Draft

自然语言处理导论

张奇 桂韬 黄萱菁

February 15, 2022

目录

绪论 1

1.1 拟定章节

目录:

附录:

附录 4 条件随机场（CRF）
附录 5 卷积神经网（CNN）
附录 6 递归神经网（RNN+LSTM）
附录 7 Transformer 模型
附录 8 图网络（GCN 等）
附录 9 预训练方法

语言模型 2

语言模型介绍

2.1 语言模型概述

语言模型基本概念

语言模型历史沿革

语言模型主要任务

2.2 n 元语言模型

任务定义和概述

基础模型

数据平滑

加法平滑 Good-Turing 估计法 Katz 平滑 KN 平滑

2.3 神经语言模型

循环神经网络语言模型

XXX

2.4 预训练语言模型

单向预训练模型

GPT

双向预训练模型

Elmo

掩码预训练模型

BERT

预训练语言模型应用

2.5 延伸阅读

2.6 习题

词法分析 3

词汇是语言知识中的重要环节，在语言学中，**词（Word）**是形式和意义相结合的单位 [1]，也是语言中能够独立运用的最小单位。懂得一个词意味着知道某个的特定读音并与特定语义关联。在书面语中正字法（Orthography）也是词的形式的一种表达。例如：英文单词 “cat’’ 具有语义是 ‘‘猫’’，读音为 “/kæt/’’。由于词是语言运用的基本单位，因此自然语言处理算法中词通常也是基本单元。词的处理也由此成为自然语言处理中重要的底层任务，是句法分析、文本分类、语言模型等任务的基础。

本章首先介绍语言学中词相关的基本概念，在此基础上以介绍词形分析算法，中文分词算法，以及词性分析算法。

3.1 语言中的词汇

词通常是由语素（Morpheme）构成。**语素**是一个语言中意义的最小单元。语素与词不同，语素不能够独立运用而词可以。只包含一个语素的词语称为**简单词（Simple Wold）**，而包含多个语素的词称为**复杂词（Complex Word）**。例如：‘‘电灯’’，包含 ‘‘电’’ 和 ‘‘灯’’ 两个语素此外，根据词在语言中的用途的不同，词还可以被被划分成为**实义词（Content Words）**和**功能词（Function Words）**。实义词包含事物、行为、属性和观念等概念。功能词则是指没有清楚词汇意义或与之有关的明显的概念的词。本节将分别针对语素如何构成词以及如何对词进行分类进行介绍。

语素又称词素

词的形态学

虽然单词的形式和意义之间的关系本质上是任意的，但是由于社会的约定俗成，词的形式具有服从于某种规则的内在结构。在语言学中，研究单词的内部结构和其构成方式的学科称为**形态学（Morphology）**。词是由一个或多个语素构成，语素主要分成两类: **词根（Lemma）**和**词缀（Affix）**。词根也称为原形或字典形，指能在字典中查到的语素，通常是一个词最主要的语素。词缀是其他附着在原形上语素，帮助在原形基础上衍生出新词，包含前缀、中缀、后缀等。例如：

形态学又称为构词学

英语单词 unhappy 中，happy 为原形，-un为前缀
邦托克语单词 fumikas（是强壮的）中，fikas（强壮）为原形，-um-为中缀
俄语单词 barabanshchik（鼓手）中，baraban（鼓）为原形，-shchik为后缀

一个词也可以包含多个词缀，例如：unhappiness 包含前缀 “un-’’ 和后缀 “-ness’’。同样，一个词也可以包含多个词根，例如：homework 包含词根 “home’’ 和 “work’’。

Morphology 本身就是由两个语素构成: morph+ology。后缀-ology 表示 “关于... 的科学”

有些语言的单词通常只包含一个或者两个语素，但是有一些语言的单词则包含多达十个以上的语素。中文中每个单词的语素都很少，也不会根据性、数、格、人称等发生形态变化。但是对于英文，在单词 dog 末尾添加 s 可以将它从单数名词变成复数名词 dogs，对于德语单词 bäcker 末尾添加 in 可以将它从阳性词（男面包师）变为阴性词 bäckerin（女面包师）。不同语言的词形变化差别非常大，以英语为例，很多英语词都包含两个或两个以上的语素，其词形变化主要有以下几种方式:

- **屈折（Inflection）**是指通过 ‘‘词根 + 词缀’’ 的方式构成和原形 ‘‘同一类型’’ 的词。同一类型指词义和词性没有发生明显的变化，或者说通过屈折变化得到的词的词义与它的原形相似。例如：

  在名词后加 -s 后缀构成复数名词（cat+s）
  在动词后加 -ed 后缀构成动词的过去式（walk+ed）

- **派生（Derivation）**是指通过“词根 + 词缀”的方式构成和原形“不同类型”的词。例如：

  ```
  employ 添加后缀 -ee 变为 employee
  employ 添加后缀 -er 变为 employer
  meaning 添加后缀 -less 变为 meaningless
  ```

  可以看到，增加后缀后，词根的词义发生了较为明显的变化。此外，通过添加词缀的方式也可以使得词的词性发生变化。例如：

  ```
  形容词可以组合 -ize 后缀变为动词 (medical/medicalize)
  名词可以组合 -al 等后缀变为形容词 (sensation/sensational)
  ```

- **组合（Compounding）**是指通过组合多个词根构成一个新词。例如：

  组合词也称复合词

  ```
  homework 是由 home 和 work 组合而成
  waterproof 是由 water 和 proof 组合而成
  ```

  根据组合词构成部分之间的语义和语法关系又可以细分为修饰型组合词（Attributive Compounds）、并列型组合词（Coordinative Compounds）以及从属型组合词（Subordinative Compounds）。
- **附着（Cliticization）**是指“词根 + 附着语”的方式。附着语通常在语法上等同于一个词，通过特殊的方式“附着”在词根上。例如：

  ```
  I'm 中的 'm 代表 am 附着在 I 上
  We're 中 're 代表 are 附着在 We 上
  ```

- **截搭（Blending）**是指将两个词语各自的一部分拼接起来构成新词。例如：

  ```
  smoke（烟） 和 fog（雾） 组合成 smog（烟雾）
  spoon（勺子）和 fork（叉子） 组合成 spork（叉勺）
  ```

- **逆构（Backformation）**是指母语使用者将简单词感知为由多个语素所构

成，将简单词的部分构成新的词汇。例如：

```
edit（编辑，动词） 是由 editor（编辑，名词） 逆构而成
burgle（盗窃，动词）是由 burglar（盗窃，名次）逆构而成
```

editor 和 burglar 历史上都是简单词。

- **缩略（Acronym）**是指短语中多个单词首字母组合在一起构词过程。例如：

```
NLP 代表 Natural Language Processing
IT  代表 Information Technology
```

- **截短（Clipping）**是指将长的单词截为较短的单词。例如：

```
demonstration 简化为 demo
refrigerator 简化为 fridge
```

- **词语新造（Coinage）**是指完全新造一个词语。通常通常包括人名、产品名等。例如：

```
iPhone （手机品牌）
Raspberry Pi （嵌入式开发板卡品牌）
```

通过语素组成词汇也可以反映了语言的一个重要特性：创造性。我们可以理解从未见过的词，也可以通过新颖的方法将语素结合起来创造新词。如果能够自动将词汇分解为语素，可以更好的进行对词汇的进行进一步的分析。

词的词性

词性（Part of Speech，POS）是根据词在句子中扮演的语法角色以及与周围词的关系对词的分类。例如：通常表示事物的名字（“钢琴”），地点（“上海”）被归为名词，而表示动作（“踢”），状态（“存在”）的词被归为动词。对词性进行划分时通常要综合考虑词的语法特性的各个方面，以某一个标注为主，同时

词性也被称为词类

参照其他标准进行。通过词性可以大致圈定一个词在上下文环境中有可能搭配词的范围[1]，从而为语法分析、语义理解提供帮助。由此，词性也被称为带有“分布式语法”信息 (Syntactic distributional properties)。

1: 例如：介词 “in” 后面通常跟名词短语

现在语言学中一个重要的词的分类是区分实义词（Content Words）和功能词（Function Words）。**实义词**表达具体的意义。由于实义词可以不断的增加，因此这类词又被称作**开放类词** (Open class words) 。实义词主要包含名词、动词、形容词等。**功能词**则主要是为了满足语法功能需求。由于功能词相对比较稳定，一个语言中通常很少增加新的功能词，因此功能词又被称作**封闭类词** (Close class words)。功能词主要包含代词、冠词、指示词等。

以英语为例，词性主要包含以下几种:

- **名词（Noun）**是指表示人、物、地点以及抽象概念的一类词。名词按其意义又可以细分为专有名词（Proper noun）和普通名词（Common noun）。普通名词还可以再细分为类名词（Class Noun）、集体名词（Collective Noun）、物质名词（Material Noun）和抽象名词（Abstract Noun）。名词还可以按照其可数性分为可数名词（Countable Noun）和不可数名词（Uncountable Noun）。例如：

  1）专有名词：Shanghai（上海） New York（纽约）
  2）类名词： city（城市） bird（鸟）
  3）集体名词：family（家庭） army（军队）
  4）物质名次：water（水） light（光）
  5）抽象名词：music（音乐） honesty（诚实）

- **动词（Verb）**是指表示动作或状态的一类词，是英语中最复杂的一类词。动词除了具有人称和数的变化之外，还具备一些语法特征，包括：时态（tense）、语态（voice）、语气（mood）、体（aspect）等。动词可以进一步细分为及物动词（Transitive verb）、不及物动词（Intransitive verb）、连系动词（Linking verb）、助动词（Auxiliary verb）、限定动词（Finite verb）、不限定动词（Non-finite verbs）、短语动词（Phrasal verb）等。例如：

1）及物动词： Boys **fly** kites. （男孩们放风筝）
2）不及物动词：Birds **fly**. （鸟会飞）
3）连系动词： The rose **smells** sweet. （玫瑰花香）
4）助动词： I **may** have meet him before.（我以前应该见过他）
5）限定动词： John **reads** papers every day. （约翰每天都读论文）
6）不限定动词：I hope **to see** you this morning.（我希望早上见到你）
7）短语动词： Tom **called up** George.（汤姆给乔治打了电话）

- **形容词（Adjective）**是用来描写或修饰名词的一类词。按照构成，形容词可以被分为简单形容词和复合形容词。按照与其所修饰的名词的关系，形容词还可以被分为限制性形容词（Restrictive adjective）和描述性形容词（Descriptive adjective）。例如：

  1）简单形容词：
     a）由一个单词构成 good（好的） long（长的）
     b）由现在分词构成 interesting（令人感兴趣的）
     c）由过去分词构成 learned（博学的）
  2）复合形容词： duty-free（免税的） hand-made（手工制作的）
  3）限制性形容词：an **Italian** dish （一道意大利菜）
  4）描述性形容词：a **delicious** Italian dish （一道美味的意大利菜）

- **副词（Adverb）**是用来修饰动词、形容词、其他副词以及全句的词。按照形式，副词可以被细分为简单副词、复合副词和派生副词。按照意义，副词可以被细分为方式副词、方向副词、时间副词、强调副词等。按照句法作用，可以被分为句子副词、连接副词、关系副词等。例如：

  1）简单副词： just（刚刚） only（仅仅）
  2）复合副词： somehow（不知怎地） somewhere（在某处）
  3）派生副词： interesting → interestingly （有趣地）
  4）方式副词： quickly（快地） awkwardly（笨拙地）
  5）方向副词： outside（外面） inside（里面）
  6）时间副词： recently（最近） always（总是）

7）强调副词： very（很）　　fairly（相当）

- **数词（Numeral）**是表示数目多少或者先后顺序的一类词。表示数目多少的叫做的基数词（Cardinal numeral）。表示顺序先后的叫做序数词（Ordinal numeral）。例如：

  1）基数词： one（1）　　nineteen（19）
  2）序数词： first（第一）　fiftieth（第五十）

- **代词（Pronoun）**是代替名词以及起名词作用的短语、子句和句子的一类词。代词的词义信息较弱，必须通过上下文来确定。代词主要可以细分为人称代词（Personal pronoun）、物主代词（Possessive pronoun）、自身代词（Self pronoun）、相互代词（Reciprocal pronoun）、指示代词（Demonstrative pronoun）、疑问代词（Interrogative pronoun）、关系代词（Relative pronoun）和不定代词（Indefinite pronoun）。例如：

  1）人称代词：
     a）主格：I, you, he, she, it, we, they
     b）宾格：me, you, him, her, it, us, them
  2）物主代词：
     a）形容词性物主代词：my, your, his, her, its, our, their
     b）名词性物主代词：mine, yours, his, hers, its, ours, theirs
  3）自身代词： myself, yourself, himself, herself, itself, ourselves, yourselves, themselves, oneself
  4）相互代词： each other, one another
  5）指示代词： this, that, these, those
  6）疑问代词： who, whom, whose, which, what
  7）关系代词： who, whom, whose, which, that, as
  8）不定代词： some, something, somebody, someone, any, anything, anybody, anyone, no, nothing, nobody, no one

- **冠词（Article）**是置于名词之前，说明名词所指的人或事务的一种功能

词。冠词不能够离开名词而独立存在。英语中冠词有三个冠词：定冠词（Definite article）“the”、不定冠词（Indefinite article）“a/an”和零冠词（Zero article）。

- **介词（Preposition）**是用于表示名词或相当于名词的词语与句中其它词语的关系的一类词。介词在句子中不单独作任何句子成分。介词后面的名词或者相当于名词的词语叫做介词宾语，与介词共同组合成介词短语。从介词的构成来看，其主要包含简单介词（Simple preposition）、复合介词（Compound preposition）、二重介词（Double preposition）、短语介词（Phrasal preposition）、分词介词（Participle preposition）。例如：

介词又称前置词

  1）简单介词： `at, in, of, since`
  2）复合介词： `as for, as to, out of`
  3）二重介词： `from under, from behind`
  4）短语介词： `according to, because of`
  5）分词介词： `including, regarding`

- **连词（Conjunction）**是连接单词、短语、从句或句子的一类词。在句子中也不单独作为句子成分。按照其构成可以细分为简单连词（Simple conjunction）、关联连词（Correlative conjunction）、分词连词（Participial conjunction）、短语连词（Phrasal conjunction）。连词按照其性质可以分为等立连词（co-ordinative conjunction）、从属连词（Subordinative conjunction）。例如：

  1）简单连词： `and, or, but, if`
  2）关联连词： `both ... and, not only ... but also`
  3）分词连词： `supposing, considering`
  4）短语连词： `as if, as long as, in order that`
  5）等立连词： `and, or, but, for`
  6）从属连词： `that, whether, when, because`

- **感叹词（Interjection）**是用来表示喜怒哀乐等情绪或情感的一类词。感叹词也没有实义，也不能在句子中构成任何句子成分，但是与全句有关联。

例如：

Oh, it's you. 啊，是你
Ah, how pitiful! 呀，多可惜!

在语言学研究中，对于词性划分目的、标准、依据等都还存在大量分歧。到目前为止，还没有一个被广泛认可的统一划分标准。在不同的语料集中所采用的划分粒度和标记符号也都不尽相同。宾州大学句法树库（Penn TreeBank）使用了 48 种不同的词性，宾州大学汉语树库（Chinese Penn Treebank）中汉语词性被划分为 33 类，而 Brown 语料库 [2] 中则使用了具有 87 个词性。虽然在语言学中词性还具有很多的仍需要研究的内容，但是由于词性可以提供关于单词和其周边邻近成分的大量有用信息，词性分析也是自然语言处理中重要的基础任务之一。

[2]: Francis (1980), "A tagged corpus–problems and prospects"

3.2 词形分析

词是由语素构成，通过组成词语的语素可以在更好对词汇进行理解和分析。**词形分析 (Morphological Parsing)** 任务就是将一个词分解成为语素的过程。词形分析一个最简单的方法是将每一个词的所有词形变换都存储下来，使用时直接匹配查找。对于英语来说，一个包含所有词形的词典能够较为有效的支撑许多应用场景。但是由于用词方式变化和新词的不断出现，对这个字典需要进行及时维护。同时，对于一些语言（特别是土耳其语，阿拉伯语等黏着语）枚举所有词的词形变换则是不可能的。

例如：土耳其语词汇 uygarlatramadklarmzdanmsnzcasna 是由以下 10 项变换组合而成:

```
uygar      +la  +tr   +ama  +dk   +lar +mz   +dan  +m    +snz +casna
civilized  +BEC +CAUS +NABL +PART +PL  +P1PL +ABL  +PAST +2PL +AsIf
```

其中除了词根 uygar 以外，其他语素的含义如下 [3]：

[3]: Jurafsky et al. (2008), *Speech and Language Processing: An Introduction to speechrecognition, natural language processing and computational linguistics*

+BEC	‘‘变成”(become)
+CAUS	标识使役动词
+NABL	‘‘不能” (not able)
+PART	过去分词
+PL	名词复数
+P1PL	第一人称复数所有格
+ABL	表来源的离格 (ablative (from/among) case maker)
+PAST	带过去时的间接引语 (indirect/inferential past)
+AsIf	从限定动词 (finite verb) 派生出的副词

可以看到，由于词性变换的复杂性，一个词的原形可能衍生出很多不同的词。因此，设计更有效率的词形分析算法是十分必要的。

表 3.1: 词形分析输入输出示例

输入	**输出**
cats	cat+N+pl
cat	cat+N+sg
goose	goose+N+sg
walk	walk+N+sg
walk	walk+V
walks	walk+V+3sg
walking	walk+V+prespart
walked	walk+V+past
walked	walk+V+pastpart

表3.1给出了典型词形分析算法的输入和输出结果样例。从结果中可以看到，词形分析的结果除了包含词根外，还包含一些**词形特征（Morphological features**）。例如：“+N”(名词)，“+V”(动词) 表示词性，“+pl’’ 表示复数，“+sg’’ 为单数，“+3sg’’ 表示第三人称单数，“+prespart’’ 表示用于进行时的 ing 形式，“+pastpart’’ 表示用于完成时或被动语态的 -ed/-en 形式。值得注意的是，对于同一个词可以有不同的词形分析结果（例如 “walk’’）。这些不同的词形分析结

果与词所在的上下文环境有关。这里主要讨论与上下文无关的分析算法。

基于有限状态转换机的词形分析

有限状态转换机（Finite State Transducer，FST）是有限状态机（Finite State Automata，FSA）的扩展。对一个输入串，有限状态机在每个输入字符后进行状态转移。有限状态转换机则是在状态转移的同时给出一个输出。换句话说，有限状态机能够识别一个输入串，有限状态转换机则能够将输入串转换到一个输出串。一个有限状态转换机的可用如下 7 个参数定义:

有限状态转换机又称有限状态转录机、有限状态转换器

- Σ: 输入字符集（有限的字符集合）
- Γ: 输出字符集（有限的字符集合）
- Q: 状态 $q_0, q_1, ...q_N$ 的有限集合
- $q_0 \in Q$: 初始状态
- $F \subseteq Q$: 最终状态集合
- $\delta(q, w)$: *状态转移函数*。对于给定的状态 $q \in Q$, 输入串 $w \in \Sigma^*$, $\delta(q, w) \subseteq Q$ 表示在状态 q 下接收输入 w 后, 所有可能的下一个状态
- $\sigma(q, w)$: *输出函数*。对于给定的状态 $q \in Q$, 输入串 $w \in \Sigma^*$, $\sigma(q, w)$, $\delta(q, w) \subseteq \Gamma^*$, 表示在状态 q 下接收输入 w 后，所有可能的输出字符串

图3.1为一个简单的有限状态转换机的示例，其中 q_0 为初始状态 q_2 为最终状态。输入符号串和输出符号串中间用冒号分割，$x : y$ 表示当输入串是 x 时，输出 y，x 为 $x : x$ 的简写。

根据自动机理论有限状态转换机与正则关系同构，有限状态转换机存在并运算、差运算、补运算和交运算，以及两个附加的闭包特征。针对基于有限状态转换机词形分析器构造问题，主要需要如下三个运算:

- 交运算 $\cap$: 有限状态转换机 T_1 和 T_2 的交 $T_1 \cap T_2$ 接收 T_1 与 T_2 所接受的输入输出对的交集。

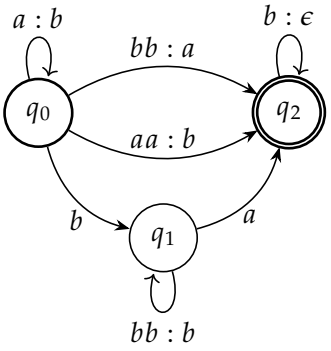


图 3.1: 有限状态转换机 FST 示例

- 取逆 T^{-1}：将 T 的输入输出互换得到一个新的有限状态转换机。
- 复合 $\circ$：假设 T_1 将输入字符串集合 I_1 映射到输出字符串集合 O_1，T_2 将输入字符串集合 O_1 映射到输出字符串集合 O_2，它们的复合 $T_1 \circ T_2$ 将 I_1 映射到 O_2。

将词形分析任务转换为一个单词的词汇层和表层的之间的对应。由于英语中很多语素边界发生拼写变化，需要引入正词法规则作为中间层。利用有限状态形态学（Finite-state morphology）范式，该有限状态转换机由三个带子（tape）组成：词汇带子（Lexical tape），中间带子（Middle tape），表层带子（Surface tape），如图3.2所示。针对 foxes，词汇层面表示词形特征（fox+N+pl），中间层面表示词语正词法（fox^#），表层层面表示词语实际拼写（foxes）。

词汇层面	f	o	x	+N	+pl		
中间层面	f	o	x	^	s	#	
表层层面	f	o	x	e	s		

图 3.2: 词汇带子、中间带子和表层带子实例

由字符集 Σ 中的字符构成，表层带子（Surface tape）由字符集 Γ 中字符组成。

以英语中名词的单数、复数屈折变换为例，简要介绍构造有限状态转换机

的过程。为了方便理解，我们首先构造*生成问题*：给定原形和词形特征，输出词汇。通过构造生成问题的有限状态转换机，可以使用取逆操作得到词形分析的有限状态转换机。

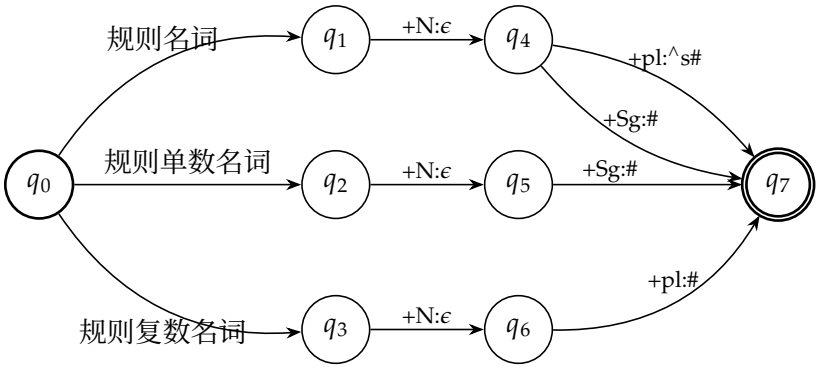


图 3.3: $\text{FST}_{\text{mid}}^{\text{feat}}$ 根据单数复数词形特征生成一种名词的中间表示形式。# 表示单词结尾，语素间用 ^ 分割。

首先，根据3.2中词汇带子和中间带子的定义，构造有限状态转换机 $\text{FST}_{\text{mid}}^{\text{feat}}$，将规则名词单数，不规则名词单数，不规则名词复数分别构造生成路径转换为中间表示，如图3.3中所示。$\text{FST}_{\text{mid}}^{\text{feat}}$ 中 ‘‘规则名词’’，‘‘不规则名词单数’’ 和 ‘‘不规则名词复数’’ 可以进一步根据词典进行展开 (如图3.4)。

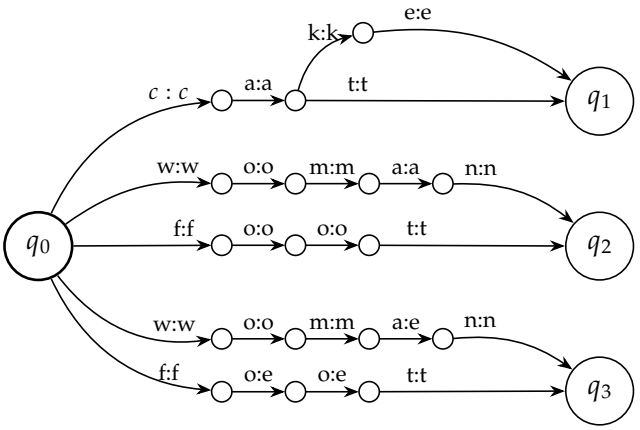


图 3.4: 图3.3中 ‘‘规则名词’’，‘‘不规则名词单数’’，‘‘不规则名词复数’’ 的展开。

通过图3.3和图3.4，可以将词形特征转化成为中间表示。例如:

规则名词单数	“cake+N+Sg” → “cake#”
规则名词复数	“cat+N+pl” → “cat^s#”
不规则名词单数	“foot+N+Sg” → “foot#”
不规则名词复数	“foot+N+pl” → “feet#”

对于规则名词 “box”(以及其他以 “z, s, x’’ 结尾的规则名词), 在生成它的复数形式时, 需要修改拼写方式: 需要插入 “-es’’ 后缀而非 “-s’’ 后缀。图3.5为一个有限状态转换机, 实现以中间表示为输入, 输入修改过的后缀, 即将 “box^s#’’ 转换成 “boxes#’’, 同时对其他词的拼写不做改动 (“cat^s#’’ 转换成 “cats#”)。类似的拼写改动还包括以 “y’’ 结尾的词需要把中间表示中的 “-s’’ 后缀修改为 “-ies’’ 后缀。每一个规则 r 对应于一个 FST_r。

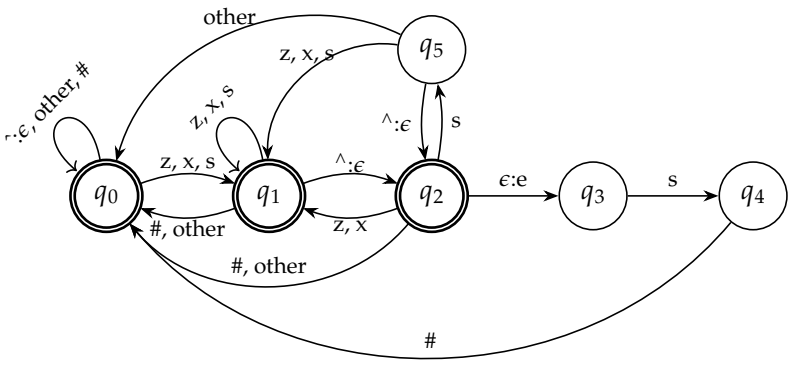


图 3.5: 有限状态转换机实现在 “z, s, x’’ 后插入 e。q_1 表示观察到 “z, s, x’’。在输入语素分隔符 “^’’ 后, 转移到状态 q_2。q_2 实现插入 “e’’, 转移到 q_3, 同时, 当 q_2 输入 “s’’, 后转移到 q_3, 如果此时输入单词解释符 #, 则判断为非法 (即不符合 ‘‘在 z, s, x 后应插入 e’’ 的拼写规则), 并拒绝接收这样的字符串。

最后, 可以通过有限状态转换机的操作将 $\text{FST}_{\text{mid}}^{\text{feat}}$ 组合 FST_r 到生成问题的有限状态转换机:

$$c\text{FST}_{\text{gen}} = \text{FST}_{\text{mid}}^{\text{feat}} \circ (\cap_r \text{FST}_r)$$

同时, 通过取逆操作可以得到词形解析的有限状态转换机:

$$c\text{FST}_{\text{parse}} = \text{FST}_{\text{gen}}^{-1}$$

3.3 词语切分

词是语言中能够独立运用的最小单位，通常也是自然语言处理算法的基础单元，以英语为代表的印欧语系（Indo-European languages）中词之间通常有分隔符（空格等）来区分，词可以较容易的从句子中分割得到。但是以汉语为代表的汉藏语系（Sino-Tibetan languages），以及以阿拉伯语为代表的闪-含语系（Semito-Hamitic languages）中却不包含明显的词之间的分隔符，而是由一串连续的字符构成。因此，针对汉语等语言的处理算法通常首先需要进行词语切分。

本节将以汉语为例介绍词语切分的基本概念以及所面临的主要问题，然后介绍基于词典、基于字统计、基于词统计以及基于神经网络的分词算法，最后介绍常见的中文分词数据集合。

中文分词概述

汉语作为汉藏语系的典型代表，其句子并不使用分割符来标识文本中的词。例如，本节标题“自然语言处理概述”一句中的八个汉字对应到四个汉语词汇（“自然”，“语言”，“处理”，“概述”）。如何将汉字序列中的词切分出来是中文分词任务的核心目标。中文分词的主要困难来自以下三个方面：分词规范、歧义切分和未登录词识别。

分词规范

汉语中对词的具体界定是一个目前还没有定论的问题。1992 年国家标准局颁布的《信息处理用现代汉语分词规范》中大部分规定都是通过举例和定性描述来体现。例如：“二字或三字词，以及结合紧密、使用稳定的二字或三字词组，一律为分词单位。”然而在实际应用中对“紧密”与“稳定”都很难界定，不可直接用于计算。

北京大学计算语言学研究所俞士汶教授为了构造包含 2600 多万字《人民日报》基本标注语料库，制订了词语切分和词性标注规范 [4] 。针对国家标准分词规范，对分词单位进行了定义和解释。针对人名、地名、机构名、其他专有名词、数词、数量词组、时间词、区别词、述补结构、成语、习用语、非汉字的字符串等情况分别进行了详细的说明。部分标注规范如下所示：

[4]: 俞士汶 et al. (2003), *北大语料库加工规范：切分·词性标注·注音*

1. 人名（nr）：汉族方式的“姓”和“名”单独切分，，“姓”标注为 nrf，“名”标注为 nrg 。例如：李/nrf 明/nrg，欧阳/nrf 洪涛/nrg；
2. 地名（ns）：国名不论长短，作为一个切分单位，地名后有“省”、“市”等单字的现代行政区划名称时，不切分开，如果地名后的行政区划有两个以上的汉字，则将地名同行政区划名称切开。例如：中华人民共和国/ns，上海市/ns，[深圳/ns 特区/n]ns；
3. 机构名（nt）：一般是短语型的，较长，且含有地名或人名等专名，按照 [4] 给出的规范需要先切分，再组合，加方括号标注为 nt。例如：[中国/ns 中文/n 信息/n 学会/n]nt，[复旦/ns 大学/n]nt；
4. 数词与数量词组：基数、序数、小数、分数、百分数一律不予切分，约数，前加副词、形容词或后加“来、多、左右”等助数词的应予切分。例如：一百二十三/m，约/d 一百/m 多/m 万/m；
5. 时间词：年月日时分秒，按年、月、日、时、分、秒切分，“牛年、虎年”等一律不予切分，标注为 t 。例如：2021 年/t 9 月/t 16 日/t，牛年/t；
6. 成语习语：四个字的成语或习用语为一个切分单位，除标注其词类标记 i 或 l 外，还要求根据其在句子中的功能进一步标注子类，超过四个字的成语或习用语，一般不予切分，不分子类。例如：胸有成竹/iv，近水楼台先得月/i；

需要注意的是，不同的分词规范之间也存在一定的不同，微软亚洲研究院黄昌宁教授 [5] 所给出的分词标注规范中有不少与《北京大学语料库加工规范》存在不同。例如，在 [5] 中姓名需要整体标出，含有外文和数字的命名实体应整体一起标注等。此外，虽然标注规范中尽可能的给出了详尽的细节，但是其中还存在一些弹性，由于中文词汇本身具有开放性和动态性，不同人之间也存在

[5]: 黄昌宁 et al. (2006), *中文文本标注规范 (5.0 版)*

认同差异，通用分词标准也是中文分词的难题。

歧义切分

对中文分词任务，汉字序列的歧义使同一个中文句子可以呈现出不同的分词结果。这些不同的分词结果也被称为**切分歧义**。例如：“南京市长江大桥”的正确词切分方式为“南京市 | 长江大桥”，但是也可能会被切分为“南京 | 市长 | 江 | 大桥”。通常汉语中常见的切分歧义可以归纳为三类：交集型切分歧义、组合型切分歧义和真歧义。

交集型切分歧义 如果汉字串 AJB 中，AJ、JB 都可以分别组成词汇，则汉字串 AJB 被称为交集型切分歧义，此时汉字串 J 称作交集串。交集型切分歧义也被称为偶发歧义，当两个有交集的词“偶然”的相邻出现时这样的歧义才会发生。

例如：乒乓球拍卖完了。该例句中存在交集型切分歧义，A，J，B 分别代表“球”，“拍”和“卖”。“球拍”和“拍卖”同时都为合法词汇，它们之间存在有一个交集串。类似的例子还包括:“今天下雨”，“很多云彩”，“北京城市规划”，“中国产品质量”等。

组合型切分歧义 如果汉字串 AB 满足 A，B，AB 同时为词，则汉字串 AB 被称为组合型切分歧义。组合性切分歧义也称为固有歧义，组合歧义的是词固有的属性，不依赖于“偶然”发生的上下文。

例如：他马上过来。该例句中“马上”为组合型切分歧义。A，B，AB 分别代表“马”，“上”和“马上”。类似的情况还包括:“才能”，“应对”，“学会”等。

真歧义 如果汉字串 ABC 满足多种切分方式下语法和语义均没有问题, 只有通过上下文环境才能给出正确的切分结果，则汉字串 ABC 被称为真歧义。

例如：白天鹅在水里游泳。对这个句子来说，两种不同的分词结果分别为“白天 | 鹅 | 在 | 水 | 里 | 游泳”以及“白天鹅 | 在 | 水 | 里 | 游泳”。这

两种切分方式在语法和语义上都是正确的，需要考虑上下文环境才能进行正确判断。

上述歧义切分的定义都是从机器识别的角度出发的。而事实上，许多歧义切分通常不会或者很少出现在真实中文文本中。例如，“平淡”根据定义属于组合型切分歧义，但实际上“平 | 淡”这样的切分方式在真实的上下文环境中非常罕见。根据 [6] 中的统计，中文文本中每 100 个词约出现 1.2 次切分歧义，其中交集型切分歧义和组合型切分歧义的比例约为 12:1。

[6]: 梁南元 (1987), “书面汉语自动分词系统—CDWS”

未登录词识别

未登录词（Out Of Vocabulary，OOV）是指在训练语料中没有出现或者词典当中没有，但是在测试数据中出现的词。根据分词算法所采用的技术不同，未登录词所代表的含义也稍有区别。基于词典的分词方法所指的未登录词就是所依赖的词典中没有的单词。对于完全基于统计方法不依赖词典特征的方法，未登录词则是指训练语料中没有出现的单词。而对于融合词典特征的统计方法，未登录词则是指训练语料和词典中均未出现的词。

未登录词又称生词（Unknown Words）

汉语具有很强的灵活性，未登录词的类型也十分复杂，可以粗略的将汉语文本中常见的未登录词可以分为以下四类：

- 新出现的普通词汇：语言的使用会随着时代的变化而演化出新的词，这个过程在互联网环境中显得更为快速。例如: 下载，给力，点赞，人艰不拆等。
- 命名实体 (Named Entity)：
  ①人名（如：杰辛达，周杰伦）；
  ②地名（例如：新江湾，张江）；
  ③组织机构名（例如：亚洲善待博士组织，中央第四巡视组）；
  ④时间和数字（例如：2021-09-16，正月初四，110 亿人民币）；
- 专业名词：出现在专业领域的新词 (例如: 图灵机，新冠病毒，埃博拉);
- 其他专有名词：新出现的产品名、电影名、书籍名等。

针对中文分词中歧义切分和未登录词造成的损失情况，黄昌宁教授和赵海教授在 Bakeoff 一 2003 的四个中文分词语料库中针对的当年最好的算法进行了测试和统计，结果标明未登录词造成的分词精度失落比歧义切分造成的精度失落至少大 10 倍左右 [7] 。宗成庆教授在新闻领域的语料也进行了类似的统计实验，结果发现未登录词造成的分词错误超过 98%，其中由命名实体引起的分词错误占到了 55% 左右 [8] 。由此可见，未登录词是中文分词的一个主要瓶颈。

[7]: 黄昌宁 et al. (2007), "中文分词十年回顾"

[8]: 宗成庆 (2013), *统计自然语言处理*

中文分词任务定义

中文分词任务可以定义为：给定一个中文句子 $x = c_1, c_2, \cdots, c_n$，其中 $c_i, 1 \leq i \leq n$ 为字 (如表 3.2所示)，输出是一个词序列 $y = h(x) = w_1, w_2, \cdots, w_m$，其中 w_j 是一个中文词（如表3.2所示）。

今	晚	的	长	安	街	流	光	溢	彩	。
c_1	c_2	c_3	c_5	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
今晚		的	长安街			流光溢彩				。
w_1		w_2	w_3			w_4				w_5

表 3.2: 中文例句及其分词结果。

中文分词方法可以分为无监督分词方法和有监督分词方法两大类。无监督分词方法通常需要依赖词典信息，而有监督分词方法则将分词转换为有监督分类问题，利用已标注中文分词结果的语料构造统计模型。在本节中我们将对上述方法分别进行介绍。

基于词典的分词方法

基于词典的无监督分词主要包含前向最大匹配，后向最大匹配以及双向最大匹配等三大类。给定一个中文句子，这些算法试图根据词典，找到针对该句子最好的分词方案。

前向最大匹配算法的基本思想是，从左向右扫描句子，对于句子中的一个位置 i, 依次考虑子串 $c[i:i+L-1], c[i:i+L-2], \ldots, c[i:i]$，其中 $c[i:j] \triangleq c_i c_{i+1} \ldots c_j$ 表示从第 i 个字到第 j 个字构成的字串（每一个这样的字串对应于一个候选的词），L 表示词典中词的最大长度。当某一个 $c[i:j]$ 能够对应字典中的一个词时，输出这个词并从 $j+1$ 开始重复以上的过程直至整个句子被遍历完成。

例如，针对句子 ‘‘今晚的长安街流光溢彩’’，前向最大分词的过程如表3.2所示（为简单起见，假设词典中的词表为 {‘‘今晚’’，‘‘晚上’’, ‘‘的’’, ‘‘长安街’’, ‘‘长安’’, ‘‘流光溢彩’’, ‘‘。’’}）

表 3.3: 表3.2中句子的前向最大匹配分词过程。eos 表示句子结尾。

时间步	句子位置	候选匹配	选择词
1	1	今晚的长，今晚的，今晚，金	今晚
2	3	的长安街，的长安，的长，的	的
3	4	长安街流，长安街，长安，长	长安街
4	7	流光溢彩，流光溢，流光, 流	流光溢彩
5	11	。eos eos eos, 。eos eos, 。eos, 。	。

后向最大匹配和正向最大匹配思想相同，区别在于对于句子从右向左扫描。双向最大匹配则是同时进行前向最大匹配和反向最大匹配，当两者的分词结果不同时，可以使用启发式的规则决定选取哪一个作为最终的输出（例如选择平均词长较大的一个）。

可以看出，基于词典的分词方法具有简单，快速，可控等优点。但对于没有在词典中出现的词没有很好的处理方案。基于统计的模型能够在一定程度上弥补它们的缺陷。

基于字的统计模型

基于统计的中文分词算法通常使用*序列标注*模型建模。具体来说，对希望输出的词序列使用一个标签序列编码。对每一个字 c_i，根据它在分词结果中的

位置，可以赋予标签 y_i。例如 *"BIES" 标签系统*:

$y_i = B$ 表示 c_i 处于一个词的开始位置

$y_i = I$ 表示 c_i 处于一个词的中间位置

$y_i = E$ 表示 c_i 处于一个词的结束位置

$y_i = S$ 表示 c_i 单独构成一个词

表3.2的所对应的序列编码如表3.4所示。经过序列编码后，输出目标 y 也变为长度为 n 的 "BIES" 标签序列 $y = w_1, w_2, \cdots, w_m = y_1, y_2, \cdots, y_n$。

今晚	的	长安街	流光溢彩	。
BE	S	BIE	BIIE	S

表 3.4: 使用 "BIES" 标签对词序列编码。

序列标注问题可以采用*条件随机场（Conditional Random Field （CRF））*等结构化机器学习方法进行解决。在条件随机场模型中，通常假设特征函数 $\varphi(x, y_i, y_{i-1})$ 都仅依赖于输入 x 和相邻的两个标签 y_i, y_{i-1}（也称为一阶马尔可夫假设）。这样的假设虽然牺牲了一定的特征表示能力，但是同时也使得序列标注模型的训练和解码能够较为高效的完成。

条件随机场模型的详细介绍详见本书第 XXX 章。

如何设计有效的 $\varphi(x, y_i, y_{i-1})$ 对于序列标注任务是至关重要的。针对中文分词问题，我们介绍一种基于模板的稀疏特征表示方法。在基于模板的特征表示中，特征函数 $\varphi(x, y_i, y_{i-1})$ 的每一维为一个 0,1 取值的函数。例如，在中文分词任务中一个典型的特征如下:

$$\varphi_k(x, y_i, y_{i-1}) = \begin{cases} 1 & \text{if } x_i = \text{c and } y_i = \text{"B" and } y_{i-1} = \text{"E"} \\ 0 & \text{otherwise} \end{cases}$$

其中，c 为一个中文字。这里针对所有可能的中文字 c(或者训练集中出现的字)都有一个对应的维度 (即，这个特征模板将展开为长度为字典长度的独热向量 (one-hot vector))。表3.5列出了中文分词任务常用的模板。

此外，在模板设计时还可以加入字符的类别（例如：阿拉伯数字、中文数字、标点符号、英文字母等）以及字典信息（例如：x_{i-1}, x_i 是否是词典中的二字

模板名	描述	例子
x_i	当前字	安
x_{i-1}	$i-1$ 位置的字	长
x_{i-2}	$i-2$ 位置的字	的
x_{i+1}	$i+1$ 位置的字	街
x_{i+2}	$i+2$ 位置的字	流
x_{i-2}, x_{i-1}	$i-2$ 开始的 bigram	长安
x_{i-1}, x_i	$i-1$ 开始的 bigram	的长
x_i, x_{i+1}	i 开始的 bigram	安街
x_{i+1}, x_{i+2}	$i+1$ 开始的 bigram	街流

表 3.5: 中文分词常见模板。“例子”一栏中包含对应模板在表3.2中句子的第 5 个位置时的取值。

词，x_i, x_{i+1} 是否是词典中某个词的开头等)。基于字的分词方法可以有效的平衡训练语料中出现的词语和未登录词，并且可以使用模板特征引入词典信息。相较于基于词典的方法，基于字的分词方法通常也可以省略未登录词的识别模块。

基于词的统计模型

基于词的中文分词任务定义为寻找一个将输入句子 $x \in X$ 转换为单词序列 $y \in Y$ 的映射，其中 X 是可能的原始输入句子集合，Y 是可能的句子切分集合，该映射用 $F(x)$ 表示，公式可表达为:

$$F(x) = \arg\max_{y \in GEN(x)} Score(y)$$

其中 $GEN(x)$ 代表对于每一个输入句子 x 可能的所有候选输出。

打分函数 $Score(y)$，针对每一个分词后的句子 y 定义一个全局特征向量 $\Phi(y) \in \mathbb{R}^d$，其中 d 代表模型中的特征数量。函数 $Score(y)$ 由矢量 $\Phi(y)$ 和一组参数 $\overline{\alpha} \in \mathbb{R}^d$ 间的点积构成，a_i 代表第 i 个特征的参数:

$$Score(y) = \Phi(y) \cdot \overline{\alpha}$$

对于参数 $\overline{\alpha}$，可以使用感知机算法进行训练。对每一句句子进行解码得到一组

候选分词结果的集合，对于集合中的每一句经过分词的句子，将之与正确答案进行比对，如果结果错误则更新参数 $\overline{\alpha}$。

Algorithm 1: 基于感知机算法的 Score 函数训练算法

```
输入: 训练数据 (x_i, y_i)
输出: ᾱ
初始化参数 ᾱ = 0
for t = 1 to T do
    for i = 1 to N do
        z_i = arg max_{y∈GEN(x_i)} Φ(y) · ᾱ
        if z_i ≠ y_i then
            ᾱ = ᾱ + Φ(y_i) − Φ(z_i)
        end if
    end for
end for
return ᾱ
```

感知机算法所需的输入特征由一系列人工选取的特征值组成，包含字、词以及长度信息。在训练时会使用特征模板将解码得到的序列映射到特征向量，特征向量将被输入到评分函数中。Y. Zhang 和 S. Clark 在其论文中所使用的具体特征模板如下表所示 [9]：

表 3.6: 输入特征模板

1	单词 w
2	二元单词 w_1w_2
3	单字符单词 w
4	初始字符 c 以及长度 l
5	终止字符 c 以及长度 l
6	由空格隔开的字符 c_1 和 c_2
7	二元字符 c_1c_2
8	所有单词的第一个与最后一个字符 $c1$ 和 c_2
9	字符 c 的前一个词 w
10	单词 w 之后的第一个字 c
11	两个连续单词的第一个字符 c_1 和 c_2
12	两个连续单词的最后一个字符 c_1 和 c_2
13	单词长度 l 以及之前的词 w
14	单词的长度 l 以及之后的单词 w

在进行解码的过程中，每一个句子都有指数级数量的候选分词结果，如果将所有可能的结果都枚举一遍的话，搜索空间将变得非常巨大，使得我们无法有效地进行训练与推断。针对于这一问题，常见的解决方式是使用 Beam Search 算法进行解码。Beam search 是一种常用的限制搜索空间的启发式算法，在每一步解码过程中，从上一步解码的所有候选结集中选取前 K 个得分最高的结果继续解码，而舍弃得分排在第 K 名之后的所有候选结果。Beam search 可以理解为一种 ''松驰'' 过的贪心算法，它并不能保证得到一定会得到得分最高的候选解码序列，但往往可以得到想要的答案。算法2给出了应用于中文分词的 Beam Search 算法详细流程。

Algorithm 2: Beam Search 解码算法

输入: 原始句子 s
输出: src
初始化 $src = [[]], tgt = []$
for $index = 0$ to s.length $- 1$ **do**
    var $char = s[index]$
    for $item$ in src **do**
        var $item_1 = item$
        $item_1$.append($char$.toWord())
        tgt.insert($item_1$)
        if $item.length > 1$ **then**
            var $item2 = item$
            $item2$[$item2$.length - 1].append($char$)
            tgt.insert($item_2$)
        end if
    end for
    $src = tgt$
    $tgt = []$
end for

用一个例子来理解分词中的 Beam Search 算法：假设有这样一句话“今晚的长安街流光溢彩。”，Beam 大小为 2，在解码到第 5 个字之前的候选集中将会有两个候选分词结果:

“今晚/的/长安街”
“今晚/的/长安/街”

对于第六字“流”可以扩展出 4 个新的候选分词句：

“今晚/的/长安街/流”
“今晚/的/长安街流”
“今晚/的/长安/街流”
“今晚/的/长安/街/流”

经过打分排序后的结果为：

“今晚/的/长安街/流”
“今晚/的/长安/街流”
“今晚/的/长安街流”
“今晚/的/长安/街/流”

由于 Beam 大小设置为 2，因此只取头两句句子继续解码，舍弃之后的句子。余下步骤依此类推，从而得到最终的结果。可以注意到，在每一步只做了 4 次解码操作，从而极大地降低了计算开销。

基于双向长短期记忆网络结合条件随机场的分词

随着深度学习技术的发展，很多中文分词算法也采用了基于神经网络模型。循环神经网络（Recurrent Neural Network, RNN）相较于前馈神经网络等要求固定输入长度的神经网络结构，更适用于处理长度不固定的序列数据。特别符合文本、语音等在内的数据特性，广泛应用于自然语言处理任务的很多任务中。长短期记忆网络（LSTM）[10, 11] 是循环神经网络的一个变体，可以在一定程度上缓解简单循环神经网络的梯度消失和梯度爆炸问题。

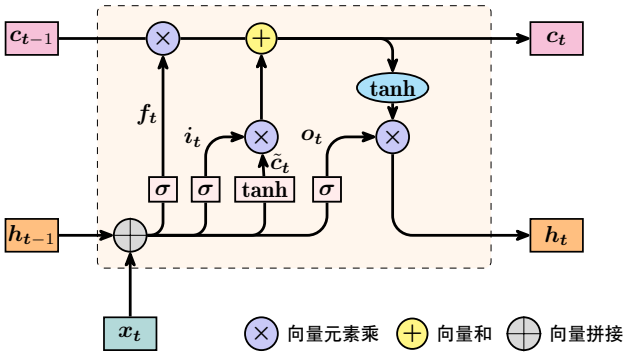


图 3.6: LSTM 网络的循环单元结构

LSTM 网络循环单元结构如图3.6所示。LSTM 网络引入了新的内部状态（internal state）$\mathbf{c_t} \in \mathbb{R}^D$，专门用来进行信息传递。此外，LSTM 网络还引入了门控机制（Gating Mechanism）来控制信息传递路径。通过**遗忘门** $\mathbf{f_t}$ 控制上一个时刻的内部状态 $\mathbf{c_{t-1}}$ 需要遗忘多少信息。**输入门** $\mathbf{i_t}$ 用来控制当前时刻的候选状态 $\mathbf{c_t}$ 有多少信息需要保存。**输出门** $\mathbf{o_t}$ 控制当前时刻内部状态 $\mathbf{c_t}$ 有多少信息需要输出给外部状态 $\mathbf{h_t}$。三个门的计算方式为：

$$\mathbf{i_t} = \sigma(\mathbf{W_i x_i} + \mathbf{U_i h_{t-1}} + \mathbf{b_i}) \tag{3.1}$$

$$\mathbf{f_t} = \sigma(\mathbf{W_f x_i} + \mathbf{U_f h_{t-1}} + \mathbf{b_f}) \tag{3.2}$$

$$\mathbf{o_t} = \sigma(\mathbf{W_o x_i} + \mathbf{U_o h_{t-1}} + \mathbf{b_o}) \tag{3.3}$$

其中 $\sigma(\cdot)$ 为 Logistic 函数。候选状态 $\tilde{\mathbf{c}}_t$、内部状态 $\mathbf{c_t}$ 以及隐藏输出 $\mathbf{h_t}$ 通过如下公式计算：

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W_c x_t} + \mathbf{U_c h_{t-1}} + \mathbf{b_c}) \tag{3.4}$$

$$\mathbf{c_t} = \mathbf{f_t} \otimes \mathbf{c_{t-1}} + \mathbf{i_t} \otimes \tilde{\mathbf{c}}_t \tag{3.5}$$

$$\mathbf{h_t} = \mathbf{o_t} \otimes \tanh(\mathbf{c_t}) \tag{3.6}$$

更为详细的介绍请参与邱锡鹏教授《神经网络与深度学习》的第六章 [12]。

在自然语言处理的很多任务中，一个时刻的输出不但与过去某个时刻的信息相关，也与后续时刻的信息相关。**双向长短期记忆网络**（Bidirectional LSTM，BiLSTM）是用来建模上述问题的一种方法。BiLSTM 是由两层长短期记忆网络组成，它们结构相同但是信息传递的方向不同。双向长短期记忆网络还可以结合条件随机场，更有效的利用结构化学习和神经网络的特点，在很多自然语言处理任务上都取得了很好的效果。图3.7给出了一个使用双向长短期记忆网络结合条件随机场（BiLSTM+CRF）进行分词的框架。

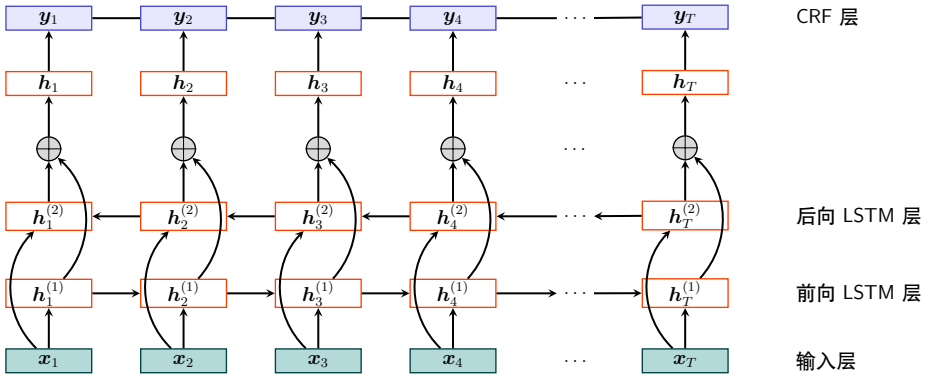


图 3.7: 基于 BiLSTM+CRF 的神经网络分词模型

在基于神经网络的分词算法中，通常采用与基于字的统计方法类似的问题建模方法，将分词任务转换为字的序列标注任务，对于给定一个中文句子 $\boldsymbol{x} = \{c_1, c_2, \cdots, c_T\}$，根据它在分词结果中的位置以及所采用标签系统（例如："BIES" 等），输出标签序列 $\boldsymbol{y} = \{y_1, y_2, \cdots, y_T\}$。具体模型如图3.7所示，BiLSTM-CRF 主要包含三层：输入层、双向长短期记忆网络层和 CRF 层。在输入层，需要将每个字转换为低维稠密的**字向量**（Character Embedding）$\boldsymbol{x}_i$。

BiLSTM 层采用双向 LSTM，其主要作用是提取句子特征。将句子中的每个字向量序列 $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ 输入到双向 LSTM 各个时间步，再将正向 LSTM 输出的隐状态序列 $(\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, ..., \mathbf{h}_T^{(1)})$ 与反向 LSTM 隐状态序列的 $(\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(2)}, ..., \mathbf{h}_T^{(2)})$。在此基础上按位置进行拼接 $\mathbf{h}_i = \mathbf{h}_i^{(1)} \oplus \mathbf{h}_i^{(2)}$，从而得到完整的隐状态序列。

对于给定的长度为 T 的输入 $\left[x\right]_1^T$，定义网络的输出矩阵为 $f_\theta\left(\left[x\right]_1^T\right)$（简写为 f_θ），其中 $\left[f_\theta\right]_{i,t}$ 表示参数为 θ 的网络对于句子 $\left[x\right]_1^T$ 的第 t 个单词的第 i 标签的打分。同时定义转移值矩阵 A，其中 $\left[A\right]_{i,j}$ 为相邻的两个单词的标签从 i 标签到第 j 标签的值，$\left[A\right]_{i,0}$ 为开始标签为第 i 标签的值。由于转移值矩阵也是模型参数的一部分，因此整个模型的参数 $\tilde{\theta} = \theta \cup \{[A]_{i,j} \quad \forall i,j\}$。对于输入句 $\left[x\right]_1^T$ 的某个特定标签序列 $\left[i\right]_1^T$ 定义为转移值和网络值的和，具体公式如下：

$$s\left(\left[x\right]_1^T, \left[i\right]_1^T, \tilde{\theta}\right) = \sum_{t=1}^{T}\left(\left[A\right]_{[i]_{t-1},[i]_t} + \left[f_\theta\right]_{[i]_t,t}\right) \tag{3.7}$$

通过 softmax 函数可以将某个标签序列的得分根据所有可能标签序列 $\left[j\right]_1^T$ 的得分进行归一化，得到标签序列的条件概率：

$$P\left([i]_i^T \mid [x]_1^T, \tilde{\theta}\right) = \frac{e^{s\left([x]_1^T,[i]_1^T,\tilde{\theta}\right)}}{\sum_{\forall [j]_1^T} e^{s\left([x]_1^T,[j]_1^T,\tilde{\theta}\right)}} \tag{3.8}$$

由此可以进一步得到对于输入 $\left[x\right]_1^T$ 的正确标签序列 $\left[y\right]_1^T$ 的条件概率的对数似然（log-likelihood）：

$$\log P\left([y]_i^T \mid [x]_1^T, \tilde{\theta}\right) = s\left(\left[x\right]_1^T, \left[y\right]_1^T, \tilde{\theta}\right) - \log\left(\sum_{\forall [j]_1^T} e^{s\left([x]_1^T,[j]_1^T,\tilde{\theta}\right)}\right) \tag{3.9}$$

基于最大化对数似然目标，以及公式3.9的线性计算方法 [13, 14]，可以根据标注语料训练得到模型参数 $\tilde{\theta}$。根据模型参数，使用维特比（Viterbi）算法可以对任意句子预测每个字的标签序列，从而得到分词结果。

中文分词语料库

如前文所述，现代分词系统的训练通常常需要依赖大规模标注语料。本节将介绍目前较为广泛使用的部分中文分词语料库。

北京大学分词语料库（PKU）

该数据集是由北京大学计算语言学研究所与富士通公（Fujitsu）合作在 110 万字《人民日报》原始数据基础上，进行了分词的信息，字符总数量约为 182 万。

示例：在 / 1998 年 / 来临 / 之际，我 / 十分 / 高兴 / 地 / 通过 / 中央 / 人民 / 广播 / 电台 / 、中国 / 国际 / 广播 / 电台 / 和 / 中央 / 电视台，向 / 全国 / 各族 / 人民，向 / 香港 / 特别 / 行政区 / 同胞、澳门 / 和 / 台湾 / 同胞、海外 / 侨胞，向 / 世界 / 各国 / 的 / 朋友 / 们，致以 / 诚挚 / 的 / 问候 / 和 / 良好 / 的 / 祝愿！

同时他们还行制定了《现代汉语语料库加工规范》，在该规范中，规定了分词要与词性标注进行结合的原则。例如，“复合”方式可将两个构词成分结合成一个新词。构词成分通常认为是语素。由于复合词的构成方式和短语的构成方式是一样的，包括定中、状中、述宾、述补、主谓、联合、连动等。当语素是成词语素时，复合词与短语的界限是不清晰的。只有当构词成分中至少有一个是不成词语素时，才有把握判断新组合的结构是一个未登录词，否则存在一定的弹性。形式上，两个字的或三个字的组合可以较宽地认为是一个词。规范中规定了许多新词的构词方式，也规定了一般性名词和专有名词切分的规范

下载地址：`http://sighan.cs.uchicago.edu/bakeoff2005/`

香港城市大学分词语料库（CITYU）

该数据集是香港城市大学语言资讯科学研究中心制作的繁体中文分词数据集，对包含 145 万字的原始数据进行了切分。

示例：一 / 宗 / 平常 / 的 / 超速 / 上訴 / ，揭露 / 了 / 青嶼 / 幹線 / 一 / 隧 / 三 / 橋 / 的 / 80 / 公里 / 車速 / 上限 / 原來 / 並 / 沒有 / 刊憲 / ，/ 立即 / 有 / 司機 / 組織 / 表示 / 考慮 / 提出 / 集體 / 訴訟 / ，希望 / 取回 / 過往 / 因 / 超速 / 失去 / 的 / 分數 / 及 / 罰款 / ；另一邊廂 / ，警方 / 表示 / 會 / 考慮 / 上訴 / ，並 / 堅稱 / 運輸署長 / 有權 / 在 / 毋須 / 刊憲 / 的 / 情況 / 下 / ，在 / 青馬 / 管制區 / 實施 / 「/ 暫時 / 的 / 速度 / 限制 / 」/ 。

他们制定了相关的切词规则，在名词，数词，时间词，略语，二字结构，三字复合词，四字词，短语，叠词，非汉字部分这十个方面的切分进行了详细的规范。另外还对其他方面进行了补充，古语方言和熟语等不进行切分，例如踏破铁鞋无觅处这句话不进行分词。

下载地址：`http://sighan.cs.uchicago.edu/bakeoff2005/`

微软研究院分词语料库（MSR）

该语料库是由微软亚洲研究院（MSRA）整理，在 230 万字的简体中文原始语料上进行划分，采用 CP936 的编码方式。

示例: 产油国 / 、/ 国际 / 石油 / 公司 / 和 / 石油 / 消费 / 国 / 应该 / 相互 / 协商，在 / 长期 / 互利 / 基础 / 上 / 建立 / 新 / 的 / 油 / 价 / 体系。

数据集将词汇分为三大类，词汇词（如教授，高兴，吃饭），命名实体（如蒙特利尔，中央民族乐团）和陈述词。其中陈述词类别较多，有日期，时间，持续时间，量词电话号码等。

下载地址：`http://sighan.cs.uchicago.edu/bakeoff2005/`

语料库名称	数据集规模	语言	标注内容
PKU	110 万	简体中文	分词、词性、专有名词
CITYU	145 万	繁体中文	分词
MSR	230 万	简体中文	分词

表 3.7: 中文分词语料库汇总

3.4 词性标注

词性是词语的基本属性，根据其在句子中所扮演的语法角色以及与周围词的关系进行分类。**词性标注**是指在给定的语境中确定句子中各词的词性 [15] 。词性标注是句法分析的基础，也是自然语言处理中一项重要的基础任务。

[15]: 吴立德 (1997), *大规模中文文本处理*

词性标注的主要难点在于歧义性，即一个词可能在不同的上下文中具有不同的词性。例如：“book” 可以表示名词 “书”，也可以表示动词 “预定”，“good” 可以表示形容词 “好”，也可以表示名词 “货物”，“China” 可以表示专有名词 “中国”，也可以表示普通名词 “瓷器” 等等。因此需要结合上下文来确定词在句子中所对应的词性。另一方面，具有兼类词多为常用词，而且越是常用词，其用法就越多。英语 “like” 就具有动词、名词、介词等多种词性。针对北京大学计算语言学研究 200 万字语料库统计，发现兼类词所占比例仅有 11%，但是出现的次数缺占到了 47%[16]。对 Brown 语料库的统计也发现超过 80% 的词通常只有一个词性。

具有多个词性的词语称为兼类词

此外，由于在语言学研究中，还没有一个被广泛认可的统一词性划分标准，在不同的语料集中所采用的划分粒度和标记符号也都不尽相同，这也在一定程度上对词性标注问题研究造成了困难。表3.8列出了在宾州树库 (PTB) 中所使用的词性。而宾州大学汉语树库（Chinese Penn Treebank）中汉语词性被划分为 33 类，北京大学计算语言学研究所给出的语料库加工规范中包含 26 个基础词性，74 个扩展词性。由于词性表以及词性定义有许多不同的变种，词性标注的结果与这些标注密切相关。本节中将主要以 PTB 标准为例。

标签	描述	标签	描述
CC	并列连词	CD	数字
DT	限定词	EX	*there*
FW	外来词	IN	介词或从属连词
JJ	形容词	JJR	形容词比较级
JJS	形容词最高级	LS	列表项标记
MD	情态助动词	NN	名词单数
NNS	名词复数	NNP	专有名次单数
NNPS	专有名词复数	PDT	前限定词
POS	所有格结束词	PRP	人称代名词
PRP$	物主代词	RB	副词
RBR	副词比较级	RBS	副词最高级
RP	小品词	SYM	符号
TO	to	UH	叹词
VB	动词	VBD	动词过去式
VBG	动词现在进行式	VBN	动词过去分词
VBP	动词一般现在式 非第三人称单数	VBZ	动词一般现在式 第三人称单数
WDT	Wh-限定词	WP	Wh-代词
WP$	所有格 Wh-代词	WRB	Wh-副词

表 3.8: 宾州树库中的词性标签

基于规则的词性标注

基于规则的词性标注算法是最早应用于词性标注任务的一类方法，其核心思想是利用词典和搭配规则针对词语和上下文进行分析，从而得到句子中每个词语的词性的方法。早期通常采用人工的方法来构建规则，随着机器学习算法的不断发展以及资源的不断完善，也出现了一些基于机器学习方法的规则自动学习算法。在本节中我们将重点介绍基于转换的 Brill Tagger 方法 [17]。

Brill Tagger 是一种利用错误驱动方法学习转换规则的词性标注算法。在 Brown 语料库上仅使用 71 个规则就得到接近 95% 的分析准确率。其分析算法的主要过程如下：

1. **初始化**：对于词典中包含的词语，根据词语最常使用的词性设置初始值；对于词典中没有的单词根据词性分析结果设置初始值（例如：以

大写字母开头的设置为专有名词)。

2. **规则转换**：根据补丁规则对初始标注进行转换，补丁规则包含以下三类：

   a) 如果某单词词性为 a，并且其所在上下文为 C，那么将其词性转换为 b；

   b) 如果某单词词性为 a，并且其具有词汇属性 P，那么将其词性转换为 b；

   c) 如果某单词词性为 a，并且其周边范围 R 内有一个词汇具有属性 P，那么将其词性转换为 b；

例如：补丁规则 **"NN VB PREV-TAG"** 表示，如果一个单词被标注为了 NN（名词），并且它前前面的单词标注为了 TO（不定式 "to"），那么将这个单词的词性转换为 VB（动词）。可以用用于解决类似 "to **book** a hotel" 中对于单词 book 的词性默认标注错误的问题。

Brill Tagger 中对于补丁规则的学习方法采用了基于错误驱动的有监督模板学习方法。首先根据现有的初始词典和补丁模板针对从训练语料中预留的用于获取模板的语料进行分析，将错误的分析结果汇总为三元组 $<tag_a, tag_b, num>$ 形式，表示一个单词的词性应该为 tag_b，但是在评测语料中有 num 次都被标注了为了词性 tag_a。根据所得到的三元组，利用以下模板生成补丁规则：

- 前一个（或者后一个）单词被标注为了 z
- 前面第二个（或者后面第二个）单词被标注为了 z
- 前面两个（或者后面两个）单词某一个被标注为了 z
- 前面三个（或者后面三个）单词某一个被标注为了 z
- 前一个单词被标注为了 z，并且后一个单词被标注为了 w
- 前一个单词被标注为了 z，并且前面第二个（或者后面第二个）单词被标注为了 w
- 当前单词是（不是）首字母大写
- 前一个单词是（不是）首字母大写

根据每个 $<tag_a, tag_b, num>$ 三元组，以及利用上述模板得到的补丁规则，可以计算利用该规则可以修复的错误标记数，以及利用该规则所引入的新的错误数。根据上述数值，选择改进最大的补丁规则加入规则列表中，并进行新一轮的分析和规则生成。

基于错误驱动的规则学法方法可以在一定程度上缓解人工规则抽取上的时间成本和人力成本。在词性标注问题中取得了不错的效果。但是其效果严重依赖于训练语料的规模和质量，同时也较难处理未登录词。此外，受到规则模板复杂度的限制，其效果通常也低于基于统计机器学习的方法。

基于隐马尔科夫模型的词性标注

隐马尔科夫模型又称隐马尔可夫模型

隐马尔科夫模型（Hidden Markov Model，HMM）是马尔科夫过程扩充而来的一种随机过程，其基本理论是由数学家 Baum 及其同事构建并逐步完善。随着隐马尔科夫模型在语音识别领域 [18] 取得巨大成功，其在自然语言处理众多序列标注任务中也得到了广泛应用并取得了非常好的效果。一个隐马尔科夫模型可用如下 5 个参数定义:

- N: 状态数。所有的状态记为 $S = \{s_1, s_2, ..., s_N\}$。系统在 t 时刻的状态记为 q_t。$Q = \{q_1, q_2, ..., q_T\}$，为长度为 T 的状态序列。
- M: 观察值数。所有的可能观察值记为 $V = \{v_1, v_2, ...v_M\}$。系统在 t 时刻的观测值记为 o_t。$O = \{o_1, o_2, ..., o_T\}$，为长度为 T 的观测序列。
- π：初始状态概率。$\pi = [\pi_i]_{1\times N}, \pi_i = P(q_1 = s_i), 1 \le i \le N$，表示初始时刻 $t = 1$ 时处于某个状态 s_i 的概率。
- A: 状态转移概率矩阵。$A = [a_{ij}]_{N\times N}, a_{ij} = P(q_{t+1} = s_j | q_t = s_i), 1 \le i, j \le N$，表示在时刻 t 处于状态 S_i 的条件下，下一时刻 $t + 1$ 转移到状态 s_j 的概率。
- B: 观测概率矩阵。$B = [b_j(k)]_{N\times M}$，$b_j(k) = P(o_t = v_k | q_t = s_j), 1 \le j \le N, 1 \le k \le M$，表示在时刻 t 处于状态 s_j 的条件下，观测到 v_k 的概率。

为了简化起见，隐马尔科夫模型可以表示成 $\lambda = (A, B, \pi)$。M，N 也隐含的已经包含在 A, B, π 中。隐马尔科夫模型的三个主要问题是：

问题 1：观测概率计算 在给定模型 $\lambda = (A, B, \pi)$ 的情况下，如何根据观测序列 $O = \{o_1, o_2, ..., o_T\}$ 计算 $P(O|\lambda)$，即在给定模型情况下，如何观测序列的概率。

问题 2：状态序列预测 在给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = \{o_1, o_2, ..., o_T\}$ 计算 $P(O|\lambda)$ 的情况下，如何得到与该观测序列最匹配的状态序列 $Q = \{q_1, q_2, ..., q_T\}$，即如何根据观测序列推断出隐藏的状态序列。

问题 3：模型参数学习 在给定观测序列 $O = \{o_1, o_2, ..., o_T\}$ 情况下，如何调整模型参数 $\lambda = (A, B, \pi)$ 使得该序列的 $P(O|\lambda)$ 最大，即如何训练模型使其能最好的建模观测序列。

关于问题 1，问题 2 以及问题 3 的求解方法可以参阅李航博士《统计学习方法（第二版）》第 10 章中的相关内容 [19]。

针对词性标注任务，使用隐马尔科夫模型可以按照如下方式构建和学习模型。N 为词性数，$S = \{s_1, s_2, ..., s_N\}$ 为词性表，包含所使用到的所有词性信息。M 为单词数，$V = \{v_1, v_2, ...v_M\}$ 为单词词表，包含所有单词。给定一个由 T 个单词组成的句子 $W = w_1, w_2, ..., w_T$，即相当于观测序列 $O = \{o_1, o_2, ..., o_T\}$，$o_i$ 为句子中第 i 个单词 w_i。状态序列 $Q = \{q_1, q_2, ..., q_T\}$ 则表示输入句子中单词对应的词性。根据训练语料, 可以使用最大似然估计的 Baum-Welch 方法高效的得到模型参数。在此基础上，针对输入的句子可以利用维特比（Viterbi）算法应用动态规划求解状态路径，从而得到对应的词性。图3.8给出了基于词性标注的隐马尔科夫模型概率图模型样例。

在实际应用过程中，使用隐马尔科夫模进行词性标注，通常还需要解决两个问题：长句子和未登录词。在《人民日报》语料库中，有些句子非常长，甚至会超过 120 个字。虽然这种长句子在真实环境中很少出现，但是对于模型的设计和实现都带来了一定挑战。因此，通常会限定一个句子中单词的最大数量。如果一个句子超过了所设定的最大长度，则寻找距离最大长度最近的标点，并

加西亚·马尔克斯所著的魔幻现实主义小说《族长的秋天》中，很多句子“一逗到底”，超过 1000 个字

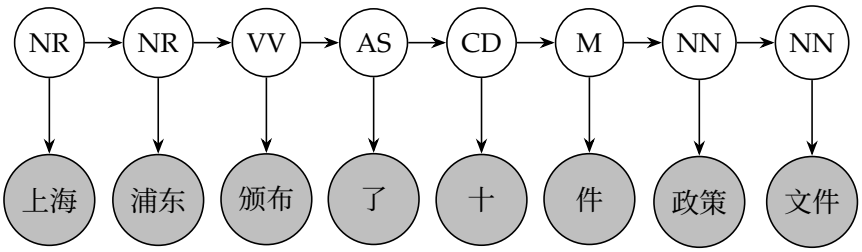


图 3.8: 词性标注隐马尔科夫模型概率图模型样例。

在标点处将句子截断。对于词典当中没有出现的未登录词，由于观测概率矩阵 B 中不存在，也需要进行特殊处理。第一种做法是在单词表中增加一个“未登录词”项，同时在观测概率矩阵中设置该词以同样的概率观察到所有标记类别。这种做法较为粗糙，在本章中我们介绍过词的一个重要分类角度是开放类词和封闭类词。未登录词通常属于名词、动词、形容词等开放类词语。其中人名、地名、机构名等名词又以占据了很大的比例。因此第二种做法是引入词法规则，对人名、地名、数词、副词等进行判断。此外，还可以根据更大规模的统计未登录词的词性，从而设定更合理的观测概率。

基于卷积神经网络的词性标注

在深度神经网络应用于自然语言处理任务之前，绝大多数自然语言处理算法依赖于特征工程。Collobert 等人 [14] 在 2011 年所提出的“从零开始的 NLP”框架利用统一的具有多个隐藏层的神经网络解决了多个自然语言处理中任务，省去了特征工程的步骤，推动了深度学习在自然语言处理任务中的快速发展。在本节我们以词性标注任务为例介绍该方法。

[14]: Collobert et al. (2011), “Natural language processing (almost) from scratch”

如图3.9所示，基于卷积神经网络的词性标注模型首先通过表查询 (Lookup Table)$LT_W(\cdot)$ 将单词通过表查询将单词转换为词性向量表示，词向量的维度是 d_{wrd}。

$$LT_W(w) = \left\langle W \right\rangle_w^1 \tag{3.10}$$

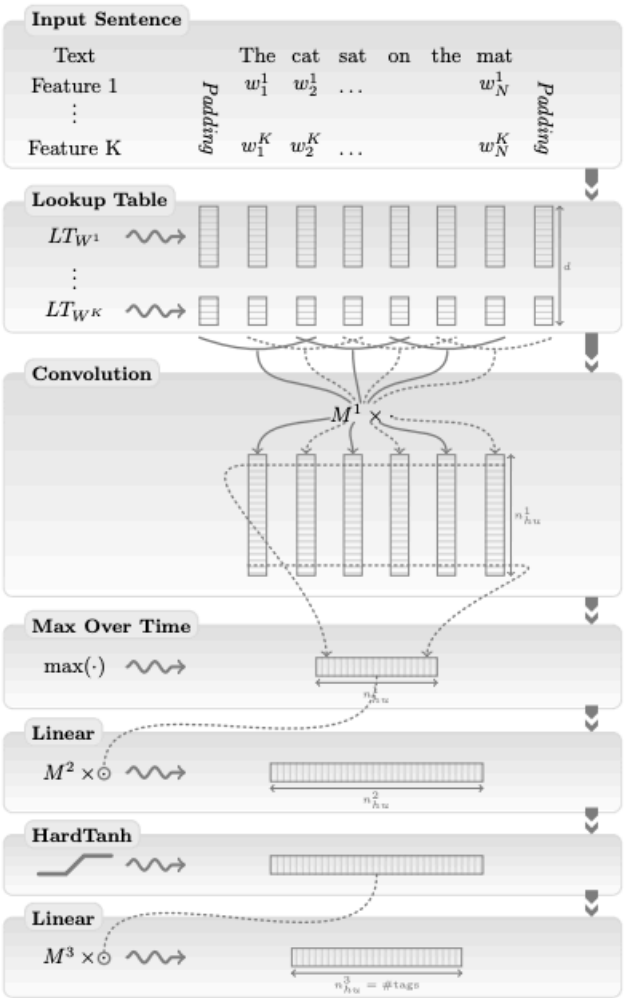


图 3.9: 基于卷积神经网的词性标注模型结构

其中 $W \in \mathbb{R}^{d_{wrd} \times |D|}$，D 是包含有限个单词的的字典，$\langle W \rangle_w^1 \in \mathbb{R}^{d_{wrd}}$ 表示 W 矩阵的第 w 列。W 矩阵也是要进行学习的参数。对于给定的任意一句包含 T 个单子的句子 $[w]_1^t$，通过表查询层对序列中的每个单词进行转换，得到如下表查询层输出矩阵：

$$LT_W([w]_1^T) = \left(\langle W \rangle_{[w]_1}^1, \langle W \rangle_{[w]_2}^1 ... \langle W \rangle_{[w]_T}^1 \right) \tag{3.11}$$

除了单词本身之外，中还可以提供一些其他特征，例如该单词在词典中最常见词性等信息。因此，可以将单词更一般的表示为 K 个离散特征 $w = \mathcal{D}^1 \times \mathcal{D}^2 \times ... \mathcal{D}^K$，$\mathcal{D}^K$ 是第 k 维特征的字典。$LT_{W^k}(\cdot)$ 是每维特征的查询表，$W^k \in \mathbb{R}^{d^k_{wrd} \times |D^k|}$ 是第 k 维特征的嵌入向量矩阵，$d^k_{wrd} \in \mathbb{N}$ 是用户给定的向量维度。对于一个单词 w，其特征向量的维度 $d_{wrd} = \sum_k d^k_{wrd}$，通过表查询得到连接后的向量：

$$LT_{W^1,...,W^K}(w) = \begin{pmatrix} LT_{W^1}(w_1) \\ \vdots \\ LT_{W^K}(w_K) \end{pmatrix} = \begin{pmatrix} \langle W^1 \rangle^1_{w_1} \\ \vdots \\ \langle W^K \rangle^1_{w_K} \end{pmatrix} \tag{3.12}$$

由此，可以得到如下表查询层输出矩阵：

$$LT_{W^1,...,W^K}([w]^T_1) = \begin{pmatrix} \langle W^1 \rangle^1_{[w_1]_1} & \dots & \langle W^1 \rangle^1_{[w_1]_T} \\ \vdots & & \vdots \\ \langle W^K \rangle^1_{[w_K]_1} & \dots & \langle W^K \rangle^1_{[w_K]_T} \end{pmatrix} \tag{3.13}$$

在表查询层后连接的是卷积层（Convolutional Layer），根据所设置的窗口大小 d_{win}，将每个单词周边的单词拼接起来构成具有 $d_{wrd} d_{win}$ 维度的向量：

$$f^1_\theta = \langle LT_W([w]^T_1) \rangle^{d_{win}}_t = \begin{pmatrix} \langle W \rangle^1_{[w]_{t-d_{win}/2}} \\ \vdots \\ \langle W \rangle^1_{[w]_t} \\ \vdots \\ \langle W \rangle^1_{[w]_{t+d_{win}/2}} \end{pmatrix} \tag{3.14}$$

f^1_θ 会被给入单层或者多层的卷积层，第 l 层的第 t 列向量可以根据如下公式计算得到：

$$\langle f^l_\theta \rangle^1_t = W^l \langle f^{l-1}_\theta \rangle^{d_{win}}_t + b^l \qquad \forall t \tag{3.15}$$

在同一层中 W^l 为相同参数。对于 f_θ^l 中每一维在公式3.15计算完成后，都要进行非线性变化，可以采用如下方式：

$$\left[f_\theta^l\right]_i = HardTanh\left(\left[f_\theta^l\right]_i\right), \tag{3.16}$$

$$HardTanh(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \le x \le 1 \\ - & \text{if } x > 1 \end{cases} \tag{3.17}$$

通过公式3.15得到特征向量更多了反映了局部特征，并且数量与句子长度相关。为了得到全局特征并且维度固定的特征向量，需要引入池化层（Pooling Layer），在这里使用的是随时间推移最大化（Max Over Time）方法。给定通过卷积层计算得到的矩阵 f_θ^{l-1}，池化层输出的向量 f_θ^l 计算如下：

$$\left[f_\theta^l\right]_i = \max_x \left[f_\theta^{l-1}\right]_{i,t} \qquad 1 \le i \le n_{hu}^{l-1} \tag{3.18}$$

针对通过池化层计算得到的向量 f_θ^l 需要继续进行线性变换，再利用公式3.16进行非线性变换，之后再叠加新的线性层后完成特征提取工作。线性变换层对于输入 f_θ^{l-1} 利用如下公式计算得到其输出 f_θ^l:

$$f_\theta^l = W^l f_\theta^{l-1} + b^l \tag{3.19}$$

在分类阶段，采用句子级别对数似然方法（Sentence-Level Log-Likelihood）。除了网络输出矩阵 $f_\theta\left(\left[x\right]_1^T\right)$（简写为 f_θ）之外，引入转移值矩阵 A，对于输入句 $\left[x\right]_1^T$ 的某个特定标签序列 $\left[i\right]_1^T$ 定义为转移值和网络值的和。基于最大化对数似然目标，可以根据标注语料训练得到模型参数 $\tilde{\theta}$。根据模型参数，使用维特比（Viterbi）算法可以获得任意句子中每个词的词性。

相关算法以及公式在第3.3节基于 BiLSTM-CRF 方法进行分词部分进行了详细介绍，也可参考文献 [14] 查看详细算法和公式推导。

词性标注语料库

从前面几节的介绍可以知道，词性标注算法的训练过程都依赖标注语料集合。对不同算法的效果进行对比也依赖于标准测试集合。本节将介绍几种常见的包含词性标签的语料库。

宾州大学树库, PTB

对于英语而言，宾州大学句法树库 (Penn TreeBank，PTB) 是最早形成一定规模的句法树库，它是一个短语结构句法树库，取自于标准新闻题材，总计五万规模的句子，为每个句子标注了词性以及短语结构句法树。WSJ-PTB 是 PTB 项目的一部分，是目前新闻语料上最常用的词性标注数据集。WSJ-PTB 原始数据来自于 1989 年的华尔街日报，按照 PTB(V2) 的标注策略进行标注，拥有一百多万个标注单词，48 种不同的词性标签。

示例：France/NNP 's/POS unemployment/NN rate/NN was/VBD steady/JJ at/IN a/DT seasonally/RB adjusted/VBN 9.5/CD %/NN in/IN September/NNP ,/, the/DT Social/NNP Affairs/NNPS Ministry/NNP said/ VBD ./.

一般来说，一个句子虽然表面上呈现词语的线性排列，其内部的成分组织是存在一定层次结构的。PTB 使用树这种数据结构来表示句子的层次结构，构建一个大型是树库，包含丰富的语言结构信息。经过处理后，除了 WSJ-PTB 之外，PTB 还发布了标注的 Brown 语料库。

下载地址：`https://catalog.ldc.upenn.edu/LDC99T42`

中文宾州树库, CTB

宾州大学汉书树库（Chinese Penn Treebank）是建立一个大型的中文句法标注语料库。该数据集基于短语结构，进行了短语结构、短语功能、空元素等

的标注。发展至今共 8.0 版，第一版的语料主要来自于新华社的文章，在第二版中加入了香港和台湾的语料，以保证语料的多样性。2005 年 1 月发布的 5.0 版本包含 507222 个词，824983 个汉字，以及 18782 个句子，是目前最常用的 POS 任务数据集。

示例：上海_NR 浦东_NR 近年_NT 来_LC 颁布_VV 实行_VV 了_AS 涉及_VV 经济_NN 、_PU 贸易建设_NN 、_PU 规划_NN 、_PU 科技_NN 、_PU 文教_NN 等_ETC 领域_NN 的_DEC 七十一_CD 件_M 法规性_NN 文件_NN 。_PU 确保_VV 了_AS 浦东_NR 开发_NN 的_DEG 有序_JJ 进行_NN 。_PU

在 CTB 中，汉语词性被划分为 33 类，包括 4 类动词和谓语形容词，3 类名词，1 类处所词，一类代词，3 类限定词和数次，一类量词，1 类副词，1 类介词，8 类语气词和 8 类其他词。

下载地址：https://catalog.ldc.upenn.edu/LDC2005T01

Universal Dependencies (UD)

UD 是一个为多种语言开发的跨语言一致的树库项目，标注了语言的词性信息，形态特征和依存关系，其目标是促进多语言解析器的开发、跨语言学习和从语言类型学的角度进行解析研究。UD 是一个开放协作的项目，目前共有超过 200 个贡献者提供了 70 多种语言上的 100 多个树库。

示例：sentence: The oboist Heinz Holliger has taken a hard line about the problems . original: DT NN NNP NNP VBZ VBN DT JJ NN IN DT NNS . universal: DET NOUN NOUN NOUN VERB VERB DET ADJ NOUN ADP DET NOUN .

对各种树库下的标记集的高级分析表明，大多数标记集都是非常细粒度的，并且是特定于语言的。UD 使用 Petrov 等人在 2011 年提出的一个跨语言统一的词性标注系统 [20] 。他们提出了一个由十二个通用词类构成的标记集，包括 NOUN (名词), VERB (动词), ADJ (形容词), ADV (副词), PRON (专有名词), DET

[20]: Petrov et al. (2011), “A universal part-of-speech tagset”

(限定词和冠词), ADP (介词和后置词), NUM (数字), CONJ (连接词), PRT (小品词), ‘.’(名词所有格) 和 X (其他)。这 12 个类涵盖了大多数语言中最常见的词性。除了标记集之外，他们还为来自 22 个语言的 25 个不同的树库开发了一个从细粒度词性标记到这个通用标记集的映射。

下载地址：https://universaldependencies.org/

表 3.9: 词性标注语料库汇总

语料库名称	数据集规模	语言	标注内容
WSJ-PTB	117 万	英文	分词、词性、句法树
CTB	50 万	中文	分词、词性、句法树
UD	70 种语言	多语言	分词、词性、句法树

3.5 延伸阅读

关于中文分词，我们介绍了基于循环神经网络的方法，循环神经网络能很好地利用字符级别特征建模上下文信息实现分词任务。实际上，神经网络着这非常灵活的结构化建模能力。想要进一步提升分词的性能，通过设计网络结构有效地引入词语级别的特征非常重要。其中基于转移的模型用于分词能够有效地结合词语特征 [21]，并将传统的特征模版和神经网络自动提取的特征结合起来，在神经网络自动提取的特征和传统的离散特征的融合方法做了尝试。结果表明，通过组合这两种特征，分词精度可以得到进一步提升。另一种引入词语特征的方法是栅格化循环神经网络 [22]，这种方法能将句子里的字与所有可能匹配的词语同时进行建模，从而提升分词准确率。

在词性标注任务中基于循环神经网络的方法已经能取得非常好的效果。如何提升词性标注的效率便成了研究者关注的问题。比如基于空洞卷积 [23]）的词性标注利用卷积神经网络并行性能，有能用空洞卷积的形式扩大感受野，在取得较好准确率的同时也能有更快的处理速度。也有研究者将循环神经网络设计为并行结构——并行隐状态循环神经网络 [24], 同时引入全局结点来弥补上下

文建模的不足，这种方法能打破了循环神经网络序列建模句子的方式，实现并行快速处理。

3.6 习题

1. 语言学中词和语素的定义分别是什么？其主要的不同是什么？
2. 如何处理词性标注算法中的未登录词？
3. 如何同时进行分词和词性标注？
4. 中文分词中歧义切分包含几种主要的类别？针对每种歧义类别请试举几例，并说明具有歧义的切分方式。
5. 在中文分词中我们也可以使用 BIO 标签来建模序列标注 (标识一个词的开始, 中间和结束). 请尝试分析使用这样的标签集合与 BIES 标签集合会有什么区别？你能通过实验验证吗？是否还可以设计其他的标签集合，它们能否帮助获得更好的中文分词器？
6. 试比较几种开源分词器在不同语料上的性能（新闻语料，淘宝评论，小说等）。
7. 是否可以用使用 BiLSTM 或者 BiLSTM-CRF 进行词性分析？与使用 BiLSTM-CRF 算法相比有什么优缺点？

4 句法分析

任何人类语言都具备构造无限数量句子的能力 [1]。可以通过增加形容词、副词、关系小句、介词短语等方法把任意的句子进一步的进行创造。因此，无法将一种语言按照词典的方式将所有句子存储起来。通过语言学研究发现，句子并非词语的随意组合，而是按照一定规则结合起来的离散单位组成 [25]。**句法（Syntax）**就是研究这些自然语言中不同成分组成句子的方式以及支配句子结构并决定句子是否成立的规则。句法反映了句子中词、词序以及层级结构之间的关系。通过句法规则，可以将词语组合成短语，将短语组合成句子，或者确定某个句子是否符合某一语言正确的词序。句法规则还阐释了词的分组与其意义的相关性，并在一定程度上解释了语言的无限性。句法分析具有非常重要的作用，是自然语言处理中的经典问题。

本章首先介绍语言学中句法基本概念，在此基础上以介绍成分句法分析算法、依存句法分析算法以及常用的句法分析语料库。

4.1 句法概述

句法是现代语言学研究中的重要课题，有大量的句法理论（syntactic theory）相关研究。句法理论在很多语法理论研究中也都是占据了主要部分。语法理论在构建时，一个重要的问题是该理论是基于成分关系还是基于依存关系。如果基于成分关系，那么该理论就属于**成分语法（Constituent Grammar）**。成分语法主要包含：范畴语法（Categorical Grammar）、词汇功能语法（Lexical Functional Grammar）、最简方案（the Minimalist Program）等。如果一个语法理论基于依存关系，那么该理论就属于**依存语法（Dependency Grammar）**。依

成分语法也称作短语结构语法（Phrase Structure Grammar）

存语法主要包含：文本-意义理论（Meaning Text Theory）、词格理论（Lexicase）、功能生成描述理论（Functional Generative Description）等。在本节中我们将分别针对成分语法和依存语法进行介绍。

成分语法理论概述

乔姆斯基（Chomsky）在其 1957 年发表的《句法结构》[26] 中奠定了成分语法的基础。**成分**（Constituent）是指一个句子内部的结构成分。成分可以独立存在，或者可以用代词替代，又或者可以在句子中不同位置移动。例如：

成分又称短语结构

他正在写一本小说。 “一本小说”是一个成分

在此基础上，根据不同成分之间是否可以进行相互替代而不会影响句子语法正确性，可以进一步的将成分进行分类，某一类短语就属于一个**句法范畴**。比如“一本小说”、“一所大学”等都属于一个句法范畴：名词短语（None Phrase，NP）。句法范畴不仅仅包含名词短语（NP）、动词短语（VP）、介词短语（PP）等短语范畴，也包含名词（N）、动词（V）、形容词（Adj）等词汇范畴。除此之外还包含功能范畴（包括：冠词、助动词等）。

句法范畴之间不是完全对等的，而是具有层级关系。例如：一个句子可以由一个名词短语和一个动词短语组成，一个名词短语可以由一个限定词和一个名词组成，一个动词短语又可以由一个动词和一个名词短语组成。我们可以定义**短语结构规则**（Phrase Structure Rules）对句法范畴间的关系进行形式化描述：

短语结构规则又称改写规则或重写规则

```
(1) S -> NP VP
(2) NP -> Det N
(3) VP -> V NP
```

短语结构规则通常可以用 $X \rightarrow YZW \cdots$ 表示，其中 X 表示短语名称，“$\rightarrow$”表示“改写为”，“$YZW \cdots$”定义了短语 X 的结构，如果 YZW 是短语，则需要构造出它们的规则。

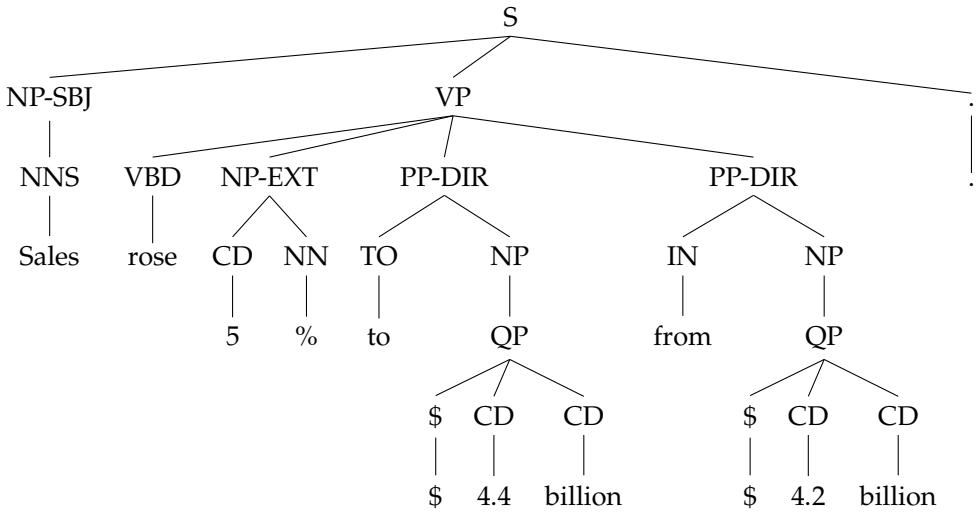


图 4.1: Penn Treebank 3.0 成分句法树样例

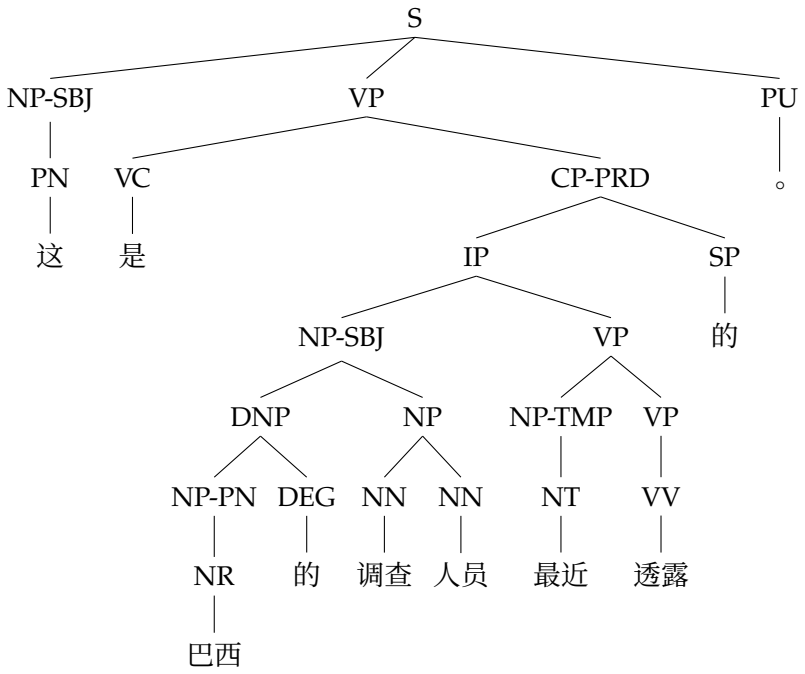


图 4.2: Chinese Treebank 7.0 成分句法树样例

成分语法就是由句法范畴以及短语结构规则定义的语法。由于短语结构规则具有递归性，可以使短语和句子无限循环组合。这也说明了语言的创造性和

无限性。如图4.1和图4.2所示，一个句子根据成分语法分析得到的层级结构，可以使用成分语法树进行表示。

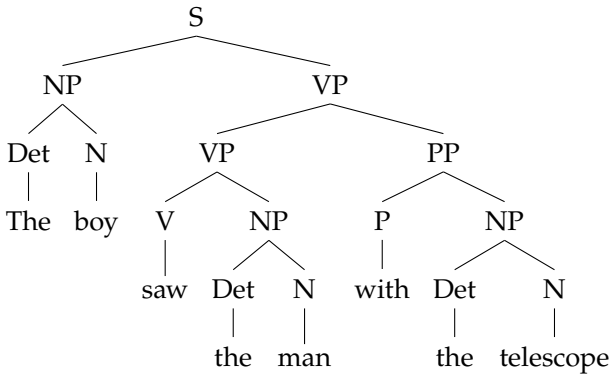


图 4.3: 句子 The boy saw the man with telescope 的第一种成分句法树

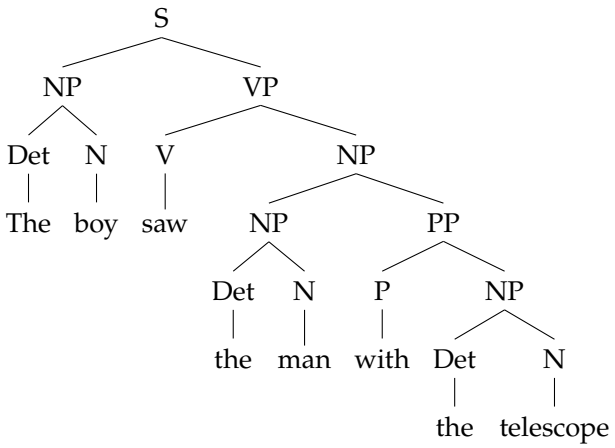


图 4.4: 句子 The boy saw the man with telescope 的第二种成分句法树

由于成分语法局限于表层结构分析，不能彻底解决句法和语义问题，因此存在非连续成分、结构歧义等问题。如图4.3和图4.4所示，对于句子 “The boy saw the man with telescope” 有两种可能的合法的句法结构树，不同的树结构对应不同的语义。第一种句法树表示 “ 男孩使用望远镜看到了这个男人 ”，第二种句法树表示 “ 男孩看到了一个拿着望远镜的男人 ”。

依存语法理论概述

吕西安·泰尼埃（Lucien Tesnière）于 1959 年发表的《句法结构分析》奠定了句法以依存关系研究的基础 [27]。在基于依存关系的语法中，句子中的每个成分对应句法结构中的唯一一个节点。两个成分之间的依存关系是二元的非对称关系，具有方向性，一个成分是中心语，另一个成分依附于中心语存在，关系从中心语成分指向依存成分。中心成分称为**中心词**（governor，regent，head），依存成分也称为**修饰词**（modifier，subordinate，dependency）。例如：“ 读书 ” 中 “ 读 ” 是中心语，“ 书 ” 依存于 “ 读 ”。有多种方式可以用于表示依存关系，如下图所示：

中心词也称为支配者

修饰词也称为从属者

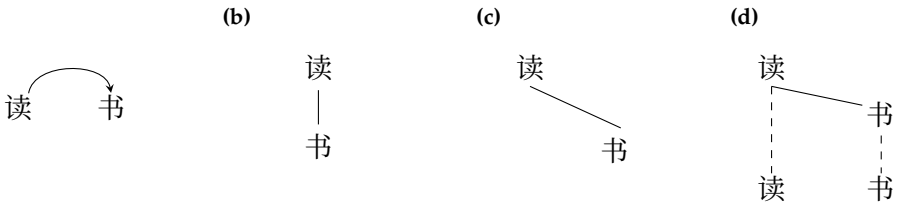


图 4.5: “ 读书 ” 的依存关系表示实例

两个单词之间是否存在依存关系？单词之间谁处于支配地为？谁处于从属地为？建立这些词与词之间关系的依据是什么？很多依存句法理论从不同方面对上述问题进行了回答。配价（Valency）理论是其中最为经典的论著之一。这里 “ 价 ” 的概念是由泰尼埃从化学中的 “ 化合价 ” 的概念引入语言学研究。价是词语的一个属性，表示某个词语与其他词结合的能力。配价模式 (Valency pattern) 则是描述了某一个具有特定意义的词的出现语境，以及当一个词出现在一个特定的模式下时，还有哪些词语会出现在这个模式下及其语义角色。通过对不同词类的支配和被支配能力（配价）以及词类间依存关系类型的定义，就可以得出某种具体的依存句法。

利用配价理论，可以将汉语里的动词 V 根据其价数，即属于几价动词，分为以下四类：

1. **零价动词**不强制与某个行动元关联的动词，例如：地震、刮风、下雨、下雪……
2. **一价动词**强制与一个行动元关联的动词，例如：病、醉、休息、咳嗽、游泳
3. **二价动词**强制与两个行动元关联的动词，例如：爱、采、参观、讨论、改良……
4. **三价动词**强制与三个行动元关联的动词，例如：给、送、告诉、退还、赔偿……

有一种特殊的关系是并列关系，构成这种关系的元素之间不存在支配与被支配关系。但是为了能够使得与其他的关系统一，也通过并列关系标记将其纳入句法树中，通常将第一个词作为中心语，其余的词作为该词的从属成分。这种情况下，在树结构上表现出来的层级关系是一种伪层级。

词语间的依存关系还可以根据语法关系定义为不同的类型，Carroll 等人[28] 将依存关系细分为了 20 种，并给出了关系之间的层级结构。Marneffe 等人[29] 在上述工作的基础上对依存关系进行了进一步的细化，定义了 48 种依存关系，主要分为论元依存关系和修饰语依存关系两大类。图4.6给了一个包含依存关系类型的句法树的样例。

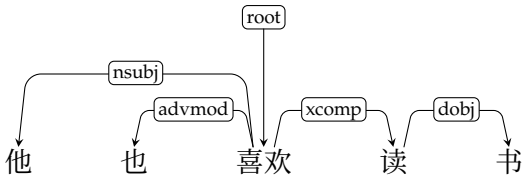


图 4.6: 句子“他也喜欢读书”的依存句法树

通常依存句法树引入 *root* 做为句子或者单棵树的主要支配者，是树的根节点。一般情况下动词是 *root* 节点的直接从属成分，其余节点应该直接或者间接依存于动词节点。没有动词的句子除外，但是应存在一个句子成分做为 *root* 的从属成分。依存句法树中的每个节点只能有一个支配者，但是可以有多个从属者。一个符合语法的句子中，所有节点应该是相连的，不允许存在游离在外的节点。如图

依存语法中根据其依存成分与中心语或姐妹成分在语序上的关系，可以分为符合投射性原则和违反投射性原则两类。在依存层级中如果每个依存成分与其中心或姐妹成分相邻，那么该依存层级就是符合投射性原则（projectivity）；如果依存成分的相邻成分使其与中心语分类那么就是违反投射性原则的。

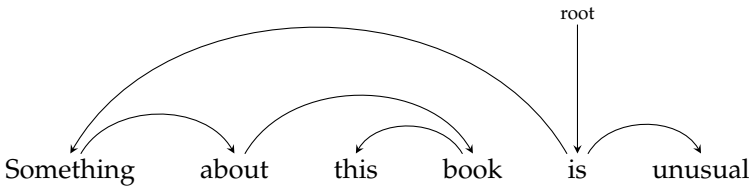


图 4.7: 符合投射原则依存句法树样例

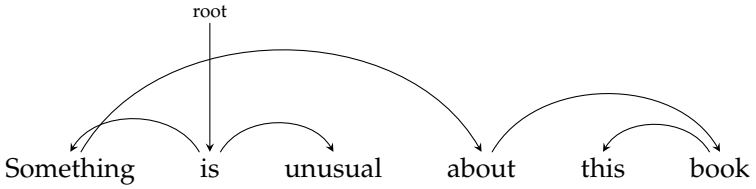


图 4.8: 违反投射原则依存句法树样例

图4.7中没有任何两线交叉，因此是投射性的。图4.8中成分 “about this book” 与其中心语不相邻，因此存在交叉线，违反了投射原则。这种现象在依存句法中通常称为远距离依赖（long-distance dependencies）。

4.2 成分句法分析

成分句法分析（Constituency Parsing）就是对给定句子根据成分语法中制定的规则构建其所对应的结构树的过程。由第4.1节中关于成分语法的基本概念和组成部分的介绍可以知道，成分语法主要包含句法范畴和短语结构规则两个部分。句法范畴又包含短语范畴、词汇范畴和功能范畴。短语结构规则通常可以由 $X \rightarrow YZW \cdots$ 形式进行表示。成分语法与数学系统中的**上下文无关文法（Context-Free Grammar，简称 CFG）**组成非常类似，因此上下文无关文法是目前最常用的模拟英语以及其他语言的成分语法的数学系统。上下文无关文法 G 包含以下四个部分：

终结符 $\sum$（**Terminal Symbols**） 与语言中词汇对应的符号，例如：上海，walk 等，用 u, v, w 等小写罗马字母表示；

非终结符 N（**Non-terminals**） ；对应词组等聚集类或概括性符号，例如：NP（名词短语），VP（动词短语），N（名词），介词（P）等，用 A,B,C 等大写字母表示；

初始符 S（**Start symbol**） 语法中指定的初始符，通常用 S 表示；

规则 R（**rules**） 对应语法中的短语结构规则，从初始符号开始可以造出合法句子的规则集合，例如：S→ NP VP，NP→ Det Adj N，Det → the 等，在上下文无关语法中，一个规则的左边是一个单独的非终结符，右边是一个或者多个终结符或非终结符组成的有序序列 $(\sum \cup N)*$，用 α, β, γ 等小写希腊字母表示 $(\sum \cup N)*$。

由于句法结构具有歧义，因此句法分析中最重要的工作之一也是如何消除歧义。成分语法中的结构歧义主要有两种：**附着歧义（attachment ambiguity）**以及**并列连接歧义（coordination ambiguity）**。图4.3和图4.3所给出的关于句子“The boy saw the man with telescope”的两种分析结果就是附着歧义的一个例子，其核心就是“with telescope”附着于动词 saw 还是名词短语 the man。并列连接歧义也是常见的结构歧义之一，例如，短语“重要政策和措施”中“重要”可以修饰“政策和措施”整体，也可以仅修饰“政策”，就可以表示为两种层级结构，“[重要 [政策和措施]]”和“[重要政策] 和 [措施]”。但是由于结构歧义通常伴随语义上的变化，因此句法结构歧义消除通常需要依赖统计知识、语义知识或语用知识才能更好的完成。

基于上下文无关文法的成分句法分析

在给定上下文无关文法 G 的情况下，对于给定的句子 $W = w_1 w_2 \cdots w_n$，输出其对应的句法结构树，通常有两大类搜索方法：自顶向下和自底向上。自顶向下搜索试图从根节点 S 出发，搜索语法中的所有规则直到叶子节点，并行构造所有的可能的树。自底向上的方法是从输入的单词开始，每次是用语法规则，

直到成功构造了以初始符 S 为根的树。针对这两大类算法，本节中将分别介绍 CYK 分析算法和移进-规约分析算法。

CYK 成分句法分析算法

CYK 是 Cocke–Younger–Kasami 的缩写，有时也称为 CKY。

CYK 算法是由 John Cocke、Daniel Younger 以及 Tadao Kasami 分别独立提出的基于动态规划思想的自底向上语法分析算法 [30–32]。CYK 算法要求所使用的语法必须符合乔姆斯基范式（Chomsky Normal From，简称 CNF），其语法规则被限制为只具有 $A \rightarrow BC$ 或 $A \rightarrow w$ 这种形式。由于任何上下文无关语法都可以转换为相应的 CNF 语法，因此这种限制并不会对表达能力造成损失。但是这种限制使得基于表的分析算法变得简单和非常易于实现。

根据 CNF 语法形式，句法树的叶子节点为单词，单词的父节点为词性符号，在词性符号层之上每一个非终结符都有两个儿子节点。因此 CYK 算法采用了二维矩阵对整个树结构进行编码。对于一个长度为 n 的句子，构造一个 $(n+1)\times(n+1)$ 的二维矩阵 T，矩阵主对角线以下全部为 0，主对角线上的元素由输入句子的终结符号 (单词) 构成，主对角线以上的元素 T_{ij} 包含由文法 G 的非终结符构成的集合，这个集合表示输入句子中横跨在位置 i 到 j 之间的单词的组成成分。输入句子中索引从 0 开始，索引位于输入句子的单词之间，也可以看成单词之间的间隔指针（例如：$_0$ 她 $_1$ 喜欢 $_2$ 跳 $_3$ 芭蕾 $_4$）。

CYK 算法按照平行于主对角线的方向，逐层向上填写矩阵中的各个元素 T_{ij}。如果存在一个正整数 k（$i+1 \le k \le j-1$），在文法规则集中具有产生式 $A \rightarrow BC$ 并且 $B \in T_{ik}, C \in T_{kj}$，那么将 A 合并到矩阵表的 T_{ij} 中。为了方便根据矩阵 T 输出句法分析结果，在矩阵 T 每个单元 T_{ij} 中不仅记录非终结符的集合，还要记录推导路径。逐层计算完成后，判断一个句子由文法 G 所产生的充要条件是：$T_{0n} = S$。具体过程如算法3所示。

以下通过一个实例来说明 CYK 算法的具体流程。给定如下符合乔姆斯基范式的文法 G 如下：

Algorithm 3: CYK 句法分析算法

输入: 语法信息 G 和句子 $w_1 w_2 \cdots w_n$;
输出: 句法树矩阵 T

```
    // 初始化
 1: for i = 1 to n do
 2:     T_ii = w_i                          ▶ 主对角线上依次放入单词 w_i
 3:     for all A|A → w_i do
 4:         T_(i-1)i = T_(i-1)i ∪ A, (A → w_i ∈ G)      ▶ 依次放入单词 w_i 的所有词性
 5:     end for
 6: end for
    // 平行于主对角线逐层计算
 7: for d = 2 to n do
 8:     for i = 0 to n − d do
 9:         j = i+d
10:         for k = i + 1 to j − 1 do
11:             if A → BC ∈ G and B ∈ T_ik and C ∈ T_kj  then
12:                 T_ij = T_ij ∪ {A, (k, B, C)}        ▶ (k,B,C) 用于保存推导路径
13:             end if
14:         end for
15:     end for
16: end for
```

非终结符号集合：$N = \{S, N, P, V, VP\}$

终结符号集合：$\sum$ = { 她，喜欢，跳，芭蕾 }

规则集合：R = {

```
(1) S -> P VP   (2) VP -> V V  (3) VP -> VP N
(4) P -> 她     (5) V -> 喜欢  (6) V -> 读    (7) N -> 芭蕾
```

}

图4.9给出了 CYK 算法分析句子“她喜欢跳芭蕾”的过程。首先初始化矩阵主对角线上的单词以及单词所对应的词性。在此基础上进行第二层 T_{02}, T_{13}, T_{24} 元素的计算，通过规则集合中 VP->V V 推导规则，可以得到在 T_{13} 添加 VP 以及相应的路径信息。针对第三层 T_{03}, T_{14} 元素，虽然针对 T_{03} 元素，可以根据推导

规则 S->P VP 添加非终结符 S，但是由于非终结符 S 是只能出现在 T_{0n} 中，因此 T_{03} 不添加 S。T_{14} 元素中根据通过规则 VP->VP N 添加 VP 信息。最后一层 T_{04} 根据 S->P VP 规则，添加 S 及其路径信息并完成分析。

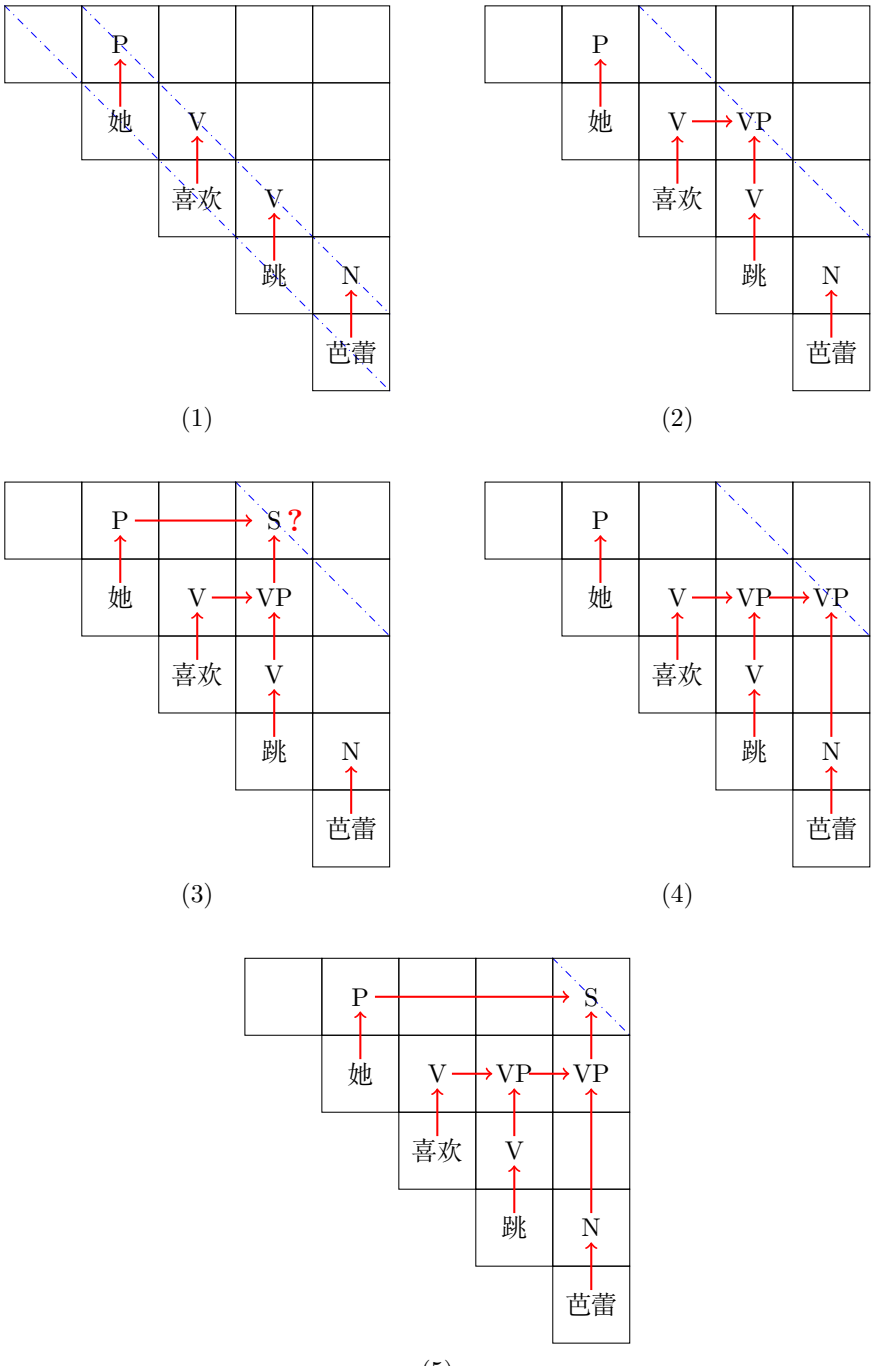


图 4.9: CYK 算法分析实例

移进-规约成分句法分析算法

移进-规约成分句法分析算法的基本思想是从左到右扫描输入的包含单词词性对的句子，使用堆栈和一系列的移进（Shift）和规约（Reduce）操作序列构建句法树 [33] 。

[33]: Ullman (1972), *The theory of parsing, translation, and compiling*

算法初始时堆栈 S 为空，队列 Q 中包含整个句子所有单词。最终通过一系列的操作，在算法结束时堆栈 S 中包含一个完整的句法树，队列 Q 为空。所采用的操作包含以下四个:

移进（Shift）： 将非空队列 Q 最左端的单词移入堆栈 S 中。

规约（Reduce）： 根据推导规则，将堆栈 S 中的最顶端的若干元素移出，然后将利用推导规则产生的新结构压入堆栈中。

根据推导规则右侧所包含非终结符数量, 在堆栈中移除相应数量元素。

接受（Accept）： 队列中所有单词都已被移到堆栈中，并且堆栈中只剩下一个由非终结符 S 为根的树，分析成功。

拒绝（Reject）： 队列中所有单词都已被移到堆栈中，但是堆栈中并非只有一个以非终结符 S 为根的树，并且无法继续规约，分析失败。

如何根据当前堆栈 S 和队列 Q 中的状态，选择下一步的操作是移进-规约分析算法中最重要的部分。由于移进和规约操作并不是完全互斥的，在很多状态下两种操作都可以选择，这就造成了移进规约冲突（Shift Reduce Conflict）。在自然语言这种非确定性语法下，这种冲突不可避免。在基于规则和策略的移进-规约分析方法中，当有多种规约可能时分析算法需要选择其中某个，当移进和规约都可以时也需要选择执行哪个动作。分析算法可以通过设置执行策略来解决这些冲突，例如采用深度优先搜索策略，只有在不能规约时才移进，有多种规约可能时选择最长的文法规则等策略。由于移进和规约操作不可撤销，因此即便输入的句子是符合语法的情况下，由于策略选择的问题，也可能分析失败，堆栈中不能规约到只有 S 为根的树的情况。为了解决这个问题也可以采用回溯策略，当分析失败时回溯顺次尝试不同选择。

	堆栈	动作	队列
0.	∅	∅	她 喜欢 跳 芭蕾
1.	她	shift	喜欢 跳 芭蕾
2.	P | 她	reduce	喜欢 跳 芭蕾
3.	P 喜欢 | 她	shift	跳 芭蕾
4.	P V | | 她 喜欢	reduce	跳 芭蕾
5.	P V 跳 | | 她 喜欢	shift	芭蕾
6.	P V V | | | 她 喜欢 跳	reduce	芭蕾
7.	P VP | /\ 她 V V | | 喜欢 跳	reduce	芭蕾
8.	P VP 芭蕾 | /\ 她 V V | | 喜欢 跳	shift	∅

图 4.10: 移进规约成分句法分析算法分析过程实例

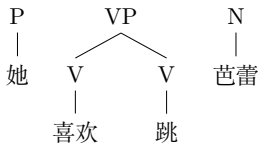
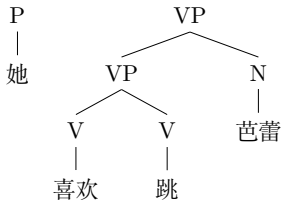
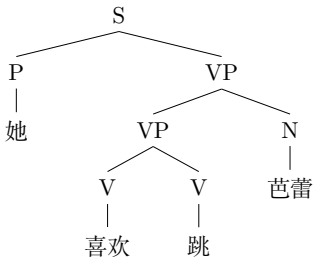
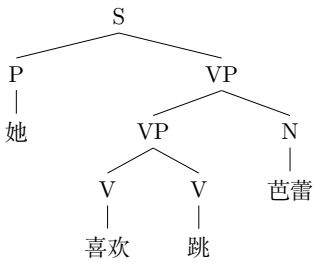
	堆栈	动作	队列
9.		reduce	∅
10.		reduce	∅
11.		reduce	∅
12.		accept	∅

图 4.11: 移进规约成分句法分析算法分析过程实例（续）

为了方便理解移进-规约分析算法，以下使用在上节介绍 CYK 算法时相同的文法 G，也以“她喜欢跳芭蕾”句子为例，介绍移进-规约算法的分析具体过

程，如图4.10和4.11所示。在初始化时，队列 Q 中保存句子中的所有单词，堆栈 S 为空。第 1 步进行移进操作，将句子的第一个单词 “ 她 ” 压入堆栈中。第 2 步利用文法 G 中的 “P-> 她 ” 规则生成子树，并将以 P 为根节点的子树压入堆栈中。第 3-6 步依次移进 “ 喜欢 ” 和 “ 跳 ” 并利用文法规则生成子树。接下来可以进行移进操作将句子中 “ 芭蕾 ” 压入栈顶，也可以利用文法 “VP -> V V” 进行规约操作，面临了移进规约冲突问题。在这里我们采用深度有限搜索策略，选择规约操作，并将生成的以 VP 为顶点的子树压入栈顶。第 8 步采用移进操作，将最后一个单词压入栈顶。此后通过第 9-11 步一系列的规约操作，最终生成的以 S 为根节点句法树，并保存在栈中。此时队列为空，最后执行接受操作，分析成功。

基于概率上下文无关文法的成分句法分析

上下文无关文法虽然比较简单且容易理解，但是对于句子结构歧义无法很好的处理。比如句子 “He eat soup with spoon” 具有两种可能的句法结构树，并且两种树结构都符合语法（如图4.12所示）。使用上下文无关文法很难从这两种句法树中进行选择。基于概率上下文无关文法（Probabilistic Context-Free Grammar，PCFG）的句法分析则可以结合规则方法和统计方法，在一定程度上解决歧义问题。

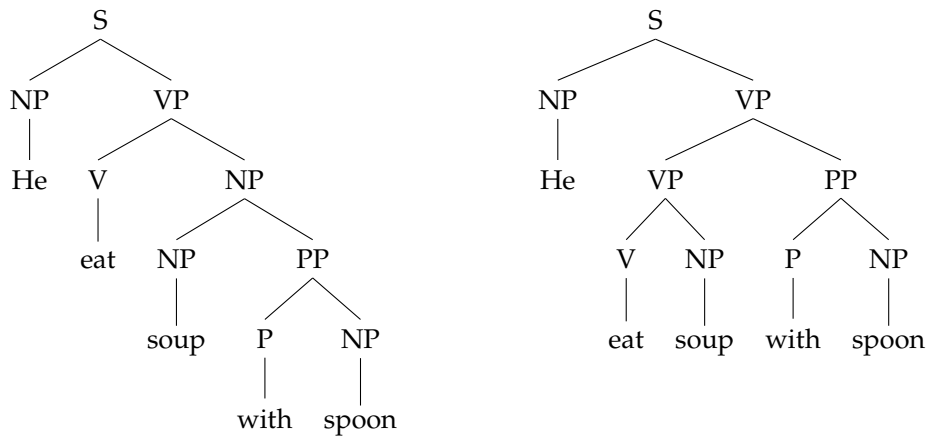


图 4.12: 句子 He eat soup with spoon 的成分语法树

PCFG 是 CFG 的扩展，因此 PCFG 的文法也是由终结符集合 $\sum$、非终结符集合 N、初始符 S 以及规则集合 R 组成。只是在 CFG 的基础上对每条规则增加了概率，其规则用如下形式表示：

$$A \rightarrow \alpha, p$$

其中 A 为非终结符，α 为终结符，p 为 A 推导出 α 的概率，即 $p = P(A \rightarrow \alpha)$，该概率分布要满足如下条件：

$$\sum_{\alpha} P(A \rightarrow \alpha) = 1 \tag{4.1}$$

也就是说，每一个非终结符展开得到的概率之和为 1。

由于 PCFG 中每个规则中包含了概率信息 $P(A \rightarrow \alpha)$，因此可以根据一个句子及其句法分析树计算特定句法分析树的概率、句子的概率以及句子片段的概率。利用特定分析树的概率可以用于消除分析树的歧义。接下来面临的问题就是如何利用 PCFG 构建句子的最佳树结构以及如何得到规则的概率参数。在本节中，将针对上述问题进行介绍。

PCFG 句法分析树概率计算

句法分析树概率计算是指在给定 PCFG 文法 G、句子 $W = w_1 w_2 \cdots w_n$ 以及句法树 T 的情况下，计算句法分析树概率 $P(T)$。为了简化句法树概率计算，句法树概率计算还要应用以下三个独立假设：

1. 位置不变性（Place in-variance）：子树的概率不依赖于该子树所在的位置；
2. 上下文无关性（Context-free）：子树的概率不依赖于子树以外的单词；
3. 祖先无关性（Ancestor-free）：子树的概率不依赖于子树的祖先节点。

基于上述独立性假设，一个特定句法树 T 的概率定义为该句法树 T 中用来得到句子 W 所使用的 m 个规则的概率乘积：

$$P(T) = \prod_{i=1}^{m} P(A_i \rightarrow \alpha) \tag{4.2}$$

假设给定如下 PCFG 文法：

G(S):	S → NP VP	1.0	NP → He	0.3
	VP → VP PP	0.8	NP → soup	0.3
	VP → V NP	0.2	NP → spoon	0.2
	NP → NP PP	0.2	V → eat	1.0
	PP → P NP	1.0	P → with	1.0

针对对于句子 “He eat soup with spoon” 的两种可能的句法结构包含概率的形式如图4.13所示。

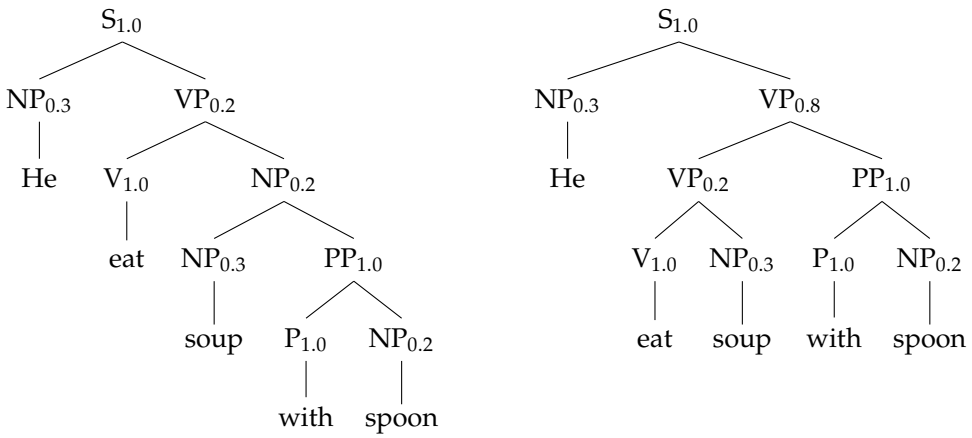


图 4.13: 句子 He eat soup with spoon 标注推导规则概率的成分语法树

根据句法树概率计算公式4.2，图4.13所示的两个句法树的概率分别为：

$$\begin{aligned}P(T_{Left}) =& P(S \rightarrow NP\ VP) \times P(NP \rightarrow He) \times P(VP \rightarrow V\ NP) \times \\ & P(V \rightarrow eat) \times P(NP \rightarrow NP\ PP) \times P(NP \rightarrow soup) \times \\ & P(PP \rightarrow P\ NP) \times P(P \rightarrow with) \times P(NP \rightarrow spoon) \\ =& 1.0 \times 0.3 \times 0.2 \times 1.0 \times 0.2 \times 0.3 \times 1.0 \times 1.0 \times 0.2 = 0.00072\end{aligned}$$

$$\begin{aligned}P(T_{Right}) =& P(S \rightarrow NP\ VP) \times P(NP \rightarrow He) \times P(VP \rightarrow VP\ PP) \times \\ & P(VP \rightarrow V\ NP) \times P(V \rightarrow eat) \times P(NP \rightarrow soup) \times \\ & P(PP \rightarrow P\ NP) \times P(P \rightarrow with) \times P(NP \rightarrow spoon) \\ =& 1.0 \times 0.3 \times 0.8 \times 0.2 \times 1.0 \times 1.0 \times 0.3 \times 1.0 \times 0.2 = 0.00288\end{aligned}$$

由此可以选择图4.13右侧树结构，从而解决针对句子 “He eat soup with spoon” 的句法树歧义问题。

PCFG 的句子概率计算

句子概率计算是指在给定 PCFG 文法 G 的情况下，计算给定句子 W 的概率 $P(W|G)$。在第二章中我们介绍了 N 元语言模型，由于 N 元语言模型仅考虑上下文中相邻的单词，缺乏对远距离单词以及句法信息的考虑。基于 PCFG 文法的句子概率计算则可以将这些信息引入概率计算。$P(W|G)$ 表示句子 W 所有的句法分析树的概率之和。可以采用**内向算法（inside algorithm）**或**外向算法（outside algorithm）**通过动态规划算法计算得到句子的概率。

采用内向算法，首先定义内变量 $a_{ij}(A)$ 为非终结符 A 推导出 W 中子串 $w_i w_{i+1} \cdots w_j$ 的概率，即：

$$\alpha_{ij}(A) = P(A \rightarrow w_i w_{i+1} \cdots w_j) \tag{4.3}$$

句子 W 的概率则相应的记为 $a_{1n}(S)$，即：

$$P(W|G) = P(S \rightarrow W|G) = \alpha_{1n}(S) \tag{4.4}$$

$alpha_{ij}(A)$ 可通过如下递归公式计算得到：

$$\alpha_{ii}(A) = P(A \rightarrow w_i) \tag{4.5}$$

$$\alpha_{ij}(A) = \sum_{B,C} \sum_{i \le k \le j} P(A \rightarrow BC)\alpha_{ik}(B)\alpha_{(k+1)j}(C) \tag{4.6}$$

上述公式表示当 $i \neq j$ 时，子串 $w_i w_{i+1} \cdots w_j$ 可以被切分成两端：$w_i \cdots w_k$ 和 $w_{k+1} \cdots w_j$，前半部分 $w_i \cdots w_k$ 是由非终结符 B 推导出，后半部分 $w_{k+1} \cdots w_j$ 是由非终结符 C 推导出，即 $A \rightarrow BC \rightarrow w_i \cdots w_k w_{k+1} \cdots w_j$，这一推导过程的概率为 $P(A \rightarrow BC)\alpha_{ik}(B)\alpha_{(k+1)j}(C)$。具体过程如算法4所示。

Algorithm 4: 面向 PCFG 句子概率求解的内向算法

输入: PCFG G(S) 和句子 $w_1 w_2 \cdots w_n$
输出: $\alpha_{ij}(A), i \le i \le j \le n$

1: **for** $i = 1$ to n **do**
2: $\quad \alpha_{ii}(A) = P(A \rightarrow w_i), 1 \le i \le n;$
3: **end for**
4: **for** $j = 1$ to n **do**
5: $\quad$ **for** $i = 1$ to $n - j$ **do**
6: $\quad\quad \alpha_{i(i+j)}(A) = \sum_{B,C} \sum_{i \le k \le i+j-1} P(A \rightarrow BC)\alpha_{ik}(B)\alpha_{(k+1)(i+j)}(C)$
7: $\quad$ **end for**
8: **end for**

计算给定句子 W 的概率 $P(W|G)$ 也可以采用**外向算法（outside algorithm）**利用动态规划方法得到。外向变量 $\beta_{ij}(A)$ 定义为初始非终结符 S 在推导出句子 W 的过程中产生符号串 $w_1 w_2 \cdots w_{i-1}\ A\ w_{j+1} \cdots w_n$ 的概率（$A \rightarrow w_i \cdots w_j$）：

$$\beta_{ij}(A) = P(S \rightarrow w_1 w_2 \cdots w_{i-1} A w_{j+1} \cdots w_n) \tag{4.7}$$

$\beta_{ij}(A)$ 可通过如下递归公式计算得到：

$$\beta_{1n}(A) = \begin{cases} 1, A = S \\ 0, A \neq S \end{cases} \tag{4.8}$$

$$\begin{aligned} \beta_{ij}(A) \quad = \quad & \sum_{B,C} \sum_{k \geq j} P(B \rightarrow AC) \alpha_{(j+1)k}(C) \beta_{ik}(B) \\ & + \sum_{B,D} \sum_{k \leq i} P(B \rightarrow DA) \alpha_{k(i-1)}(D) \beta_{kj}(B) \end{aligned} \tag{4.9}$$

上述公式表示当 $i = 1, j = n$ 时，如果 $A = S$，那么 $\beta_{1n}(A) = P(S \rightarrow W)$ 按照 PCFG 的定义 $P(S \rightarrow W) = 1$，即 $\beta_{1n}(A) = 1$。如果 $A \neq S$，同样按照 PCFG 的定义以及规范语法中不存在 $S \rightarrow A$ 的推导规则，则 $\beta_{1n}(A) = 0$。当 $i \neq 1$ 或 $j \neq n$ 时，如果在 S 推导出 W 的过程中出现了符号串 $w_1 w_2 \cdots w_{i-1}\ A\ w_{j+1} \cdots w_n$，那么根据上下文无关语法的性质一定存在 $B \rightarrow AC$ 或者 $B \rightarrow DA$ 的规则。因此构造上述递推公式。具体过程如算法5所示。

Algorithm 5: 面向 PCFG 句子概率求解的外向算法

输入: PCFG G(S) 和句子 $w_1 w_2 \cdots w_n$
输出: $\beta_{ij}(A), i \leq i \leq j \leq n$

1: $\beta_{1n}(A) = \begin{cases} 1, A = S \\ 0, A \neq S \end{cases}$
2: **for** $j = n - 1$ to 0 **do**
3: **for** $i = 1$ to $n - j$ **do**
4: $\beta_{ij}(A) = \sum_{B,C} \sum_{k \geq j} P(B \rightarrow AC) \alpha_{(j+1)k}(C) \beta_{ik}(B) + \sum_{B,D} \sum_{k \leq i} P(B \rightarrow DA) \alpha_{k(i-1)}(D) \beta_{kj}(B)$
5: **end for**
6: **end for**

PCFG 的最佳树结构求解

最佳树结构求解是指对于给定句子 $W = w_1 w_2 \cdots w_n$ 和 PCFG 文法 G，求解该句子的最佳树结构，即如何选择句法结构树使得其概率最大：

$$\hat{t} = \arg\max_{t \in \mathcal{T}_G(W)} P(t|W, G) \tag{4.10}$$

$\mathcal{T}_G(W)$ 表示句子 W 所有符合文法 G 的句法树。该问题可以通过利用基于概率的 CYK 算法进行。与 CYK 算法一样，概率 CYK 算法也要求所使用的语法必须符合乔姆斯基范式（CNF），其规则被限制为 $A \rightarrow BC$ 或 $A \rightarrow w$ 两种形式。与 CYK 算法一样，概率 CYK 算法也将输入的句子表示为单词之间带有索引号的句子形式。

例如：$_0$ He $_1$ eat $_2$ soup $_3$ with $_4$ spoon $_5$。

针对句子长度为 n 的句子，概率 CYK 算法同样也使用 $(n+1) \times (n+1)$ 的矩阵 T 的上三角形部分进行存储。T_{ij} 表示输入句子中横跨在位置 i 到 j 之间的单词的组成成分，包含由文法 G 的非终结符构成的集合以及以该非终结符为根的句法树的概率。为了方便期间这里我们使用 $T_{ij,A}$ 表示矩阵 T_{ij} 保存的非终结符集合中 A 的概率。算法6给出了概率 CYK 算法的基本流程。

以下通过一个实例来说明使用 CYK 算法求解 PCFG 的最佳树结构具体流程。给定如下符合乔姆斯基范式的文法 G 如下：

非终结符号集合：$N = \{S, P, V, NP, PP, VP\}$

终结符号集合：$\sum$ = { eat, he, soup, spoon, with}

规则集合：R = {

```
(1) S -> NP VP  1.0   (2) VP -> VP PP 0.8  (3) VP -> V NP  0.2
(4) NP -> NP PP 0.2   (5) PP -> P NP  1.0  (6) NP -> He    0.3
(7) NP -> soup  0.3   (8) NP -> spoon 0.2  (9) V -> eat    1.0
(10) P -> with  1.0
```

Algorithm 6: 概率 CYK 句法分析算法

输入: 语法信息 G 和句子 $w_1 w_2 \cdots w_n$;
输出: 句法树矩阵 T

```
    // 初始化
 1: for i = 1 to n do
 2:     T_ii = w_i                          ▶ 主对角线上依次放入单词 w_i
 3:     for all A|A → w_i do
 4:         T_(i-1)i = T_(i-1)i ∪ (A, p), (A → w_i, p ∈ G)      ▶ 依次放入单词 w_i 的所
            有词性及其概率
 5:     end for
 6: end for
    // 平行于主对角线逐层计算
 7: for d = 2 to n do
 8:     for i = 0 to n − d do
 9:         j = i+d
10:         for k = i + 1 to j − 1 do
11:             if (A → BC, p) ∈ G and T_ik,B > 0 and T_kj,C > 0 then
12:                 if T_ij,A < p × T_ik,B × T_kj,C then
13:                     T_ij = T_ij ∪ {A, p × T_ik,B × T_kj,C, (k, B, C)}      ▶ (k,B,C) 用于保存
                        推导路径
14:                 end if
15:             end if
16:         end for
17:     end for
18: end for
```

}

图4.14和图4.15给出了概率 CYK 算法分析句子 “He eat soup with spoon ” 的过程。首先初始化矩阵主对角线上的单词以及单词所对应的词性和概率。之后进行 $T_{02}, T_{13}, T_{24}, T_{35}$ 元素的计算，通过规则集合中 VP->V NP 0.2 的推导规则，可以得到在 T_{13} 添加 VP 以及相应的路径信息，其概率为 $0.2 \times 1.0 \times 0.3 = 0.06$。根据 PP->P NP 1.0 的推导规则，在 T_{35} 添加 PP 以及相应的路径信息，其概率为 $1.0 \times 1.0 \times 0.2 = 0.2$。依次逐层进行分析，在对 $T_{1,6}$ 进行分析时，分别根据 VP->VP PP 0.8 和 VP->V NP 0.2 两个不同的推导规则得到 VP_1，其概

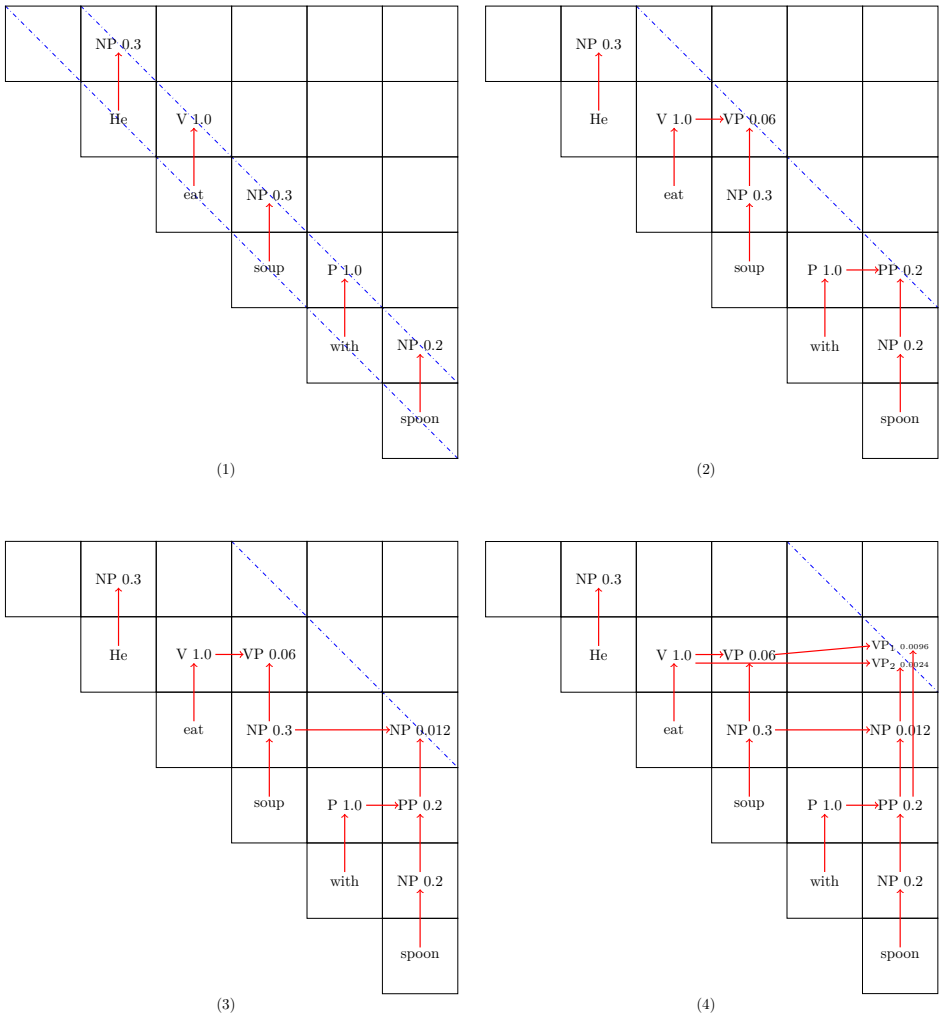


图 4.14: 使用 CYK 算法求解 PCFG 的最佳树结构分析实例

率为 $0.8 \times 0.06 \times 0.2 = 0.0096$，$VP_2$ 的概率为 $0.2 \times 1.0 \times 0.012 = 0.0024$。据此在 T_{05} 节点可以根据 S->NP VP 1.0，得到两个不同分析结果 S_1 的概率为 $1.0 \times 0.3 \times 0.0096 = 0.00288$ 和 S_2 的概率为 $1.0 \times 0.3 \times 0.0024 = 0.00072$。根据不同所得到的不同分析树的概率大小，选择最终结果。

通过上述例子我们可以看到，利用构建好的 PCFG 文法，利用概率 CYK 分

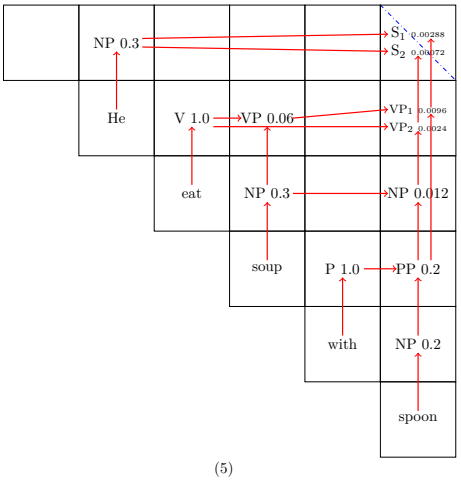


图 4.15: 使用 CYK 算法求解 PCFG 的最佳树结构分析实例 (续)

析等最佳树结构求解算法，可以在一定程度上对句法分析中的歧义进行很好的处理，根据概率选择可能性较大的树结构。

PCFG 的模型参数学习

模型参数学习是指给定 CFG 文法 G 下，如何学习 G 中规则的概率参数构建 PCFG 文法。有两种常见的方法用于学习语法规则概率：1）有大规模句法树标注数据（树库）；2）有大规模的没有标注句法树的句子。

在有大规模标注句法树标注的情况下，可以基于最大似然估计，统计非终结符的出现次数进行概率参数估计：

第三章中介绍的宾州大学树库（PTB）就是一个典型的大规模句法树标注语料库

$$P(A \rightarrow \alpha) = \frac{Count(A \rightarrow \alpha)}{\sum_{\lambda} Count(A \rightarrow \lambda)} = \frac{Count(A \rightarrow \alpha)}{Count(A)} \tag{4.11}$$

$Count(A \rightarrow \alpha)$ 是指规则 $A \rightarrow \alpha$ 在整个树库中出现的次数，$Count(A)$ 是指在树库中非终结符 A 出现的次数。

在仅有大规模无标记句子的情况下，也可以通过期望最大化算法（Expectation M，EM）估计规则的概率参数。基本思想是首先对给定的 CFG 文法 G 中的每个规则，在满足归一化条件的情况下随机赋值概率值，构造文法 G_0。在此基础上利用 G_0 对所有句子进行分析，计算出每条规则使用次数的期望值。之后利用期望次数根据最大似然估计原理，得到文法 G 的概率参数更新，记为 G_1。循环执行该过程直到 G 的概率参数收敛于最大似然估计值。利用当前的文法 G_i 估算每条规则出现的期望值是期望步（Expectation Step，E-步骤），重新估算概率得到 G_{i+1} 的步骤是最大化步（Maximization Step，M-步骤）。具体的计算公式可以参考 [34, 35]。

4.3 依存句法分析

依存句法分析（Dependency Parsing）是依据依存语法理论，分析输入句子得到其依存句法结构树。依存句法的基本假设是句法结构由单词和单词之间的依存关系组成。依存关系具有方向性，从中心语成分指向依存成分。依存关系根据中心成分和依存成分之间的关系又可以被划定义为不同的依存关系类型。依存句法结构使用依存图（Dependency Graph）进行表示。如图4.16所示。

依存语法理论介绍参考 4.1 节

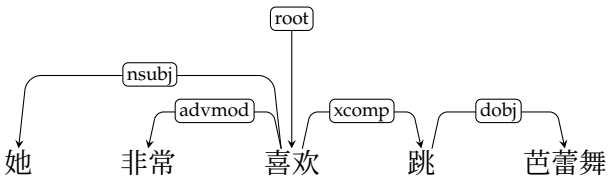


图 4.16: 句子“她非常喜欢跳芭蕾舞”的依存句法树

在本节中，使用 $S = w_0, w_1, \ldots w_n$ 表示输入的句子，其中 $w_0 = root$ 表示虚拟根节点，$w_1 \ldots w_n$ 为输入句子中的 n 个单词。$R = \{r_0, r_1, \ldots, r_m\}$ 表示依存关系类型集合，$r \in R$ 表示在句子中单词之间的依存关系，也叫做边标签（arc label）。例如在图4.16中，“喜欢”和“她”之间的依存关系类型 $r = nsubj$。依存图 $G = (V, A)$ 是一个有标记的有向图，顶点 V 和边 A 针对句子 S 和依存关系集合符合以下要求：

1. $V \subseteq \{w_0, w_1, \cdots w_n\}$
2. $R \subseteq V \times R \times V$
3. 如果 $(w_i, r, w_j) \in A$，那么 $(w_i, r, w_j) \notin A, \forall r' \neq r$

边 (w_i, r, w_j) 表示以 w_i 为头 w_j 为尾边标记为 r 的边，其表示以 w_i 为中心词，w_j 为修饰词，类型为 r 的依存关系。依存图 G 表示了一组句子 W 中的单词之间的有标记的依存关系。依存图中的节点集合 V 通常包含句子中的所有单词，针对句子 S 通常用 $V_S = \{w_0, w_1, \cdots w_n\}$ 表示。

根据上述定义图4.16可以表示为：

1. $G = (V, A)$
2. $V = V_W$ = {ROOT, 她, 非常, 喜欢, 跳, 芭蕾舞 }
3. A = {(ROOT, root, 喜欢), (喜欢, nsubj, 她), (喜欢, advmod, 非常), (喜欢, xcomp, 跳), (跳, dobj, 芭蕾舞) }

如果依存图 $G = (V, A)$ 对于输入句子 S 和关系集合 R，是一个从 w_0 出发的有向树并且包含句子中的所有单词，那么这依存图 G 就成为*形式良好的依存图（well-formed dependency graph）*，也称为**依存树（Dependency Tree）**。对于输入句子 W 和关系集合 R，所有形式良好的依存图集合记为 $\mathcal{G}_S$。依存句法分析就是对输入句子 W 根据一定的原则从 $\mathcal{G}_S$ 中选择得分最高的依存句法树。

基于图的依存句法分析

基于图的依存句法分析核心是构造评分函数对句子 S 所有依存句法树 $G = (V, A) \in \mathcal{G}_W$ 进行评分。这个评分代表了一个句法树做为句子分析正确结果的可能性。不同的基于图的分析方法采用不同的假设来计算该得分。基于图的依存句法分析算法通常将对依存句法树的 $G = (V, A)$ 的评分转化为对其树上的边的评分：

$$score(G_S) = \sum_{(w_i, r, w_j) \in A} \lambda_{(w_i, r, w_j)} \tag{4.12}$$

λ 表示所有边的评分，$\lambda_{(w_i,r,w_j)}$ 表示边 (w_i, r, w_j) 的得分，其具体计算方法将在本节后续部分进行详细介绍。基于图得分计算，可以将基于图的依存句法分析形式化表示为：

$$\hat{G} = \underset{G=(V,A)\in\mathcal{G}_S}{\arg\max} \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} \tag{4.13}$$

可以证明在对依存句法树不考虑投射性（Projectivity）的情况下，对于输入句子 S 的依存句法分析问题等价于基于边评分 $\lambda_{(w_i,r,w_j)}$ 的图 G_S 的最大生成树（maximum spanning tree）寻找问题 [36]。此外，可以定义 $\lambda_{(w_i,w_j)} = \max_r \lambda_{(w_i,r,w_j)}$，并利用 $\lambda_{(w_i,w_j)}$ 作为边的权重构建图 G'。通过图 G' 得到的最大生成树 $\hat{G}'$，可以证明与通过 $\lambda_{(w_i,r,w_j)}$ 构建得到的最大生成树 $\hat{G}$ 相同 [36]。由此，可以将包含依存关系类别的分析转换为无依存关系类别的分析，进一步降低分析算法的复杂性。

需要注意的是，利用最大生成树算法得到的依存句法树不具备投射性。针对具有投射性要求的依存句法树，可以利用其与上下文无关语法之间的强相关性，利用基于 CYK 算法等上下文无关语法分析算法进行依存句法树分析。在本节中我们将针对上述两种情况分别进行介绍。

非投射性依存句法分析方法

由上节的介绍我们可以知道，非投射性的依存句法分析等价于图的最大生成树寻找问题。朱-刘/埃德蒙兹算法（Chu-Liu/Edmonds）方法 [37, 38] 是一种常见的带权有向图最小/大生成树寻找算法，因此也常被应用于非投射性依存句法分析。朱-刘/埃德蒙兹算法的核心是采用贪心的思想对边进行选择，利用递归方法来实现整个过程。

朱-刘/埃德蒙兹算法用于非投射性依存句法分析的输入是待分析句子 $S = w_0, w_1, \ldots, w_n$ 和边之间权重 $\lambda_{(w_i,w_j)} \in \lambda$。根据定义，$w_0$ 是句子的虚拟根节点，依存句法树中不存在指向 w_0 的边，因此设置所有 $\lambda_{(w_i,w_0)} = -\infty$。分析算法首先根据句子中的单词和权重组成有向图 $G_S = (V_S, A_S)$，$V_S = \{w_0, w_1, \ldots, w_n\}$，

$A_S = \{(w_i, w_j)|\forall w_i, w_j \in V_S\}$。朱-刘/埃德蒙兹算法的第一步是针对图 G 中每个顶点选择入边权重最大的边构建子图 $G' = (V_S, A')$。如果该子图中没有环，那么该子图就是图 G 的最大生成树。否则，说明图 G' 中至少包含一个环。那么选择其中任意一个环 C，其边集合为 A_C，将环 C 用一个节点 w_c 来代表，图 G_C 中包含所有在图 G 中但是不在环 C 中的节点，以及 w_C，G_C 的边通过以下规则构建：

1. 对于所有不在环 C 的顶点 w_j，如果存在一个边 (w_i, w_j)，w_i 在环 C 中，那么就添加边 (w_c, w_j) 到 G_C 中，定义 $ep(w_c, w_j) = \arg\max_{w_i \in C} \lambda_{(w_i, w_j)}$，用于记录环 C 中相对应的顶点，相应的修改 (w_c, w_j) 的边权重 $\lambda_{(w_c, w_j)} = \lambda_{(ep(w_c, w_j), w_j)}$；
2. 对于所有不在环 C 的顶点 w_i，如果存在一个边 (w_i, w_j)，w_j 在环 C 中，那么添加边 (w_i, w_c) 到 G_C 中，定义 $ep(w_i, w_c) = \arg\max_{w_j \in C} \left[\lambda_{(w_i, w_j)} - \lambda_{(a(w_j), w_j)}\right]$，相应的修改 (w_i, w_c) 的边权重 $\lambda_{(w_i, w_c)} = \lambda_{(w_i, ep(w_i, w_c))} - \lambda_{a(ep(w_i, w_c)), (ep(w_i, w_c))} + \sum_{w \in C} \lambda_{(a(w), w)}$；
3. 对于所有边 (w_i, w_j)，其顶点 w_i 和 w_j 都不在环 C 中，直接添加该边到图 G' 中，保持原有权重不变

将图 G_C 作为输入递归调用上述算法得到其最大生成树 $G = (V, A)$。之后根据所返回的最大生成树信息对原始图信息进行修正，移除环 C，并且将 w_c 所指向的节点的实际边还原，返回输入图所对应的最大生成树。朱-刘/埃德蒙兹算法的伪代码如算法7所示。

投射性依存句法分析方法

设置虚拟根节点在句子首位的情况下，投射性依存句法分析树等价于嵌套依存句法分析树（Nested Dependency Trees）。因此投射性依存句法分析与上下文无关语法具有非常强的关系，很多用于上下文无关语法分析算法也可以应用于投射性依存句法分析。在 4.2 节中介绍了 CYK 算法用于 CFG 和 PCFG 的句法分析，本节中将介绍基于 CYK 算法改进的依存句法分析算法。

Algorithm 7: 朱-刘/埃德蒙兹算法求解依存句法分析树

输入: 图 $G=(V,A)$，$\lambda_{(w_i,w_j)} \in \lambda$

输出: 最大生成树图 G

Function `Chu-Liu-Edmonds(`G,λ`)`**:**

$A' = \{(w_i,w_j) | w_j \in V, w_i = \arg\max_{w_i} \lambda_{(w_i,w_j)}\}$;

$G' = (V,A')$;

if G' *中没有环* **then**

  return G';

A'_C= 图 G' 中任意一个环 C 的边集合;

V'_C= 环 C 顶点集合;

$V_C = V \cup \{w_c\} - V'_C$;

foreach $w_j \in V - V'_C : \exists_{w_i \in V'_C}(w_i,w_j) \in A$ **do**

  $A_C = A_C \cup \{(w_c,w_j)\}$;

  $ep(w_c,w_j) = \arg\max_{w_i \in C} \lambda_{(w_i,w_j)}$;

  $\lambda_{(w_c,w_j)} = \lambda_{(ep(w_c,w_j),w_j)}$;

foreach $w_i \in V - V'_C : \exists_{w_j \in V'_C}(w_i,w_j) \in A$ **do**

  $A_C = A_C \cup \{(w_i,w_c)\}$;

  $ep(w_i,w_c) = \arg\max_{w_j \in C} \left[\lambda_{(w_i,w_j)} - \lambda_{(a(w_j),w_j)}\right]$;

  $\lambda_{(w_i,w_c)} = \lambda_{(w_i,ep(w_i,w_c))} - \lambda_{a(ep(w_i,w_c)),(ep(w_i,w_c))} + \sum_{w \in C} \lambda_{(a(w),w)}$;

foreach $w_i \in V - V'_C : \exists_{w_j \in V - V'_C}(w_i,w_j) \in A$ **do**

  $A_C = A_C \cup \{(w_c,w_j)\}$;

G_C=(V_C,A_C);

$G = (A,V)$ = Chu-Liu-Edmonds(G_C,λ);

寻找 A 中指向 w_c 的边 (w_i,w_c)，w_j=$ep(w_i,w_c)$;

寻找环 C 中指向 w_j 的边 (w_k,w_j);

寻找 A 中所有以 w_c 为头节点的边 (w_c,w_l);

A=$A \cup (ep(w_c,w_l),wl)_{forall(w_c,w_l) \in A} \cup A_c \cup (w_i,w_j) - (w_k,w_j)$;

return G;

首先定义动态规划表 C[s][t][i]（$s \le i \le t$），表示投射性句法树以单词 w_i 为根节点覆盖从单词 w_s 到单词 w_t 的句子片段的最高得分。由此可以得到 C[0][n][0] 对于输入句子 $S = w_0, w_1, \ldots, w_n$ 的依存句法树的最高得分。任何以

w_i 为根节点覆盖 w_s 到 w_t 的投射性句法树都是由更小的子树构成。同时更大的子树是通过相邻的子树由内向外添加依存关系得到。因此对于 C[s][t][i] 可以构造如下递推公式：

$$C[s][t][i] = \max_{s \le k \le t, s \le j \le t} \begin{cases} C[s][k][i] + C[k+1][t][j] + \lambda(w_i, w_j), & \text{如果} j > i \\ C[s][k][j] + C[k+1][t][i] + \lambda(w_i, w_j), & \text{如果} j < i \end{cases} \tag{4.14}$$

图4.17给出了合并过程示意图。

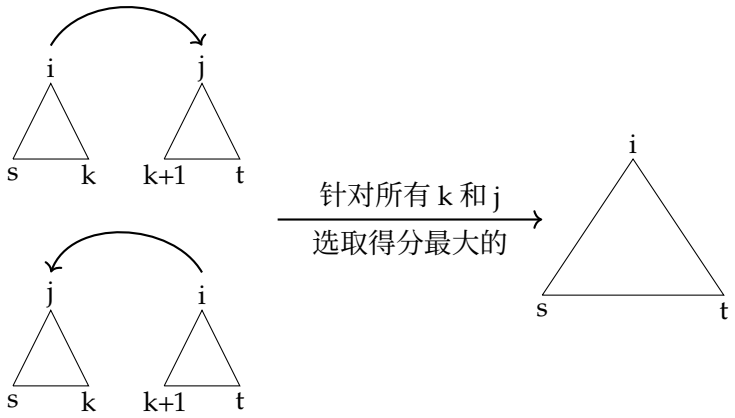


图 4.17: 面向依存句法分析的 CYK 算法递推样例

由于 C[s][t][i] 中仅保存了子树的最高得分，但是没有保存子树的结构。因此在实际构建过程中，还需要增加树结构矩阵 A[s][t][i]，用于保存依存句法树结构。通过公式4.14计算得到最优的 k 和 j 之后，A[s][t][i] 按照如下公式记录树结构：

$$A[s][t][i] = \begin{cases} A[s][k][i] \cup A[k+1][t][j] \cup (w_i, w_j), & \text{如果} j > i \\ A[s][k][j] \cup A[k+1][t][i], & \text{如果} j < i \end{cases} \tag{4.15}$$

针对句子 S 得到的依存句法树 $G = (V, A[0][n][0])$ 组成。

边评分模型学习方法

边评分模型可以使用基于高维特征向量的线性函数进行建模。使用 $\mathbf{f}(w_i, r, w_j) \in \mathbb{R}^m$ 表示特征函数，其输出是高维特征向量，$\mathbf{w}$ 是 $\mathbf{f}(\cdot)$ 的对应的权重向量。边评分参数 $\lambda_{(w_i,r,w_j)}$ 可以表示为：

$$\lambda_{(w_i,r,w_j)} = \mathbf{w} \cdot \mathbf{f}(w_i, r, w_j) \tag{4.16}$$

$\mathbf{f}(\cdot)$ 包含了边和输入句子 S 中各类型相关特征，例如：

- w_i= 喜欢
- w_j= 跳
- w_i 的词性 =V
- w_j 的词性 =V
- r 的依存关系类型 =xcomp
- w_{i-1} 的词性 =ADV
- w_{j+1} 的词性 =N
- w_i 和 w_j 之间距离 =1

这些特征还可以组合为更复杂的类型例如：

- w_i 的词性 =V & w_j 的词性 =V & w_i= 喜欢
- w_i= 喜欢 & w_j= 跳 & w_i 和 w_j 之间距离 =1

类别特征通常也会通过二值化操作转换为 0/1 特征，具有 m 个类别的特征通常会转化为 m 个 0/1 特征，表示某个类别出现或不出现。从上述特征的构成我们可以看到，特征向量维度通常会非常高，但是对于一个边来说特征向量非常稀疏，仅有非常小部分的特征不是 0。因此可以利用稀疏性进行表示和计算。

基于特征向量的线性函数建模的假设下，对于输入句子的依存句法分析转

换为了如下问题：

$$\hat{G} = \underset{G=(V,A)\in\mathcal{G}_S}{\arg\max} \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} = \underset{G=(V,A)\in\mathcal{G}_S}{\arg\max} \sum_{(w_i,r,w_j)\in A} \mathbf{w}\cdot\mathbf{f}(w_i,r,w_j) \quad (4.17)$$

训练语料用 $D=\{(S_d,G_d)\}_{d=1}^{|D|}$ 表示，其中 S_d 表示输入句子，所对应的正确的依存句法树用 G_d 表示。针对输入句子可以利用特征函数 $\mathbf{f}(\cdot)$ 构造句子中所有单词对和根据不同依存关系类型输出特征向量。边评分模型的学习就转换为了权重向量 $\mathbf{w}$ 的学习问题。该问题可以采用感知器算法（Perceptron Algorithm）完成，伪代码如算法8所示。

感知器算法是一种典型的基于推理（Inference-based Learning）的在线学习方法，也称为错误驱动（Error-driven）的学习算法

Algorithm 8: 感知器算法学习参数向量 $\mathbf{w}$

输入: 训练语料 $D=\{(S_d,G_d)\}_{d=1}^{|D|}$
输出: 权重向量 $\mathbf{w}$

Function `Perceptron(`D`)`:
  $\mathbf{w}$=0;
  for $n:1\ldots N$ **do**
    for $d:1\ldots|D|$ **do**
      $G'=\underset{G=(V,A)\in\mathcal{G}_{S_d}}{\arg\max} \sum_{(w_i,r,w_j)\in A} \mathbf{w}\cdot\mathbf{f}(w_i,r,w_j)$;
      if $G\neq G'$ **then**
        w=$w+\sum_{(w_i,r,w_j)\in A_d}\mathbf{f}(w_i,r,w_j)-\sum_{(w_i,r,w_j)\in A'}\mathbf{f}(w_i,r,w_j)$;
  return $\mathbf{w}$

感知器算法每次仅考虑一个训练样本，利用当前的权重向量 $\mathbf{w}$ 和依存树构建算法针对当前样本 S_d 构建依存句法树 G'。如果 G' 与标准标注 G_d 不同，则通过增加正确依存树所对应的边的特征向量并减去不正确的句法树分析结果的边更新当前的权重向量 $\mathbf{w}$。最大生成树依存句法分析器（Maximum Spanning Tree Dependency Parser，MSTParser）[39] 采用了上述方法，并在上述基础上增加了最大间隔（Max Margin）目标函数并引入了边缘注入松弛算法（Margin-infused relaxed algorithm，MIRA），进一步提升了所学习得到的权重向量的泛化能力。

基于神经网络的图依存句法分析

基于图的依存句法分析主要包含边评分模型和句法树生成算法两个部分组成。其中边评分模型对于分析效果具有决定性的影响。传统的分析方法依赖人工设计的数百万甚至数千万特征，模型的泛化能力不高，很容易出现过拟合问题。此外，由于特征函数中使用大量的单词关联信息，也很容易使得特征空间巨大。神经网络方法可以在一定程度上缓解上述问题，同时也不需要人工设计特征函数。在本节中，将介绍三种基于神经网络图依存句法分析算法。

Pei 等人 [40] 提出的依存句法分析器采用最大生成树方法构造依存句法树。但是在边评分模型方面他们提出了仅利用最基本信息作为输入的神经网络方法。与上节介绍的方法需要利用人工构造特征模板对输入句子中的两个单词转化为特征向量表示。Pei 等人提出的算法的输入仅为包括单个单词、单个单词词性、单词之间距离等在内的基本信息。所采用的神经网络结构如图4.18所示。

[40]: Pei et al. (2015), "An effective neural network model for graph-based dependency parsing"

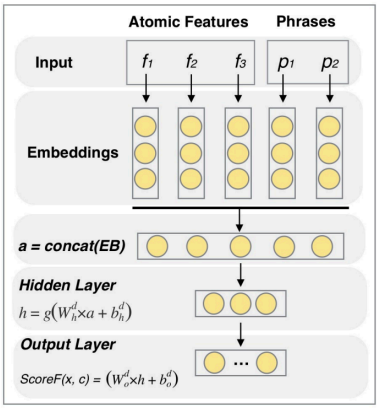


Figure 2: Architecture of the Neural Network

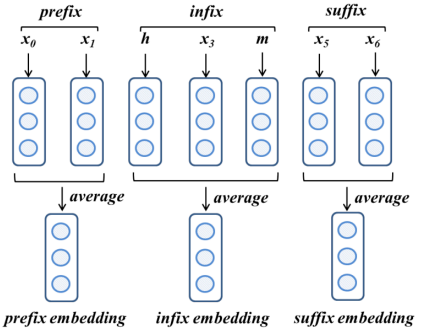


Figure 3: Illustration for phrase embeddings. h, m and x_0 to x_6 are words in the sentence.

图 4.18: Pei 等人提出的基于神经网络的边评分模型网络结构图 [40]

输入的基本信息首先通过查找嵌入矩阵 $M = \mathbb{R}^{d \times |D|}$ 转换为相应的嵌入（Embedding）表示，同时通过向量均值组合成短语嵌入（Phrase Embedding）。

第二层将基本信息的分布式表示和短语嵌入合并到单个向量 a。第三层隐藏层利用激活函数 tanh-cube 学习线性变换后的向量 a 多个特征的综合。tanh-cube 函数 $g(l) = tanh(l^3 + l)$。最后一层输出层利用线性变换得到边评分函数。通过定义结构间隔距离使用最大间隔标准进行模型训练。

Dozat 和 Manning 提出的 Deep Biaffine Parser[41]，也是采用最大生成树方法构造依存句法树。边评分模型的神经网络结构如图4.19所示。

[41]: Dozat et al. (2017), "Deep Biaffine Attention for Neural Dependency Parsing"

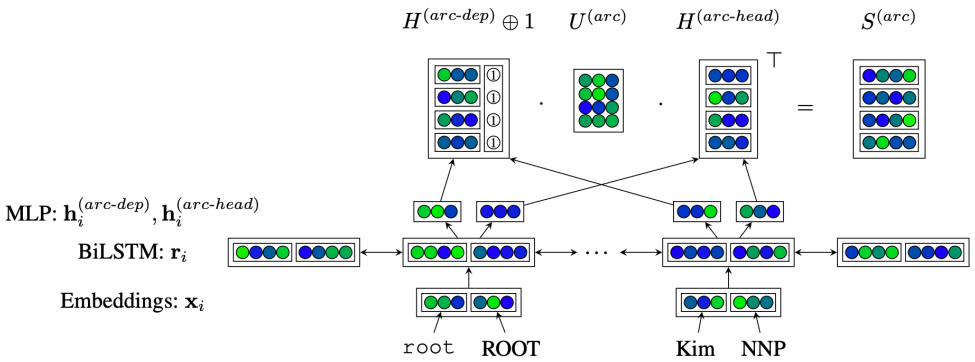


图 4.19: Deep Biaffine Parser 边评分模型神经网络结构图 [41]

句子中所有单词的词嵌入和对应的词性嵌入合并作为双向长短时记忆网络（BiLSTM）的输入，记为 $\mathbf{x}_i = v_i^{word} \oplus v_i^{POS}$。所有单词经过 BiLSTM 层计算后得到每个时刻的输出记为 $\mathbf{r}_i$。接下来，使用多层感知器（MLP）对 $\mathbf{r}_i$ 用于中心词和修饰词的不同，分别进行两种不同的降维 $\mathbf{h}_i^{(arc-head)}$ 和 $\mathbf{h}_i^{(arc-dep)}$：

$$\mathbf{h}_i^{(arc-head)} = \mathbf{MAP}^{(arc-head)}(\mathbf{r}_i) \tag{4.18}$$

$$\mathbf{h}_i^{(arc-dep)} = \mathbf{MAP}^{(arc-dep)}(\mathbf{r}_i) \tag{4.19}$$

所有时刻的 $\mathbf{h}_i^{(arc-head)}$ 和 $\mathbf{h}_i^{(arc-dep)}$ 分别合并组成矩阵 $H^{(arc-head)}$ 和 $H^{(arc-dep)}$。之后利用不定类别双仿射分类器（Variable-class Biaffine Classifier）对 $H^{(arc-dep)}$

额外拼接了一个单位向量，利用矩阵 $U^{(arc)}$ 进行仿射变换，得到边评分矩阵：

$$S^{(arc)} = H^{(arc-dep)} \bigoplus 1 \cdot U^{(arc)} \cdot H^{(arc-head)^{\top}} \tag{4.20}$$

由于依存关系类别数目是确定的，针对类别的分类问题，可以采用下述公式计算单词 w_i 作为修饰词 w_{y_i} 做为中心词的依存关系类别得分：

$$\mathbf{s}_i^{(label)} = \mathbf{r}_{y_i}^{\top} \mathbf{U}^{(1)} \mathbf{r}_i + \left(\mathbf{r}_{y_i} \bigoplus \mathbf{r}_i\right)^{\top} U^{(2)} + \mathbf{b} \tag{4.21}$$

Ji 等人同样采用了在对边评分的基础上利用贪心或者最大生成树等算法构建依存句法生成树的框架，但是提出了基于图神经网络的边评分算法方法 [42]，算法的网络架构如图4.20所示。

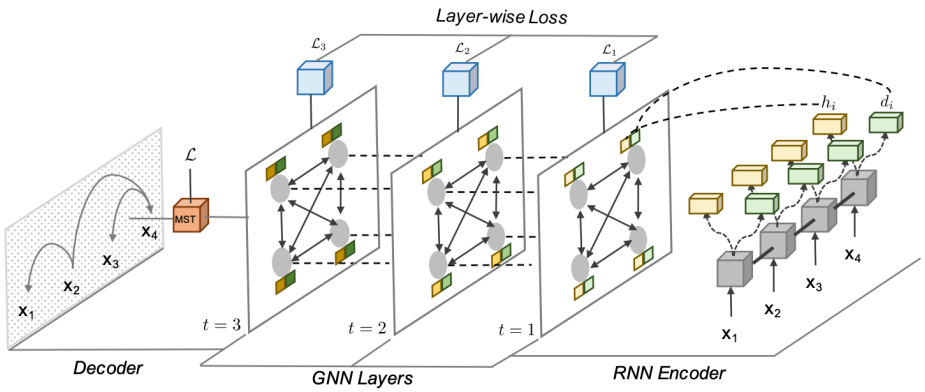


图 4.20: 基于图神经网络的依存句法分析网络结构图 [42]

基于转移的依存句法分析

转移系统（Transition system）包含状态（state 或 configuration）集合以及状态之间的转移动作（transition）集合。在第三章中所介绍的用于词性分析的有限状态自动机就是属于转移系统的最简单的一种。应用于依存句法分析的

转移系统更为复杂，标准弧（arc-Standard）转移系统 [43] 是其中最常用的投射性依存句法分析转移系统之一。按照标准弧转移系统定义，状态 $c = (\sigma, \beta, A)$ 由三个部分组织，σ 表示已处理的单词的堆栈，β 表示尚未处理的单词的缓冲器，A 表示已经构建的依存关系边集合。对于给定的句子 $S = w_0, w_1, \ldots, w_n$，其初始状态 $c_0(S) = ([w_0]_\sigma, [w_1, \ldots, w_n]_\beta, [\varnothing]_A)$，$(\sigma, []_\beta, A)$ 则对应终止状态的形式。标准弧转移系统中转移动作包含以下三类：

[43]: Nivre (2008), "Algorithms for deterministic incremental dependency parsing"

1. **左弧**（LA_r）：$([\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma, w_j|\beta, A \cup \{(w_j, r, w_i)\})$，$w_i$ 是堆栈 σ 最上面的单词，w_j 是缓冲器 β 中最前面的单词，将 w_i 从堆栈中弹出并增加从 w_j 到 w_i 关系类型为 r 的依存关系。前置条件是 $i \neq 0$ 并且 $\neg\exists w_k \exists r'[(w_k, r', w_i) \in A]$。
2. **右弧**（RA_r）：$(\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma, w_i|\beta, A \cup \{(w_i, r, w_j)\})$，$w_i$ 是堆栈 σ 最上面的单词，w_j 是缓冲器 β 中最前面的单词，将 w_i 从堆栈中弹出，将缓冲器中最前面的单词 w_j 替换为 w_i，增加从 w_i 到 w_j 关系类型为 r 的依存关系。前置条件是 $\neg\exists w_k \exists r'[(w_k, r', w_j) \in A]$。
3. **移进**（SH）：$(\sigma, w_i|\beta, A) \Rightarrow (\sigma|w_i, \beta, A)$，从缓冲器中移除最前面的单词 w_i，并压入堆栈中。

针对一个句子的转移动作序列（transition sequence）可以对应状态序列 $C_{0,m} = (c_0, c_1, \ldots, c_m)$。$c_0$ 表示起始状态，$c_0(S)$ 表示针对句子 S 的起始状态，c_m 是终止状态。转换动作为 $t \in \mathcal{T}$ 使得状态 c_{i-1} 转换到状态 c_i 的，使用 $c_i = t(c_{i-1})$ 表示。针对句子“她非常喜欢跳芭蕾”的依存句法分析转移动作序列如表4.1所示。

假设存在函数 o，可以根据当前的状态 c 正确的确定下一步的转移动作 t，即 $o(c) = t$。那么整个句法分析的过程就可以使用非常简单的算法完成，伪代码如算法9所示。针对输入句子 S，首先，构造初始状态 $c_0(S)$，调用函数 o 得到下一步转移动作 $t = o(c)$。之后，利用根据转移动作得到下一个状态 $c = t(c)$。如此循环直到达到终止状态形式 $(\sigma, []_\beta, A)$。

转移动作预测函数可以使用分类器进行建模。构造一个分类器 $f(c)$，输入

表 4.1: 依存句法分析转移动作序列样例

	σ	β	A
	([ROOT],	[她, ...],	∅)
SH⇒	([ROOT, 她],	[非常, ...],	∅)
SH⇒	([ROOT, 她, 非常],	[喜欢, ...],	∅)
LA_{advmod} ⇒	([ROOT, 她],	[喜欢, ...],	A_1=(喜欢, advmod, 非常)
LA_{nsubj} ⇒	([ROOT],	[喜欢, ...],	A_2=A_1∪(喜欢, nsubj, 她)
SH⇒	([ROOT, 喜欢],	[跳, ...],	A_2)
SH⇒	([ROOT, 喜欢，跳],	[芭蕾舞],	A_2)
RA_{dobj} ⇒	([ROOT，喜欢],	[跳],	A_3=A_2∪(跳, dobj, 芭蕾舞)
RA_{xcomp} ⇒	([ROOT],	[喜欢],	A_4=A_3∪(喜欢, xcomp, 跳)
RA_{root} ⇒	([],	[ROOT],	A_5=A_4∪(ROOT, root, 喜欢)
SH⇒	([ROOT],	[],	A_5)

Algorithm 9: 基于转移动作函数 o 的依存句法分析算法

```
c = c_0(S);
while c 不是终止状态形式 (σ, []_β, A) do
    t = o(c);
    c = t(c);
return G_c
```

为状态 c，输出为其所对应的标准转移动作 $o(c)$。由此，基于转移的依存句法分析问题转化为了典型的机器学习问题。如果采用有监督机器学习算法，那么需要解决如下三个基本问题：1）如何表示状态 c；2）如何构造训练语料；3）如何选择和训练分类器。

针对状态 c 的表示问题，可以采用从堆栈 σ、缓存器 β 以及边集合 A 中对特定位置的单词、单词词性、树结构等抽取信息构造特征向量的方法。Maltparser[43] 针对标准弧转移系统利用如表4.2所示的信息构造针对状态 c 的特征向量。$\beta[0]$ 表示缓冲器中最前面的单词，$\sigma[0]$ 表示堆栈最顶端的单词，$ld(x)$ 表示 x 最左面的修饰词，$rd(x)$ 表示 x 最右面的修饰词。利用这些特征模板，可以构造状态的高维向量表示。使用 $\mathbf{f}(c)$ 表示特征函数，其输出是状态 c 的特征向量。

	单词	词元	粗粒度词性	细粒度词性	词形特征	依存关系类型
$\beta[0]$	×	×	×	×	×	
$\beta[1]$	×			×		
$\beta[2]$				×		
$\beta[3]$				×		
$ld(\beta[0])$						×
$rd(\beta[0])$						×
$\sigma[0]$	×	×	×	×	×	
$\sigma[1]$				×		
$ld(\sigma[0])$						×
$rd(\sigma[0])$						×

表 4.2: Maltparser[43] 针对标准弧转移系统采用的特征向量

依存句法树库 $\mathcal{D} = \{(S_d, G_d)_{d=0}^{|D|}\}$ 是由大规模的句子 S_d 以及所对应的正确的依存句法树 G_d 组成。但是用于转移动作预测的分类器所需的训练数据集 $\mathcal{D}'$ 需要由状态的特征表示 $\mathbf{f}(c)$ 和对应的正确的转移动作 t 组成, $\mathcal{D}' = \{(\mathbf{f}(c_d), t_d)_{d=0}^{|D'|}\}$。因此，需要将原始依存句法树库 $\mathcal{D}$ 中每个句子和对应依存句法树 (S_d, G_d) 转换为转移序列 $C_{0,m}^d = (c_0^d, c_1^d, \ldots, c_m^d)$。其中每个非终结状态 $c_i^d \in C_{o,m}^d$，都可以根据如下方式得到其对应的转移动作 $t_i^d = o(c_i^d)$:

$$o(c = (\sigma, \beta, A)) = \begin{cases} LA_r & \text{如果}(\beta[0], r, \sigma[0]) \in A_d \\ RA_r & \text{如果}(\sigma[0], r, \beta[0]) \in A_d \text{ 并且} \\ & \text{如果}(\beta[0], r', w) \in A_d \text{那么}(\beta[0], r', w) \in A \\ SH & \text{其他情况} \end{cases} \tag{4.22}$$

基于上述公式，可以根据非终结状态 c_i^d 和对应的转移动作 t_i^d，添加实例 $(\mathbf{f}(c_i^d), t_i^d)$ 到训练数据集合 $\mathcal{D}'$ 中。

利用训练数据集合 $\mathcal{D}' = \{(\mathbf{f}(c_d), t_d)_{d=0}^{|D'|}\}$ 可以选用各类型的分类器，已经成功应用于该任务的分类器包括：SVM[44]、基于记忆的学习 [45]、最大熵 [46] 等。相关的有监督分类算法在机器学习相关书籍和本书其他章节中也多有介绍，这里就不再赘述。

基于神经网络的转移依存句法分析

基于转移的依存句法分析通过定义转移系统，将依存句法分析转换为了根据当前状态 $c = (\sigma, \beta, A)$ 预测转移动作 t 的分类问题。上节中介绍的传统方法依赖特征工程，需要人工构建特征模板，构建状态的特征向量表示，在此基础上利用分类模型构建转移动作预测模型。神经网络方法不需要人工设计特征函数，并且可以在一定程度上缓解人工设计特征所带来的泛华能力不足的问题。在本节中，将介绍三种基于神经网络转移依存句法分析算法。

Chen 和 Manning（2014）基于 MaltPraser 的基本思想，将当前状态 c 中的堆栈 σ 和缓冲器 β 利用神经网络提取特征并预测下一步的转移动作[47] 。其神经网络结构如图4.21所示，包含输入层、隐藏层和输出层等三层。输入层是由单词嵌入、词性嵌入以及边类别嵌入根据上下文分别连接起来构成 x^w, x^t, x^l。x^w 是由堆栈最顶端的 3 个单词和缓冲器中最前面的 3 个单词：$\sigma[0], \sigma[1], \sigma[2], \beta[0], \beta[1], \beta[2]$，以及堆栈中最顶端的 2 个单词的最左、次左、最右和次右的儿子节点：$lc_1(\sigma[i]), lc_2(\sigma[i]), rc_1(\sigma[i]), rc_2(\sigma[i]), i = 1, 2$，再加上堆栈中最顶端的 2 个单词的最左的孙子节点和最右的孙子节点：$lc_1(lc_1(\sigma[i])), rc_1(rc_1(\sigma[i])), i = 1, 2$，共 18 个单词嵌入连接组成。$x^t$ 是由组成 x_w 的 18 个单词的词性嵌入组成。x^l 是 x^w 中除了堆栈和缓冲器的 6 个单词之外的其他 12 个单词对应的边标签嵌入连接组成。隐藏层为了有效的融合单词、词性以及边类别信息，采用立方激活函数结合线性变换：

$$h = \left(W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1\right)^3 \tag{4.23}$$

输出层则是采用标准的 softmax 函数决定转移动作。

[47]: Chen et al. (2014), “A fast and accurate dependency parser using neural networks”

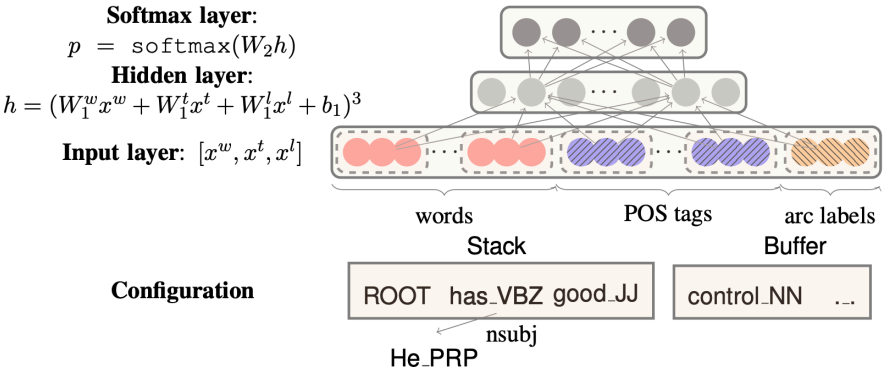


图 4.21: Chen 和 Manning 提出的基于神经网络的依存句法分析算法神经网络结构图

4.4 句法分析语料库

4.5 延伸阅读

4.6 习题

1. 如何判断一个语法理论属于成分语法还是依存语法?
2. 试举例说明什么是句法范畴以及句法范畴之间的层级关系。
3. 除了本章中介绍的标准弧转移系统，还是什么转移系统可以应用于依存句法分析？试说明该种转移系统的优缺点。
4.
5. 通常情况下对中文句子进行句法分析时，需要首先进行分词，并在基础上对词语进行词性标注，然后进行句法分析。这种流水线方法有什么优点和缺点？如何设计一种方法可以同时进行中文分词、词性标注和句法分析？
6. 试比较几种开源句法分析器在常见数据集合上的性能。

语义分析 5

语言模型介绍

5.1 语义学

基本概念

一阶谓词

语义网

槽填充

词汇语义表示

CBOW 和 Skipgram

5.2 语义分析概述

语义分析基本概念

语义分析历史沿革

语义分析主要任务

5.3 词义消歧

任务概述

基于词典的消歧方法

基于最大熵的消歧方法

基于深度神经网络的的消歧方法

词义消歧语料库

5.4 语义角色标注

任务概述

基于句法树的语义角色标注算法

基于深度神经网络的语义角色标注算法

语义角色标注语料库

5.5 延伸阅读

5.6 习题

信息抽取 6

语言模型介绍

6.1 信息抽取概述

信息抽取基本概念

信息抽取历史沿革

信息抽取主要任务

6.2 命名实体识别

基本概念

一阶谓词

6.3 关系抽取

任务概述

基于词典的消歧方法

6.4 事件抽取

任务概述

6.5 信息抽取析语料库

6.6 延伸阅读

6.7 习题

篇章分析 7

语言模型介绍

7.1 篇章分析概述

篇章分析基本概念

篇章分析历史沿革

篇章分析主要任务

7.2 篇章语言理论

7.3 篇章关系分析

论辩挖掘

?? 任务概述

篇章关系语料库

7.4 话题分割

话题分割语料库

7.5 共指消解

共指消解语料库

7.6 延伸阅读

7.7 习题

机器翻译 8

机器翻译介绍

8.1 机器翻译概述

机器翻译基本概念

机器翻译历史沿革

机器翻译主要任务

8.2 规则机器翻译

8.3 统计机器翻译

统计机器翻译基本概念

基于词的模型

基于词的机器翻译基础概念，基本假设和流程

基于词的机器翻译基本问题

IBM 模型 1

IBM 模型 2

IBM 模型 3-5

基于短语的模型

基于短语的机器翻译基础概念，基本假设和流程

短语抽取与对齐

短语翻译调序

基于距离基于方向

8.4 神经机器翻译

基于神经网络的机器翻译概述

循环神经网络翻译模型

卷积神经网络翻译模型

自注意力神经网络翻译模型

8.5 机器翻译评测

人工评价

自动评价

有参考答案的自动评价方法无参考答案的自动评价方法

8.6 机器翻译语料库

8.7 延伸阅读

8.8 习题

情感分析 9

情感分析的概述

9.1 情感分析概述

情感分析基本概念

什么是情感倾向分析

观点、情绪、心情等

观点的不同类型

情感分析历史沿革

情感分析主要任务

粒度

9.2 文档级情感分析

文档级情感分析（document-level sentiment analysis）的目标是将一篇给定观点的文档（如产品评论）根据所持有观点为正面或负面进行分类。正面或者负面观点又称为情感的倾向性或极性。这个任务即文档级别的分析，将一篇文档看成一个整体，并不研究文档中具体的实体或者属性，也不研究对这些实体的情感倾向。

[48] [49]

基于有监督方法的文档级情感分类

基于传统机器学习的文档级情感分析方法

SVM and ANN [50]

Document-level sentiment classification using hybrid machine learning approach [51]

基于深度学习的文档级情感分析方法

情感特定的文本表示用于文档级情感分析 [52]

LSTM [53]

CNN-LSTM model for document level sentiment analysis [54]

gated recurrent neural network [55]

RST discourse parsing [56]

层级结构 [57]

多层级文档级情感分析 [58]

从句子到文档 [59]

多层注意力模型 [60] 用于文档级别情感分析。

结合用户和产品的特征 [61, 62]

Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis [63]

基于无监督方法的文档级情感分类

文档级情感分析语料库

在情感分类的研究中，有几个数据源用于文档级别研究，但在文档级别使用最多的是评论。

Movie reviews [56]

9.3 句子级情感分析

基本概念、任务定义

[64]

[65]

基于传统机器学习的句子级情感分析

基于深度学习的句子级情感分析

[66] [67]

句子级情感分析语料库

9.4 属性级情感分析

基本概念、任务定义

[68]

Issues and challenges of aspect-based sentiment analysis: a comprehensive survey [69]

[70]

[71]

[72]

[73]

属性与实体抽取

属性级情感分类

属性抽取与情感分类联合模型

属性级情感分析语料库

使用最多的属性级情感分析数据集是语义评测国际研讨会发布的 [74–76]，包括 SemEval 2014 [74]，SemEval 2015 [75] 和 SemEval 2016 [76]. 同时，包括 Twitter [77]，Sentihood [78] 和 Mitchell [79] 也被用于该任务。

SemEval 2014

SemEval 2014 task4* [74] 关注的是基于属性级别情感分析，该任务的目标是检测给定目标实体的方面并确定每个方面所表达的情感极性。有两个针对笔记本电脑和餐馆的特定领域的数据集，即 Restaurants14 和 Laptop14，由 6000 多个句子组成，这些句子有属性级别的人类标注用于训练。每个句子都被归入句子中讨论的以下五个方面的一个或多个类别：(1) 食物；(2) 服务；(3) 价格；(4) 氛围（指餐厅的气氛和环境的句子）；(5) 轶事/杂事（不属于上述四个类别的句子）。具体来说，每个单字或多字的属性词都根据句子中对它所表达的情感而被赋予以下极性之一：(1) 积极；(2) 消极；(3) 中性（指既非积极也非消极的情绪）；(4) 冲突（意味着既是积极又是消极的情绪）。

* http://alt.qcri.org/semeval2014/task4/

Restaurants14

Restaurants14 由从 Ganu 等人 [80] 从餐厅评论/评论中提取的 3000 多条英文句子组成，作为训练数据集。额外评论以相同的方式进行标注作为测试数据集。在去除有冲突的情绪极性或没有属性词的数据后，剩下 1,978 个训练样本和 600 个测试样本。该数据集包括对粗略的属性类别、属性词、属性词特定极性和属性类别特定极性的标注。

Laptop14

这个数据集由超过 3000 个从客户笔记本评论中获得的英文句子组成。该数据集的一部分被划分为测试数据。在除去有冲突的情感极性或没有属性词的数据后，剩下 1462 个训练样本和 411 个测试样本。该数据集只包括对句子的属性词及其极性的标注。

SemEval 2015

SemEval-2015 任务 12[†] [75] 是 SemEval-2015 task4 的延续。这项任务的目标是确定所有的属性和它们的整体极性。特别是，SemEval2015 任务 12 的输入数据集包含整个评论而不是孤立的句子。对于训练，提供了两个数据集，其中包括大约 500 条餐馆和笔记本电脑的评论，并标注了各个属性及其极性。对于测试数据集，提供了额外的数据集。

Restaurants15

该数据集由 254 条和 96 条餐厅评论组成，分别为训练和测试的属性及其情感极性做了标注。每个评论可能包含多个句子，每个句子包括类别、属性词和

[†] http://alt.qcri.org/semeval2015/task12/

属性极性的标注。在去除冲突情绪极性的数据后，有 1,120 句用于训练，582 句用于测试。

SemEval 2016

SemEval-2016 task5[‡] [76] 与 SemEval-2015 任务 12 类似，该数据集由整个评论组成。此外，该数据集包含五个领域，涵盖八种语言。

Restaurants16

该数据集由 350 条餐厅评论组成，其中有用于训练的属性词、属性类别和极性的标注，有 92 条用于测试。在去除有冲突情绪极性的数据后，有 1,708 个标注的句子用于训练，587 个用于测试。

Twitter

Dong Li 等人 [77] 介绍了一个手动标注的数据集，用于目标依赖的 twitter 情感分析。这是最大的目标依赖的 twitter 情感分类数据集，它是由人工标注的。训练数据有 6,248 条推文，测试数据包括 692 条推文，其情感类别平衡为 25% 负面，50% 中性，25% 正面。

[‡] http://alt.qcri.org/semeval2016/task5/

Others

Mitchell

Mitchell 数据集[§] [79] 由大约 30,000 条西班牙推文和 10,000 条英文推文组成，以 Begin, Inside, Outside（BIO）的编码方式标注了命名实体（NE），NE 的开始部分用 B-NE 标注，NE 的其余部分用 I-NE 标注。7,105 条西班牙推文包含 9,870 个名称实体，2,350 条英文推文在删除转发后包含 3,577 个名称实体。为了获得情感标签（正面、负面或无情感），通过亚马逊的 Mechanical Turk 进行了众包。为了进行 10 倍交叉验证，英语数据被分成若干折。

Sentihood

SentiHood [78] 是一个基准数据集，它被标注为城市街区领域中属性级目标情感分析任务。它是基于与伦敦市街区有关的问题，这些问题是通过过滤雅虎答案的问题回答平台的文本而获得的。SentiHood 由 5215 个句子组成，其中 3862 个句子包含一个地点，1353 个句子包含两个地点。在整个数据集中，位置实体名称被 location1 和 location2 所掩盖，所以这项任务不涉及命名实体识别。

MPQA

MPQA[¶] [81] 包含了标注了意见和其他状态（如情感、信念、情绪和猜测）的新闻文章和其他文本文件。在 MPQA 3.0 中，增加了实体-目标和事件-目标（eTarget）标注。请注意，MPQA 2.0 中以前的基于片段的目标标注在这个新的语料库中被保留，它们被重新命名为 sTarget（基于片段的目标）。特别是，目前的数据集包含 70 个文档，其中包括 1,029 个表达性主观元素（ESE），1,287 个态

[§] http://www.m-mitchell.com/code/index.html
[¶] http://mpqa.cs.pitt.edu/corpora/

度，以及 MPQA 2.0 中 1,213 个态度的目标片段。此外，1,366 个 eTargets 被添加到 ESEs 中，1,608 个 eTargets 被添加到目标跨度中。

9.5 延伸阅读

9.6 习题

智能问答 10

智能问答概述

10.1 智能问答概述

智能问答基本概念

智能问答历史沿革

智能问答主要任务

10.2 阅读理解

基本概念、任务定义

基于特征的阅读理解

基于深度神经网络的阅读理解

阅读理解语料库

10.3 表格问答

基本概念、任务定义

基于特征的表格问答

基于深度神经网络的表格问答

表格问答语料库

10.4 社区问答

基本概念、任务定义

基于特征的问题问题匹配

基于深度神经网络的问题匹配

社区问答语料库

10.5 开放问答

基本概念、任务定义

开放问答语料库

10.6 延伸阅读

10.7 习题

文本摘要 11

文本摘要介绍

11.1 文本摘要概述

文本摘要基本概念

文本摘要历史沿革

文本摘要主要任务

11.2 抽取式摘要

抽取式摘要

传统方法

基于序列标注的方法

基于排序的方法

11.3 生成式摘要

生成式摘要

序列到序列

复制和覆盖机制

全局优化与强化学习

抽取与生成结合

11.4 文本摘要应用

应用

单文档摘要

多文档摘要

对话摘要

跨语言文本摘要

多模态文本摘要

11.5 文本摘要的评测

相对于其他自然语言处理任务，例如文本分类、序列标注等，文本摘要的评测是困难的，因为显著信息的选取和文摘的表述没有统一的标准答案。与机器翻译、对话系统等面向内容产生（content generation）的任务一样，文本摘要也有多维度的评价系统，包括人工评测方案和自动评测指标。

文本摘要的评测按照与任务的相关性可以分为两类，内在（Intrinsic）评价方法和外在（Extrinsic）评价方法。内在评价方法与摘要任务相关，它通过直接分析摘要的质量来评价摘要系统，一般从总体和细分维度两个粒度进行评估。其中，总体评估是指从总体上评价摘要的质量，并给出一个综合分数。细分维度一般可以从“与参考标准的一致性”和“类人性”两个角度来考虑，多方面地对摘要的质量进行评价。外在评价方法则是一种间接的评价方法，与系统的功能对应，将摘要应用于下游任务或者实际应用系统中，根据摘要在这些任务和系统中产生的实际效果来间接评估摘要的质量。这里我们仅讨论内部评价方法。

一般而言，在评测文本摘要的质量时，我们会关注以下 5 个细分维度。(1) 信息量（Informativeness），即摘要的内容含量，它可以衡量一段文本所提供的新信息的程度。Peyrard [82] 对信息量作出了直观上的定义：阅读者一般都拥有

[82]: Peyrard (2019), “A Simple Theoretical Model of Importance for Summarization”

背景知识和常识，如果一段摘要能使其获得新信息或者产生认知上的变化，则可认为此摘要具有丰富的信息量。(2) 非冗余性（Non-redundancy），它体现了摘要精简的特性，即不能反复使用重复或相似的文本描述同一个关键点。(3) 流畅度（Fluency），它包含了摘要结构的连贯性和语法的正确性两个方面。(4) 忠实度（Faithfulness），也称为相关性（Relevance），即摘要内容是否忠于原文。作为原文的子集，摘要不能包含凭空产生的信息，也不能包含与原文和事实相悖的内容。(5) 聚焦程度（Focus），也称为显著性（Saliency），它衡量了摘要包含关键信息的程度。除了主要信息，原文档往往具有大量的细节描述和补充内容等次要信息。通过聚焦程度，我们可以评估一个摘要系统的甄别并捕捉原文主要信息的能力。

以上多数评测维度需要借助人工方式进行。评测者一般会根据预定义好的评测指南和范例对摘要质量进行评估。人工评测能灵活应用于多种场景和标准，适合所有的主观性任务。因此，在目前的文本摘要任务中，人工评测被认为是评价模型优劣的黄金标准。但是，人工评测也存在成本高昂和结果难以复现等问题。相比人工评测，自动评测具有方便快捷、容易复现等优势，因此被广泛应用于摘要评价。尤其是在模型开发的早期阶段，自动评测能帮助开发者快速定位问题。在大多数情况下，自动摘要的评价可以将人工评测方案与自动评测指标相结合，对模型在测试集上的整体效果做自动评测，并随机采样部分测试点进行人工评测，从而兼顾评测的效率和质量。

人工评测

人工评测按照执行方式一般分为两类，逐点评估（Point-wise）和逐对评估（Pair-wise）。在逐点评估中，评估者对系统产生的每一个结果按照预定义的维度进行评估打分。一个常见的方案是李克特五点量表（Likert Scale）。给定原始文档和模型输出的摘要，评估者会按照 1～5 分对摘要某一方面的评测维度进行打分。假设评测维度是流畅度，则 1～5 的分值依次对应：非常不流畅、不流畅、一般、流畅、非常流畅五种程度。但是，逐点评估具有很强的主观性，导致评

估者之间的偏差较大，一致性很低。在逐对评估中，给定相同的原始文档和两个不同系统的输出摘要 A 和 B，评估者需要判断 A 与 B 相比哪个更好。与逐点评估的多选项打分相比，逐对评估采用两两比较的方式，降低了评估难度，可以提高评测结果的一致性。但是，如果存在多个需要评测的摘要系统，总评估次数会随着系统数目的增加而呈指数上升，成本较高。

人工评测存在主观性，包括摘要任务本身的主观性和评估者自身的主观性。为了尽可能消除人工评测的主观性偏差，一般会让多个评估者对同一条数据进行独立重复打分。因此，衡量多个评估者之间的评测一致性是一个重要的过程。一致性不仅可以体现人工评测质量的高低，还能反映评测任务的难易程度。Fleiss 卡帕系数（Fleiss's κ）是度量多个系统一致性的方法，也经常被用于计算人工评测的一致性。

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij}-1) = \frac{1}{n(n-1)}(\sum_{j=1}^{k} n_{ij}^2 - n) \tag{11.1}$$

$$P_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij} \tag{11.2}$$

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i = \frac{1}{Nn(n-1)}(\sum_{i=1}^{N} \sum_{j=1}^{k} n_{ij}^2 - Nn) \tag{11.3}$$

$$\bar{P}_e = \sum_{j=1}^{k} P_j^2 \tag{11.4}$$

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{11.5}$$

TODO：人工评测的不足

自动评测

ROUGE

Embedding-based Method，BERTScore

自动评测的不足

11.6 延伸阅读

11.7 习题

知识图谱 12

知识图谱的概述

12.1 知识图谱概述

知识图谱基本概念

知识图谱历史沿革

知识图谱主要任务

12.2 知识表示学习

基本概念、任务定义

知识度量方法

知识表示编码模型

编码额外信息方法

12.3 知识图谱构建

基本概念、任务定义说明实体识别、关系抽取、事件抽取在别的章节详细介绍

实体消歧

实体链接

关系补全

12.4 知识图谱问答

基本概念、任务定义

基于语义解析的知识图谱问答

基于搜索排序的知识图谱问答

基于深度神经网络的知识图谱问答

12.5 知识图谱存储

基于表的知识谱图谱存储

基于图的知识谱图谱存储

知识图谱检索

12.6 延伸阅读

12.7 习题

模型鲁棒性 13

模型鲁棒性概述

13.1 鲁棒性概述

鲁棒性基本概念

鲁棒性发展沿革

鲁棒性研究内容

13.2 文本攻击方法

字符级别攻击方法

词语级别攻击方法

句子级别攻击方法

投毒攻击

13.3 文本防御方法

基于对抗训练的文本防御方法

基于正则化的文本防御方法

基于数据增强的文本防御方法

对抗样本检测

13.4 模型鲁棒性评价基准

模型通用评价基准

模型可解释性 14

知识图谱的概述

14.1 模型可解释性概述

基本概念、任务定义

模型可解释性基本概念

模型可解释性历史沿革

模型可解释性主要任务

14.2 模型结构分析

基本概念、任务定义

探针任务

错误类型分析

可解释评估

14.3 模型行为分析

基本概念、任务定义

预测行为分析

14.4 具体任务中的可解释性

基本概念、任务定义

对话系统

智能问答

情感分析

自动摘要

14.5 延伸阅读

14.6 习题

参考文献

参考文献按照引用先后排序.

[1] Victoria Fromkin, Robert Rodman, and Nina Hyams. *An introduction to language*. Cengage Learning, 2018 (引用页：5, 48).

[2] W Nelson Francis. "A tagged corpus–problems and prospects". In: *Studies in English linguistics for Randolph Quirk* (1980), pp. 192–209 (引用页：13).

[3] Dan Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to speechrecognition, natural language processing and computational linguistics*. 2nd ed. Pearson, Apr. 2008 (引用页：14).

[4] 俞士汶 et al. *北大语料库加工规范：切分·词性标注·注音*. 北京大学计算语言学研究所, 2003 (引用页：20).

[5] 黄昌宁, 李玉梅, and 朱晓丹. *中文文本标注规范 (5.0 版)*. 微软亚洲研究院, 2006 (引用页：20).

[6] 梁南元. "书面汉语自动分词系统—CDWS". In: *中文信息学报* 1.2, 46 (1987), p. 46 (引用页：22).

[7] 黄昌宁 and 赵海. "中文分词十年回顾 ". In: *中文信息学报* 21.3 (2007), pp. 8–19 (引用页：23).

[8] 宗成庆. *统计自然语言处理*. 清华大学出版社, Aug. 2013 (引用页：23).

[9] Yue Zhang and Stephen Clark. "Chinese segmentation with a word-based perceptron algorithm". In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 840–847 (引用页：27).

[10] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780 (引用页：29).

[11] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: *Neural computation* 12.10 (2000), pp. 2451–2471 (引用页：29).

[12] 邱锡鹏. *神经网络与深度学习*. 北京: 机械工业出版社, 2020 (引用页：30).

[13] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286 (引用页：32).

[14] Ronan Collobert et al. "Natural language processing (almost) from scratch". In: *Journal of machine learning research* 12.ARTICLE (2011), pp. 2493–2537 (引用页：32, 40, 43).

[15] 吴立德. *大规模中文文本处理*. 复旦大学出版社, 1997 (引用页：35).

[16] 张虎, 郑家恒, and 刘江. "语料库词性标注一致性检查方法研究 ". In: *中文信息学报* 18.5 (2004), pp. 12–17 (引用页：35).

[17] Eric Brill. "A Simple Rule-Based Part of Speech Tagger". In: *Proceedings of the Third Conference on Applied Natural Language Processing*. ANLC '92. Trento, Italy: Association for Computational Linguistics, 1992, pp. 152–155. DOI: 10.3115/974499.974526 (引用页：36).

[18] Lalit Bahl et al. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 11. IEEE. 1986, pp. 49–52 (引用页：38).

[19] 李航. *统计学习方法（第二版）*. 清华大学出版社, May 2019 (引用页：39).

[20] Slav Petrov, Dipanjan Das, and Ryan McDonald. "A universal part-of-speech tagset". In: *arXiv preprint arXiv:1104.2086* (2011) (引用页：45).

[21] Meishan Zhang, Yue Zhang, and Guohong Fu. "Transition-Based Neural Word Segmentation". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 421–431. DOI: 10.18653/v1/P16-1040 (引用页：46).

[22] Jie Yang, Yue Zhang, and Shuailong Liang. "Subword Encoding in Lattice LSTM for Chinese Word Segmentation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2720–2725 (引用页：46).

[23] Emma Strubell et al. "Fast and Accurate Entity Recognition with Iterated Dilated Convolutions". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 2670–2680 (引用页：46).

[24] Yue Zhang, Qi Liu, and Linfeng Song. "Sentence-State LSTM for Text Representation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 317–327 (引用页：46).

[25] 维多利亚·弗罗姆金 [著] et al. *语言引论 (第八版)*. 北京大学出版社, 2017 (引用页：48).

[26] Noam Chomsky. *Syntactic structures*. De Gruyter Mouton, 2009 (引用页：49).

[27] Tesnière Lucien. "Eléments de syntaxe structurale". In: *Paris, Klincksieck* (1959), p. 25 (引用页：52).

[28] John Carroll, Ted Briscoe, and Antonio Sanfilippo. "Parser evaluation: a survey and a new proposal". In: *Proceedings of the 1st International Conference on Language Resources and Evaluation*. Vol. 32. Granada. 1998 (引用页：53).

[29] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. "Generating typed dependency parses from phrase structure parses." In: *Lrec*. Vol. 6. 2006, pp. 449–454 (引用页：53).

[30] John Cocke. *Programming languages and their compilers: Preliminary notes*. New York University, 1969 (引用页：56).

[31] Daniel H Younger. "Recognition and parsing of context-free languages in time n3". In: *Information and control* 10.2 (1967), pp. 189–208 (引用页：56).

[32] Tadao Kasami. "An efficient recognition and syntax-analysis algorithm for context-free languages". In: *Coordinated Science Laboratory Report no. R-257* (1966) (引用页：56).

[33] Jeffrey D Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall, 1972 (引用页：60).

[34] Karim Lari and Steve J Young. "The estimation of stochastic context-free grammars using the inside-outside algorithm". In: *Computer speech & language* 4.1 (1990), pp. 35–56 (引用页：73).

[35] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999 (引用页：73).

[36] Sandra Kubler, Ryan McDonald, and Joakim Nivre. *Dependency Parsing*. Morgan & Claypool Publishers, 2009 (引用页：75).

[37] Yoeng-Jin Chu. "On the shortest arborescence of a directed graph". In: *Scientia Sinica* 14 (1965), pp. 1396–1400 (引用页：75).

[38] Jack Edmonds. "Optimum branchings". In: *Journal of Research of the National Bureau of Standards, B* 71 (1967), pp. 233–240 (引用页：75).

[39] Ryan McDonald et al. "Non-projective dependency parsing using spanning tree algorithms". In: *Proceedings of human language technology conference and conference on empirical methods in natural language processing*. 2005, pp. 523–530 (引用页：80).

[40] Wenzhe Pei, Tao Ge, and Baobao Chang. "An effective neural network model for graph-based dependency parsing". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 313–322 (引用页: 81).

[41] Timothy Dozat and Christopher D. Manning. "Deep Biaffine Attention for Neural Dependency Parsing". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (引用页: 82).

[42] Tao Ji, Yuanbin Wu, and Man Lan. "Graph-based dependency parsing with graph neural networks". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2475–2485 (引用页: 83).

[43] Joakim Nivre. "Algorithms for deterministic incremental dependency parsing". In: *Computational Linguistics* 34.4 (2008), pp. 513–553 (引用页: 84–86).

[44] Joakim Nivre, Johan Hall, and Jens Nilsson. "Maltparser: A data-driven parser-generator for dependency parsing." In: *LREC*. Vol. 6. 2006, pp. 2216–2219 (引用页: 86).

[45] Joakim Nivre, Johan Hall, and Jens Nilsson. "Memory-based dependency parsing". In: *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*. 2004, pp. 49–56 (引用页: 86).

[46] Wanxiang Che et al. "A cascaded syntactic and semantic dependency parsing system". In: *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*. 2008, pp. 238–242 (引用页: 86).

[47] Danqi Chen and Christopher D Manning. "A fast and accurate dependency parser using neural networks". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 740–750 (引用页: 87).

[48] Salima Behdenna, Fatiha Barigou, and Ghalem Belalem. “Document level sentiment analysis: a survey”. In: *EAI Endorsed Transactions on Context-aware Systems and Applications* 4.13 (2018) (引用页：100).

[49] Salima Behdenna, Fatiha Barigou, and Ghalem Belalem. “Sentiment analysis at document level”. In: *International Conference on Smart Trends for Information Technology and Computer Communications*. Springer. 2016, pp. 159–168 (引用页：100).

[50] Rodrigo Moraes, João Francisco Valiati, and Wilson P GaviãO Neto. “Document-level sentiment classification: An empirical comparison between SVM and ANN”. In: *Expert Systems with Applications* 40.2 (2013), pp. 621–633 (引用页：100).

[51] Abinash Tripathy, Abhishek Anand, and Santanu Kumar Rath. “Document-level sentiment classification using hybrid machine learning approach”. In: *Knowledge and Information Systems* 53.3 (2017), pp. 805–831 (引用页：100).

[52] Duyu Tang. “Sentiment-specific representation learning for document-level sentiment analysis”. In: *Proceedings of the eighth ACM international conference on web search and data mining*. 2015, pp. 447–452 (引用页：100).

[53] Jiacheng Xu et al. “Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1660–1669 (引用页：100).

[54] Maryem Rhanoui et al. “A CNN-BiLSTM model for document-level sentiment analysis”. In: *Machine Learning and Knowledge Extraction* 1.3 (2019), pp. 832–847 (引用页：100).

[55] Duyu Tang, Bing Qin, and Ting Liu. “Document modeling with gated recurrent neural network for sentiment classification”. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 1422–1432 (引用页：100).

[56] Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. “Better Document-level Sentiment Analysis from RST Discourse Parsing”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 2212–2218 (引用页：100, 101).

[57] Guozheng Rao et al. “LSTM with sentence representations for document-level sentiment classification”. In: *Neurocomputing* 308 (2018), pp. 49–57 (引用页：100).

[58] Ainur Yessenalina, Yisong Yue, and Claire Cardie. “Multi-level structured models for document-level sentiment classification”. In: *Proceedings of the 2010 conference on empirical methods in natural language processing*. 2010, pp. 1046–1056 (引用页：101).

[59] Changli Zhang et al. “Sentiment analysis of Chinese documents: From sentence to document level”. In: *Journal of the American Society for Information Science and Technology* 60.12 (2009), pp. 2474–2487 (引用页：101).

[60] Zichao Yang et al. “Hierarchical attention networks for document classification”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 1480–1489 (引用页：101).

[61] Duyu Tang, Bing Qin, and Ting Liu. “Learning semantic representations of users and products for document level sentiment classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 1014–1023 (引用页：101).

[62] Zi-Yi Dou. “Capturing user and product information for document level sentiment analysis with deep memory network”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 521–526 (引用页：101).

[63] Rui Xia et al. “Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis”. In: *Information Processing & Management* 52.1 (2016), pp. 36–45 (引用页：101).

[64] Vrushali K Bongirwar. “A survey on sentence level sentiment analysis”. In: *International Journal of Computer Science Trends and Technology (IJCST)* 3.3 (2015), pp. 110–113 (引用页：101).

[65] VS Jagtap and Karishma Pawar. “Analysis of different approaches to sentence-level sentiment classification”. In: *International Journal of Scientific Engineering and Technology* 2.3 (2013), pp. 164–170 (引用页：101).

[66] Arun Meena and TV Prabhakar. “Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis”. In: *European conference on information retrieval*. Springer. 2007, pp. 573–580 (引用页：102).

[67] Orestes Appel et al. “A hybrid approach to the sentiment analysis problem at the sentence level”. In: *Knowledge-Based Systems* 108 (2016), pp. 110–124 (引用页：102).

[68] Naveen Kumar Laskari and Suresh Kumar Sanampudi. “Aspect based sentiment analysis survey”. In: () (引用页：102).

[69] Ambreen Nazir et al. “Issues and challenges of aspect-based sentiment analysis: a comprehensive survey”. In: *IEEE Transactions on Affective Computing* (2020) (引用页：102).

[70] Haoyue Liu et al. “Aspect-based sentiment analysis: A survey of deep learning methods”. In: *IEEE Transactions on Computational Social Systems* 7.6 (2020), pp. 1358–1375 (引用页：102).

[71] Kim Schouten and Flavius Frasincar. “Survey on aspect-level sentiment analysis”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.3 (2015), pp. 813–830 (引用页：102).

[72] Hai Ha Do et al. “Deep learning for aspect-based sentiment analysis: a comparative review”. In: *Expert systems with applications* 118 (2019), pp. 272–299 (引用页：102).

[73] Kaustubh Yadav et al. “A comprehensive survey on aspect-based sentiment analysis”. In: *International Journal of Engineering Systems Modelling and Simulation* 12.4 (2021), pp. 279–290 (引用页：102).

[74] Suresh Manandhar. “SemEval-2014 Task 4: Aspect Based Sentiment Analysis”. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 2014 (引用页：103).

[75] Maria Pontiki et al. “Semeval-2015 Task 12: Aspect Based Sentiment Analysis”. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. 2015, pp. 486–495 (引用页：103, 104).

[76] Maria Pontiki et al. “SemEval-2016 Task 5: Aspect Based Sentiment Analysis”. In: *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*. 2016, pp. 19–30 (引用页：103, 105).

[77] Li Dong et al. “Adaptive Recursive Neural Network for Target-Dependent Twitter Sentiment classification”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2. 2014, pp. 49–54 (引用页：103, 105).

[78] Marzieh Saeidi et al. “SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods”. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 1546–1556 (引用页：103, 106).

[79] Margaret Mitchell et al. “Open Domain Targeted Sentiment”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1643–1654 (引用页：103, 106).

[80] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. "Beyond the Stars: Improving Rating Predictions using Review Text Content." In: *WebDB*. Vol. 9. Citeseer. 2009, pp. 1–6 (引用页：104).

[81] Lingjia Deng and Janyce Wiebe. "Mpqa 3.0: An entity/event-level sentiment corpus". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1323–1328 (引用页：106).

[82] Maxime Peyrard. "A Simple Theoretical Model of Importance for Summarization". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1059–1073 (引用页：113).