出现，只是不知道在什么时候而已。神经机器翻译的红利还没有被挖尽，还存在很好的发展空间，在可预期的将来，神经机器翻译技术还属于主流技术，但会产生大量变种。我们愿意把新一代机器翻译技术称为面向具体应用场景的第四代机器翻译技术，它在本质上是针对不同应用条件、不同应用场景提出的能力更强的机器翻译技术。它将不是一个简单的技术，而是一个技术集合，这是完全可能的。从另一方面讲，当前的机器翻译不具有很好的解释性，其与语言学的关系并不明确。那么在第四代机器翻译技术中，是否能让研究人员或使用者更方便地了解它的工作原理，并可以根据其原理对其进行干预。甚至，我们还可以研究更合理的面向机器翻译解释性的方法，笔者相信这也是未来需要突破的点。

最后，简单谈谈笔者对机器翻译市场发展趋势的看法。机器翻译本身是个强刚需，用于解决全球用户多语言交流障碍的问题。机器翻译产业真正热起来，应该归功于神经机器翻译技术的应用，虽然基于规则的方法和统计机器翻译技术也在工业界得到了应用，但翻译品质没有达到用户预期，用户付费欲望比较差，没有良好的商业变现能力，导致机器翻译产业在早些年有些“鸡肋”。严格来说，近些年神经机器翻译技术在工业界的广泛应用快速激活了用户需求，用户对机器翻译的认可度急剧

上升，越来越丰富的应用模式和需求被挖掘出来。除了传统计算机辅助翻译，语音和OCR与机器翻译技术结合，使得语音翻译APP、翻译机、翻译笔、会议AI同传和垂直行业（专利、医药、旅游等）的机器翻译解决方案逐渐得到了广泛应用。总体来说，机器翻译产学研正处于快速上升期，市场规模每年都会有显著增长。随着多模态机器翻译和大数据翻译技术的应用，机器翻译的应用场景会越来越丰富。随着5G，甚至6G技术的发展，视频翻译和电话通信翻译等应用会进一步爆发。另外，随着人工智能芯片领域的发展，机器翻译芯片也会逐渐被广泛应用，如嵌入到手机、打印机、复印机、传真机和电视机等智能终端设备中，实现所有内容皆可翻译，任何场景皆可运行的愿景。机器翻译服务将进入人们的日常生活，无处不在，让生活更加美好！

朱靖波 肖桐

2020.12.16

于东北大学

# 后 记

我知道这里本应该再写点什么，感慨一下蹉跎岁月，最后致敬所有人。

不过我还是最想说：

谢谢你，我的妻子。没有你的支持与照顾，我应该没有勇气来完成这本书。爱你

~

肖桐

2020.12.27

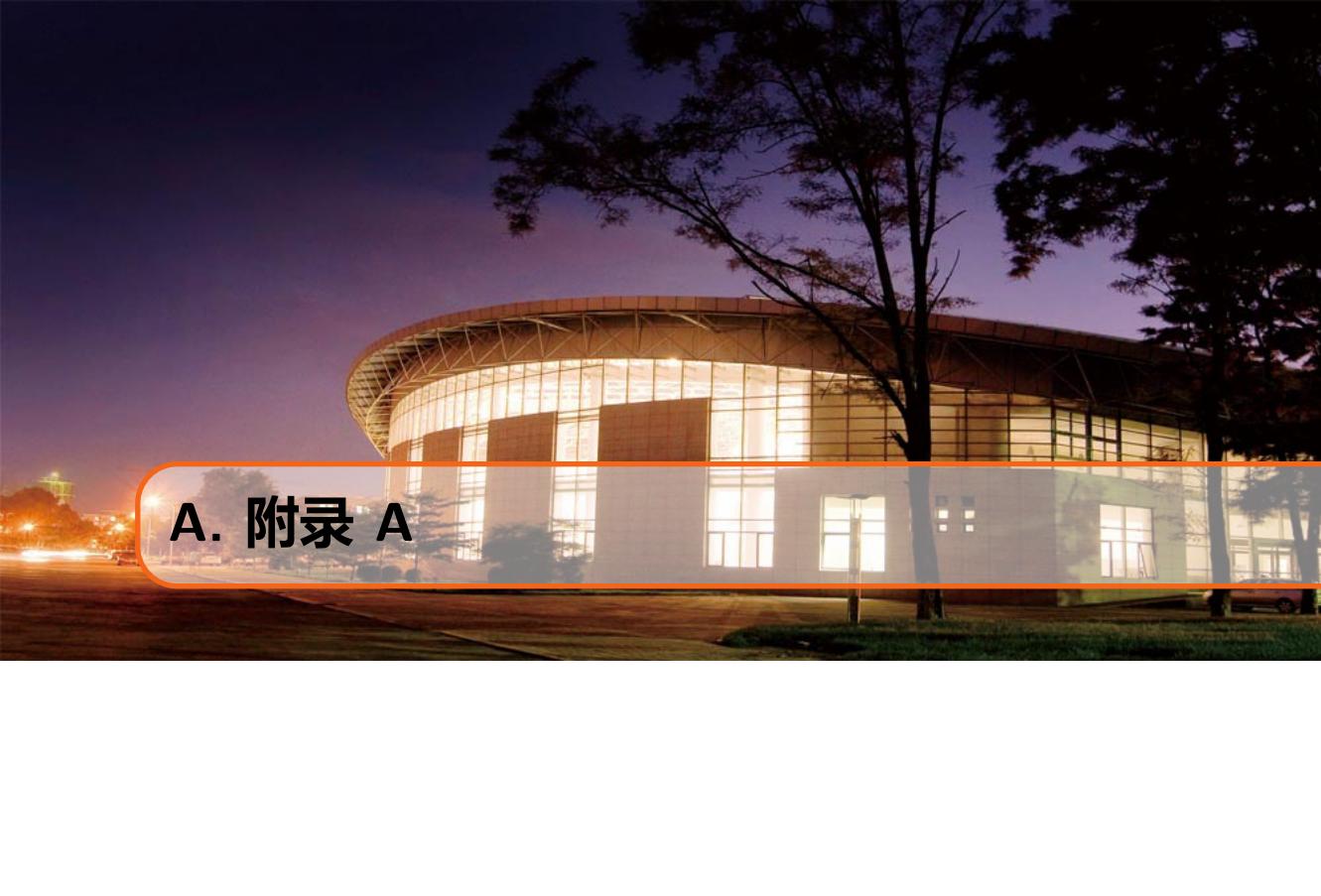


# 附录



<b>A</b>	<b>附录 A .....</b>	<b>597</b>
A.1	统计机器翻译开源系统	
A.2	神经机器翻译开源系统	
<b>B</b>	<b>附录 B .....</b>	<b>601</b>
B.1	公开评测任务	
B.2	基准数据集	
B.3	平行语料	
<b>C</b>	<b>附录 C .....</b>	<b>607</b>
C.1	IBM 模型 2 训练方法	
C.2	IBM 模型 3 训练方法	
C.3	IBM 模型 4 训练方法	
C.4	IBM 模型 5 训练方法	





## A. 附录 A

从实践的角度，机器翻译的发展离不开开源系统的推动作用。开源系统通过代码共享的方式使得最新的研究成果可以快速传播，同时实验结果可以复现。此外，开源项目也促进了不同团队之间的协作，让研究人员在同一个平台上集中力量攻关。

### A.1 统计机器翻译开源系统

- NiuTrans.SMT。NiuTrans<sup>[1191]</sup> 是由东北大学自然语言处理实验室自主研发的统计机器翻译系统，该系统可支持基于短语的模型、基于层次短语的模型以及基于句法的模型。由于使用 C++ 语言开发，所以该系统运行时间快，所占存储空间少。系统中内嵌有  $n$ -gram 语言模型，故无需使用其他的系统即可对完成语言建模。
- Moses。Moses<sup>[80]</sup> 是统计机器翻译时代最著名的系统之一，(主要)由爱丁堡大学的机器翻译团队开发。最新的 Moses 系统支持很多的功能，例如，它既支持基于短语的模型，也支持基于句法的模型。Moses 提供因子化翻译模型 (Factored Translation Model)，因此该模型可以很容易地对不同层次的信息进行建模。此外，它允许将混淆网络和字格作为输入，可缓解系统的 1-best 输出中的错误。Moses 还提供了很多有用的脚本和工具，被机器翻译研究人员广泛使用。
- Joshua。Joshua<sup>[1192]</sup> 是由约翰霍普金斯大学的语言和语音处理中心开发的层次短语翻译系统。由于 Joshua 是由 Java 语言开发，所以它在不同的平台上运行或

开发时具有良好的可扩展性和可移植性。Joshua 也是使用非常广泛的开源机器翻译系统之一。

- SilkRoad。SilkRoad 是由五个国内机构（中科院计算所、中科院软件所、中科院自动化所、厦门大学和哈尔滨工业大学）联合开发的基于短语的统计机器翻译系统。该系统是中国乃至亚洲地区第一个开源的统计机器翻译系统。SilkRoad 支持多种解码器和规则提取模块，这样可以组合成不同的系统，提供多样的选择。
- SAMT。SAMT<sup>[85]</sup> 是由卡内基梅隆大学机器翻译团队开发的语法增强的统计机器翻译系统。SAMT 在解码的时候使用目标树来生成翻译规则，而不严格遵守目标语言的语法。SAMT 的一个亮点是它提供了简单但高效的方式在机器翻译中使用句法信息。由于 SAMT 在 hadoop 中实现，它可受益于大数据集的分布式处理。
- HiFST。HiFST<sup>[1193]</sup> 是剑桥大学开发的统计机器翻译系统。该系统完全基于有限状态自动机实现，因此非常适合对搜索空间进行有效的表示。
- cdec。cdec<sup>[1194]</sup> 是一个强大的解码器，是由 Chris Dyer 和他的合作者们一起开发。cdec 的主要功能是它使用了翻译模型的一个统一的内部表示，并为结构预测问题的各种模型和算法提供了实现框架。所以，cdec 也可以被用来做一个对齐系统或者一个更通用的学习框架。此外，由于使用 C++ 语言编写，cdec 的运行速度较快。
- Phrasal。Phrasal<sup>[1195]</sup> 是由斯坦福大学自然语言处理小组开发的系统。除了传统的基于短语的模型，Phrasal 还支持基于非层次短语的模型，这种模型将基于短语的翻译延伸到非连续的短语翻译，增加了模型的泛化能力。
- Jane。Jane<sup>[1196]</sup> 是一个基于短语和基于层次短语的机器翻译系统，由亚琛工业大学的人类语言技术与模式识别小组开发。Jane 提供了系统融合模块，因此可以非常方便的对多个系统进行融合。
- GIZA++。GIZA++<sup>[240]</sup> 是 Franz Och 研发的用于训练 IBM 模型 1-5 和 HMM 单词对齐模型的工具包。在早期，GIZA++ 是所有统计机器翻译系统中词对齐的标配工具。
- FastAlign。FastAlign<sup>[250]</sup> 是一个快速，无监督的词对齐工具，由卡内基梅隆大学开发。

## A.2 神经机器翻译开源系统

- **GroundHog**。GroundHog<sup>[22]</sup> 基于 Theano<sup>[1197]</sup> 框架，由蒙特利尔大学 LISA 实验室使用 Python 语言编写的一个框架，旨在提供灵活而高效的方式来实现复杂的循环神经网络模型。它提供了包括 LSTM 在内的多种模型。Bahdanau 等人在此框架上又编写了 GroundHog 神经机器翻译系统。该系统也作为了很多论文的基线系统。
- **Nematus**。Nematus<sup>[469]</sup> 是英国爱丁堡大学开发的，基于 Theano 框架的神经机器翻译系统。该系统使用 GRU 作为隐层单元，支持多层网络。Nematus 编码端有正向和反向的编码方式，可以同时提取源语言句子中的上下文信息。该系统的一个优点是，它可以支持输入端有多个特征的输入（例如词的词性等）。
- **ZophRNN**。ZophRNN<sup>[1198]</sup> 是由南加州大学的 Barret Zoph 等人使用 C++ 语言开发的系统。Zoph 既可以训练序列表示模型（如语言模型），也可以训练序列到序列的模型（如神经机器翻译模型）。当训练神经机器翻译系统时，ZophRNN 也支持多源输入。
- **Fairseq**。Fairseq<sup>[814]</sup> 是由 Facebook 开发的，基于 PyTorch 框架的用以解决序列到序列问题的工具包，其中包括基于卷积神经网络、基于循环神经网络、基于 Transformer 的模型等。Fairseq 是当今使用最广泛的神经机器翻译开源系统之一。
- **Tensor2Tensor**。Tensor2Tensor<sup>[535]</sup> 是由谷歌推出的，基于 TensorFlow 框架的开源系统。该系统基于 Transformer 模型，因此可以支持大多数序列到序列任务。得益于 Transformer 的网络结构，系统的训练速度较快。现在，Tensor2Tensor 也是机器翻译领域广泛使用的开源系统之一。
- **OpenNMT**。OpenNMT<sup>[692]</sup> 系统是由哈佛大学自然语言处理研究组开源的，基于 Torch 框架的神经机器翻译系统。OpenNMT 系统的早期版本使用 Lua 语言编写，现在也扩展到了 TensorFlow 和 PyTorch，设计简单易用，易于扩展，同时保持效率和翻译精度。
- 斯坦福神经机器翻译开源代码库。斯坦福大学自然语言处理组（Stanford NLP）发布了一篇教程，介绍了该研究组在神经机器翻译上的研究信息，同时实现了多种翻译模型<sup>[562]</sup>。
- **THUMT**。清华大学 NLP 团队实现的神经机器翻译系统，支持 Transformer 等模型<sup>[1199]</sup>。该系统主要基于 TensorFlow 和 Theano 实现，其中 Theano 版本包含了 RNNsearch 模型，训练方式包括 MLE（Maximum Likelihood Estimate），MRT（Minimum Risk Training），SST（Semi-Supervised Training）。TensorFlow 版本实现了 Seq2Seq, RNNsearch, Transformer 三种基本模型。

- NiuTrans.NMT。由小牛翻译团队基于 NiuTensor 实现的神经机器翻译系统。支持循环神经网络、Transformer 等结构，并支持语言建模、序列标注、机器翻译等任务。支持机器翻译 GPU 与 CPU 训练及解码。其小巧易用，为开发人员提供快速二次开发基础。此外，NiuTrans.NMT 已经得到了大规模应用，形成了支持 304 种语言翻译的小牛翻译系统。
- MARIAN。主要由微软翻译团队搭建<sup>[1200]</sup>，其使用 C++ 实现的用于 GPU/CPU 训练和解码的引擎，支持多 GPU 训练和批量解码，最小限度依赖第三方库，静态编译一次之后，复制其二进制文件就能在其他平台使用。
- Sockeye。由 Awslabs 开发的神经机器翻译框架<sup>[1201]</sup>。其中支持 RNNsearch、Transformer、CNN 等翻译模型，同时提供了从图片翻译到文字的模块以及 WMT 德英新闻翻译、领域适应任务、多语言零资源翻译任务的教程。
- CytonMT。由 NICT 开发的一种用 C++ 实现的神经机器翻译开源工具包<sup>[1202]</sup>。主要支持 Transformer 模型，并支持一些常用的训练方法以及解码方法。
- OpenSeq2Seq。由 NVIDIA 团队开发的<sup>[1203]</sup> 基于 TensorFlow 的模块化架构，用于序列到序列的模型，允许从可用组件中组装新模型，支持混合精度训练，利用 NVIDIA Volta Turing GPU 中的 Tensor 核心，基于 Horovod 的快速分布式训练，支持多 GPU，多节点多模式。
- NMTPyTorch。由勒芒大学语言实验室发布的基于序列到序列框架的神经网络翻译系统<sup>[1204]</sup>，NMTPyTorch 的核心部分依赖于 Numpy，PyTorch 和 tqdm。其允许训练各种端到端神经体系结构，包括但不限于神经机器翻译、图像字幕和自动语音识别系统。



## B. 附录 B

除了开源系统，机器翻译的发展还离不开评测比赛。评测比赛使得各个研究组织的成果可以进行科学的对比，共同推动机器翻译的发展与进步。另外在构建机器翻译系统的过程中，数据是必不可少的，尤其是现在主流的神经机器翻译系统，系统的性能往往受限于语料库规模和质量。所幸的是，随着语料库语言学的发展，一些主流语种的相关语料资源已经十分丰富。

为了方便读者进行相关研究，本书汇总了几个常见的评测比赛、一些常用的基准数据集和常用的平行语料。

### B.1 公开评测任务

机器翻译相关评测主要有两种组织形式，一种是由政府及国家相关机构组织，权威性强。如由美国国家标准技术研究所组织的 NIST 评测、日本国家科学咨询系统中心主办的 NACSIS Test Collections for IR（NTCIR）PatentMT、日本科学振兴机构（Japan Science and Technology Agency，简称 JST）等组织联合举办的 Workshop on Asian Translation（WAT）以及国内由中文信息学会主办的全国机器翻译大会（China Conference on Machine Translation，简称 CCMT）；另一种是由相关学术机构组织，具有领域针对性的特点，如倾向新闻领域的 Conference on Machine Translation（WMT）以及面向口语的 International Workshop on Spoken Language Translation（IWSLT）。下面将针对上述评测进行简要介绍。

- CCMT。CCMT（全国机器翻译大会），前身为 CWMT（全国机器翻译研讨会）

是国内机器翻译领域的旗舰会议，自 2005 年起已经组织多次机器翻译评测，对国内机器翻译相关技术的发展产生了深远影响。该评测主要针对汉语、英语以及国内的少数民族语言（蒙古语、藏语、维吾尔语等）进行评测，领域包括新闻、口语、政府文件等，不同语言方向对应的领域也有所不同。评价方式不同届略有不同，主要采用自动评价的方式，自 CWMT 2013 起则针对某些领域增设人工评价。自动评价的指标一般包括 BLEU-SBP、BLEU-NIST、TER、METEOR、NIST、GTM、mWER、mPER 以及 ICT 等，其中以 BLEU-SBP 为主，汉语为目标语言的翻译采用基于字符的评价方式，面向英语的翻译采用基于词的评价方式。每年该评测吸引国内外近数十家企业及科研机构参赛，业内认可度极高。关于 CCMT 的更多信息可参考中文信息学会机器翻译专业委员会相关页面。

- WMT。WMT 由 Special Interest Group for Machine Translation (SIGMT) 主办，会议自 2006 年起每年召开一次，是一个涉及机器翻译多种任务的综合性会议，包括多领域翻译评测任务、质量评价任务以及其他与机器翻译的相关任务（如文档对齐评测等）。现在 WMT 已经成为机器翻译领域的旗舰评测会议，很多研究工作都以 WMT 评测结果作为基准。WMT 评测涉及的语言范围较广，包括英语、德语、芬兰语、捷克语、罗马尼亚语等十多种语言，翻译方向一般以英语为核心，探索英语与其他语言之间的翻译性能，领域包括新闻、信息技术、生物医学。最近，也增加了无指导机器翻译等热门问题。WMT 在评价方面类似于 CCMT，也采用人工评价与自动评价相结合的方式，自动评价的指标一般为 BLEU、TER 等。此外，WMT 公开了所有评测数据，因此也经常被机器翻译相关人员所使用。更多 WMT 的机器翻译评测相关信息可参考 SIGMT 官网。
- NIST。NIST 机器翻译评测开始于 2001 年，是早期机器翻译公开评测中颇具代表性的任务，现在 WMT 和 CCMT 很多任务的设置也大量参考了当年 NIST 评测的内容。NIST 评测由美国国家标准技术研究所主办，作为美国国防高级计划署 (DARPA) 中 TIDES 计划的重要组成部分。早期，NIST 评测主要评价阿拉伯语和汉语等语言到英语的翻译效果，评价方法一般采用人工评价与自动评价相结合的方式。人工评价采用 5 分制评价。自动评价使用多种方式，包括 BLEU、METEOR、TER 以及 HyTER。此外 NIST 从 2016 年起开始对稀缺语言资源技术进行评估，其中机器翻译作为其重要组成部分共同参与评测，评测指标主要为 BLEU。除对机器翻译系统进行评测之外，NIST 在 2008 和 2010 年对于机器翻译的自动评价方法 (MetricsMaTr) 也进行了评估，以鼓励更多研究人员对现有评价方法进行改进或提出更加贴合人工评价的方法。同时 NIST 评测所提供的数据集由于数据质量较高受到众多科研人员喜爱，如 MT04、MT06 等（汉英）平行语料经常被科研人员在实验中使用。不过，近几年 NIST 评测已经停止。更多 NIST 的机器翻译评测相关信息可参考官网。
- IWSLT。从 2004 年开始举办的 IWSLT 也是颇具特色的机器翻译评测，它主

要关注口语相关的机器翻译任务，测试数据包括 TED talks 的多语言字幕以及 QED 教育讲座影片字幕等，语言涉及英语、法语、德语、捷克语、汉语、阿拉伯语等众多语言。此外在 IWSLT 2016 中还加入了对于日常对话的翻译评测，尝试将微软 Skype 中一种语言的对话翻译成其他语言。评价方式采用自动评价的模式，评价标准和 WMT 类似，一般为 BLEU 等指标。另外，IWSLT 除了对文本到文本的翻译评测外，还有自动语音识别以及语音转另一种语言的文本的评测。更多 IWSLT 的机器翻译评测相关信息可参考 IWSLT 官网。

- WAT。日本举办的机器翻译评测 WAT 是亚洲范围内的一个重要评测之一，由日本科学振兴机构（JST）、情报通信研究机构（NICT）等多家机构共同组织，旨在为亚洲各国之间交流融合提供便宜之处。语言方向主要包括亚洲主流语言（汉语、韩语、印地语等）以及英语对日语的翻译，领域丰富多样，包括学术论文、专利、新闻、食谱等。评价方式包括自动评价（BLEU、RIBES 以及 AMFM 等）以及人工评价，其特点在于对于测试语料以段落为单位进行评价，考察其上下文关联的翻译效果。更多 WAT 的机器翻译评测相关信息可参考官网。
- NTCIR。NTCIR 计划是由日本国家科学咨询系统中心策划主办的，旨在建立一个用在自然语言处理以及信息检索相关任务上的日文标准测试集。在 NTCIR-9 和 NTCIR-10 中开设的 Patent Machine Translation（PatentMT）任务主要针对专利领域进行翻译测试，其目的在于促进机器翻译在专利领域的发展和应用。在 NTCIR-9 中，评测方式采取人工评价与自动评价相结合，以人工评价为主导。人工评价主要根据准确度和流畅度进行评估，自动评价采用 BLEU、NIST 等方式进行。NTCIR-10 评价方式在此基础上增加了专利审查评估、时间评估以及多语种评估，分别考察机器翻译系统在专利领域翻译的实用性、耗时情况以及不同语种的翻译效果等。更多 NTCIR 评测相关信息可参考官网。

以上评测数据大多可以从评测网站上下载，此外部分数据也可以从 LDC（Linguistic Data Consortium）上申请。ELRA（European Language Resources Association）上也有一些免费的语料库供研究使用。从机器翻译发展的角度看，这些评测任务给相关研究提供了基准数据集，使得不同的系统都可以在同一个环境下进行比较和分析，进而建立了机器翻译研究所需的实验基础。此外，公开评测也使得研究人员可以第一时间了解机器翻译研究的最新成果，比如，有多篇 ACL 会议最佳论文的灵感就来自当年参加机器翻译评测任务的系统。

## B.2 基准数据集

表B.1所展示的数据集已经在机器翻译领域中被广泛使用，有很多之前的相关工作可以进行复现和对比。

表 B.1 基准数据集

任务	语种	领域	描述
WMT	En-Zh、En-De 等	新闻、医学、翻译	以英语为核心的多语种机器翻译数据集, 涉及多种任务
IWSLT	En-De、En-Zh 等	口语翻译	文本翻译数据集来自 TED 演讲, 数据规模较小
NIST	Zh-En、En-Cs 等	新闻翻译	评测集包括 4 句参考译文, 质量较高
TVsub	Zh-En	字幕翻译	数据抽取自电视剧字幕, 用于对话中长距离上下文研究
Flickr30K	En-De	多模态翻译	31783 张图片, 每张图片 5 个语句标注
Multi30K	En-De、En-Fr	多模态翻译	31014 张图片, 每张图片 5 个语句标注
IAPRTC-12	En-De	多模态翻译	20000 张图片及对应标注
IKEA	En-De、En-Fr	多模态翻译	3600 张图片及对应标注

### B.3 平行语料

神经机器翻译系统的训练需要大量的双语数据, 这里本节汇总了一些公开的平行语料, 方便读者获取。

- **News Commentary Corpus:** 包括汉语、英语等 12 个语种, 64 个语言对的双语数据, 爬取自 Project Syndicate 网站的政治、经济评论。
- **CWMT Corpus:** 中国计算机翻译研讨会社区收集和共享的中英平行语料, 涵盖多种领域, 例如新闻、电影字幕、小说和政府文档等。
- **Common Crawl corpus:** 包括捷克语、德语、俄语、法语 4 种语言到英语的双语数据, 爬取自互联网网页。
- **Europarl Corpus:** 包括保加利亚语、捷克语等 20 种欧洲语言到英语的双语数据, 来源于欧洲议会记录。
- **ParaCrawl Corpus:** 包括 23 种欧洲语言到英语的双语语料, 数据来源于网络爬取。
- **United Nations Parallel Corpus:** 包括阿拉伯语、英语、西班牙语、法语、俄语、汉语 6 种联合国正式语言, 30 种语言对的双语数据, 来源自联合国公共领域的官方记录和其他会议文件。
- **TED Corpus:** TED 大会演讲在其网站公布了自 2007 年以来的演讲字幕, 以及超过 100 种语言的翻译版本。WIT 收集整理了这些数据, 以方便科研工作者使

用，同时，会为每年的 IWSLT 评测比赛提供评测数据集。

- **OpenSubtile:** 由 P. Lison 和 J. Tiedemann 收集自 opensubtitles 电影字幕网站，包含 62 种语言、1782 个语种对的平行语料，资源相对比较丰富。
- **Wikititles Corpus:** 包括古吉拉特语等 14 个语种，11 个语言对的双语数据，数据来源自维基百科的标题。
- **CzEng:** 捷克语和英语的平行语料，数据来源于欧洲法律、信息技术和小说领域。
- **Yandex Corpus:** 俄语和英语的平行语料，爬取自互联网网页。
- **Tilde MODEL Corpus:** 欧洲语言的多语言开放数据，包含多个数据集，数据来自于经济、新闻、政府、旅游等门户网站。
- **Setimes Corpus:** 包括克罗地亚语、阿尔巴尼亚等 9 种巴尔干语言，72 个语言对的双语数据，来源于东南欧时报的新闻报道。
- **TVsub:** 收集自电视剧集字幕的中英文对话语料库，包含超过 200 万的句对，可用于对话领域和长距离上下文信息的研究。
- **Recipe Corpus:** 由 Cookpad 公司创建的日英食谱语料库，包含 10 万多的句对。





## C. 附录 C

### C.1 IBM 模型 2 训练方法

IBM 模型 2 与模型 1 的训练过程完全一样，本质上都是 EM 方法，因此可以直接复用第五章中训练模型 1 的流程。对于源语言句子  $s = \{s_1, \dots, s_m\}$  和目标语言句子  $t = \{t_1, \dots, t_l\}$ ，E-Step 的计算公式如下：

$$c(s_u|t_v; s, t) = \sum_{j=1}^m \sum_{i=0}^l \frac{f(s_u|t_v)a(i|j, m, l)\delta(s_j, s_u)\delta(t_i, t_v)}{\sum_{k=0}^l f(s_u|t_k)a(k|j, m, l)} \quad (C.1)$$

$$c(i|j, m, l; s, t) = \frac{f(s_j|t_i)a(i|j, m, l)}{\sum_{k=0}^l f(s_j|t_k)a(k|j, m, l)} \quad (C.2)$$

M-Step 的计算公式如下：

$$f(s_u|t_v) = \frac{c(s_u|t_v; s, t)}{\sum_{s'_u} c(s'_u|t_v; s, t)} \quad (C.3)$$

$$a(i|j, m, l) = \frac{c(i|j, m, l; s, t)}{\sum_{i'} c(i'|j, m, l; s, t)} \quad (C.4)$$

其中， $f(s_u|t_v)$  与 IBM 模型 1 一样表示目标语言单词  $t_v$  到源语言单词  $s_u$  的翻译概率， $a(i|j, m, l)$  表示调序概率。

对于由  $K$  个样本组成的训练集  $\{(s^{[1]}, t^{[1]}), \dots, (s^{[K]}, t^{[K]})\}$ , 可以将 M-Step 的计算调整为:

$$f(s_u|t_v) = \frac{\sum_{k=1}^K c(s_u|t_v; s^{[k]}, t^{[k]})}{\sum_{s'_u} \sum_{k=1}^K c(s'_u|t_v; s^{[k]}, t^{[k]})} \quad (\text{C.5})$$

$$a(i|j, m, l) = \frac{\sum_{k=1}^K c(i|j, m^{[k]}, l^{[k]}; s^{[k]}, t^{[k]})}{\sum_{i'} \sum_{k=1}^K c(i'|j, m^{[k]}, l^{[k]}; s^{[k]}, t^{[k]})} \quad (\text{C.6})$$

其中,  $m^{[k]} = |s^{[k]}|$ ,  $l^{[k]} = |t^{[k]}|$ 。

## C.2 IBM 模型 3 训练方法

IBM 模型 3 的参数估计与模型 1 和模型 2 采用相同的方法, 辅助函数被定义如下:

$$\begin{aligned} h(t, d, n, p, \lambda, \mu, \nu, \zeta) &= P_\theta(s|t) - \sum_{t_v} \lambda_{t_v} \left( \sum_{s_u} t(s_u|t_v) - 1 \right) \\ &\quad - \sum_i \mu_{iml} \left( \sum_j d(j|i, m, l) - 1 \right) \\ &\quad - \sum_{t_v} \nu_{t_v} \left( \sum_\varphi n(\varphi|t_v) - 1 \right) - \zeta(p_0 + p_1 - 1) \end{aligned} \quad (\text{C.7})$$

这里略去推导步骤, 直接给出不同参数对应的期望频次计算公式, 如下:

$$c(s_u|t_v, s, t) = \sum_a [P_\theta(s, a|t) \times \sum_{j=1}^m (\delta(s_j, s_u) \cdot \delta(t_{aj}, t_v))] \quad (\text{C.8})$$

$$c(j|i, m, l; s, t) = \sum_a [P_\theta(s, a|t) \times \delta(i, a_j)] \quad (\text{C.9})$$

$$c(\varphi|t_v; s, t) = \sum_a [P_\theta(s, a|t) \times \sum_{i=1}^l \delta(\varphi, \varphi_i) \delta(t_v, t_i)] \quad (\text{C.10})$$

$$c(0|s, t) = \sum_a [P_\theta(s, a|t) \times (m - 2\varphi_0)] \quad (\text{C.11})$$

$$c(1|s, t) = \sum_a [P_\theta(s, a|t) \times \varphi_0] \quad (\text{C.12})$$

进一步，对于由  $K$  个样本组成的训练集，有：

$$t(s_u|t_v) = \lambda_{t_v}^{-1} \times \sum_{k=1}^K c(s_u|t_v; s^{[k]}, t^{[k]}) \quad (\text{C.13})$$

$$d(j|i, m, l) = \mu_{iml}^{-1} \times \sum_{k=1}^K c(j|i, m, l; s^{[k]}, t^{[k]}) \quad (\text{C.14})$$

$$n(\varphi|t_v) = \nu_{t_v}^{-1} \times \sum_{k=1}^K c(\varphi|t_v; s^{[k]}, t^{[k]}) \quad (\text{C.15})$$

$$p_x = \zeta^{-1} \sum_{k=1}^K c(x; s^{[k]}, t^{[k]}) \quad (\text{C.16})$$

在模型 3 中，因为繁衍率的引入，并不能像模型 1 那样，通过简单的数学技巧加速参数估计的过程（见第五章）。因此在计算公式(C.8)-(C.12)时，我们不得不面对大小为  $(l+1)^m$  的词对齐空间。遍历所有  $(l+1)^m$  个词对齐所带来的高时间复杂度显然是不能被接受的。因此就要考虑能否仅利用词对齐空间中的部分词对齐对这些参数进行估计。比较简单的方法是仅使用 Viterbi 对齐来进行参数估计，这里 Viterbi 词对齐可以被简单的看作搜索到的最好词对齐。遗憾的是，在模型 3 中并没有方法直接获得 Viterbi 对齐。这样只能采用一种折中的策略，即仅考虑那些使得  $P_\theta(s, a|t)$  达到较高值的词对齐。这里把这部分词对齐组成的集合记为  $S$ 。以公式(C.8)为例，它可以被修改为：

$$c(s_u|t_v, s, t) \approx \sum_{a \in S} [P_\theta(s, a|t) \times \sum_{j=1}^m (\delta(s_j, s_u) \cdot \delta(t_{a_j}, t_v))] \quad (\text{C.17})$$

可以以同样的方式修改公式(C.9)-(C.12)的修改结果。进一步，在 IBM 模型 3 中，可以定义  $S$  如下：

$$S = N(b^\infty(V(s|t; 2))) \cup (\bigcup_{ij} N(b_{i \leftrightarrow j}^\infty(V_{i \leftrightarrow j}(s|t; 2)))) \quad (\text{C.18})$$

为了理解这个公式，先介绍几个概念。

- $V(s|t)$  表示 Viterbi 词对齐， $V(s|t; 1)$ 、 $V(s|t; 2)$  和  $V(s|t; 3)$  就分别对应了模型 1、2 和 3 的 Viterbi 词对齐；
- 把那些满足第  $j$  个源语言单词对应第  $i$  个目标语言单词 ( $a_j = i$ ) 的词对齐构成的集合记为  $a_{i \leftrightarrow j}(s, t)$ 。通常称这些对齐中  $j$  和  $i$  被“钉”在了一起。在  $a_{i \leftrightarrow j}(s, t)$  中使  $P(s, a|t)$  达到最大的那个词对齐被记为  $V_{i \leftrightarrow j}(s|t)$ ；
- 如果两个词对齐，通过交换两个词对齐连接就能互相转化，则称它们为邻居。一个词对齐  $a$  的所有邻居记为  $N(a)$ 。

公式(C.18)中，应该使用  $V(s|t;3)$  和  $V_{i \leftrightarrow j}(s|t;3)$  进行计算，但其复杂度较高，因此使用  $b^\infty(V(s|t;2))$  和  $b_{i \leftrightarrow j}^\infty(V_{i \leftrightarrow j}(s|t;2))$  分别对  $V(s|t;3)$  和  $V_{i \leftrightarrow j}(s|t;3)$  进行估计。在计算  $S$  的过程中，需要知道一个对齐  $a$  的邻居  $a'$  的概率，即通过  $P_\theta(a, s|t)$  计算  $P_\theta(a', s|t)$ 。在模型 3 中，如果  $a$  和  $a'$  仅区别于某个源语言单词  $s_j$  对齐从  $a_j$  变到  $a'_j$ ，且  $a_j$  和  $a'_j$  均不为零，令  $a_j = i$ ,  $a'_j = i'$ , 那么

$$\begin{aligned} P_\theta(a', s|t) &= P_\theta(a, s|t) \cdot \\ &\quad \frac{\varphi_{i'} + 1}{\varphi_i} \cdot \frac{n(\varphi_{i'} + 1|t_{i'})}{n(\varphi_i|t_i)} \cdot \frac{n(\varphi_i - 1|t_i)}{n(\varphi_{i'}|t_{i'})} \cdot \\ &\quad \frac{t(s_j|t_{i'})}{t(s_j|t_i)} \cdot \frac{d(j|i', m, l)}{d(j|i, m, l)} \end{aligned} \quad (\text{C.19})$$

如果  $a$  和  $a'$  区别于两个位置  $j_1$  和  $j_2$  的对齐，即  $a_{j_1} = a'_{j_2}$  且  $a_{j_2} = a'_{j_1}$ ，那么

$$\begin{aligned} P_\theta(a', s|t) &= P_\theta(a, s|t) \cdot \\ &\quad \frac{t(s_{j_1}|t_{a_{j_2}})}{t(s_{j_1}|t_{a_{j_1}})} \cdot \frac{t(s_{j_2}|t_{a_{j_1}})}{t(s_{j_2}|t_{a_{j_2}})} \cdot \\ &\quad \frac{d(j_1|a_{j_2}, m, l)}{d(j_1|a_{j_1}, m, l)} \cdot \frac{d(j_2|a_{j_1}, m, l)}{d(j_2|a_{j_2}, m, l)} \end{aligned} \quad (\text{C.20})$$

相比整个词对齐空间， $S$  只是一个非常小的子集，因此计算时间可以被大大降低。可以看到，模型 3 的参数估计过程是建立在模型 1 和模型 2 的参数估计结果上的。这不仅是因为模型 3 要利用模型 2 的 Viterbi 对齐，而且还因为模型 3 参数的初值也要直接利用模型 2 的参数。从这个角度说，模型 1、2、3 是有序的且向前依赖的。单独的对模型 3 的参数进行估计是较为困难的。实际上 IBM 的模型 4 和模型 5 也具有这样的性质，即它们都可以利用前一个模型参数估计的结果作为自身参数的初始值。

### C.3 IBM 模型 4 训练方法

模型 4 的参数估计基本与模型 3 一致。需要修改的是扭曲度的估计公式，对于目标语言的第  $i$  个 cept. 生成的第一单词，可以得到（假设有  $K$  个训练样本）：

$$d_1(\Delta_j|ca, cb) = \mu_{1cacb}^{-1} \times \sum_{k=1}^K c_1(\Delta_j|ca, cb; s^{[k]}, t^{[k]}) \quad (\text{C.21})$$

其中，

$$c_1(\Delta_j|ca, cb; s, t) = \sum_a [P_\theta(s, a|t) \times z_1(\Delta_j|ca, cb; a, s, t)] \quad (\text{C.22})$$

$$\begin{aligned} z_1(\Delta_j|ca, cb; a, s, t) &= \sum_{i=1}^l [\varepsilon(\varphi_i) \cdot \delta(\pi_{i1} - \odot_i, \Delta_j) \cdot \\ &\quad \delta(A(t_{i-1}), ca) \cdot \delta(B(\tau_{i1}), cb)] \end{aligned} \quad (\text{C.23})$$

且

$$\varepsilon(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \quad (\text{C.24})$$

对于目标语言的第  $i$  个 cept. 生成的其他单词（非第一个单词），可以得到：

$$d_{>1}(\Delta_j|cb) = \mu_{>1cb}^{-1} \times \sum_{k=1}^K c_{>1}(\Delta_j|cb; s^{[k]}, t^{[k]}) \quad (\text{C.25})$$

其中，

$$c_{>1}(\Delta_j|cb; s, t) = \sum_a [P_\theta(s, a|t) \times z_{>1}(\Delta_j|cb; a, s, t)] \quad (\text{C.26})$$

$$\begin{aligned} z_{>1}(\Delta_j|cb; a, s, t) &= \sum_{i=1}^l [\varepsilon(\varphi_i - 1) \sum_{k=2}^{\varphi_i} \delta(\pi_{[i]k} - \pi_{[i]k-1}, \Delta_j) \cdot \\ &\quad \delta(B(\tau_{[i]k}), cb)] \end{aligned} \quad (\text{C.27})$$

这里， $ca$  和  $cb$  分别表示目标语言和源语言的某个词类。注意，在公式(C.23)和(C.27)中，求和操作  $\sum_{i=1}^l$  是从  $i = 1$  开始计算，而不是从  $i = 0$ 。这实际上跟 IBM 模型 4 的定义相关，因为  $d_1(j - \odot_{i-1}|A(t_{[i-1]}), B(s_j))$  和  $d_{>1}(j - \pi_{[i]k-1}|B(s_j))$  是从  $[i] > 0$  开始定义的，详细信息可以参考第六章的内容。

模型 4 需要像模型 3 一样，通过定义一个词对齐集合  $S$ ，使得每次训练迭代都在  $S$  上进行，进而降低运算量。模型 4 中  $S$  的定义为：

$$S = N(\tilde{b}^\infty(V(s|t; 2))) \cup (\bigcup_{ij} N(\tilde{b}_{i \leftrightarrow j}^\infty(V_{i \leftrightarrow j}(s|t; 2)))) \quad (\text{C.28})$$

对于一个对齐  $a$ , 可用模型 3 对它的邻居进行排名, 即按  $P_\theta(b(a)|s, t; 3)$  排序, 其中  $b(a)$  表示  $a$  的邻居。 $\tilde{b}(a)$  表示这个排名表中满足  $P_\theta(a'|s, t; 4) > P_\theta(a|s, t; 4)$  的最高排名的  $a'$ 。同理可知  $\tilde{b}_{i \leftrightarrow j}^\infty(a)$  的意义。这里之所以不用模型 3 中采用的方法直接利用  $b^\infty(a)$  得到模型 4 中高概率的对齐, 是因为模型 4 中要想获得某个对齐  $a$  的邻居  $a'$  必须做很大调整, 比如: 调整  $\tau_{[i]1}$  和  $\odot_i$  等等。这个过程要比模型 3 的相应过程复杂得多。因此在模型 4 中只能借助于模型 3 的中间步骤来进行参数估计。

## C.4 IBM 模型 5 训练方法

模型 5 的参数估计过程也和模型 4 的过程基本一致, 二者的区别在于扭曲度的估计公式。在模型 5 中, 对于目标语言的第  $i$  个 cept. 生成的第一单词, 可以得到 (假设有  $K$  个训练样本):

$$d_1(\Delta_j|cb) = \mu_{1cb}^{-1} \times \sum_{k=1}^K c_1(\Delta_j|cb; s^{[k]}, t^{[k]}) \quad (\text{C.29})$$

其中,

$$\begin{aligned} c_1(\Delta_j|cb, v_x, v_y; s, t) &= \sum_a \left[ P(s, a|t) \times z_1(\Delta_j|cb, v_x, v_y; a, s, t) \right] \\ z_1(\Delta_j|cb, v_x, v_y; a, s, t) &= \sum_{i=1}^l \left[ \varepsilon(\varphi_i) \cdot \delta(v_{\pi_{i1}}, \Delta_j) \cdot \delta(v_{\odot_{i-1}}, v_x) \right. \\ &\quad \left. \cdot \delta(v_m - \varphi_i + 1, v_y) \cdot \delta(v_{\pi_{i1}}, v_{\pi_{i1}-1}) \right] \end{aligned} \quad (\text{C.30})$$

对于目标语言的第  $i$  个 cept. 生成的其他单词 (非第一个单词), 可以得到:

$$d_{>1}(\Delta_j|cb, v) = \mu_{>1cb}^{-1} \times \sum_{k=1}^K c_{>1}(\Delta_j|cb, v; s^{[k]}, t^{[k]}) \quad (\text{C.32})$$

其中,

$$\begin{aligned} c_{>1}(\Delta_j|cb, v; s, t) &= \sum_a \left[ P(a, s|t) \times z_{>1}(\Delta_j|cb, v; a, s, t) \right] \\ z_{>1}(\Delta_j|cb, v; a, s, t) &= \sum_{i=1}^l \left[ \varepsilon(\varphi_i - 1) \sum_{k=2}^{\varphi_i} \left[ \delta(v_{\pi_{ik}} - v_{\pi_{[i]k}-1}, \Delta_j) \right. \right. \\ &\quad \left. \cdot \delta(B(\tau_{[i]k}), cb) \cdot \delta(v_m - v_{\pi_{i(k-1)}} - \varphi_i + k, v) \right. \\ &\quad \left. \cdot \delta(v_{\pi_{i1}}, v_{\pi_{i1}-1}) \right] \end{aligned} \quad (\text{C.33})$$

从公式(C.30)中可以看出，因子  $\delta(v_{\pi_{i1}}, v_{\pi_{i1-1}})$  保证了，即使对齐  $a$  不合理（一个源语言位置对应多个目标语言位置）也可以避免在这个不合理的对齐上计算结果。也就是因子  $\delta(v_{\pi_{p1}}, v_{\pi_{p1-1}})$  确保了  $a$  中不合理的部分不产生坏的影响，而  $a$  中其他正确的部分仍会参与迭代。

不过上面的参数估计过程与 IBM 前 4 个模型的参数估计过程并不完全一样。IBM 前 4 个模型在每次迭代中，可以在给定  $s$ 、 $t$  和一个对齐  $a$  的情况下直接计算并更新参数。但是在模型 5 的参数估计过程中（如公式(C.30)），需要模拟出由  $t$  生成  $s$  的过程才能得到正确的结果，因为从  $t$ 、 $s$  和  $a$  中是不能直接得到的正确结果的。具体说，就是要从目标语言句子的第一个单词开始到最后一个单词结束，依次生成每个目标语言单词对应的源语言单词，每处理完一个目标语言单词就要暂停，然后才能计算公式(C.30)中求和符号里面的内容。

从前面的分析可以看出，虽然模型 5 比模型 4 更精确，但是模型 5 过于复杂以至于给参数估计增加了计算量（对于每组  $t$ 、 $s$  和  $a$  都要模拟  $t$  生成  $s$  的翻译过程）。因此模型 5 的系统实现是一个挑战。

在模型 5 中同样需要定义一个词对齐集合  $S$ ，使得每次迭代都在  $S$  上进行。可以对  $S$  进行如下定义

$$S = N(\tilde{b}^\infty(V(s|t;2))) \cup (\bigcup_{ij} N(\tilde{b}_{i \leftrightarrow j}^\infty(V_{i \leftrightarrow j}(s|t;2)))) \quad (C.35)$$

其中， $\tilde{b}(a)$  借用了模型 4 中  $\tilde{b}(a)$  的概念。不过  $\tilde{b}(a)$  表示在利用模型 3 进行排名的列表中满足  $P_\theta(a'|s,t;5)$  的最高排名的词对齐，这里  $a'$  表示  $a$  的邻居。





## Bibliography

- [1] 慧立, 彦悰, and 道宣. 大慈恩寺三藏法師傳. Volume 2. 中华书局, 2000 (cited on page 26).
- [2] 中国翻译协会. 2019 中国语言服务行业发展报告. 中国翻译协会, 2019 (cited on page 26).
- [3] 姚恺璇 赵军峰. “深化改革探讨创新推进发展——全国翻译专业学位研究生教育 2019 年会综述”. In: 中国翻译, 2019 (cited on page 26).
- [4] James Knowlson. *Universal Language Schemes in England and France 1600-1800*. University of Toronto Press, 1975 (cited on page 26).
- [5] Claude E. Shannon. “A mathematical theory of communication”. In: volume 27. 3. Bell System Technical Journal, 1948, pages 379–423 (cited on page 27).
- [6] Claude E. Shannon and Warren Weaver. “The mathematical theory of communication”. In: volume 13. IEEE Transactions on Instrumentation and Measurement, 1949 (cited on page 27).
- [7] Warren Weaver. “Translation”. In: volume 14. 15-23. Cambridge: Technology Press, MIT, 1955, page 10 (cited on page 27).
- [8] Noam Chomsky. “Syntactic Structures”. In: volume 33. 3. Language, 1957 (cited on pages 27, 96).

- [9] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. “A Statistical Approach to Machine Translation”. In: volume 16. 2. Computational Linguistics, 1990, pages 79–85 (cited on pages 28, 29, 38, 154).
- [10] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. “The Mathematics of Statistical Machine Translation: Parameter Estimation”. In: volume 19. 2. Computational Linguistics, 1993, pages 263–311 (cited on pages 28, 29, 139, 140, 154, 155, 171, 173, 176, 180, 217).
- [11] Makoto Nagao. “A framework of a mechanical translation between Japanese and English by analogy principle”. In: Artificial and human intelligence, 1984, pages 351–354 (cited on pages 29, 37).
- [12] Satoshi Sato and Makoto Nagao. “Toward Memory-based Translation”. In: International Conference on Computational Linguistics, 1990, pages 247–252 (cited on page 29).
- [13] Sergei Nirenburg. “Knowledge-based machine translation”. In: volume 4. 1. Springer, 1989, pages 5–24 (cited on page 33).
- [14] William John Hutchins. *Machine translation: past, present, future*. Ellis Horwood Chichester, 1986 (cited on page 33).
- [15] Michael Zarechnak. “The history of machine translation”. In: volume 1979. Machine Translation, 1979, pages 1–87 (cited on page 33).
- [16] 冯志伟. 机器翻译研究. 中国对外翻译出版公司, 2004 (cited on page 34).
- [17] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall, Pearson Education International, 2009 (cited on pages 34, 64, 140).
- [18] 姚天顺 王宝库 张中义. “机器翻译系统中一种规则描述语言 (CTRDL)”. In: volume 5. 4. 中文信息学报, 1991 (cited on page 37).
- [19] 姚天顺 唐泓英. “基于搭配词典的词汇语义驱动算法”. In: volume 6. A01. 软件学报, 1995, pages 78–85 (cited on page 37).
- [20] William A. Gale and Kenneth W. Church. “A program for aligning sentences in bilingual corpora”. In: volume 19. 1. Computational Linguistics, 1993, pages 75–102 (cited on page 38).
- [21] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: Advances in Neural Information Processing Systems, 2014, pages 3104–3112 (cited on pages 39, 332, 342).

- [22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: International Conference on Learning Representations, 2015 (cited on pages 39, 187, 327, 332, 342, 352, 357, 369, 382, 453, 599).
- [23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is All You Need”. In: International Conference on Neural Information Processing, 2017, pages 5998–6008 (cited on pages 39, 75, 187, 322, 333, 336, 369, 393, 394, 407, 488, 499).
- [24] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. “Convolutional Sequence to Sequence Learning”. In: volume 70. International Conference on Machine Learning, 2017, pages 1243–1252 (cited on pages 39, 333, 336, 371, 378).
- [25] Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2015, pages 1412–1421 (cited on pages 39, 342, 352, 369, 382, 480).
- [26] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010 (cited on page 41).
- [27] Philipp Koehn. *Neural Machine Translation*. Cambridge University Press, 2020 (cited on page 41).
- [28] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. Massachusetts Institute of Technology Press, 1999 (cited on page 41).
- [29] 宗成庆. 统计自然语言处理. 清华大学出版社, 2013 (cited on page 42).
- [30] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. MIT Press, 2016 (cited on pages 42, 327).
- [31] Yoav Goldberg. “Neural network methods for natural language processing”. In: volume 10. 1. Morgan & Claypool Publishers, 2017, pages 1–309 (cited on pages 42, 327).
- [32] 周志华. 机器学习. 清华大学出版社, 2016 (cited on pages 42, 93).
- [33] 李航. 统计学习方法. 清华大学出版社, 2019 (cited on pages 42, 93, 94).
- [34] 邱锡鹏. 神经网络与深度学习. 机械工业出版社, 2020 (cited on page 42).
- [35] 魏宗舒. 概率论与数理统计教程: 第二版. 北京: 高等教育出版社, 2011 (cited on page 46).

- [36] Andre Nikolaevich Kolmogorov and Albert T Bharucha-Reid. *Foundations of the theory of probability: Second English Edition*. Courier Dover Publications, 2018 (cited on page 46).
- [37] 刘克. 实用马尔可夫决策过程. 清华大学出版社, 2004 (cited on page 57).
- [38] A. Barbour and Sidney Resnick. “Adventures in Stochastic Processes.” In: volume 88. Journal of the American Statistical Association, Dec. 1993, page 1474 (cited on page 57).
- [39] Irving J Good. “The population frequencies of species and the estimation of population parameters”. In: volume 40. 3-4. Oxford University Press, 1953, pages 237–264 (cited on page 61).
- [40] William A. Gale and Geoffrey Sampson. “Good-Turing Frequency Estimation Without Tears”. In: volume 2. 3. Journal of Quantitative Linguistics, 1995, pages 217–237 (cited on page 61).
- [41] Reinhard Kneser and Hermann Ney. “Improved backing-off for M-gram language modeling”. In: International Conference on Acoustics, Speech, and Signal Processing, 1995, pages 181–184 (cited on page 62).
- [42] Stanley F. Chen and Joshua Goodman. “An empirical study of smoothing techniques for language modeling”. In: volume 13. 4. Computer Speech & Language, 1999, pages 359–393 (cited on pages 62, 64, 74).
- [43] Hermann Ney and Ute Essén. “On smoothing techniques for bigram-based natural language modelling”. In: International Conference on Acoustics, Speech, and Signal Processing, 1991, pages 825–828 (cited on page 62).
- [44] Hermann Ney, Ute Essén, and Reinhard Kneser. “On structuring probabilistic dependences in stochastic language modelling”. In: volume 8. 1. Computer Speech & Language, 1994, pages 1–38 (cited on pages 62, 64).
- [45] Kenneth Heafield. “KenLM: Faster and Smaller Language Model Queries”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 187–197 (cited on pages 64, 75).
- [46] Andreas Stolcke. “SRILM - an extensible language modeling toolkit”. In: International Conference on Spoken Language Processing, 2002 (cited on page 64).
- [47] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989 (cited on page 66).
- [48] Shimon Even. *Graph algorithms*. Cambridge University Press, 2011 (cited on page 69).

- [49] Robert Endre Tarjan. “Depth-First Search and Linear Graph Algorithms”. In: volume 1. 2. SIAM Journal on Computing, 1972, pages 146–160 (cited on page 69).
- [50] Ashish Sabharwal and Bart Selman. “S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Third Edition”. In: volume 175. 5-6. Artificial Intelligence, 2011, pages 935–937 (cited on page 70).
- [51] Sartaj Sahni and Ellis Horowitz. *Fundamentals of Computer Algorithms*. Computer Science Press, 1978 (cited on page 70).
- [52] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: volume 4. 2. IEEE Transactions on Systems Science and Cybernetics, 1968, pages 100–107 (cited on page 72).
- [53] Bruce T. Lowerre. *The HARPY speech recognition system*. Carnegie Mellon University, 1976 (cited on page 72).
- [54] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995 (cited on page 72).
- [55] Karl Johan Åström. “Optimal control of Markov processes with incomplete state information”. In: volume 10. 1. Journal of Mathematical Analysis and Applications, 1965, pages 174–205 (cited on page 72).
- [56] Richard E. Korf. “Real-time heuristic search”. In: volume 42. 2. Artificial Intelligence, 1990, pages 189–211 (cited on page 72).
- [57] Liang Huang, Kai Zhao, and Mingbo Ma. “When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size)”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 2134–2139 (cited on pages 73, 453).
- [58] Yilin Yang, Liang Huang, and Mingbo Ma. “Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 3054–3059 (cited on pages 73, 453, 455).
- [59] F. Jelinek. “Interpolated estimation of Markov source parameters from sparse data”. In: Pattern Recognition in Practice, 1980, pages 381–397 (cited on page 74).
- [60] S. Katz. “Estimation of probabilities from sparse data for the language model component of a speech recognizer”. In: volume 35. 3. International Conference on Acoustics, Speech and Signal Processing, 1987, pages 400–401 (cited on page 74).
- [61] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text compression*. Prentice Hall, 1990 (cited on page 74).

- [62] I.H. Witten and T.C. Bell. “The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression”. In: volume 37. 4. IEEE Transactions on Information Theory, 1991, pages 1085–1094 (cited on page 74).
- [63] Joshua T. Goodman. “A bit of progress in language modeling”. In: volume 15. 4. Computer Speech & Language, 2001, pages 403–434 (cited on page 74).
- [64] Katrin Kirchhoff and Mei Yang. “Improved Language Modeling for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 125–128 (cited on page 75).
- [65] Ruhi Sarikaya and Yonggang Deng. “Joint Morphological-Lexical Language Modeling for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 145–148 (cited on page 75).
- [66] Philipp Koehn and Hieu Hoang. “Factored Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 868–876 (cited on page 75).
- [67] Marcello Federico and Mauro Cettolo. “Efficient Handling of N-gram Language Models for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 88–95 (cited on page 75).
- [68] Marcello Federico and Nicola Bertoldi. “How Many Bits Are Needed To Store Probabilities for Phrase-Based Translation?” In: Annual Meeting of the Association for Computational Linguistics, 2006, pages 94–101 (cited on page 75).
- [69] David Talbot and Miles Osborne. “Smoothed Bloom Filter Language Models: Tera-Scale LMs on the Cheap”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 468–476 (cited on page 75).
- [70] David Talbot and Miles Osborne. “Randomised Language Modelling for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 512–519 (cited on page 75).
- [71] Kun Jing and Jungang Xu. “A Survey on Neural Network Language Models.” In: arXiv preprint arXiv:1906.03591, 2019 (cited on page 75).
- [72] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. “A neural probabilistic language model”. In: volume 3. 6. Journal of Machine Learning Research, 2003, pages 1137–1155 (cited on pages 75, 121, 274, 318).
- [73] Tomas Mikolov, Martin Karafiat, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. “Recurrent neural network based language model”. In: International Speech Communication Association, 2010, pages 1045–1048 (cited on pages 75, 274, 321).

- [74] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. “LSTM Neural Networks for Language Modeling”. In: International Speech Communication Association, 2012, pages 194–197 (cited on page 75).
- [75] Franz Josef Och, Nicola Ueffing, and Hermann Ney. “An Efficient A\* Search Algorithm for Statistical Machine Translation”. In: Proceedings of the ACL Workshop on Data-Driven Methods in Machine Translation, 2001 (cited on page 75).
- [76] Ye-Yi Wang and Alex Waibel. “Decoding Algorithm in Statistical Machine Translation”. In: Morgan Kaufmann Publishers, 1997, pages 366–372 (cited on pages 75, 217).
- [77] Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. “A DP-based Search Using Monotone Alignments in Statistical Translation”. In: Morgan Kaufmann Publishers, 1997, pages 289–296 (cited on pages 75, 216).
- [78] Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. “Fast Decoding and Optimal Decoding for Machine Translation”. In: Morgan Kaufmann Publishers, 2001, pages 228–235 (cited on pages 75, 170).
- [79] Ulrich Germann. “Greedy decoding for statistical machine translation in almost linear time”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 1–8 (cited on pages 75, 170).
- [80] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. “Moses: Open Source Toolkit for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on pages 75, 187, 205, 216, 448, 451, 597).
- [81] Philipp Koehn. “Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models”. In: volume 3265. Springer, 2004, pages 115–124 (cited on pages 75, 186, 216, 448).
- [82] S. Bangalore and G. Riccardi. “A finite-state approach to machine translation”. In: Annual Meeting of the Association for Computational Linguistics, 2001, pages 381–388 (cited on page 75).
- [83] Srinivas Bangalore and Giuseppe Riccardi. “Stochastic Finite-State Models for Spoken Language Machine Translation”. In: volume 17. 3. Machine Translation, 2002, pages 165–184 (cited on page 75).

- [84] Ashish Venugopal, Andreas Zollmann, and Vogel Stephan. “An Efficient Two-Pass Approach to Synchronous-CFG Driven Statistical MT”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 500–507 (cited on page 75).
- [85] Andreas Zollmann, Ashish Venugopal, Matthias Paulik, and Stephan Vogel. “The Syntax Augmented MT (SAMT) System at the Shared Task for the 2007 ACL Workshop on Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 216–219 (cited on pages 75, 598).
- [86] Yang Liu, Qun Liu, and Shouxun Lin. “Tree-to-String Alignment Template for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on pages 75, 239, 264).
- [87] Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. “Scalable Inference and Training of Context-Rich Syntactic Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on pages 75, 239, 245, 264).
- [88] David Chiang. “A Hierarchical Phrase-Based Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 263–270 (cited on pages 75, 224, 264).
- [89] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on pages 79, 415, 416, 455, 521).
- [90] 刘挺, 吴岩, and 王开铸. “最大概率分词问题及其解法”. In: 06. 哈尔滨工业大学学报, 1998, pages 37–41 (cited on page 83).
- [91] 丁洁. “基于最大概率分词算法的中文分词方法研究”. In: 21. 科技信息, 2010, pages I0075–I0075 (cited on page 83).
- [92] Richard Bellman. “Dynamic programming”. In: volume 153. 3731. Science, 1966, pages 34–37 (cited on page 83).
- [93] Kevin Humphreys, Robert J. Gaizauskas, Saliha Azzam, Charles Huyck, Brian Mitchell, Hamish Cunningham, and Yorick Wilks. *University of Sheffield: Description of the LaSIE-II system as used for MUC-7*. Annual Meeting of the Association for Computational Linguistics, 1995 (cited on page 85).
- [94] George Krupka and Kevin Hausman. “IsoQuest Inc.: Description of the NetOwl™ Extractor System as Used for MUC-7”. In: Annual Meeting of the Association for Computational Linguistics, 1998 (cited on page 85).

- [95] William J Black, Fabio Rinaldi, and David Mowatt. “FACILE: Description of the NE System Used for MUC-7”. In: Annual Meeting of the Association for Computational Linguistics, 1998 (cited on page 85).
- [96] Sean R Eddy. “Hidden Markov models.” In: volume 6. 3. Current Opinion in Structural Biology, 1996, pages 361–5 (cited on pages 85, 87).
- [97] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: proceedings of the Eighteenth International Conference on Machine Learning, 2001, pages 282–289 (cited on pages 85, 91, 92, 103).
- [98] Jagat Narain Kapur. *Maximum-entropy models in science and engineering*. John Wiley & Sons, 1989 (cited on page 85).
- [99] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. “Support vector machines”. In: volume 13. 4. IEEE Intelligent Systems & Their Applications, 1998, pages 18–28 (cited on page 85).
- [100] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. “Natural Language Processing (almost) from Scratch”. In: volume 12. 1. Journal of Machine Learning Research, 2011, pages 2493–2537 (cited on pages 85, 327, 389, 390, 524).
- [101] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. “Neural Architectures for Named Entity Recognition”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 260–270 (cited on page 86).
- [102] Leonard E Baum and Ted Petrie. “Statistical Inference for Probabilistic Functions of Finite State Markov Chains”. In: volume 37. 6. Annals of Mathematical Stats, 1966, pages 1554–1563 (cited on page 87).
- [103] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”. In: volume 41. 1. Annals of Mathematical Stats, 1970, pages 164–171 (cited on pages 87, 89).
- [104] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: volume 39. 1. Journal of the Royal Statistical Society: Series B (Methodological), 1977, pages 1–22 (cited on page 89).
- [105] Andrew Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: volume 13. 2. IEEE Transactions on Information Theory, 1967, pages 260–269 (cited on page 89).

- [106] Peter Harrington. “机器学习实战”. In: 人民邮电出版社, 北京, 2013 (cited on page 94).
- [107] Andrew Y Ng and Michael I Jordan. “On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes”. In: MIT Press, 2001, pages 841–848 (cited on page 103).
- [108] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008 (cited on page 103).
- [109] Adam Berger, Stephen A Della Pietra, and Vincent J Della Pietra. “A maximum entropy approach to natural language processing”. In: volume 22. 1. Computational linguistics, 1996, pages 39–71 (cited on page 103).
- [110] Tom Mitchell. *Machine Learning*. McCraw Hill, 1996 (cited on page 103).
- [111] Franz Josef Och and Hermann Ney. “Discriminative Training and Maximum Entropy Models for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2002, pages 295–302 (cited on pages 103, 197).
- [112] Liang Huang. “Coling 2008: Advanced Dynamic Programming in Computational Linguistics: Theory, Algorithms and Applications-Tutorial notes”. In: International Conference on Computational Linguistics, 2008 (cited on page 103).
- [113] Mehryar Mohri, Fernando Pereira, and Michael Riley. “Speech recognition with weighted finite-state transducers”. In: Springer, 2008, pages 559–584 (cited on page 103).
- [114] Alfred V Aho and Jeffrey D Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall Englewood Cliffs, NJ, 1973 (cited on page 103).
- [115] Thorsten Brants. “TnT - A Statistical Part-of-Speech Tagger”. In: Annual Meeting of the Association for Computational Linguistics, 2000, pages 224–231 (cited on page 104).
- [116] Yoshimasa Tsuruoka and Jun’ichi Tsujii. “Chunk Parsing Revisited”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 133–140 (cited on page 104).
- [117] Sujian Li, Houfeng Wang, Shiwen Yu, and Chengsheng Xin. “News-Oriented Automatic Chinese Keyword Indexing”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 92–97 (cited on page 104).
- [118] Noam Chomsky. *Lectures on government and binding: The Pisa lectures*. Walter de Gruyter, 1993 (cited on page 104).

- [119] Zhiheng Huang, Wei Xu, and Kai Yu. “Bidirectional LSTM-CRF Models for Sequence Tagging”. In: CoRR, 2015 (cited on page 104).
- [120] Jason PC Chiu and Eric Nichols. “Named entity recognition with bidirectional LSTM-CNNs”. In: volume 4. MIT Press, 2016, pages 357–370 (cited on page 104).
- [121] Andrej Zukov Gregoric, Yoram Bachrach, and Sam Cope. “Named Entity Recognition With Parallel Recurrent Neural Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 69–74 (cited on page 104).
- [122] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. “A Survey on Deep Learning for Named Entity Recognition”. In: volume PP. 99. IEEE Transactions on Knowledge and Data Engineering, 2020, pages 1–1 (cited on page 104).
- [123] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4171–4186 (cited on pages 104, 122, 450, 468, 520, 524–526, 546).
- [124] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. “Improving language understanding by generative pre-training”. In: 2018 (cited on pages 104, 122, 524–526).
- [125] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. “Unsupervised Cross-lingual Representation Learning at Scale”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 8440–8451 (cited on page 104).
- [126] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2002, pages 311–318 (cited on pages 105, 112, 113).
- [127] Kenneth W Church and Eduard H Hovy. “Good applications for crummy machine translation”. In: volume 8. 4. Springer, 1993, pages 239–258 (cited on page 106).
- [128] John B. Carroll. “An experiment in evaluating the quality of translations”. In: volume 9. 3-4. Mech. Transl. Comput. Linguistics, 1966, pages 55–66 (cited on page 108).
- [129] John S White, Theresa A O’ Connell, and Francis E O’ Mara. “The ARPA MT evaluation methodologies: evolution, lessons, and future approaches”. In: Proceedings of the First Conference of the Association for Machine Translation in the Americas, 1994 (cited on pages 108, 109).

- [130] Keith J. Miller and Michelle Vanni. “Inter-rater Agreement Measures, and the Refinement of Metrics in the PLATO MT Evaluation Paradigm”. In: The tenth Machine Translation Summit, 2005, pages 125–132 (cited on page 108).
- [131] Margaret King, Andrei Popescu-Belis, and Eduard Hovy. “FEMTI: creating and using a framework for MT evaluation”. In: Proceedings of MT Summit IX, New Orleans, LA, 2003, pages 224–231 (cited on page 108).
- [132] Mark A. Przybocki, Kay Peterson, Sebastien Bronsart, and Gregory A. Sanders. “The NIST 2008 Metrics for machine translation challenge - overview, methodology, metrics, and results”. In: volume 23. 2-3. Machine Translation, 2009, pages 71–103 (cited on page 109).
- [133] Florence Reeder. “Direct application of a language learner test to MT evaluation”. In: Proceedings of AMTA, 2006 (cited on page 109).
- [134] Chris Callison-Burch, Cameron S. Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. “(Meta-) Evaluation of Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 136–158 (cited on page 109).
- [135] Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. “Findings of the 2012 Workshop on Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 10–51 (cited on page 110).
- [136] Adam Lopez. “Putting Human Assessments of Machine Translation Systems in Order”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 1–9 (cited on page 110).
- [137] Philipp Koehn. “Simulating human judgment in machine translation evaluation campaigns”. In: International Workshop on Spoken Language Translation, 2012, pages 179–184 (cited on page 110).
- [138] Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. “Findings of the 2015 Workshop on Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 1–46 (cited on page 110).
- [139] Shujian Huang and Kevin Knight. *Machine Translation: 15th China Conference, CCMT 2019, Nanchang, China, September 27–29, 2019, Revised Selected Papers*. Volume 1104. Springer Nature, 2019 (cited on page 110).

- [140] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000 (cited on pages 111, 569).
- [141] Christoph Tillmann, Stephan Vogel, Hermann Ney, Arkaitz Zubiaga, and Hassan Sawaf. “Accelerated DP based search for statistical translation”. In: European Conference on Speech Communication and Technology, 1997 (cited on page 111).
- [142] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. “A study of translation edit rate with targeted human annotation”. In: volume 200. 6. Proceedings of association for machine translation in the Americas, 2006 (cited on page 111).
- [143] Nancy Chinchor. “MUC-4 evaluation metrics”. In: Annual Meeting of the Association for Computational Linguistics, 1992, pages 22–29 (cited on page 113).
- [144] David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. “Decomposability of Translation Metrics for Improved Evaluation and Efficient Algorithms”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 610–619 (cited on page 113).
- [145] Matt Post. “A Call for Clarity in Reporting BLEU Scores”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 186–191 (cited on page 113).
- [146] Satanjeev Banerjee and Alon Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 65–72 (cited on page 114).
- [147] Michael J. Denkowski and Alon Lavie. “METEOR-NEXT and the METEOR Paraphrase Tables: Improved Evaluation Support for Five Target Languages”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 339–342 (cited on page 117).
- [148] Michael J. Denkowski and Alon Lavie. “Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 85–91 (cited on page 117).
- [149] Michael J. Denkowski and Alon Lavie. “Meteor Universal: Language Specific Translation Evaluation for Any Target Language”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 376–380 (cited on page 117).
- [150] Shiwen Yu. “Automatic evaluation of output quality for Machine Translation systems”. In: volume 8. 1-2. Mach. Transl., 1993, pages 117–126 (cited on page 117).

- [151] Ming Zhou, Bo Wang, Shujie Liu, Mu Li, Dongdong Zhang, and Tiejun Zhao. “Diagnostic Evaluation of Machine Translation Systems Using Automatically Constructed Linguistic Check-Points”. In: International Conference on Computational Linguistics, 2008, pages 1121–1128 (cited on page 118).
- [152] Joshua Albrecht and Rebecca Hwa. “A Re-examination of Machine Learning Approaches for Sentence-Level MT Evaluation”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on page 118).
- [153] Joshua Albrecht and Rebecca Hwa. “Regression for Sentence-Level MT Evaluation with Pseudo References”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on page 118).
- [154] Ding Liu and Daniel Gildea. “Source-Language Features and Maximum Correlation Training for Machine Translation Evaluation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 41–48 (cited on page 119).
- [155] Jesús Giménez and Lluís Màrquez. “Heterogeneous Automatic MT Evaluation Through Non-Parametric Metric Combinations”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 319–326 (cited on page 119).
- [156] Markus Dreyer and Daniel Marcu. “HyTER: Meaning-Equivalent Semantics for Translation Evaluation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 162–171 (cited on page 119).
- [157] Ondřej Bojar, Matouš Macháček, Aleš Tamchyna, and Daniel Zeman. “Scratching the Surface of Possible Translations”. In: volume 8082. Springer, 2013, pages 465–474 (cited on page 120).
- [158] Ying Qin and Lucia Specia. “Truly Exploring Multiple References for Machine Translation Evaluation”. In: European Association for Machine Translation, 2015 (cited on page 121).
- [159] Boxing Chen and Hongyu Guo. “Representation Based Translation Evaluation Metrics”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 150–155 (cited on page 121).
- [160] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. “Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 151–161 (cited on pages 121, 122).

- [161] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 1631–1642 (cited on page 121).
- [162] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space”. In: arXiv preprint arXiv:1301.3781, 2013 (cited on pages 122, 327).
- [163] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: International conference on machine learning, 2014, pages 1188–1196 (cited on pages 122, 523).
- [164] Ben Athiwaratkun and Andrew Gordon Wilson. “Multimodal Word Distributions”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1645–1656 (cited on page 122).
- [165] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. “Deep Contextualized Word Representations”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 2227–2237 (cited on pages 122, 327, 524, 525, 551).
- [166] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global Vectors for Word Representation”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 1532–1543 (cited on pages 122, 325, 327).
- [167] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. “Skip-thought vectors”. In: Advances in neural information processing systems, 2015, pages 3294–3302 (cited on page 122).
- [168] Junki Matsuo, Mamoru Komachi, and Katsuhito Sudoh. “Word-Alignment-Based Segment-Level Machine Translation Evaluation using Word Embeddings”. In: volume abs/1704.00380. CoRR, 2017 (cited on page 122).
- [169] Francisco Guzmán, Shafiq Joty, Lluís Márquez, and Preslav Nakov. “Machine translation evaluation with neural networks”. In: volume 45. Computer Speech & Language, 2017, pages 180–200 (cited on page 123).
- [170] Karl Pearson. “Notes on the history of correlation”. In: volume 13. 1. JSTOR, 1920, pages 25–45 (cited on page 123).
- [171] Deborah Coughlin. “Correlating automated and human assessments of machine translation quality”. In: 2003 (cited on pages 123, 124).

- [172] Andrei Popescu-Belis. “An experiment in comparative evaluation: humans vs. computers”. In: Proceedings of the Ninth Machine Translation Summit. New Orleans, 2003 (cited on page 123).
- [173] Christopher Culy and Susanne Z Riehemann. “The limits of N-gram translation evaluation metrics”. In: MT Summit IX, 2003, pages 71–78 (cited on page 124).
- [174] Andrew Finch, Yasuhiro Akiba, and Eiichiro Sumita. “Using a paraphraser to improve machine translation evaluation”. In: International Joint Conference on Natural Language Processing, 2004 (cited on page 124).
- [175] Olivier Hamon and Djamel Mostefa. “The Impact of Reference Quality on Automatic MT Evaluation”. In: International conference on machine learning, 2008, pages 39–42 (cited on page 124).
- [176] George Doddington. “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics”. In: Proceedings of the second international conference on Human Language Technology Research, 2002, pages 138–145 (cited on page 124).
- [177] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. “Re-evaluation the role of bleu in machine translation research”. In: 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006 (cited on page 124).
- [178] Hirotugu Akaike. “A new look at the statistical model identification”. In: volume 19. 6. IEEE, 1974, pages 716–723 (cited on page 124).
- [179] Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Springer, 1993 (cited on page 125).
- [180] Philipp Koehn. “Statistical Significance Tests for Machine Translation Evaluation”. In: ACL, 2004, pages 388–395 (cited on page 125).
- [181] Eric W Noreen. *Computer-intensive methods for testing hypotheses*. Wiley New York, 1989 (cited on page 125).
- [182] Stefan Riezler and John T. Maxwell III. “On Some Pitfalls in Automatic Evaluation and Significance Testing for MT”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 57–64 (cited on page 125).
- [183] Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. “An Empirical Investigation of Statistical Significance in NLP”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 995–1005 (cited on pages 125, 126).
- [184] Michael Gamon, Anthony Aue, and Martine Smets. “Sentence-level MT evaluation without reference translations: Beyond language modeling”. In: Proceedings of EAMT, 2005, pages 103–111 (cited on page 129).

- [185] Christopher Quirk. “Training a Sentence-Level Machine Translation Confidence Measure”. In: European Language Resources Association, 2004 (cited on page 129).
- [186] Douglas A. Jones, Edward Gibson, Wade Shen, Neil Granoien, Martha Herzog, Douglas A. Reynolds, and Clifford J. Weinstein. “Measuring human readability of machine generated text: three case studies in speech recognition and machine translation”. In: IEEE, 2005, pages 1009–1012 (cited on page 130).
- [187] Carolina Scarton, Marcos Zampieri, Mihaela Vela, Josef van Genabith, and Lucia Specia. “Searching for Context: a Study on Document-Level Labels for Translation Quality Estimation”. In: European Association for Machine Translation, 2015 (cited on page 130).
- [188] Pablo Fetter, Frédéric Dandurand, and Peter Regel-Brietzmann. “Word graph rescoring using confidence measures”. In: volume 1. Proceeding of Fourth International Conference on Spoken Language Processing, 1996, pages 10–13 (cited on page 131).
- [189] Ergun Biçici. “Referential Translation Machines for Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 343–351 (cited on pages 131, 132).
- [190] José Guilherme Camargo de Souza, Christian Buck, Marco Turchi, and Matteo Negri. “FBK-UEdin Participation to the WMT13 Quality Estimation Shared Task”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 352–358 (cited on page 131).
- [191] Ergun Biçici and Andy Way. “Referential Translation Machines for Predicting Translation Quality”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 313–321 (cited on pages 131, 135).
- [192] José Guilherme Camargo de Souza, Jesús González-Rubio, Christian Buck, Marco Turchi, and Matteo Negri. “FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 322–328 (cited on pages 131, 132).
- [193] Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel L. Forcada. “UAlacant word-level machine translation quality estimation system at WMT 2015”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 309–315 (cited on page 131).
- [194] Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. “QUality Estimation from ScraTCH (QUETCH): Deep Learning for Word-level Translation Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 316–322 (cited on page 132).

- [195] André F. T. Martins, Ramón Fernández Astudillo, Chris Hokamp, and Fabio Kepler. “Unbabel’s Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 806–811 (cited on page 132).
- [196] Zhiming Chen, Yiming Tan, Chenlin Zhang, Qingyu Xiang, Lilin Zhang, Maoxi Li, and Mingwen Wang. “Improving Machine Translation Quality Estimation with Neural Network Features”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 551–555 (cited on page 132).
- [197] Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. “Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation”. In: Proceedings of the Tenth Workshop on Statistical Machine Translation, 2015, pages 316–322 (cited on page 132).
- [198] Kashif Shah, Varvara Logacheva, Gustavo Paetzold, Frédéric Blain, Daniel Beck, Fethi Bougares, and Lucia Specia. “SHEF-NN: Translation Quality Estimation with Neural Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 342–347 (cited on page 132).
- [199] Carolina Scarton, Daniel Beck, Kashif Shah, Karin Sim Smith, and Lucia Specia. “Word embeddings and discourse information for Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 831–837 (cited on page 132).
- [200] Amal Abdelsalam, Ondrej Bojar, and Samhaa El-Beltagy. “Bilingual Embeddings and Word Alignments for Translation Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 764–771 (cited on page 132).
- [201] Prasenjit Basu, Santanu Pal, and Sudip Kumar Naskar. “Keep It or Not: Word Level Quality Estimation for Post-Editing”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 759–764 (cited on page 132).
- [202] Hou Qi. “NJU Submissions for the WMT19 Quality Estimation Shared Task”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 95–100 (cited on page 132).
- [203] Junpei Zhou, Zhisong Zhang, and Zecong Hu. “SOURCE: SOURce-Conditional Elmo-style Model for Machine Translation Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 106–111 (cited on page 132).

- [204] Chris Hokamp. “Ensembling Factored Neural Machine Translation Models for Automatic Post-Editing and Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 647–654 (cited on page 132).
- [205] Ziyang Wang, Hui Liu, Hexuan Chen, Kai Feng, Zeyang Wang, Bei Li, Chen Xu, Tong Xiao, and Jingbo Zhu. “NiuTrans Submission for CCMT19 Quality Estimation Task”. In: Springer, 2019, pages 82–92 (cited on page 132).
- [206] Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, António Góis, M Amin Farajian, António V Lopes, and André FT Martins. “Unbabel’s Participation in the WMT19 Translation Quality Estimation Shared Task”. In: 2019, pages 78–84 (cited on page 132).
- [207] Elizaveta Yankovskaya, Andre Tättar, and Mark Fishel. “Quality Estimation and Translation Metrics via Pre-trained Word and Sentence Embeddings”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 101–105 (cited on page 132).
- [208] Hyun Kim, Joon-Ho Lim, Hyun-Ki Kim, and Seung-Hoon Na. “QE BERT: Bilin-gual BERT Using Multi-task Learning for Neural Quality Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 85–89 (cited on page 132).
- [209] Silja Hildebrand and Stephan Vogel. “MT Quality Estimation: The CMU System for WMT’13”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 373–379 (cited on page 132).
- [210] André FT Martins, Ramón Astudillo, Chris Hokamp, and Fabio Kepler. “Unbabel’s participation in the wmt16 word-level translation quality estimation shared task”. In: Proceedings of the First Conference on Machine Translation, 2016, pages 806–811 (cited on page 132).
- [211] Ding Liu and Daniel Gildea. “Syntactic Features for Evaluation of Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 25–32 (cited on page 134).
- [212] Jesús Giménez and Lluís Màrquez. “Linguistic Features for Automatic Evaluation of Heterogenous MT Systems”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 256–264 (cited on page 134).
- [213] Sebastian Padó, Daniel M. Cer, Michel Galley, Dan Jurafsky, and Christopher D. Manning. “Measuring machine translation quality as semantic equivalence: A metric based on entailment features”. In: volume 23. 2-3. Machine Translation, 2009, pages 181–193 (cited on page 134).

- [214] Karolina Owczarzak, Josef van Genabith, and Andy Way. “Dependency-Based Automatic Evaluation for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 80–87 (cited on page 134).
- [215] Karolina Owczarzak, Josef van Genabith, and Andy Way. “Labelled Dependencies in Machine Translation Evaluation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 104–111 (cited on page 134).
- [216] Hui Yu, Xiaofeng Wu, Jun Xie, Wenbin Jiang, Qun Liu, and Shouxun Lin. “RED: A Reference Dependency Based MT Evaluation Metric”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 2042–2051 (cited on page 134).
- [217] Rafael E. Banchs and Haizhou Li. “AM-FM: A Semantic Framework for Translation Quality Assessment”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 153–158 (cited on page 134).
- [218] Florence Reeder. “Measuring MT adequacy using latent semantic analysis”. In: Proceedings of the 7th Conference of the Association for Machine Translation of the Americas. Cambridge, Massachusetts, 2006, pages 176–184 (cited on page 134).
- [219] Chi-ku Lo, Meriem Beloucif, Markus Saers, and Dekai Wu. “XMEANT: Better semantic MT evaluation without reference translations”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 765–771 (cited on page 134).
- [220] David Vilar, Jia Xu, Luis Fernando D’Haro, and Hermann Ney. “Error Analysis of Statistical Machine Translation Output”. In: European Language Resources Association (ELRA), 2006, pages 697–702 (cited on page 134).
- [221] Maja Popovic, Aljoscha Burchardt, et al. “From human to automatic error classification for machine translation output”. In: European Association for Machine Translation, 2011 (cited on page 134).
- [222] Ângela Costa, Wang Ling, Tiago Luís, Rui Correia, and Luísa Coheur. “A linguistically motivated taxonomy for Machine Translation error analysis”. In: volume 29. 2. Machine Translation, 2015, pages 127–161 (cited on page 134).
- [223] Arle Lommel, Aljoscha Burchardt, Maja Popovic, Kim Harris, Eleftherios Avramidis, and Hans Uszkoreit. “Using a new analytic measure for the annotation and analysis of MT errors on real data”. In: European Association for Machine Translation, 2014, pages 165–172 (cited on page 134).

- [224] Maja Popovic, Adrià de Gispert, Deepa Gupta, Patrik Lambert, Hermann Ney, José B. Mariño, Marcello Federico, and Rafael E. Banchs. “Morpho-syntactic Information for Automatic Error Analysis of Statistical Machine Translation Output”. In: Annual Meeting of the Association for Computational Linguistics, 2006, pages 1–6 (cited on page 134).
- [225] Maja Popovic and Hermann Ney. “Word Error Rates: Decomposition over POS classes and Applications for Error Analysis”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 48–55 (cited on page 134).
- [226] Meritxell González, Laura Mascarell, and Lluís Màrquez. “tSEARCH: Flexible and Fast Search over Automatic Translations for Improved Quality/Error Analysis”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 181–186 (cited on page 134).
- [227] Alex Kulesza and Stuart Shieber. “A learning approach to improving sentence-level MT evaluation”. In: Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation, 2004 (cited on page 134).
- [228] Simon Corston-Oliver, Michael Gamon, and Chris Brockett. “A machine learning approach to the automatic evaluation of machine translation”. In: Annual Meeting of the Association for Computational Linguistics, 2001, pages 148–155 (cited on page 134).
- [229] Joshua S Albrecht and Rebecca Hwa. “Regression for machine translation evaluation at the sentence level”. In: volume 22. 1-2. Springer, 2008, page 1 (cited on page 134).
- [230] Kevin Duh. “Ranking vs. regression in machine translation evaluation”. In: Proceedings of the Third Workshop on Statistical Machine Translation, 2008, pages 191–194 (cited on page 134).
- [231] Boxing Chen, Hongyu Guo, and Roland Kuhn. “Multi-level evaluation for machine translation”. In: Proceedings of the Tenth Workshop on Statistical Machine Translation, 2015, pages 361–365 (cited on page 134).
- [232] Franz Josef Och. “Minimum Error Rate Training in Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 160–167 (cited on pages 134, 207, 542).

- [233] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. “Minimum Risk Training for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on pages 134, 360, 428, 431, 454, 455).
- [234] Xiaodong He and Li Deng. “Maximum expected bleu training of phrase and lexicon translation models”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 292–301 (cited on page 134).
- [235] Markus Freitag, Isaac Caswell, and Scott Roy. “APE at Scale and Its Implications on MT Evaluation Biases”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 34–44 (cited on page 135).
- [236] Ergun Biçici, Declan Groves, and Josef van Genabith. “Predicting sentence translation quality using extrinsic and language independent features”. In: volume 27. 3-4. Machine Translation, 2013, pages 171–192 (cited on page 135).
- [237] Ergun Biçici, Qun Liu, and Andy Way. “Referential Translation Machines for Predicting Translation Quality and Related Statistics”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 304–308 (cited on page 135).
- [238] Kevin Knight. “Decoding Complexity in Word-Replacement Translation Models”. In: volume 25. 4. Computational Linguistics, 1999, pages 607–615 (cited on pages 151, 170, 211).
- [239] Claude Elwood Shannon. “Communication theory of secrecy systems”. In: volume 28. 4. Bell system technical journal, 1949, pages 656–715 (cited on page 154).
- [240] Franz Josef Och and Hermann Ney. “A Systematic Comparison of Various Statistical Alignment Models”. In: volume 29. 1. Computational Linguistics, 2003, pages 19–51 (cited on pages 156, 169, 176, 184, 598).
- [241] Robert C. Moore. “Improving IBM Word Alignment Model 1”. In: Annual Meeting of the Association for Computational Linguistics, 2004, pages 518–525 (cited on page 169).
- [242] 肖桐, 李天宁, 陈如山, 朱靖波, and 王会珍. “面向统计机器翻译的重对齐方法研究”. In: volume 24. 110–116. 中文信息学报, 2010 (cited on page 169).
- [243] Hua Wu and Haifeng Wang. “Improving Statistical Word Alignment with Ensemble Methods”. In: volume 3651. International Joint Conference on Natural Language Processing, 2005, pages 462–473 (cited on page 169).
- [244] Ye-Yi Wang and Wayne Ward. “Grammar Inference and Statistical Machine Translation”. In: Carnegie Mellon University, 1999 (cited on page 169).

- 
- [245] Ido Dagan, Kenneth Ward Church, and Willian Gale. “Robust Bilingual Word Alignment for Machine Aided Translation”. In: Very Large Corpora, 1993 (cited on page 169).
  - [246] Abraham Ittycheriah and Salim Roukos. “A Maximum Entropy Word Aligner for Arabic-English Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005 (cited on page 169).
  - [247] William A. Gale and Kenneth Ward Church. “Identifying Word Correspondences in Parallel Texts”. In: Morgan Kaufmann, 1991 (cited on page 169).
  - [248] Tong Xiao and Jingbo Zhu. “Unsupervised sub-tree alignment for tree-to-tree translation”. In: volume 48. Journal of Artificial Intelligence Research, 2013, pages 733–782 (cited on pages 169, 255, 256).
  - [249] Percy Liang, Benjamin Taskar, and Dan Klein. “Alignment by Agreement”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 169).
  - [250] Chris Dyer, Victor Chahuneau, and Noah A. Smith. “A Simple, Fast, and Effective Reparameterization of IBM Model 2”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 644–648 (cited on pages 169, 201, 598).
  - [251] Benjamin Taskar, Simon Lacoste-Julien, and Dan Klein. “A Discriminative Matching Approach to Word Alignment”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 73–80 (cited on pages 169, 201).
  - [252] Alexander Fraser and Daniel Marcu. “Measuring Word Alignment Quality for Statistical Machine Translation”. In: volume 33. 3. Computational Linguistics, 2007, pages 293–303 (cited on page 169).
  - [253] John DeNero and Dan Klein. “Tailoring Word Alignments to Syntactic Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on page 169).
  - [254] Paul C Davis, Zhuli Xie and Kevin Small. “All Links are not the Same: Evaluating Word Alignments for Statistical Machine Translation”. In: Machine Translation Summit XI, 2007 (cited on page 169).
  - [255] 黄书剑, 奚宁, 赵迎功, 戴新宇, and 陈家骏. “一种错误敏感的词对齐评价方法”. In: volume 23. 88-94. 中文信息学报, 2009 (cited on page 169).
  - [256] Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. “Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model”. In: volume abs/1601.03317. CoRR, 2016 (cited on page 170).

- [257] Raghavendra Udupa, Tanveer A. Faruque, and Hemanta Kumar Maji. “An Algorithmic Framework for Solving the Decoding Problem in Statistical Machine Translation”. In: International Conference on Computational Linguistics, 2004 (cited on page 170).
- [258] Sebastian Riedel and James Clarke. “Revisiting Optimal Decoding for Machine Translation IBM Model 4”. In: Annual Meeting of the Association for Computational Linguistics, 2009 (cited on page 170).
- [259] Raghavendra Udupa and Hemanta Kumar Maji. “Computational Complexity of Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 170).
- [260] Gregor Leusch, Evgeny Matusov, and Hermann Ney. “Complexity of Finding the BLEU-optimal Hypothesis in a Confusion Network”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 839–847 (cited on page 170).
- [261] Noah Fleming, Antonina Kolokolova, and Renesa Nizamee. “Complexity of alignment and decoding problems: restrictions and approximations”. In: volume 29. 3-4. Machine Translation, 2015, pages 163–187 (cited on page 170).
- [262] Stephan Vogel, Hermann Ney, and Christoph Tillmann. “HMM-Based Word Alignment in Statistical Translation”. In: International Conference on Computational Linguistics, 1996, pages 836–841 (cited on pages 171, 175).
- [263] Brown D.C. “Decentering Distortion of Lenses”. In: volume 32. Photogrammetric Engineering, 1966, pages 444–462 (cited on page 186).
- [264] David Claus and Andrew W. Fitzgibbon. “A Rational Function Lens Distortion Model for General Cameras”. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pages 213–219 (cited on page 186).
- [265] Jerneja Žganec Gros. “MSD Recombination Method in Statistical Machine Translation”. In: volume 1060. American Institute of Physics, 2008, pages 186–189 (cited on page 187).
- [266] Deyi Xiong, Qun Liu, and Shouxun Lin. “Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on pages 187, 206, 216).
- [267] Franz Josef Och and Hermann Ney. “The Alignment Template Approach to Statistical Machine Translation”. In: volume 30. 4. Computational Linguistics, 2004, pages 417–449 (cited on pages 187, 206, 216).

- [268] Shankar Kumar and William J. Byrne. “Local Phrase Reordering Models for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 161–168 (cited on pages 187, 206, 216).
- [269] Peng Li, Yang Liu, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. “A Neural Reordering Model for Phrase-based Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 1897–1907 (cited on pages 187, 206).
- [270] David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. “The Hiero Machine Translation System: Extensions, Evaluation, and Analysis”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 779–786 (cited on page 187).
- [271] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. “Non-Autoregressive Neural Machine Translation”. In: International Conference on Learning Representations, 2018 (cited on pages 187, 365, 452, 461, 463, 465).
- [272] Andrew J. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: volume 13. 2. IEEE Transactions on Information Theory, 1967, pages 260–269 (cited on page 197).
- [273] Philipp Koehn and Kevin Knight. “Estimating Word Translation Probabilities from Unrelated Monolingual Corpora Using the EM Algorithm”. In: AAAI Press, 2000, pages 711–715 (cited on page 201).
- [274] Franz Josef Och and Hermann Ney. “A Comparison of Alignment Models for Statistical Machine Translation”. In: Morgan Kaufmann, 2000, pages 1086–1090 (cited on page 201).
- [275] Kevin Knight. “Learning a translation lexicon from monolingual corpora”. In: Annual Meeting of the Association for Computational Linguistics, 2002, pages 9–16 (cited on page 203).
- [276] M. J. D. Powell. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In: volume 7. 2. The Computer Journal, 1964, pages 155–162 (cited on page 208).
- [277] David Chiang, Yuval Marton, and Philip Resnik. “Online Large-Margin Training of Syntactic and Structural Translation Features”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 224–233 (cited on pages 211, 217).

- [278] Mark Hopkins and Jonathan May. “Tuning as Ranking”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 1352–1362 (cited on pages 211, 217).
- [279] Franz Josef Och and Hans Weber. “Improving Statistical Natural Language Translation with Categories and Rules”. In: Annual Meeting of the Association for Computational Linguistics, 1998, pages 985–989 (cited on page 216).
- [280] Franz Josef Och. “Statistical machine translation: from single word models to alignment templates”. PhD thesis. 2002 (cited on page 216).
- [281] Ye-Yi Wang and Alex Waibel. “Modeling with Structures in Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 1998, pages 1357–1363 (cited on pages 216, 264).
- [282] Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. “Chunk-Based Statistical Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 303–310 (cited on page 216).
- [283] Daniel Marcu. “Towards a Unified Approach to Memory- and Statistical-Based Machine Translation”. In: Morgan Kaufmann Publishers, 2001, pages 378–385 (cited on page 216).
- [284] Philipp Koehn, Franz Josef Och, and Daniel Marcu. “Statistical Phrase-Based Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2003 (cited on pages 216, 217).
- [285] Richard Zens, Franz Josef Och, and Hermann Ney. “Phrase-Based Statistical Machine Translation”. In: Annual Conference on Artificial Intelligence, 2002, pages 18–32 (cited on pages 216, 541).
- [286] Richard Zens and Hermann Ney. “Improvements in Phrase-Based Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2004, pages 257–264 (cited on page 216).
- [287] Daniel Marcu and Daniel Wong. “A Phrase-Based, Joint Probability Model for Statistical Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2002, pages 133–139 (cited on page 216).
- [288] John DeNero, Dan Gillick, James Zhang, and Dan Klein. “Why Generative Phrase Models Underperform Surface Heuristics”. In: Annual Meeting of the Association for Computational Linguistics, 2006, pages 31–38 (cited on page 216).

- [289] German Sanchis-Trilles, Daniel Ortiz-Martinez, Jesus Gonzalez-Rubio, Jorge Gonzalez, and Francisco Casacuberta. “Bilingual segmentation for phrasetable pruning in Statistical Machine Translation”. In: Conference of the European Association for Machine Translation, 2011, pages 257–264 (cited on page 216).
- [290] Graeme W. Blackwood, Adrià de Gispert, and William Byrne. “Phrasal Segmentation Models for Statistical Machine Translation”. In: International Conference on Computational Linguistics, 2008, pages 19–22 (cited on page 216).
- [291] Deyi Xiong, Min Zhang, and Haizhou Li. “Learning Translation Boundaries for Phrase-Based Decoding”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 136–144 (cited on page 216).
- [292] Christoph Tillman. “A Unigram Orientation Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2004 (cited on page 216).
- [293] Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. “A Clustered Global Phrase Reordering Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 216).
- [294] Richard Zens and Hermann Ney. “Discriminative Reordering Models for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006, pages 55–63 (cited on page 216).
- [295] Spence Green, Michel Galley, and Christopher D. Manning. “Improved Models of Distortion Cost for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 867–875 (cited on page 216).
- [296] Colin Cherry. “Improved Reordering for Phrase-Based Translation using Sparse Features”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 22–31 (cited on page 216).
- [297] Matthias Huck, Joern Wuebker, Felix Rietig, and Hermann Ney. “A Phrase Orientation Model for Hierarchical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 452–463 (cited on page 216).
- [298] Matthias Huck, Stephan Peitz, Markus Freitag, and Hermann Ney. “Discriminative Reordering Extensions for Hierarchical Phrase-Based Machine Translation”. In: International Conference on Material Engineering and Advanced Manufacturing Technology, 2012 (cited on page 216).

- [299] Vinh Van Nguyen, Akira Shimazu, Minh Le Nguyen, and Thai Phuong Nguyen. “Improving a Lexicalized Hierarchical Reordering Model Using Maximum Entropy”. In: Machine Translation Summit XII, 2009 (cited on page 216).
- [300] Arianna Bisazza and Marcello Federico. “A Survey of Word Reordering in Statistical Machine Translation: Computational Models and Language Phenomena”. In: volume 42. 2. Computational Linguistics, 2016, pages 163–205 (cited on page 216).
- [301] Fei Xia and Michael C. McCord. “Improving a Statistical MT System with Automatically Learned Rewrite Patterns”. In: International Conference on Computational Linguistics, 2004 (cited on page 216).
- [302] Michael Collins, Philipp Koehn, and Ivona Kucerova. “Clause Restructuring for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 531–540 (cited on page 216).
- [303] Chao Wang, Michael Collins, and Philipp Koehn. “Chinese Syntactic Reordering for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 737–745 (cited on pages 216, 528).
- [304] Xianchao Wu, Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. “Extracting Pre-ordering Rules from Predicate-Argument Structures”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 29–37 (cited on page 216).
- [305] Christoph Tillmann and Hermann Ney. “Word Re-ordering and DP-based Search in Statistical Machine Translation”. In: Morgan Kaufmann, 2000, pages 850–856 (cited on page 217).
- [306] Wade Shen, Brian Delaney, and Timothy R. Anderson. “An efficient graph search decoder for phrase-based statistical machine translation”. In: International Symposium on Computer Architecture, 2006, pages 197–204 (cited on page 217).
- [307] Robert C. Moore and Chris Quirk. “Faster Beam-Search Decoding for Phrasal Statistical Machine Translation”. In: Machine Translation Summit XI, 2007 (cited on page 217).
- [308] Kenneth Heafield, Michael Kayser, and Christopher D. Manning. “Faster Phrase-Based Decoding by Refining Feature State”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 130–135 (cited on page 217).
- [309] Joern Wuebker, Hermann Ney, and Richard Zens. “Fast and Scalable Decoding with Language Model Look-Ahead for Phrase-based Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 28–32 (cited on page 217).

- [310] Richard Zens and Hermann Ney. “Improvements in dynamic programming beam search for phrase-based statistical machine translation”. In: International Symposium on Computer Architecture, 2008, pages 198–205 (cited on page 217).
- [311] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander M. Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir R. Radev. “A Smorgasbord of Features for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2004, pages 161–168 (cited on pages 217, 265).
- [312] David Chiang, Kevin Knight, and Wei Wang. “11,001 New Features for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 218–226 (cited on page 217).
- [313] Daniel Gildea. “Loosely Tree-Based Alignment for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 80–87 (cited on page 217).
- [314] Phil Blunsom, Trevor Cohn, and Miles Osborne. “A Discriminative Latent Variable Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 200–208 (cited on page 217).
- [315] Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. “A Gibbs Sampler for Phrasal Synchronous Grammar Induction”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 782–790 (cited on page 217).
- [316] Trevor Cohn and Phil Blunsom. “A Bayesian Model of Syntax-Directed Tree to String Grammar Induction”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 352–361 (cited on page 217).
- [317] David A. Smith and Jason Eisner. “Minimum Risk Annealing for Training Log-Linear Models”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 217).
- [318] Zhifei Li and Jason Eisner. “First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 40–51 (cited on page 217).
- [319] Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. “Online Large-Margin Training for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 764–773 (cited on page 217).

- [320] Markus Dreyer and Yuanzhe Dong. “APRO: All-Pairs Ranking Optimization for MT Tuning”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 1018–1023 (cited on page 217).
- [321] Tong Xiao, Derek F. Wong, and Jingbo Zhu. “A Loss-Augmented Approach to Training Syntactic Machine Translation Systems”. In: volume 24. 11. IEEE Transactions on Audio, Speech, and Language Processing, 2016, pages 2069–2083 (cited on page 217).
- [322] Harold Charles Daume III. *Practical structured learning techniques for natural language processing*. University of Southern California, 2006 (cited on page 217).
- [323] Holger Schwenk, Marta R. Costa-jussà, and José A. R. Fonollosa. “Smooth Bilin-gual N-Gram Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 430–438 (cited on page 217).
- [324] Boxing Chen, Roland Kuhn, George Foster, and Howard Johnson. “Unpacking and Transforming Feature Functions: New Ways to Smooth Phrase Tables”. In: Machine Translation Summit, 2011 (cited on page 217).
- [325] Nan Duan, Hong Sun, and Ming Zhou. “Translation Model Generalization using Probability Averaging for Machine Translation”. In: International Conference on Computational Linguistics, 2010 (cited on page 217).
- [326] Christopher Quirk and Arul Menezes. “Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 217).
- [327] José B. Mariño, Rafael E. Banchs, Josep Maria Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. “ $N$ -gram-based Machine Translation”. In: volume 32. 4. Computational Linguistics, 2006, pages 527–549 (cited on page 217).
- [328] Richard Zens, Daisy Stanton, and Peng Xu. “A Systematic Comparison of Phrase Table Pruning Techniques”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 972–983 (cited on pages 217, 456).
- [329] Howard Johnson, Joel D. Martin, George F. Foster, and Roland Kuhn. “Improving Translation Quality by Discarding Most of the Phrasetable”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 967–975 (cited on pages 217, 456).
- [330] Wang Ling, João Graça, Isabel Trancoso, and Alan W. Black. “Entropy-based Pruning for Phrase-based Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 962–971 (cited on pages 217, 456).

- [331] Luke S. Zettlemoyer and Robert C. Moore. “Selective Phrase Pair Extraction for Improved Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 209–212 (cited on page 217).
- [332] Matthias Eck, Stephan Vogel, and Alex Waibel. “Translation Model Pruning via Usage Statistics for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 21–24 (cited on page 217).
- [333] Chris Callison-Burch, Colin J. Bannard, and Josh Schroeder. “Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 255–262 (cited on page 217).
- [334] Richard Zens and Hermann Ney. “Efficient Phrase-Table Representation for Machine Translation with Applications to Online MT and Speech Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 492–499 (cited on page 217).
- [335] Ulrich Germann. “Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario”. In: Association for Machine Translation in the Americas, 2014 (cited on page 217).
- [336] David Chiang. “Hierarchical Phrase-Based Translation”. In: volume 33. 2. Computational Linguistics, 2007, pages 201–228 (cited on pages 224, 229, 239).
- [337] John Cocke and J.T. Schwartz. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, 1970 (cited on page 230).
- [338] Daniel H. Younger. “Recognition and Parsing of Context-Free Languages in Time n<sup>3</sup>”. In: volume 10. 2. Information and Control, 1967, pages 189–208 (cited on page 230).
- [339] Tadao Kasami. “An efficient recognition and syntax-analysis algorithm for context-free languages”. In: Coordinated Science Laboratory Report no. R-257, 1966 (cited on page 230).
- [340] Liang Huang and David Chiang. “Better k-best Parsing”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 53–64 (cited on page 233).
- [341] Dekai Wu. “Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora”. In: volume 23. 3. Computational Linguistics, 1997, pages 377–403 (cited on pages 239, 264).

- [342] Liang Huang, Kevin Knight, and Aravind Joshi. “Statistical syntax-directed translation with extended domain of locality”. In: *Computationally Hard Problems & Joint Inference in Speech & Language Processing*, 2006, pages 66–73 (cited on page 239).
- [343] Michel Galley and Mark Hopkins, Kevin Knight, and Daniel Marcu. “What’s in a translation rule?” In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2004, pages 273–280 (cited on pages 239, 245).
- [344] Jason Eisner. “Learning Non-Isomorphic Tree Mappings for Machine Translation”. In: *Annual Meeting of the Association for Computational Linguistics*, 2003, pages 205–208 (cited on page 239).
- [345] Min Zhang, Hongfei Jiang, AiTi Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. “A Tree Sequence Alignment-based Tree-to-Tree Translation Model”. In: *Annual Meeting of the Association for Computational Linguistics*, 2008, pages 559–567 (cited on page 239).
- [346] Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. “SPMT: Statistical Machine Translation with Syntactified Target Language Phrases”. In: *Annual Meeting of the Association for Computational Linguistics*, 2006, pages 44–52 (cited on pages 251, 264).
- [347] Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. “Building a large annotated Chinese corpus: the Penn Chinese treebank”. In: *volume 11. 2. Journal of Natural Language Engineering*, 2005, pages 207–238 (cited on page 252).
- [348] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. “Building a Large Annotated Corpus of English: The Penn Treebank”. In: *volume 19. 2. Computational Linguistics*, 1993, pages 313–330 (cited on page 252).
- [349] Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. “Synchronous Binarization for Machine Translation”. In: *Annual Meeting of the Association for Computational Linguistics*, 2006 (cited on pages 253, 264).
- [350] Tong Xiao, Mu Li, Dongdong Zhang, Jingbo Zhu, and Ming Zhou. “Better Synchronous Binarization for Machine Translation”. In: *Annual Meeting of the Association for Computational Linguistics*, 2009, pages 362–370 (cited on pages 253, 264).
- [351] Dan Klein and Christopher D. Manning. “Accurate Unlexicalized Parsing”. In: *Annual Meeting of the Association for Computational Linguistics*, 2003, pages 423–430 (cited on page 253).

- [352] Yang Liu, Yajuan Lü, and Qun Liu. “Improving Tree-to-Tree Translation with Packed Forests”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 558–566 (cited on pages 253, 264).
- [353] Declan Groves, Mary Hearne, and Andy Way. “Robust Sub-Sentential Alignment of Phrase-Structure Trees”. In: International Conference on Computational Linguistics, 2004 (cited on page 255).
- [354] Jun Sun, Min Zhang, and Chew Lim Tan. “Discriminative Induction of Sub-Tree Alignment using Limited Labeled Data”. In: International Conference on Computational Linguistics, 2010, pages 1047–1055 (cited on pages 255, 256).
- [355] Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. “Weighted Alignment Matrices for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 1017–1026 (cited on pages 255, 256).
- [356] Jun Sun, Min Zhang, and Chew Lim Tan. “Exploring Syntactic Structural Features for Sub-Tree Alignment Using Bilingual Tree Kernels”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 306–315 (cited on page 256).
- [357] Dan Klein and Christopher D. Manning. “Parsing and Hypergraphs”. In: volume 65. 3. New Developments in Parsing Technology, 2001, pages 123–134 (cited on page 257).
- [358] Joshua Goodman. “Semiring Parsing”. In: volume 25. 4. Computational Linguistics, 1999, pages 573–605 (cited on page 258).
- [359] Jason Eisner. “Parameter Estimation for Probabilistic Finite-State Transducers”. In: Annual Meeting of the Association for Computational Linguistics, 2002, pages 1–8 (cited on page 258).
- [360] Jingbo Zhu and Tong Xiao. “Improving Decoding Generalization for Tree-to-String Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 418–423 (cited on pages 262, 265).
- [361] Hiyan Alshawi, Adam L. Buchsbaum, and Fei Xia. “A Comparison of Head Transducers and Transfer for a Limited Domain Translation Application”. In: Morgan Kaufmann Publishers, 1997, pages 360–365 (cited on page 264).
- [362] Dekai Wu. “Trainable Coarse Bilingual Grammars for Parallel Text Bracketing”. In: Third Workshop on Very Large Corpor, 1995 (cited on page 264).
- [363] Dekai Wu and Hongsing Wong. “Machine Translation with a Stochastic Grammatical Channel”. In: Morgan Kaufmann Publishers, 1998, pages 1408–1415 (cited on page 264).

- [364] J.A. Sánchez and J.M. Benedí. “Obtaining Word Phrases with Stochastic Inversion Transduction Grammars for Phrase-based Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 264).
- [365] Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. “Bayesian Learning of Non-Compositional Phrases with Synchronous Parsing”. In: Annual Meeting of the Association for Computational Linguistics, 2008 (cited on page 264).
- [366] Andreas Zollmann, Ashish Venugopal, Franz Josef Och, and Jay M. Ponte. “A Systematic Comparison of Phrase-Based, Hierarchical and Syntax-Augmented Statistical MT”. In: International Conference on Computational Linguistics, 2008, pages 1145–1152 (cited on page 264).
- [367] Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. “Left-to-Right Target Generation for Hierarchical Phrase-Based Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 264).
- [368] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. “What’s in a translation rule?” In: Annual Meeting of the Association for Computational Linguistics, 2004, pages 273–280 (cited on page 264).
- [369] Bryant Huang and Kevin Knight. “Relabeling Syntax Trees to Improve Syntax-Based Machine Translation Quality”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 264).
- [370] Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. “What Can Syntax-Based MT Learn from Phrase-Based MT?” In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 755–763 (cited on pages 264, 508).
- [371] Ding Liu and Daniel Gildea. “Improved Tree-to-String Transducer for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 62–69 (cited on page 264).
- [372] Andreas Zollmann and Ashish Venugopal. “Syntax Augmented Machine Translation via Chart Parsing”. In: Annual Meeting of the Association for Computational Linguistics, 2006, pages 138–141 (cited on pages 264, 265).
- [373] Yuval Marton and Philip Resnik. “Soft Syntactic Constraints for Hierarchical Phrase-Based Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 1003–1011 (cited on pages 264, 265).
- [374] Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. “Induction of probabilistic synchronous tree-insertion grammars for machine translation”. In: Annual Meeting of the Association for Computational Linguistics, 2006 (cited on page 264).

- [375] Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li, and Chew Lim Tan. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*. 2007 (cited on page 264).
- [376] Haitao Mi, Liang Huang, and Qun Liu. “Forest-Based Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 192–199 (cited on page 264).
- [377] Haitao Mi and Liang Huang. “Forest-based Translation Rule Extraction”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 206–214 (cited on page 264).
- [378] Jiajun Zhang, Feifei Zhai, and Chengqing Zong. “Augmenting String-to-Tree Translation Models with Fuzzy Use of Source-side Syntax”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 204–215 (cited on page 265).
- [379] Martin Popel, David Marecek, Nathan Green, and Zdenek Zabokrtský. “Influence of Parser Choice on Dependency-Based MT”. In: Annual Meeting of the Association for Computational Linguistics, 2011, pages 433–439 (cited on page 265).
- [380] Tong Xiao, Jingbo Zhu, Hao Zhang, and Muhua Zhu. “An Empirical Study of Translation Rule Extraction with Multiple Parsers”. In: Chinese Information Processing Society of China, 2010, pages 1345–1353 (cited on page 265).
- [381] Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. “Unsupervised Tree Induction for Tree-based Translation”. In: volume 1. Transactions of Association for Computational Linguistic, 2013, pages 243–254 (cited on page 265).
- [382] Christopher Quirk and Arul Menezes. “Dependency treelet translation: the convergence of statistical and example-based machine-translation?” In: volume 20. 1. Machine Translation, 2006, pages 43–65 (cited on page 265).
- [383] Deyi Xiong, Qun Liu, and Shouxun Lin. “A Dependency Treelet String Correspondence Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 40–47 (cited on page 265).
- [384] Dekang Lin. “A Path-based Transfer Model for Machine Translation”. In: International Conference on Computational Linguistics, 2004 (cited on page 265).
- [385] Yuan Ding and Martha Palmer. “Machine Translation Using Probabilistic Synchronous Dependency Insertion Grammars”. In: Annual Meeting of the Association for Computational Linguistics, 2005, pages 541–548 (cited on page 265).

- [386] Hongshen Chen, Jun Xie, Fandong Meng, Wenbin Jiang, and Qun Liu. “A Dependency Edge-based Transfer Model for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 1103–1113 (cited on page 265).
- [387] Jinsong Su, Yang Liu, Haitao Mi, Hongmei Zhao, Yajuan Lv, and Qun Liu. “Dependency-Based Bracketing Transduction Grammar for Statistical Machine Translation”. In: Chinese Information Processing Society of China, 2010, pages 1185–1193 (cited on page 265).
- [388] Jun Xie, Jinan Xu, and Qun Liu. “Augment Dependency-to-String Translation with Fixed and Floating Structures”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 2217–2226 (cited on page 265).
- [389] Liangyou Li, Andy Way, and Qun Liu. “Dependency Graph-to-String Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 33–43 (cited on page 265).
- [390] Haitao Mi and Qun Liu. “Constituency to Dependency Translation with Forests”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 1433–1442 (cited on page 265).
- [391] Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. “Dependency Forest for Statistical Machine Translation”. In: International Conference on Computational Linguistics, 2010, pages 1092–1100 (cited on page 265).
- [392] German Bordel Srinivas Bangalore and Giuseppe Riccardi. “Computing consensus translation from multiple machine translation systems”. In: IEEE Workshop on Automatic Speech Recognition and Understanding, 2001, pages 351–354 (cited on page 265).
- [393] Antti-Veikko I. Rosti, Necip Fazil Ayan, Bing Xiang, Spyridon Matsoukas, Richard M. Schwartz, and Bonnie J. Dorr. “Combining Outputs from Multiple Machine Translation Systems”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 228–235 (cited on page 265).
- [394] Tong Xiao, Jingbo Zhu, and Tongran Liu. “Bagging and boosting statistical machine translation systems”. In: volume 195. Artificial Intelligence, 2013, pages 496–527 (cited on pages 265, 453, 454, 469).
- [395] Yang Feng, Yang Liu, Haitao Mi, Qun Liu, and Yajuan Lü. “Lattice-based System Combination for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 1105–1113 (cited on page 265).

- [396] Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert C. Moore. “Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 98–107 (cited on page 265).
- [397] Chi-Ho Li, Xiaodong He, Yupeng Liu, and Ning Xi. “Incremental HMM Alignment for MT System Combination”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 949–957 (cited on page 265).
- [398] Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. “Joint Decoding with Multiple Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 576–584 (cited on page 265).
- [399] Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. “Collaborative Decoding: Partial Hypothesis Re-ranking Using Translation Consensus between Decoders”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 585–592 (cited on page 265).
- [400] Tong Xiao, Jingbo Zhu, Chunliang Zhang, and Tongran Liu. “Syntactic Skeleton-Based Translation”. In: AAAI Conference on Artificial Intelligence, 2016, pages 2856–2862 (cited on pages 265, 508).
- [401] Eugene Charniak. “Immediate-Head Parsing for Language Models”. In: Morgan Kaufmann Publishers, 2001, pages 116–123 (cited on page 265).
- [402] Libin Shen, Jinxi Xu, and Ralph M. Weischedel. “A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model”. In: Annual Meeting of the Association for Computational Linguistics, 2008, pages 577–585 (cited on page 265).
- [403] Tong Xiao, Jingbo Zhu, and Muhua Zhu. “Language Modeling for Syntax-Based Machine Translation Using Tree Substitution Grammars: A Case Study on Chinese-English Translation”. In: volume 10. 4. ACM Transactions on Asian Language Information Processing (TALIP), 2011, pages 1–29 (cited on pages 265, 473).
- [404] Peter F. Brown, Vincent J. Della Pietra, Peter V. De Souza, Jennifer C. Lai, and Robert L. Mercer. “Class-based n-gram models of natural language”. In: volume 18. 4. Computational linguistics, 1992, pages 467–479 (cited on page 274).
- [405] Tomas Mikolov and Geoffrey Zweig. “Context dependent recurrent neural network language model”. In: IEEE Spoken Language Technology Workshop, 2012, pages 234–239 (cited on pages 274, 327).

- [406] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. “Recurrent Neural Network Regularization”. In: arXiv: Neural and Evolutionary Computing, 2014 (cited on page 274).
- [407] Julian G. Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. “Recurrent Highway Networks”. In: International Conference on Machine Learning, 2016 (cited on page 274).
- [408] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. “Regularizing and optimizing LSTM language models”. In: International Conference on Learning Representations, 2017 (cited on page 274).
- [409] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. “Language models are unsupervised multitask learners”. In: volume 1. 8. OpenAI Blog, 2019, page 9 (cited on pages 274, 416).
- [410] Atilim Güneş Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. “Automatic differentiation in machine learning: a survey”. In: volume 18. 1. Journal of Machine Learning Research, 2017, pages 5595–5637 (cited on pages 302, 304).
- [411] Ning Qian. “On the momentum term in gradient descent learning algorithms”. In: volume 12. 1. Neural Networks, 1999, pages 145–151 (cited on page 305).
- [412] John C. Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: volume 12. Journal of Machine Learning Research, 2011, pages 2121–2159 (cited on pages 305, 306).
- [413] Matthew D. Zeiler. “ADADELTA:An Adaptive Learning Rate Method”. In: arXiv preprint arXiv:1212.5701, 2012 (cited on page 305).
- [414] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: volume 4. 2. COURSERA: Neural networks for machine learning, 2012, pages 26–31 (cited on pages 305, 307).
- [415] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: International Conference on Learning Representations, 2015 (cited on pages 305, 307, 360).
- [416] Timothy Dozat. “Incorporating Nesterov Momentum into Adam”. In: International Conference on Learning Representations, 2016 (cited on page 305).
- [417] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. “On the Convergence of Adam and Beyond”. In: International Conference on Learning Representations, 2018 (cited on page 305).

- [418] Tong Xiao, Jingbo Zhu, Tongran Liu, and Chunliang Zhang. “Fast Parallel Training of Neural Language Models”. In: International Joint Conference on Artificial Intelligence, 2017, pages 4193–4199 (cited on pages 309, 363).
- [419] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: volume 37. International Conference on Machine Learning, 2015, pages 448–456 (cited on page 310).
- [420] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey Hinton. “Layer Normalization”. In: volume abs/1607.06450. CoRR, 2016 (cited on pages 310, 403, 487).
- [421] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pages 770–778 (cited on pages 310, 371, 379, 403, 487).
- [422] Ngoc-quan Pham, German Kruszewski, and Gemma Boleda. “Convolutional Neural Network Language Models”. In: Conference on Empirical Methods in Natural Language Processing, 2016 (cited on page 322).
- [423] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: Conference on Neural Information Processing Systems, 2013, pages 3111–3119 (cited on pages 325, 327).
- [424] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. “Causal Interpretability for Machine Learning-Problems, Methods and Evaluation”. In: volume 22. 1. ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020, pages 18–33 (cited on page 327).
- [425] Boris Kovalerchuk, Muhammad Ahmad, and Ankur Teredesai. “Survey of explainable machine learning with visual and granular methods beyond quasi-explanations”. In: volume abs/2009.10221. ArXiv, 2020 (cited on page 327).
- [426] Finale Doshi-Velez and Been Kim. “Towards A Rigorous Science of Interpretable Machine Learning”. In: arXiv preprint arXiv:1702.08608, 2017 (cited on page 327).
- [427] Philip Arthur, Graham Neubig, and Satoshi Nakamura. “Incorporating Discrete Translation Lexicons into Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1557–1567 (cited on page 327).
- [428] Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. “Prior Knowledge Integration for Neural Machine Translation using Posterior Regularization”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1514–1523 (cited on pages 327, 369).

- [429] Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. “Syntactically Guided Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on page 327).
- [430] Anna Currey and Kenneth Heafield. “Incorporating source syntax into transformer-based neural machine translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 24–33 (cited on page 327).
- [431] Baosong Yang, Derek Wong, Tong Xiao, Lidia Chao, and Jingbo Zhu. “Towards Bidirectional Hierarchical Representations for Attention-based Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1432–1441 (cited on pages 327, 369, 503).
- [432] David Mareček and Rudolf Rosa. “Extracting syntactic trees from transformer encoder self-attentions”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 347–349 (cited on page 327).
- [433] Terra Blevins, Omer Levy, and Luke Zettlemoyer. “Deep rnns encode soft hierarchical syntax”. In: Annual Meeting of the Association for Computational Linguistics, 2018 (cited on page 327).
- [434] Youzheng Wu, Xugang Lu, Hitoshi Yamamoto, Shigeki Matsuda, Chiori Hori, and Hideki Kashioka. “Factored Language Model based on Recurrent Neural Network”. In: International Conference on Computational Linguistics, 2012 (cited on page 327).
- [435] Heike Adel, Ngoc Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. “Syntactic and Semantic Features For Code-Switching Factored Language Models”. In: volume 23. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, pages 431–440 (cited on page 327).
- [436] Tian Wang and Kyunghyun Cho. “Larger-Context Language Modelling”. In: Annual Meeting of the Association for Computational Linguistics, 2015 (cited on page 327).
- [437] Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. “A Neural Knowledge Language Model”. In: arXiv preprint arXiv:1608.00318, 2016 (cited on page 327).
- [438] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. “Character-Aware Neural Language Models”. In: AAAI Conference on Artificial Intelligence, 2016 (cited on page 327).
- [439] Kyuyeon Hwang and Wonyong Sung. “Character-level language modeling with hierarchical recurrent neural networks”. In: International Conference on Acoustics, Speech and Signal Processing, 2017, pages 5720–5724 (cited on page 327).

- [440] Yasumasa Miyamoto and Kyunghyun Cho. “Gated Word-Character Recurrent Language Model”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1992–1997 (cited on page 327).
- [441] Lyan Verwimp, Joris Pelemans, Hugo Van Hamme, and Patrick Wambacq. “Character-Word LSTM Language Models”. In: Annual Conference of the European Association for Machine Translation, 2017 (cited on page 327).
- [442] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. “Hybrid speech recognition with Deep Bidirectional LSTM”. In: IEEE Workshop on Automatic Speech Recognition and Understanding, 2013, pages 273–278 (cited on page 327).
- [443] Jetic Gu, Hassan S. Shavarani, and Anoop Sarkar. “Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 401–413 (cited on pages 327, 369).
- [444] Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. “StructVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing”. In: Annual Meeting of the Association for Computational Linguistics, 2018 (cited on page 327).
- [445] Roei Aharoni and Yoav Goldberg. “Towards String-To-Tree Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017 (cited on pages 327, 507).
- [446] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. “Graph Convolutional Encoders for Syntax-aware Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017 (cited on page 327).
- [447] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. “Text Generation from Knowledge Graphs with Graph Transformers”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019 (cited on page 327).
- [448] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. “Learned in Translation: Contextualized Word Vectors”. In: Conference on Neural Information Processing Systems, 2017, pages 6294–6305 (cited on pages 327, 525).
- [449] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. “Fast and Robust Neural Network Joint Models for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 1370–1380 (cited on pages 332, 371).

- [450] Holger Schwenk. “Continuous Space Translation Models for Phrase-Based Statistical Machine Translation”. In: International Conference on Computational Linguistics, 2012, pages 1071–1080 (cited on page 332).
- [451] Nal Kalchbrenner and Phil Blunsom. “Recurrent Continuous Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 1700–1709 (cited on pages 332, 342, 371, 378).
- [452] Sepp Hochreiter. “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”. In: volume 6. 2. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 1998, pages 107–116 (cited on page 332).
- [453] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: volume 5. 2. IEEE Transportation Neural Networks, 1994, pages 157–166 (cited on page 332).
- [454] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: volume abs/1609.08144. CoRR, 2016 (cited on pages 333, 342, 358, 378, 452).
- [455] Felix Stahlberg. “Neural Machine Translation: A Review”. In: volume 69. Journal of Artificial Intelligence Research, 2020, pages 343–418 (cited on pages 333, 387, 454).
- [456] Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. “Neural versus Phrase-Based Machine Translation Quality: a Case Study”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 257–267 (cited on pages 334, 335).
- [457] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. “Achieving Human Parity on Automatic Chinese to English News Translation”. In: volume abs/1803.05567. CoRR, 2018 (cited on pages 335, 518, 529).

- [458] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. “The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 76–86 (cited on pages 336, 458, 484).
- [459] Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. “Layer-Wise Coordination between Encoder and Decoder for Neural Machine Translation”. In: Conference on Neural Information Processing Systems, 2018 (cited on page 336).
- [460] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 464–468 (cited on pages 336, 399, 409, 471, 476, 477, 479).
- [461] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek Wong, and Lidia Chao. “Learning Deep Transformer Models for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1810–1822 (cited on pages 336, 404, 407, 409, 471, 487, 489, 490, 492, 493, 496–499).
- [462] Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. “Shallow-to-Deep Training for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2020 (cited on pages 336, 409, 476, 492, 497, 500).
- [463] Xiangpeng Wei, Heng Yu, Yue Hu, Yue Zhang, Rongxiang Weng, and Weihua Luo. “Multiscale Collaborative Deep Models for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020 (cited on pages 336, 409).
- [464] Yanyang Li, Qiang Wang, Tong Xiao, Tongran Liu, and Jingbo Zhu. “Neural Machine Translation with Joint Representation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 8285–8292 (cited on pages 339, 515, 516).
- [465] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 103–111 (cited on page 342).

- [466] Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. “On Using Very Large Target Vocabulary for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 1–10 (cited on pages 342, 414, 455).
- [467] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: volume 9. Neural Computation, Dec. 1997, pages 1735–80 (cited on page 346).
- [468] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 1724–1734 (cited on page 348).
- [469] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Barry Haddow, Alexandra Birch, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. “Nematus: a Toolkit for Neural Machine Translation”. In: Annual Conference of the European Association for Machine Translation, 2017, pages 65–68 (cited on pages 357, 599).
- [470] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: volume 9. International Conference on Artificial Intelligence and Statistics, 2010, pages 249–256 (cited on pages 360, 493, 494).
- [471] Hirotugu Akaike. “Fitting autoregressive models for prediction”. In: volume 21(1). Annals of the institute of Statistical Mathematics, 2015, pages 243–247 (cited on page 365).
- [472] Yanyang Li, Tong Xiao, Yiniao Li, Qiang Wang, Changming Xu, and Jingbo Zhu. “A Simple and Effective Approach to Coverage-Aware Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 292–297 (cited on pages 368, 449, 452, 455, 524).
- [473] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. “Modeling Coverage for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on pages 368, 452, 524).
- [474] Biao Zhang and Rico Sennrich. “A Lightweight Recurrent Network for Sequence Modeling”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1538–1548 (cited on page 369).
- [475] Tao Lei, Yu Zhang, and Yoav Artzi. “Training RNNs as Fast as CNNs”. In: volume abs/1709.02755. CoRR, 2017 (cited on page 369).

- [476] Biao Zhang, Deyi Xiong, Jinsong Su, Qian Lin, and Huiji Zhang. “Simplifying Neural Machine Translation with Addition-Subtraction Twin-Gated Recurrent Networks”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 4273–4283 (cited on page 369).
- [477] Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. “Neural Machine Translation Advised by Statistical Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2017, pages 3330–3336 (cited on page 369).
- [478] Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. “Improved Neural Machine Translation with SMT Features”. In: AAAI Conference on Artificial Intelligence, 2016, pages 151–157 (cited on page 369).
- [479] Xintong Li, Guanlin Li, Lema Liu, Max Meng, and Shuming Shi. “On the Word Alignment from Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1293–1303 (cited on page 369).
- [480] Yau-Shian Wang, Hung-yi Lee, and Yun-Nung Chen. “Tree Transformer: Integrating Tree Structures into Self-Attention”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1061–1070 (cited on page 369).
- [481] Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. “A Tree-based Decoder for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 4772–4777 (cited on pages 369, 508).
- [482] Jiajun Zhang and Chengqing Zong. “Bridging Neural Machine Translation and Bilingual Dictionaries”. In: volume abs/1610.07272. CoRR, 2016 (cited on page 369).
- [483] Xiangyu Duan, Baijun Ji, Hao Jia, Min Tan, Min Zhang, Boxing Chen, Weihua Luo, and Yue Zhang. “Bilingual Dictionary Based Neural Machine Translation without Using Parallel Sentences”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 1570–1579 (cited on page 369).
- [484] Qian Cao and Deyi Xiong. “Encoding Gated Translation Memory into Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 3042–3047 (cited on page 369).
- [485] Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. “Supervised Attentions for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 2283–2288 (cited on page 369).
- [486] Lema Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. “Neural Machine Translation with Supervised Attention”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 3093–3102 (cited on page 369).

- [487] Lesly Miculicich Werlen, Dhananjay Ram, Nikolaos Pappas, and James Henderson. “Document-Level Neural Machine Translation with Hierarchical Attention Networks”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2947–2954 (cited on pages 369, 571–573).
- [488] Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. “Context-Aware Neural Machine Translation Learns Anaphora Resolution”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 1264–1274 (cited on pages 369, 569, 571, 572).
- [489] Bei Li, Hui Liu, Ziyang Wang, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and Changliang Li. “Does Multi-Encoder Help? A Case Study on Context-Aware Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3512–3518 (cited on pages 369, 427, 571, 572, 576).
- [490] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. “Phoneme recognition using time-delay neural networks”. In: volume 37. International Conference on Acoustics, Speech and Signal Processing, 1989, pages 328–339 (cited on page 371).
- [491] Yann Lecun, Bernhard Boser, John Denker, Don Henderson, Richard E. Howard, Wayne E. Hubbard, and Larry Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: volume 1. Neural Computation, 1989, pages 541–551 (cited on page 371).
- [492] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: volume 86. 11. Proceedings of the IEEE, 1998, pages 2278–2324 (cited on pages 371, 389).
- [493] Yu Zhang, William Chan, and Navdeep Jaitly. “Very deep convolutional networks for end-to-end speech recognition”. In: International Conference on Acoustics, Speech and Signal Processing, 2017, pages 4845–4849 (cited on page 371).
- [494] Li Deng, Ossama Abdel-Hamid, and Dong Yu. “A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion”. In: International Conference on Acoustics, Speech and Signal Processing, 2013, pages 6669–6673 (cited on page 371).
- [495] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. “A Convolutional Neural Network for Modelling Sentences”. In: Annual Meeting of the Association for Computational Linguistics, 2014, pages 655–665 (cited on pages 371, 376, 386, 390).

- 
- [496] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: Conference on Empirical Methods in Natural Language Processing, 2014, pages 1746–1751 (cited on pages 371, 376, 377, 386, 389, 390).
  - [497] Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. “Dependency-based Convolutional Neural Networks for Sentence Embedding”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 174–179 (cited on page 371).
  - [498] Cícero Nogueira dos Santos and Maira Gatti. “Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts”. In: International Conference on Computational Linguistics, 2014, pages 69–78 (cited on pages 371, 376).
  - [499] Mingxuan Wang, Zhengdong Lu, Hang Li, Wenbin Jiang, and Qun Liu. “genCNN: A Convolutional Architecture for Word Sequence Prediction”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 1567–1576 (cited on page 371).
  - [500] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. “Language Modeling with Gated Convolutional Networks”. In: volume 70. International Conference on Machine Learning, 2017, pages 933–941 (cited on pages 371, 379, 380).
  - [501] Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. “A Convolutional Encoder Model for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 123–135 (cited on pages 371, 378).
  - [502] Lukasz Kaiser, Aidan N. Gomez, and François Chollet. “Depthwise Separable Convolutions for Neural Machine Translation”. In: International Conference on Learning Representations, 2018 (cited on pages 371, 378, 385, 386).
  - [503] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: volume 9905. European Conference on Computer Vision, 2016, pages 21–37 (cited on page 372).
  - [504] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: volume 39. 6. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, pages 1137–1149 (cited on pages 372, 567).

- [505] Rie Johnson and Tong Zhang. “Effective Use of Word Order for Text Categorization with Convolutional Neural Networks”. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2015, pages 103–112 (cited on pages 376, 386).
- [506] Thien Huu Nguyen and Ralph Grishman. “Relation Extraction: Perspective from Convolutional Neural Networks”. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2015, pages 39–48 (cited on page 376).
- [507] Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. “Pay Less Attention with Lightweight and Dynamic Convolutions”. In: International Conference on Learning Representations, 2019 (cited on pages 378, 386, 388, 389, 409, 458, 482, 484).
- [508] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. “End-To-End Memory Networks”. In: Conference on Neural Information Processing Systems, 2015, pages 2440–2448 (cited on page 379).
- [509] Md. Amirul Islam, Sen Jia, and Neil Bruce. “How much Position Information Do Convolutional Neural Networks Encode?” In: International Conference on Learning Representations, 2020 (cited on page 380).
- [510] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey Hinton. “On the importance of initialization and momentum in deep learning”. In: International Conference on Machine Learning, 2013, pages 1139–1147 (cited on page 384).
- [511] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. “Advances in optimizing recurrent networks”. In: International Conference on Acoustics, Speech and Signal Processing, 2013, pages 8624–8628 (cited on page 385).
- [512] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: volume 15. Journal of Machine Learning Research, 2014, pages 1929–1958 (cited on pages 385, 407, 424).
- [513] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 1800–1807 (cited on pages 385, 512).
- [514] Andrew Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: CoRR, 2017 (cited on page 385).

- [515] Rie Johnson and Tong Zhang. “Deep Pyramid Convolutional Neural Networks for Text Categorization”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 562–570 (cited on page 386).
- [516] Laurent Sifre and Stéphane Mallat. “Rigid-motion scattering for image classification”. In: Citeseer, 2014 (cited on page 386).
- [517] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pages 1701–1708 (cited on page 389).
- [518] Yu-hsin Chen, Ignacio Lopez-Moreno, Tara Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada. “Locally-connected and convolutional neural networks for small footprint speaker recognition”. In: Conference of the International Speech Communication Association, 2015, pages 1136–1140 (cited on page 389).
- [519] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. “Dynamic Convolution: Attention Over Convolution Kernels”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2020, pages 11027–11036 (cited on page 389).
- [520] Peng Zhou, Suncong Zheng, Jiaming Xu, Zhenyu Qi, Hongyun Bao, and Bo Xu. “Joint Extraction of Multiple Relations and Entities by Using a Hybrid Neural Network”. In: volume 10565. Springer, 2017, pages 135–146 (cited on pages 389, 390).
- [521] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. “Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 167–176 (cited on pages 389, 390).
- [522] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. “Relation Classification via Convolutional Deep Neural Network”. In: International Conference on Computational Linguistics, 2014, pages 2335–2344 (cited on page 389).
- [523] Thien Huu Nguyen and Ralph Grishman. “Event Detection and Domain Adaptation with Convolutional Neural Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 365–371 (cited on page 390).
- [524] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. “Recurrent Convolutional Neural Networks for Text Classification”. In: AAAI Conference on Artificial Intelligence, 2015, pages 2267–2273 (cited on page 390).

- [525] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. “Molding CNNs for text: non-linear, non-consecutive convolutions”. In: Conference on Empirical Methods in Natural Language Processing, 2015, pages 1565–1575 (cited on page 390).
- [526] Emma Strubell, Patrick Verga, David Belanger, and Andrew Mccallum. “Fast and Accurate Entity Recognition with Iterated Dilated Convolutions”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 2670–2680 (cited on page 390).
- [527] Xuezhe Ma and Eduard H. Hovy. “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on page 390).
- [528] Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. “Leveraging Linguistic Structures for Named Entity Recognition with Bidirectional Recursive Neural Networks”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 2664–2669 (cited on page 390).
- [529] Changhan Wang, Kyunghyun Cho, and Douwe Kiela. “Code-Switched Named Entity Recognition with Embedding Attention”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 154–158 (cited on page 390).
- [530] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. “A Structured Self-Attentive Sentence Embedding”. In: International Conference on Learning Representations, 2017 (cited on pages 392, 491).
- [531] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, and Alexander Ku. “Image Transformer”. In: volume abs/1802.05751. CoRR, 2018 (cited on page 394).
- [532] Linhao Dong, Shuang Xu, and Bo Xu. “Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition”. In: International Conference on Acoustics, Speech and Signal Processing, 2018, pages 5884–5888 (cited on page 394).
- [533] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. “Conformer: Convolution-augmented Transformer for Speech Recognition”. In: International Speech Communication Association, 2020, pages 5036–5040 (cited on pages 394, 482).

- [534] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. “Rethinking the Inception Architecture for Computer Vision”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pages 2818–2826 (cited on pages 407, 421).
- [535] Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. “Tensor2Tensor for Neural Machine Translation”. In: Association for Machine Translation in the Americas, 2018, pages 193–199 (cited on pages 408, 409, 451, 488, 599).
- [536] Matthieu Courbariaux and Yoshua Bengio. “BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1”. In: volume abs/1602.02830. CoRR, 2016 (cited on page 408).
- [537] Ye Lin, Yanyang Li, Tengbo Liu, Tong Xiao, Tongran Liu, and Jingbo Zhu. “Towards Fully 8-bit Integer Inference for the Transformer Model”. In: International Joint Conference on Artificial Intelligence, 2020, pages 3759–3765 (cited on pages 408, 409).
- [538] Tong Xiao, Yiniao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. “Sharing Attention Weights for Fast Transformer”. In: International Joint Conference on Artificial Intelligence, 2019, pages 5292–5298 (cited on pages 408, 409, 456–458).
- [539] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. “Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5797–5808 (cited on pages 408, 458, 474, 516).
- [540] Biao Zhang, Deyi Xiong, and Jinsong Su. “Accelerating Neural Transformer via an Average Attention Network”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 1789–1798 (cited on pages 409, 458).
- [541] Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. “Weight Distillation: Transferring the Knowledge in Neural Network Parameters”. In: volume abs/2009.09152. ArXiv, 2020 (cited on page 409).
- [542] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. “Lite Transformer with Long-Short Range Attention”. In: International Conference on Learning Representations, 2020 (cited on pages 409, 483).
- [543] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. “Reformer: The Efficient Transformer”. In: International Conference on Learning Representations, 2020 (cited on pages 409, 458, 486, 570).

- [544] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. “Scaling Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018 (cited on page 409).
- [545] Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. “Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model”. In: volume abs/1906.00532. CoRR, 2019 (cited on pages 409, 460, 474).
- [546] Abigail See, Minh-Thang Luong, and Christopher D. Manning. “Compression of Neural Machine Translation Models via Pruning”. In: International Conference on Computational Linguistics, 2016, pages 291–301 (cited on page 409).
- [547] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: volume abs/1503.02531. CoRR, 2015 (cited on pages 409, 435, 436, 458, 474, 533, 534).
- [548] Yoon Kim and Alexander Rush. “Sequence-Level Knowledge Distillation”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1317–1327 (cited on pages 409, 436).
- [549] Yun Chen, Yang Liu, Yong Cheng, and Victor O. K. Li. “A Teacher-Student Framework for Zero-Resource Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1925–1935 (cited on pages 409, 532, 533).
- [550] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 2978–2988 (cited on pages 409, 476, 479).
- [551] Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. “Learning to Encode Position for Transformer with Continuous Dynamical Model”. In: volume abs/2003.09229. ArXiv, 2020 (cited on pages 409, 480).
- [552] Ganesh Jawahar, Benoît Sagot, and Djamel Seddah. “What Does BERT Learn about the Structure of Language?” In: Annual Meeting of the Association for Computational Linguistics, 2019 (cited on pages 409, 482).
- [553] Baosong Yang, Zhaopeng Tu, Derek Wong, Fandong Meng, Lidia Chao, and Tong Zhang. “Modeling Localness for Self-Attention Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 4449–4458 (cited on pages 409, 480, 482).

- [554] Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. “Convolutional Self-Attention Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4040–4045 (cited on pages 409, 482).
- [555] Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yiniao Li, and Jingbo Zhu. “Multi-layer Representation Fusion for Neural Machine Translation”. In: volume abs/2002.06714. International Conference on Computational Linguistics, 2018 (cited on pages 409, 487, 490).
- [556] Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. “Training Deeper Neural Machine Translation Models with Transparent Attention”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 3028–3033 (cited on pages 409, 487, 490–492).
- [557] Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. “Exploiting Deep Representations for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 4253–4262 (cited on pages 409, 487, 490, 493, 498).
- [558] Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. “Exploiting Sentential Context for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019 (cited on pages 409, 487).
- [559] Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. “Dynamic Layer Aggregation for Neural Machine Translation with Routing-by-Agreement”. In: AAAI Conference on Artificial Intelligence, 2019, pages 86–93 (cited on pages 409, 487, 490, 493).
- [560] Mercedes Garcia-Martinez, Loïc Barrault, and Fethi Bougares. “Factored Neural Machine Translation Architectures”. In: International Workshop on Spoken Language Translation (IWSLT’16), 2016 (cited on page 414).
- [561] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. “Fully Character-Level Neural Machine Translation without Explicit Segmentation”. In: volume 5. Transactions of the Association for Computational Linguistics, 2017, pages 365–378 (cited on pages 414, 536, 551).
- [562] Minh-Thang Luong and Christopher Manning. “Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on pages 415, 599).
- [563] Philip Gage. “A new algorithm for data compression”. In: volume 12. The C Users Journal archive, 1994, pages 23–38 (cited on page 415).

- [564] Taku Kudo. “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 66–75 (cited on pages 416, 418).
- [565] Mike Schuster and Kaisuke Nakajima. “Japanese and Korean voice search”. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2012, pages 5149–5152 (cited on page 416).
- [566] Taku Kudo and John Richardson. “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 66–71 (cited on page 418).
- [567] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. “BPE-Dropout: Simple and Effective Subword Regularization”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 1882–1892 (cited on page 418).
- [568] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. “Dynamic Programming Encoding for Subword Segmentation in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3042–3051 (cited on page 418).
- [569] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: volume 521. 7553. Nature, 2015, pages 436–444 (cited on pages 421, 428).
- [570] Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: volume abs/1207.0580. CoRR, 2012 (cited on page 422).
- [571] Mathias Müller, Annette Rios, and Rico Sennrich. “Domain Robustness in Neural Machine Translation”. In: Association for Machine Translation in the Americas, 2020, pages 151–164 (cited on page 424).
- [572] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: IEEE Symposium on Security and Privacy, 2017, pages 39–57 (cited on page 424).
- [573] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pages 2574–2582 (cited on pages 424, 446).
- [574] Yong Cheng, Lu Jiang, and Wolfgang Macherey. “Robust Neural Machine Translation with Doubly Adversarial Inputs”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4324–4333 (cited on pages 424, 427).

- [575] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pages 427–436 (cited on pages 424, 446).
- [576] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. “Intriguing properties of neural networks”. In: International Conference on Learning Representations, 2014 (cited on page 424).
- [577] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: International Conference on Learning Representations, 2015 (cited on pages 424–426).
- [578] Robin Jia and Percy Liang. “Adversarial Examples for Evaluating Reading Comprehension Systems”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 2021–2031 (cited on pages 425, 446).
- [579] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. “Adversarial training for multi-context joint entity and relation extraction”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2830–2836 (cited on page 425).
- [580] Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. “Robust Multilingual Part-of-Speech Tagging via Adversarial Training”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 976–986 (cited on page 425).
- [581] Yonatan Belinkov and Yonatan Bisk. “Synthetic and Natural Noise Both Break Neural Machine Translation”. In: International Conference on Learning Representations, 2018 (cited on page 425).
- [582] Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. “On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 3103–3114 (cited on page 425).
- [583] Zhitao Gong, Wenlu Wang, B. Li, D. Song, and W. Ku. “Adversarial Texts with Gradient Methods”. In: volume abs/1801.07175. ArXiv, 2018 (cited on page 425).
- [584] Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. “Improving Robustness of Machine Translation with Synthetic Noise”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1916–1920 (cited on page 425).

- [585] Antonios Anastasopoulos, Alison Lui, Toan Nguyen, and David Chiang. “Neural Machine Translation of Text from Non-Native Speakers”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 3070–3080 (cited on page 425).
- [586] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. “Semantically Equivalent Adversarial Rules for Debugging NLP models”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 856–865 (cited on page 425).
- [587] Suranjana Samanta and Sameep Mehta. “Towards Crafting Text Adversarial Samples”. In: volume abs/1707.02812. CoRR, 2017 (cited on page 426).
- [588] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. “Deep Text Classification Can be Fooled”. In: International Joint Conference on Artificial Intelligence, 2018, pages 4208–4215 (cited on page 426).
- [589] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. “On Adversarial Examples for Character-Level Neural Machine Translation”. In: International Conference on Computational Linguistics, 2018, pages 653–663 (cited on page 426).
- [590] Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. “Soft Contextual Data Augmentation for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5539–5544 (cited on pages 426, 522, 551).
- [591] Zhengli Zhao, Dheeru Dua, and Sameer Singh. “Generating Natural Adversarial Examples”. In: International Conference on Learning Representations, 2018 (cited on page 426).
- [592] Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. “Towards Robust Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 1756–1766 (cited on pages 427, 558).
- [593] Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. “Robust Neural Machine Translation with Joint Textual and Phonetic Embedding”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3044–3049 (cited on page 427).
- [594] Stanley Chen and Ronald Rosenfeld. “A Gaussian prior for smoothing maximum entropy models”. In: CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1999 (cited on page 428).
- [595] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. “Extracting and composing robust features with denoising autoencoders”. In: International Conference on Machine Learning, 2008 (cited on pages 428, 520, 521).

- [596] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. “Neural machine translation with reconstruction”. In: volume 31. 1. AAAI Conference on Artificial Intelligence, 2017 (cited on page 428).
- [597] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. “Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks”. In: Annual Conference on Neural Information Processing Systems, 2015, pages 1171–1179 (cited on pages 428, 429, 455).
- [598] Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. “Sequence Level Training with Recurrent Neural Networks”. In: International Conference on Learning Representations, 2016 (cited on pages 428, 446, 455).
- [599] Chen Xu, Bojie Hu, Yufan Jiang, Kai Feng, Zeyang Wang, Shen Huang, Qi Ju, Tong Xiao, and Jingbo Zhu. “Dynamic Curriculum Learning for Low-Resource Neural Machine Translation”. In: International Committee on Computational Linguistics, 2020, pages 3977–3989 (cited on pages 429, 443, 444).
- [600] Lijun Wu, Yingce Xia, Fei Tian, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. “Adversarial Neural Machine Translation”. In: volume 95. Proceedings of Machine Learning Research. Asian Conference on Machine Learning, 2018, pages 534–549 (cited on page 430).
- [601] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. “An Actor-Critic Algorithm for Sequence Prediction”. In: International Conference on Learning Representations, 2017 (cited on pages 431, 433, 446).
- [602] Sham M. Kakade. “A Natural Policy Gradient”. In: Advances in Neural Information Processing Systems, 2001, pages 1531–1538 (cited on page 432).
- [603] Peter Henderson, Joshua Romoff, and Joelle Pineau. “Where Did My Optimum Go?: An Empirical Analysis of Gradient Descent Optimization in Policy Gradient Methods”. In: volume abs/1810.02525. CoRR, 2018 (cited on page 432).
- [604] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. “Understanding Back-Translation at Scale”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 489–500 (cited on pages 432, 518, 519).
- [605] Wouter Kool, Herke van Hoof, and Max Welling. “Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement”. In: volume 97. Proceedings of Machine Learning Research. International Conference on Machine Learning, 2019, pages 3499–3508 (cited on page 432).

- [606] Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cited on page 434).
- [607] David Silver, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: volume 529. 7587. Nature, 2016, pages 484–489 (cited on page 435).
- [608] Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus. “Learning Simple Algorithms from Examples”. In: volume 48. JMLR Workshop and Conference Proceedings. International Conference on Machine Learning, 2016, pages 421–429 (cited on page 435).
- [609] Andrew Ng, Daishi Harada, and Stuart Russell. “Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping”. In: International Conference on Machine Learning, 1999, pages 278–287 (cited on page 435).
- [610] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. “Train large, then compress: Rethinking model size for efficient training and inference of transformers”. In: arXiv preprint arXiv:2002.11794, 2020 (cited on page 436).
- [611] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. “Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers”. In: volume abs/2002.11794. CoRR, 2020 (cited on page 436).
- [612] Sauleh Eetemadi, William Lewis, Kristina Toutanova, and Hayder Radha. “Survey of data-selection methods in statistical machine translation”. In: volume 29. 3-4. Machine Translation, 2015, pages 189–223 (cited on page 440).
- [613] Denny Britz, Quoc Le, and Reid Pryzant. “Effective domain mixing for neural machine translation”. In: Proceedings of the Second Conference on Machine Translation, 2017, pages 118–126 (cited on pages 440, 549).
- [614] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. “Domain Adaptation via Pseudo In-Domain Data Selection”. In: Conference on Empirical Methods in Natural Language Processing, 2011, pages 355–362 (cited on pages 440, 441).

- [615] Amitai Axelrod, Philip Resnik, Xiaodong He, and Mari Ostendorf. “Data Selection With Fewer Words”. In: Conference on Empirical Methods in Natural Language Processing, 2015, pages 58–65 (cited on pages 440, 441).
- [616] Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. “Instance Weighting for Neural Machine Translation Domain Adaptation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1482–1488 (cited on pages 440, 441, 579).
- [617] Saab Mansour, Joern Wuebker, and Hermann Ney. “Combining translation and language model scoring for domain-specific data filtering”. In: International Workshop on Spoken Language Translation, 2011, pages 222–229 (cited on page 440).
- [618] Boxing Chen and Fei Huang. “Semi-supervised Convolutional Networks for Translation Adaptation with Tiny Amount of In-domain Data”. In: The SIGNLL Conference on Computational Natural Language Learning, 2016, pages 314–323 (cited on pages 440, 441).
- [619] Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. “Bilingual methods for adaptive training data selection for machine translation”. In: Association for Machine Translation in the Americas, 2016, pages 93–103 (cited on pages 440, 441).
- [620] Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. “Cost Weighting for Neural Machine Translation Domain Adaptation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 40–46 (cited on pages 440, 441).
- [621] Mirela-Stefania Duma and Wolfgang Menzel. “Automatic Threshold Detection for Data Selection in Machine Translation”. In: Proceedings of the Second Conference on Machine Translation, 2017, pages 483–488 (cited on page 440).
- [622] Ergun Biçici and Deniz Yuret. “Instance Selection for Machine Translation using Feature Decay Algorithms”. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, 2011, pages 272–283 (cited on page 440).
- [623] Alberto Poncelas, Gideon Maillette de Buy Wenniger, and Andy Way. “Feature decay algorithms for neural machine translation”. In: European Association for Machine Translation, 2018 (cited on page 440).
- [624] Xabier Soto, Dimitar Sht. Shterionov, Alberto Poncelas, and Andy Way. “Selecting Backtranslated Data from Multiple Sources for Improved Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3898–3908 (cited on page 440).

- [625] Marlies van der Wees, Arianna Bisazza, and Christof Monz. “Dynamic Data Selection for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1400–1410 (cited on pages 441, 549).
- [626] Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. “Denoising Neural Machine Translation Training with Trusted Data and Online Data Selection”. In: Proceedings of the Third Conference on Machine Translation, 2018, pages 133–143 (cited on pages 441, 442).
- [627] Rui Wang, Masao Utiyama, and Eiichiro Sumita. “Dynamic Sentence Sampling for Efficient Training of Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 298–304 (cited on page 441).
- [628] Huda Khayrallah and Philipp Koehn. “On the Impact of Various Types of Noise on Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 74–83 (cited on page 441).
- [629] Lluís Formiga and José A. R. Fonollosa. “Dealing with Input Noise in Statistical Machine Translation”. In: International Conference on Computational Linguistics, 2012, pages 319–328 (cited on page 441).
- [630] Lei Cui, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. “Bilingual Data Cleaning for SMT using Graph-based Random Walk”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 340–345 (cited on page 441).
- [631] Mohammed Mediani. “Learning from Noisy Data in Statistical Machine Translation”. PhD thesis. Karlsruhe Institute of Technology, Germany, 2017 (cited on page 441).
- [632] Spencer Rarrick, Chris Quirk, and Will Lewis. “MT detection in web-scraped parallel corpora”. In: Machine Translation, 2011, pages 422–430 (cited on page 441).
- [633] Kaveh Taghipour, Shahram Khadivi, and Jia Xu. “Parallel corpus refinement as an outlier detection algorithm”. In: Machine Translation, 2011, pages 414–421 (cited on page 441).
- [634] Hainan Xu and Philipp Koehn. “Zipporah: a Fast and Scalable Data Cleaning System for Noisy Web-Crawled Parallel Corpora”. In: *Conference on Empirical Methods in Natural Language Processing*. 2017 (cited on page 441).
- [635] Marine Carpuat, Yogarshi Vyas, and Xing Niu. “Detecting Cross-Lingual Semantic Divergence for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 69–79 (cited on page 441).

- [636] Yogarshi Vyas, Xing Niu, and Marine Carpuat. “Identifying Semantic Divergences in Parallel Text without Annotations”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 1503–1515 (cited on page 441).
- [637] Wei Wang, Isaac Caswell, and Ciprian Chelba. “Dynamically Composing Domain-Data Selection with Clean-Data Selection by ”Co-Curricular Learning” for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1282–1292 (cited on pages 442, 443).
- [638] Jingbo Zhu, Huizhen Wang, and Eduard H. Hovy. “Multi-Criteria-Based Strategy to Stop Active Learning for Data Annotation”. In: International Conference on Computational Linguistics, 2008, pages 1129–1136 (cited on page 442).
- [639] Jingbo Zhu and Matthew Ma. “Uncertainty-based active learning with instability estimation for text classification”. In: volume 8. 4. ACM Transactions on Speech and Language Processing, 2012, 5:1–5:21 (cited on page 442).
- [640] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. “Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification”. In: International Conference on Computational Linguistics, 2008, pages 1137–1144 (cited on page 442).
- [641] Ming Liu, Wray L. Buntine, and Gholamreza Haffari. “Learning to Actively Learn Neural Machine Translation”. In: The SIGNLL Conference on Computational Natural Language Learning, 2018, pages 334–344 (cited on page 442).
- [642] Yuekai Zhao, Haoran Zhang, Shuchang Zhou, and Zhihua Zhang. “Active Learning Approaches to Enhancing Neural Machine Translation: An Empirical Study”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 1796–1806 (cited on page 442).
- [643] Álvaro Peris and Francisco Casacuberta. “Active Learning for Interactive Neural Machine Translation of Data Streams”. In: The SIGNLL Conference on Computational Natural Language Learning, 2018, pages 151–160 (cited on pages 442, 453).
- [644] Marco Turchi, Matteo Negri, M. Amin Farajian, and Marcello Federico. “Continuous Learning from Human Post-Edits for Neural Machine Translation”. In: volume 108. The Prague Bulletin of Mathematical Linguistics, 2017, pages 233–244 (cited on page 442).
- [645] Álvaro Peris and Francisco Casacuberta. “Online learning for effort reduction in interactive neural machine translation”. In: volume 58. Computer Speech Language, 2019, pages 98–126 (cited on pages 442, 580).

- [646] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. “Curriculum learning”. In: volume 382. ACM International Conference Proceeding Series. International Conference on Machine Learning, pages 41–48 (cited on page 442).
- [647] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabás Póczos, and Tom M. Mitchell. “Competence-based Curriculum Learning for Neural Machine Translation”. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pages 1162–1172 (cited on pages 443, 444).
- [648] Tom Kocmi and Ondrej Bojar. “Curriculum Learning and Minibatch Bucketing in Neural Machine Translation”. In: International Conference Recent Advances in Natural Language Processing, 2017, pages 379–386 (cited on page 443).
- [649] Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. “Curriculum Learning for Domain Adaptation in Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1903–1915 (cited on pages 443, 550).
- [650] Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. “An empirical exploration of curriculum learning for neural machine translation”. In: arXiv preprint arXiv:1811.00739, 2018 (cited on pages 443, 444, 446).
- [651] Yikai Zhou, Baosong Yang, Derek Wong, Yu Wan, and Lidia S. Chao. “Uncertainty-Aware Curriculum Learning for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 6934–6944 (cited on page 444).
- [652] Zhizhong Li and Derek Hoiem. “Learning without Forgetting”. In: volume 40. 12. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, pages 2935–2947 (cited on page 445).
- [653] Amal Rannen Triki, Rahaf Aljundi, Matthew Blaschko, and Tinne Tuytelaars. “Encoder Based Lifelong Learning”. In: IEEE International Conference on Computer Vision, 2017, pages 1329–1337 (cited on page 445).
- [654] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. “iCaRL: Incremental Classifier and Representation Learning”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 5533–5542 (cited on page 445).

- [655] Francisco Castro, Manuel Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Kartik Alahari. “End-to-End Incremental Learning”. In: volume 11216. Lecture Notes in Computer Science. European Conference on Computer Vision, 2018, pages 241–257 (cited on page 445).
- [656] Andrei Rusu, Neil Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. “Progressive neural networks”. In: arXiv preprint arXiv:1606.04671, 2016 (cited on page 445).
- [657] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei Rusu, Alexander Pritzel, and Daan Wierstra. “PathNet: Evolution Channels Gradient Descent in Super Neural Networks”. In: volume abs/1701.08734. CoRR, 2017 (cited on page 445).
- [658] Paul Michel and Graham Neubig. “MTNT: A Testbed for Machine Translation of Noisy Text”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 543–553 (cited on page 446).
- [659] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. “Delving into Transferable Adversarial Examples and Black-box Attacks”. In: International Conference on Learning Representations, 2017 (cited on page 446).
- [660] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. “Adversarial Examples: Attacks and Defenses for Deep Learning”. In: volume 30. 9. IEEE Transactions on Neural Networks and Learning Systems, 2019, pages 2805–2824 (cited on page 446).
- [661] Xiaoyong Yuan, Pan He, Xiaolin Li, and Dapeng Wu. “Adaptive Adversarial Attack on Scene Text Recognition”. In: IEEE Conference on Computer Communications, 2020, pages 358–363 (cited on page 446).
- [662] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: volume 15. JMLR Proceedings. JMLR.org, 2011, pages 627–635 (cited on page 446).
- [663] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. “Improving Multi-Step Prediction of Learned Time Series Models”. In: AAAI Conference on Artificial Intelligence, 2015, pages 3024–3030 (cited on page 446).
- [664] Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. “Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback”. In: Empirical Methods in Natural Language Processing, 2017, pages 1464–1474 (cited on pages 446, 581).

- [665] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Improving Neural Machine Translation Models with Monolingual Data”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on pages 446, 518, 519).
- [666] Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. “A Study of Reinforcement Learning for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 3612–3621 (cited on page 446).
- [667] Ajay Surendranath and Dinesh Babu Jayagopi. “Curriculum Learning for Depth Estimation with Deep Convolutional Neural Networks”. In: Mediterranean Conference on Pattern Recognition and Artificial Intelligence, 2018, pages 95–100 (cited on page 446).
- [668] Haw-Shiuan Chang, Erik G. Learned-Miller, and Andrew McCallum. “Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples”. In: Annual Conference on Neural Information Processing Systems, 2017, pages 1002–1012 (cited on page 446).
- [669] Felix Stahlberg, Eva Hasler, Danielle Saunders, and Bill Byrne. “SGNMT - A Flexible NMT Decoding Platform for Quick Prototyping of New Models and Search Strategies”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 25–30 (cited on pages 448, 453).
- [670] Felix Stahlberg and Bill Byrne. “On NMT Search Errors and Model Errors: Cat Got Your Tongue?” In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 3354–3360 (cited on pages 449, 454).
- [671] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Edinburgh Neural Machine Translation Systems for WMT 16”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 371–376 (cited on pages 450, 471).
- [672] Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. “Agreement on Target-bidirectional Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2016, pages 411–416 (cited on page 450).
- [673] Bei Li, Yiniao Li, Chen Xu, Ye Lin, Jiqiang Liu, Hui Liu, Ziyang Wang, Yuhao Zhang, Nuo Xu, Zeyang Wang, Kai Feng, Hexuan Chen, Tengbo Liu, Yanyang Li, Qiang Wang, Tong Xiao, and Jingbo Zhu. “The NiuTrans Machine Translation Systems for WMT19”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 257–266 (cited on pages 450, 471).

- [674] Felix Stahlberg, Adrià de Gispert, and Bill Byrne. “The University of Cambridge’s Machine Translation Systems for WMT18”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 504–512 (cited on page 450).
- [675] Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. “Asynchronous Bidirectional Decoding for Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2018, pages 5698–5705 (cited on page 450).
- [676] Long Zhou, Jiajun Zhang, and Chengqing Zong. “Synchronous Bidirectional Neural Machine Translation”. In: volume 7. Transactions of the Association for Computational Linguistics, 2019, pages 91–105 (cited on page 450).
- [677] Aodong Li, Shiyue Zhang, Dong Wang, and Thomas Fang Zheng. “Enhanced neural machine translation by learning from draft”. In: IEEE Asia-Pacific Services Computing Conference, 2017, pages 1583–1587 (cited on page 450).
- [678] Ayah ElMaghraby and Ahmed Rafea. “Enhancing Translation from English to Arabic Using Two-Phase Decoder Translation”. In: Intelligent Systems and Applications, 2018, pages 539–549 (cited on page 450).
- [679] Xinwei Geng, Xiaocheng Feng, Bing Qin, and Ting Liu. “Adaptive Multi-pass Decoder for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 523–532 (cited on page 450).
- [680] Jason Lee, Elman Mansimov, and Kyunghyun Cho. “Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 1173–1182 (cited on pages 450, 465, 467, 468).
- [681] Jiatao Gu, Changhan Wang, and Jake Zhao. “Levenshtein Transformer”. In: Annual Conference on Neural Information Processing Systems, 2019, pages 11179–11189 (cited on page 450).
- [682] Junliang Guo, Linli Xu, and Enhong Chen. “Jointly Masked Sequence-to-Sequence Model for Non-Autoregressive Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 376–385 (cited on page 450).
- [683] Shikib Mehri and Leonid Sigal. “Middle-Out Decoding”. In: Conference on Neural Information Processing Systems, 2018, pages 5523–5534 (cited on page 450).
- [684] Felix Stahlberg, Danielle Saunders, and Bill Byrne. “An Operation Sequence Model for Explainable Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 175–186 (cited on page 450).

- [685] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. “Insertion Transformer: Flexible Sequence Generation via Insertion Operations”. In: International Conference on Machine Learning, 2019, pages 5976–5985 (cited on page 450).
- [686] Robert Östling and Jörg Tiedemann. “Neural machine translation for low-resource languages”. In: volume abs/1708.05729. CoRR, 2017 (cited on page 450).
- [687] Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. “Controlling Output Length in Neural Encoder-Decoders”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1328–1338 (cited on page 451).
- [688] Sho Takase and Naoaki Okazaki. “Positional Encoding to Control Output Sequence Length”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 3999–4004 (cited on page 451).
- [689] Kenton Murray and David Chiang. “Correcting Length Bias in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 212–223 (cited on pages 451, 454).
- [690] Pavel Sountsov and Sunita Sarawagi. “Length bias in Encoder Decoder Models and a Case for Global Conditioning”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1516–1525 (cited on pages 451, 454).
- [691] Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. “Montreal Neural Machine Translation Systems for WMT’15”. In: Conference on Empirical Methods in Natural Language Processing, 2015, pages 134–140 (cited on page 451).
- [692] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. “OpenNMT: Open-Source Toolkit for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 67–72 (cited on pages 451, 453, 488, 599).
- [693] Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q. Zhu. “Improving Attention Modeling with Implicit Distortion and Fertility for Machine Translation”. In: International Conference on Computational Linguistics, 2016, pages 3082–3092 (cited on page 452).
- [694] Jing Yang, Biao Zhang, Yue Qin, Xiangwen Zhang, Qian Lin, and Jinsong Su. “Otem&Utem: Over- and Under-Translation Evaluation Metric for NMT”. In: CCF International Conference on Natural Language Processing and Chinese Computing, 2018, pages 291–302 (cited on page 452).

- [695] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. “Coverage Embedding Models for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 955–960 (cited on page 452).
- [696] M. Kazimi and Marta R. Costa-jussà. “Coverage for Character Based Neural Machine Translation”. In: volume 59. arXiv preprint arXiv:1810.02340, 2017, pages 99–106 (cited on page 452).
- [697] Sam Wiseman and Alexander M. Rush. “Sequence-to-Sequence Learning as Beam-Search Optimization”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1296–1306 (cited on page 453).
- [698] Mingbo Ma, Renjie Zheng, and Liang Huang. “Learning to Stop in Structured Prediction for Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1884–1889 (cited on page 453).
- [699] J. Eisner and Hal Daumé. “Learning Speed-Accuracy Tradeoffs in Nondeterministic Inference Algorithms”. In: Annual Conference on Neural Information Processing Systems, 2011 (cited on page 453).
- [700] Jiarong Jiang, Adam R. Teichert, Hal Daumé, and Jason Eisner. “Learned Prioritization for Trading Off Accuracy and Speed”. In: Annual Conference on Neural Information Processing Systems, 2012, pages 1340–1348 (cited on page 453).
- [701] Renjie Zheng, Mingbo Ma, Baigong Zheng, Kaibo Liu, and Liang Huang. “Opportunistic Decoding with Timely Correction for Simultaneous Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 437–442 (cited on page 453).
- [702] Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. “STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3025–3036 (cited on page 453).
- [703] Álvaro Peris, Miguel Domingo, and F. Casacuberta. “Interactive neural machine translation”. In: volume 45. Computer Speech and Language, 2017, pages 201–220 (cited on pages 453, 581).
- [704] Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. “A Systematic Exploration of Diversity in Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2013, pages 1100–1111 (cited on page 453).

- [705] Jiwei Li and Dan Jurafsky. "Mutual Information and Diverse Decoding Improve Neural Machine Translation". In: volume abs/1601.00372. CoRR, 2016 (cited on pages 453, 454).
- [706] Nan Duan, Mu Li, Tong Xiao, and Ming Zhou. "The Feature Subspace Method for SMT System Combination". In: Conference on Empirical Methods in Natural Language Processing, 2009, pages 1096–1104 (cited on pages 453, 454, 469).
- [707] Tong Xiao, Jingbo Zhu, Muhua Zhu, and Huizhen Wang. "Boosting-Based System Combination for Machine Translation". In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 739–748 (cited on pages 453, 454, 469).
- [708] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. "A Diversity-Promoting Objective Function for Neural Conversation Models". In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2016, pages 110–119 (cited on page 454).
- [709] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. "Sequence to Sequence Mixture Model for Diverse Machine Translation". In: International Conference on Computational Linguistics, 2018, pages 583–592 (cited on page 454).
- [710] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. "Mixture Models for Diverse Machine Translation: Tricks of the Trade". In: International Conference on Machine Learning, 2019, pages 5719–5728 (cited on page 454).
- [711] Xuanfu Wu, Yang Feng, and Chenze Shao. "Generating Diverse Translation from Model Distribution with Dropout". In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 1088–1097 (cited on page 454).
- [712] Zewei Sun, Shujian Huang, Hao Ran Wei, Xin Yu Dai, and Jiajun Chen. "Generating Diverse Translation by Manipulating Multi-Head Attention". In: AAAI Conference on Artificial Intelligence, 2020, pages 8976–8983 (cited on page 454).
- [713] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. "Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models". In: volume abs/1610.02424. CoRR, 2016 (cited on page 454).
- [714] Tong Xiao, Derek F. Wong, and Jingbo Zhu. "A Loss-Augmented Approach to Training Syntactic Machine Translation Systems". In: volume 24. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2016, pages 2069–2083 (cited on page 454).

- [715] Lemao Liu and Liang Huang. “Search-Aware Tuning for Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2014, pages 1942–1952 (cited on page 454).
- [716] Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. “Max-Violation Perceptron and Forced Decoding for Scalable MT Training”. In: Conference on Empirical Methods in Natural Language Processing, 2013, pages 1112–1123 (cited on page 454).
- [717] Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. “Analyzing Neural MT Search and Model Performance”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 11–17 (cited on page 454).
- [718] Philipp Koehn and Rebecca Knowles. “Six Challenges for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 28–39 (cited on page 454).
- [719] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. “Bridging the Gap between Training and Inference for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4334–4343 (cited on page 455).
- [720] Jianhua Lin. “Divergence measures based on the Shannon entropy”. In: volume 37. 1. IEEE Transactions on Information Theory, 1991, pages 145–151 (cited on page 456).
- [721] Raj Dabre and Atsushi Fujita. “Recurrent Stacking of Layers for Compact Neural Machine Translation Models”. In: AAAI Conference on Artificial Intelligence, 2019, pages 6292–6299 (cited on page 457).
- [722] Sharan Narang, Eric Undersander, and Gregory Diamos. “Block-Sparse Recurrent Neural Networks”. In: volume abs/1711.02782. CoRR, 2017 (cited on page 458).
- [723] Trevor Gale, Erich Elsen, and Sara Hooker. “The State of Sparsity in Deep Neural Networks”. In: volume abs/1902.09574. CoRR, 2019 (cited on page 458).
- [724] Paul Michel, Omer Levy, and Graham Neubig. “Are Sixteen Heads Really Better than One?” In: Annual Conference on Neural Information Processing Systems, 2019, pages 14014–14024 (cited on pages 458, 474, 516).
- [725] Raden Mu’az Mun’im, Nakamasa Inoue, and Koichi Shinoda. “Sequence-level Knowledge Distillation for Model Compression of Attention-based Sequence-to-sequence Speech Recognition”. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2019, pages 6151–6155 (cited on pages 458, 474).

- [726] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”. In: volume abs/2006.16236. International Conference on Machine Learning, 2020 (cited on pages 458, 487).
- [727] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. “Linformer: Self-Attention with Linear Complexity”. In: volume abs/2006.04768. CoRR, 2020 (cited on pages 458, 473, 516).
- [728] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. “Fast-BERT: a Self-distilling BERT with Adaptive Inference Time”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 6035–6044 (cited on page 458).
- [729] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. “Depth-Adaptive Transformer”. In: International Conference on Learning Representations, 2020 (cited on page 458).
- [730] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pages 2704–2713 (cited on page 460).
- [731] Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. “Fully Quantized Transformer for Improved Translation”. In: volume abs/1910.10485. CoRR, 2019 (cited on page 460).
- [732] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. “Binarized Neural Networks”. In: Annual Conference on Neural Information Processing Systems, 2016, pages 4107–4115 (cited on page 461).
- [733] Chunting Zhou, Graham Neubig, and Jiatao Gu. “Understanding Knowledge Distillation in Non-autoregressive Machine Translation”. In: volume abs/1911.02727. ArXiv, 2020 (cited on page 465).
- [734] Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. “Fine-Tuning by Curriculum Learning for Non-Autoregressive Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 7839–7846 (cited on page 465).

- [735] Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. “Imitation Learning for Non-Autoregressive Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1304–1312 (cited on page 465).
- [736] Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. “Non-Autoregressive Neural Machine Translation with Enhanced Decoder Input”. In: AAAI Conference on Artificial Intelligence, 2019, pages 3723–3730 (cited on pages 465, 474).
- [737] Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. “Non-Autoregressive Machine Translation with Auxiliary Regularization”. In: AAAI Conference on Artificial Intelligence, 2019, pages 5377–5384 (cited on pages 465, 466).
- [738] Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard H. Hovy. “FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 4281–4291 (cited on pages 465, 474).
- [739] Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. “Fast Decoding in Sequence Models using Discrete Latent Variables”. In: International Conference on Machine Learning, 2018, pages 2395–2404 (cited on page 465).
- [740] Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. “Learning to Recover from Multi-Modality Errors for Non-Autoregressive Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3059–3069 (cited on page 465).
- [741] Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. “ENGINE: Energy-Based Inference Networks for Non-Autoregressive Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 2819–2826 (cited on page 465).
- [742] Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. “Latent-Variable Non-Autoregressive Neural Machine Translation with Deterministic Inference using a Delta Posterior”. In: AAAI Conference on Artificial Intelligence, 2020, pages 8846–8853 (cited on page 465).
- [743] Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. “Hint-Based Training for Non-Autoregressive Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 5707–5712 (cited on page 465).

- [744] Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. “Syntactically Supervised Transformers for Faster Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1269–1281 (cited on page 466).
- [745] Chunqi Wang, Ji Zhang, and Haiqing Chen. “Semi-Autoregressive Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 479–488 (cited on pages 466, 467).
- [746] Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. “Guiding Non-Autoregressive Neural Machine Translation Decoding with Reordering Information”. In: volume abs/1911.02215. CoRR, 2019 (cited on pages 467, 474).
- [747] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. “Mask-Predict: Parallel Decoding of Conditional Masked Language Models”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 6111–6120 (cited on pages 467, 468).
- [748] Jungo Kasai, J. Cross, Marjan Ghazvininejad, and Jiatao Gu. “Non-Autoregressive Machine Translation with Disentangled Context Transformer”. In: arXiv: Computation and Language, 2020 (cited on page 467).
- [749] Yoav Freund and Robert E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: volume 55. 1. Journal of Computer and System Sciences, 1997, pages 119–139 (cited on page 469).
- [750] Khe Chai Sim, William J. Byrne, Mark J. F. Gales, Hichem Sahbi, and Philip C. Woodland. “Consensus Network Decoding for Statistical Machine Translation System Combination”. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2007, pages 105–108 (cited on page 469).
- [751] Antti-Veikko I. Rostí, Spyridon Matsoukas, and Richard M. Schwartz. “Improved Word-Level System Combination for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on page 469).
- [752] Antti-Veikko I. Rostí, Bing Zhang, Spyros Matsoukas, and Richard M. Schwartz. “Incremental Hypothesis Alignment for Building Confusion Networks with Application to Machine Translation System Combination”. In: Proceedings of the Third Workshop on Statistical Machine Translation, 2008, pages 183–186 (cited on page 469).
- [753] Jiwei Li, Will Monroe, and Dan Jurafsky. “A Simple, Fast Diverse Decoding Algorithm for Neural Generation”. In: volume abs/1611.08562. CoRR, 2016 (cited on page 469).

- [754] Mingxuan Wang, Li Gong, Wenhuan Zhu, Jun Xie, and Chao Bian. “Tencent Neural Machine Translation Systems for WMT18”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 522–527 (cited on page 471).
- [755] Yuhao Zhang, Ziyang Wang, Runzhe Cao, Binghao Wei, Weiqiao Shan, Shuhan Zhou, Abudurexiti Reheman, Tao Zhou, Xin Zeng, Laohu Wang, Yongyu Mu, Jingnan Zhang, Xiaoqian Liu, Xuanjun Zhou, Yinqiao Li, Bei Li, Tong Xiao, and Jingbo Zhu. “The NiuTrans Machine Translation Systems for WMT20”. In: Annual Meeting of the Association for Computational Linguistics, Nov. 2020, pages 336–343 (cited on page 471).
- [756] Roy Tromble, Shankar Kumar, Franz Josef Och, and Wolfgang Macherey. “Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2008, pages 620–629 (cited on page 471).
- [757] Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. “Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2017, pages 3302–3308 (cited on page 472).
- [758] Leonhard Held and D Sabanés Bové. “Applied statistical inference”. In: volume 10. 978-3. Springer, 2014, page 16 (cited on page 473).
- [759] S. D. Silvey. “Statistical Inference”. In: Encyclopedia of Social Network Analysis and Mining, 2018 (cited on page 473).
- [760] Matthew J. Beal. “Variational algorithms for approximate Bayesian inference”. In: University College London, 2003 (cited on page 473).
- [761] Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. “Variational Decoding for Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 593–601 (cited on page 473).
- [762] Jasmijn Bastings, Wilker Aziz, Ivan Titov, and Khalil Sima’an. “Modeling Latent Sentence Structure in Neural Machine Translation”. In: volume abs/1901.06436. CoRR, 2019 (cited on page 473).
- [763] Harshil Shah and David Barber. “Generative Neural Machine Translation”. In: Annual Conference on Neural Information Processing Systems, 2018, pages 1353–1362 (cited on page 473).
- [764] Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. “Variational Recurrent Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2018, pages 5488–5495 (cited on pages 473, 516).

- [765] Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. “Variational Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 521–530 (cited on page 473).
- [766] Angela Fan, Edouard Grave, and Armand Joulin. “Reducing Transformer Depth on Demand with Structured Dropout”. In: International Conference on Learning Representations, 2020 (cited on pages 473, 474).
- [767] Qiang Wang, Tong Xiao, and Jingbo Zhu. “Training Flexible Depth Model by Multi-Task Learning for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 4307–4312 (cited on pages 473, 584).
- [768] Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. “BERT-of-Theseus: Compressing BERT by Progressive Module Replacing”. In: Conference on Empirical Methods in Natural Language Processing, 2020 (cited on page 473).
- [769] Alexei Baevski and Michael Auli. “Adaptive Input Representations for Neural Language Modeling”. In: arXiv preprint arXiv:1809.10853, 2019 (cited on page 473).
- [770] Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. “DeFINE: DEep Factorized INput Word Embeddings for Neural Sequence Modeling”. In: volume abs/1911.12385. CoRR, 2019 (cited on page 473).
- [771] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Dawei Song, and Ming Zhou. “A Tensorized Transformer for Language Modeling”. In: volume abs/1906.09777. CoRR, 2019 (cited on page 473).
- [772] Zhilin Yang, Thang Luong, Ruslan Salakhutdinov, and Quoc V. Le. “Mixtape: Breaking the Softmax Bottleneck Efficiently”. In: Conference on Neural Information Processing Systems, 2019, pages 15922–15930 (cited on page 473).
- [773] Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. “Deep Encoder, Shallow Decoder: Reevaluating the Speed-Quality Tradeoff in Machine Translation”. In: volume abs/2006.10369. CoRR, 2020 (cited on pages 473, 489).
- [774] Chi Hu, Bei Li, Yinqiao Li, Ye Lin, Yanyang Li, Chenglong Wang, Tong Xiao, and Jingbo Zhu. “The NiuTrans System for WNGT 2020 Efficiency Task”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 204–210 (cited on pages 473, 489).
- [775] Yi-Te Hsu, Sarthak Garg, Yi-Hsiu Liao, and Ilya Chatsviorkin. “Efficient Inference For Neural Machine Translation”. In: volume abs/2010.02416. CoRR, 2020 (cited on page 473).

- [776] Song Han, Jeff Pool, John Tran, and William J. Dally. “Learning both Weights and Connections for Efficient Neural Network”. In: Annual Conference on Neural Information Processing Systems, 2015, pages 1135–1143 (cited on page 474).
- [777] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. “Snip: single-Shot Network Pruning based on Connection sensitivity”. In: International Conference on Learning Representations, 2019 (cited on page 474).
- [778] Jonathan Frankle and Michael Carbin. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks”. In: International Conference on Learning Representations, 2019 (cited on page 474).
- [779] Christopher Brix, Parnia Bahar, and Hermann Ney. “Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3909–3915 (cited on page 474).
- [780] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. “Learning Efficient Convolutional Networks through Network Slimming”. In: IEEE International Conference on Computer Vision, 2017, pages 2755–2763 (cited on page 474).
- [781] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. “Rethinking the Value of Network Pruning”. In: volume abs/1810.05270. ArXiv, 2019 (cited on page 474).
- [782] Robin Cheong and Robel Daniel. “transformers.zip : Compressing Transformers with Pruning and Quantization”. In: Stanford University, 2019 (cited on page 474).
- [783] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. “Scalable Methods for 8-bit Training of Neural Networks”. In: Conference on Neural Information Processing Systems, 2018, pages 5151–5159 (cited on page 474).
- [784] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. “Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations”. In: volume 18. Journal of Machine Learning Research, 2017, 187:1–187:30 (cited on page 474).
- [785] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. “Distilling Task-Specific Knowledge from BERT into Simple Neural Networks”. In: volume abs/1903.12136. CoRR, 2019 (cited on page 474).
- [786] Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. “Aligned Cross Entropy for Non-Autoregressive Machine Translation”. In: volume abs/2004.01655. CoRR, 2020 (cited on page 474).

- [787] Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. “Minimizing the Bag-of-Ngrams Difference for Non-Autoregressive Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 198–205 (cited on page 474).
- [788] Peter Battaglia, Jessica Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çağlar Gülcehre, H. Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. “Relational inductive biases, deep learning, and graph networks”. In: volume abs/1806.01261. CoRR, 2018 (cited on page 476).
- [789] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. “Improve Transformer Models with Better Relative Position Embeddings”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 3327–3335 (cited on page 478).
- [790] Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. “Self-Attention with Structural Position Representations”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1403–1409 (cited on page 479).
- [791] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. “Neural Ordinary Differential Equations”. In: Annual Conference on Neural Information Processing Systems, 2018, pages 6572–6583 (cited on page 480).
- [792] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Xiangyang Xue, and Zheng Zhang. “Multi-Scale Self-Attention for Text Classification”. In: AAAI Conference on Artificial Intelligence, 2020, pages 7847–7854 (cited on page 481).
- [793] Kawin Ethayarajh. “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 55–65 (cited on page 482).
- [794] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated Residual Transformations for Deep Neural Networks”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 5987–5995 (cited on page 482).
- [795] David So, Quoc Le, and Chen Liang. “The Evolved Transformer”. In: volume 97. International Conference on Machine Learning, 2019, pages 5877–5886 (cited on pages 482, 509–512, 514).

- [796] Jianhao Yan, Fandong Meng, and Jie Zhou. “Multi-Unit Transformers for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 1047–1059 (cited on pages 482, 483).
- [797] Yang Fan, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. “Multi-branch Attentive Transformer”. In: volume abs/2006.10270. CoRR, 2020 (cited on pages 482, 483).
- [798] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. “Weighted Transformer Network for Machine Translation”. In: volume abs/1711.02132. CoRR, 2017 (cited on page 483).
- [799] 李北, 王强, 肖桐, 姜雨帆, 张哲旸, 刘继强, 张俐, and 于清. “面向神经机器翻译的集成学习方法分析”. In: volume 33. 3. 中文信息学报, 2019 (cited on page 483).
- [800] Andreas Veit, Michael Wilber, and Serge Belongie. “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”. In: Annual Conference on Neural Information Processing Systems, 2016, pages 550–558 (cited on page 484).
- [801] Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. “Highway and Residual Networks learn Unrolled Iterative Estimation”. In: International Conference on Learning Representations, 2017 (cited on pages 484, 501).
- [802] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. “Multi-level Residual Networks from Dynamical Systems View”. In: International Conference on Learning Representations, 2018 (cited on page 484).
- [803] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. “Universal Transformers”. In: International Conference on Learning Representations, 2019 (cited on page 484).
- [804] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: International Conference on Learning Representations, 2020 (cited on page 484).
- [805] Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. “Modeling Recurrence for Transformer”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1198–1207 (cited on page 485).

- [806] Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. “Blockwise Self-Attention for Long Document Understanding”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 2555–2565 (cited on page 485).
- [807] Peter Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. “Generating Wikipedia by Summarizing Long Sequences”. In: International Conference on Learning Representations, 2018 (cited on pages 485, 486).
- [808] Iz Beltagy, Matthew Peters, and Arman Cohan. “Longformer: The Long-Document Transformer”. In: volume abs/2004.05150. CoRR, 2020 (cited on pages 485, 516).
- [809] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. “Efficient Content-Based Sparse Attention with Routing Transformers”. In: volume abs/2003.05997. CoRR, 2020 (cited on page 486).
- [810] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. “Rethinking Attention with Performers”. In: volume abs/2009.14794. CoRR, 2020 (cited on pages 487, 516).
- [811] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. “On Layer Normalization in the Transformer Architecture”. In: volume abs/2002.04745. International Conference on Machine Learning, 2020 (cited on page 487).
- [812] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. “Understanding the Difficulty of Training Transformers”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 5747–5763 (cited on pages 487, 496).
- [813] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity Mappings in Deep Residual Networks”. In: volume 9908. European Conference on Computer Vision, 2016, pages 630–645 (cited on page 488).
- [814] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 48–53 (cited on pages 488, 599).

- [815] Bei Li, Ziyang Wang, Hui Liu, Quan Du, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. “Learning Light-Weight Translation Models from Deep Transformer”. In: volume abs/2012.13866. CoRR, 2020 (cited on page 489).
- [816] Xiangpeng Wei, Heng Yu, Yue Hu, Yue Zhang, Rongxiang Weng, and Weihua Luo. “Multiscale Collaborative Deep Models for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 414–426 (cited on page 493).
- [817] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. “Training Very Deep Networks”. In: Conference on Neural Information Processing Systems, 2015, pages 2377–2385 (cited on page 493).
- [818] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. “The Shattered Gradients Problem: If resnets are the answer, then what is the question?” In: volume 70. International Conference on Machine Learning, 2017, pages 342–350 (cited on page 493).
- [819] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. “A Convergence Theory for Deep Learning via Over-Parameterization”. In: volume 97. International Conference on Machine Learning, 2019, pages 242–252 (cited on page 493).
- [820] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. “Gradient Descent Finds Global Minima of Deep Neural Networks”. In: volume 97. International Conference on Machine Learning, 2019, pages 1675–1685 (cited on page 493).
- [821] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: IEEE International Conference on Computer Vision, 2015, pages 1026–1034 (cited on page 494).
- [822] Hongfei Xu, Qiuwei Liu, Josef van Genabith, Deyi Xiong, and Jingyi Zhang. “Lipschitz Constrained Parameter Initialization for Deep Transformers”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 397–402 (cited on page 495).
- [823] Xiao Shi Huang, Juan Perez, Jimmy Ba, and Maksims Volkovs. “Improving Transformer Optimization Through Better Initialization”. In: International Conference on Machine Learning, 2020 (cited on page 496).
- [824] Lijun Wu, Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. “Depth Growing for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5558–5563 (cited on pages 497, 498).

- [825] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger. “Densely Connected Convolutional Networks”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 2261–2269 (cited on page 498).
- [826] Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. “Modeling Source Syntax for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 688–697 (cited on pages 502, 505, 507).
- [827] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. “Tree-to-Sequence Attentional Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on page 502).
- [828] Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. “Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1936–1945 (cited on page 503).
- [829] Rico Sennrich and Barry Haddow. “Linguistic Input Features Improve Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 83–91 (cited on page 504).
- [830] Xing Shi, Inkit Padhi, and Kevin Knight. “Does String-Based Neural MT Learn Source Syntax?” In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 1526–1534 (cited on pages 507, 508).
- [831] Emanuele Bugliarello and Naoaki Okazaki. “Enhancing Machine Translation with Dependency-Aware Self-Attention”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 1618–1627 (cited on page 507).
- [832] David Alvarez-Melis and Tommi Jaakkola. “Tree-structured decoding with doubly-recurrent neural networks”. In: International Conference on Learning Representations, 2017 (cited on page 507).
- [833] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah Smith. “Recurrent Neural Network Grammars”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 199–209 (cited on page 507).
- [834] Minh-Thang Luong, Quoc Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. “Multi-task Sequence to Sequence Learning”. In: International Conference on Learning Representations, 2016 (cited on pages 508, 528).
- [835] Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. “Sequence-to-Dependency Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 698–707 (cited on page 508).

- [836] Barret Zoph and Quoc Le. “Neural Architecture Search with Reinforcement Learning”. In: International Conference on Learning Representations, 2017 (cited on pages 509, 511–513).
- [837] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc Le. “Learning Transferable Architectures for Scalable Image Recognition”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pages 8697–8710 (cited on pages 509, 514).
- [838] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc Le. “Aging Evolution for Image Classifier Architecture Search”. In: AAAI Conference on Artificial Intelligence, 2019 (cited on pages 509, 514).
- [839] Geoffrey Miller, Peter Todd, and Shailesh Hegde. “Designing Neural Networks using Genetic Algorithms”. In: International Conference on Genetic Algorithms, 1989, pages 379–384 (cited on pages 509, 512).
- [840] John Koza and James Rice. “Genetic generation of both the weights and architecture for a neural network”. In: volume 2. international joint conference on neural networks, 1991, pages 397–404 (cited on page 509).
- [841] Steven Harp, Tariq Samad, and Aloke Guha. “Designing Application-Specific Neural Networks Using the Genetic Algorithm”. In: Advances in Neural Information Processing Systems, 1989, pages 447–454 (cited on page 509).
- [842] Hiroaki Kitano. “Designing Neural Networks Using Genetic Algorithms with Graph Generation System”. In: volume 4. 4. Complex Systems, 1990 (cited on page 509).
- [843] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “DARTS: Differentiable Architecture Search”. In: International Conference on Learning Representations, 2019 (cited on pages 510–513).
- [844] Yinqiao Li, Chi Hu, Yuhao Zhang, Nuo Xu, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and Changliang Li. “Learning Architectures from an Extended Search Space for Language Modeling”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 6629–6639 (cited on pages 510, 512–515).
- [845] Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. “Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3583–3588 (cited on pages 510, 514).
- [846] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. “Efficient Neural Architecture Search via Parameter Sharing”. In: volume 80. International Conference on Machine Learning, 2018, pages 4092–4101 (cited on page 511).

- [847] Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. “AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search”. In: International Joint Conference on Artificial Intelligence, 2020, pages 2463–2469 (cited on pages 511, 515).
- [848] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. “HAT: Hardware-Aware Transformers for Efficient Natural Language Processing”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 7675–7688 (cited on pages 511, 515).
- [849] Esteban Real, Chen Liang, David So, and Quoc Le. “AutoML-Zero: Evolving Machine Learning Algorithms From Scratch”. In: volume abs/2003.03384. CoRR, 2020 (cited on page 512).
- [850] Yang Fan, Fei Tian, Yingce Xia, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. “Searching Better Architectures for Neural Machine Translation”. In: volume 28. IEEE Transactions on Audio, Speech, and Language Processing, 2020, pages 1574–1585 (cited on pages 512, 514).
- [851] Peter Angeline, Gregory Saunders, and Jordan Pollack. “An evolutionary algorithm that constructs recurrent neural networks”. In: volume 5. 1. IEEE Transactions on Neural Networks, 1994, pages 54–65 (cited on page 512).
- [852] Kenneth Stanley and Risto Miikkulainen. “Evolving neural networks through augmenting topologies”. In: volume 10. 2. Evolutionary computation, 2002, pages 99–127 (cited on page 512).
- [853] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Sue-matsu, Jie Tan, Quoc Le, and Alexey Kurakin. “Large-Scale Evolution of Image Classifiers”. In: volume 70. International Conference on Machine Learning, 2017, pages 2902–2911 (cited on pages 512, 514).
- [854] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. “Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution”. In: International Conference on Learning Representations, 2019 (cited on pages 512, 514).
- [855] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. “Hierarchical Representations for Efficient Architecture Search”. In: International Conference on Learning Representations, 2018 (cited on page 512).

- [856] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. “FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pages 10734–10742 (cited on page 513).
- [857] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. “PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search”. In: International Conference on Learning Representations, 2020 (cited on page 513).
- [858] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. “Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets”. In: volume 54. International Conference on Artificial Intelligence and Statistics, 2017, pages 528–536 (cited on page 514).
- [859] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. “A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets”. In: volume abs/1707.08819. CoRR, 2017 (cited on page 514).
- [860] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. “Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search”. In: International Conference on Machine Learning, 2018 (cited on page 514).
- [861] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. “Efficient Architecture Search by Network Transformation”. In: AAAI Conference on Artificial Intelligence, 2018, pages 2787–2794 (cited on page 514).
- [862] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. “Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves”. In: International Joint Conference on Artificial Intelligence, 2015, pages 3460–3468 (cited on page 514).
- [863] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. “Learning Curve Prediction with Bayesian Neural Networks”. In: International Conference on Learning Representations, 2017 (cited on page 514).
- [864] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. “Accelerating Neural Architecture Search using Performance Prediction”. In: International Conference on Learning Representations, 2018 (cited on page 514).
- [865] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. “Neural Architecture Optimization”. In: Advances in Neural Information Processing Systems, 2018, pages 7827–7838 (cited on page 514).

- [866] Yingce Xia, Xu Tan, Fei Tian, Fei Gao, Di He, Weicong Chen, Yang Fan, Linyuan Gong, Yichong Leng, Renqian Luo, Yiren Wang, Lijun Wu, Jinhua Zhu, Tao Qin, and Tie-Yan Liu. “Microsoft Research Asia’s Systems for WMT19”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 424–433 (cited on page 514).
- [867] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. “Searching for Activation Functions”. In: International Conference on Learning Representations, 2018 (cited on page 514).
- [868] Wei Zhu, Xiaoling Wang, Xipeng Qiu, Yuan Ni, and Guotong Xie. “AutoTrans: Automating Transformer Design via Reinforced Architecture Search”. In: volume abs/2009.02070 CoRR, 2020 (cited on page 515).
- [869] Henry Tsai, Jayden Ooi, Chun-Sung Ferng, Hyung Won Chung, and Jason Riesa. “Finding Fast Transformers: One-Shot Neural Architecture Search by Component Composition”. In: volume abs/2008.06808. CoRR, 2020 (cited on page 515).
- [870] Jian Li, Zhaopeng Tu, Baosong Yang, Michael Lyu, and Tong Zhang. “Multi-Head Attention with Disagreement Regularization”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2897–2903 (cited on page 516).
- [871] Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. “Multi-Granularity Self-Attention for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 887–897 (cited on page 516).
- [872] Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li, and Qi Su. “Learning When to Concentrate or Divert Attention: Self-Adaptive Attention Temperature for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2985–2990 (cited on page 516).
- [873] Hendra Setiawan, Matthias Sperber, Udhayakumar Nallasamy, and Matthias Paulik. “Variational Neural Machine Translation with Normalizing Flows”. In: Annual Meeting of the Association for Computational Linguistics, 2020 (cited on page 516).
- [874] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. “Star-Transformer”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1315–1325 (cited on page 516).
- [875] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. “Data Augmentation for Low-Resource Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 567–573 (cited on pages 517, 521).

- [876] Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. “SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 856–861 (cited on pages 517, 522, 551).
- [877] Yuval Marton, Chris Callison-Burch, and Philip Resnik. “Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 381–390 (cited on page 517).
- [878] Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. “Paraphrasing Revisited with Neural Machine Translation”. In: Annual Conference of the European Association for Machine Translation, 2017, pages 881–893 (cited on pages 517, 522).
- [879] Connor Shorten and Taghi M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. In: volume 6. Journal of Big Data, 2019, page 60 (cited on page 518).
- [880] Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. “Iterative Back-Translation for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 18–24 (cited on pages 518, 519).
- [881] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc’Aurelio Ranzato. “Unsupervised Machine Translation Using Monolingual Corpora Only”. In: International Conference on Learning Representations, 2018 (cited on pages 518, 520, 543).
- [882] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc’Aurelio Ranzato. “Phrase-Based & Neural Unsupervised Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 5039–5049 (cited on pages 518, 542, 544).
- [883] Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. “Copied Monolingual Data Improves Low-Resource Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 148–156 (cited on page 519).
- [884] Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. “Enhancement of Encoder and Attention Using Target Monolingual Corpora in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 55–63 (cited on page 519).

- [885] Lijun Wu, Yiren Wang, Yingce Xia, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. “Exploiting Monolingual Data at Scale for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4205–4215 (cited on pages 519, 520, 551).
- [886] Myle Ott, Michael Auli, David Grangier, and Marc’Aurelio Ranzato. “Analyzing Uncertainty in Neural Machine Translation”. In: volume 80. International Conference on Machine Learning, 2018, pages 3953–3962 (cited on page 519).
- [887] Jiajun Zhang and Chengqing Zong. “Exploiting Source-side Monolingual Data in Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 1535–1545 (cited on pages 520, 521, 528, 551).
- [888] Wael Farhan, Bashar Talafha, Analle Abuammar, Ruba Jaikat, Mahmoud Al-Ayyoub, Ahmad Bisher Tarakji, and Anas Toma. “Unsupervised dialectal neural machine translation”. In: volume 57. 3. Information Processing & Management, 2020, page 102181 (cited on pages 520, 545).
- [889] Rahul Bhagat and Eduard Hovy. “What Is a Paraphrase?” In: volume 39. 3. Computational Linguistics, 2013, pages 463–472 (cited on page 522).
- [890] Nitin Madnani and Bonnie Dorr. “Generating Phrasal and Sentential Paraphrases: A Survey of Data-Driven Methods”. In: volume 36. 3. Computational Linguistics, 2010, pages 341–387 (cited on page 522).
- [891] Yinuo Guo and Junfeng Hu. “Meteor++ 2.0: Adopt Syntactic Level Paraphrase Knowledge into Machine Translation Evaluation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 501–506 (cited on page 522).
- [892] Zhong Zhou, Matthias Sperber, and Alexander Waibel. “Paraphrases as Foreign Languages in Multilingual Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 113–122 (cited on page 522).
- [893] Sisay Fissaha Adafre and Maarten de Rijke. “Finding Similar Sentences across Multiple Languages in Wikipedia”. In: Annual Conference of the European Association for Machine Translation, 2006 (cited on pages 522, 523).
- [894] Dragos Stefan Munteanu and Daniel Marcu. “Improving Machine Translation Performance by Exploiting Non-Parallel Corpora”. In: volume 31. 4. Computational Linguistics, 2005, pages 477–504 (cited on pages 522, 523).
- [895] Lijun Wu, Jinhua Zhu, Di He, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. “Machine Translation With Weakly Paired Documents”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4374–4383 (cited on pages 522, 523).

- [896] Keiji Yasuda and Eiichiro Sumita. “Method for building sentence-aligned corpus from wikipedia”. In: AAAI Conference on Artificial Intelligence, 2008 (cited on page 523).
- [897] Jason Smith, Chris Quirk, and Kristina Toutanova. “Extracting Parallel Sentences from Comparable Corpora using Document Level Alignment”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 403–411 (cited on page 523).
- [898] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. “Exploiting Similarities among Languages for Machine Translation”. In: volume abs/1309.4168. CoRR, 2013 (cited on pages 523, 538, 540).
- [899] Sebastian Ruder, Ivan Vulic, and Anders Søgaard. “A Survey of Cross-lingual Word Embedding Models”. In: volume 65. Journal of Artificial Intelligence Research, 2019, pages 569–631 (cited on page 523).
- [900] Gulcehre Caglar, Firat Orhan, Xu Kelvin, Cho Kyunghyun, Barrault Loic, Lin Huei Chi, Bougares Fethi, Schwenk Holger, and Bengio Yoshua. “On Using Monolingual Corpora in Neural Machine Translation”. In: Computer Science, 2015 (cited on pages 523, 550).
- [901] Çaglar Gülcöhre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. “On integrating a language model into neural machine translation”. In: volume 45. Computational Linguistics, 2017, pages 137–148 (cited on page 523).
- [902] Felix Stahlberg, James Cross, and Veselin Stoyanov. “Simple Fusion: Return of the Language Model”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 204–211 (cited on page 523).
- [903] Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. “Context Gates for Neural Machine Translation”. In: volume 5. Annual Meeting of the Association for Computational Linguistics, 2017, pages 87–99 (cited on page 524).
- [904] Andrew Dai and Quoc Le. “Semi-supervised Sequence Learning”. In: Annual Conference on Neural Information Processing Systems, 2015, pages 3079–3087 (cited on page 524).
- [905] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: deep neural networks with multitask learning”. In: volume 307. International Conference on Machine Learning, 2008, pages 160–167 (cited on page 524).
- [906] Felipe Almeida and Geraldo Xexéo. “Word Embeddings: A Survey”. In: CoRR, 2019 (cited on page 524).

- [907] Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. “A Bag of Useful Tricks for Practical Neural Machine Translation: Embedding Layer Initialization and Large Batch Size”. In: Asian Federation of Natural Language Processing, 2017, pages 99–109 (cited on page 524).
- [908] Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. “When and Why are Pre-trained Word Embeddings Useful for Neural Machine Translation?” In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018 (cited on pages 524, 527).
- [909] Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. “Semi-supervised sequence tagging with bidirectional language models”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1756–1765 (cited on page 525).
- [910] Stéphane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. “On the use of BERT for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 108–117 (cited on pages 526, 527).
- [911] Kenji Imamura and Eiichiro Sumita. “Recycling a Pre-trained BERT Encoder for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 23–31 (cited on pages 526, 527).
- [912] Sergey Edunov, Alexei Baevski, and Michael Auli. “Pre-trained language model representations for language generation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 4052–4059 (cited on page 526).
- [913] Tianyu He, Xu Tan, and Tao Qin. “Hard but Robust, Easy but Sensitive: How Encoder and Decoder Perform in Neural Machine Translation”. In: volume abs/1908.06259. CoRR, 2019 (cited on page 526).
- [914] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. “Incorporating BERT into Neural Machine Translation”. In: International Conference on Learning Representations, 2020 (cited on page 526).
- [915] Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu, and Lei Li. “Towards Making the Most of BERT in Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 9378–9385 (cited on pages 526, 527).

- [916] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. “MASS: Masked Sequence to Sequence Pre-training for Language Generation”. In: volume 97. International Conference on Machine Learning, 2019, pages 5926–5936 (cited on pages 526, 551).
- [917] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 7871–7880 (cited on pages 526, 527, 551).
- [918] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. “ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 2401–2410 (cited on page 526).
- [919] Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. “Acquiring Knowledge from Pre-Trained Model to Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 9266–9273 (cited on page 527).
- [920] Yinhan Liu, Jitao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. “Multilingual Denoising Pre-training for Neural Machine Translation”. In: volume 8. Transactions of the Association for Computational Linguistics, 2020, pages 726–742 (cited on page 527).
- [921] Baijun Ji, Zhirui Zhang, Xiangyu Duan, Min Zhang, Boxing Chen, and Weihua Luo. “Cross-Lingual Pre-Training Based Transfer for Zero-Shot Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2020, pages 115–122 (cited on page 527).
- [922] Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. “CSP: Code-Switching Pre-training for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 2624–2636 (cited on page 527).
- [923] Dusan Varis and Ondrej Bojar. “Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 130–135 (cited on page 527).
- [924] Sebastian Ruder. “An Overview of Multi-Task Learning in Deep Neural Networks”. In: volume abs/1706.05098. CoRR, 2017 (cited on page 527).
- [925] Rich Caruana. “Multitask Learning”. In: Springer, 1998, pages 95–133 (cited on page 527).

- [926] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. “Multi-Task Deep Neural Networks for Natural Language Understanding”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4487–4496 (cited on page 527).
- [927] Tobias Domhan and Felix Hieber. “Using Target-side Monolingual Data for Neural Machine Translation through Multi-task Learning”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1500–1505 (cited on pages 528, 550).
- [928] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. “Multi-Task Learning for Multiple Language Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 1723–1732 (cited on pages 528, 536, 551).
- [929] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. “Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation”. In: volume 5. Transactions of the Association for Computational Linguistics, 2017, pages 339–351 (cited on pages 528, 532, 536, 537, 551).
- [930] Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. “Joint Training for Neural Machine Translation Models with Monolingual Data”. In: AAAI Conference on Artificial Intelligence, 2018, pages 555–562 (cited on page 529).
- [931] Meng Sun, Bojian Jiang, Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. “Baidu Neural Machine Translation Systems for WMT19”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 374–381 (cited on page 529).
- [932] Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. “Dual Supervised Learning”. In: volume 70. International Conference on Machine Learning, 2017, pages 3789–3798 (cited on pages 530, 531).
- [933] Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. “Model-Level Dual Learning”. In: volume 80. Proceedings of Machine Learning Research. International Conference on Machine Learning, 2018, pages 5379–5388 (cited on page 530).
- [934] Tao Qin. “Dual Learning for Machine Translation and Beyond”. In: Springer, 2020, pages 49–72 (cited on pages 530, 531).

- [935] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. “Dual Learning for Machine Translation”. In: 2016, pages 820–828 (cited on page 530).
- [936] Zhibing Zhao, Yingce Xia, Tao Qin, Lirong Xia, and Tie-Yan Liu. “Dual Learning: Theoretical Study and an Algorithmic Extension”. In: arXiv preprint arXiv:2005.08238, 2020 (cited on page 530).
- [937] Richard Sutton, David Allen McAllester, Satinder Singh, and Yishay Mansour. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: The MIT Press, 1999, pages 1057–1063 (cited on page 532).
- [938] Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. “A survey of multilingual neural machine translation”. In: volume 53. 5. ACM Computing Surveys, 2020, pages 1–38 (cited on pages 532, 536).
- [939] Hua Wu and Haifeng Wang. “Pivot language approach for phrase-based statistical machine translation”. In: volume 21. 3. Machine Translation, 2007, pages 165–181 (cited on pages 532, 533).
- [940] Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, and Hermann Ney. “Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 866–876 (cited on pages 532, 533).
- [941] Masao Utiyama and Hitoshi Isahara. “A Comparison of Pivot Methods for Phrase-Based Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2007, pages 484–491 (cited on page 533).
- [942] Samira Tofighi Zahabi, Somayeh Bakhshaei, and Shahram Khadivi. “Using Context Vectors in Improving a Machine Translation System with Bridge Language”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 318–322 (cited on page 533).
- [943] Xiaoning Zhu, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang, and Tiejun Zhao. “Improving Pivot-Based Statistical Machine Translation by Pivoting the Co-occurrence Count of Phrase Pairs”. In: Conference on Empirical Methods in Natural Language Processing, 2014, pages 1665–1675 (cited on page 533).
- [944] Akiva Miura, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. “Improving Pivot Translation by Remembering the Pivot”. In: Annual Meeting of the Association for Computational Linguistics, 2015, pages 573–577 (cited on page 533).

- [945] Trevor Cohn and Mirella Lapata. “Machine Translation by Triangulation: Making Effective Use of Multi-Parallel Corpora”. In: Annual Meeting of the Association for Computational Linguistics, 2007 (cited on page 533).
- [946] Hua Wu and Haifeng Wang. “Revisiting Pivot Language Approach for Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 154–162 (cited on page 533).
- [947] Adrià De Gispert and Jose B Marino. “Catalan-English statistical machine translation without parallel corpus: bridging through Spanish”. In: International Conference on Language Resources and Evaluation, 2006, pages 65–68 (cited on page 533).
- [948] Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. “Neural Machine Translation with Pivot Languages”. In: volume abs/1611.04928. CoRR, 2016 (cited on page 533).
- [949] Michael Paul, Hirofumi Yamamoto, Eiichiro Sumita, and Satoshi Nakamura. “On the Importance of Pivot Language Selection for Statistical Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009, pages 221–224 (cited on page 533).
- [950] Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. “Multilingual Neural Machine Translation with Knowledge Distillation”. In: International Conference on Learning Representations, 2019 (cited on page 533).
- [951] Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. “Meta-Learning for Low-Resource Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 3622–3631 (cited on page 535).
- [952] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: volume 70. Proceedings of Machine Learning Research. International Conference on Machine Learning, 2017, pages 1126–1135 (cited on page 535).
- [953] Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. “Universal Neural Machine Translation for Extremely Low Resource Languages”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 344–354 (cited on page 535).
- [954] Tom Kocmi and Ondrej Bojar. “Trivial Transfer Learning for Low-Resource Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 244–252 (cited on page 536).

- [955] Baijun Ji, Zhirui Zhang, Xiangyu Duan, Min Zhang, Boxing Chen, and Weihua Luo. “Cross-Lingual Pre-Training Based Transfer for Zero-Shot Neural Machine Translation”. In: volume 34. 01. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pages 115–122 (cited on page 536).
- [956] Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. “Pre-training Multilingual Neural Machine Translation by Leveraging Alignment Information”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 2649–2663 (cited on page 536).
- [957] Matiss Rikters, Marcis Pinnis, and Rihards Krislauks. “Training and Adapting Multilingual NMT for Less-resourced and Morphologically Rich Languages”. In: European Language Resources Association, 2018 (cited on page 536).
- [958] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. “Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2016, pages 866–875 (cited on pages 536, 551).
- [959] Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. “Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 1628–1639 (cited on page 537).
- [960] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. “Beyond English-Centric Multilingual Machine Translation”. In: volume abs/2010.11125. CoRR, 2020 (cited on page 537).
- [961] 黄书剑. “统计机器翻译中的词对齐研究”. In: 南京大学, 2012 (cited on page 538).
- [962] Ivan Vulic and Anna Korhonen. “On the Role of Seed Lexicons in Learning Bilateral Word Embeddings”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on page 538).
- [963] Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. “Offline bilingual word vectors, orthogonal transformations and the inverted softmax”. In: International Conference on Learning Representations, 2017 (cited on pages 538, 540).
- [964] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. “Learning bilingual word embeddings with (almost) no bilingual data”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 451–462 (cited on page 538).

- [965] Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. “Unsupervised Cross-lingual Transfer of Word Embedding Spaces”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2465–2474 (cited on pages 538–540).
- [966] Schnemann and Peter. “A generalized solution of the orthogonal procrustes problem”. In: volume 31. 1. Psychometrika, 1966, pages 1–10 (cited on page 539).
- [967] Guillaume Lample, Alexis Conneau, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. “Word translation without parallel data”. In: International Conference on Learning Representations, 2018 (cited on pages 539, 540).
- [968] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. “Adversarial Training for Unsupervised Bilingual Lexicon Induction”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1959–1970 (cited on pages 539, 540).
- [969] Tasnim Mohiuddin and Shafiq Rayhan Joty. “Revisiting Adversarial Autoencoder for Unsupervised Word Translation with Cycle Consistency and Improved Training”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3857–3867 (cited on pages 539, 540).
- [970] David Alvarez-Melis and Tommi S. Jaakkola. “Gromov-Wasserstein Alignment of Word Embedding Spaces”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 1881–1890 (cited on pages 539, 540).
- [971] Nicolas Garneau, Mathieu Godbout, David Beauchemin, Audrey Durand, and Luc Lamontagne. “A Robust Self-Learning Method for Fully Unsupervised Cross-Lingual Mappings of Word Embeddings: Making the Method Robustly Reproducible as Well”. In: Language Resources and Evaluation Conference, 2020, pages 5546–5554 (cited on page 539).
- [972] Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. “Unsupervised Hyperalignment for Multilingual Word Embeddings”. In: International Conference on Learning Representations, 2018 (cited on pages 539, 540).
- [973] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. “Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2015, pages 1006–1011 (cited on page 540).

- [974] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. “Earth Mover’s Distance Minimization for Unsupervised Bilingual Lexicon Induction”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1934–1945 (cited on page 540).
- [975] Mareike Hartmann, Yova Kementchedjhieva, and Anders Søgaard. “Empirical observations on the instability of aligning word vector spaces with GANs”. In: openreview.net, 2018 (cited on page 540).
- [976] Zi-Yi Dou, Zhi-Hao Zhou, and Shujian Huang. “Unsupervised Bilingual Lexicon Induction via Latent Variable Models”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 621–626 (cited on page 540).
- [977] Jiaji Huang, Qiang Qiu, and Kenneth Church. “Hubless Nearest Neighbor Search for Bilingual Lexicon Induction”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 4072–4080 (cited on page 540).
- [978] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. “Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 2979–2984 (cited on page 540).
- [979] Xilun Chen and Claire Cardie. “Unsupervised Multilingual Word Embeddings”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 261–270 (cited on page 540).
- [980] Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. “Multilingual word translation using auxiliary languages”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1330–1335 (cited on page 540).
- [981] Geert Heyman, Bregt Verreet, Ivan Vulic, and Marie-Francine Moens. “Learning Unsupervised Multilingual Word Embeddings with Incremental Multilingual Hubs”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1890–1902 (cited on page 540).
- [982] Yedid Hoshen and Lior Wolf. “Non-Adversarial Unsupervised Word Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 469–478 (cited on page 540).
- [983] Tanmoy Mukherjee, Makoto Yamada, and Timothy Hospedales. “Learning Unsupervised Word Translations Without Adversaries”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 627–632 (cited on page 540).

- [984] Ivan Vulic, Goran Glavas, Roi Reichart, and Anna Korhonen. “Do We Really Need Fully Unsupervised Cross-Lingual Embeddings?” In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 4406–4417 (cited on page 540).
- [985] Yanyang Li, Yingfeng Luo, Ye Lin, Quan Du, Huizhen Wang, Shujian Huang, Tong Xiao, and Jingbo Zhu. “A Simple and Effective Approach to Robust Unsupervised Bilingual Dictionary Induction”. In: International Conference on Computational Linguistics, 2020 (cited on pages 540, 541).
- [986] Anders Søgaard, Sebastian Ruder, and Ivan Vulic. “On the Limitations of Unsupervised Bilingual Dictionary Induction”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 778–788 (cited on page 541).
- [987] Benjamin Marie and Atsushi Fujita. “Iterative Training of Unsupervised Neural and Statistical Machine Translation Systems”. In: volume 19. 5. ACM Transactions on Asian and Low-Resource Language Information Processing, 2020, 68:1–68:21 (cited on page 541).
- [988] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. “Unsupervised Statistical Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 3632–3642 (cited on pages 541, 542, 545).
- [989] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. “An Effective Approach to Unsupervised Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 194–203 (cited on page 542).
- [990] Nima Pourdamghani, Nada Aldarrab, Marjan Ghazvininejad, Kevin Knight, and Jonathan May. “Translating Translationese: A Two-Step Approach to Unsupervised Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3057–3062 (cited on page 543).
- [991] Alexis Conneau and Guillaume Lample. “Cross-lingual Language Model Pretraining”. In: Annual Conference on Neural Information Processing Systems, 2019, pages 7057–7067 (cited on pages 545, 551).
- [992] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: IEEE International Conference on Computer Vision, 2017, pages 843–852 (cited on page 547).
- [993] Kevin Duh, Graham Neubig, Katsuhiro Sudoh, and Hajime Tsukada. “Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 678–683 (cited on page 547).

- [994] Spyros Matsoukas, Antti-Veikko I. Rostí, and Bing Zhang. “Discriminative Corpus Weight Estimation for Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2009, pages 708–717 (cited on page 547).
- [995] George F. Foster, Cyril Goutte, and Roland Kuhn. “Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2010, pages 451–459 (cited on page 547).
- [996] Jingbo Zhu and Eduard H. Hovy. “Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem”. In: Conference on Empirical Methods in Natural Language Processing, 2007, pages 783–790 (cited on page 547).
- [997] Kashif Shah, Loïc Barrault, and Holger Schwenk. “Translation Model Adaptation by Resampling”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 392–399 (cited on page 547).
- [998] Masao Utiyama and Hitoshi Isahara. “Reliable Measures for Aligning Japanese-English News Articles and Sentences”. In: Annual Meeting of the Association for Computational Linguistics, 2003, pages 72–79 (cited on page 548).
- [999] Nicola Bertoldi and Marcello Federico. “Domain Adaptation for Statistical Machine Translation with Monolingual Resources”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 182–189 (cited on page 548).
- [1000] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. “An Empirical Comparison of Domain Adaptation Methods for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 385–391 (cited on pages 548, 550).
- [1001] Mohammad Amin Farajian, Marco Turchi, Matteo Negri, Nicola Bertoldi, and Marcello Federico. “Neural vs. Phrase-Based Machine Translation in a Multi-Domain Scenario”. In: Annual Conference of the European Association for Machine Translation, 2017, pages 280–284 (cited on page 549).
- [1002] Jiali Zeng, Yang Liu, Jinsong Su, Yubin Ge, Yaojie Lu, Yongjing Yin, and Jiebo Luo. “Iterative Dual Domain Adaptation for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 845–855 (cited on page 550).

- [1003] Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. “Regularization techniques for fine-tuning in neural machine translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 1489–1494 (cited on pages 550, 579).
- [1004] Huda Khayrallah, Gaurav Kumar, Kevin Duh, Matt Post, and Philipp Koehn. “Neural lattice search for domain adaptation in machine translation”. In: International Joint Conference on Natural Language Processing, 2017, pages 20–25 (cited on page 550).
- [1005] Rico Sennrich. “Perplexity Minimization for Translation Model Domain Adaptation in Statistical Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 539–549 (cited on page 550).
- [1006] Markus Freitag and Yaser Al-Onaizan. “Fast Domain Adaptation for Neural Machine Translation”. In: volume abs/1612.06897. CoRR, 2016 (cited on page 550).
- [1007] Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. “Domain Adaptive Inference for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 222–228 (cited on page 550).
- [1008] Ankur Bapna and Orhan Firat. “Non-Parametric Adaptation for Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1921–1931 (cited on page 550).
- [1009] Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. “Generalized Data Augmentation for Low-Resource Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5786–5796 (cited on page 551).
- [1010] Marzieh Fadaee and Christof Monz. “Back-Translation Sampling by Targeting Difficult Words in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 436–446 (cited on page 551).
- [1011] Nuo Xu, Yiniao Li, Chen Xu, Yanyang Li, Bei Li, Tong Xiao, and Jingbo Zhu. “Analysis of Back-Translation Methods for Low-Resource Neural Machine Translation”. In: volume 11839. Natural Language Processing and Chinese Computing, 2019, pages 466–475 (cited on page 551).
- [1012] Isaac Caswell, Ciprian Chelba, and David Grangier. “Tagged Back-Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 53–63 (cited on page 551).

- [1013] Zi-Yi Dou, Antonios Anastasopoulos, and Graham Neubig. “Dynamic Data Selection and Weighting for Iterative Back-Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 5894–5904 (cited on page 551).
- [1014] Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. “Improving Back-Translation with Uncertainty-based Confidence Estimation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 791–802 (cited on page 551).
- [1015] Guanlin Li, Lemao Liu, Guoping Huang, Conghui Zhu, and Tiejun Zhao. “Understanding Data Augmentation in Neural Machine Translation: Two Perspectives towards Generalization”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5688–5694 (cited on page 551).
- [1016] Benjamin Marie, Raphael Rubino, and Atsushi Fujita. “Tagged Back-translation Revisited: Why Does It Really Work?” In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 5990–5997 (cited on page 551).
- [1017] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: Annual Conference on Neural Information Processing Systems, 2019, pages 5754–5764 (cited on page 551).
- [1018] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: International Conference on Learning Representations, 2020 (cited on page 551).
- [1019] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. “ERNIE: Enhanced Language Representation with Informative Entities”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1441–1451 (cited on page 551).
- [1020] Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Dixin Jiang, and Ming Zhou. “Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 2485–2494 (cited on page 551).
- [1021] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. “VideoBERT: A Joint Model for Video and Language Representation Learning”. In: International Conference on Computer Vision, 2019, pages 7463–7472 (cited on page 551).

- [1022] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks”. In: Annual Annual Conference on Neural Information Processing Systems, 2019, pages 13–23 (cited on page 551).
- [1023] Yung-Sung Chuang, Chi-Liang Liu, Hung-yi Lee, and Lin-Shan Lee. “Speech-BERT: An Audio-and-Text Jointly Learned Language Model for End-to-End Spoken Question Answering”. In: Annual Conference of the International Speech Communication Association, 2020, pages 4168–4172 (cited on page 551).
- [1024] Matthew Peters, Sebastian Ruder, and Noah A. Smith. “To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 7–14 (cited on page 551).
- [1025] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. “How to Fine-Tune BERT for Text Classification?” In: volume 11856. Chinese Computational Linguistics, 2019, pages 194–206 (cited on page 551).
- [1026] Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. “Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder”. In: volume abs/1611.04798. CoRR, 2016 (cited on page 551).
- [1027] Graeme W. Blackwood, Miguel Ballesteros, and Todd Ward. “Multilingual Neural Machine Translation with Task-Specific Attention”. In: International Conference on Computational Linguistics, 2018, pages 3112–3122 (cited on page 551).
- [1028] Devendra Singh Sachan and Graham Neubig. “Parameter Sharing Methods for Multilingual Self-Attentional Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 261–271 (cited on page 551).
- [1029] Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. “A neural interlingua for multilingual machine translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 84–92 (cited on page 551).
- [1030] Yining Wang, Long Zhou, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. “A Compact and Language-Sensitive Multilingual Translation Method”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1213–1223 (cited on page 551).
- [1031] Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. “Multilingual Neural Machine Translation With Soft Decoupled Encoding”. In: International Conference on Learning Representations, 2019 (cited on page 551).

- [1032] Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. “Multilingual Neural Machine Translation with Language Clustering”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 963–973 (cited on page 551).
- [1033] Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. “Zero-Resource Translation with Multi-Lingual Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2016, pages 268–277 (cited on page 551).
- [1034] Lierni Sestorain, Massimiliano Ciaramita, Christian Buck, and Thomas Hofmann. “Zero-Shot Dual Machine Translation”. In: volume abs/1805.10338. CoRR, 2018 (cited on page 551).
- [1035] Maruan Al-Shedivat and Ankur P. Parikh. “Consistency by Agreement in Zero-Shot Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 1184–1197 (cited on page 551).
- [1036] Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. “The Missing Ingredient in Zero-Shot Neural Machine Translation”. In: volume abs/1903.07091. CoRR, 2019 (cited on page 551).
- [1037] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. “Improved Zero-shot Neural Machine Translation via Ignoring Spurious Correlations”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1258–1268 (cited on page 551).
- [1038] Anna Currey and Kenneth Heafield. “Zero-Resource Neural Machine Translation with Monolingual Pivot Data”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 99–107 (cited on page 551).
- [1039] 李琳 洪青阳. 语音识别:原理与应用. 电子工业出版社, 2020 (cited on pages 555, 556).
- [1040] 陈果果, 都家宇, 那兴宇, and 张俊博. *Kaldi* 语音识别实战. 电子工业出版社, 2020 (cited on page 555).
- [1041] Tara N. Sainath, Ron J. Weiss, Andrew W. Senior, Kevin W. Wilson, and Oriol Vinyals. “Learning the speech front-end with raw waveform CLDNNs”. In: Annual Conference of the International Speech Communication Association, 2015, pages 1–5 (cited on page 555).

- [1042] Abdel-rahman Mohamed, Geoffrey E. Hinton, and Gerald Penn. “Understanding how Deep Belief Networks perform acoustic modelling”. In: International Conference on Acoustics, Speech and Signal Processing, 2012, pages 4273–4276 (cited on page 555).
- [1043] Mark J. F. Gales and Steve J. Young. “The Application of Hidden Markov Models in Speech Recognition”. In: Found Trends Signal Process, 2007, pages 195–304 (cited on page 557).
- [1044] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton. “Acoustic Modeling Using Deep Belief Networks”. In: IEEE Transactions on Speech and Audio Processing, 2012, pages 14–22 (cited on page 557).
- [1045] G Hinton, L Deng, D Yu, GE Dahl, and B Kingsbury. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: IEEE Signal Processing Magazine, 2012, pages 82–97 (cited on page 557).
- [1046] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. “Attention-Based Models for Speech Recognition”. In: Annual Conference on Neural Information Processing Systems, 2015, pages 577–585 (cited on page 557).
- [1047] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition”. In: International Conference on Acoustics, Speech and Signal Processing, 2016, pages 4960–4964 (cited on page 557).
- [1048] Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. “An Attentional Model for Speech Translation Without Transcription”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2016, pages 949–959 (cited on page 559).
- [1049] Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. “Sequence-to-Sequence Models Can Directly Translate Foreign Speech”. In: International Symposium on Computer Architecture, 2017, pages 2625–2629 (cited on page 559).
- [1050] Alexandre Berard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. “Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation”. In: Conference and Workshop on Neural Information Processing Systems, 2016 (cited on page 559).

- [1051] Mattia Antonino Di Gangi, Matteo Negri, Roldano Cattoni, Roberto Dessì, and Marco Turchi. “Enhancing Transformer for End-to-end Speech-to-Text Translation”. In: European Association for Machine Translation, 2019, pages 21–31 (cited on page 559).
- [1052] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: volume 148. International Conference on Machine Learning, 2006, pages 369–376 (cited on page 559).
- [1053] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. “Hybrid CTC/Attention Architecture for End-to-End Speech Recognition”. In: IEEE Journal of Selected Topics in Signal Processing, 2017, pages 1240–1253 (cited on page 559).
- [1054] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. “Joint CTC-attention based end-to-end speech recognition using multi-task learning”. In: International Conference on Acoustics, Speech and Signal Processing, 2017, pages 4835–4839 (cited on page 559).
- [1055] Baoguang Shi, Xiang Bai, and Cong Yao. “An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, pages 2298–2304 (cited on page 559).
- [1056] Antonios Anastasopoulos and David Chiang. “Tied Multitask Learning for Neural Speech Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 82–91 (cited on page 560).
- [1057] Parnia Bahar, Tobias Bieschke, and Hermann Ney. “A Comparative Study on End-to-End Speech to Text Translation”. In: IEEE Automatic Speech Recognition and Understanding Workshop, 2019, pages 792–799 (cited on page 560).
- [1058] Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. “Pre-training on high-resource speech recognition improves low-resource speech-to-text translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 58–68 (cited on page 562).
- [1059] Alexandre Berard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. “End-to-End Automatic Speech Translation of Audiobooks”. In: International Conference on Acoustics, Speech and Signal Processing, 2018, pages 6224–6228 (cited on page 562).

- [1060] Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J. Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. “Leveraging Weakly Supervised Data to Improve End-to-end Speech-to-text Translation”. In: International Conference on Acoustics, Speech and Signal Processing, 2019, pages 7180–7184 (cited on page 562).
- [1061] Anne Wu, Changhan Wang, Juan Pino, and Jiatao Gu. “Self-Supervised Representations Improve End-to-End Speech Translation”. In: International Symposium on Computer Architecture, 2020, pages 1491–1495 (cited on page 562).
- [1062] Yuchen Liu, Hao Xiong, Jiajun Zhang, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. “End-to-End Speech Translation with Knowledge Distillation”. In: Annual Conference of the International Speech Communication Association, 2019, pages 1128–1132 (cited on page 562).
- [1063] Ashkan Alinejad and Anoop Sarkar. “Effectively pretraining a speech translation decoder with Machine Translation data”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 8014–8020 (cited on page 562).
- [1064] Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. “Structured-Based Curriculum Learning for End-to-End English-Japanese Speech Translation”. In: Annual Conference of the International Speech Communication Association, 2017, pages 2630–2634 (cited on page 562).
- [1065] Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. “Curriculum Pre-training for End-to-End Speech Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3728–3738 (cited on page 562).
- [1066] Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. “A Shared Task on Multimodal Machine Translation and Crosslingual Image Description”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 543–553 (cited on page 563).
- [1067] Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. “LIUM-CVC Submissions for WMT17 Multimodal Translation Task”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 432–439 (cited on page 563).
- [1068] Jindrich Libovický, Jindrich Helcl, Marek Tlustý, Ondrej Bojar, and Pavel Pecina. “CUNI System for WMT16 Automatic Post-Editing and Multimodal Translation Tasks”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 646–654 (cited on page 563).

- [1069] Iacer Calixto and Qun Liu. “Incorporating Global Visual Features into Attention-based Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 992–1003 (cited on page 563).
- [1070] Jean-Benoit Delbrouck and Stéphane Dupont. “Modulating and attending the source image during encoding improves Multimodal Translation”. In: Conference and Workshop on Neural Information Processing Systems, 2017 (cited on pages 563, 565).
- [1071] Jindrich Helcl, Jindrich Libovický, and Dusan Varis. “CUNI System for the WMT18 Multimodal Translation Task”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 616–623 (cited on page 563).
- [1072] Desmond Elliott and Ákos Kádár. “Imagination Improves Multimodal Translation”. In: International Joint Conference on Natural Language Processing, 2017, pages 130–141 (cited on pages 563, 565).
- [1073] Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. “A Novel Graph-based Multi-modal Fusion Encoder for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 3025–3035 (cited on page 563).
- [1074] Yuting Zhao, Mamoru Komachi, Tomoyuki Kajiwara, and Chenhui Chu. “Double Attention-based Multimodal Neural Machine Translation with Semantic Image Regions”. In: Annual Conference of the European Association for Machine Translation, 2020, pages 105–114 (cited on page 563).
- [1075] Desmond Elliott, Stella Frank, and Eva Hasler. “Multi-Language Image Description with Neural Sequence Models”. In: *CoRR* abs/1510.04709 (2015) (cited on page 563).
- [1076] Pranava Swaroop Madhyastha, Josiah Wang, and Lucia Specia. “Sheffield MultiMT: Using Object Posterior Predictions for Multimodal Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 470–476 (cited on page 563).
- [1077] Shaowei Yao and Xiaojun Wan. “Multimodal Transformer for Multimodal Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 4346–4350 (cited on page 565).
- [1078] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. “Hierarchical Question-Image Co-Attention for Visual Question Answering”. In: Conference on Neural Information Processing Systems, 2016, pages 289–297 (cited on page 565).

- [1079] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. “VQA: Visual Question Answering”. In: International Conference on Computer Vision, 2015, pages 2425–2433 (cited on page 565).
- [1080] Raffaella Bernardi, Ruket Çakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli İkizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. “Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures (Extended Abstract)”. In: International Joint Conference on Artificial Intelligence, 2017, pages 4970–4974 (cited on page 565).
- [1081] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. “Show and tell: A neural image caption generator”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pages 3156–3164 (cited on page 566).
- [1082] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: International Conference on Machine Learning, 2015, pages 2048–2057 (cited on pages 566, 567).
- [1083] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. “Image Captioning with Semantic Attention”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pages 4651–4659 (cited on page 567).
- [1084] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. “SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 6298–6306 (cited on page 567).
- [1085] Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. “Aligning Where to See and What to Tell: Image Captioning with Region-Based Attention and Scene-Specific Contexts”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, pages 2321–2334 (cited on page 567).
- [1086] Chang Liu, Fuchun Sun, Changhu Wang, Feng Wang, and Alan L. Yuille. “MAT: A Multimodal Attentive Translator for Image Captioning”. In: International Joint Conference on Artificial Intelligence, 2017, pages 4033–4039 (cited on page 567).
- [1087] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: CoRR, 2018 (cited on page 567).
- [1088] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: CoRR, 2020 (cited on page 567).

- [1089] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. “Exploring Visual Relationship for Image Captioning”. In: Lecture Notes in Computer Science. European Conference on Computer Vision, 2018 (cited on page 567).
- [1090] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. “Knowing When to Look: Adaptive Attention via a Visual Sentinel for Image Captioning”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pages 3242–3250 (cited on page 568).
- [1091] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. “Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pages 6077–6086 (cited on page 568).
- [1092] Jyoti Aneja, Aditya Deshpande, and Alexander G. Schwing. “Convolutional Image Captioning”. In: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pages 5561–5570 (cited on page 568).
- [1093] Fang Fang, Hanli Wang, Yihao Chen, and Pengjie Tang. “Looking deeper and transferring attention for image captioning”. In: Multimedia Tools Applications, 2018, pages 31159–31175 (cited on page 568).
- [1094] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. “Generative Adversarial Text to Image Synthesis”. In: International Conference on Machine Learning, 2016, pages 1060–1069 (cited on page 568).
- [1095] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. “Generative Adversarial Nets”. In: Conference on Neural Information Processing Systems, 2014, pages 2672–2680 (cited on page 568).
- [1096] Hajar Emami, Majid Moradi Aliabadi, Ming Dong, and Ratna Babu Chinnam. “SPA-GAN: Spatial Attention GAN for Image-to-Image Translation”. In: IEEE Transactions on Multimedia, 2019 (cited on page 568).
- [1097] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, and Muhammad Zeshan Afzal. “TAC-GAN - Text Conditioned Auxiliary Classifier Generative Adversarial Network”. In: CoRR, 2017 (cited on page 568).
- [1098] Yehoshua Bar-Hillel. “The Present Status of Automatic Translation of Languages”. In: volume 1. Advances in computers, 1960, pages 91–163 (cited on page 568).
- [1099] Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. “A Survey on Document-level Machine Translation: Methods and Evaluation”. In: volume abs/1912.08494. CoRR, 2019 (cited on page 568).

- [1100] Andrei Popescu-Belis. “Context in Neural Machine Translation: A Review of Models and Evaluations”. In: volume abs/1901.09115. CoRR, 2019 (cited on page 568).
- [1101] Daniel Marcu, Lynn Carlson, and Maki Watanabe. “The Automatic Translation of Discourse Structures”. In: Applied Natural Language Processing Conference, 2000, pages 9–17 (cited on page 569).
- [1102] George Foster, Pierre Isabelle, and Roland Kuhn. “Translating structured documents”. In: Proceedings of AMTA, 2010 (cited on page 569).
- [1103] Annie Louis and Bonnie L. Webber. “Structured and Unstructured Cache Models for SMT Domain Adaptation”. In: Annual Conference of the European Association for Machine Translation, 2014, pages 155–163 (cited on page 569).
- [1104] Christian Hardmeier and Marcello Federico. “Modelling pronominal anaphora in statistical machine translation”. In: International Workshop on Spoken Language Translation, 2010, pages 283–289 (cited on pages 569, 570).
- [1105] Ronan Le Nagard and Philipp Koehn. “Aiding Pronoun Translation with Co-Reference Resolution”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 252–261 (cited on page 569).
- [1106] Ngoc-Quang Luong and Andrei Popescu-Belis. “A Contextual Language Model to Improve Machine Translation of Pronouns by Re-ranking Translation Hypotheses”. In: European Association for Machine Translation, 2016, pages 292–304 (cited on page 569).
- [1107] Jörg Tiedemann. “Context adaptation in statistical machine translation using models with exponentially decaying cache”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 8–15 (cited on page 569).
- [1108] Zhengxian Gong, Min Zhang, and Guodong Zhou. “Cache-based Document-level Statistical Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2011, pages 909–919 (cited on page 569).
- [1109] Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lv, and Qun Liu. “Modeling Lexical Cohesion for Document-Level Machine Translation”. In: International Joint Conference on Artificial Intelligence, 2013, pages 2183–2189 (cited on page 569).
- [1110] Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. “Document-level consistency verification in machine translation”. In: *Machine Translation Summit*. Volume 13. 2011, pages 131–138 (cited on page 569).

- [1111] Thomas Meyer, Andrei Popescu-Belis, Sandrine Zufferey, and Bruno Cartoni. “Multilingual Annotation and Disambiguation of Discourse Connectives for Machine Translation”. In: Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2011, pages 194–203 (cited on page 569).
- [1112] Thomas Meyer and Andrei Popescu-Belis. “Using Sense-labeled Discourse Connectives for Statistical Machine Translation”. In: Hybrid Approaches to Machine Translation, 2012, pages 129–138 (cited on page 569).
- [1113] Jörg Tiedemann and Yves Scherrer. “Neural Machine Translation with Extended Context”. In: Proceedings of the Third Workshop on Discourse in Machine Translation, 2017, pages 82–92 (cited on pages 569, 571).
- [1114] Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. “Evaluating Discourse Phenomena in Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 1304–1313 (cited on pages 569–571).
- [1115] Annette Rios Gonzales, Laura Mascarell, and Rico Sennrich. “Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 11–19 (cited on pages 569, 571).
- [1116] Valentin Macé and Christophe Servan. “Using Whole Document Context in Neural Machine Translation”. In: The International Workshop on Spoken Language Translation, 2019 (cited on pages 569, 571).
- [1117] Sébastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. “Does Neural Machine Translation Benefit from Larger Context?” In: volume abs/1704.05135. CoRR, 2017 (cited on pages 569, 571, 572).
- [1118] Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. “Improving the Transformer Translation Model with Document-Level Context”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 533–542 (cited on pages 569, 571, 572).
- [1119] Sameen Maruf, André F. T. Martins, and Gholamreza Haffari. “Selective Attention for Context-aware Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pages 3092–3102 (cited on pages 569, 571, 574).
- [1120] Sameen Maruf and Gholamreza Haffari. “Document Context Neural Machine Translation with Memory Networks”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 1275–1284 (cited on pages 569, 574).

- [1121] Zhengxin Yang, Jinchao Zhang, Fandong Meng, Shuhao Gu, Yang Feng, and Jie Zhou. “Enhancing Context Modeling with a Query-Guided Capsule Network for Document-level Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1527–1537 (cited on pages 569, 574).
- [1122] Zaixiang Zheng, Xiang Yue, Shujian Huang, Jiajun Chen, and Alexandra Birch. “Towards Making the Most of Context in Neural Machine Translation”. In: International Joint Conference on Artificial Intelligence, 2020, pages 3983–3989 (cited on pages 569, 574).
- [1123] Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. “Modeling Coherence for Neural Machine Translation with Dynamic and Topic Caches”. In: International Conference on Computational Linguistics, 2018, pages 596–606 (cited on pages 569, 571, 574).
- [1124] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. “Learning to Remember Translation History with a Continuous Cache”. In: Transactions of the Association for Computational Linguistics, 2018, pages 407–420 (cited on pages 569, 571, 574, 575).
- [1125] Eva Martínez Garcia, Carles Creus, and Cristina España-Bonet. “Context-Aware Neural Machine Translation Decoding”. In: Proceedings of the Fourth Workshop on Discourse in Machine Translation, 2019, pages 13–23 (cited on pages 569, 575).
- [1126] Lei Yu, Laurent Sartran, Wojciech Stokowiec, Wang Ling, Lingpeng Kong, Phil Blunsom, and Chris Dyer. “Better Document-Level Machine Translation with Bayes’ Rule”. In: volume 8. Transactions of the Association for Computational Linguistics, 2020, pages 346–360 (cited on pages 569, 575).
- [1127] Amane Sugiyama and Naoki Yoshinaga. “Context-aware Decoder for Neural Machine Translation using a Target-side Document-Level Language Model”. In: volume abs/2010.12827. CoRR, 2020 (cited on pages 569, 575).
- [1128] Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. “Modeling Coherence for Discourse Neural Machine Translation”. In: AAAI Conference on Artificial Intelligence, 2019, pages 7338–7345 (cited on pages 570, 576).
- [1129] Elena Voita, Rico Sennrich, and Ivan Titov. “When a Good Translation is Wrong in Context: Context-Aware Machine Translation Improves on Deixis, Ellipsis, and Lexical Cohesion”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1198–1212 (cited on pages 570, 576).

- [1130] Elena Voita, Rico Sennrich, and Ivan Titov. “Context-Aware Monolingual Repair for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 877–886 (cited on pages 570, 576).
- [1131] Lesly Miculicich Werlen and Andrei Popescu-Belis. “Validation of an Automatic Metric for the Accuracy of Pronoun Translation (APT)”. In: Proceedings of the Third Workshop on Discourse in Machine Translation, 2017, pages 17–25 (cited on page 570).
- [1132] Billy Tak-Ming Wong and Chunyu Kit. “Extending Machine Translation Evaluation Metrics with Lexical Cohesion to Document Level”. In: Conference on Empirical Methods in Natural Language Processing, 2012, pages 1060–1068 (cited on page 570).
- [1133] Zhengxian Gong, Min Zhang, and Guodong Zhou. “Document-Level Machine Translation Evaluation with Gist Consistency and Text Cohesion”. In: Proceedings of the Second Workshop on Discourse in Machine Translation, 2015, pages 33–40 (cited on page 570).
- [1134] Najeh Hajlaoui and Andrei Popescu-Belis. “Assessing the Accuracy of Discourse Connective Translations: Validation of an Automatic Metric”. In: volume 7817. Springer, 2013, pages 236–247 (cited on page 570).
- [1135] Annette Rios, Mathias Müller, and Rico Sennrich. “The Word Sense Disambiguation Test Suite at WMT18”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 588–596 (cited on page 570).
- [1136] Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. “A Large-Scale Test Set for the Evaluation of Context-Aware Pronoun Translation in Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2018, pages 61–72 (cited on page 570).
- [1137] Ruchit Rajeshkumar Agrawal, Marco Turchi, and Matteo Negri. “Contextual handling in neural machine translation: Look behind, ahead and on both sides”. In: Annual Conference of the European Association for Machine Translation, 2018, pages 11–20 (cited on page 571).
- [1138] Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. “Exploiting Cross-Sentence Context for Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2017, pages 2826–2831 (cited on pages 571, 574).

- [1139] Xin Tan, Longyin Zhang, Deyi Xiong, and Guodong Zhou. “Hierarchical Modeling of Global Context for Document-Level Neural Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1576–1585 (cited on page 571).
- [1140] Yves Scherrer, Jörg Tiedemann, and Sharid Loáiciga. “Analysing concatenation approaches to document-level NMT in two different domains”. In: Proceedings of the Fourth Workshop on Discourse in Machine Translation, 2019, pages 51–61 (cited on page 571).
- [1141] Amane Sugiyama and Naoki Yoshinaga. “Data augmentation using back-translation for context-aware neural machine translation”. In: Proceedings of the Fourth Workshop on Discourse in Machine Translation, 2019, pages 35–44 (cited on pages 571, 576).
- [1142] Shaohui Kuang and Deyi Xiong. “Fusing Recency into Neural Machine Translation with an Inter-Sentence Gate Model”. In: International Conference on Computational Linguistics, 2018, pages 607–617 (cited on pages 571, 572).
- [1143] Hayahide Yamagishi and Mamoru Komachi. “Improving Context-Aware Neural Machine Translation with Target-Side Context”. In: International Conference of the Pacific Association for Computational Linguistics, 2019 (cited on pages 571, 572).
- [1144] Alan V. Oppenheim and Ronald W. Schafer. “Discrete-time Signal Processing”. In: Pearson, 2009 (cited on page 576).
- [1145] Thomas F. Quatieri. *Discrete-Time Speech Signal Processing: Principles and Practice*. Prentice Hall PTR, 2001 (cited on page 576).
- [1146] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993 (cited on page 576).
- [1147] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, 2001 (cited on page 576).
- [1148] Li Deng Dong Yu. *Automatic Speech Recognition: a Deep Learning Approach*. Springer, 2008 (cited on page 576).
- [1149] Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. “STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3025–3036 (cited on page 576).

- [1150] Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. “Speculative Beam Search for Simultaneous Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1395–1402 (cited on page 576).
- [1151] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. “Incremental Decoding and Training Methods for Simultaneous Translation in Neural Machine Translation”. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pages 493–499 (cited on page 576).
- [1152] Kyunghyun Cho and Masha Esipova. “Can neural machine translation do simultaneous translation?” In: CoRR, 2016 (cited on page 576).
- [1153] Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. “Learning to Translate in Real-time with Neural Machine Translation”. In: Annual Conference of the European Association for Machine Translation, 2017, pages 1053–1062 (cited on page 576).
- [1154] Alvin Grissom II, He He, Jordan L. Boyd-Graber, John Morgan, and Hal Daumé III. “Don’t Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2014, pages 1342–1352 (cited on page 576).
- [1155] Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. “Simultaneous Translation Policies: From Fixed to Adaptive”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 2847–2853 (cited on page 576).
- [1156] Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. “Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation”. In: Conference on Empirical Methods in Natural Language Processing, 2019, pages 1349–1354 (cited on page 576).
- [1157] Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. “Simultaneous Translation with Flexible Policy via Restricted Imitation Learning”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 5816–5822 (cited on page 576).
- [1158] Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. “Monotonic Infinite Lookback Attention for Simultaneous Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 1313–1323 (cited on page 576).

- [1159] Yunsu Kim, Duc Thanh Tran, and Hermann Ney. “When and Why is Document-level Context Useful in Neural Machine Translation?” In: Proceedings of the Fourth Workshop on Discourse in Machine Translation, 2019, pages 24–34 (cited on page 576).
- [1160] Sébastien Jean and Kyunghyun Cho. “Context-Aware Learning for Neural Machine Translation”. In: volume abs/1903.04715. CoRR, 2019 (cited on page 576).
- [1161] Danielle Saunders, Felix Stahlberg, and Bill Byrne. “Using Context in Neural Machine Translation Training Objectives”. In: Annual Meeting of the Association for Computational Linguistics, 2020, pages 7764–7770 (cited on page 576).
- [1162] Dario Stojanovski and Alexander M. Fraser. “Improving Anaphora Resolution in Neural Machine Translation Using Curriculum Learning”. In: Annual Conference of the European Association for Machine Translation, 2019, pages 140–150 (cited on page 576).
- [1163] Tejas Gokhale, Pratyay Banerjee, Chitta Baral, and Yezhou Yang. “MUTANT: A Training Paradigm for Out-of-Distribution Generalization in Visual Question Answering”. In: Conference on Empirical Methods in Natural Language Processing, 2020, pages 878–892 (cited on page 576).
- [1164] Ruixue Tang, Chao Ma, Wei Emma Zhang, Qi Wu, and Xiaokang Yang. “Semantic Equivalent Adversarial Data Augmentation for Visual Question Answering”. In: European Conference on Computer Vision, 2020, pages 437–453 (cited on page 576).
- [1165] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J. Corso, and Jianfeng Gao. “Unified Vision-Language Pre-Training for Image Captioning and VQA”. In: AAAI Conference on Artificial Intelligence, 2020, pages 13041–13049 (cited on page 576).
- [1166] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. “VL-BERT: Pre-training of Generic Visual-Linguistic Representations”. In: International Conference on Learning Representations, 2020 (cited on page 576).
- [1167] Liangyou Li, Xin Jiang, and Qun Liu. “Pretrained Language Models for Document-Level Neural Machine Translation”. In: volume abs/1911.03110. CoRR, 2019 (cited on page 576).
- [1168] Shuhao Gu and Yang Feng. “Investigating Catastrophic Forgetting During Continual Training for Neural Machine Translation”. In: International Committee on Computational Linguistics, 2020, pages 4315–4326 (cited on page 579).

- [1169] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. “An Empirical Comparison of Simple Domain Adaptation Methods for Neural Machine Translation”. In: volume abs/1701.03214. CoRR, 2017 (cited on page 579).
- [1170] Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. “Regularized Training Objective for Continued Training for Domain Adaptation in Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 36–44 (cited on page 579).
- [1171] Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. “Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 2062–2068 (cited on page 579).
- [1172] Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. “Models and Inference for Prefix-Constrained Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2016 (cited on page 580).
- [1173] Franz Josef Och, Richard Zens, and Hermann Ney. “Efficient Search for Interactive Statistical Machine Translation”. In: *the European Chapter of the Association for Computational Linguistics*. 2003, pages 387–393 (cited on page 580).
- [1174] Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio L. Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan Miguel Vilar. “Statistical Approaches to Computer-Assisted Translation”. In: Computer Linguistics, 2009, pages 3–28 (cited on page 580).
- [1175] Miguel Domingo, Álvaro Peris, and Francisco Casacuberta. “Segment-based interactive-predictive machine translation”. In: Machine Translation, 2017, pages 163–185 (cited on page 581).
- [1176] Tsz Kin Lam, Julia Kreutzer, and Stefan Riezler. “A Reinforcement Learning Approach to Interactive-Predictive Neural Machine Translation”. In: CoRR, 2018 (cited on page 581).
- [1177] Miguel Domingo, Mercedes García-Martínez, Amando Estela, Laurent Bié, Alexandre Helle, Álvaro Peris, Francisco Casacuberta, and Manuel Herranz. “Demonstration of a Neural Machine Translation System with Online Learning for Translators”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 70–74 (cited on page 581).

- [1178] Kun Wang, Chengqing Zong, and Keh-Yih Su. “Integrating Translation Memory into Phrase-Based Machine Translation during Decoding”. In: Annual Meeting of the Association for Computational Linguistics, 2013, pages 11–21 (cited on page 582).
- [1179] Mengzhou Xia, Guoping Huang, Lemao Liu, and Shuming Shi. “Graph Based Translation Memory for Neural Machine Translation”. In: the Association for the Advance of Artificial Intelligence, 2019, pages 7297–7304 (cited on page 582).
- [1180] Chris Hokamp and Qun Liu. “Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 1535–1546 (cited on page 582).
- [1181] Matt Post and David Vilar. “Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 1314–1324 (cited on page 582).
- [1182] Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. “Guiding Neural Machine Translation Decoding with External Knowledge”. In: Annual Meeting of the Association for Computational Linguistics, 2017, pages 157–168 (cited on page 582).
- [1183] Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. “Neural Machine Translation Decoding with Terminology Constraints”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 506–512 (cited on page 582).
- [1184] Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. “Code-Switching for Enhancing NMT with Pre-Specified Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 449–459 (cited on page 583).
- [1185] Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. “Training Neural Machine Translation to Apply Terminology Constraints”. In: Annual Meeting of the Association for Computational Linguistics, 2019, pages 3063–3068 (cited on page 583).
- [1186] Tao Wang, Shaohui Kuang, Deyi Xiong, and António Branco. “Merging External Bilingual Pairs into Neural Machine Translation”. In: volume abs/1912.00567. CoRR, 2019 (cited on page 583).
- [1187] Guanhua Chen, Yun Chen, Yong Wang, and Victor O. K. Li. “Lexical-Constraint-Aware Neural Machine Translation via Data Augmentation”. In: International Joint Conference on Artificial Intelligence, 2020, pages 3587–3593 (cited on page 583).

- [1188] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. “Adaptive Neural Networks for Fast Test-Time Prediction”. In: volume abs/1702.07811. CoRR, 2017 (cited on page 584).
- [1189] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. “Multi-Scale Dense Networks for Resource Efficient Image Classification”. In: International Conference on Learning Representations, 2018 (cited on page 584).
- [1190] Jiarui Fang, Yang Yu, Chengduo Zhao, and Jie Zhou. “TurboTransformers: An Efficient GPU Serving System For Transformer Models”. In: CoRR, 2020 (cited on page 585).
- [1191] Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. “NiuTrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2012, pages 19–24 (cited on page 597).
- [1192] Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar Zaidan. “Joshua: An Open Source Toolkit for Parsing-Based Machine Translation”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 135–139 (cited on page 597).
- [1193] Gonzalo Iglesias, Adrià de Gispert, Eduardo Rodríguez Banga, and William J. Byrne. “Hierarchical Phrase-Based Translation with Weighted Finite State Transducers”. In: Annual Meeting of the Association for Computational Linguistics, 2009, pages 433–441 (cited on page 598).
- [1194] Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Türe, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. “cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 7–12 (cited on page 598).
- [1195] Daniel M. Cer, Michel Galley, Daniel Jurafsky, and Christopher D. Manning. “Phrasal: A Statistical Machine Translation Toolkit for Exploring New Model Features”. In: Annual Meeting of the Association for Computational Linguistics, 2010, pages 9–12 (cited on page 598).
- [1196] David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. “Jane: an advanced freely available hierarchical machine translation toolkit”. In: volume 26. 3. Machine Translation, 2012, pages 197–216 (cited on page 598).

- [1197] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermüller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul F. Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron C. Courville, Yann N. Dauphin, Olivier Delalleau, et al. “Theano: A Python framework for fast computation of mathematical expressions”. In: volume abs/1605.02688. CoRR, 2016 (cited on page 599).
- [1198] Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. “Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies”. In: Annual Meeting of the Association for Computational Linguistics, 2016, pages 1217–1222 (cited on page 599).
- [1199] Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huan-Bo Luan, and Yang Liu. “THUMT: An Open Source Toolkit for Neural Machine Translation”. In: volume abs/1706.06415. CoRR, 2017 (cited on page 599).
- [1200] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. “Marian: Fast Neural Machine Translation in C++”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 116–121 (cited on page 600).
- [1201] Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. “Sockeye: A Toolkit for Neural Machine Translation”. In: volume abs/1712.05690. CoRR, 2017 (cited on page 600).
- [1202] Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. “CytonMT: an Efficient Neural Machine Translation Open-source Toolkit Implemented in C++”. In: Annual Meeting of the Association for Computational Linguistics, 2018, pages 133–138 (cited on page 600).
- [1203] Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. “OpenSeq2Seq: extensible toolkit for distributed and mixed precision training of sequence-to-sequence models”. In: volume abs/1805.10387. CoRR, 2018 (cited on page 600).
- [1204] Ozan Caglayan, Mercedes García-Martínez, Adrien Bardet, Walid Aransa, Fethi Bougares, and Loïc Barrault. “NMTPY: A Flexible Toolkit for Advanced Neural Machine Translation Systems”. In: volume 109. The Prague Bulletin of Mathematical Linguistics, 2017, pages 15–28 (cited on page 600).



# Index

- n*-gram Precision, 102  
*n* 元语法单元, 47
- Action-value Function, 425  
Active Learning, 434  
Actor-critic, 425  
AdaGrad, 296  
Adam, 297  
Additive Smoothing, 50  
Adequacy, 98  
Adversarial Attack, 417  
Adversarial Examples, 416  
Adversarial Training, 416  
Adversarial-NMT, 423  
Algorithm, 27  
Ambiguity, 89  
Anchor, 536  
Anneal, 434  
Annotated Data, 72  
Artificial Neural Networks, 259  
Asymmetric Word Alignment, 146
- Asynchronous Update, 298  
Attention Mechanism, 322  
Attention Weight, 344  
Automated Machine Learning, 503  
Automatic Differentiation, 293  
Automatic Post-editing, 123  
Automatic Speech Recognition, 552  
Autoregressive Decoding, 453  
Autoregressive Model, 356  
Autoregressive Translation, 453  
Average Pooling, 365
- Back Translation, 514  
Backward Mode, 293  
Backward Propagation, 287  
Batch Gradient Descent, 290  
Batch Inference, 451  
Batch Normalization, 300  
Bayes' Rule, 40  
Beam Pruning, 63  
Beam Search, 63

- Bellman Equation, 426  
BERT, 521  
Bias, 44  
Bidirectional Inference, 442  
Bilingual Dictionary Induction, 534  
Binarization, 242  
Binarized Neural Networks, 453  
Bits Per Second, 551  
Breadth-first Search, 59  
Brevity Penalty, 103  
Broadcast Mechanism, 283  
Bucket, 451  
Byte Pair Encoding, 69  
  
Calculus, 287  
Catastrophic Forgetting, 437  
Chain Rule, 39  
Chart, 249  
Chart Cell, 249  
Child Model, 532  
Chomsky Normal Form, 220  
Clarity, 98  
Class Imbalance Problem, 544  
Classifier, 83  
Co-adaptation, 414  
Coherence, 98  
Composition, 250  
Compositional Translation, 180  
Computation Graph, 287  
Computational Linguistics, 32  
Concept, 170  
Conditional Probability, 37  
Conditional Random Fields, 75  
Confusion Network, 465  
Connectionism, 260  
Connectionist Temporal Classification, 555  
Constraint-based Translation, 580  
Context Vector, 343  
  
Context-free Grammar, 87  
Continual Learning, 437  
Continuous Dynamic Model, 474  
Convex Function, 173  
Convolution Kernel, 363  
Convolutional Neural Network, 361  
Correlation, 113  
Cost Function, 289  
Coverage Model, 201  
Cross-entropy, 42  
Cross-entropy Difference, 432  
Cube Pruning, 225  
Curriculum Learning, 435  
curriculum learning, 422  
Curriculum Schedule, 435  
  
Data Augmentation, 514  
Data-driven, 18  
Decay, 296  
Decoding, 55  
Deep Learning, 259  
Deficiency, 171  
Denoising, 411  
Denoising Autoencoder, 516  
Dependency Parsing, 85  
Depth-first Search, 59  
Depthwise Convolution, 376  
Depthwise Separable Convolution, 376  
Derivation, 88  
Difficulty Criteria, 435  
Direct Assessment, 99  
Directed Hyper-graph, 247  
Disambiguation, 89  
Discriminative Model, 93  
Disfluency Detection, 553  
Distributed Representation, 111  
Domain Adaptation, 432  
DREEM, 111

- Dropout, 399  
Dual Supervised Learning, 526  
Dual Unsupervised Learning, 526  
Dynamic convolution, 376  
Dynamic Linear Combination of Layers, 484  
Dynamic Programming Encoding, DPE, 410  
Element-wise Addition, 266  
Element-wise Product, 268  
Emission Probability, 77  
Empty Alignment, 147  
Encoder-Decoder, 30  
Encoder-Decoder Attention Sub-layer, 387  
Encoder-Decoder Paradigm, 328  
End-to-End Learning, 262  
End-to-End Speech Translation, 555  
Ensemble Learning, 415  
Entropy, 17  
Estimate, 36  
Estimation, 36  
Euclidean Norm, 270  
Event, 35  
Evil Feature, 197  
Exact Model, 104  
Expectation Maximization, 156  
Expected Count, 156  
Explainable Machine Learning, 317  
Exposure Bias, 420  
Expression Swell, 293  
Feature, 75  
Feature Decay Algorithms, 433  
Feature Engineering, 76  
Feed-Forward Sub-layer, 387  
Fertility, 166  
Fidelity, 96  
Filter-bank, 552  
Fine-tuning, 520  
First Moment Estimation, 351  
Fluency, 96  
FNNLM, 308  
Forward Propagation, 286  
Frame Length, 551  
Frame Shift, 551  
Framing, 551  
Frobenius Norm, 270  
Frobenius 范数, 270  
Frontier Set, 238  
Full Parsing, 85  
Full Search, 356  
Future Mask, 394  
Gated Linear Units, 370  
Gated Recurrent Unit, 338  
Generalization, 412  
Generation, 335  
Generative Adversarial Networks, 418  
Generative Model, 79  
Glue Rule, 216  
Good-Turing Estimate, 51  
GPT, 521  
GPU, 20  
Gradient Clipping, 299  
Gradient Explosion, 299  
Gradient Vanishing, 299  
Gradual Warmup, 352  
Grammer, 27  
Graphics Processing Unit, 20  
Greedy Search, 62  
Grid Search, 199  
Hamming, 551  
Harmonic-mean, 106  
Heuristic Function, 62  
Heuristic Search, 62  
Hidden Markov Model, 75

- Hierarchical Phrase-based Grammar, 215  
 Hierarchical Phrase-based Model, 214  
 High-resource Language, 513  
 Histogram Pruning, 63  
 Human Translation Error Rate, 120  
 Hyper-edge, 247  
 Hypothesis Recombination, 203  
 Hypothesis Selection, 463  
 Ill-posed Problem, 411  
 Image Captioning, 559  
 Image-to-Image Translation, 565  
 Implicit Bridging, 534  
 Incremental Learning, 437  
 Inductive Bias, 470  
 Inference, 54  
 Informativeness, 98  
 Intelligibility, 98  
 Interactive Machine Translation, 578  
 Interlingua-based Translation, 26  
 Inverse Problem, 410  
 Iterative Back Translation, 515  
 Iterative Refinement, 538  
 Joint Probability, 37  
 Joint-BPE, 408  
 KL Distance, 41  
 KL 距离, 41  
 Kullback-Leibler 距离, 41  
 Label, 83  
 Label Bias, 81  
 Label Smoothing, 399  
 Language, 89  
 Language Model, 46  
 Language Modeling, 46  
 Law of Total Probability, 39  
 Layer Normalization, 300  
 Learning Difficulty, 428  
 Learning Rate, 290  
 Left-hand Side, 87  
 Left-most Derivation, 89  
 Left-to-Right Generation, 56  
 Lexical Analysis, 69  
 Lexical Chain, 567  
 Lexical Translation Probability, 192  
 Lexicalized Norm Form, 253  
 Lexically Constrained Translation, 580  
 Lifelong Learning, 437  
 Lightweight Convolution, 376  
 Line Search, 198  
 Linear Mapping, 268  
 Linear Transformation, 268  
 Linearization, 230  
 Locally Connected, 362  
 Log-linear Model, 187  
 Logical Deficiency, 175  
 Long Short-term Memory, 336  
 Loss Function, 288  
 Low-resource Language, 513  
 LSH Attention, 480  
 Machine Translation, 13  
 Marginal Probability, 38  
 Mask, 394  
 MASS, 523  
 Matrix, 265  
 Max Pooling, 365  
 Maximum Entropy, 75  
 Mel 频率倒谱系数, 552  
 Meteor Normalizer, 107  
 Mini-batch Gradient Descent, 291  
 Mini-batch Training, 398  
 Minimal Rules, 239  
 Minimum Error Rate Training, 197  
 Minimum Risk Training, 423

- MLM, 522  
Modality, 550  
Model Compression, 428  
Model Parameters, 287  
Model Score, 59  
Model Training, 197  
Momentum, 295  
Multi-branch, 476  
Multi-head Attention, 393  
Multi-hop Attention, 370  
Multi-lingual Single Model-based Method, 532  
Multi-model Machine Translation, 550  
Multi-stage Inference, 442  
Multi-step Attention, 370  
Multimodality Problem, 456  
Multitask Learning, 502
- Named Entity, 74  
Named Entity Recognition, 74  
Natural Language Processing, 32  
Nesterov Accelerated Gradient, 375  
Nesterov 加速梯度下降法, 375  
Neural Architecture Search, 503  
Neural Language Model, 307  
Neural Machine Translation, 319  
Neural Networks, 259  
Noise Channel Model, 144  
Non-autoregressive Model, 356  
Non-Autoregressive Translation, 454  
Non-terminal, 85  
Norm, 269  
Numerical Differentiation, 292
- Objective Function, 288  
Offline Speech Translation, 550  
One-hot 编码, 312  
Open Vocabulary, 406  
Optimal Stopping Criteria, 64
- Out-of-vocabulary Word, 49  
Over Translation, 444  
Overfitting, 301
- Padding Mask, 394  
Parameter, 48  
Parameter Estimation, 43  
Parameter Server, 299  
Paraphrase Matcher, 107  
Paraphrasing, 518  
Parent Model, 532  
Parsing, 68  
Perplexity, 54  
Phrasal Segmentation, 183  
Phrase Extraction, 189  
Phrase Pairs, 184  
Phrase Structure Parsing, 85  
Phrase Table, 192  
Physical Deficiency, 175  
Piecewise Constant Decay, 352  
Pivot Language, 529  
Pointwise Convolution, 376  
Policy Gradient, 424  
Porter Stem Model, 105  
Position Embedding, 368  
Position-independent word Error Rate, 101  
Post-editing, 578  
Post-norm, 396  
Post-processing, 69  
Pre-emphasis, 551  
Pre-norm, 396  
Pre-processing, 69  
Pre-terminal, 85  
Pre-training, 520  
Precision, 103  
Prediction, 54  
Probabilistic Context-free Grammar, 90  
Probabilistic Graphical Model, 76

- Probability, 36  
Procrustes Analysis, 535  
Procrustes Problem, 536  
Production Rule, 87  
Pruning, 61  
  
Quality Estimation, 116  
Quality Evaluation of Translation, 95  
  
Random Variable, 36  
Rank, 281  
Recall, 103  
Receptive Field, 367  
Reconstruction Loss, 528  
Recurrent Neural Network, 311  
Recursive Auto-encoder Embedding, 112  
Regularization, 301  
Reinforcement Learning, 423  
Relative Entropy, 42  
Relative Positional Representation, 471  
Relative Ranking, 99  
Reordering, 193  
Representation Learning, 262  
Reranking, 442  
Residual Connection, 370  
Residual Networks, 300  
Reverse Mode, 293  
Reward Shaping, 427  
Right-hand Side, 87  
RMSProp, 297  
RNN Cell, 311  
RNNLM, 311  
Robustness, 416  
Round-off Error, 292  
  
Scalar, 265  
Scalar Multiplication, 267  
Scaled Dot-product Attention, 391  
Scheduled Sampling, 421  
  
Search Problem, 55  
Second Moment Estimation, 351  
Segmentation, 68  
Selection, 432  
Self-attention Mechanism, 312  
Self-Attention Sub-layer, 387  
Self-information, 41  
Self-paced Learning, 436  
Self-supervised, 559  
Semi-ring Parsing, 248  
Sentence, 89  
Sequence Generation, 55  
Sequence Labeling, 74  
Sequence-level Knowledge Distillation, 429  
Significance Level, 115  
Simultaneous Translation, 550  
Singular Value Decomposition, 536  
Skip Connection, 300  
Smoothing, 49  
Solver, 474  
Source Domain, 432  
Source Language, 13  
Span, 221  
Speech Translation, 550  
Speech-to-Speech Translation, 550  
Speech-to-Text Translation, 550  
Spiritual Deficiency, 175  
Stability-Plasticity, 437  
Statistical Hypothesis Testing, 114  
Statistical Inference, 466  
Statistical Language Modeling, 35  
Stochastic Gradient Descent, 291  
Stride Convolution, 480  
String-based Decoding, 250  
Structural Position Representations, 473  
Student Model, 429  
Sub-word, 407

- Supervised Training, 288  
Support Vector Machine, 75  
Symbolic Differentiation, 292  
Symbolicism, 261  
Symmetrization, 174  
Synchronous Context-free Grammar, 214  
Synchronous Tree-substitution Grammar, 232  
Synchronous Update, 298  
Syntax, 85  
System Bias, 176  
System Combination, 462  
Target Domain, 432  
Target Language, 13  
Teacher Model, 429  
Technical Deficiency, 175  
Tensor, 281  
Terminal, 85  
Text-to-Image Translation, 565  
The Gradient Descent Method, 289  
The Gradient-based Method, 290  
The Lagrange Multiplier Method, 154  
The Reversible Residual Network, 480  
Threshold Pruning, 203  
Time-constrained Search, 445  
Token, 69  
Top-Down Parsing, 221  
Training, 48  
Training Data Set, 288  
Transfer Learning, 531  
Transfer-based Translation, 24  
Transformer, 385  
Transition Probability, 77  
Translation Candidate, 132  
Translation Error Rate, 101  
Translation Hypothesis, 202  
Translation Memory, 579  
Transpose, 266  
Tree Fragment, 231  
Tree-based Decoding, 250  
Tree-to-String Translation Rule, 230  
Tree-to-Tree Translation Rule, 231  
Treebank, 92  
Truncation Error, 292  
Tunable Weight Vector, 107  
Tuning Set, 198  
Under Translation, 444  
Uniform-cost Search, 60  
Uninformed Search, 60  
Unsupervised Machine Translation, 534  
Update Rule, 290  
Variational Autoencoders, 537  
Variational Methods, 467  
Vector, 265  
Vectorization, 283  
Viterbi Algorithm, 79  
Warmup, 398  
Waveform, 551  
Weight, 270  
Weight Sharing, 362  
Weight Tuning, 197  
Well-posed, 411  
Windowing, 551  
WN Synonymy Model, 105  
Word Alignment, 138  
Word Alignment Link, 138  
Word Embedding, 111  
Word Error Rate, 101  
Word Feature, 76  
Word Lattice, 465  
Word-level Knowledge Distillation, 429  
Word-to-Word, 104  
“同义词” 匹配模型, 105

- “波特词干”匹配模型, 105  
“绝对”匹配模型, 104  
  
一致代价搜索, 60  
一阶矩估计, 351  
上下文向量, 343  
上下文无关文法, 87  
不适当问题, 411  
与位置无关的单词错误率, 101  
主动学习, 434  
乔姆斯基范式, 220  
事件, 35  
二值网络, 453  
二叉化, 242  
二阶矩估计, 351  
交互式机器翻译, 578  
交叉熵, 42  
交叉熵差, 432  
产出率, 166  
产生式规则, 87  
人工神经网络, 259  
人工神经网络方法, 48  
人工译后编辑距离, 120  
代价函数, 289  
估计, 36  
估计值, 36  
位置编码, 368  
低资源机器翻译, 513  
低资源语言, 513  
依存分析, 85  
信息性, 98  
  
假设选择, 463  
假设重组, 203  
偏置, 44  
健壮性, 416  
充分性, 98  
全搜索, 356  
  
全概率公式, 39  
准确率, 103  
凸函数, 173  
函数塑形, 427  
分布式表示, 111  
分布式表示评价度量, 111  
分帧, 551  
分段常数衰减, 352  
分类器, 83  
分词, 68  
判别式模型, 93  
前向传播, 286  
前标准化, 396  
前馈神经网络子层, 387  
前馈神经网络语言模型, 308  
剪枝, 61  
加法平滑, 50  
加窗, 551  
动作价值函数, 425  
动态卷积, 376  
动态线性聚合网络, 484  
动态规划编码, 410  
半环分析, 248  
单元, 69  
单词, 68  
单词到单词, 104  
单词错误率, 101  
卷积核, 363  
卷积神经网络, 361  
参数, 48  
参数估计, 43  
参数更新的规则, 290  
参数服务器, 299  
双向推断, 442  
双向编码器表示, 521  
双字节编码, 69  
双字节联合编码, 408

- 反向传播, 287  
反向模式, 293  
反问题, 410  
发射概率, 77  
变分方法, 466  
变分自编码器, 537  
古德-图灵估计, 51  
句子, 89  
句子的表示, 315  
句子表示模型, 315  
句法, 85  
句法分析, 68  
句长补全掩码, 394  
召回率, 103  
可理解度, 98  
可解释机器学习, 317  
可调权值向量, 107  
可逆残差网络结构, 480  
右部, 87  
同步上下文无关文法, 214  
同步更新, 298  
同步树替换文法, 232  
后处理, 69  
后标准化, 395  
向量, 265  
向量化, 283  
启发式函数, 61  
启发式搜索, 62  
命名实体, 74  
命名实体识别, 74  
噪声信道模型, 144  
回译, 514  
困惑度, 54  
图像到图像的翻译, 565  
图片描述生成, 559  
基于中间语言的机器翻译, 26  
基于串的解码, 250  
基于单词的知识蒸馏, 429  
基于句法的特征, 246  
基于层次短语的文法, 215  
基于层次短语的模型, 214  
基于序列的知识蒸馏, 429  
基于树的解码, 250  
基于梯度的方法, 290  
基于短语的特征, 246  
基于约束的翻译, 580  
基于结构化位置编码, 473  
基于转换规则的机器翻译, 24  
基于连续动态系统, 474  
基于频次的方法, 47  
增量式学习, 437  
多任务学习, 502  
多分支, 476  
多头注意力机制, 393  
多峰问题, 456  
多模态机器翻译, 550  
多语言单模型方法, 532  
多跳注意力机制, 370  
多阶段推断, 442  
奇异值分解, 536  
子模型, 532  
子词, 407  
学习率, 290  
学习难度, 428  
学生模型, 429  
完全分析, 85  
实时语音翻译, 550  
宽度优先搜索, 59  
富资源语言, 513  
对抗攻击, 417  
对抗样本, 416  
对抗神经机器翻译, 423  
对抗训练, 416  
对数线性模型, 187

- 对称化, 174  
小批量梯度下降, 291  
小批量训练, 398  
局部哈希敏感注意力机制, 480  
局部连接, 362  
层标准化, 300  
左部, 87  
帧移, 551  
帧长, 551  
平均池化, 365  
平滑, 49  
广播机制, 283  
序列化, 230  
序列标注, 74  
序列生成, 55  
开放词表, 406  
异步更新, 298  
张量, 281  
强化学习, 423  
归纳偏置, 470  
循环单元, 311  
循环神经网络, 311  
循环神经网络语言模型, 311  
微调, 520  
忠诚度, 96  
  
感受野, 367  
成分分析, 85  
截断误差, 292  
批量推断, 451  
批量标准化, 300  
批量梯度下降, 290  
技术缺陷, 175  
拉格朗日乘数法, 154  
持续学习, 437  
按元素乘积, 268  
按元素加法, 266  
损失函数, 288  
  
推导, 88  
推断, 54  
掩码, 394  
掩码端到端预训练, 523  
掩码语言模型, 522  
搜索问题, 55  
支持向量机, 75  
教师模型, 429  
数乘, 267  
数值微分, 292  
数据增强, 514  
数据并行, 298  
数据选择, 432  
数据驱动, 18  
文本到图像的翻译, 565  
文本规范器, 107  
无信息搜索, 60  
无监督机器翻译, 534  
时间受限的搜索, 445  
映射锚点, 536  
显著性水平, 115  
普氏分析, 535  
普鲁克问题, 536  
曝光偏置, 420  
最佳停止条件, 63  
最大池化, 365  
最大熵, 75  
最小规则, 239  
最小错误率训练, 197  
最小风险训练, 423  
最左优先推导, 89  
有向超图, 247  
有害特征, 197  
有指导的训练, 288  
有监督的训练, 288  
期望最大化, 156  
期望频次, 156

- 未来信息掩码, 394  
未登录词, 49  
机器翻译, 13  
权值共享, 362  
权重, 270  
权重调优, 197  
束剪枝, 63  
束搜索, 63  
条件概率, 37  
条件随机场, 75  
枢轴语言, 529  
标注偏置, 81  
标注数据, 72  
标签, 83  
标签平滑, 399  
标量, 265  
树到串翻译规则, 230  
树到树翻译规则, 231  
树库, 92  
树片段, 231  
格搜索, 199  
桶, 451  
梯度下降方法, 289  
梯度消失, 299  
梯度爆炸, 299  
梯度裁剪, 299  
概念, 170  
概念单元, 170  
概率, 36  
概率上下文无关文法, 90  
概率分布函数, 37  
概率图模型, 76  
概率密度函数, 37  
模型压缩, 428  
模型参数, 287  
模型并行, 354  
模型得分, 59  
模型训练, 197  
模态, 550  
欠翻译, 444  
欧几里得范数, 270  
正则化, 301  
歧义, 89  
残差网络, 300  
残差连接, 370  
比特率, 551  
求解器, 474  
汉明窗, 551  
池化层, 365  
泛化, 412  
注意力机制, 322  
注意力权重, 344  
流畅度, 96  
消歧, 89  
深度优先搜索, 59  
深度可分离卷积, 376  
深度学习, 259  
混淆网络, 465  
清晰度, 98  
源语言, 13  
源领域, 432  
滤波器, 363  
滤波器组, 552  
演员-评论家, 425  
灾难性遗忘, 437  
熵, 17  
父模型, 532  
物理缺陷, 175  
特征, 75  
特征工程, 76  
特征衰减算法, 433  
独热编码, 312  
生成, 335  
生成对抗网络, 418

- 生成式模型, 79  
生成式预训练, 521  
目标函数, 288  
目标语言, 13  
目标领域, 432  
直接评估, 99  
直方图剪枝, 63  
相互适应, 414  
相关性, 113  
相对位置编码, 471  
相对排序, 99  
相对熵, 42  
矩阵, 265  
短句惩罚因子, 103  
短语, 181  
短语切分, 183  
短语对, 184  
短语抽取, 189  
短语结构分析, 85  
短语表, 192  
神经机器翻译, 319  
神经架构搜索, 503  
神经网络, 259  
神经语言模型, 307  
离线语音翻译, 550  
稳定性- 可塑性, 437  
空对齐, 147  
立方剪枝, 225  
端到端学习, 262  
端到端的语音翻译模型, 555  
符号主义, 261  
符号微分, 292  
策略梯度, 424  
算子, 287  
算法, 27  
类别不均衡问题, 544  
系统偏置, 176  
系统融合, 462  
繁衍率, 166  
线性化, 230  
线性变换, 268  
线性映射, 268  
线搜索, 198  
组合, 250  
组合性翻译, 180  
终结符, 85  
统计假设检验, 114  
统计推断, 466  
统计语言建模, 35  
维特比算法, 79  
编码-解码注意力子层, 387  
编码器-解码器, 30  
编码器-解码器框架, 328  
编码器-解码器模型, 328  
缩放的点乘注意力, 391  
缺陷, 171  
翻译候选, 132  
翻译假设, 202  
翻译记忆, 579  
翻译错误率, 101  
联合概率, 37  
胶水规则, 216  
自下而上的分析, 221  
自信息, 41  
自动后编辑, 123  
自动微分, 293  
自动机器学习, 503  
自动评价, 101  
自动语音识别, 552  
自回归模型, 356  
自回归翻译, 453  
自回归解码, 453  
自左向右生成, 56  
自步学习, 436

- 自注意力子层, 387  
自注意力机制, 312  
自然语言处理, 32  
自监督, 559  
舍入误差, 292  
范数, 269  
表格, 249  
表格单元, 249  
表示学习, 262  
表达式膨胀, 293  
衰减, 296  
覆盖度模型, 201  
解码, 55  
计算图, 287  
计算语言学, 32  
训练, 48  
训练数据集合, 288  
记忆更新, 337  
词典归纳, 534  
词对齐, 138  
词对齐连接, 138  
词嵌入, 111  
词格, 465  
词汇化标准形式, 253  
词汇化翻译概率, 192  
词汇约束翻译, 580  
词汇链, 567  
词法分析, 69  
词特征, 76  
译后编辑, 578  
译文质量评价, 95  
语法, 27  
语言, 89  
语言建模, 46  
语言模型, 46  
语音到文本翻译, 550  
语音到语音翻译, 550  
语音翻译, 550  
课程学习, 422, 435  
课程规划, 435  
调优集合, 198  
调和均值, 106  
调序, 193  
调度采样, 421  
贝叶斯法则, 40  
贝尔曼方程, 426  
质量评估, 116  
贪婪搜索, 62  
超边, 247  
跨度, 221  
跨步卷积, 480  
跳接, 300  
转移概率, 77  
转置, 266  
转述, 518  
轻量卷积, 376  
输出, 338  
边缘概率, 38  
边缘集合, 238  
迁移学习, 531  
过拟合, 301  
过翻译, 444  
连接主义, 260  
连接时序分类, 555  
连贯性, 98  
迭代优化, 538  
迭代式回译, 515  
退火, 434  
适定的, 411  
逐渐预热, 352  
逐点卷积, 376  
逐通道卷积, 376  
递归自动编码, 112  
逻辑缺陷, 175

遗忘, 336  
释义匹配器, 107  
重排序, 442  
重构损失, 528  
链式法则, 39  
长短时记忆, 336  
门循环单元, 338  
阈值剪枝, 203  
阶, 281  
降噪, 411  
降噪自编码器, 516  
随机事件, 36  
随机变量, 36  
随机梯度下降, 291  
隐式桥接, 534  
隐马尔可夫模型, 75  
难度评估准则, 435  
集成学习, 415  
非对称的词对齐, 146  
非终结符, 85  
非自回归模型, 356  
非自回归翻译, 454  
顺滑, 553  
预加重, 551  
预处理, 69  
预测, 54  
预热, 398  
预终结符, 85  
预训练, 520  
领域适应, 432