第 9 章　自然语言处理

本章学习如何实现人与计算机之间用自然语言进行有效通信的各种理论和方法。介绍如何用自然语言描述丰富知识。

3.1 自然语言处理基础

在本章中，你将学到与自然语言处理相关的基础知识。

本章的要点包括：

- 自然语言处理基础概念
- 自然语言处理的发展与应用
- 自然语言处理常用术语以及扩展介绍

3.1.1 自然语言处理基本概念

自然语言处理（Natural Language Processing，NLP）是计算机科学领域以及人工智能领域的一个重要的研究方向，它研究用计算机来处理、理解以及运用人类语言（如中文、英文等），达到人与计算机之间进行有效通讯。所谓“自然”乃是寓意自然进化形成，是为了区分一些人造语言，类似 C++、Java 等人为设计的语言。在人类社会中，语言扮演着重要的角色，语言是人类区别于其他动物的根本标志，没有语言，人类的思维无从谈起，沟通交流更是无源之水。在一般情况下，用户可能不熟悉机器语言，所以自然语言处理技术可以帮助这样的用户使用自然语言和机器交流。从建模的角度看，为了方便计算机处理，自然语言可以被定义为一组规则或符号的集合，我们组合集合中的符号来传递各种信息。自然语言处理研究表示语言能力、语言应用的模型，通过建立计算机框架来实现这样的语言模型，并且不断完善这样的语言模型，还需要根据该语言模型来设计各种实用的系统，并且探讨这些实用技术的评测技术。这一定义有点宽泛，但是语言本身就是人类最为复杂的概念之一。这些年，NLP 研究取得了长足的进步，逐渐发展成为一门独立的学科，从自然语言的角度出发，NLP 基本可以分为两个部分：自然语言处理以及自然语言生成，演化为理解和生成文本的任务，如图 3.1 所示。

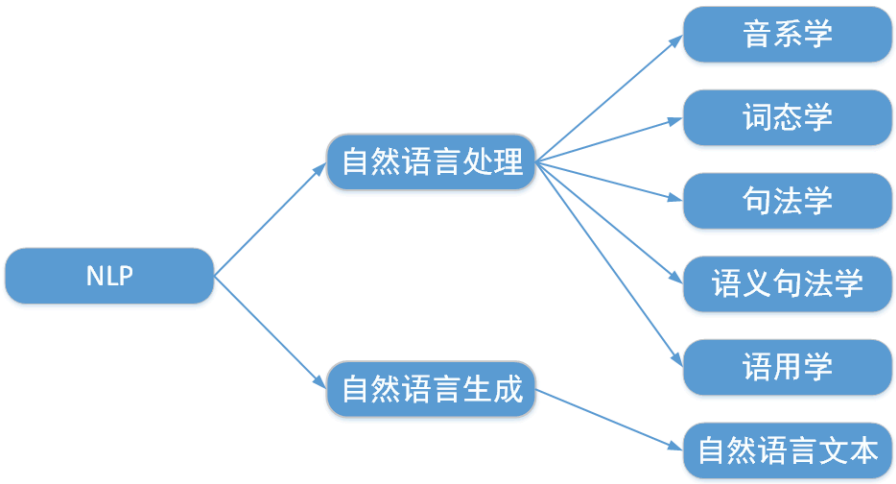


图 3.1 NLP 的基本分类

自然语言的理解是个综合的系统工程，它又包含了很多细分学科，有代表声音的音系学，代表构词法的词态学，代表语句结构的句法学，代表理解的语义句法学和语用学。

- 音系学：指代语言中发音的系统化组织。
- 词态学：研究单词构成以及相互之间的关系。

●句法学：给定文本的哪部分是语法正确的。

●语义学：给定文本的含义是什么？

●语用学：文本的目的是什么？

语言理解涉及语言、语境和各种语言形式的学科。而自然语言生成（Natural Language Generation，NLG）恰恰相反，从结构化数据中以读取的方式自动生成文本。该过程主要包含三个阶段：文本规划（完成结构化数据中的基础内容规划）、语句规划（从结构化数据中组合语句来表达信息流）、实现（产生语法通顺的语句来表达文本）。

3.1.2 自然语言处理的发展历程

NLP 的发展大致经历了 3 个阶段：1956 年以前的萌芽期，1980 年～1999 年的快速发展期和 21 世纪的突飞猛进期。

萌芽期（1956 年以前）

早期的自然语言处理具有鲜明的经验主义色彩。如 1913 年马尔科夫提出马尔可夫随机过程与马尔可夫模型的基础就是“手工查频”，具体说就是统计了《欧根·奥涅金》长诗中元音与辅音出现频度；1948 年香农把离散马尔可夫的概率模型应用于语言的自动机，同时采用手工方法统计英语字母的频率。然而这种经验主义到了乔姆斯基时期出现了转变。1956 年乔姆斯基借鉴香农的工作，把有限状态机作为刻画语法的工具，建立了自然语言的有限状态模型，具体来说就是用“代数”和“集合”将语言转化为符号序列，建立了一大堆有关语法的数学模型。这些工作非常伟大，为自然语言和形式语言找到了一种统一的数学描述理论，一个叫做“形式语言理论”的新领域诞生了。但乔姆斯基否定了有限状态模型在自然语言中的适用性，然后主张采用有限的、严格的规则去描述无限的语言现象，提出了风靡一时的转换生成语法。这一时期，虽然诸如贝叶斯方法、隐马尔可夫、最大熵、支持向量机等经典理论和算法也有提出，但自然语言处理领域的主流仍然是基于规则的理性主义方法。

快速发展期（1980 年～1999 年）

这种情况一直持续到 20 世纪 80 年代初期才发生变化，很多学者开始反思有限状态模型以及经验主义方法的合理性。20 世纪 80 年代初，话语分析（Discourse Analysis）也取得了重大进展。之后，由于自然语言处理研究者对于过去的研究进行了反思，有限状态模型和经验主义研究方法也开始复苏。90 年代后，基于统计的自然语言处理开始大放异彩。首先是在机器翻译领域取得了突破，因为引入了许多基于语料库的方法。1990 年在芬兰赫尔辛基举办的第 13 届国际计算语言学会议确定的主题是“处理大规模真实文本的理论、方法与工具”，研究的重心开始转向大规模真实文本了，传统的基于规则的自然语言处理显然力不从心了。学者们认为，大规模语料至少是对基于规则方法有效的补充。在 1994 年～1999 年间，经验主义空前繁荣，如句法剖析、词类标注、参照消解、话语处理的算法几乎把“概率”与“数据”作为标准方法，成为自然语言处理的主流。

20 世纪 90 年代中期，有两件事从根本上促进了自然语言处理研究的复苏与发展。一件事是 20 世纪 90 年代中期以来，计算机的运行速度和存储量大幅增加，为自然语言处理改善了物质基础，使得语音和语言处理的商品化开发成为可能；另一件事是 1994 年 Internet 商业化和同期网络技术的发展使得基于自然语言的信息检索和信息抽取的需求变得更加突出。这样，自然语言处理的社会需求更加迫切，自然语言处理的应用面也更加宽广，自然语言处理不再局限于机器翻译、语音控制等早期研究领域了。

从 20 世纪 90 年代末到 21 世纪初，人们逐渐认识到，仅用基于规则或统计的方法是无法成功进行自然语言处理的。基于统计、基于实例和基于规则的语料库技术在这一时期开始蓬勃发展，各种处理技术开始融合，自然语言处理的研究再次繁荣。

突飞猛进期（2000 年至今）

进入 21 世纪以后，自然语言处理又有了突飞猛进的变化。2006 年，以 Hinton 为首的几位科学家历经近 20 年的努力，终于成功设计出第一个多层神经网络算法——深度学习。这是一种将原始数据通过一些简单但是非线性的模型转变成更高层次、更加抽象表达的特征学习方法，一定程度上解决了人类处理“抽象概念”这个亘古难题。目前，深度学习在机器翻译、问答系统等多个自然语言处理任务中均取得了不错的成果，

相关技术也被成功应用于商业化平台中。

未来，深度学习作为人工智能皇冠上的明珠，将会在自然语言处理领域发挥着越来越重要的作用。

3.1.3 自然语言处理相关知识的构成

一、 基本术语

为了帮助读者更好地学习 NLP，这里会一一介绍 NLP 领域的一些基础专业词汇。

（1）分词（segment）

词是最小的能够独立活动的有意义的语言成分，英文单词之间是以空格作为自然分界符的，而汉语是以字为基本的书写单位，词语之间没有明显的区分标记，因此，中文词语分析是中文分词的基础与关键。中文和英文都存在分词的需求，不过相较而言，英文单词本来就有空格进行分割，所以处理起来相对方便。但是，由于中文是没有分隔符的，所以分词的问题就比较重要。分词常用的手段是基于字典的最长串匹配，据说可以解决 85%的问题，但是歧义分词很难。举个例子，“美国会通过对台售武法案”，我们既可以切分为“美国/会/通过对台售武法案”，又可以切分成“美/国会/通过对台售武法案”。

（2）词性标注（part-of-speech tagging）

基于机器学习的方法里，往往需要对词的词性进行标注。词性一般是指动词、名词、形容词等。标注的目的是表征词的一种隐藏状态，隐藏状态构成的转移就构成了状态转移序列。例如：我/r 爱/v 北京/ns 天安门/ns。其中，ns 代表名词，v 代表动词，ns、v 都是标注，以此类推。

（3）命名实体识别（NER，Named Entity Recognition）

命名实体是指从文本中识别具有特定类别的实体（通常是名词），例如人名、地名、机构名、专有名词等。

（4）句法分析（syntax parsing）

句法分析往往是一种基于规则的专家系统。当然也不是说它不能用统计学的方法进行构建，不过最初的时候，还是利用语言学专家的知识来构建的。句法分析的目的是解析句子中各个成分的依赖关系。所以，往往最终生成的结果是一棵句法分析树。句法分析可以解决传统词袋模型不考虑上下文的问题。比如，“小李是小杨的班长”和“小杨是小李的班长”，这两句话，用词袋模型是完全相同的，但是句法分析可以分析出其中的主从关系，真正理清句子的关系。

（5）指代消解（anaphora resolution）

中文中代词出现的频率很高，它的作用的是用来表征前文出现过的人名、地名等。例如，清华大学坐落于北京，这家大学是目前中国最好的大学之一。在这句话中，其实“清华大学”这个词出现了两次，“这家大学”指代的就是清华大学。但是出于中文的习惯，我们不会把“清华大学”再重复一遍。

（6）情感识别（emotion recognition）

所谓情感识别，本质上是分类问题，经常被应用在舆情分析等领域。情感一般可以分为两类，即正面、负面，也可以是三类，在前面的基础上，再加上中性类别。一般来说，在电商企业，情感识别可以分析商品评价的好坏，以此作为下一个环节的评判依据。通常可以基于词袋模型+分类器，或者现在流行的词向量模型+RNN。经过测试发现，后者比前者准确率略有提升。

（7）纠错（correction）

自动纠错在搜索技术以及输入法中利用得很多。由于用户的输入出错的可能性比较大，出错的场景也比较多。所以，我们需要一个纠错系统。具体做法有很多，可以基于 N-Gram 进行纠错，也可以通过字典树、有限状态机等方法进行纠错。

（8）问答系统（QA system）

这是一种类似机器人的人工智能系统。比较著名的有：苹果 Siri、IBM Watson、微软小冰等。问答系统往往需要语音识别、合成，自然语言理解、知识图谱等多项技术的配合才会实现得比较好。

二、知识结构

作为一门综合学科，NLP 是研究人与机器之间用自然语言进行有效通信的理论和方法。这需要很多跨学科的知识，需要语言学、统计学、最优化理论、机器学习、深度学习以及自然语言处理相关理论模型知

识做基础。作为一门杂学，NLP 可谓是包罗万象，体系化与特殊化并存，这里简单罗列其知识体系：

●句法语义分析：针对目标句子，进行各种句法分析，如分词、词性标记、命名实体识别及链接、句法分析、语义角色识别和多义词消歧等。

●关键词抽取：抽取目标文本中的主要信息，比如从一条新闻中抽取关键信息。主要是了解是谁、于何时、为何、对谁、做了何事、产生了有什么结果。涉及实体识别、时间抽取、因果关系抽取等多项关键技术。

●文本挖掘：主要包含了对文本的聚类、分类、信息抽取、摘要、情感分析以及对挖掘的信息和知识的可视化、交互式的呈现界面。

●机器翻译：将输入的源语言文本通过自动翻译转化为另一种语言的文本。根据输入数据类型的不同，可细分为文本翻译、语音翻译、手语翻译、图形翻译等。机器翻译从最早的基于规则到二十年前的基于统计的方法，再到今天的基于深度学习（编解码）的方法，逐渐形成了一套比较严谨的方法体系。

●信息检索：对大规模的文档进行索引。可简单对文档中的词汇，赋以不同的权重来建立索引，也可使用算法模型来建立更加深层的索引。查询时，首先对输入比进行分析，然后在索引里面查找匹配的候选文档，再根据一个排序机制把候选文档排序，最后输出排序得分最高的文档。

●问答系统：针对某个自然语言表达的问题，由问答系统给出一个精准的答案。需要对自然语言查询语句进行语义分析，包括实体链接、关系识别，形成逻辑表达式，然后到知识库中查找可能的候选答案并通过一个排序机制找出最佳的答案。

●对话系统：系统通过多回合对话，跟用户进行聊天、回答、完成某项任务。主要涉及用户意图理解、通用聊天引擎、问答引擎、对话管理等技术。此外，为了体现上下文相关，要具备多轮对话能力。同时，为了体现个性化，对话系统还需要基于用户画像做个性化回复。知识结构结构图如图 3.2 所示。

图 3.2 知识结构图示

3.1.4 语料库

巧妇难为无米之炊，语料库就是 NLP 的“米”，本书用到的语料库主要有：

（1）中文维基百科

维基百科是最常用且权威的开放网络数据集之一，作为极少数的人工编辑、内容丰富、格式规范的文本语料，各类语言的维基百科在 NLP 等诸多领域应用广泛。维基百科提供了开放的词条文本整合下载，可以找到你需要的指定时间、指定语言、指定类型、指定内容的维基百科数据，中文维基百科数据是维基提供的语料库。

（2）搜狗新闻语料库

来自若干新闻站点 2012 年 6 月～7 月期间国内、国际、体育、社会、娱乐等 18 个频道的新闻数据，提供 URL 和正文信息。

（3）IMDB 情感分析语料库

互联网电影资料库（Internet Movie Database，简称 IMDb）是一个关于电影演员、电影、电视节目、电视明星和电影制作的在线数据库。IMDb 的资料中包括了影片的众多信息、演员、片长、内容介绍、分级、评论等。对于电影的评分目前使用最多的就是 IMDb 评分。

还有豆瓣读书相关语料（爬虫获取）、邮件相关语料等。

3.1.5 探讨的自然语言处理的几个层面

本书所探讨的自然语言处理可以分为以下三个层面：

(1) 第一层面：词法分析

词法分析包括汉语的分词和词性标注这两部分。之前有提过，汉语分词与英文不同，汉语书面词语之间没有明显的空格标记，文本中的句子以字符串的方式出现，句子中由逗号分隔，句子和句子之间常以句号分隔。针对汉语这种独特的书面表现形式，汉语的自然语言处理的首要工作就是要将输入的文本切分为单独的词语，然后在此技术上进行其他更高级的分析。

上述这个步骤称为分词。除了分词之外，词性标注也通常被认为是词法分析的一部分，词性标注的目的是为每一个词赋予一个类别，这个类别可以是名词（noun）、动词（verb）、形容词（adjective）等。通常来说，属于相同词性的词，在句法中承担类似的角色。

(2) 第二层面：句法分析

句法分析是对输入的文本以句子为单位，进行分析以得到句子的句法结构的处理过程。对句法结构进行分析，一方面是为了帮助理解句子的含义，另一方面也为更高级的自然语言处理任务提供支持（比如机器翻译、情感分析等）。目前业界存在三种比较主流的句法分析方法：短语结构句法体系，作用是识别出句子中的短语结构以及短语之间的层次句法关系；依存结构句法体系，作用是识别句子中词与词之间的相互依赖关系；深层文法句法分析，利用深层文法，例如词汇化树邻接文法，组合范畴文法等对句子进行深层的句法以及语义分析。

上述几种句法分析，依存句法分析属于浅层句法分析，其实现过程相对来说比较简单而且适合在多语言环境下应用，但是其所能提供的信息也相对较少。深层文法句法分析可以提供丰富的句法和语义信息，但是采用的文法相对比较复杂，分析器的运行复杂度也比较高，这使得深层句法分析不太适合处理大规模的数据。短语结构句法分析介于依存句法分析和深层文法句法分析之间。

(3) 第三个层面：语义分析

语义分析的最终目的是理解句子表达的真实语义。但是，语义应该采用什么表示形式一直困扰着研究者们，至今这个问题也没有一个统一的答案。语义角色标注（semantic role labeling）是目前比较成熟的浅层语义分析技术。语义角色标注一般都在句法分析的基础上完成，句法结构对于语义角色标注的性能至关重要。基于逻辑表达的语义分析也得到学术界的长期关注。出于机器学习模型复杂度、效率的考虑，自然语言处理系统通常采用级联的方式，即分词、词性标注、句法分析、语义分析分别训练模型。实际使用时，给定输入句子，逐一使用各个模块进行分析，最终得到所有结果。

近年来，随着研究工作的深入，研究者们提出了很多有效的联合模型，将多个任务联合学习和解码，如分词词性联合、词性句法联合、分词词性句法联合、句法语义联合等。联合模型通常都可以显著提高分析质量，原因在于联合模型可以让相互关联的多个任务互相帮助，同时对于任何单任务而言，人工标注的信息也更多了。然而，联合模型的复杂度更高，速度也更慢。

3.1.6 自然语言处理与人工智能

NLP 是计算机领域与人工智能领域中的一个重要分支。人工智能（Artificial Intelligence，AI）在 1955 年达特茅斯特会议上被提出，而后人工智能先后经历了三次浪潮，但是在 20 世纪 70 年代第一次 AI 浪潮泡沫破灭之后，这一概念迅速进入沉寂，相关研究者都不愿提起自己是研究人工智能的，转而研究机器学习、数据挖掘、自然语言处理等各个方向。1990 年迎来第二次黄金时代，同期日本意欲打造传说中的“第五代计算机”，日本当时宣称第五代计算机的能力就是能够自主学习，而随着第五代计算机研制的失败，人工智能再次进入沉寂期。2008 年左右，由于数据的大幅增强、计算力的大幅提升、深度学习实现端到端的训

练，深度学习引领人工智能进入第三波浪潮。人们也逐渐开始将如日中天的深度学习方法引入到 NLP 领域中，在机器翻译、问答系统、自动摘要等方向取得成功。

3.2 中文文本处理步骤

中文文本处理通常包含三个步骤，分词、词性标注、句法分析。

3.2.1 中文分词技术

在语言理解中，词是最小的能够独立活动的有意义的语言成分。将词确定下来是理解自然语言的第一步，只有跨越了这一步，中文才能像英文那样过渡到短语划分、概念抽取以及主题分析，以至自然语言理解，最终达到智能计算的最高境界。因此，每个 NLP 工作者都应掌握分词技术。

词”这个概念一直是汉语语言学界纠缠不清而又绕不开的问题。“词是什么”（词的抽象定义）和“什么是词”（词的具体界定），这两个基本问题迄今为止也未能有一个权威、明确的表述，更无法拿出令大众认同的词表来。主要难点在于汉语结构与印欧体系语种差异甚大，对词的构成边界方面很难进行界定。比如，在英语中，单词本身就是“词”的表达，一篇英文文章就是“单词”加分隔符（空格）来表示的，而在汉语中，词以字为基本单位的，但是一篇文章的语义表达却仍然是以词来划分的。因此，在处理中文文本时，需要进行分词处理，将句子转化为词的表示。这个切词处理过程就是中文分词，它通过计算机自动识别出句子的词，在词间加入边界标记符，分隔出各个词汇。整个过程看似简单，然而实践起来却很复杂，主要的困难在于分词歧义。以 NLP 分词的经典语句举例，“结婚的和尚未结婚的”，应该分词为“结婚/的/和/尚未/结婚/的”，还是“结婚/的/和尚/未/结婚/的”？这个由人来判定都是问题，机器就更难处理了。此外，像未登录词、分词粒度粗细等都是影响分词效果的重要因素。

自中文自动分词被提出以来，历经将近 30 年的探索，提出了很多方法，可主要归纳为“规则分词”“统计分词”和“混合分词（规则+统计）”这三个主要流派。规则分词是最早兴起的方法，主要是通过人工设立词库，按照一定方式进行匹配切分，其实现简单高效，但对新词很难进行处理。随后统计机器学习技术的兴起，应用于分词任务上后，就有了统计分词，能够较好应对新词发现等特殊场景。然而实践中，单纯的统计分词也有缺陷，那就是太过于依赖语料的质量，因此实践中多是采用这两种方法的结合，即混合分词。

一、 规则分词

基于规则的分词是一种机械分词方法，主要是通过维护词典，在切分语句时，将语句的每个字符串与词表中的词进行逐一匹配，找到则切分，否则不予切分。按照匹配切分的方式，主要有正向最大匹配法、逆向最大匹配法以及双向最大匹配法三种方法。

（一）正向最大匹配法

正向最大匹配（Maximum Match Method，MM 法）的基本思想为：假定分词词典中的最长词有 i 个汉字字符，则用被处理文档的当前字串中的前 i 个字作为匹配字段，查找字典。若字典中存在这样的一个 i 字词，则匹配成功，匹配字段被作为一个词切分出来。如果词典中找不到这样的一个 i 字词，则匹配失败，将匹配字段中的最后一个字去掉，对剩下的字串重新进行匹配处理。如此进行下去，直到匹配成功，即切分出一个词或剩余字串的长度为零为止。这样就完成了一轮匹配，然后取下一个 i 字字串进行匹配处理，直到文档被扫描完为止。其算法描述如下：

1）从左向右取待切分汉语句的 m 个字符作为匹配字段，m 为机器词典中最长词条的字符数。

2）查找机器词典并进行匹配。若匹配成功，则将这个匹配字段作为一个词切分出来。若匹配不成功，则将这个匹配字段的最后一个字去掉，剩下的字符串作为新的匹配字段，进行再次匹配，重复以上过程，直到切分出所有词为止。

比如我们现在有个词典，最长词的长度为 5，词典中存在“南京市长”和“长江大桥”两个词。现采用正向最大匹配对句子“南京市长江大桥”进行分词，那么首先从句子中取出前五个字“南京市长江”，发现词典中没有该词，于是缩小长度，取前 4 个字“南京市长”，词典中存在该词，于是该词被确认切分。再将剩下的“江大桥”按照同样方式切分，得到“江”“大桥”，最终分为“南京市长”“江”“大桥”3 个词。显然，这种结果还

不是我们想要的。

（二）逆向最大匹配法

逆向最大匹配（Reverse Maximum Match Method，RMM 法）的基本原理与 MM 法相同，不同的是分词切分的方向与 MM 法相反。逆向最大匹配法从被处理文档的末端开始匹配扫描，每次取最末端的 i 个字符（i 为词典中最长词数）作为匹配字段，若匹配失败，则去掉匹配字段最前面的一个字，继续匹配。相应地，它使用的分词词典是逆序词典，其中的每个词条都将按逆序方式存放。在实际处理时，先将文档进行倒排处理，生成逆序文档。然后，根据逆序词典，对逆序文档用正向最大匹配法处理即可。

由于汉语中偏正结构较多，若从后向前匹配，可以适当提高精确度。所以，逆向最大匹配法比正向最大匹配法的误差要小。统计结果表明，单纯使用正向最大匹配的错误率为 1/169，单纯使用逆向最大匹配的错误率为 1/245。比如之前的“南京市长江大桥”，按照逆向最大匹配，最终得到“南京市”“长江大桥”。当然，如此切分并不代表完全正确，可能有个叫“江大桥”的“南京市长”也说不定。

（三）双向最大匹配法

双向最大匹配法（Bi-directction Matching method）是将正向最大匹配法得到的分词结果和逆向最大匹配法得到的结果进行比较，然后按照最大匹配原则，选取词数切分最少的作为结果。据 SunM.S.和 Benjamin K.T.（1995）的研究表明，中文中 90.0%左右的句子，正向最大匹配法和逆向最大匹配法完全重合且正确，只有大概 9.0%的句子两种切分方法得到的结果不一样，但其中必有一个是正确的（歧义检测成功），只有不到 1.0%的句子，使用正向最大匹配法和逆向最大匹配法的切分虽重合却是错的，或者正向最大匹配法和逆向最大匹配法切分不同但两个都不对（歧义检测失败）。这正是双向最大匹配法在实用中文信息处理系统中得以广泛使用的原因。

前面举例的“南京市长江大桥”，采用该方法，中间产生“南京市/长江/大桥”和“南京市/长江大桥”两种结果，最终选取词数较少的“南京市/长江大桥”这一结果。

下面是一段实现逆向最大匹配的代码。

```
#逆向最大匹配
class IMM(object):
    def__init__(self, dic_path):
        self.dictinary = set()
        self.maximum = 0
        #读取词典
        with open(dic_path, ‘r’, enconding = ‘utf-8’) as f :
            for line in f :
                line = line.strip()
                if not line :
                    continue
                self.dictionary.add(line)
                self.maximum = len(line)
    def cut(self, test):
        result = []
        index = len(text)
        while index >0 :
            word = None
            for size in rang(self.maximum, 0, -1):
                if index  -  size < 0 :
                    continue
                piece = text[(index  -  size ):index]
                if piece in self.dictionary:
                    word = piece
                    result.append(word)
                    index - = size
                    break
                if word is None :
                    index - =1
            return result[::-1]
def main():
    text = “南京市长江大桥”
    tokenizer = IMM( ‘./date/imm_dic.utf8’ )
    print(tokenizer.cut(text))
运行 main 函数，结果为:
[‘南京市’,’长江大桥’]
```

基于规则的分词，一般都较为简单高效，但是词典的维护是一个很庞大的工程。在网络发达的今天，网络新词层出不穷，很难通过词典覆盖到所有词。

二、统计分词

随着大规模语料库的建立，统计机器学习方法的研究和发展，基于统计的中文分词算法渐渐成为主流。

其主要思想是把每个词看做是由词的最小单位的各个字组成的，如果相连的字在不同的文本中出现的次数越多，就证明这相连的字很可能就是一个词。因此我们就可以利用字与字相邻出现的频率来反应成词的可靠度，统计语料中相邻共现的各个字的组合的频度，当组合频度高于某一个临界值时，我们便可认为此字组可能会构成一个词语。

基于统计的分词，一般要做如下两步操作：

1）建立统计语言模型。

2）对句子进行单词划分，然后对划分结果进行概率计算，获得概率最大的分词方式。这里就用到了统计学习算法，如隐含马尔可夫（HMM）、条件随机场（CRF）等。

（一）语言模型

语言模型在信息检索、机器翻译、语音识别中承担着重要的任务。用概率论的专业术语描述语言模型就是：为长度为 m 的字符串确定其概率分布 P(ω1，ω2，…，ωm)，其中 ω1 到 ωm 依次表示文本中的各个词语。一般采用链式法则计算其概率值，如式（3.1）所示：

$$P(\omega_1,\omega_2,\ldots,\omega_m)=P(\omega_1)P(\omega_2|\omega_1)P(\omega_3|\omega_1,\omega_2)P(\omega_i|\omega_1,\omega_2,\ldots,\omega_{i-1})\ldots P(\omega_m|\omega_1,\omega_2,\ldots,\omega_{m-1}) \quad (3.1)$$

观察式（3.1）易知，当文本过长时，公式右部从第三项起的每一项计算难度都很大。为解决该问题，有人提出 n 元模型（n-gram model）降低该计算难度。所谓 n 元模型就是在估算条件概率时，忽略距离大于等于 n 的上文词的影响，因此 P(ωi|ω1，ω2，…，ωi-1)的计算可简化为：

$$P(\omega_i|\omega_1,\omega_2,\ldots,\omega_{i-1})\approx P(\omega_i|\omega_{i-(n-1)},\ldots,\omega_{i-1}) \quad (3.2)$$

当 n=1 时称为一元模型（unigram model），此时整个句子的概率可表示为：$P(\omega_1,\omega_2,\ldots,\omega_m)=P(\omega_1)P(\omega_2)\ldots P(\omega_m)$观察可知，在一元语言模型中，整个句子的概率等于各个词语概率的乘积。言下之意就是各个词之间都是相互独立的，这无疑是完全损失了句中的词序信息。所以一元模型的效果并不理想。

当 n=2 时称为二元模型（bigram model），式（3.2）变为$P(\omega_i|\omega_1,\omega_2,\ldots,\omega_{i-1})=P(\omega_i|\omega_{i-1})$。当 n=3 时称为三元模型（trigram model），式（3.2）变为$P(\omega_i|\omega_1,\omega_2,\ldots,\omega_{i-1})=P(\omega_i|\omega_{i-2},\omega_{i-1})$。显然当 n≥2 时，该模型是可以保留一定的词序信息的，而且 n 越大，保留的词序信息越丰富，但计算成本也呈指数级增长。

一般使用频率计数的比例来计算 n 元条件概率，如式（3.3）所示：

$$P(\omega_i|\omega_{i-(n-1)},\ \ldots,\ \omega_{i-1})=\frac{count(\omega_{i-(n-1)},\ \ldots,\ \omega_{i-1},\ \omega_i)}{count(\omega_{i-(n-1)},\ \ldots,\ \omega_{i-1})} \quad (3.3)$$

式中$count(\omega_{i-(n-1)},\ldots,\omega_{i-1})$表示词语$\omega_{i-(n-1)},\ldots,\omega_{i-1}$在语料库中出现的总次数。

由此可见，当 n 越大时，模型包含的词序信息越丰富，同时计算量随之增大。与此同时，长度越长的文本序列出现的次数也会减少，如按照公式（3.3）估计 n 元条件概率时，就会出现分子分母为零的情况。因此，一般在 n 元模型中需要配合相应的平滑算法解决该问题，如拉普拉斯平滑算法等。

（二）HMM 模型

隐含马尔可夫模型（HMM）是将分词作为字在字串中的序列标注任务来实现的。其基本思路是：每个字在构造一个特定的词语时都占据着一个确定的构词位置（即词位），现规定每个字最多只有四个构词位置：即 B（词首）、M（词中）、E（词尾）和 S（单独成词），那么下面句子 1）的分词结果就可以直接表示成如 2）所示的逐字标注形式：

1）中文/分词/是/.文本处理/不可或缺/的/一步！

2）中/B 文/E 分/B 词/E 是/S 文/B 本/M 处/M 理/E 不/B 可/M 或/M 缺/E 的/S 一/B 步/E！/S

用数学抽象表示如下：用$\lambda=\lambda_1\lambda_2\ldots\lambda_n$代表输入的句子，n 为句子长度，$\lambda_i$表示字，$o=o_1o_2\ldots o_n$代表输出的标签，那么理想的输出即为：

$$\max=\max P(o_1o_2\ldots o_n|\lambda_1\lambda_2\ldots\lambda_n) \quad (3.4)$$

在分词任务上，o 即为 B、M、E、S 这 4 种标记，λ 为诸如“中”“文”等句子中的每个字（包括标点等非中文字符）。

需要注意的是，P(o|λ)是关于 2n 个变量的条件概率，且 n 不固定。因此，几乎无法对 P(o|λ)进行精

确计算。这里引入观测独立性假设，即每个字的输出仅仅与当前字有关，于是就能得到下式：

$$P(o_1 o_2 \dots o_n|\lambda_1\lambda_2 \dots \lambda_n) = P(o_1|\lambda_1)P(o_2|\lambda_2) \dots P(o_n|\lambda_n) \quad (3.5)$$

事实上，$P(o_k|\lambda_k)$的计算要容易得多。通过观测独立性假设，目标问题得到极大简化。然而该方法完全没有考虑上下文，且会出现不合理的情况。比如按照之前设定的 B、M、E 和 S 标记，正常来说 B 后面只能是 M 或者 E，然而基于观测独立性假设，我们很可能得到诸如 BBB、BEM 等的输出，显然是不合理的。

HMM 就是用来解决该问题的一种方法。在上面的公式中，我们一直期望求解的是 P(o|λ)，通过贝叶斯公式能够得到：

$$P(o|\lambda) = \frac{P(o,\lambda)}{P(\lambda)} = \frac{P(\lambda|o)P(o)}{P(\lambda)} \quad (3.6)$$

λ 为给定的输入，因此 P(λ)计算为常数，可以忽略，因此最大化 P(o|λ)等价于最大化 P(λ|o)P(o)。

针对 P(λ|o)P(o)作马尔可夫假设，得到：

$$P(\lambda|o) = P(\lambda_1|o_1)P(\lambda_2|o_2)\dots P(\lambda_n|o_n) \quad (3.7)$$

同时，对 P(o)有：

$$P(o) = P(o_1)P(o_2|o_1)P(o_3|o_1,o_2)\dots P(o_n|o_1,o_2\dots,o_{n-1}) \quad (3.8)$$

这里 HMM 做了另外一个假设——齐次马尔可夫假设，每个输出仅仅与上一个输出有关，那么：

$$P(o) = P(o_1)P(o_2|o_1)P(o_3|o_2)\dots P(o_n|o_{n-1}) \quad (3.9)$$

于是：

$$P(\lambda|o)P(o) \sim P(\lambda_1|o_1)P(o_2|o_1)P(\lambda_2|o_2)P(o_3|o_2)\dots P(o_n|o_{n-1})P(\lambda_n|o_n) \quad (3.10)$$

在 HMM 中，将$P(\lambda_k|o_k)$称为发射概率，$P(o_k|o_{k-1})$称为转移概率。通过设置某些$P(o_k|o_{k-1}) = 0$，可以排除类似 BBB、EM 等不合理的组合。

事实上，式（3.9）的马尔可夫假设就是一个二元语言模型（bigram model），当将齐次马尔可夫假设改为每个输出与前两个有关时，就变成了三元语言模型（trigram model）。当然在实际分词应用中还是多采用二元模型，因为相比三元模型，其计算复杂度要小不少。

在 HMM 中，求解$max\, P\,(\lambda|o)P(o)$的常用方法是 Veterbi 算法。它是一种动态规划方法，核心思想是：如果最终的最优路径经过某个o_i，那么从初始节点到$o_{i\text{-}1}$点的路径必然也是一个最优路径——因为每一个节点o_i只会影响前后两个$P(o_{i\text{-}1}|o_i)$和$P(o_i|o_{i+1})$。

根据这个思想，可以通过递推的方法，在考虑每个$o_i o_i$时只需要求出所有经过各$o_{i\text{-}1}$的候选点的最优路径，然后再与当前的o_i结合比较。这样每步只需要算不超过l^2次，就可以逐步找出最优路径。Viterbi 算法的效率是$O(n \cdot l^2)$，l 是候选数目最多的节点o_i的候选数目，它正比于 n，这是非常高效率的。HMM 的状态转移图如图 3.3 所示。

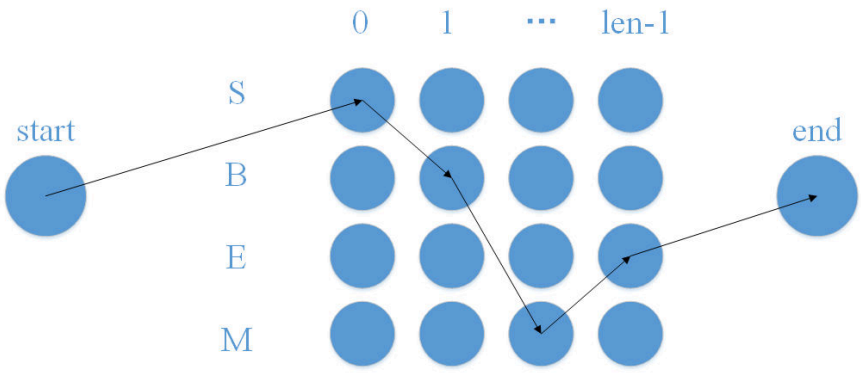


图 3.3 HMM 状态转移示意图

```
class HMM(object):
    def __init__ (self):
        pass
    def try_load_model(self,trained):
        pass
    def train(self, path):
        pass
    def viterbi(self,text,states,start_p,trans_p,emit_p):
        pass
    def cut(self,text):
        pass
```

__init__主要是初始化一些全局信息，用于初始化一些成员变量。如状态集合（标记 S、B、E、M），以及存取概率计算的中间文件。

```
def_init(self):
    import os
    #主要是用于存取算法中间结果，不用每次都训陈模型
    self.model_file='./data/hmmmodel.pkl'

    #状态值集合
    self.state lit=['Bl,'W', 'E', 's']
    #参数加载，用于判断是否需要重新加载model_file
    self.load_para = False
```

try_load_model 接收一个参数，用于判别是否加载中间文件结果。当直接加载中间结果时，可以不通过语料库训练，直接进行分词调用。否则，该函数用于初始化初始概率、转移概率以及发射概率等信息。这里初始概率是指，一句话第一个字被标记成“S”“B”“E”或“M”的概率。

```
#用于加载已计算的中间结果，当需要重新训陈时，需初始化清空结果
def try_load_model(self,trained):
    if Erained:
        import pickle
        with open(self.modelfile,'rb') as f:
        self.A_dic = pickle.load(f)
        self.B_dic = pickle.load(f)
        self.Pi_dic =pickle.load(f)
        self.load_para = True
    else:
        #状态转移概率（状态->状态的条件概率）#发射概率（状态->词语的条件概率）
        self.A_dic = {}
        #发射概率（状态->词语的条件概率）
        self.B dic ={}
        #状态的初始概率
```

```
        seif.Pi_dic ={}
        self.load_para= False
```

train 函数主要用于通过给定的分词语料进行训练。语料的格式为每行一句话（这里以逗号隔开也算一

```
            line_state = [ ]
            foe w in linelist:
                line_state.extend(makeLabel(w))
            assert len(word_list) == len(line_state)

            for k, v in enumerate(line_state):
                Count_dic[v] += 1
                if k == 0:
                    self.A_dic[v] += 1  #每个句子的第一个字的状态，用于计算初始
        状态概率
                else:
                    self.A_dic[line_state[k-1]][v] += 1 #就算转移率
                    self.B_dic[line_state[k]][word_list[k]] = \
                        self.B_dic[line_state[k].get(word_list[k],0) + 1.0  #计
        算发射概率
        self.Pi_dic = {k: v *1.0 / line_num for k, v in self.Pi_dic.items()}
        self.A_dic = {k: {k1: v1 / Count_dic[k] for k1, v1, in v.items()}
                    for k, v in self.A_dic.items()}
         #加 1 平滑
        self.B_dic = {k: {k1: (v1 + 1) / Count_dic[k] for k1, v1, in v.items()}
                    for k, v in self.B_dic.items()}       #序列化
        import pickle
        with open(self.model_file, 'wb') as f:
            pickle.dump(self.A_dic,f)
            pickle.dump(self.B_dic,f)
            pickle.dump(self.Pi_dic,f)
    words = set()
    with open(path, encoding = 'utf8') as f:
        for line in f :
            line_num +=1

            line = line.strip()
            if not line:
                continue
            word_list = [i for i in line if i !=' ']
            words |= set(word_list) #更新字的集合

            linelist = line.splist()
```

句），每个词以空格分隔，示例中采用了人民日报的分词语料，放置于 chapter2/data/trainCorpus.txt_utf8 中。该函数主要通过对语料的统计，得到 HMM 所需的初始概率、转移概率以及发射概率。其代码如下：

cut 函数用于切词，其通过加载中间文件，调用 veterbi 函数来完成。veterbi 函数即为 Veterbi 算法的实现，是基于动态规划的一种实现，主要是求最大概率的路径。其输入参数为初始概率、转移概率以及发射概率，加上需要切分的句子。veterbi 函数和 cut 函数代码如下：

```
                text[t] not in emit_p['E'].keys() and \
                text[t] not in emit_p['B'].keys()
            for y in states:
                emitP = emit_p[y].get(text[t],0) if not neverSeen else 1.0   #设置位置字单独成词
                (prob,state)= max([(V[t - 1][y0]* trans_p[y0].get(y,0)*emitP,70)
                    for y0 in states if V[t – 1][y0] > 0])
                V[t][y] = prob
                newpath[y] = path[state] + [y]
            path = newpath
        if emit_p['M'].get(text[-1], 0)> emit_p['S'].get(text[-1], 0):
            (prob,state) = max([(V[len(text) - 1][y], y) for y in('E', 'M')])
        else:
            (prob, state) = max([(V[len(text) - 1][y],y) for y in states])

            Return (prob, path[state])
def cut(self, text):
        import os
        if not self.load_para:
            self.try_load_model(os.path.exists(self.model_file))
        prob,pos_list=se1f.viterbi(text, self.state_list, self.Pi_dic, self.A_dic, self.B_dic)
        begin, next = 0, 0
        for i, char in enumerate(text):
            pos = pos_list[i]
            if pos =='B':
                begin = i
            elif pos == 'E':
                yield text[begin: i+1]
            elif pos =='s':
                yield char
                next = i+1
        if next < len(text):
            yield text[next:]

            V.append({})
            newpath = {}
            neverSeen = text[t] not in emit_p['S'].keys() and \
                text[t] not in emit_p['M'].keys() and \
```

我们可以测试下上面的分词实现，比如查看切分“中文博大精深！”这句话。基本上分词效果还可以。当然这里示例的 HMM 程序较为简单，且训练采用的语料规模并不大。实际项目实战中，读者可通过扩充语料、词典补充等手段予以优化。

```
hmm=HDM()
hmm.train('./data/trainCorpus.txt_utf8')
text ='这是一个非常棒的方案!'
res =hmm.cut(text)
print(text)
print(str(list(res)))
------可以查看结果------
这是一个非常棒的方案!
['这是', '一个', '非常', '棒', '的', '方案', '!']
```

（三）其他统计分词算法

条件随机场（CRF）也是一种基于马尔可夫思想的统计模型。在隐含马尔可夫中，有个很经典的假设，那就是每个状态只与它前面的状态有关。这样的假设显然是有偏差的，于是学者们提出了条件随机场算法，使得每个状态不止与他前面的状态有关，还与他后面的状态有关。该算法在本节将不会重点介绍，会在后续章节详细介绍。

神经网络分词算法是深度学习方法在 NLP 上的应用。通常采用 CNN、LSTM 等深度学习网络自动发现一些模式和特征，然后结合 CRF、softmax 等分类算法进行分词预测。基于深度学习的分词方法，我们将在后续介绍完深度学习相关知识后，再做拓展。

对比机械分词法，这些统计分词方法不需耗费人力维护词典，能较好地处理歧义和未登录词，是目前分词中非常主流的方法。但其分词的效果很依赖训练语料的质量，且计算量相较于机械分词要大得多。

三、混合分词

事实上，目前不管是基于规则的算法、还是基于 HMM、CRF 或者 deep learning 等的方法，其分词效果在具体任务中，其实差距并没有那么明显。在实际工程应用中，多是基于一种分词算法，然后用其他分词算法加以辅助。最常用的方式就是先基于词典的方式进行分词，然后再用统计分词方法进行辅助。如此，能在保证词典分词准确率的基础上，对未登录词和歧义词有较好的识别，下节介绍的 Jieba 分词工具便是基于这种方法的实现。

四、中文分词工具 Jieba

Jieba 提供了三种分词模式：

- 精确模式：试图将句子最精确地切开，适合文本分析。
- 全模式：把句子中所有可以成词的词语都扫描出来，速度非常快，但是不能解决歧义。
- 搜索引擎模式：在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

下面是使用这三种模式的对比

```
import jieba
sent = '中文分词是文本处理不可或缺的一步!'
seg_list = jieba.cut(sent,    cut_all =True)
print('全模式:'，'/'.join(seg_list)seg_list)
seg_lsit =jieba.cut(sent, cut_all=False)
print('精确模式:'，'/'.join(seg_list))
seg_list = jieba.cut(sent)
print('默认精确模式:'，'/'.join(seg_list))
seg_list = jieba.cut_for_search(sent)
print('搜索引擎模式'，'/'.join(seg_list))
```

运行结果如下：

```
全模式：中文/分词/是/文本/本处处理/本处/处理/不可/不可或缺/或缺/的/一步/!
精确模式：中文/分词/是/文本处理/不可或缺/的/一步/!
默以精确模式：中文/分词是/文本处理/不可或缺/的/一步/!
搜索引擎模式：中文/分词/是/文本/本处/处理/文本处理/不可/或缺/不可或缺/的/一步!
```

可以看到，全模式和搜索引擎模式下，Jieba 将会把分词的所有可能都打印出来。一般直接使用精确模式即可，但是在某些模糊匹配场景下，使用全模式或搜索引擎模式更适合。

3.2.2 词性标注

词性是词汇基本的语法属性，通常也称为词类。词性标注是在给定句子中判定每个词的语法范畴，确定其词性并加以标注的过程。例如，表示人、地点、事物以及其他抽象概念的名称即为名词，表示动作或状态变化的词为动词，描述或修饰名词属性、状态的词为形容词。如给定一个句子：“这儿是个非常漂亮的公园”，对其的标注结果应如下：“这儿/代词 是/动词 个/量词 非常/副词 漂亮/形容词 的/结构助词 公园/名词”。

在中文中，一个词的词性很多时候都不是固定的，一般表现为同音同形的词在不同场景下，其表示的语法属性截然不同，这就为词性标注带来很大的困难；但是另外一方面，从整体上看，大多数词语，尤其是实词，一般只有一到两个词性，且其中一个词性的使用频次远远大于另一个，即使每次都将高频词性作为词性选择进行标注，也能实现 80%以上的准确率。如此，若我们对常用词的词性能够进行很好地识别，那么就能够覆盖绝大多数场景，满足基本的准确度要求。

词性标注最简单的方法是从语料库中统计每个词所对应的高频词性，将其作为默认词性，但这样显然还有提升空间。目前较为主流的方法是如同分词一样，将句子的词性标注作为一个序列标注问题来解决，那么分词中常用的手段，如隐含马尔可夫模型、条件随机场模型等皆可在词性标注任务中使用。本节将继续介绍如何使用 Jieba 分词来完成词性标注任务。

一、词性标注规范

词性标注需要有一定的标注规范，如将词分为名词、形容词、动词，然后用“n”“adj”“v”等来进行表示。中文领域中尚无统一的标注标准，较为主流的主要为北大的词性标注集和宾州词性标注集两大类。两类标注方式各有千秋，一般我们任选一种方式即可。本书中采用北大词性标注集作为标准，其部分标注的词性如表 3.1 所示。

表 3.1 词性标注规范表

标记	词性	说明
ag	形语素	形容词性语素。形容词代码为 a，语素代码 g 前面置以 a
a	形容词	取英语形容词 adjective 的第一个字母
ad	副形词	直接作状语的形容词。形容词代码 a 和副词代码 d 并在一起
an	名形词	具有名词功能的形容词。形容词代码 a 和名词代码 n 并在一起
b	区别词	取汉字“别”的声母
c	连词	取英语连词 conjunction 的第一个字母
dg	副语素	副词性语素。副词代码为 d，语素代码 g 前面置以 d
d	副词	取 adverb 的第二个字母，因其第一个字母已用于形容词
e	叹词	取英文叹词 exclamation 的第一个字母
f	方位词	取汉字“方”

g	语素	绝大多数语速都能作为合成词的“词根”，取汉字“根”的声母
h	前接成分	取英语 head 的第一个字母
i	成语	取英语成语 idiom 的第一个字母
j	简称略语	取汉字“简”的声母
k	后接成分	
l	习用语	习用语尚未成为成语，有点“临时性”，取“临”的声母
m	数词	取英语 numeral 的第三个字母，n、u 已有他用
ng	名语素	名词性语素。名词代码为 n，语素代码 g 前面置以 n
n	名词	取英语名词 noun 的第一个字母
nr	人名	名词代码 n 和“人（ren）”的声母并在一起
ns	地名	名词代码 n 和处所词代码 s 并在一起
nt	机构团体	“团”的声母为 t，名词代码 n 和 t 并在一起
nz	其他专名	“专”的声母的第一个字母为 z，名词代码 n 和 z 并在一起
o	拟声词	取英文拟声词 onomatopoeia 的第一个字母
p	介词	取英语介词 prepositional 的第一个字母
q	量词	取英语 quantity 的第一个字母
r	代词	取英语代词 pronoun 的第二个字母，因为 p 已用于介词
s	处所词	取英语 space 的第一个字母
tg	时语素	时间词性语素。时间词代码为 t，在语素的代码 g 前面置以 t
t	时间词	取英语 time 的第一个字母
u	助词	取英语助词 auxiliary 第二个字母
vg	动语素	动词性语素。动词代码为 v。在语素的代码 g 前面置以 v
v	动词	取英语动词 verb 的第一个字母
vd	副动词	直接作状语的动词。动词和副词的代码并在一起
vn	名动词	指具有名字功能的动词。动词和名词的代码并在一起
w	标点符号	
x	非语素字	非语素字只是一个符号，字母 x 通常用于代表未知数、符号
y	语气词	取汉字“语”的声母
z	状态词	取汉字“状”的声母的前一个字母

二、Jieba 分词中的词性标注

在上节分词中，我们介绍了 Jieba 分词的分词功能，这里将介绍其词性标注功能。类似 Jieba 分词的分词流程，Jieba 的词性标注同样是结合规则和统计的方式，具体为在词性标注的过程中，词典匹配和 HMM 共同作用。词性标注流程如下。

1） 首先基于正则表达式进行汉字判断，正则表达式如下：

```
re_han_internal = re.compile("([\u4E00-\u9FD5a-zA-20-9+#&\._]+)")
```

2）若符合上面的正则表达式，则判定为汉字，然后基于前缀词典构建有向无环图，再基于有向无环图计算最大概率路径，同时在前缀词典中找出它所分出的词性，若在词典中未找到，则赋予词性为“x”（代表未知）。当然，若在这个过程中，设置使用 HMM，且待标注词为未登录词，则会通过 HMM 方式进行词性标注。

3）若不符合上面的正则表达式，那么将继续通过正则表达式进行类型判断，分别赋予“x”“m”（数词）和“eng”（英文）。

这里简单说明下 HMM 是如何应用于词性标注任务中的。分词任务中，我们设置了“B”“S”“M”和“E”四种标签，与句子中的每个字符一一对应。而在词性标注任务中，Jieba 分词采用了 simultaneous 思想的联合模型方法，即将基于字标注的分词方法与词性标注结合起来，使用复合标注集。比如，对于名词“人民”，它的词性标注是“n”，而分词的标注序列是“BE”，于是“人”的标注就是“B_n”，“民”的标注就是“E_n”。这样，就与 HMM 分词的实现过程一致了，只需更换合适的训练语料即可。

轮到你来：借鉴 3.3.2 节的实现，尝试自己实现 HMM 进行词性标注（语料可选用 1998 年人民日报词性标注集）。

使用 Jieba 分词进行词性标注的示例如下：

```
import jieba.posseg as psg
sent ='中文分词是文本处理不可或缺的一步!'
seg_list = psg.cut(sent)
print(' '.join(['{0}/{1}'.format(w, t) for w, t in seg_list]))
```

```
中文/nz 分词/n 是/v 文本处理/n 不可或缺/1 的/uj 一步/m! /x
```

输出如下，每个词后面跟着其对应的词性，相关词性具体可参考表 3.1。

之前我们介绍过，Jieba 分词支持自定义词典，其中的词频和词性可以省略。然而需要注意的是，若在词典中省略词性，那么采用 Jieba 分词进行词性标注后，最终切分词的词性将变成“x”，这在如语法分析或词性统计等场景下会对结果有一定的影响。因此，在使用 Jieba 分词设置自定义词典时，尽量在词典中补充完整的信息。

三、命名实体识别

与自动分词、词性标注一样，命名实体识别也是自然语言处理的一个基础任务，是信息抽取、信息检索、机器翻译、问答系统等多种自然语言处理技术必不可少的组成部分。其目的是识别语料中人名、地名、组织机构名等命名实体。由于这些命名实体数量不断增加，通常不可能在词典中穷尽列出，且其构成方法具有各自的规律性，因此，通常把对这些词的识别在词汇形态处理（如汉语切分）任务中独立处理，称为命名实体识别（Named Entities Recognition，NER）。NER 研究的命名实体一般分为 3 大类（实体类、时间类和数字类）和 7 小类（人名、地名、组织机构名、时间、日期、货币和百分比）。由于数量、时间、日期、货币等实体识别通常可以采用模式匹配的方式获得较好的识别效果，相比之下人名、地名、机构名较复杂，因此近年来的研究主要以这几种实体为主。

命名实体识别当前并不是一个大热的研究课题，因为学术界部分认为这是一个已经解决了的问题，但是也有学者认为这个问题还没有得到很好地解决，原因主要有：命名实体识别只是在有限的文本类型（主要是新闻语料）和实体类别（主要是人名、地名）中取得了效果；与其他信息检索领域相比，实体命名评测语料较小，容易产生过拟合；命名实体识别更侧重高召回率，但在信息检索领域，高准确率更重要；通用的识别多种类型的命名实体的系统性很差。

同时，中文的命名实体识别与英文的相比，挑战更大，目前未解决的难题更多。命名实体识别效果的

评判主要看实体的边界是否划分正确以及实体的类型是否标注正确。在英文中，命名实体一般具有较为明显的形式标志（如英文实体中的每个词的首字母要大写），因此其实体边界识别相对容易很多，主要重点是在对实体类型的确定。而在汉语中，相较于实体类别标注子任务，实体边界的识别更加困难。

中文命名实体识别主要有以下难点：

●各类命名实体的数量众多。根据对人民日报 1998 年 1 月的语料库（共计 2305896 字）进行的统计，共有人名 19965 个，而这些人名大多属于未登录词。

●命名实体的构成规律复杂。例如由于人名的构成规则各异，中文人名识别又可以细分为中国人名识别、日本人名识别和音译人名识别等；此外机构名的组成方式也最为复杂，机构名的种类繁多，各有独特的命名方式，用词也相当广泛，只有结尾用词相对集中。

●嵌套情况复杂。一个命名实体经常和一些词组合成一个嵌套的命名实体，人名中嵌套着地名，地名中也经常嵌套着人名。嵌套的现象在机构名中最为明显，机构名不仅嵌套了大量的地名，而且还嵌套了相当数量的机构名。互相嵌套的现象大大制约了复杂命名实体的识别，也注定了各类命名实体的识别并不是孤立的，而是互相交织在一起的。

●长度不确定。与其他类型的命名实体相比，长度和边界难以确定使得机构名更难识别。中国人名一般二至四字，常用地名也多为二至四字。但是机构名长度变化范围极大，少到只有两个字的简称，多达几十字的全称。在实际语料中，由十个以上词构成的机构名占了相当一部分比例。

在分词章节，我们介绍了分词主要有三种方式，主要有基于规则的方法、基于统计的方法以及二者的混合方法。这在整个 NLP 的各个子任务基本上也多是同样的划分方式，命名实体识别也不例外：

1）基于规则的命名实体识别：规则加词典是早期命名实体识别中最行之有效的方式。其依赖手工规则的系统，结合命名实体库，对每条规则进行权重赋值，然后通过实体与规则的相符情况来进行类型判断。当提取的规则能够较好反映语言现象时，该方法能明显优于其他方法。但在大多数场景下，规则往往依赖于具体语言、领域和文本风格，其编制过程耗时且难以涵盖所有的语言现象，存在可移植性差、更新维护困难等问题。

2）基于统计的命名实体识别：与分词类似，目前主流的基于统计的命名实体识别方法有：隐马尔可夫模型、最大熵模型、条件随机场等。其主要思想是基于人工标注的语料，将命名实体识别任务作为序列标注问题来解决。基于统计的方法对语料库的依赖比较大，而可以用来建设和评估命名实体识别系统的大规模通用语料库又比较少，这是该方法的一大制约。

3）混合方法：自然语言处理并不完全是一个随机过程，单独使用基于统计的方法使状态搜索空间非常庞大，必须借助规则知识提前进行过滤修剪处理。目前几乎没有单纯使用统计模型而不使用规则知识的命名实体识别系统，在很多情况下是使用混合方法，结合规则和统计方法。

序列标注方式是目前命名实体识别中的主流方法，鉴于 HMM 在之前的章节已有介绍，本节重点介绍基于条件随机场的方法。

4.2.2 基于条件随机场的命名实体识别

在进入条件随机场的命名实体识别之前，我们先温习下分词章节中介绍到的 HMM。HMM 将分词作为字标注问题来解决，其中有两条非常经典的独立性假设：一是输出观察值之间严格独立，二是状态的转移过程中当前状态只与前一状态有关（一阶马尔可夫模型）。通过这两条假设，使得 HMM 的计算成为可能，模型的计算也简单许多。但多数场景下，尤其在大量真实语料中，观察序列更多的是以一种多重的交互特征形式表现出来，观察元素之间广泛存在长程相关性。这样，HMM 的效果就受到了制约。

基于此，在 2001 年，Lafferty 等学者们提出了条件随机场，其主要思想来源于 HMM，也是一种用来标记和切分序列化数据的统计模型。不同的是，条件随机场是在给定观察的标记序列下，计算整个标记序列的联合概率，而 HMM 是在给定当前状态下，定义下一个状态的分布。

条件随机场的定义为：

设 $X=(X_1,X_2,X_3,...,X_n)$ 和 $Y=(Y_1,Y_2,Y_3,...,Y_m)$ 是联合随机变量，若随机变量 Y 构成一个

无向图 G=(V，E) 表示的马尔可夫模型，则其条件概率分布 P(Y|X) 称为条件随机场（Conditional Random Field，CRF），即

$$P(Y_v \mid X, Y_w, w \neq v) = P(Y_v \mid X, Y_w, w \sim v) \qquad (3.11)$$

其中 w～v 表示图 G=(V，E) 中与结点 v 有边连接的所有节点，w≠v 表示结点 v 以外的所有节点。

这里简单举例说明随机场的概念：现有若干个位置组成的整体，当给某一个位置按照某种分布随机赋予一个值后，该整体就被称为随机场。以地名识别为例，假设我们定义了如表 3.2 所示规则。

表 3.2 地理命名实体标记

标注	含义
B	当前词为地理命名实体的首部
M	当前词为地理命名实体的内部
E	当前词为地理命名实体的尾部
S	当前词单独构成地理命名实体
O	当前词不是地理命名实体或组成部分

现有个由 n 个字符构成的 NER 的句子，每个字符的标签都在我们已知的标签集合（“B”“M”“E”“S”和“O”）中选择，当我们为每个字符选定标签后，就形成了一个随机场。若在其中加一些约束，如所有字符的标签只与相邻的字符的标签相关，那么就转化成马尔可夫随机场问题。从马尔可夫随机场到条件随机场就好理解很多，其假设马尔可夫随机场中有 X 和 Y 两种变量，X 一般是给定的，Y 是在给定 X 条件下的输出。在前面的例子中，X 是字符，Y 为标签，P(X|Y) 就是条件随机场。

在条件随机场的定义中，我们并没有要求变量 X 与 Y 具有相同的结构。实际在自然语言处理中，多假设其结构相同，即

$$X = (X_1, X_2, X_3, ..., X_n)，\ Y = (Y_1, Y_2, Y_3, ..., Y_n) \qquad (4.2)$$

结构如图 3.4 所示。

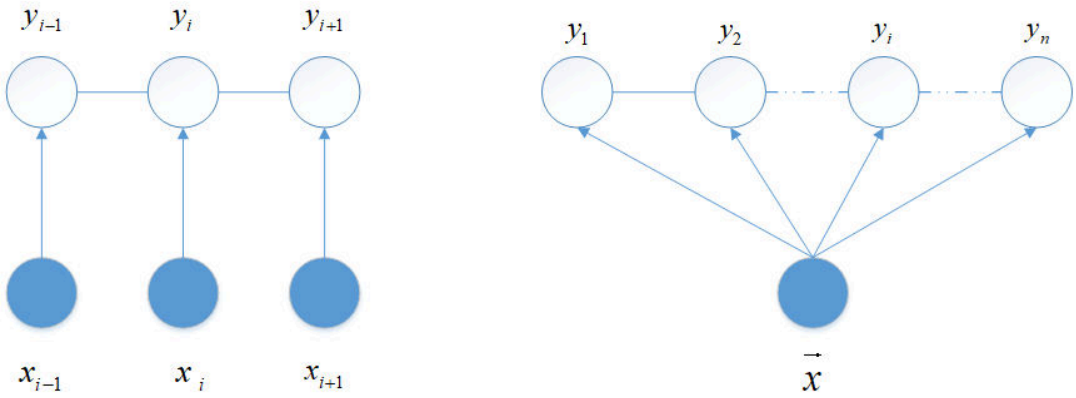


图 3.4 线性链条件随机场

一般将这种结构称为线性链条件随机场(linear-chain Conditional Random Fields，linear-chain CRF)。其定义如下：

设 $X = (X_1, X_2, X_3, ..., X_n)$ 和 $Y = (Y_1, Y_2, Y_3, ..., Y_n)$ 均为线性链表示的随机变量序列，若在给定的随机变量序列 X 的条件下，随机变量序列 Y 的条件概率分布 P(Y|X) 构成条件随机场，且满足马尔可夫性：

$$P(Y_i \mid X, Y_1, Y_2, Y_3, ..., Y_n) = P(Y_i \mid X, Y_{i-1}, Y_{i+1}) \qquad (4.3)$$

则称 P(Y|X)为线性链的条件随机场。（注意，本书中如非特别声明，所说的 CRF 指的就是线性链 CRF。）

细心的读者会发现，相较于 HMM，这里的线性链 CRF 不仅考虑了上一状态Y_{i-1}，还考虑后续的状态结果Y_{i+1}。我们在图 3.5 中对 HMM 和 CRF 做一个对比。

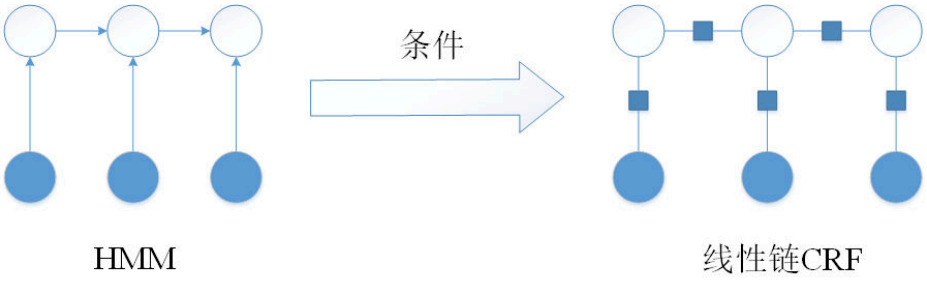


图 3.5 HMM 和线性链 CRF 联系图

在图 3.5 中，可以看到 HMM 是一个有向图，而线性链 CRF 是一个无向图。因此，HMM 处理时，每个状态依赖上一个状态，而线性链 CRF 依赖于当前状态的周围结点状态。

对于线性链 CRF 的算法思想已经介绍不少，接下来讲解如何将其应用于命名实体识别过程中。

仍以地名识别为例，对句子“我来到牛家村”进行标注，正确标注后的结果应为“我/O 来/O 到/O 牛/B 家/M 村/E”。采用线性链 CRF 来进行解决，那么（O，O，O，B，M，E）是其一种标注序列，（O，O，O，B，B，E）也是一种标注选择，类似的可选标注序列有很多，在 NER 任务中就是在如此多的可选标注序列中，找出最靠谱的作为句子的标注。

判断标注序列靠谱与否就是我们要解决的问题。就上面的两种分法，显然第二种没有第一种准确，因为其将“牛”和“家”都作为地名首字标成了“B”，一个地名两个首字符，显然不合理。假如给每个标注序列打分，分值代表标注序列的靠谱程度，越高代表越靠谱，那么可以定一个规则，若在标注中出现连续两个“B”结构的标注序列，则给它低分（如负分、零分等）。

上面说的连续“B”结构打低分就对应一条特征函数。在 CRF 中，定义一个特征函数集合，然后使用这个特征集合为标注序列进行打分，据此选出最靠谱的标注序列。该序列的分值是通过综合考虑特征集合中的函数得出的。

在 CRF 中有两种特征函数，分别为转移函数$t_k(y_{i-1},y_i,i)$和状态函数$s_l(y_i,X,i)$。$t_k(y_{i-1},y_i,i)$依赖于当前和前一个位置，表示从标注序列中位置 i-1 的标记y_{i-1}转移到位置 i 上的标记y_i的概率。$s_l(y_i,X,i)$依赖当前位置，表示标记序列在位置 i 上为标记y_i的概率。通常特征函数取值为 1 或 0，表示符不符合该条规则约束。完整的线性链 CRF 的参数化形式如下：

$$P(y|x)=\frac{1}{Z(x)}\exp\left(\sum_{i,k}\lambda_k t_k(y_{i-1},y_i,i)+\sum_{i,l}\mu_l s_l(y_i,X,i)\right) \quad (4.4)$$

其中

$$Z(x)=\sum_y \exp\left(\sum_{i,k}\lambda_k t_k(y_{i-1},y_i,i)+\sum_{i,l}\mu_l s_l(y_i,X,i)\right) \quad (4.5)$$

Z(x)是规范化因子，其求和操作是在所有可能的输出序μ_l列上做的；λ_k和为转移函数和状态函数对

应的权值。

通常为了方便计算，将式（4.5）简化为下式：

$$P(y \mid x)=\frac{1}{Z(x)}\exp\left(\sum_j \sum_i w_j f_j(y_{i-1}, y_i, x, i)\right) \quad (4.6)$$

对应的 Z(x)表示如下：

$$Z(x)=\sum_y \exp\left(\sum_j \sum_i w_j f_j\left(y_{i-1}, y_i, x, i\right)\right) \quad (4.7)$$

其中，$f_j\left(y_{i-1}, y_i, x, i\right)$为式（4.4）中$t_k\left(y_{i-1}, y_i, i\right)$和$s_l(y_i, X, i)$的统一符号表示。

使用 CRF 来做命名实体识别时，目标是求$\arg\max_y P(y \mid x)$。该问题与 HMM 求解最大可能序列路径一样，也是采用的 Veterbi 算法。Veterbi 算法在前面章节已有介绍，这里就不再赘述了。

当解决标注问题时，HMM 和 CRF 都是不错的选择。然而相较于 HMM，CRF 能够捕捉全局的信息，并能够进行灵活的特征设计，因此一般效果要比 HMM 好不少。当然，也由于此，一般实现起来复杂度会高很多

3.2.3 句法分析

在自然语言处理中，机器翻译是一个重要的课题，也是 NLP 应用的主要领域，而句法分析是机器翻译的核心数据结构。句法分析是自然语言处理的核心技术，是对语言进行深层次理解的基石。句法分析的主要任务是识别出句子所包含的句法成分以及这些成分之间的关系，一般以句法树来表示句法分析的结果。从 20 世纪 50 年代初机器翻译课题被提出时算起，自然语言处理研究已经有 60 余年的历史，句法分析一直是自然语言处理前进的巨大障碍。句法分析主要有以下两个难点：

●歧义。自然语言区别于人工语言的一个重要特点就是它存在大量的歧义现象。人类自身可以依靠大量的先验知识有效地消除各种歧义，而机器由于在知识表示和获取方面存在严重不足，很难像人类那样进行句法消歧。

●搜索空间。句法分析是一个极为复杂的任务，候选树个数随句子增多呈指数级增长，搜索空间巨大。因此，必须设计出合适的解码器，以确保能够在可以容忍的时间内搜索到模型定义最优解。

句法分析（Parsing）是从单词串得到句法结构的过程，而实现该过程的工具或程序被称为句法分析器（Parser）。句法分析的种类很多，这里我们根据其侧重目标将其分为完全句法分析和局部句法分析两种。两者的差别在于，完全句法分析以获取整个句子的句法结构为目的；而局部句法分析只关注于局部的一些成分，例如常用的依存句法分析就是一种局部分析方法。

句法分析中所用方法可以简单地分为基于规则的方法和基于统计的方法两大类。基于规则的方法在处理大规模真实文本时，会存在语法规则覆盖有限、系统可迁移差等缺陷。随着大规模标注树库的建立，基于统计学习模型的句法分析方法开始兴起，句法分析器的性能不断提高，最典型的就是风靡于 20 世纪 70 年代的 PCFG（Probabilistic Context Free Grammar），它在句法分析领域得到了极大的应用，也是现在句法分析中常用的方法。统计句法分析模型本质是一套面向候选树的评价方法，其会给正确的句法树赋予一个较高的分值，而给不合理的句法树赋予一个较低的分值，这样就可以借用候选句法树的分值进行消歧。在本节中，我们将着重于基于统计的句法分析方法（简称统计分析方法）的介绍。

一、句法分析的数据集与评测方法

统计分析方法一般都离不开语料数据集和相应的评价体系的支撑，本节将详细介绍这方面的内容。

1） 句法分析的数据集

统计学习方法多需要语料数据的支撑，统计句法分析也不例外。相较于分词或词性标注，句法分析的数据集要复杂很多，其是一种树形的标注结构，因此又称树库。图 3.6 所示是一个典型的句法树。

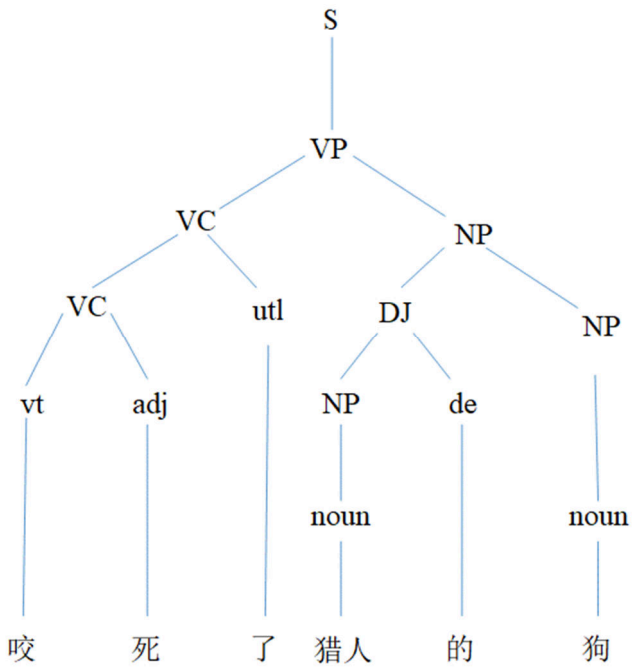


图 3.6 句法树示例

目前使用最多的树库来自美国宾夕法尼亚大学加工的英文宾州树库（Penn TreeBank，PTB）。PTB 的前身为 ATIS（Air Travel Information System）和 WSJ（Wall Street Journa）树库，具有较高的一致性和标注准确率。

中文树库建设较晚，比较著名的有中文宾州树库（Chinese TreeBank，CTB）、清华树库（Tsinghua Chinese TreeBank，TCT）、台湾中研院树库。其中 CTB 是宾夕法尼亚大学标注的汉语句法树库，也是目前绝大多数的中文句法分析研究的基准语料库。TCT 是清华大学计算机系智能技术与系统国家重点实验室人员从汉语平衡语料库中提取出 100 万规模的汉字语料文本，经过自动句法分析和人工校对，形成的高质量的标注有完整句法结构的中文句法树语料库。Sinica TreeBank 是中国台湾中研院词库小组从中研院平衡语料库中抽取句子，经过电脑自动分析成句法树，并加以人工修改、检验后所得的成果。

不同的树库有着不同的标记体系，使用时切忌使用一种树库的句法分析器，然后用其他树库的标记体系来解释。由于树库众多，这里不再讲述具体每一种树库的标记规范，感兴趣的读者可网上搜索自行查阅。图 3.7 所示为清华树库的部分标记集。

序号	标记代码	标记名称	序号	标记代码	标记名称
1	np	名词短语	9	mbar	数词准短语
2	tp	时间短语	10	mp	数量短语
3	sp	空间短语	11	dj	单句句型
4	vp	动词短语	12	fj	复句句型
5	ap	形容词短语	13	zj	整句
6	bp	区别词短语	14	jp	句群
7	dp	副词短语	15	dlc	独立成分
8	pp	介词短语	16	yj	直接引语

图 3.7 清华树库的汉语成分标记集（部分）

2）句法分析的评测方法

句法分析评测的主要任务是评测句法分析器生成的树结构与手工标注的树结构之间的相似程度。其主要考虑两方面的性能：满意度和效率。其中满意度是指测试句法分析器是否适合或胜任某个特定的自然

语言处理任务；而效率主要用于对比句法分析器的运行时间。

目前主流的句法分析评测方法是 PARSEVAL 评测体系，它是一种粒度比较适中、较为理想的评价方法，主要指标有准确率、召回率、交叉括号数。准确率表示分析正确的短语个数在句法分析结果中所占的比例，即分析结果中与标准句法树中相匹配的短语个数占分析结果中所有短语个数的比例。召回率表示分析得到的正确短语个数占标准分析树全部短语个数的比例。交叉括号表示分析得到的某一个短语的覆盖范围与标准句法分析结果的某个短语的覆盖范围存在重叠又不存在包含关系，即构成了一个交叉括号。

二、句法分析的常用方法

相较于词法分析（分词、词性标注或命名实体识别等），句法分析成熟度要低上不少。为此，学者们投入了大量精力进行探索，他们基于不同的语法形式，提出了各种不同的算法。在这些算法中，以短语结构树为目标的句法分析器目前研究得最为彻底，应用也最为广泛，与很多其他形式语法对应的句法分析器都能通过对短语结构语法（特别是上下文无关文法）的改造而得到。因此，本节将主要介绍基于上下文无关文法的句法分析。这里需要强调的是，因为本书是 NLP 入门实践书籍，而句法分析又属于 NLP 中较为高阶的问题，故本节不会深陷算法的细节中。读者了解这些算法即可，重要的是能够在后面的实践环节中使用起来。

1） 基于 PCFG 的句法分析

PCFG（Probabilistic Context Free Grammar）是基于概率的短语结构分析方法，是目前研究最为充分、形式最为简单的统计句法分析模型，也可以认为是规则方法与统计方法的结合。

PCFG 是上下文无关文法的扩展，是一种生成式的方法，其短语结构文法可以表示为一个五元组（X，V，S，R，P）：

●X 是一个有限词汇的集合（词典），它的元素称为词汇或终结符。

●V 是一个有限标注的集合，称为非终结符集合。

●S 称为文法的开始符号，其包含于 V，即 S∈V。

●R 是有序偶对（α，β）的集合，也就是产生的规则集。

●P 代表每个产生规则的统计概率。

PCFG 可以解决以下问题：

●基于 PCFG 可以计算分析树的概率值。

●若一个句子有多个分析树，可以依据概率值对所有的分析树进行排序。

●PCFG 可以用来进行句法排歧，面对多个分析结果选择概率值最大的。

如果把→看作一个运算符，PCFG 可以写成如下的形式：

形式：A→α，P

约束：ΣαP（A→α）

下面根据一个例子来看 PCFG 求解最优句法树的过程。有一个规则集，内容如下：

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → astronomers	0.1
VP → V NP	0.7	NP → ears	0.18
VP → VP PP	0.3	NP → saw	0.04
P → with	1.0	NP → stars	0.18
V → saw	1.0	NP → telescope	0.1

其中第一列表示规则，第二列表示该规则成立的概率。

给定句子 S：astronomers saw stars with ears，得到两个句法树，如图 3.8 所示。

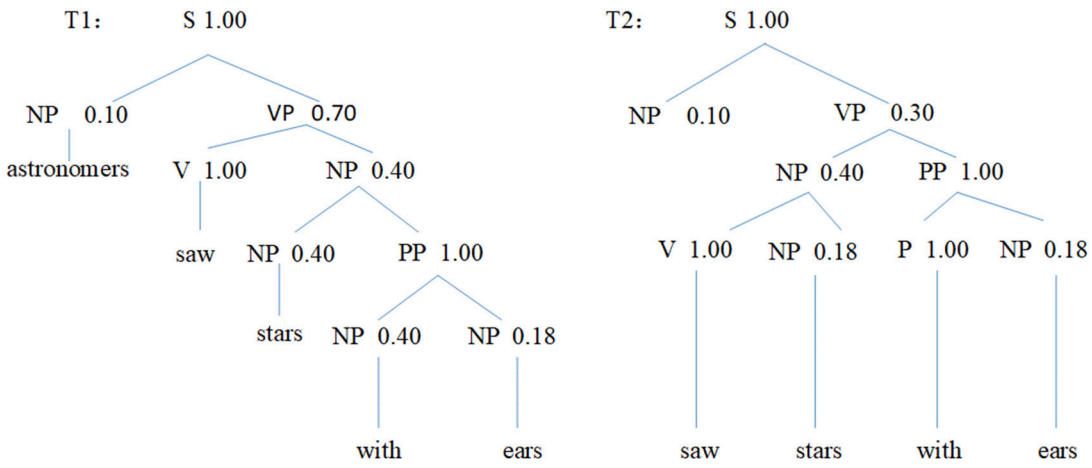


图 3.8 单句的不同句法树示例

计算两棵句法树的概率如下：

P(T1)=S×NP×VP×V×NP×NP×PP×P×NP

=1.0×0.1×0.7×1.0×0.4×0.18×1.0×1.0×0.18

=0.0009072

P(T2)=S×NP×VP×VP×V×NP×PP×P×NP

=1.0×0.1×0.3×0.7×1.0×0.18×1.0×1.0×0.18

=0.0006804

因此选择 T1 作为最终的句法树。

根据上述例子，我们很自然想到关于 PCFG 的三个基本问题。

●给定上下文无关文法 G，如何计算句子 S 的概率，即计算 P(S|G)？

●给定上下文无关文法 G 以及句子 S，如何选择最佳的句法树，即计算 arg maxT P(T/S，G)？

●如何为文法规则选择参数，使得训练句子的概率最大，即计算 arg maxG P(S/G)？

可以使用内向算法和外向算法解决第一个问题，使用 Viterbi 算法解决第二个问题，使用 EM 算法解决第三个问题。

作为目前最成功的基于语法驱动的统计句法分析方法，PCFG 衍生出了各种形式的算法，包括基于单纯 PCFG 的句法分析方法、基于词汇化的 PCFG 的句法分析方法、基于子类划分 PCFG 的句法分析方法等。这些方法各有千秋，使用时可根据具体效果进行甄选。

2）基于最大间隔马尔可夫网络的句法分析

最大间隔是 SVM（支持向量机）中的重要理论，而马尔可夫网络是概率图模型中一种具备一定结构处理关系能力的算法。最大间隔马尔可夫网络（Max-Margin Markov Networks）就是这两者的结合，能够解决复杂的结构化预测问题，尤为适合用于句法分析任务。这是一种判别式的句法分析方法，通过丰富特征来消解分析过程中产生的歧义。其判别函数采用如下形式：

fx(x)=arg maxy∈G(x)〈w，Φ(x，y)〉

其中，Φ(x，y)表示与 x 相对应的句法树 y 的特征向量，w 表示特征权重。

类似 SVM 算法，最大间隔马尔可夫网络要实现多元分类，可以采用多个独立而且可以并行训练的二分类器来代替。这样，每个二分类器识别一个短语标记，通过组合这些分类器就能完成句法分析任务，同时也能通过并行方式，大大提升训练速度。

3）基于 CRF 的句法分析

当将句法分析作为序列标注问题来解决时，同样可以采用条件随机场（CRF）模型。该方法在第 4 章中已做详细介绍，因此这里不再展开，读者将标注序列做相应变换，然后参照上一节实战环节进行即可。

与前面 PCFG 的模型相比，采用 CRF 模型进行句法分析，主要不同点在于概率计算方法和概率归一化

的方式。CRF 模型最大化的是句法树的条件概率值而不是联合概率值，并且对概率进行归一化。

同基于最大间隔马尔可夫网络的句法分析一样，基于 CRF 的句法分析也是一种判别式的方法，需要融合大量的特征。

3.3 自然语言处理应用实例-文本分类

文本分类就是让计算机对一定的文本集合按照一定的标准进行分类。比如，小李是个足球迷，喜欢看足球类的新闻，新闻推荐系统使用文本分类技术为小李自动推荐足球类的新闻。文本分类程序把一个未见过的文档分成已知类别中的一个或多个，例如把新闻分成国内新闻和国际新闻。利用文本分类技术可以对网页分类，也可以用于为用户提供个性化新闻或者垃圾邮件过滤。

把给定的文档归到两个类别中的一个叫作两类分类，例如垃圾邮件过滤，就只需要确定“是”还是“不是”垃圾邮件。分到多个类别中的一个叫作多类分类，例如中图法分类目录把图书分成22个基本大类。
文本分类主要分为训练阶段和预测阶段。训练阶段得到分类的依据，也叫作分类模型。预测阶段根据分类模型对新文本分类。训练阶段一般先分词，然后提取能够作为分类依据的特征词，最后把分类特征词和相关的分类参数写入模型文件。提取特征词这个步骤叫作特征提取。

例如要把新闻分为四类{政治,体育,商业,艺术}。首先准备好训练文本集，也就是一些已经分好类的文本。每个类别路径下包含属于该类别的一些文本文件。

3.3.1 情感文档分类

从本节开始，我们将讨论当前主要的研究方向或主题及其核心技术。情感分类可能是研究最广泛的话题（Pang和Lee，2008）。它旨在将意见文档分类为表达积极或消极的意见或情感。该任务通常也称为文档级情感分类，因为它将整个文档视为基本信息单元。关于该主题的绝大多数研究都是针对在线评论进行分类。因此，我们还在评论上下文中定义了问题，但该定义也适用于其他类似的上下文。

问题的定义:对于给定评估实体的意见文档d，需要确定意见持有者关于实体的整体情感s，即确定在五元组中方面(Aspect)的GENERAL上表达的s，即(_，GENERAL，s，_，_)，其中实体e、意见持有人h和意见时间t假定已知或不相关。根据s所采用的数值(Value）类型，这里有两种表述。如果s采用分类值，例如正数和负数，它就是分类问题;如果s采用给定范围内的数值或序数数值，例如1-5，问题就变为回归。为了确保相关任务在实践中更具有意义，有研究人员做了以下隐含假设（Liu，2010）。

情感分类或回归的假设是:意见文档d(例如产品评论）表达了对单个实体e的意见并且包含来自单个意见持有者h的意见。在实践中，如果一个意见文档评估了不止一个实体，那么实体上所
表达的情绪可以是不同的。例如,意见持有人可能对某些实体持肯定态度,对其他实体持否定态度。因此，在这种情况下，将一个情感取向分配给整个文档是没有实际意义的。多个意见持有者在单个文档中表达意见也没有多大意义，因为他们的意见也可能不同。

此假设适用于产品和服务的评论，因为每个评论通常侧重于评估单个产品或服务，并且由单个评论者所写。然而，该假设可能不适用于论坛和博客贴文，因为在这样的贴文中，作者可以表达对多个实体的意见并使用比较句子来比较它们。

3.3.2 句子主观性与情感分类

文档级情感分类对于大多数应用来说可能是过于粗糙的。我们现在转到句子级别，即对每个句子中所表达的情感进行分类。但是，文档和句子级别的分类之间没有根本性的区别，因为句子其实只是一个简短的文档。研究人员经常对句子级分析做出的一个假设是，句子通常包含单一意见。文档通常包含多个意见。下面以示例评论进行讨论。

“两周前我买了一部摩托罗拉手机。最初一切都很好，声音很清晰，电池寿命很长，只是它有点笨重。但是，它昨天停止了工作。”

第一句话没有表达任何意见，因为它只是陈述了一个事实。而所有其他句子则表达了明示或暗示的情感。注意一点，通常认为没有意见是中立的。

问题定义:给定句子x，确定x是表示正面、负面或中立（或没有）意见。

这里不使用五元组（e，a，s，h，t）定义，因为句子级别分类是一个中间步骤。在大多数应用中，需要了解意见目标。只知道一个句子表达了积极或消极的意见，而不是意见所涉及的实体/方面（Entity/Aspect），其用途是有限的。然而，句子级别分类仍然有用，因为在许多情况下，如果我们知道在一个句子中谈论了哪些实体和实体的方面,这一步骤可以帮助我们确定关于实体及其方面的意见是积极的还是消极的。

句子情感分类既可以作为三类分类问题来解决，也可以作为两个独立的分类问题来解决。在后一种情况下，第一个问题（也称为第一步）是对句子是否表达意见进行分类。第二个问题（也称为第二步）将这些意见句子归类为正面和负面类型。第一个问题通常称为主观性分类，它确定一个句子是表达主观信息还是事实（客观）信息（Hatzivassiloglou和Wiebe，2000;Riloff等等，2006;Riloff和Wiebe，2003，Wiebe等，2004:Wilson等等，2004和2006;Yu和Hatzivassiloglou，2003）。客观句子被视为不表达任何情感或意见。这是我们前面讨论过的问题，客观句子也可以暗示意见，例如，在上述评论中，“但是，它昨天停止了工作”是一个客观的句子，但它暗示了对手机的负面情绪，因为这是不受欢迎的事实。因此，第一步更适合将每个句子分类为评论者是否有自己的意见，无论是主观的还是客观的。

3.3.3 垃圾评论检测

随着社交媒体的快速发展，来自网络上的评论意见越来越多地影响着组织或个人进行购买决策的制定、投票和市场产品设计等事宜。对于公司和个人而言，正面的评论意见常常意味着更高的利润和更好的口碑。有研究表明:在Yelp.com网站上，如果增加半颗星评分，就能帮餐馆提升约19%的销量（Anderson和Magruder，2012），平均评分若增长一颗星,将带来5%~9%的利润增长（Luca，2011）。由于商人对于利润和市场规模的追逐,网络媒体上出现了越来越多的虚假评论或虚假意见。通过发布这种虚假的意见或评论来达到推销或诋毁一些目标产品、服务、组织或个人的目的。这样的个人或组织称为垃圾评论发布者（Opinion Spammer），他们的活动被称为垃圾评论发布（Opinion Spamming）。

目前，垃圾评论已经十分普遍。例如，有调查研究表明，在Yelp.com网站上有接近25%的评论是欺骗性的评论（http://www.bbc.com/news/technology-24299742）。相比于商业评论，针对社交和政治事件发布垃圾评论的活动具有更大的危害性，它们可能会利用歪曲的意见或观点来将民众调动到法律和道德的对立面上。因此，为了使得社交媒体继续作为公开意见或观点的可信来源，而不是充斥着越来越多的欺骗、谎言与伪造观点或意见,垃圾评论检测有着极其重要的意义和研究价值。

个人和组织越来越多地使用社交媒体的意见来做出购买决定，并在营销、产品设计、产品或服务投诉等方面做出自己的选择。积极的意见通常意味着企业和个人的利益和名誉得以维系，而不幸的是，通过发布虚假的意见或评论来促进或诋毁某些目标产品、服务、组织、个人，实际上就刺激了大家围观和加入其中的冲动。这些人被称为垃圾评论发送者，他们的活动被称为垃圾评论（Jindal和Liu，2007，2008）。关于社会和政治舆论的垃圾评论甚至可能令人恐惧，因为它们可以歪曲意见并煽动群众反对法律或道德规范。可以肯定地说,随着社交媒体中的意见越来越多地被用于实践，垃圾评论将变得越来越猖獗和复杂，这样一来，对于它们的检测将面临更大的挑战。但是，它们必须被检测，以确保社交媒体继续成为公众舆论的可靠来源，而不是充满虚假的观点、虚假的意见、谎言和欺骗。

研究人员已经在许多领域进行了垃圾评论检测的研究工作，而研究最多的两类是Web垃圾网页信息和垃圾邮件。然而，垃圾评论却大不相同。有两种主要类型的Web垃圾，即垃圾链接和垃圾内容（Castillo和Davison，2010:Liu，2006和2011）。垃圾链接是超链接上的垃圾信息，在评论中几乎不存在。虽然广告链接在其他形式的社交媒体中很常见，但它们相对容易被发现。垃圾内容会在目标网页中添加流行（但不相关）的关键词，以欺骗搜索引擎使其与许多搜索查询相关，但是这种情况很少出现在评论文章中。而垃圾邮件是指未经请求的广告,这在在线网络意见中很少见。

垃圾评论检测的主要挑战是，与其他形式的垃圾评论不同，通过人工阅读来识别虚假意见是非常困难的，这很难找到垃圾评论数据来帮助设计和评估检测算法。对于其他形式的垃圾评论，人们可以相当容易地识别它们。

事实上，在极端情况下，仅仅通过阅读垃圾评论在逻辑上是不可能识别的。例如，人们可以为一家好餐馆写一份真实的评论，然后将其发布为对坏餐馆的虚假评论，以便推广它。如果不考虑评论文本本身之外的信息，就无法检测到这种虚假评论，因为不能同时为同一评论给出真实和虚假的两个标签。

3.3.4 垃圾评论类型

Jindal和Liu(2008）经过分析指出，商业网站上目前主要有以下三种类型的垃圾评论。

类型1(虚假评论）:这些评论都是评论者对真实产品或服务没有实际使用经历和体验的非真实评论。其中蕴含着一些隐藏的动机。通常情况下，他们会针对目标实体（产品或服务）发表不应得的正面评价，以促销实体（产品或服务）；或用不公正的虚假负面评论来诋毁目标实体的声誉。

类型2(仅关于品牌的评论）:这些评论没有如期望地评论具体的产品或服务，而是对品牌或产品的制造商发表评论。虽然评论可能是真实的，但因为它们没有针对具体的产品，而且往往是有偏见的，因而被视为垃圾评论。例如，一个关于特定型号的H打印机评论说:“我讨厌HP，我从来不买他们的任何产品”。

类型3(非评论文本):这类垃圾评论往往都不是评论，其包含两种类型:(1)广告;(2)不含观点的（例如问题、答案和随机文本）不相关文本。严格地说，它们不被当作评论，因为它们没有提供用户对于产品和服务的意见和评论。

3.3.5 利用TensorFlow实现中文情感分类

这里我们使用谭松波的酒店评论语料进行酒店正面和负面评论的情感分析。

1．训练语料

训练样本分别放置在两个文件夹里: pos和 neg，每个文件夹里有2000个TXT文件，每个文件内有一段评语，共有4000个训练样本，这样大小的样本数据在自然语言处理(NLP)中属于非常迷你类型的。

2．分词和切词

首先我们去掉每个样本的标点符号，然后用Jieba（结巴，中文分词的工具包之一)分词，Jieba分词返回一个生成器,没法直接进行切分词(Tokenize)，所以我们将分词结果转换成一个列表(List),并将它索引化，这样每一例评价的文本变成一段索引数字，对应着预训练词向量模型中的词。

```
    # 进行分词和 tokenize
    # train_tokens 是一个长长的 list，其中含有 4000 个小 list，对应每一条评价
    train_tokens = []
    for text in train_texts_orig:
        # 去掉标点
        text = re.sub("[\s+\.\!\/_,$%^*(+\"\']+|[+——！，。？、~@#￥%……&*（）]+",
"",text)
        # 结巴分词
        cut = jieba.cut(text)
        # 结巴分词的输出结果为一个生成器
        # 把生成器转换为 list
        cut_list = [ i for i in cut ]
        for i, word in enumerate(cut_list):
            try:
                # 将词转换为索引 index
                cut_list[i] = cn_model.vocab[word].index
            except KeyError:
                # 如果词不在字典中，则输出 0
                cut_list[i] = 0
```

3．索引长度标准化

因为每段评语的长度是不一样的，我们如果单纯取最长的一个评语，并把其他评语填充成同样的长度，就

```
# 获得所有 tokens 的长度
num_tokens = [ len(tokens) for tokens in train_tokens ]
num_tokens = np.array(num_tokens)
# 取 tokens 平均值并加上两个 tokens 的标准差，
# 假设 tokens 长度的分布为正态分布，则 max_tokens 这个值可以涵盖 95%左右的样本
max_tokens = np.mean(num_tokens) + 2 * np.std(num_tokens)
max_tokens = int(max_tokens)
```

会十分浪费计算资源，所以我们取一个折中的长度。

4．反向切分词

我们定义一个函数，用来把索引转换成可阅读的文本，这对于程序调试(Debug)很重要。

```
# 用来将 tokens 转换为文本
def reverse_tokens(tokens):
    text = ''
    for i in tokens:
        if i != 0:
            text = text + cn_model.index2word[i]
        else:
            text = text + ' '
    return text

reverse = reverse_tokens(train_tokens[0])
```

5．准备词向量矩阵

现在我们来为模型准备词向量矩阵（Embedding Matrix）。根据Keras的要求，我们需要准备一个维度为(numwords, embeddingdim)的矩阵, numwords代表我们使用的词汇的数量， emdeddingdim在我们使用的预训练词向量模型中是300，每一个词汇都用一个长度为300的向量表示。

注意，我们只选择使用前50 000个使用频率最高的词，在这个预训练词向量模型中，一共有2600 000个词汇，全部使用在分类问题上会很浪费计算资源，因为我们的训练样本很小，一共只有4000个，如果有100 000个、200 000个甚至更多的训练样本，在分类问题上可以考虑减少使用的词汇量。

6．填充和截短

我们把文本转换为短语索引之后，每一串索引的长度并不相等，为了方便模型的训练，我们需要把索引的长度标准化。前面我们选择了236这个可以涵盖95%训练样本的长度，接下来进行填充(Padding）和截短（Truncating)，我们一般采用'pre'的方法，这会在文本索引的前面填充0，根据一些研究资料中的实践，如果在文本索引后面填充0，就会对模型造成一些不良影响。

7．构建模型

在这个教程中尝试了几种神经网络结构，因为训练样本比较少，所以我们可以尽情尝试，训练过程等待时间并不长。

GRU:如果使用GRU，测试样本可以达到87%的准确率，但作者测试自己的文本内容时发现，GRU最后一层激活函数的输出都在0.5左右，说明模型的判断不是很明确，信心比较低，而且经过测试发现模型对于否定句的判断有时会失误，我们期望对于负面样本输出接近0，正面样本接近1，而不是都徘徊于0.5之间。

BiLSTM:测试了LSTM 和BiLSTM，发现BiLSTM的表现最好，LSTM的表现略好于GRU，这可能是因为BiLSTM对于比较长的句子结构有更好的记忆，有兴趣的朋友可以深入研究一下。

向量化之后，第一层我们用BiLSTM返回序列(Sequence），然后第二层16个单元的LSTM不返回序列(Sequence)，只返回最终结果，最后是一个全链接层，用sigmoid激活函数输出结果。

```
model.add(Dense(1, activation='sigmoid'))
# 我们使用 adam 以 0.001 的 learning rate 进行优化
optimizer = Adam(lr=1e-3)

model.compile(loss='binary_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
# 定义 early stoping 如果 3 个 epoch 内 validation loss 没有改善则停止训练
earlystopping = EarlyStopping(monitor='val_loss', patience=3, verbose=1)
# 自动降低 learning rate
lr_reduction = ReduceLROnPlateau(monitor='val_loss',
                                       factor=0.1, min_lr=1e-5, patience=0,
                                       verbose=1)
# 开始训练
model.fit(X_train, y_train,
          validation_split=0.1,
          epochs=20,
          batch_size=128,
          callbacks=callbacks)
```

8．实验结果

首先对测试样本进行预测，得到了还算满意的准确度。

之后定义一个预测函数来预测输入的文本的极性（正面，负面或中性），可见模型对于否定句和一些简单的逻辑结构都可以进行准确的判断，预测结果如下所示。

酒店设施不是新的，服务态度很不好

是一例负面评价 output=0.14

酒店卫生条件非常不好

是一例负面评价 output=0.09

床铺非常舒适

是一例正面评价 output=0.76

房间很凉，不给开暖气

是一例负面评价 output=0.17

房间很凉爽，空调冷气很足

是一例正面评价 output=0.66

酒店环境不好，住宿体验很不好

是一例负面评价 output=0.06

房间隔音不到位

是一例负面评价 output=0.17

晚上回来发现没有打扫卫生

是一例负面评价 output=0.25

因为过节所以要我临时加钱，比团购的价格贵

是一例负面评价 output=0.06

```
def predict_sentiment(text):
    print(text)
    # 去标点
    text = re.sub("[\s+\.\!\/_,$%^*(+\"\']+|[+——！，。？、~@#￥%……&*（）]+",
"",text)
    # 分词
    cut = jieba.cut(text)
    cut_list = [ i for i in cut ]
    # tokenize
    for i, word in enumerate(cut_list):
        try:
            cut_list[i] = cn_model.vocab[word].index
        except KeyError:
            cut_list[i] = 0
    # padding
    tokens_pad = pad_sequences([cut_list], maxlen=max_tokens,
                           padding='pre', truncating='pre')
    # 预测
    result = model.predict(x=tokens_pad)
    coef = result[0][0]
    if coef >= 0.5:
        print('是一例正面评价','output=%.2f'%coef)
    else:
        print('是一例负面评价','output=%.2f'%coef)
```

经过查看，发现被错误分类的文本的含义大多比较含糊，即使是人类也不容易判断其极性，如下面索引(index）为 101 的这个句子，好像没有一点满意的成分，但这个评价在训练样本中被标注成正面评价，而我们的模型做出的负面评价的预测似乎是合理的。

```
# 我们来找出错误分类的样本看看
idx=101
print(reverse_tokens(X_test[idx]))
print('预测的分类', y_pred[idx])
print('实际的分类', y_actual[idx])
```

代码运行后的输出结果如下：

```
由于 2007 年 有一些新问题可能还没来得及解决我因为工作需要经常要住那里所以慎重的提出以 下 ： 1 后
的 淋浴喷头的位置都太高我换了房间还是一样很不好用 2 后的一些管理和服务还很不到位尤其是前台入住
和 时代效率太低每次 都超过 10 分钟好像不符合 宾馆的要求
预测的分类 0
实际的分类 1.0
```

3.4 自然语言处理实例-问答系统

问答（Question Answering，QA）是自然语言处理中一项具有挑战性的任务。近年来，随着深度学习在语义和句法分析、机器翻译、关系抽取等自然语言处理任务上取得了显著的成功，利用深度学习来完

成问答任务也得到了越来越多的关注。本章将简要介绍深度学习方法在两个典型问答任务上的最新进展的情况。(1) 基于知识库深度学习的问答（KBQA），主要采用深度神经网络来理解问题的含义，并尝试将其转化为结构化查询，或者直接转化为分布式语义表示。(2) 机器理解中的深度学习(Machine Comprehension，MC) 是一种基于新型神经网络的端到端范式，用于直接计算问题、答案和给定段落之间的深层语义匹配。

3.4.1 问答系统概述

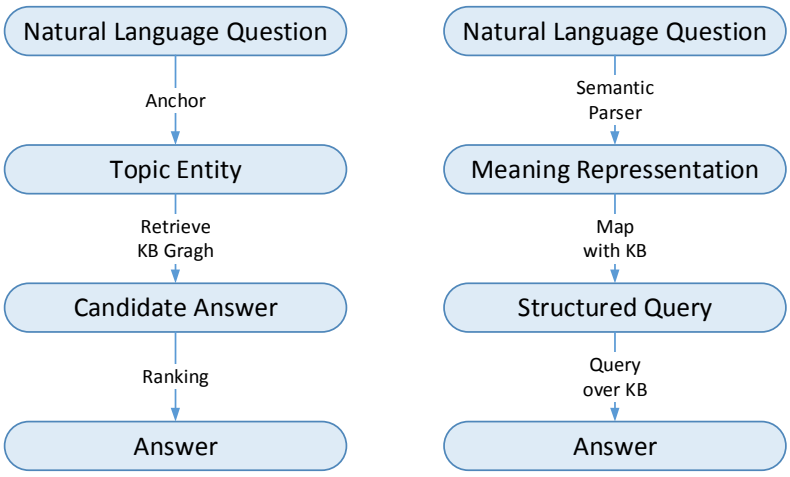
从简单的文档检索到自然语言问答(QA)，网络搜索正处于深刻变革的前沿中。它需要准确理解用户在自然语言问题方面的含义，从网络上各种信息中提取有用的事实，并选择出合理的答案。与其他自然语言处理(NLP)任务，如词性标注、解析和机器翻译类似，大多数传统的QA方法都是基于符号表示的。在这样的范式中，问题和答案中的所有元素，包括单词、短语、句子、文档等，通常都是借助于NLP基本模块来处理的，然后转换为某些结构化或非结构化格式，如词袋、解析树、逻辑形式等。在给定的文档或网页中，计算问题和候选答案之间语义的相似性或相关性，并且具有最高分数的候选项便是最终答案。尽管如此, 这种范式的弱点在于所谓的“语义鸿沟”(Semantic Gap)，即具有相似含义的文本跨度可能会用不同的符号表示。

在神经网络模型中，文本通常表示为分布式向量，而文本跨度之间的精确匹配可以由分布式向量之间的运算来实现。这样传统方法中的语义鸿沟问题就可以在一定程度上得到缓解。

此外，问答任务中还有几个分支，包括基于检索的问答(IRQA)、社区的问答(CQA)、基于知识库的问答(KBQA)和机器理解(MC)的问答。在这里，我们主要关注基于知识库的问答(KBQA)和基于机器理解(MC)的问答，因为这两个问答需要对文本从问题到文档进行更多的语义分析和理解。另外，我们还将从两个角度讨论KBQA的最新进展，并进一步回顾针对机器理解(MC)的深度学习工作等。

3.4.2 基于知识库的问答

截至目前，其实已经有研究人员在扩展新的神经网络模型方面进行了许多成功的尝试，并且提高了知识库问答系统的性能。这里有多种新型的神经网络组件或架构及其变体，如CNN、RNN(LSTM、BLSTM)、注意力机制和记忆网络，并且已经在任务中得到了检验。这些工作内容可以被分为两种模式: 信息抽取(Information Extraction，IE）和语义分析(Semantic Parsing，SP），如图3.9所示。信息抽取模式通常使用各种关系提取技术从知识库中检索一组候选答案，然后将这些候选答案与压缩特征空间中的问题进行比较;语义分析模式则是借助新型的组件或神经网络结构，从句子中提取出正式的/符号化的表示或结构化的查询。



(a) 信息抽取模式　　(b) 语义分析模式

图 3.9 信息抽取和语义分析模式

从另一个角度来看，我们可以将有关利用深度学习方法来促进基于知识库的问答（KBQA)开发工作分为两类:在传统的KBQA框架内使用新型的神经网络模型来改进特定的组件和在统一的神经网络架构中标准化工作任务。前一种观点主要侧重于利用先进的神经网络模型来改进现有的组件，如特征提取、关系识别、语义匹配或相似度计算等;后者则是强调利用新型的深度学习框架来使自然语言的问题和候选答案被构建在同一低维语义空间内。因此，该KBQA任务可以转换为问题的向量和该空间中候选答案之间相似度的计算问题，通常以信息抽取的方式展开。

3.4.3 信息抽取

我们在使用深度学习方法的过程中，主要是侧重于寻找更好的方法将自然语言的问题和来自知识库的候选答案嵌入同一个且被压缩的语义空间中。一般情况下，这些工作会在统一的神经网络架构中以“检索-嵌入-比较”（Retrieval-Embedding-Comparing）流水线式（有时也称为管道）的形式对解决方案进行标准化。

1. 基本介绍

信息抽取(Information Extraction，E）是把文本中包含的信息进行结构化处理，变成表格一样的组织形式。输入信息抽取系统的是原始文本，输出的是固定格式的信息点。信息点首先从各种各样的文档中被抽取出来，然后以统一的形式集成在一起，这就是信息抽取的主要任务。信息以统一的形式集成在一起的好处是方便检索和比较。信息抽取技术并不试图全面理解整篇文档，只是对文档中包含相关信息的部分进行分析，至于哪些信息是相关的,将由系统设计时指定的范围来决定。

信息抽取技术对于从大量的文档中抽取需要的特定事实来说是非常有用的,互联网就是这么一个文档库。在互联网上，同一主题的信息通常分散存放在不同的网站上，表现形式也各不相同，如果能将这些信息收集在一起，并利用结构化形式存储，那将是非常有益的。

由于互联网上的信息载体主要是文本,所以信息抽取技术对于那些把互联网当成知识来源的人来说是至关重要的。信息抽取系统可以看作是把信息从不同文档中转换成数据库记录的系统，成功的信息抽取系统将把互联网变成巨大的数据库。

虽然信息抽取技术是近年来发展起来的新技术领域，但也面临着许多新的挑战。信息抽取原来的目标是从自然语言文档中找到特定的信息，是自然语言处理领域特别有用的一个子领域。它所开发的信息抽取系统既能处理含有表格信息的结构化文本，又能处理自由式文本(如新闻报道)。信息抽取系统中的关键组成部分是一系列的抽取规则或模式，其作用是确定需要抽取的信息。互联网上文本信息的大量增加导致这方面的研究得到高度重视。

2. 向量表述

Bordes等人于2014年率先提出了信息抽取模式的方法，他们没有单独将类别、实体引用和关系模式映射到知识库中对应的类型、实体和谓词，而是提出了一种更直接的方法。他们设计了一个联合嵌入框架来学习结构化知识库中的单词、实体、关系和其他语义项的向量表示，并设法将自然语言问题映射到知识库中的子图，如图3.10所示。

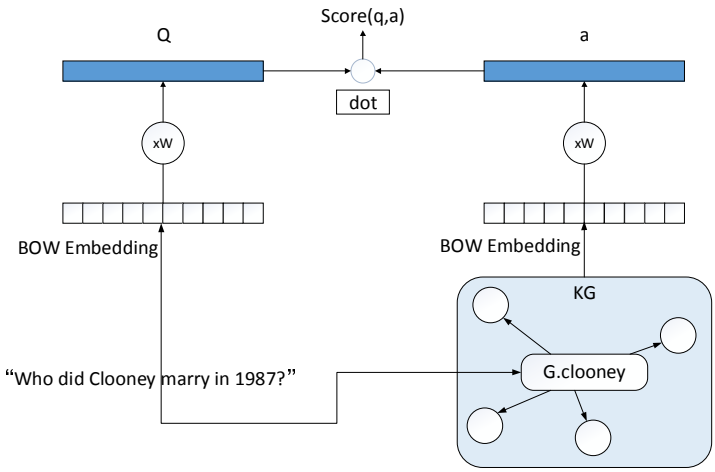


图 3.10 向量表示的示意图

当自然语言问题和候选子图用低维向量表示时,可以很容易地计算出问题和子图之间的相似性。该模型需要带注释的问答配对作为训练数据，也可以通过简单的模式和多任务范式自动收集更多的训练实例。通过同时优化其他资源或相关的辅助任务，如转述任务，该模型旨在确保相似的话语具有更高的相似性，从而减轻客服中心对人力的需求。

图12-2所示的框架同样也遵循简单明了的流水线式（有时也叫管道）结构，即检索-嵌入-比较，而不依赖于人工构造的特征、额外的句法分析或传统抽取模型所采用的经验规则那样，并在基准数据集上实现了具有竞争力的性能。

为了便于实现，首先用词袋模式来表示，然后经过一个压缩过程，这个过程忽略了自然语言问题中的句法结构。类似的方法也适用于候选答案，其中子图简单地表示为其所涉及实体和关系的Multi-Hot 形式。这种简化的方法其实是防止模型在自然语言表示或知识库本身使用更多的信息来源，例如问题中的关系短语或答案类型指示器，或者知识库中的实体谓词一致性。

此外，目前神经网络模型的处理方法还不能很好地处理语义的复合性，以及除了词袋或实体袋关系表示之外的各种约束，如“小明的父亲的母亲的儿子”和“小明的母亲的父亲的儿子”。而在某种程度上，可以通过对问题进行更深入的语法分析，或者频繁地对知识库进行结构挖掘来解决。

3．使用CNN嵌入特征

Yih等人（2014）提出的是利用卷积神经网络（CNN）来讲解决单一关系问题，这与 Bordes 等人在词袋模型里处理所有问题的模式是不同的。在Yih等人的文章中，实际上是使用基于CNN的语义模型（CNNSM）构造两个不同的映射模型:一个用于标识问题中的实体;另一个用于将关系映射到知识库的关系。这里需要说明的是，假设目标问题只包含一个实体和一个关系，这确实占据了各种KBQA（基于知识库的问答）基准数据集的很大比例，而这类问题的结构化查询相对比较简单，只涉及一个<主语、谓语、宾语>三元组。因此，不需要一个结构预测过程来恢复多个实体和关系之间固有的查询结构。

其实这里的关键思想与Bordes等人提出的思想非常类似，即自然语言问题中表达的关系模式和结构化知识库中的关系/谓词可以通过CNN投射到相同低维度的语义空间中。同样，知识库中的实体表现形式与问题中提到的实体相同，并且可以由CNN 捕获到。因此，CNNSM可以提供自然语言问题和知识库中的候选三元组之间的相似性，并选择得分最高的一个作为最终答案。

这种解决方案得益于卷积神经网络模型，其优于简单的词袋模式，并且在一定程度上以Letter-Trigrams向量作为输入来处理词汇表外(Out-Of-Vocabulary，OOV)问题。但是，这也让我们想起了KBQA任务中的两个重要问题:实体链接和关系识别，它们本身都具有足够的挑战性，并且需要足够的训练数据，即“分类-实体”对和“自然语言模式-知识库关系”对来训练模型。特别是目前大型知识库中存在大量的实体和关系，如Freebase，使得处理多个实体和关系的问题变得更加具有挑战性。

另一方面，Dong 等等人(2015）提出使用CNN对问题和候选答案之间的不同类型特征进行编码。他们提出利用多列卷积神经网络（Multicolumn Convolutional Neural Network，MCCNN）模型来捕捉一个问题的不同方面，并通过三个渠道（答案路径、答案语境和答案类型）进一步评分一对问答。

与简单的向量表示（Bordes等，2014）相比，MCCNN使用CNN抽取不同的特征，这些特征可以显式地捕获问题中的主题实体与知识库上的候选答案之间的路径以及期望的答案类型,这两点在评估一个候选答案时显得尤为重要。通过向神经网络中添加所需的列，MCCNN还可以轻松扩展更多类型的功能。

对于基于特征的模型，实体链接仍然是一个悬而未决的问题。应答路径的编码有助于MCCNN在某种程度上沿路径执行浅层推理，然而由于典型的检索-嵌入-比较（Retrieval-Embedding-Comparing）框架的性质，MCCNN仍然无法找到更好的解决方案来处理候选答案之间的比较。

4．利用注意力机制嵌入特征

Hao等人（2017年）采用了双向RNN模型来捕获给定问题的语义。他们认为，一个问题应该根据不同答案不同方面的关注点（答案方面可以是答案实体本身、答案类型、答案语境等）以不同的方式表示。以“谁是百度公司董事长?”为题，并将其候选答案之一“李彦宏”作为一个例子。在处理答案实体“李彦宏”时，问题中的“董事长“和“百度公司”两个词更加集中，问题表征应偏向于这两个词。而当面对答案类型是“公司”或“董事会成员”时，“谁”则应该是最突出的词。同时，有些问题可能更看重答案类型而不是其他方面。在其他问题中，答案的关系可能是我们应该考虑的重要信息，对于不同的问题和答案，它是动态和灵活的。显然，这需要一种注意力机制，它揭示了问题表征与相应答案方面之间的相互影响。

在处理答案不同方面（包括答案路径、答案语境和答案类型）时，Hao等人提出了一种基于交叉注意力的神经网络来执行KBQA，而不是使用具有不同参数的三个CNN来表示问题。

交叉注意力模型代表问题和答案方面之间的相互关注，包含两个部分:答案-问题注意部分和问题-答案注意力部分，前者有助于学习灵活、充分的问题表示，后者有助于调整问答权重值。最后计算问题与不同方面的每个对应候选答案之间的相似度得分，并根据相应的问题-答案权重值将每个候选项的最终得分组合在一起，选择得分最高的候选项作为最终答案。

5．利用记忆机制回答问题

记忆网络是一种新型的学习框架，它是围绕一种可以在特定任务中读取和修改/附加记忆机制而设计的(Weston等人，2015）。在记忆网络范式下，基于知识库问答任务的研究已经有了一些尝试，主要是遵循信息抽取模式的检索比较方式。

Bordes等人于2015年进行了第一次尝试，主要侧重于简单问题的讨论，问题可以用一个〈主体、关系、对象〉(〈subject、relation、object〉)三元组来回答。在输入组件中，我们以bag-of-symbols(符号袋）的形式读取结构化的知识库并将其存储在记忆中，并且问题被处理成bag-of-ngrams 的形式。然后在输出组件中,将bag-of-ngrams化的问题与记忆中的条目进行比较以找到候选三元组，并用输入的问题进行求值（得分）。其中，得分最高的三元组的对象将由响应组件提供且作为模型给出答案。这应该是KBQA任务中记忆网络的一种简单应用，实际上显示了记忆网络在管理大量知识库(KB)条目方面的潜力，甚至在管理来自多个资源的大量知识库条目时也是如此，如图3.11所示。

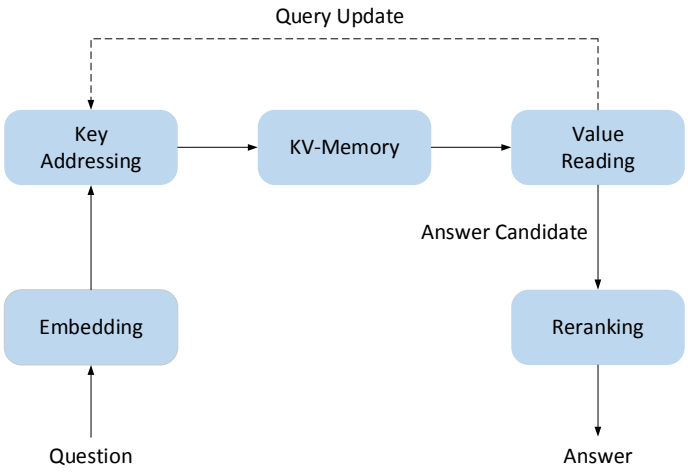


图 3.11 键-值（Key-Value）记忆网络的示意图（Miller 等，2016）

米勒等人(Miller，2016）通过研究记忆机制中各种形式的键-值（Key-Value）的知识库进一步扩展了这一思想，改进后的模型还允许从记忆中读取多个地址来收集线索/上下文，从而动态更新问题以获得最终答案。键-值（Key-Value）设计的优点是使记忆机制更加灵活地存储各种知识，从知识库三元组（主体+关系作为键Key，对象作为值Value）到文档（句子或单词窗口作为键或值），它支持用异构资源来回答更复杂的问题。

3.4.4 语义分析模式

语义分析的思路是通过对自然语言进行语义上的分析，转化成为一种能够让知识库“看懂”的语义表示，进而对知识库中的知识进行推理（Inference）、查询（Query）得出最终的答案。简而言之，语义解析要做的事情就是将自然语言的问题转化为一种能够让知识库“看懂”的语义表示，即逻辑形式(Logic Form)。

“检索-嵌入-比较”框架其实是受益于各种神经网络组件来捕获问答相似度的，并且在简单的问题中表现得更好，其中的实体和关系位于知识库中简单化的子图中。但是，它们并不善于解决复杂的语义组合，因为在理解问题时并没有明确的信息抽取机制来捕捉这种组合。相比之下，KBQA中的其他主流工作(语义分析模式模型）在尝试正式地表示问题的含义，然后使用知识库进行实例化以构建基于知识库的结构化查询，从而可以显式地捕获复杂的查询。

因此，这些模型的核心组件是从自然语言问题中恢复形式意义的表示，如逻辑形式或结构化查询，并通过将表示映射到知识库组件并查询知识库，从而在知识库中找到答案。深度学习方法的工作主要是为了改进框架的某些组件。

也可以将CNNSM模型(Yih等，2014)看作是一种语义分析模式的方法，它只能产生一个<subject、predicate、object>三元组作为查询，其中CNN用于执行实体链接和关系识别，但不适合用于稍微复杂的问题，如涉及多个实体和关系，以及约束限制方面。主要是因为神经网络组件只负责与知识库组件(实体或关系)之间的映射，而没有明确的机制来识别多个实体或关系之间的内在结构。事实上，在传统的基于语义分析模型中，已经通过PCCG、PCFG、依赖结构或其他句法、语义分析范式对这些结构进行了深入研究。

1. STAGG:搜索和剪枝时的语义分析

除了单一关系问题，Yih等人（2015）建议使用查询图的方式来表示问题的含义，其中包含4种节点:基础实体（Grounded Entity，标准实体）、存在变量(Existential Variable)、入变量和约束/函数。在这里，入变量是非标准的实体，并且有望成为最终的答案;存在变量可以指代(引用)中间节点，如“小明的父亲的母亲”中的“父亲”的表达，或者抽象节点，如Freebase中的复合值类型(CVT)节点，并且约束或函数被设计为根据某些数值属性（如argmin）过滤一组实体。在查询图中，节点之间通过有向边连接，表示两个节点之间的关系期望通过知识库谓词进行映射，如图3.12所示。

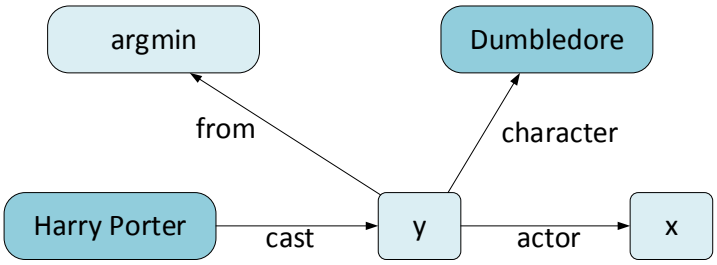


图 3.12 Yih 等人（2015）提出的分阶段查询图生成模型（STAGG）

然后，任务变成如何将一个自然语言问题转换成这样的查询图。Yih等人（2015）提出了一个分阶段查询图生成模型（STAGG），利用知识库从头逐步剪枝搜索空间，并构建结构化查询。

STAGG的关键组件包括主题实体链接和识别核心推理链，最后用约束和函数进行扩充，这基本上是一个逐步搜索的解析和排序过程。这里核心推理链捕获主题实体和lambda变量之间的关系，并为查询提供主干支撑。Yi等人（2015）使用深度卷积神经网络模型在语义上匹配一个问题和一个谓词序列（长度为2，中间有一个CVT节点）。

尽管STAGG在基准数据集上取得了成功，但是我们可以从STAGG的设计中吸取一些经验。

主题实体：第一步，找到主题实体并链接到知识库，也是关键的一步。STAGG使用了S-MART（Yang和Chang，2015），这是一种短文本实体链接的统计模型，它对后续步骤及整体性能都起着重要作用。当将主题实体链接更改为Freebase API时，STAGG的F1总分下降4.1%。

识别核心推理链：基本上这是一个关系抽取步骤，用于捕获如何从KB图上的主题实体开始获取入变量，通过CNN捕获，类似于Yih等人2014年提出的CNNSM。鉴于所有候选关系的巨大空间，STAGG仅考虑与主题实体相关的那些候选项，并且捕获问题如何在语义上与主题实体周围的KB关系序列匹配。因此，识别核心推理链的这一过程成为匹配和排名（Match-and-Rank）的步骤，同时避免了大规模的多分类方式。

增加约束和聚合：STAGG会将问题中的其他实体或时间表达式看成为核心推理链的约束节点，并且还会引入某些函数以进一步过滤答案，如将first、smallest转换为arg min。通过一组规则，我们将看到KBQA系统会引入聚合函数作为正式表示的一部分。

理解最高级别的表达方式：正如Berant和Liang（2014）、Zhang等（2015）中所讨论的那样，在问题中可以看到最高级的话语（表达）。大多数KBQA工作都采用模板或规则来分析最高级的表达式，只需从arg min或arg max中进行选择即可（Berant和Liang，2014；Yih等，2015）。然而，正式将最高级表达式分析作为针对KB的结构化比较结构，将有助于KBQA系统更好地处理最高级的话语，以及那些具有序数约束的话语。Zhang 等人（2015）设计了一个神经网络模型来学习最高级别话语和KB关系之间的潜在对应关系，作为比较结构中的比较维度。例如，从the longest river到元组<river. length、descending、1>，我们期望所有river在KB谓词river. length上进行比较（按降序排序），排名最高的就是我们的目标。

3. 神经符号机器

还有另一个有趣的语义分析模式的工作，试图结合神经网络和符号推理的优点来改善问答效果Liang等人于2017年引入神经符号机器（Neural Symbolic Machine，NSM），其配备了一个神经网络组件，负责从自然语言表示映射到可执行的代码，以及一个符号组件来执行代码，以剪枝搜索空间来找到答案。

具体来说，神经网络组件基本上是一个序列到序列模型，它维护一个关键变量存储器（Key-Variable Memory），在生成程序序列时处理中间结果。但是，神经网络组件和符号解释器的混合设计会使整个框架难以训练，然后要将其转换为一个强化学习问题来解决。

3.4.5 利用TensorFlow实现问答任务

问答（QA）系统是一个旨在为回答自然语言提出的问题而设计的系统。一些QA系统从诸如文本或图像之类的信息源中获取信息以回答具体的问题。这些依赖“信息来源”的系统可以分为两个主要子类别：开放领域，它需要回答的问题几乎可以是里面的任何选项，不限于特定领域；封闭领域，它涉及的问题具有一些特定的限制，因为它们是与一些预定义的信息源相关（如提供的上下文或类似于医学等特定领域）。

我们将创建一个基于神经网络的QA系统，并使用封闭领域的信息来源。为了做到这一点，我们使用Ankit Kumar等人在其论文《Ask Me Anything: Dynamic Memory Networks for Natural Language Processing》中介绍的动态记忆网络(Dynamic Memory Network，DMN)的简化版本(读者也可以自行查阅，笔者在https://arxiv.org/abs/1506.07285上获取到该文）。

1. bAbI数据集

对于这个项目，我们使用Facebook创建的bAbI数据集，与所有QA数据集一样，此数据集包含了我们需要的所有问题信息。bAbI数据集中的所有问题都有一个与之相关的上下文，这是一个句子序列，以便能够保证回答问题时存有所必需的详细信息。此外，数据集还为每个问题提供了正确答案。

根据回答问题所需的方法，bAbI数据集中的问题被划分为20个不同的子任务。每个子任务都有自己的一套训练问题集和一套单独的测试问题集,这些子任务测试各种标准的自然语言处理能力，包括时间推理和归纳逻辑。为了更好地了解这一点，我们来看一个QA系统将要回答问题的具体示例，如图3.13所示。

Context
Fred picked to the apple there. Bill travelled to the kitchen.Bill got the milk there.Jeff wen to the kitchen. Bill passed the milk to Jeff.Jeff handed the milk to Bill.
Question
Who did Jeff give the milk to ?

图3.13 BAbI数据集示例图

上图中蓝色部分为上下文（或语境），黄色的为问题及其对应答案(斜体为答案)。

此任务测试神经网络对于这三个对象之间内在关系所涉及相关运算或操作的理解。从语法上讲该任务将测试系统是否能够区分主语、直接宾语和间接宾语。在本例中，问题询问的是最后一句中的间接宾语——谁是那个从Jeff手里接收牛奶的人.神经网络必须对第五句中所涉及的对象做出精准区分，其中Bill是主语，Jeff是间接宾语，而在第六句中，Jeff则是主语。当然，我们的神经网络并没有收到任何关于主语或宾语是什么的明确训练,它必须从训练数据的示例中推断出这种合理的理解。

系统必须解决的另一个小问题是理解在整个数据集中使用的各种同义词，如Jeff把牛奶“递给(handed)”Bill，但是Jeff也可以同样容易地“给(gave)”或“交给(passed)”给Bill。不过，在这点上神经网络也不必从头开始，因为它可以从词向量那里得到一些帮助。由于神经网络中存在大量的词向量，这些词向量可以给出对应词的定义及其与其他词之间关系的信息，而相似词具有相似的向量化表示，这就意味着神经网络可以将它们视为几乎相同的词。对于词向量化，我们将使用Stanford 的全局词向量表示（GloVe）。

许多任务都有一个限制，那就是强制上下文包含用于回答问题的确切文字，如在我们的示例中就可以在上下文找到答案“Bill”。我们可以将这一限制当作优势，因为可以在上下文中搜索与最终结果最接近的词。

2. 分析GloVe并处理未知令牌

这里我们给出一个sentence2sequence,它是一个根据GloVe定义的映射并将字符串转换为矩阵的函数。该函数将字符串切分为多个令牌(Token，这个过程叫Tokenization，即切分词或词条)，这些比较小的字符串相当于标点符号、单词或单词的一部分。例如，把“Bill traveled to the kitchen.”切分为6个令牌，前5个令牌对应于单词，最后一个令牌对应于表示句末的“.”。每个令牌都被单独地向量化，从而产生对应于每个句子的向量列表，如图3.14所示。

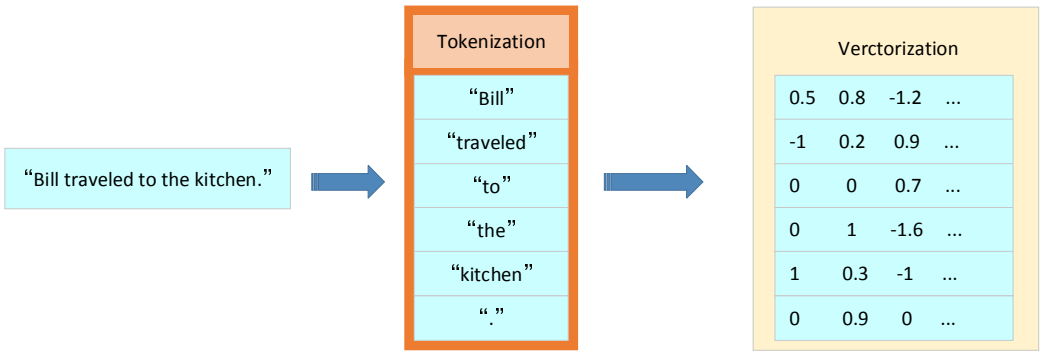


图 3.14 将句子转换成多个向量的过程

在bAbI的任务中，系统会遇到一些GloVe单词向量化中没有的单词，为了让神经网络能够处理这些未知的词条，我们需要为这些单词维护一个的向量化状态。通常的做法是用单个<UNK>向量替换所有未知令牌，但这不总是有效的。相反，我们可以使用随机化方法来为每个未知令牌创建一个新的向量。

当第一次遇到一个新的未知令牌时，我们简单地从原始GloVe向量化的分布(近似高斯分布)中获取一个新的词向量，并将该向量添加到GloVe词映射上。为了收集分布超参数，可以使用Numpy中的函数来自动计算方差和平均值。

```
# 反序列化 GloVe 向量
glove_wordmap = {}
with open(glove_vectors_file, "r", encoding="utf8") as glove:
    for line in glove:
        name, vector = tuple(line.split(" ", 1))
        glove_wordmap[name] = np.fromstring(vector, sep=" ")
wvecs = []
for item in glove_wordmap.items():
    wvecs.append(item[1])
s = np.vstack(wvecs)

# 收集分布超参数
v = np.var(s,0)
m = np.mean(s,0)
RS = np.random.RandomState()

# 而每当我们需要时，fill_unk 会提供一个新的词向量
def fill_unk(unk):
    global glove_wordmap
    glove_wordmap[unk] = RS.multivariate_normal(m,np.diag(v))
    return glove_wordmap[unk]
```

3. 已知或未知的数据部分

其实bAbI任务中的词汇量是有限的，这就需要神经网络学好单词之间的关系，即使不知道某些单词的含义，为了加快学习速度，我们也应该尽可能地选择具有内在意义的向量。为此，我们使用贪婪搜索方法来查找存在于斯坦福GLoVe词向量数据集中的单词，如果该单词不存在，则用一个未知的、随机创建的词向量表示来替换整个单词。

在这个词向量化模型下，我们可以定义一个新的sentence2sequence。

```
def sentence2sequence(sentence):
    """
       将输入段落转换为(m,d)矩阵,其中 n 是句子中的词条数量,d 是每个词向量的维数.
            这里不需要使用 TensorFlow,因为简单地将句子转换为基于映射的序列不需要
TensorFlow
       提供计算支持,普通 Python 模块足以完成此任务.

    """
    tokens = sentence.strip('"(),-').lower().split(" ")
    rows = []
    words = []
    #Greedy search for tokens
    for token in tokens:
        i = len(token)
        while len(token) > 0:
            word = token[:i]
            if word in glove_wordmap:
                rows.append(glove_wordmap[word])
                words.append(word)
                token = token[i:]
                i = len(token)
                continue
            else:
                i = i-1
            if i == 0:
                # word OOV
                # https://arxiv.org/pdf/1611.01436.pdf
                rows.append(fill_unk(token))
                words.append(token)
                break
    return np.array(rows), words
```

现在我们可以将每个问题所需的所有数据打包在一起，包括上下文、问题和答案的向量。在 bAbI 中，上下文被定义成了含有序列号的句子，使用反序列化(contextualize)函数来完成该任务。问题和答案在同一行，用制表符分隔开，所以我们可以利用制表符来标记某一行是否指向了问题。当编号重置时，未来的问题将引用新的上下文（注意，通常一个上下文对应多个问题）。答案还包含我们已保留但不需要使用的另一条信息:参考顺序，与回答问题所需的句子相对应的数字。在我们的系统中，神经网络将自学需要哪些句子来回答问题。

```
def contextualize(set_file):
    """
     读入问题数据集并构建问题+答案 - >上下文集.
         输出是一个数据点列表,每个数据点都是一个包含以下内容的 7 元素元组:
                上下文中的句子以向量化形式出现
                上下文中作为字符串词条列表的句子
                向量化形式的问题
                作为字符串词条列表的问题
                向量化形式的答案
                作为字符串词条列表的答案
                用于支持当前未使用的语句的数字列表
    """
    data = []
    context = []
    with open(set_file, "r", encoding="utf8") as train:
        for line in train:
            l, ine = tuple(line.split(" ", 1))
            # 将行号从所指向的句子中分离出来
            if l is "1":
                # 因为新的上下文总是从 1 开始，所以这是一个重置上下文的信号
                context = []
            if "\t" in ine:
                # 制表符是问题和答案之间的分隔符,在上下文语句中不存在
                question, answer, support = tuple(ine.split("\t"))
                data.append((tuple(zip(*context))+
                             sentence2sequence(question)+
                             sentence2sequence(answer)+
                             ([int(s) for s in support.split()],)))
                # 多个问题可能涉及相同的上下文,所以我们不用重置它
            else:
                # 上下文句子
                context.append(sentence2sequence(ine[:-1]))
    return data
train_data = contextualize(train_set_post_file)
test_data = contextualize(test_set_post_file)
        - dim: 产生噪声的维度
        输出:
        TensorFlow Tensor 范围 [-1, 1] 尺寸 [batch_size, dim]
        """
        return tf.random.uniform(shape=[batch_size, dim], minval=-1, maxval=1)
```

4. 定义超参数

```
tf.reset_default_graph()
```

由于这是实际神经网络的开始,因此还要定义网络所需的所有常量,称为“超参数”,它们定义了网络的“外观”（结构）和训练方式。

```
# 用于存储在网络中循环层之间传递的数据的维数
recurrent_cell_size = 128

# 词向量中的维数
D = 50

#  神经网络学习的速率。若过速率高的话，则可能会遇到数值不稳定或其他问题
learning_rate = 0.005

# dropout 概率
input_p, output_p = 0.5, 0.5

# 一次训练问题的个数
batch_size = 128

# 情景记忆传递的次数
passes = 4

# 前馈层大小：用于存储从前馈层传递的数据的维数
ff_hidden_size = 256

weight_decay = 0.00000001
# 奖励情景记忆的稀疏性，但会使训练变慢，不要让它大于 learning_rate
training_iterations_count = 400000
# 每次训练时神经网络训练问题的个数，有些问题会被多次计算

display_step = 100
# 在每次验证之前需要进行训练的迭代次数
```

此时，我们已经充分准备好了训练数据和测试数据，下一个任务是构建用于理解数据的神经网络。我们先从清除TensorFlow默认计算图开始，如果想更改一些东西，就可以选择再次运行神经网络。

5. 神经网络结构部分

神经网络结构被松散地划分为4个模块，在2015年 Ankit Kumar等人撰写的文章《Ask Me Anything: Dynamic Memory Networks for Natural Language Processing》中给出了相应的说明。

由于神经网络设计了一个循环层，其能够基于文本里的其他信息被动态地定义，因此被称为动态记忆网络(DMN)。DMN是基于对人类如何回答阅读理解型问题的理解。首先，人类会阅读上下文并在其中建立对相关事实的记忆，记住这些事实后再去阅读问题并重新检查上下文，特别是寻找与问题相关的一些答案的信息，并将问题与每个事实进行比对。

有时一个事实会引导我们走向另一个事实。在bAbI数据集中,神经网络希望找到足球的位置，它会先搜索关于足球的句子，并发现John是最后一个接触足球的人。然后搜索关于John的句子，发现John曾经在卧

室和走廊里待过。它一旦意识到John在走廊上是最后一个人，就可以回答这个问题并自信地说足球在走廊上，如图12-18所示。

在每一集或片段中，神经网络都会关注新的事实，以便能够找出正确答案。Kumar注意到神经网络错误地将一些权重值放在了第2句中，这是有原因的，因为John已经在那里了，尽管当时他没有足球（Ankit Kumar等，2015）。

6. 输入部分

输入模块是动态记忆网络4个模块中第一个给出答案的，它包含一个简单的输入，带有一个门控循环单元或GRU（TensorFlow 的tf. contrib. nn. GRUCell）的输入管道并以此来收集相关证据。每一个片段的证据或事实，在上下文中都对应一个句子，并且由该时间步长的输出表示，这需要一些非TensorFlow 的预处理。因此，我们可以收集句子末尾的位置并将其传递给TensorFlow，以便在后面的模块中使用。

我们可以利用TensorFlow 中的gather_nd()函数对这些数据进行处理，并使用这些被处理过的数据来选择相应的输出（gather_nd(函数是一个非常有用的工具）。

7. 问题部分

问题模块是第二个模块，也可以说是最简单的模块。它包括另一个GRU的管道。这次是在处理问题的文本上而不是寻找一些证据，我们可以简单地进到结尾状态，因为数据集保证问题的长度是一个句子。

8. 情景记忆部分

第三个模块是情景记忆模块，就是事情开始变得有意义的地方。它使用注意力执行多个过程，每个过程都包含多个GRU并对输入进行迭代。每个过程中的迭代都是基于当时对相应事实的关注程度并对当前记忆的权重进行更新。

9. 注意力部分

神经网络中的注意力最初是为了进行图像分析而设计的,特别是对于图像的某些部分信息远比其他部分的信息与分析主题更具相关性的情况。在执行任务的过程中，神经网络使用注意力能够在深入分析时确定所需对象的最佳位置,如查找图像中目标对象的位置、跟踪在图像之间移动的对象、面部识别或其他需要在图像中寻找最相关信息的任务。

这里的主要问题是注意力或至少硬性注意力（Hard Attention，只关注一个输入位置）很难被优化。与大多数神经网络一样，我们的优化方案是计算损失函数相对于输入和权重的导数，由于其二进制的性质，硬性注意力根本不可微分。因此，我们不得不使用基于实数的变体，被称为软性注意力(Soft Attention)，它对所有输入的位置使用某种形式的权重来表明相应的注意程度。而且，这里涉及的权重是完全可微分的，可以被正常训练。虽然学习硬性注意力也是可以的，但它比软性注意力更难以实现且有时表现更差,因此这里我们使用软性注意力。不用担心对导数函数进行编码，TensorFlow 的优化方案已经为我们做了相关处理。

在这个模型中,我们通过构造每个事实、当前记忆和原始问题之间的相似性度量来计算注意力。注意，这里的计算方法与普通注意力的计算方法是不同的，普通注意力只构造事实与当前记忆之间的相似性度量。我们将结果推送进一个两层的前馈神经网络来获得每个事实的注意力常数，然后对输入的事实使用一个GRU来赋予权重(通过相应的注意力常数赋予权重)，进而修改当前的记忆。为了避免在上下文短于矩阵的全长时向记忆中添加错误的信息，我们创建了一个掩码层，当事实不存在的情况下，模型不必在意它（即保留相同的记忆）。

另一个值得注意的方面是，注意力掩码层几乎总是围绕着神经网络的一个层对表示层进行封包对于图像而言，该封包最有可能发生在卷积层上(最有可能是直接映射到图像的位置)。而对于自然语言来说，该封包最有可能发生在循环层上。虽然将注意力集中在一个前馈层上，在技术上是可行的，但是通常没有什么用处——至少在那些后续的前馈神经网络层上难以模拟这种方式。

```
def attention(c, mem, existing_facts):
    """自定义注意力机制。
        c: 张量 [batch_size, maximum_sentence_count, recurrent_cell_size]包含上下文中的所有事实
        mem: 包含当前记忆的张量[batch_size, maximum_sentence_count, recurrent_cell_size]。为了得到准确的结果，对所有事实都应该有相同的记忆。
        existing_facts:张量[batch_size, maximum_sentence_count, 1]，就是我们上面提到的（二元）掩码。"""
    with tf.variable_scope("attending") as scope:
        # attending:度量我们决定去关注什么
        attending = tf.concat([c, mem, re_q, c * re_q,  c * mem, (c-re_q)**2, (c-mem)**2], 2)
        # m1: 前馈网络的第一层权重乘法值。
        #     我们平铺权重值以便手动传递，因为 tf.matmul 从 TensorFlow 1.2 开始就不会自动传递批量矩阵乘法。
        m1 = tf.matmul(attending * existing_facts,
                       tf.tile(w_1, tf.stack([tf.shape(attending)[0],1,1]))) * existing_facts
        # bias_1: 第一个前馈层的掩码变体仅对现有事实产生偏差
        bias_1 = b_1 * existing_facts
        # tnhan: 第一个非线性。选择 relu 而不是 tanh 等类似的非线性，是为了避免当 tanh 之类返回值接近 1 或-1 时出现低梯度量级的问题。
        tnhan = tf.nn.relu(m1 + bias_1)
        # m2:前馈网络的第二层乘法权重值
        m2 = tf.matmul(tnhan, tf.tile(w_2, tf.stack([tf.shape(attending)[0],1,1])))
        # bias_2: 第二个前馈层偏差的掩码变体。
        bias_2 = b_2 * existing_facts
        # norm_m2: 第二层权重的标准化版本，用于帮助确保 softmax 非线性不会饱和。
        norm_m2 = tf.nn.l2_normalize(m2 + bias_2, -1)
        # softmaxable: 在原本密集的张量上使用 sparse_softmax 的技巧。我们让 norm_m2 成为稀疏张量，使其在操作之后再次致密。
        softmax_idx = tf.where(tf.not_equal(norm_m2, 0))[:,:-1]
        softmax_gather = tf.gather_nd(norm_m2[...,0], softmax_idx)
        softmax_shape = tf.shape(norm_m2, out_type=tf.int64)[:-1]
        softmaxable = tf.SparseTensor(softmax_idx, softmax_gather, softmax_shape)
        return
tf.expand_dims(tf.sparse_tensor_to_dense(tf.sparse_softmax(softmaxable)),-1)
```

10. 答案模块

最后一个模块是答案模块，它使用完全连接层对问题和情景记忆模块的输出进行回归并获得“最终结果”的词向量，而与该结果距离最近的上下文中的词便是我们的最终输出（确保结果是一个真实中的词）。我们通过为每个词创建一个“分数”来计算最接近的词，该分数表示最终结果与当前词的距离。虽然可以设计一个返回多个词的答案模块，但是对于我们要解决的bAbI任务来说是没有必要的。

11. 模型优化

梯度下降是神经网络模型的默认优化器，其目标是减少神经网络的“损失”，这是衡量网络性能有多差的一个指标。它通过找到在当前输入下损失相对于每个权重值的导数，然后通过“降低”权重值来减少损失。大多数时候，这种方法效果很好的，但却不是最理想的方法。目前有许多不同的模型优化方法，如使用“动量”或其他更直接路径的近似的方法来最优化权重，而这些优化方法中最有用的一种被称为自适应矩估计（Adaptive Moment Estimation，Adam）。

Adam方法集成了AdaGrad和RMSProp算法的优点，通过计算梯度的一阶矩估计和二阶矩估计来为各种不同的参数创建独立的自适应性学习率。进一步讲，Adam方法计算了梯度的指数移动均值，使用两个超参数(betal和 beta2）控制着这些移动均值的衰减速度。而移动均值的初始化值和betal、beta2两参数值接近于Ⅰ时，矩估计的偏差就会接近于0。若移动均值初始化为零（有些参考资料这里给出的初始化值为l，笔者认为不准确）和两个超参数（betal、beta2）值接近于1时，就会引起向矩估计的偏差接近于零。

为了抵消这种偏差，Adam计算偏差校正的矩离估计，其值一般情况下会大于原始值，然后使用校正的估计值来更新整个神经网络的权重值。这些估计的组合使得Adam成为整体优化的最佳选择之一，尤其是对于复杂神经网络优化而言。这对于非常稀疏的数据（比如在自然语言处理任务中常见的数据）尤其适用。

在TensorFlow 中，我们可以通过创建tf. train. AdamOptimizer来使用Adam。

```
optimizer = tf.train.AdamOptimizer(learning_rate)
# 一旦我们有了一个优化器，我们就要求它将模型损失降到最低，以便进行适当的模型训练
opt_op = optimizer.minimize(total_loss)
```

12. 训练模型并分析预测

一切准备就绪后,我们就可以开始批处理训练数据以训练我们的神经网络。在进行训练过程中，我们应该持续监测神经网络在准确性指标方面做得如何。我们从测试数据里取出一部分作为验证数据集，这样它们和训练数据就不存在重叠问题了。

使用基于测试数据的验证集，可以更好地了解神经网络的泛化能力，即从训练数据里学习到的东西能否应用于其他未知的上下文中。如果在训练数据集中的部分数据上进行验证，那么神经网络可能会出现过拟合现象。换句话说，学习到了特定的示例并记住它们的答案，将无益于神经网络回答新问题。

经过一些训练之后，让我们来看看从神经网络中得到了什么样的答案。在图3.15中(笔者这里只给出三个问题的图像，其余两个问题的图像读者可以在代码文件中查看），我们可以将注意力可视化到上下文中所有句子(列)和每个情景（行）上;较深的颜色表示模型对该情景中那个句子有更多关注。

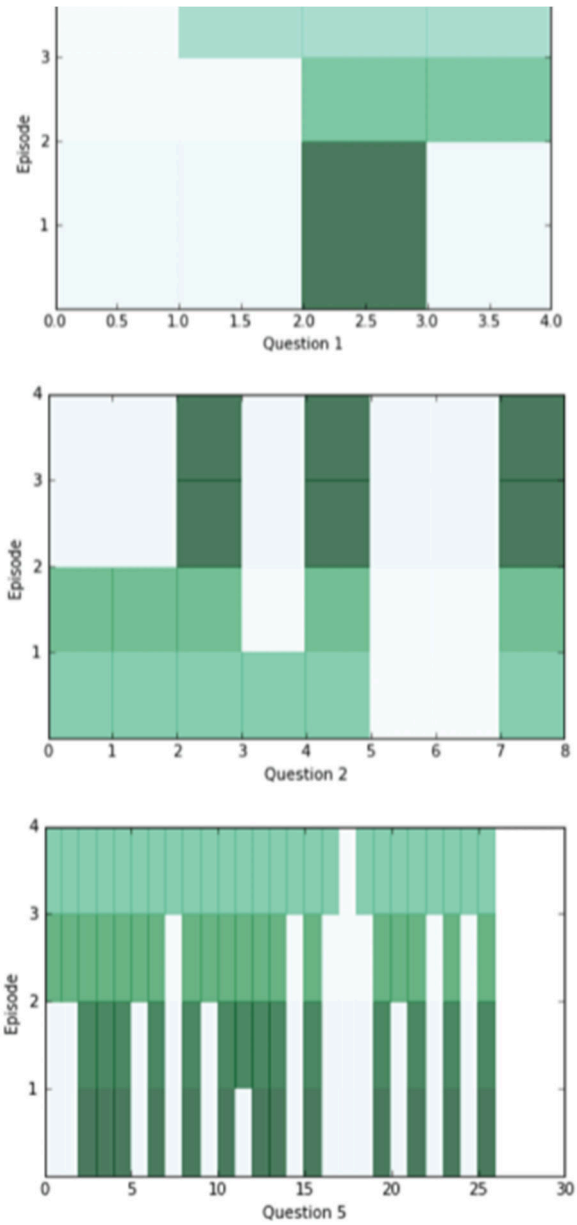


图3.15 模型输出的部分答案示意图

对于每个问题，应该看到注意力在至少两个情景里发生过变化，但有时注意力可以在一个情景内找到答案，有时需要全部4个情景才有可能找到答案。如果注意力看起来是空白的，那么它可能是饱和的，应该注意到了所有的事项。在这种情况下，可以尝试一个更高的weight_decay来训练，以减少甚至阻止这种情况的发生。在训练后期，饱和变得非常普遍。

为了了解上述问题的答案,我们可以使用上下文中距离分数的位置作为索引并查看该索引处的词。

为了得到更好的结果，我们可能需要长时间的训练（一般情况下需要12小时左右），最终应该能够达到非常高的精度（超过90%）。对Jupyter Notebook有经验的用户应该知道，在任何时候只要保持相同的tf. Session，都可以中断训练，并且仍然可以保存网络训练的进展。如果希望可视化注意力和当前网络给出的答案，这种方式很有用。

一旦查看完模型返回的内容，就可以关闭会话以释放系统资源。

3.5 自然语言处理实例-对话系统

3.5.1 对话系统概述

对话系统近一两年来变得异常火爆，在这儿我也把对话系统简单总结下。大家都觉得对话是最自然的交互方式，是未来的趋势等等，然而受限于技术不成熟，更多的还是自然语言理解的不成熟，到目前来说还没有一个真正意义上能解决用户某一方面问题而且体验很好的对话产品出现。亚马逊的echo算是比较成功的一个产品了，但更多的也是一些指令型的控制。不管怎样，要想做好对话系统，如下这几个技术点少不了，如图3. 16所示。

(1) 语义理解。包括实体识别、句法分析和意图识别等。

(2) 对话管理。包括对话状态追踪、上下文建模、指代消解和省略补全等。

(3) 知识表示。包括语义分析和知识建设。

(4) 用户管理。包括用户画像、长时记忆和短时记忆等。

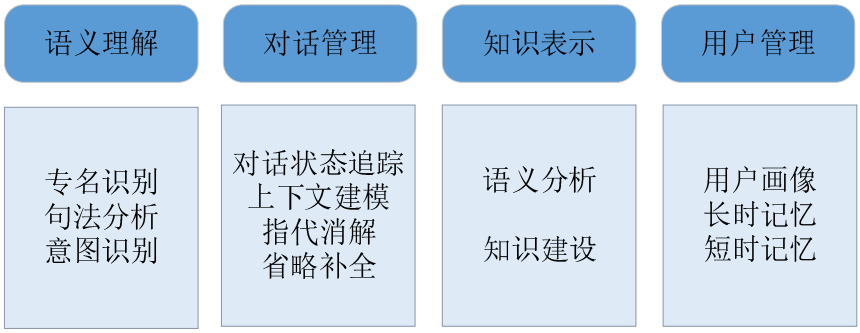


图3.16 对话系统技术点

从完成的任务来说，对话系统主要有这 3 个方面: 问答型、任务型和闲聊型。

3.5.2 问答型对话系统

问答系统研究有很长时间了，它解决的更多的是知识型的问题。目前研究最多的是答案是一个实体的客观性知识的任务，比如“中国的首都是哪儿?——北京”，对于答案是一句话或者一段话，或者是主观性的，甚至是个性化的问题，那就更是困难了。所以问答系统比较困难也比较复杂，涉及的技术点也很多，因此，要想做一个还可用的问答系统，就要针对某个具体场景去解决相应的问答需求，要做通用的自动问答系统，还是很困难的。问答系统的另一个应用就是客服系统，即使用机器来辅助人回答高频经常被问到的问题，提高客服人员的效率。问答系统目前主要有如下几类方法。

(1) 基于语义分析的方法

该方法的思路就是来一个 Query 之后首先语义分析出逻辑表达式，然后根据这个逻辑表达式去知识库中推理查询出答案。知识图谱的存储方案有两种: 一种是基于图的 RDF 方案; 一种是基于索引的方案。这个方法的重点就在于语义分析，如图 3. 17 所示。

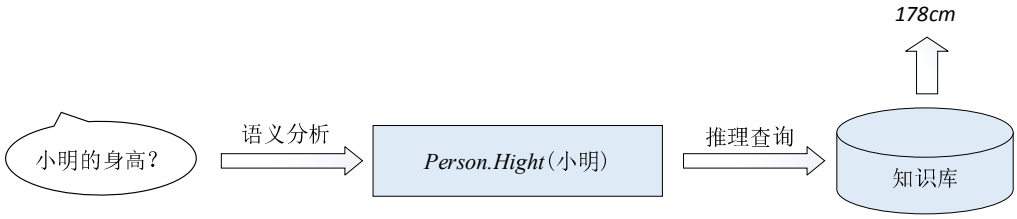


图3.17 语义分析流程图

(2) 基于信息抽取的方法

这种方法的思想就是来一个问题之后，首先是问题的各种分析，包括抽取关键词、关系词、焦点词以及问题的各种分类信息，然后从海量文档中检索出可能包含答案的文档段落，再在证据库中找到相关的证

据支撑，最后根据许多模型对结果排序找到最终的答案。

IBM的Watson是做的比较有影响力的一个系统，在其官网发表了一系列相关论文”，它就是使用这种方法，如图8.15所示。论文《Information Extraction over Structured Data: Question Answering with Freebase》也是使用类似的方法，其最大的不同在于从结构化知识图谱中找到候选子图是先确定实体，然后该实体周围一定范围内的子图即可作为候选子图。从无结构化数据中检索出潜在包含答案的段落使用的是基于搜索的方法。

(3)端对端的方法

这种方法是基于深度学习的模型，它首先将问题表征成一个向量，然后将答案也表征成向量，最后计算这两个向量的关联度，值越高那么就越可能是答案。它的核心就是在表征答案的时候如何把候选知识表征进来。然而一个真正的问答系统一般都是根据要解决的问题融合多种方法来处理。

3.5.3 任务型对话系统

任务型对话系统更多的是完成一些任务，比如订机票、订餐等等。这类任务有个较明显的特点，就是需要用户提供一些明显的信息（slot，槽位），如订机票就需要和用户交互得到出发地、目的地和出发时间等槽位，然后有可能还要和用户确认等等，最后帮用户完成一件事情。

系统会根据当前状态(state）和相应的动作（action）来决定下一步的状态和反馈，即求状态转移概率$P(r,s'|s,a)$，这其实就是马尔科夫决策过程的思想（MDP)。针对对话系统，它的流程如图 8.16 所示。首先是得到用户的 Query(如果是语音，需要语音识别转化成文字)。然后是自然语言理解模块(Natural Language Understanding，有语音的也叫 Spoken Language Understanding，现在一般语音识别是独立的模块)，主要是槽位识别和意图识别，而且这时候识别的意图有可能是有多个的，对应的槽位也会不同，都会有个置信度。然后就是对话管理模块，它包括 Dialog State Tracking（DST）和 Dialog Policy。DST 就是根据之前的信息得到它的 state，state 其实就是 slot 的信息:得到了多少 slot，还差什么 slot，以及它们的得分等等。Dialog Policy 就是根据 state 做出一个决策，叫 action，如还需要什么 slot，是否要确认等等。最后就是自然语言生成模块（Natural Language Generation)，把相应的 action 生成一句话回复给用户。Dialog Policy 就是根据 state 做一个决策，只要有了 state，就比较容易了，所以 DST 就比较关键。目前 DST 主要有这么几种方法。

(1)生成式模型（Generative Model ）

生成式模型把对话状态抽象成有关系统 action 和用户 SLU 结果的贝叶斯网络，所以它的求解可以使用贝叶斯推理，如图 3.18 所示。

其中，$\mathbf{b}'(s')$就是要求解的新的对话状态概率，u 是 SLU 的结果，这个是可观察到的，s 是上一轮对话状态，是个不可观察变量，u 是上一轮用户的真实 action，也是个不可观察变量，a 是系统 action，是可观察到的。加一撇的都是本轮的表示，$P(\tilde{u}'|u')$是给定用户真实 action 下 SLU 结果的概率，$P(u'|s',a)$是给定系统 action 和本轮对话状态下用户选择 action 为 u'的概率，$P(s'|s,a)$是给定上一轮对话状态和系统 action 下对话状态转变成$b(s)$是之前的对话状态概率。

$$\mathbf{b}'(s') = P(s,a,\tilde{u}) = \eta \sum_{u'} P(\tilde{u}'|u')P(u'|s',a) \sum_{s} P(s'|s,a)b(s)$$

其中，$\mathbf{b}'(s')$就是要求解的新的对话状态概率，u 是 SLU 的结果，这个是可观察到的，s 是上一轮对话状态，是个不可观察变量，u 是上一轮用户的真实 action，也是个不可观察变量，α 是系统 action，是可观察到的。加一撇的都是本轮的表示，$P(\tilde{u}'|u')$是给定用户真实 action 下 SLU 结果的概率，$P(u'|s',a)$是给定系统 action 和本轮对话状态下用户选择 action 为 u'的概率，$P(s'|s,a)$是给定上一轮对话状态和系统 action 下对话状态转变成$b(s)$是之前的对话状态概率。

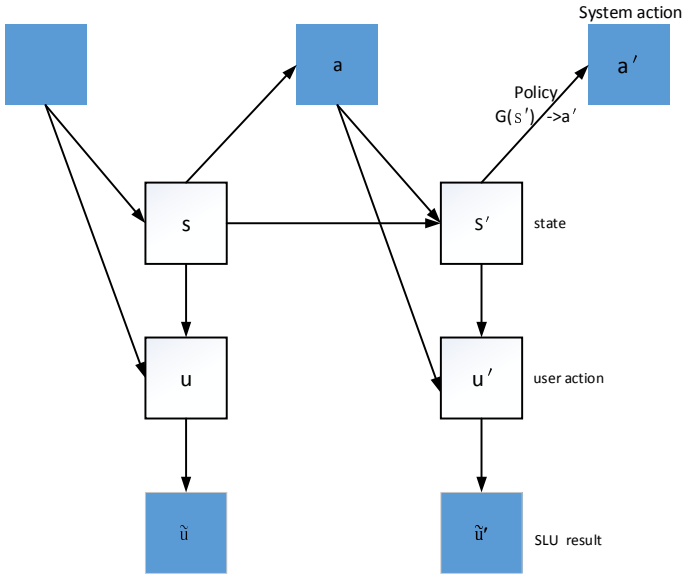


图3.18 生成式模型贝叶斯推理

(2) 判别式模型(Discriminative Model)

判别式模型是把问题抽象成一个分类问题，根据之前对话的一些特征来预测新的 state 的概率

$$b(s') = P(s'|f')$$

其中，f' 是从 SLU 结果和历史对话信息中抽取出来的特征。

这其实就是个分类问题，当然也可以把 slot 打分看成序列标注问题，这也就可以使用 CRF 等序列标注模型来处理了。只要是分类模型或者序列标注问题，那么也就可以使用深度学习了。而且只要是有明确 action 的地方，也就可以融入强化学习的激励惩罚机制。

(3) 规则系统

规则系统其实很好理解了，也是一种很可控的方法。对话管理主要有两个关键因素: state 和 action。我们知道 NLU 模块会分析出句子意图和相关 slot，意图和 slot 都是有置信度的，那么 state 其实就是意图和 slot，action 就是根据 state 的一些回复。对于某一个具体任务来说，需要询问的 slot 其实是可以预先知道的，那么缺哪些 slot 就询问哪些 slot 就可以了，不确定的 slot 和意图就和用户进行确认。对置信度低的不确定的进行确认并影响到之后相似的情况其实就是强化学习的思想了。当然还有一些细节需要考虑到系统里面，如对 slot 的范围限定、slot 和 slot 之间的冲突，以及用户有意捣乱的情况等等。举个例子，对于订机票任务，用户回复出发时间的时候，这个时间不能是当前时间之前的时间，否则就要和用户进一步确认；出发地点和到达地点也不能是同一个地点，否则也要确认。

所以规则系统就是有一个配置文件，写一些规则，然后线上把意图和 slot 的相关字段传过来进行解析处理就可以了。

我们把机票类简化成只需要用户的出发地和目的地，来看一个具体的如下示例，大家就会比较清楚规则系统了:

```
参数:
{
this_intent = flight
last_intent = flight
}

补全槽位:
```

```
{
if(sub_type = SEARCH)                    //子意图类型
start_loc<city>|end_loc<city>     //该子意图下需要处理的槽位
confirm = #出发地是[start_loc]|目的地是[end_loc]#吗?
final = 好的，你订了从[start_loc]到[end_loc]的航班，请确认。

if(sub_type = LOW_WEIGHT|UNCERTAIN)
NO_SLOT
confirm =
final = 哎呀，你是要聊天还是要订机票?
}

槽间约束:
{
start_loc != end_loc :: 出发地与目的地不能一样哦。
}

槽内约束:
{
// 类型约束
type:
start_loc == LOC_CITY|LOC_TOWN|LOC_FLIGHT_AIRPORT      // 正确类型
start_loc ~= LOC_COUNTRY|LOC_PROVINCE|LOC_ROAD          // 不正确类型
end_loc == LOC_CITY|LOC_TOWN|LOC_FLIGHT_AIRPORT
end_loc ~= LOC_COUNTRY|LOC_PROVINCE|LOC_ROAD

// 范围约束
content:
start_loc ==
end_loc ==
}

槽位定义:
{
start_loc:
is_must = 1          // 0:选填槽 1:必填槽 2:选填槽，但要向用户提问
repeat_time = 2      // 未填时重复问几次

end_loc:
is_must = 1
```

```
repeat_time = 2
}

回答:
{
normal:
null_out = 我问的不是这个啊
intent_true = 好的，我跟您确认下具体的信息。

start_loc:
type_true_content_true =出发地已更新成功。
type_true_content_false =出发地已更新成功。
type_correlate = 您的出发城市是?
type_false = 您从哪个城市出发呢?

end_loc:
type_true_content_true =目的地已更新成功。
type_true_content_false =目的地已更新成功。
type_correlate = 您要去哪个城市?
type_false =您去哪个城市呢?
}
```

针对任务型对话系统的这 3 类方法各有特点，基于模型的方法需要有标注的训练数据，而在数据缺失的情况下，规则系统会比较有效。规则系统突出了 NLU 的重要性，因为它用到了 NLU 的 1-best 结果，而不是 N-best 结果。

3.5.4 闲聊型对话系统

闲聊型的对话系统更多地是人和机器没有明确限定的聊天，如果前两个类型是打机器的“智商”牌的话，那么这个类型就是打机器的“情商”牌，让人感觉机器更加亲切，而不是冷冰冰的完成任务（如果回复语句自然且有意思的话，其实也不那么冷冰冰）。闲聊型对话系统主要有 3 种方法:规则方法、生成模型和检索方法。

(1) 规则方法

20 世纪 60 年代，由约瑟夫·魏泽堡和肯尼斯·科尔比共同编写的一个聊天机器人 ELIZA 就是使用纯规则方法，这种方法就是写一个较泛化的模板，然后回复一个或多个相应的模板，如下面两个规则。当用户说 I want to play basketball，那么就会回复 Why do you want to play basketball 或者 You really want to play basketball。所以规则系统关键是如何写一堆规则和线上的快速匹配。目前没有哪个系统是纯规则的了，规则方法顶多只是在一些其他方法处理不好的情况下的一个补充。

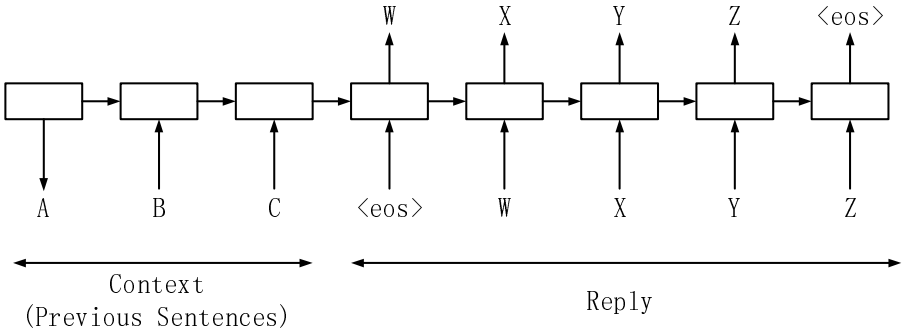
```
(?X are you ?Y)
(Would you prefer it if I weren 't ?Y)

((?* ?X) I want (?* ?Y))
(You really want ?Y)
(Why do you want ?Y)
```

(2) 生成模型

生成模型是随着深度学习的热潮而提出的比较火热的方向。前面的深度学习章节详细介绍了很多模型，现在我们简单回顾下思想。它首先使用一个 RNN 模型把输入句子“ABC”表示成一个向量，然后把这个向量作为另一个 RNN 模型的输入，最后使用语言模型生成目标句子“WXYZ”。这种方法的优点是省去了中间的模块，缺点是生成的大多是泛泛的无意义的回复、前后回复不一致，或者有句子不通顺的问题（一句话不通顺其实都很难解决）。好多人也在融合上下文、Topic、互信息等来解决多样性问题，但遗憾的是，只

使用这种方法效果并不尽人意（而且非常依赖于大量高质量的训练语料），它可以结合其他模型和策略来处理。例如 Google 发表的这篇论文《Smart Reply: Automated Response Suggestion for Email》，就非常值得一读，它要完成的任务是邮件的自动回复。这更多的是一个简化版的对话系统，论文为了能生成高质量的回复，首先用图方法事先聚类出一个高质量集合，最终的回复都是在该集合中，这算是较实用的一个生成系统了，如图 3.19 所示。



$$P(y_1,\ldots,y_{T'}\mid x_1,\ldots,x_T)=\prod_{t=1}^{T'}P(y_t\mid v,y_1,\ldots,y_{t-1})$$

图 3.19 生成系统结构图

(3) 检索方法

检索方法的思想是，如果机器要给人回复一句话，假设这句话或相似的话之前有人说过，只需要把它找出来就可以了。这种方法就需要事先挖掘很多的语料。它最基本的流程就是首先进行 NLU，然后从语料库中召回一些可能的回复，最后使用更精细和丰富的模型（语义相似度、上下文模型等）找出最合适的回复给用户，期间一定要注意处理“答非所问”的现象。语义相似度的技术前面也介绍过了。对话还少不了这几个核心技术: 意图识别、上下文模型、个性化模型和自学习。意图识别其实是语言量化的问题，目前是通过分类、聚类、语义相似度和模板等多个技术来解决。上下文模型就需要省略补全，把缺失的信息补全了，那其实就是单轮对话了，否则就要结合上下文的 Topic 和 Keyword 来处理，主要的宗旨就是不能语义跑偏了。个性化模型不光要考虑用户画像，还要考虑场景（时间、地点等）。个性化我觉得重点在机器能否记住用户的状态并给出相应的回复，就像朋友之间一样，假如你感冒了，然后又想去打球，那么朋友就会问:“你感冒了还去打球啊”?是不是感觉很亲切，这就需要进行话题识别，自然就会引出话题的关系识别，如话题冲突、话题连贯等等。自学习需要结合用户聊天过程中的反馈（情感分析）来对当前的对话进行评估是否可再利用。

如果把问答型、任务型和聊天型都融合到一个对话系统中（不同的系统侧重点会不同），就是如图 3.20 所示。

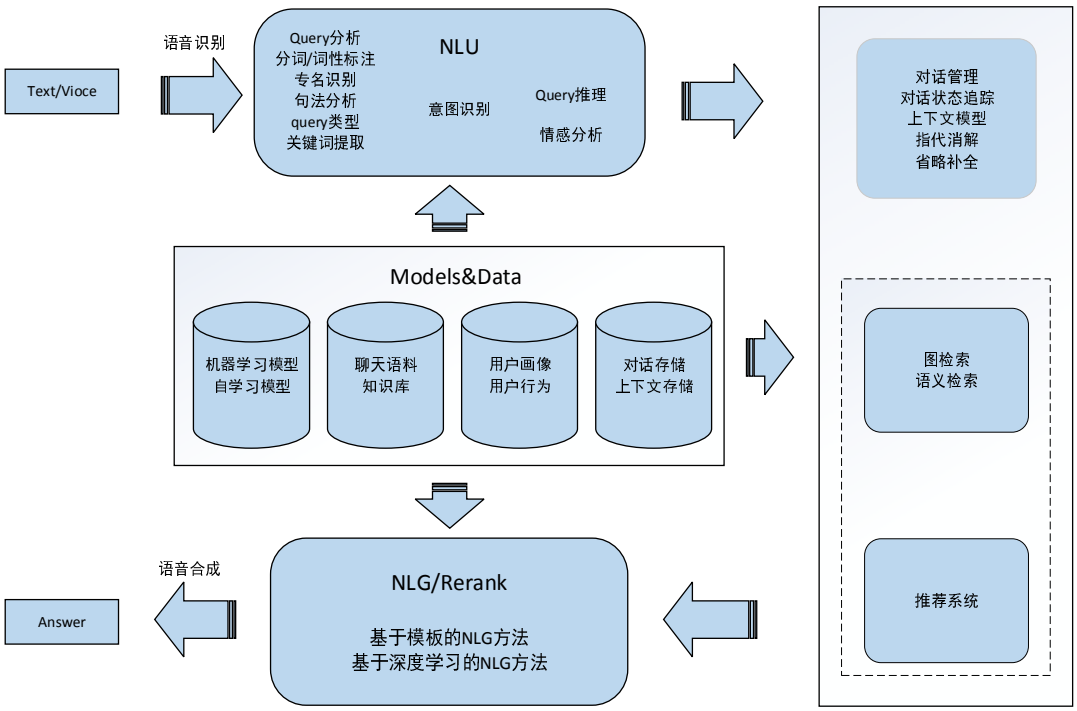


图3.20 融合的对话系统框架图

对话系统评价。对于问答型，现在能回答的更多的是实体，所以也很容易评价。对于任务型，从整体就是看该对话是否完成了用户的任务，所以也是可评价的。对于闲聊型，由于闲聊回复太多样，所以这个任务是很难自动评价的，它等同于图灵停机问题（是否存在这样一个程序，它能够计算任何程序在给定输入上是否会结束），被证明是不存在的。举个通俗例子：你能写个程序，来判断你的另一个程序是否有bug吗？所以闲聊型任务更多的需要人工评价以及反馈。

3.6 自然语言处理实例-机器翻译

机器翻译技术的发展一直与计算机技术、信息论、语言学等学科的发展紧密相随。从早期的词典匹配，到词典结合语言学专家知识的规则翻译，再到基于语料库的统计机器翻译，随着计算机计算能力的提升和多语言信息的爆发式增长，机器翻译技术逐渐走出象牙塔，开始为普通用户提供实时便捷的翻译服务。

3.6.1 发展道路

机器翻译的研究历史可以追溯到 20 世纪三四十年代。20世纪30年代初，法国科学家G. B. 阿尔楚尼提出了用机器来进行翻译的想法。1933年，苏联发明家П. П. 特罗扬斯基设计了把一种语言翻译成另一种语言的机器，并在同年9月5日登记了他的发明；但是，由于30年代技术水平还很低，他的翻译机没有制成。1946 年，第一台现代电子计算机 ENIAC 诞生，随后不久，信息论的先驱、美国科学家 W. Weaver 和英国工程师A. D. Booth 在讨论电子计算机的应用范围时，于1947年提出了利用计算机进行语言自动翻译的想法。1949年，W. Weaver 发表《翻译备忘录》，正式提出机器翻译的思想。走过六十年的风风雨雨，机器翻译经历了一条曲折而漫长的发展道路，学术界一般将其划分为如下四个阶段：

- 开创期（1947-1964）

1954 年，美国乔治敦大学（Georgetown University）在IBM公司协同下，用IBM-701计算机首次完成了英俄机器翻译试验，向公众和科学界展示了机器翻译的可行性，从而拉开了机器翻译研究的序幕。

中国开始这项研究也并不晚， 早在1956年，国家就把这项研究列入了全国科学工作发展规划，课题名称是“机器翻译、自然语言翻译规则的建设和自然语言的数学理论”。1957 年，中国科学院语言研究所与计算技术研究所合作开展俄汉机器翻译试验，翻译了9 种不同类型的较为复杂的句子。

从20世纪50年代开始到20世纪60年代前半期，机器翻译研究呈不断上升的趋势。美国和前苏联两个超级大国出于军事、政治、经济目的，均对机器翻译项目提供了大量的资金支持，而欧洲国家由于地缘政治和经济的需要也对机器翻译研究给予了相当大的重视，机器翻译一时出现热潮。这个时期机器翻译虽然刚刚处于开创阶段，但已经进入了乐观的繁荣期。

- 受挫期（1964-1975）

1964 年，为了对机器翻译的研究进展作出评价，美国科学院成立了语言自动处理咨询委员会(Automatic Language Processing Advisory Committee，简称 ALPAC 委员会)，开始了为期两年的综合调查分析和测试。

1966年11月，该委员会公布了一个题为《语言与机器》的报告（简称ALPAC报告），该报告全面否定了机器翻译的可行性，并建议停止对机器翻译项目的资金支持。这一报告的发表给了正在蓬勃发展的机器翻译当头一棒，机器翻译研究陷入了近乎停滞的僵局。无独有偶，在此期间，中国爆发了“十年文革”，基本上这些研究也停滞了。机器翻译步入萧条期。

- 恢复期（1975-1989）

进入 70 年代后，随着科学技术的发展和各国科技情报交流的日趋频繁，国与国之间的语言障碍显得更为严重，传统的人工作业方式已经远远不能满足需求，迫切地需要计算机来从事翻译工作。 同时，计算机科学、语言学研究的发展，特别是计算机硬件技术的大幅度提高以及人工智能在自然语言处理上的应用，从技术层面推动了机器翻译研究的复苏，机器翻译项目又开始发展起来，各种实用的以及实验的系统被先后推出，例如 Weinder 系统、EURPOTRA 多国语翻译系统、TAUM-METEO系统等。

而我国在“十年浩劫”结束后也重新振作起来，机器翻译研究被再次提上日程。“784”工程给予了机器翻译研究足够的重视，80 年代中期以后，我国的机器翻译研究发展进一步加快，首先研制成功了 KY-1 和MT/EC863 两个英汉机译系统，表明我国在机器翻译技术方面取得了长足的进步。

- 新时期（1990 至今）

随着 Internet 的普遍应用，世界经济一体化进程的加速以及国际社会交流的日渐频繁，传统的人工作业的方式已经远远不能满足迅猛增长的翻译需求，人们对于机器翻译的需求空前增长，机器翻译迎来了一个新的发展机遇。国际性的关于机器翻译研究的会议频繁召开，中国也取得了前所未有的成就，相继推出了一系列机器翻译软件，例如“译星”、“雅信”、“通译”、“华建”等。在市场需求的推动下，商用机器翻译系统迈入了实用化阶段，走进了市场，来到了用户面前。

新世纪以来，随着互联网的出现和普及，数据量激增，统计方法得到充分应用。互联网公司纷纷成立机器翻译研究组，研发了基于互联网大数据的机器翻译系统，从而使机器翻译真正走向实用，例如“百度翻译”，“谷歌翻译”等。近年来，随着深度学习的进展，机器翻译技术的到了进一步的发展，促进了翻译质量的快速提升，在口语等领域的翻译更加地道流畅。

- 翻译过程

整个机器翻译的过程可以分为原文分析、原文译文转换和译文生成3个阶段。在具体的机器翻译系统中，根据不同方案的目的和要求，可以将原文译文转换阶段与原文分析阶段结合在一起，而把译文生成阶段独立起来，建立相关分析独立生成系统。在这样的系统中，原语分析时要考虑译语的特点，而在译语生成时则不考虑原语的特点。在研究多种语言对一种语言的翻译时，宜于采用这样的相关分析独立生成系统。也可以把原文分析阶段独立起来，把原文译文转换阶段同译文生成阶段结合起来，建立独立分析相关生成系统。在这样的系统中，原语分析时不考虑译语的特点，而在译语生成时要考虑原语的特点，在研究一种语言对多种语言的翻译时，宜于采用这样的独立分析相关生成系统。还可以把原文分析、原文译文转换与译文生成分别独立开来，建立独立分析独立生成系统。在这样的系统中，分析原语时不考虑译语的特点，生成译语时也不考虑原语的特点，原语译语的差异通过原文译文转换来解决。在研究多种语言对多种语言的翻译时，宜于采用这样的独立分析独立生成系统。

- 翻译简史

中国机器翻译研究起步于 1957 年,是世界上第 4 个开始研究机器翻译的国家，60 年代中期以后一度

中断，70 年代中期以来有了进一步的发展。中国社会科学院语言研究所、中国科学技术情报研究所、中国科学院计算技术研究所、黑龙江大学、哈尔滨工业大学等单位都在进行机器翻译的研究；上机进行过实验的机器翻译系统已有十多个，翻译的语种和类型有英汉、俄汉、法汉、日汉、德汉等一对一的系统,也有汉译英、法、日、俄、德的一对多系统（FAJRA 系统）。此外，还建立了一个汉语语料库和一个科技英语语料库。中国机器翻译系统的规模正在不断地扩大，内容正在不断地完善。近年来，中国的互联网公司也发布了互联网翻译系统，如“百度翻译”“有道翻译”等。

- 系统划分

机译系统可划分为基于规则（ Rule-Based ）和基于语料库（Corpus-Based）两大类。前者由词典和规则库构成知识源；后者由经过划分并具有标注的语料库构成知识源，既不需要词典也不需要规则，以统计规律为主。机译系统是随着语料库语言学的兴起而发展起来的，世界上绝大多数机译系统都采用以规则为基础的策略，一般分为语法型、语义型、知识型和智能型。不同类型的机译系统由不同的成分构成。抽象地说，所有机译系统的处理过程都包括以下步骤：对源语言的分析或理解，在语言的某一平面进行转换，按目标语言结构规则生成目标语言。技术差别主要体现在转换平面上。

- 词汇型

从美国乔治敦大学的机器翻译试验到50年代末的系统，基本上属于这一类机器翻译系统。它们的特点是:①以词汇转换为中心，建立双语词典,翻译时,文句加工的目的在于立即确定相应于原语各个词的译语等价词；②如果原语的一个词对应于译语的若干个词，机器翻译系统本身并不能决定选择哪一个，而只能把各种可能的选择全都输出；③语言和程序不分，语法的规则与程序的算法混在一起，算法就是规则。由于第一类机器翻译系统的上述特点，它的译文质量是极为低劣的，并且，设计这样的系统是一种十分琐碎而繁杂的工作，系统设计成之后没有扩展的余地，修改时牵一发而动全身，给系统的改进造成极大困难。

- 语法型

研究重点是词法和句法，以上下文无关文法为代表，早期系统大多数都属这一类型。语法型系统包括源文分析机构、源语言到目标语言的转换机构和目标语言生成机构3部分。源文分析机构对输入的源文加以分析，这一分析过程通常又可分为词法分析、语法分析和语义分析。通过上述分析可以得到源文的某种形式的内部表示。转换机构用于实现将相对独立于源文表层表达方式的内部表示转换为与目标语言相对应的内部表示。目标语言生成机构实现从目标语言内部表示到目标语言表层结构的转化。

60年代以来建立的机器翻译系统绝大部分是这一类机器翻译系统。它们的特点是：①把句法的研究放在第一位,首先用代码化的结构标志来表示原语文句的结构,再把原语的结构标志转换为译语的结构标志,最后构成译语的输出文句;②对于多义词必须进行专门的处理,根据上下文关系选择出恰当的词义，不容许把若干个译文词一揽子列出来;③语法与算法分开，在一定的条件之下，使语法处于一定类别的界限之内，使语法能由给定的算法来计算，并可由这种给定的算法描写为相应的公式，从而不改变算法也能进行语法的变换，这样，语法的编写和修改就可以不考虑算法。第2类机器翻译系统不论在译文的质量上还是在使用的方便上,都比第1类机器翻译系统大大地前进了一步。

- 语义型

研究重点是在机译过程中引入语义特征信息，以Burtop提出的语义文法和Charles Fillmore提出的格框架文法为代表。语义分析的各种理论和方法主要解决形式和逻辑的统一问题。利用系统中的语义切分规则，把输入的源文切分成若干个相关的语义元成分。再根据语义转化规则，如关键词匹配，找出各语义元成分所对应的语义内部表示。系统通过测试各语义元成分之间的关系，建立它们之间的逻辑关系，形成全文的语义表示。处理过程主要通过查语义词典的方法实现。语义表示形式一般为格框架，也可以是概念依存表示形式。最后，机译系统通过对中间语义表示形式的解释，形成相应的译文。

70年代以来，有些机器翻译者提出了以语义为主的第3类机器翻译系统。引入语义平面之后,就要求在语言描写方面作一些实质性的改变，因为在以句法为主的机器翻译系统中,最小的翻译单位是词,最大的翻译单位是单个的句子，机器翻译的算法只考虑对一个句子的自动加工，而不考虑分属不同句子的词与词之

间的联系。第3类机器翻译系统必须超出句子范围来考虑问题,除了义素、词、词组、句子之外，还要研究大于句子的句段和篇章。为了建立第3类机器翻译系统,语言学家要深入研究语义学,数学家要制定语义表示和语义加工的算法,在程序设计方面，也要考虑语义加工的特点。

➢ 知识型

目标是给机器配上人类常识，以实现基于理解的翻译系统，以Tomita提出的知识型机译系统为代表。知识型机译系统利用庞大的语义知识库，把源文转化为中间语义表示，并利用专业知识和日常知识对其加以精练，最后把它转化为一种或多种译文输出。

➢ 智能型

目标是采用人工智能的最新成果，实现多路径动态选择以及知识库的自动重组技术，对不同句子实施在不同平面上的转换。这样就可以把语法、语义、常识几个平面连成一有机整体，既可继承传统系统优点，又能实现系统自增长的功能。这一类型的系统以中国科学院计算所开发的IMT/EC系统为代表。

➢ 基于统计的机器翻译

基于统计的机器翻译方法把机器翻译看成是一个信息传输的过程，用一种信道模型对机器翻译进行解释。这种思想认为，源语言句子到目标语言句子的翻译是一个概率问题，任何一个目标语言句子都有可能是任何一个源语言句子的译文，只是概率不同，机器翻译的任务就是找到概率最大的句子。具体方法是将翻译看做对原文通过模型转换为译文的解码过程。因此统计机器翻译又可以分为以下几个问题：模型问题、训练问题、解码问题。所谓模型问题，就是为机器翻译建立概率模型，也就是要定义源语言句子到目标语言句子的翻译概率的计算方法。而训练问题，是要利用语料库来得到这个模型的所有参数。所谓解码问题，则是在已知模型和参数的基础上，对于任何一个输入的源语言句子，去查找概率最大的译文。

实际上，用统计学方法解决机器翻译问题的想法并非是20世纪90年代的全新思想，1949年W. Weaver在那个机器翻译备忘录就已经提出使用这种方法，只是由于乔姆斯基(N. Chomsky)等人对计的批判，这种方法很快就被放弃了。批判的理由主要是一点：语言是无限的，基于经验主义的统计描述无法满足语言的实际要求。

- 基于实例的机器翻译

与统计方法相同，基于实例的机器翻译方法也是一种基于语料库的方法，其基本思想由日本著名的机器翻译专家长尾真提出，他研究了外语初学者的基本模式，发现初学外语的人总是先记住最基本的英语句子和对应的日语句子，而后做替换练习。参照这个学习过程，他提出了基于实例的机器翻译思想，即不经过深层分析，仅仅通过已有的经验知识，通过类比原理进行翻译。其翻译过程是首先将源语言正确分解为句子，再分解为短语碎片，接着通过类比的方法把这些短语碎片译成目标语言短语，最后把这些短语合并成长句。对于实例方法的系统而言，其主要知识源就是双语对照的实例库，不需要什么字典、语法规则库之类的东西，核心的问题就是通过最大限度的统计，得出双语对照实例库。

基于实例的机器翻译对于相同或相似文本的翻译有非常显著的效果，随着例句库规模的增加，其作用也越来越显著。对于实例库中的已有文本，可以直接获得高质量的翻译结果。对与实例库中存在的实例十分相似的文本，可以通过类比推理，并对翻译结果进行少量的修改，构造出近似的翻译结果。

3.6.2 实现神经网络机器翻译

现在我们将实现一个真正的神经网络机器翻译(NMT）系统，这里使用原始的TensorFlow操作变量来实现神经网络机器翻译。这里之所以使用原始的TensorFlow操作变量方法，是因为一旦我们掌握了从零开始学习实现机器翻译而不借助于任何辅助函数，那么我们将能够快速学习诸如序列到序列(seq2seq）之类的库。另外，使用原始TensorFlow操作变量来实现序列到序列模型的资源其实很少，而关于如何使用seq2seq库进行机器翻译的在线资源却非常多。

1．关于样本数据

我们使用https://nlp.stanford.edu/projects/nmt/上提供的英语－德语句子对，上面有大约450万个句子对可用。由于考虑计算可行性，我们仅使用250 000个句子对。词汇有50000个常用的英语单词和50000个常

见的德语单词,而在词汇表中未找到的单词将被替换为特殊标记<unk>。在这里，我们将列出在数据集中存在的一些例句;

DE: Das GroBunternehmen sieht sich einfach die Produkte des kleinen Unternehmens an und unterstellt so viele Patentverletzungen ， wie es nur geht .

EN: The large corporation will look at the products of the small company and bring up as many patent infringement assertions as possible .

DE: In der ordentlichen sitzung am 22. September 2008 befasste sich der Aufsichtsrat mit strategischen rhemen aus den einzelnen Geschaftsbereichen wie der Positionierung des Kassamarktes im wettbewerb mit auBerb8rslichen Handelsplattformen ， den Innovationen im Derivatesegment und verschiedenen ktivitaten im Nachhandelsbereich

EN: At the regular meeting on 22 September 2008 ， the Supervisory Board dealt with strategic issues from the various business areas , such as the positioning of the cash market in competition with orc trading platforms ， innovation in the derivatives segment and various post ##AT##一##AT## trading activities .

2．预处理数据

按照代码文件中的操作，下载了训练数据（train.en和 train.de）后，让我们来看看这些文件中有什么内容。接着，我们将从这些样本数据的大型语料库中选择250,000个句子对，同时从训练数据中收集100个句子作为测试数据。这两种语言的词汇表可以在vocab.50K.en.txt和vocab.50K.de.txt中找到。

对于词向量学习(如果单独执行)，反转句子是可选的，因为反转句子不会改变给定单词的上下文，所以我们将使用以下简单的标记化算法将句子标记为单词。我们要在各种标点符号之前引入空格，以便将它们标记为单个元素，然后对于未在词汇表中发现的任何单词，会用一个特殊的标记<unk>替换。is_source参数告诉我们是正在处理源语句(is_source=True)还是在处理目标语句(is_source=False):

```
def split_to_tokens(sent,is_source):
    '''
    此函数接受一个句子（源语言或目标语言）并使用各种措施（例如删除标点符号）预处理该句子
    '''

    global src_unk_count, tgt_unk_count

    # 移除标点符号和换行符
    sent = sent.replace(',',' ,')
    sent = sent.replace('.',' .')
    sent = sent.replace('\n',' ')

    sent_toks = sent.split(' ')
    for t_i, tok in enumerate(sent_toks):
        if is_source:
          #  src_dictionary 包含源语言词汇表中字->字 ID 的映射
          if tok not in src_dictionary.keys():
              if not len(tok.strip())==0:
                  sent_toks[t_i] = '<unk>'
                  src_unk_count += 1
```

```
        else:
            # tgt_dictionary 包含目标语言词汇表中字->字 ID 的映射
            if tok not in tgt_dictionary.keys():
                if not len(tok.strip())==0:
                    sent_toks[t_i] = '<unk>'
                    #打印(tok)
                    tgt_unk_count += 1
    return sent_toks
```

3．学习词向量

接下来，我们将继续学习词向量。要学习词向量，我们需要使用连续词袋（CBOW）模型，也可以尝试其他词向量学习的方法，如 GloVe。在这里不再贴出相关代码，具体可以查看代码文件:ch11/1word2vec.py（其中给出了一些学习词向量的代码）。

在机器翻译系统训练中，可以同时学习词向量，另一种选择是使用预训练的词向量。

```
German word Embeddings
    Nearest to In: in，Aus，An，Neben，Bei，Mit，Trotz，Auf，
    Nearest to war: ist，hat，scheint，wre，hatte，bin，waren，kam,
    Nearest to so: verbreitet，eigentlich，ausserdem，ziemlich，Rad-,zweierlei,wollten,ebenso,
    Nearest to schritte: Meter，Minuten，Gehminuten，Autominuten，km，Kilometer,Fahrminuten，
steinwurf,
    Nearest to sicht: Aussicht，Ausblick，Blick，Kombination，Milde,Erscheinung Terroranschlage，Ebenen,
    English word Embeddings
    Nearest to more: cheaper，less，easier，better，further，greater,bigger,More,
    Nearest to states: Kingdom，Nations，accross，attrition，Efex，Republic, authoritative，sorbonne,
    Nearest to Italy: Spain，Poland，France，Switzerland，Madrid，Portugal，Fuengirola，5l,
    Nearest to island: shores，Principality，outskirts，islands，skyline,ear,continuation,capital,
    Nearest to 2004: 2005，2001，2003，2007，1996，2006，1999，1995,
```

4．定义编码器和解码器

我们将使用两个单独的LSTM作为编码器和解码器。

首先定义以下超参数。

batch_size:设置批量大小时必须非常小心。神经网络机器翻译在运行时可能需要非常多的内存。

num_nodes:这是LSTM中隐藏单元的数量.较大的超参数num_nodes将导致更好的性能和更高的计算成本。

enc_num_unrollings:将其设置为源语句中的单词数量。我们将在一个单独的计算中为句子的完整长度展开LSTM。虽然cnc_num_unrollings越高，模型的性能越好，但是这会降低算法速度。

dec_num_unrollings:将其设置为目标语句中的单词数量。dec_num_unrollings越高，模型的性能表现就越佳，同时对应的计算成本也就越高。

embedding_size:向量的维数，对于使用词向量的大多数实际问题而言，100~300 的词向量大小就足够了。

```
tgt_emb_mat = np.load('en-embeddings.npy')
input_size = tgt_emb_mat.shape[1]

num_nodes = 128
batch_size = 10

# 我们一次性展开源语句和目标语句的全长
enc_num_unrollings = 40
dec_num_unrollings = 60
```

如果 batch size 较大（在标准笔记本电脑上超过 20 个），则可能会遇到诸如计算机资源耗尽的问题。在这种情况下，应减少批量大小并重新运行代码。

接下来，我们将定义LSTM和softmax层的权重值和偏差，使用编码器和解码器变量范围来使变量的命名更直观，这是标准的LSTM细胞，我们不会重复权重值的定义。

下面，我们定义4个TensorFlow 的占位符进行训练。

enc_train_inputs:这是enc_num_unrollings占位符的列表，其中每个占位符的大小为[batch_size，input_size]，用于将一批源语言句子提供给编码器。

dec_train_inputs:这是dec_num_unrollings占位符的列表，其中每个占位符的大小为[batch_size，input_size]，用于提供相应批次的目标语言句子.

dec_train_labels:这是 dec_num_unrollings 占位符的列表，其中每个占位符的大小为[batch_size，vocabulary_size]，包含偏移量为 1 的 dec_train_inputs 的单词。

dec_train_masks:这与 dec_train_inputs 的大小相同，并且屏蔽了损失计算中具有</s>标签的任何元素。这很重要，因为有许多带</s>标记的数据点，被用于将现有句子填充成固定长度的句子。

```
# 定义展开的训练输入
for ui in range(dec_num_unrollings):
    dec_train_inputs.append(tf.placeholder(tf.float32,
shape=[batch_size,input_size],name='dec_train_inputs_%d'%ui))
    dec_train_labels.append(tf.placeholder(tf.float32,
shape=[batch_size,vocabulary_size], name = 'dec_train_labels_%d'%ui))
    dec_train_masks.append(tf.placeholder(tf.float32,
shape=[batch_size,1],name='dec_train_masks_%d'%ui))
```

为了初始化 LSTM 细胞和 softmax 层的权重值，我们将使用 Xavier 初始化，由 Glorot 和 Bengio 于 2010 年在他们的论文《Understanding the Difficulty of Training Deep Feedforward Neural Networks, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (2010)》中引入。这是一种初始化技术的原理，旨在缓解深度网络的梯度消失问题，可以通过 TensorFlow 中提供的 tf. contrib. layers. xavier_initializer(变量初始化获得。具体来说，在 Xavier 初始化中，神经网络的第 j 层的权重值根据均匀分布 U[a，b]初始化，其中 a 是最小值，b 是最大值。

$$W \sim U\left[\frac{-\sqrt{6}}{\sqrt{n_j + n_{(j+1)}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{(j+1)}}}\right]$$

这里，n_j是第 j 层的大小。

5．定义端到端输出计算

通过定义变量和输入/输出占位符，我们将继续定义从编码器到解码器的输出计算及损失函数。对于输出，首先计算给定批次句子中所有单词的 LSTM 细胞状态和隐藏状态，这是通过运行 for 循环语句来实现

```
# 迭代地更新编码器的输出和状态
for i in enc_train_inputs:
    output, state = enc_lstm_cell(i, output,state)
```

的，其中在第 i 次迭代中，我们在 enc_train_inputs 中输入第 i 个占位符，从第 i-1 次迭代输入细胞状态和输出隐藏状态。

接下来，我们同样计算整个目标语句的解码器的输出。为了做到这一点，我们应该完成前面代码中所示的计算，以便可以获得 v 来初始化解码器状态，这是通过调用 tf.control_dependencies （...）来实现的。因此，with 语句中的嵌套命令仅在完成计算编码器输出后执行。

在计算解码器输出之后，我们将使用 LSTM 的隐藏状态作为层的输入来计算 softmax 层的 logits。

通过计算 logits 可以计算损失。注意我们使用掩码来掩盖不应该导致损失的元素（附加的元素</s>使得句子具有固定的长度）

```
# 完成 enc_lstm_cell 的计算后,计算解码器的输出和状态。
with tf.control_dependencies([saved_output.assign(output)
loss_batch =
tf.concat(axis=0,values=dec_train_masks)*tf.nn.softmax_cross_entropy_with_logits_v2(
    logits=logits, labels=tf.concat(axis=0, values=dec_train_labels))
loss = tf.reduce_mean(loss_batch)


# 我们使用两个优化器:Adam 和朴素 SGD
# 从整体上来看,一直使用 Adam 会产生一些不理想的结果,（例如）BLEU 的突然波动
# 因此,我们使用 Adam 来获得优化的良好起点,然后从该点起切换到 SGD 优化器
with tf.variable_scope('Adam'):
    optimizer = tf.train.AdamOptimizer(learning_rate)
with tf.variable_scope('SGD'):
    sgd_optimizer = tf.train.GradientDescentOptimizer(sgd_learning_rate)
# 利用裁剪方法计算 Adam 的梯度
gradients, v = zip(*optimizer.compute_gradients(loss))
gradients, _ = tf.clip_by_global_norm(gradients, 5.0)
optimize = optimizer.apply_gradients(zip(gradients, v))
# 利用裁剪方法计算 SGD 的梯度
sgd_gradients, v = zip(*sgd_optimizer.compute_gradients(loss))
sgd_gradients, _ = tf.clip_by_global_norm(sgd_gradients, 5.0)
sgd_optimize = optimizer.apply_gradients(zip(sgd_gradients, v))
```

与前几章不同，我们将使用两个优化器:Adam 和标准随机梯度下降。由于长期使用 Adam 会产生不良后果（如 BLEU 的突然波动），因此我们使用 Adam 来获得优化的良好起点，然后从那时起切换到 SGD。

我们将使用以下语句，通过确保对所有可训练变量存在梯度，确保梯度正确地从解码器流向编码器。

```
for (g_i,v_i) in zip(gradients,v):
    assert g_i is not None, ' Gradient none for %s '%(v_i.name)
```

注意，与之前的练习相比，运行神经网络机器翻译系统(NMT)会慢得多，而在单个 GPU 上，可能需要 12 个小时。

3.6.3 神经网络机器翻译系统运行结果

在 10000 步后获得如下结果。

DE： Der nordwestliche Teil der Insel besteht aus Granit und Gneis ， von Ton überlagert ， und bildet eine ca.

EN （TRUE）:A battle between Denmark and Sweden in 1645 led to Swedish control of the island ， but it was brief – they left again the same year ． In the Roskildepeace of 1658 Bornholm ， Skaane ， Halland and Blekinge were given to Sweden ．

EN （Predicted）： The <unk> <unk> is a small town of the <unk> ， which is the ideal place for a relaxing break ． </s>

DE： In dem traditionellen Fischerdorf werden ausländische Gäste sehr herzlich empfangen ． Seien Sie allerdings gewarnt vor dem Zustand der Strasse und fahren Sie vorsichtig ！

EN （TRUE）:It 's a small traditional fishing village ， with friendly locals ， always very welcoming to British tourists ．

EN （Predicted）： The hotel is located in the heart of the city ， with its own private swimming pool ， a bar and a bar ． </s>

DE： „ Für uns junge slowenische Architekten ist prägend ， wie Plečnik den öffentlichen Raum in Ljubljana zu gliedern verstand ．

EN （TRUE）:" For us young Slovene architects the way in which Plečnik was able to shape public space in Ljubljana is highly influential ．

EN （Predicted）： <unk> .com ： <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> ） ． </s>

DE： Genießen Sie hier in gemütlicher Atmosphäre ein kühles Kölsch oder einen erfrischenden Cocktail ．

EN （TRUE）:Enjoy a local beer or refreshing cocktail in a cosy atmosphere ．

EN （Predicted）： Enjoy your holiday in the Costa Brava ， Spain ， Spain ， Spain ． </s>

我们可以看到第一句翻译得不错，但第二局翻译得很差

下面是运行 100000 步后获得的结果。

DE： In der Hotelbeschreibung im Internet müßte die Zufahrt beschrieben werden ．

EN （TRUE）:There are no adverse comments about this hotel at all ．

EN （Predicted）： The <unk> <unk> <unk> <unk> <unk> <unk> ． </s>

DE： Man schließt die Gitarre über Mikrofon oder Pickup an die PC Soundkarte an und schon kann es losgehen ．

EN （TRUE）:You connect the guitar via microphone or pickup with the PC soundcard and you are ready ！

EN （Predicted）： You can also use the <unk> <unk> <unk> （ <unk> ） ， which is a bit of a few minutes ． </s>

DE： Leicht und ergonomisch gebaut ， mit einer Hand zu bedienen ， stellen diese Messgeräte eine wirtschaftliche Lösung dar ， wenn bei Verdacht auf Wanddickenverlust schnell geprüft werden soll ．

EN （TRUE）:Lightweight and ergonomically designed for easy one ##AT##-##AT## hand operation ， these gages provide cost ##AT##-##AT## effective measurement solutions in many applications that require quick inspection of materials suspected of metal wall thinning ．

EN （Predicted）： <unk> ， the <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk>

DE： Wenn eine Speicherung der Daten auf dem Client erfolgen soll ， werden Cookys verwendet ．

EN （TRUE）:When client data storage is needed ， cookies are used ．

EN （Predicted）： If the use of the use of the use of the <unk> ． </s>

我们可以看到，尽管翻译并不算完美，但是大多数时候它却可以捕捉到源语句的上下文，并且这里的神经网络机器翻译系统也善于生成语法正确的句子。

图 3.21 描绘了神经网络机器翻译系统随时间变化其 BLEU 得分的情况。随着时间的推移，训练和测试数据集的 BLEU 得分明显增加。

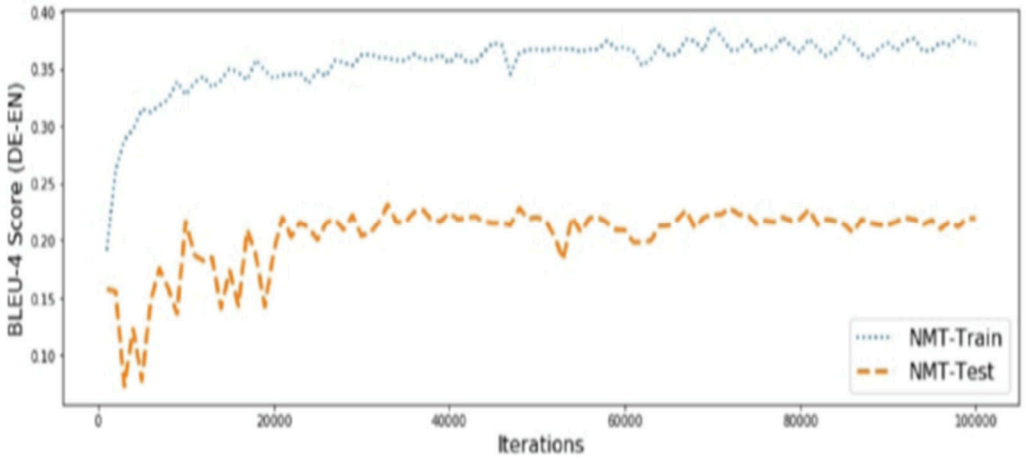


图 3.21 翻译系统的 BLEU 得分

3.6.4 结合词向量训练神经网络机器翻译系统

在本节中，我们将讨论如何结合词向量来训练神经网络机器翻译（NMT）系统，这个训练涉及以下两个方面：与词嵌入层一起训练神经网络机器翻译系统。使用预训练词向量，而不是随机初始化的词嵌入层。有几种多语言的词向量存储库可供使用：

1） Facebook's fastText: https://github.com/facebookresearcl/fastTextblob/master/pretrained-vectors.md

2） CMU multilingual embeddings: http://www.cs.cmmL.edu-afmprojects/multilingual_cmbeddings.html

从现在开始，我们将使用 CMU 词向量（~200 MB），因为它与 fastText 文本（~5GB）相比要小得多。首先需要下载德语（multilingual_embeddings.de）和英语（multilingual_embeddings.en）词向量，这可以在 chll 文件夹的代码文件 1l_nmt_with_pretrained_wordvecs.ipynb 中找到。

1．最大化数据集词汇和预训练词向量之间的匹配

首先必须获得预训练词向量的子集，这些子集与我们有意要解决的问题相关。这很重要，因为预训练词向量的词汇表可能很大，并且包含许多在数据集词汇表中找不到的单词。预先训练的词向量是一行，其中在行中的第一个是单词，其他是由空格分隔的词向量。预训练词向量的示例行如下所示：

door 0.283259492301 0.198089365764 0.335635845187 -0.385702777914 0.491404970211 ...

而实现这一目标的一种直白且朴素的方法是逐行运行预训练数据集词汇，如果发现当前行中的单词与数据集词汇表中的单词匹配的话，那么我们将该单词予以保存以便后续进行使用。但是这将是非常低效的，因为一般情况下词汇往往会倾向于设计者给出的各类设计策略。例如，有些人可能认为 cat's、cat 和 Cta 是同一个单词，而其他人则可能认为它们是独立的单词。如果我们只是一味地匹配预训练词向量词汇和数据集词汇，那么结果可能会错过很多单词。因此，我们将使用以下逻辑来确保能够获得预训练词向量中的大多数词向量。

首先，我们将定义两个 NumPy 数组来保存源语言和目标语言的关联词向量。

```
# 随机初始化德语和英语嵌入
de_embeddings = np.random.uniform(size=(vocabulary_size, embeddings_size), low=-1.0,
high=1.0)
en_embeddings = np.random.uniform(size=(vocabulary_size, embeddings_size), low=-1.0,
high=1.0)
```

然后，打开包含词向量的文本文件，如下所示。filename 参数是针对德语的 multilingual_embeddings.de

```
with open(filename,'r',encoding='utf-8') as f:
```

和英语的 miltilingual_embeddings.en。

接下来将通过空格分割行来分隔单词和词向量。我们将编写一组级联条件来检查源语言和目标语言的匹配。首先检查来自预训练司向量(lword）的单词是否在数据集词汇表中。如果有，则检查第一个字母是否大写(cat 变为 Cat 的情形），且假设我们能够在数据集词汇表中找到它。

如果没有，则通过从数据集词汇单词中删除特殊字符(如重音符号)，以检查来自预训练词向量(lword）

```
# 用空格对行进行分隔
            line_tokens = line.split(' ')

# 为嵌入更新随机初始化的矩阵
            # 更新与预训练嵌入匹配的单词数量
            try:
                dword = dictionary[lword]
                words_found_ids.append(dictionary[lword])
                embeddings[dictionary[lword],:] = vector
                words_found += 1
            # 对这个词进行解码以消除重音
            lword = unidecode.unidecode(lword)
            # 获取向量
            vector = [float(v) for v in line_tokens[1:]]
```

的单词是否与任何单词相似。如果满足其中一个条件,我们将获得该词向量，并将其分配给由该单词 ID 索引的行(word→ID)映射存储在 src_dictionary 和 tgt_dictionary 中（源语言和目标语言）。接下来，执行下面的操作:

```
# 如果一个指定单词未出现在词汇表中
            except KeyError:
                try:
                    if len(lword)>0:
                        firt_letter_cap = lword[0].upper()+lword[1:]

                    else:
                        continue

                    # 更新词嵌入矩阵
                    dword = dictionary[firt_letter_cap]
                    words_found_ids.append(dictionary[firt_letter_cap])
                    embeddings[dictionary[firt_letter_cap],:] = vector
                    words_found += 1

                except KeyError:
                    # 如果未找到,请尝试将该单词与非重音单词匹配
                    try:
                        dword = unaccented_dict[lword]
                        words_found_ids.append(dictionary[lword])
                        embeddings[dictionary[lword],:] = vector
                        words_found += 1
                    except KeyError:

                        continue
```

2．为词嵌入层定义 TensorFlow 变量

我们将定义两个可训练的 TensorFlow 变量，用于词嵌入层（即 tgt_word_embeddings 和 src_word_cmbeddings），如下所示。

```
tgt_word_embeddings = tf.get_variable(
    'target_embeddings',shape=[vocabulary_size, embeddings_size],
    dtype=tf.float32, initializer = tf.constant_initializer(en_embeddings)
)
src_word_embeddings = tf.get_variable(
    'source_embeddings',shape=[vocabulary_size, embeddings_size],
    dtype=tf.float32, initializer = tf.constant_initializer(de_embeddings)
)
```

接着，我们将 dec__inputs 和 enc__inputs 中占位符的维度更改为[batch_size]，并将数据类型更改为 tf. int32。因此，我们可以使用它们对每个展开的输入执行词向量查找(tf. nn. cmbeddimg, lookup(. . .)，如下所示。

```
# 定义展开的训练输入以及嵌入查找（编码器）
for ui in range(enc_num_unrollings):
    enc_train_inputs.append(tf.placeholder(tf.int32, shape=[batch_size],name='train_inputs_%d'%ui))

enc_train_input_embeds.append(tf.nn.embedding_lookup(src_word_embeddings,enc_train_inputs[ui]))

# 训练输入数据和相应的嵌入向量
dec_train_inputs, dec_train_input_embeds = [],[]
# 训练输出数据（用于优化）
dec_train_labels = []
# 用于在计算损失期间屏蔽不相关的单词
dec_train_masks = []

# 定义展开的训练输入、嵌入、输出和掩码（解码器）
for ui in range(dec_num_unrollings):
    dec_train_inputs.append(tf.placeholder(tf.int32, shape=[batch_size],name='dec_train_inputs_%d'%ui))
    dec_train_input_embeds.append(tf.nn.embedding_lookup(tgt_word_embeddings,
dec_train_inputs[ui]))
    dec_train_labels.append(tf.placeholder(tf.float32, shape=[batch_size,vocabulary_size], name =
'dec_train_labels_%d'%ui))
    dec_train_masks.append(tf.placeholder(tf.float32,
shape=[batch_size,1],name='dec_train_masks_%d'%ui))
```

编码器和解码器的 LSTM 细胞计算如下所示进行变化。在这部分中，我们首先使用源语句作为输入以便计算编码器 LSTM 细胞的输出，然后通过使用出自于编码器的最终状态信息作为解码器的初始化状态（调用 tf. control_dependencies(..)）。计算解码器输出、计算 softmax logits 和预测：

```
# 更新所有输入的输出和状态值
for i in enc_train_input_embeds:
    output, state = enc_lstm_cell(i, output,state)
    # 将所有输出值累加到一个列表中
    enc_outputs.append(output)

print('Calculating Decoder Output with Attention')
# 在开始解码器计算之前，确保已经对编码器输出进行计算
with tf.control_dependencies([saved_output.assign(output),
                              saved_state.assign(state)]):
```

```
# 解码器展开期间进行迭代
    for ii,i in enumerate(dec_train_input_embeds):

        # 计算每个解码位置的注意力值
        c_i,_ = attn_layer(enc_outputs, output)

        # 在列表中累积 c_i
        context_vecs.append(c_i)

        output, state = dec_lstm_cell(i, output, state, c_i)

        # 在列表中累积解码器输出
        dec_outputs.append(output)
```

请注意，代码文件的输出计算与此处显示的略有不同。我们不提供先前的预测作为输入，而是提供真实的单词作为输入，这倾向于提供比先前预测中更好的性能，会在下一节中详细讨论。
最后一个步骤包括计算解码器的损失和定义优化器以优化模型参数，如前所述。

【289-297】
联邦学习 p116（联邦自然语言处理部分）

随着 DNN 的发展,自然语言处理(Natural Language Processingnp，NLP)模型在诸多领域的任务中取得了瞩目的成就。在这些神经网络模型中，能够有效处理序列信息的循环神经网络（RNN）极大的提高了语言建模的性能。著名的 RNN 变体有长短时记忆（LSTM）和门控循环单元（Gated Recurrent Unit, GRU）。然而,这些方法需要将许多用户生成的训练数据聚合到一个中央存储站点中。在
真实的场景中，用户的自然语言数据是敏感的，可能包含隐私内容。因此，为了在保护用户隐私的同时，仍然能够建立健壮的 NLP 模型，我们可以使用联邦学习技术。

8.2.1 联邦自然语言处理
联邦学习在 NLP 中的一个典型应用是基于移动设备用户频繁键入的单词来学习词库外(Out-of-Vocabulary)单词。词库外单词是指不包含在用户移动设备的词库表中的词汇。词库表中缺少的单词无法通过键盘提示、自动更正或手势输入来预测。从单个用户的移动设备学习 OOV 单词来生成模型是不切实际的，因为每个用户的设备通常只会存储有限大小的词库表。收集所有用户的数据来训练 OOV 单词生成模型也是不可行的，因为 OOV 单词通常包含用户的敏感内容。在这种场景中，联邦学习显得特别实用，因为它可以根据所有移动用户的数据，训练一个共享的 OOV 生成模型，并且不需要将敏感内容传输到中心服务器或云服务器上。

任何序列化模型,例如 LSTM、GRU 和 WaveNet,都能用来学习 OOV 单词。
联邦 OOV 模型训练流程类似于图 8-2。该流程迭代地执行以下步骤来训练共享 OOV 生成模型，知道模型收敛为止。
步骤 1 每一台移动设备从服务器下载共享模型。
步骤 2 每一台移动设备基于用户输入的内容,训练共享模型。
步骤 3 每一台移动设备将模型更新,再通过安全协议上传至服务器。
步骤 4 服务器从移动设备收集更新,聚合这些更新并以此改良共享模型。

在联邦学习期间，位于每一台移动设备的 OOV 生成模型将不断得到更新，而训练数据将留在设备中，所以

每台移动设备最终都能得到一个更强大的 OOV 生成模型。如图 8-3 所示，基于所有移动用户数据而训练得到的 OOV 模型，能够为每个移动端用户提供丰富多样的查询建议。需要注意的是，用户可以完全决定自己加入或离开联邦学习，所以，服务器应该设立一种分析机制，以监测设备的相关统计数据，例如每轮训练中有多少台设备加入或离开联邦学习过程。感兴趣的读者可以参考文献【64】，了解关于设计可扩展联邦学习系统的更多细节内容。

8.2.2 业界研究进展

除了学习联邦 OOV 生成模型，联邦学习还可应用于设备上的唤醒词检测(wake--word detection)。例如，对 iPhone 说“，Siri！”来唤醒 iPhone 的语音识别和语言理解模块。唤醒词检测是智能设备上实现基于语音的用户交互的关键组件。由于它们一直处于运行状态，唤醒词检测应用程序必须只消耗非常有限的能源预算，并且通常运行在内存和计算资源有限的微控制器上。此外，它们必须在各种复杂情况下保持行为的一致性，并且对于背景噪声有强鲁棒性。再者，它们应该在命令捕捉上具有很高的召回度，误报率也需要达到非常低的水平。

Snips 最近发表了一篇研究基于众包数据集如何使用联邦学习来训练一个资源有限的唤醒词检测器的论文。该众包数据集模拟了真实世界中数据是非独立同分布、不平衡且高度分散的设置条件。受到 Adam293］的启发，这一研究通过自适应平均策略优化了联邦平均算法。得益于这种优化，联邦学习算法在每个客户上的通信开销仅为 8MB，并且能在 100 个通信轮次内收敛。

注意力机制已经被广泛用于序列到序列(sequence--sequence-to)的 NLP 任务中，如神经语言翻译和图像标题生成。最近的一项工作将注意力机制引入移动键盘输入建议的联邦学习中——注意力联邦聚合(the Attentive Federated Aggregation，FedAtt)方法。不同于传统的用于数据流的注意力机制， Fed Att 方法将服务器模型参数作为查询对象，以客户模型参数作为键值，计算每个客户端的 GRU 神经网络各层相对于服务器 GRU 神经网络各层的注意力权值。之后，服务器通过聚合来自各客户端模型同一层的注意力加权参数，更新服务器模型的每一层。 Fed Att 带来的最大好处是，我们可以通过客户端模型的细粒度聚合，对服务器模型进行微调以达到更好的泛化能力。

8.3.3 挑战与展望

在 NLP 领域中，最主要的方法是监督学习。为了在未遇见过的数据上得到良好的泛化性能，监督学习对于每一个新场景都需要足够多的标注训练数据。手工标注每一条训练数据毫无疑问将耗费大量时间，并且工作非常繁重枯燥。联邦学习通过利用来自许多参与方的数据，可以在一定程度上解决这个问题。然而，对于标注数据极其稀缺的场景，所有带标记的和不带标记的可用数据都应该被利用起来。

联邦学习与无监督学习、半监督学习或迁移学习的结合是解决数据稀缺问题的个很有市场的研究方向。尤其是在 NLP 领域，大量数据都是未经标注的。如何有效地利用这些数据，目前已是一个有趣而赋有挑战性的研究课题。研究人员在跨语言学习 295、多任务学习 296 及多源域适应 297 等非联邦学习环境中已经开展了许多创新性的研究工作。在联邦学习设定下，如何有效地利用未标注数据将是更赋有挑战性的课题。例如，在分布式环境中，我们需要确定如何从未标注数据中学习可迁移或可分离表征，同时需要保护参与方的隐私安全。此外，我们需要仔细设计或选择安全协议，从而在安全和效率之间达到一种平衡。在发展联邦学习的过程中，我们一定还会遭遇许多其他的挑战和困难。但是，我们设基于联邦学习的 NLP 的发展将会显著地推动人工智能在诸多行业中的应用。

参考文献

Chen, Stanley F, Goodman, et al. An empirical study of smoothing techniques for language modeling[J]. Computer Speech & Language, 1996, 13(4):310-318.

Liu D C, Nocedal J. On the limited memory BFGS method for large scale optimization[J]. Mathematical Programming, 1989, 45(1-3):503.528.

Rendle S. Factorization Machines[C]// IEEE, International Conference on Data Mining. IEEE, 2011 :995-1000.

Mcmahan H B, Streeter M. Adaptive Bound Optimization for Online Convex Optimization[J]. Computer Science, 2012, 7(2):163.71.

Mcmahan H B. Follow-the-Regularized-Lead已r and Mirror Descent: Equivalence Theorems and Ll Regularization[J]. Jmlr, 2011 , 15:2011.

Mcmahan H B, Holt G, Sculley D, et al. Ad click prediction: a view from the trenches[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013 :1222-1230.

Berger A. The improved iterative scaling algorithm: A gentle introduction[J]. Unpublished Manuscript, 1997.

Wallach HM. Conditional Random Fields: An In位oduction[J]. Technical Reports, 2004, 53(2): 267-272.

Marti, Nez C, Prodinger H. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms[C]// Acl-02 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2002: 1-8.

Winn J M. Variational Message Passing and its Applications[J]. University of Cambridge, 2004, 6(2):661--694.

Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of Machine Learning Research, 2003, 3:993.1022.

Heinrich G. Parameter Estimation for Text Analysis[J]. Technical Rep。此， 2008. Borman S. The Expectation Maximization Algorithm A short tutorial[J]. 2009. Griths T. Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation [J].

Standford University, 2002. Collobert R, Weston J, Karlen M, et al. Natural Language Processing (Almost) from

Scratch[J]. Journal of Machine Learning Research, 2011 , 12(1):2493.2537. Le QV, Mikolov T. Distributed Representations of Sentences and Documents[J]. 2014,

4:11-1188. Li J, Hovy E. A Model of Coherence Based on Distributed Sentence Representation[C]//

Conference on Empirical Methods in Natural Language Processing. 2014:2039-2048. Kiros R, Zhu Y, Salakhutdinov R, et al. Skip-Thought Vectors[巧. Computer Science,

2015 . Pascanu R, Mikolov T, Bengio Y. On the difficulty of training Recurrent Neural

Networks[耳 Computer Sci已nce, 2012, 52(3):III-1310. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization[J]. Computer Science,

2014. Duchi J, Hazan E, Singer Y. Adaptive Subgradient Methods for Online Learning and

Stochastic Optimization[J]. Journal of Machine Learning Research, 2011 , 12(7):257-269. Zeiler M D. ADADELTA : An Adaptive Learning Rate Method[J]. Computer Science,

2012 . Tieleman T, Hinton G. RMSProp: Divide the gradient by a running average of its recent

magnitude. COURSERA: Neural Networks for Machine Learning. Technical report, 2012. Cho K, Merrienboer B V, Gulcehre C, et al. Learning Phrase Representations using

RNN Encoder-Decoder for Statistical Machine Translation[J] . Computer Science, 2014. Koutnik J, Greff K, Gomez F, et al. A Clockwork RNN[J] . Computer Science,

2014:1863.1871.

Greff K, Srivastava R K, Koutnik J, et al. LSTM: A Search Space Odyssey[J]. IEEE Transactions on Neural Networks & Learning Systems, 2017, PP(99): 1-11.

Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Leaming to Align and Translate[J]. Computer Science, 2014.

Ghosh S, Vinyals 0, Strope B, et al. Contextual LSTM (CLSTM) models for Large scale NLPtasks[月. 2016.

Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]// International Conference on Neural Information Processing Systems. MIT Press, 2014:2672- 2680.

Creswell A, White T, Dumoulin V, et al. Generative Adversarial Networks: An Overview[J]. 2017.

Salimans T, Goodfellow I, Zaremba W, et al. Improved Techniques for Training GANs[巧. 2016.

Sutskever I, Vinyals 0 , Le QV. Sequence to sequence learning with neural networks[J]. 2014, 4:3104-3112.

Vinyals 0 , Le Q. A Neural Conversational Model[J]. Computer Science, 2015.

Shang L, Lu Z, Li H. Neural Responding Machine for Short-Text Conversation[J]. 2015: 52-58.

Yao K, Zweig G, Peng B. Attention with Intention for a Neural Network Conversation Model[J]. Computer Science, 2015.

Pascual B, Gurruchaga M, Ginebra M P, et al. A Neural Network Approach to Context- Sensitive Generation of Conversational Responses[J]. Transactions of the Royal Society of Tropical Medicine & Hygiene, 2015, 51(6):502-504.

Li J, Galley M, Brockett C, et al. A Diversity-Promoting Objective Function for Neural Conversation Models[J]. Computer Science, 2015 .

Mou L, Song Y, Yan R, et al. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation[巧. 2016.

Li J, Monroe W, Ritter A, et al. Deep Reinforcement Learning for Dialogue Generation[J]. 2016.

Li J, Galley M, Brockett C, et al. A Persona-Based Neural Conversation Model[J]. 2016.

Lowe R, Pow N, Charlin L, et al. Incorporating Unstructured Textual Knowledge Sources into Neural Dialogue SystemsLiJ. 2015.

Su P H, Gasic M, Mrksic N, et al. On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems[J]. 2016:2431 -2441.

Iyyer M, Manjunatha V, Boyd-Graber J, et al. Deep Unordered Composition Rivals Syntactic Methods for Text Classification[C]// Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2015: 1681-1691.

Li S, Jouppi NP, Faraboschi P, et al. MEMORY NETWORK:, W0/2014/178856[P]. 2014.

Sukhbaatar S, Szlam A, Weston J, et al. End-To-End Memory Networks[J]. Computer Science, 2015.

Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend[J]. 2015: 1693.1701.

Kadlec R, Schmid M, Bajgar 0, et al. Text Understanding with the Attention Sum Reader Network[J]. 2016:908-918.

Lowe R, Pow N, Serban I, et al. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Tum Dialogue Systems[J]. Computer Science, 2015.

Yu L, Hermann KM, Blunsom P, et al. Deep Learning for Answer Sentence Selection[J]. Computer Science, 2014.

Feng M, Xiang B, Glass M R, et al. Applying Deep Leaming to Answer Selection: A Study and An Open Task[J] . 2015:813.820.

Tan M, Santos CD, Xiang B, et al. LSTM-based Deep Leaming Models for Non-factoid Answer Selection[J]. Computer Science, 2015.

Yin W, Schutze H, Xiang B, et al. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs[J]. Computer Science, 2015.

Bordes A, Weston J, Usunier N. Open Question Answering with Weakly Supervised Embedding Models[J]. 2014, 8724:165-180.

Bordes A, Chopra S, Weston J. Question Answering with Subgraph Embeddings[J]. Computer Science, 2014.

Friedman J H. Greedy Function Approximation: A Gradient Boosting Machine[J]. Annals of Statistics, 2001, 29(5):1189-1232.

Mohan A, Chen Z, Weinberger K. Web-search ranking with initialized gradient boosted regression trees[C]// International Conference on Yahoo! Learning To Rank Challenge. 几lfLR. org, 2010:77-89.

Cortes C, Vapnik V. Support-vector networks[J]. Machine Leaming, 1995, 20(3):273. 297.

Statnikov A, Aliferis C F, Hardin D P, et al. A Gentle Introduction to Support Vector Machines in Biomedicine[M]. WORLD SCIENTIFIC, 2013.

Yang Y, Pedersen J 0. A Comparative Study on Feature Selection in Text Categorization[C]// Fourteenth International Conference on Machine Learning. Morgan Kau企nano Publishers Inc. 1997:412-420.

Leung S T, Leung S T, Leung S T. The Google file system[C]// Nineteenth ACM Symposium on Operating Systems Principles. ACM, 2003 :29-43 .

Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Conference on Symposium on Opearting Systems Design & Implementation. USENIX Association, 2008:I0-10.

Amy N. Langville, Carl D. Meyer. Deeper Inside PageRank[J]. Internet Mathematics, 2004, 1(3):335-380.

Robertson S, Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond[J] . Foundations & Trends(in Information Retrieval, 2009, 3(4):333.389.

Rasolofo 飞 Savoy J. Term Proximity Scoring for Keyword-Based Retrieval Systems[J]. Lecture Notes in Computer Science, 2002, 2633:79-79.

Hofmann T. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization[J]. Advances in Neural Information Processing Systems, 1999:914-920.

Hakkani-Tur D, Hakkani-Tur D, Tur G. LDA based similarity modeling for question answering[C]// NAACL Hit 2010 Workshop on Semantic Search. Association for Computational Linguistics, 20I0:1-9.

Diaz, Fernando. Regularizing ad hoc retrieval scores[耳 2005:672-679.

Huang PS, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data[C]// ACM International Conference on Conference on Information & Knowledge Management. ACM, 2013:2333.2338.

Chapelle 0, Zhang Y. A dynamic bayesian network click model for web search ranking[C] // International Conference on World Wide Web. ACM, 2009: 1-10.

Agrawal R, Gollapudi S, Halverson A, et al. Diversifying search results[C)// Acm International Conference on Web Search & Data Mining. ACM, 2009:5-14.

Rafiei D, Bharat K, Shukla A. Diversifying web search results[C]// International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, Usa, April. DBLP, 2010:781-790.

Welch M J, Cho J, Olston C. Search result diversity for informational queries[C]// International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April. DBLP, 2011 :237-246.

Li L, Jin Y K, Zitouni I. Toward Predicting the Outcome of an *NB* Experiment for Search Relevance[C]// Eighth ACM International Conference on Web Search and Data Mining. ACM, 2015:37-46.

Zhang Y, Dai H, Kozareva Z, et al. Variational Reasoning for Question Answering with Knowledge Graph[J]. 2017.

Loshchilov I, Hutter F. SGDR: Stochastic Gradient Descent with Warm Restarts[J]. 2016.

Burges C, Shaked T, Renshaw E, et al. Learning to rank using gradient descent[C]// International Conference on Machine Learning. ACM, 2005:89-96.

Burges CJ C. From ranknet to lambdarank to lambdamart: An overview[J]. Learning, 2010, 11.

Pugh W. Skip lists: A probabilistic alternative to balanced trees[C]// Algorithms and Data Structures, Workshop WADS '89, Ottawa, Canada, August 17-19, 1989, Proceedings. DBLP, 1990:437-449.

Broder A Z, Carmel D, Herscovici M, et al. Efficient Query evaluation using a two- level retrieval process[C]// Twel他 International Conference on Information and Knowledge Management. ACM, 2003:426-434.

Yin D, Hu 飞 Tang J, et al. Ranking Relevance in Yahoo Search[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016:323.332.

Mitra B, Craswell N. Neural Models for Information Retrieval[J]. 2017.

Jeh G, Widom J. SimRank: a measure of structural-context similarity[C]// Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2002:538-543.

He X, Pan J, Jin 0 , et al. Practical Lessons from Predicting Clicks on Ads at Facebook[J] . 2014(12): 1-9.

Gai _K, Zhu X, Li H, et al. Learning Piece-wise Linear Models from Large Scale Data for Ad Click Prediction[J]. 2017.

Mcmahan H B, Holt G, Sculley D, et al. Ad click prediction: a view from the trenches[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013:1222-1230.

Lemire D, Maclachlan A. Slope One Predictors for Online Rating-Based Collaborative Filtering[巧. Computer Science, 2007:21--23.

Das A S, Datar M, Garg A, et al. Google news personalization: scalable online collaborative filtering[C]// International Conference on World Wide Web. ACM, 2007:271- 280.

Sarw缸 B M, Karypis G, Konstan J A, et al. Application of Dimensionality Reduction in Recommender System . . . A Case Study[C]// ACM Webkdd Workshop. 2000.

Bendersky M, Garcia-Pueyo L, Harmsen J, et al. Up next: retrieval methods for large scale related video suggestion[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2014:1769-1778.

Covington P, Adams J, Sargin E. Deep Neural Networks for YouTube Recommendations[C]// ACM Conference on R巳commender Systems. ACM, 2016:191-198.

Cheng H T, Koc L, Harmsen J, et al. Wide & Deep Learning for Recommender Systems[月. 2016:7-10.

Teo C H, Hill D, Hill D, et al. Adaptive, Personalized Diversity for Visual Discovery[C]// ACM Conference on Recommender Systems. ACM, 2016:35-38.

Castells P, V缸gas S, Wang J. Novelty and Diversity Metrics for Recommender Systems:

Choice, Discovery and Relevance[J]. In Proceedings of International Workshop on Diversity in Document Retrieval (DDR, 2009:29--37.

Li L, Wei C, Langford J, et al. A contextual-bandit approach to personalized news article recornmendation[J]. 2010:661-670.

Meng R, Zhao S, Han S, et al. Deep Keyphrase Generation[J]. 2017.

Liu J, Dolan P. Personalized news recommendation based on click behavior[C]// International Conference on Intelligent User Interfaces. ACM, 2010:31-40.

Chu W, Park S T. Personalized recommendation on dynamic content using predictive bilinear models[C]// International Conference on World Wide Web. ACM, 2009:691-700.

Li L, Wei C, Langford J, et al. A contextual-bandit approach to personalized news article recommendation[J]. 2010:661-670.

Kouki AB. Recommender system performance evaluation and prediction an information retrieval perspective[J]. 2012.

Said A. Evaluating the Accuracy and Utility of Recommender Systems[J]. Prof. Dr.-Ing. habil. Dr. h. c. Sahin Albayrak, 2013.

Shani G, Gunawardana A. Evaluating Recommendation Systems[J]. Recommender Systems Handbook, 2011 :257-297.

Singhal A, Sinha P, Pant R, Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works[C]// Published with International Journal of Computer Applications. IJCA, 2017: 17-22.

Lample G, Ballesteros M, Subramanian S, et al. Neural Architectures for Named Entity Recognition[J]. 2016:260-270.

Chen D, Manning C. A Fast and Accurate Dependency Parser using Neural Networks[C]// Conference on Empirical Methods in Natural Language Processing. 2014:740- 750.

Berant J, Liang P. Semantic Parsing via Paraphrasing[C]// Meeting of the Association for Computational Linguistics . 2014: 1415-1425 .

Berant J, Chou A, Frostig R, et al. Semantic parsing on freebase from question-answer pairs[J]. Proceedings ofEmnlp, 2014.

Cai Q, Yates A. Large-scale Semantic Parsing via Schema Matching and Lexicon

习题集

第一章

1. 简述 NLP 的发展历史
2. 如何结合深度学习来完成专名识别？
3. 句法分析有哪些方法？
4. 如何建立知识图谱？如何量化知识？
5. 理解语言有哪些难点？
6. 语言如何量化？
7. Seq2Seq 模型在完成对话系统时有哪些问题？
8. 人工智能在现阶段真的很可怕吗？
9. 如何做特征工程？文本分类的特征选择有哪些方法？
10. LSTM 为何比 RNN 好？
11. CNN 模型的特点是什么？
12. 如何训练一词多义的词向量？
13. 如何评价一个机器学习模型？
14. 句子表示成向量的意义是什么？
15. 规则分词常用的方法有哪些？
16. 中文文本的处理步骤包含哪几个部分？

第二章