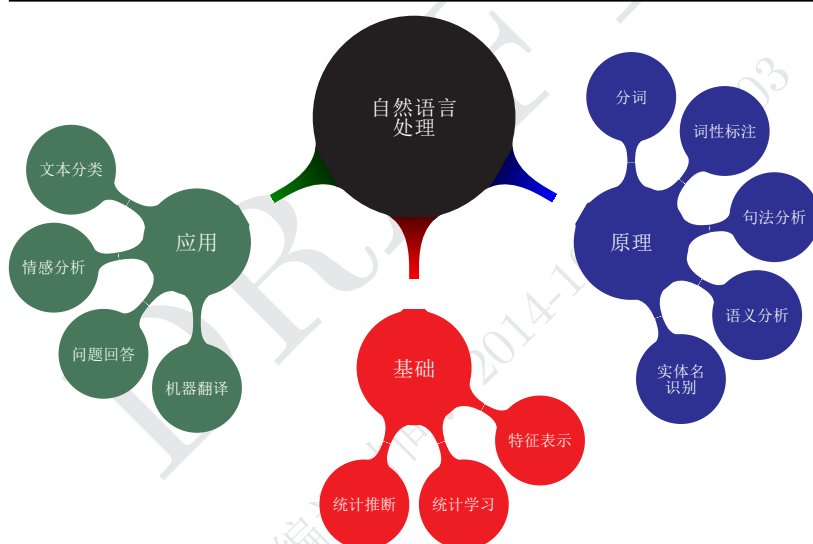


自然语言处理 原理与实现

Natural Language Processing Principles and Practice



邱锡鹏
xpqiu@fudan.edu.cn

2014 年 10 月 31 日

前言

自然语言通常指自然地随文化演化的语言，是人类交流和思维的主要工具，比如英语、汉语、日语等。区别于自然语言，人工语言是由人特意为某些特定目的而创造的语言，比如数学语言、世界语以及为计算机而设的程序语言。随着信息的爆炸式增长，人们越来越需要一些自动化技术来处理和分析自然语言，以减轻自己获取和利用信息的负担。比如在日常生活中人们越来越多地使用搜索引擎、拼音输入法、语音识别、辅助翻译等工具，这些工具的基本理论方法就是自然语言处理。我们很难准确地定义自然语言处理的具体研究内容，

自然语言处理是人工智能和语言学领域的分支学科，主要研究和探讨如何让计算机来处理及运用自然语言。与自然语言处理有紧密联系的领域有计算语言学。计算语言学是一门跨学科的研究领域，试图找出自然语言的规律，建立运算模型，最终让电脑能够像人类般分析，理解和处理自然语言。自然语言处理已经成为国际上最活跃的科学研究领域之一，引起了全球科技界和企业界的密切关注。

我们的大脑像个复杂的学习系统，我们可以很容易的理解一个句子，一篇文章，但是我们并不需要知道具体的词法以及语法规则。而且，每个人对词法、语法规则的理解也不尽相同。但这不影响我们通过语言进行交流。

本书详细阐述自然语言处理所涉及的相关原理、模型以及具体实现方法。主要内容包括基础介绍、词法分析、词性标注、句法分析、语义分析以及应用。

有很多优秀的书籍阐述了自然语言处理的理论和方法。但是当初学者开发实际系统时，还是觉得无从下手。本书在讲解原理的同时，通过大量的实践帮助初学者获得自然语言处理的实际经验。本书中的实例通过 Java 语言进行描述，但同样适合有其它面向对象程序语言（比如 C++，C#，Python、Ruby 等）知识的读者。在具体的实践实例中，部分例子使用了基于开源项目 FNLTP 中的代码和模块。

本书可以作为一到两个学期自然语言处理的教材。学生将学习到自然语言处理系统隐含的理论知识，以及将这些理论付诸实际系统中的需要考虑的程序设计技术。读者需要有一定的概率论基础。

习题。本书中每一章都有一些习题和项目实践。标有一个星号的习题难度一般，标有两个星号的习题有一定的难度。标有三个星号的习题是一些还处于研究阶段的问题。

FNLP 简介

FNLP 主要是为中文自然语言处理而开发的工具包，也包含为实现这些任务的机器学习算法和数据集。本工具包及其包含数据集使用 LGPL3.0 许可证。FNLP 是基于 Java 的开源项目，利用统计机器学习和规则方法来处理中文自然语言处理的经典问题，比如：分词、词性标注、句法分析、实体名识别等。

Java 基础

Java 是一种可以撰写跨平台应用程序的面向对象的程序设计语言。Java 编程语言的风格十分接近 C、C++ 语言，它继承了 C++ 语言面向对象技术的核心。Java 舍弃了 C 语言中容易引起错误的指针（以引用取代）、运算符重载（operator overloading）、多重继承（以接口取代）等特性，增加了垃圾回收器功能用于回收不再被引用的对象所占据的内存空间，使得程序员不用再为内存管理而担忧。为了更好的阅读本书，你有一定的 Java 基础。Java 语言支持四种类型：接口 (interface)、类 (class)、数组 (array) 和原语类型 (primitive)。

构造函数、成员。Java Development Kit (JDK) 帮助文档, <http://docs.oracle.com/javase/>。

为了提高可读性，书中的实例都省略了导言（除非必要）。为了正常运行这些例子，你可能需要在程序中加入 import 语句：

```
import java.util.*;
import java.io.*;
```

当然，本书中所有的代码示例都可以从 <http://www.fnlp.org/> 获得，你可以直接编译和运行这些例子。

Java IDE

当今最流行的是 EclipseJava 集成开发环境 IDE(Integrated Development Environment, 集成开发环境)

致谢

首先，特别感谢 FNL P 项目的主要研发人员，他们是计峰、高文君、赵嘉亿、曹零、赵建双、田乐、缪有栋、刘昭等。此外，我所在实验室的其他研究生和本科生也贡献了部分代码。在此一并感谢大家的工作。

对于本书，很多人提出了富有建设性的意见，或在其他方面提供了帮助，在此表示感谢。

特别声明，本书的部分内容和参考材料来源于互联网，如：维基百科、百度百科以及一些论坛、博客等。因此这些材料的原始出处已无法溯源，后面正文中不再一一引用，在此一并致谢。若有需要特别标注引用的，请联系作者更正。

邱锡鹏

第一章 绪论

一个人在不接触对方的情况下，通过一种特殊的方式，和对方进行一系列的问答。如果在相当长时间内，他无法根据这些问题判断对方是人还是计算机，那么就可以认为这个计算机是智能的。

— Alan Turing [1950], 《机器能思维吗?》

语言是什么呢？它是一套约定了的符号以及相应的规则系统，可以用来传递信息。这里的符号可以通过视觉、听觉或者触觉等方式来进行感知。但一般来讲，语言是指人类沟通所使用的自然语言。

自然语言是人们日常生活中使用的语言，比如汉语、英语等。自然语言是人们交流和思维的主要工具，是人类智慧的结晶。之所以把人类使用的语言称为“自然”语言，主要是区别于为计算机而设的“人工”语言，如高级程序语言（比如C语言、Java语言等）以及标记语言（比如XML、HTML等）。

高级程序语言的出现极大地促进了计算机的发展和普及。计算机上的各种软件，智能手机上的各种应用，甚至来操作系统本身，都是通过程序语言实现的。人们可以通过程序语言来高效地将自己的意图告诉计算机，让它来执行各种任务。很难想象，如果没有这些人工语言，程序员只能通过机器指令来编写程序，我们现在的和生活和工作方式是否还停留在上个世纪。

为了让计算机理解程序语言，我们需要设计一个**编译器**。编译器的作用把高级程序语言等价地翻译成汇编语言或机器指令。不同程序语言有不同的编译器。编译技术的发展也得益于自然语言的研究成果。在20世纪50年代，编译器的编写一直被认为是十分困难的事情，第一代Fortran的编译器据说花了18年的时间才完成。60年代初，Noam Chomsky通过对自然语言结构的研究，根据语言文法的难易程度以及识别它们所需要的算法来对语言分类，提出了四个层次的文法：0型文法、1型文法、2型文法和3型文法，且其中的每一个都是其前者的特殊情况。这些成果极大地促进了编译技术的发展。其中，

2型文法（或上下文无关文法）被证明是程序语言中最有用的，并且也成为程序语言的标准规范。

但是，要让计算机理解自然语言，设计一个自然语言的“编译器”，这个任务则要困难很多。最根本的区别在于程序语言是确定的、无歧义的，需要一套严格的语法规则，如果输入的文本不符合语法规则，编译器就不能正确解析。而自然语言普遍存在不确定性或歧义性。并且，自然语言一般都是需要一定的上下文才能理解的，已经超出了2型文法的范畴。

1.1 什么是自然语言处理？

自然语言处理（Natural Language Processing, NLP）是人工智能和语言学领域的分支学科，主要是研究如何让计算机处理及运用自然语言，实现人与计算机之间用自然语言进行有效通信的各种理论和方法。它是计算机科学领域与人工智能领域中的一个重要部分，甚至核心部分，也是人工智能中最为困难的问题之一。它与语言学的研究有着密切的联系，但又有重要的区别。自然语言处理并不是一般地研究自然语言，而在于研究能有效地实现让计算机分析和处理自然语言的技术，特别是利用计算机的能力来高效地处理大规模的文本。比如，文本分类是自然语言处理的一个主要应用，但是不太需要计算语言学的知识。

自然语言处理技术需要融合语言学、计算机科学、数学、心理学以及认知学等多学科的知识。严格来自，自然语言处理可分为两部分：

- **自然语言理解**（Natural Language Understanding, NLU）是指让计算机“懂”人类的语言，能理解自然语言文本的意义。
- **自然语言生成**（Natural Language Generation, NLG）是指把计算机数据转化为自然语言，以自然语言文本来表达给定的意图、思想等。

这两部分是互逆过程。自然语言理解是从人到计算机的信息传递，自然语言生成是从计算机到人的信息传递。但通常我们所说的自然语言理解也指自然语言处理，并不严格区分其内在的含义。

自然语言处理的主要对象就是语言，表现形式为语音或文本。这里的语言不但包括自然语言，也包括人工语言。例如，如果我们要分析一个网页，既需要处理网页中自然语言的内容，也需要能利用HTML语言来处理网页的页面结构信息。

1.2 自然语言处理的范畴

通过计算机的高效计算能力，分析和处理各种语言，提高人们的生活水平和工作效率，这是自然语言处理的目标。从这个角度看，很难去界定自然语言处理的范畴。只要符合这个目标并以语言为处理对象的技术都可以看成自然语言处理技术。自然语言处理的范畴是随着时代的发展而不断变化的。新事物会对自然语言处理提出新的需求，而以前的需求可能会随着技术的发展而不再存在。

对于目前的自然语言处理技术而言，我们可以从不同的层面来对各种不同的技术进行划分。

首先，从应用需求来看，主要技术有：

1. 文本分类和聚类
2. 信息检索和过滤
3. 信息抽取
4. 问答系统
5. 对话系统
6. 机器翻译
7. 情感分析
8. 文本挖掘

除了上述技术，还要很多涉及到自然语言处理的相关技术。

其次，从语法层面来看，主要技术有：

1. 中文分词：词是最小的能够独立活动的有意义的语言成分，英文单词之间是以空格作为自然分界符的，而中文是以字为基本的书写单位，词语之间没有明显的区分标记，因此，中文分词是中文自然语言处理的基础技术。
2. 词性标注：词性指作为划分词类的根据的词的特点。词性标注是给句子中每个词标记出最合适的词性。
3. 句法分析：对句子中的词语语法功能进行分析。

再次，从语义层面来看，主要技术有：

1. 指代消解
2. 语义消歧

3. 文本蕴涵

1

后来人们意识到，究竟依据什么特征来判断文本应当隶属的类别这个问题，就连人类自己都不太回答得清楚，有太多所谓“只可意会，不能言传”的东西在里面。人类的判断大多依据经验以及直觉，因此自然而然的会有人想到何让机器像人类一样自己来通过对大量同类文档的观察来自己总结经验，作为今后分类的依据。这便是统计学习方法的基本思想。

从某个意义上说，科学研究就是“盲人摸象，自圆其说”。这 8 个字可能有人会不同意，但我认为这是科学研究的真实写照，或者说真实描绘。事实上，就汉语语法研究来说，已有的结论或看法，都只能说是一种假设性的结论或看法；随着研究的逐步深入，其中有的将会被证明可以确认为定论，而大多说的结论或看法，将会被修正，甚至被完全放弃。因此我们在研究上，必须坚持“继承，借鉴，怀疑，假设，探索，求证”这 12 个字，这可以说是科学研究能有所突破的必由之路，也是汉语语法研究能有所突破的必由之路。

1.3 本书的组织结构

本书主要从五个方面来介绍自然语言处理的相关知识，如图1.1所示。

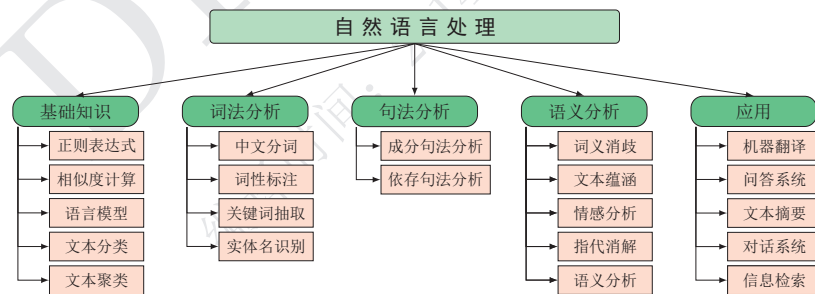


图 1.1: 本书的组织结构

1.3.1 入门基础

正则表达式、相似度计算、语言模型、文本分类、文本聚类

¹ 参考 <http://zh.wikipedia.org/wiki/自然语言处理>

中文分词

中文的语素绝大部分是单音节的。语素和语素可以组合成词（比如：吃 + 饭 \Rightarrow 吃饭）。有的语素本身就是词（手、洗），有的语素本身不是词，只能跟别的语素一起组成复合词（民人民失丧失）。现代汉语里双音节词占的比重最大。大部分双音词都是按照上面提到的复合方式造成的。有些语素虽然在现代汉语里不能作为一个词单独用，但是有时候在借用古汉语的词句时，也偶尔作为词来使用。

中文¹由于继承自古代汉语的传统，词语之间没有分隔。古代汉语中除了连绵词和人名地名等，词通常就是单个汉字，所以当时没有分词书写的必要。现代汉语的基本表达单元为“词”，以双字或多字词居多，一个字不再等同于一个词。

词性标注

词性指词的语法类别，也叫**词类**。

以现代汉语为例，词从语法功能上可以粗分为12类。

- 实词：名词、动词、形容词、数词、量词和代词
- 虚词：副词、介词、连词、助词、拟声词和叹词

但这样的划分对于后续的语义分析远远不够。因此，不同的语料构建者都会对词性进行更细的划分。比如“名词”可以再进一步分为“一般名词”、“专用名词”、“抽象名词”和“方位名词”等。“专用名词”可以再分为“人名”、“地名”、“机构名”等。不同的数据集采用的词性规范都不尽相同。

此外，对于不同语言，词性也略有不同。比如，中文里有“量词”，而英文里没有。英文中的“冠词”在中文里也不存在。

词性标注是给句子中每个词标记出最合适的词性，即在给定的词性集合中选择一个合适的词性。一个词在不同的上下文环境下会有充当不同的语法功能，表现为不同的词性。比如，“希望”即是名词又是动词。

watch n. 手表 v. 观看

但在特定的使用场合下，比如一个句子中，每个词都有唯一确定的词性。

词	去年	他	取得	了	可喜	的	进步	。
词性	时间短语	代词	动词	时态词	形容词	结构助词	名词	标点

¹ 中文，或汉语，按照字型分为简体中文和繁体中文。按照时间可分为古代汉语和现代汉语。本书所述的中文在不作特别说明是一般指现代汉语。

在FNLP中，词性分为：动词、能愿动词、趋向动词、把动词、被动词、形谓词、形容词、副词、名词、方位词、人名、地名、机构名、时间短语、邮件、网址、型号名、实体名、疑问代词、指示代词、人称代词、量词、介词、数词、惯用词、限定词、连词、叹词、序数词、省略词、语气词、结构助词、时态词、标点、拟声词、表情词等。

命名实体识别

命名实体识别（NE）是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名和其它实体名等。

1.3.2 句法分析

成分句法分析、依存句法分析句法分析 (Parsing) 是指对句子中的词语语法功能进行分析。

FNLP 的依存关系类型包括：核心词、主语、补语、宾语、定语、状语、并列、同位语、数量、介宾、连动、疑问连动、兼语、关联、重复、标点、的字结构、地字结构、得字结构、语气、时态等。

1.3.3 语义分析

在自然语言处理中，对语义的研究也是一个非常重要的研究方向。对自然语言的词法和句法分析并不是最终的目标，而是一种让计算机更准确地理解语言的工具。

“Colorless green ideas sleep furiously”，是Chomsky在1957年的《Syntactic Structures》中使用的著名例句。虽然这个句子语法正确，但语义荒谬，可以用来说明语法和语义的区别，也可以说明目前主流的统计语法模型的不足。

比如“红烧肉的做法”、“怎么做红烧肉”和“红烧肉制作方法”，这三句表示的含义是一样的。对于自然语言的语义可以从两个维度来看。

一是从不同粒度来看，可以分为词、短语、句子、篇章的语义。二是从表示方法来看，形式化语义（Formal Semantics）和分布式语义（Distributional Semantics）两种表示模型。

现在比较成功的是对词汇意义的研究，但是对更高粒度上的语义研究还在发展中。研究的内容主要在于词汇的意义和结构，比如说义素分析，语义场，词义之间的结构关系等等。这样的语义学研究也可以称为词汇语义学，词和词之间的各种关系是词汇语义学研究的一个方面，例如同义词、反义词，同音词等，找出词语之间的细微差别。

范畴错误 (Category Mistake) 是指将既有的属性归属到不可能应该拥有该属性的对象上，为语义学或存在论的错误。

词义消歧、文本蕴涵、情感分析、指代消解、语义分析。

指代消解

共指 (Coreference) 和指代 (Anaphora) 是自然语言表达中的常见现象。在语言学中，为了避免已经出现的字词重复出现在文章的句子上，导致语句结构过于赘述和语义不够清晰，所以使用代名词、普通名词或缩略语来代替已经出现过的字、词、短语或句子。共指是指两个词 (或短语) 都指向同一对象 (或实体)。指代是指一个指代词对先行词的依赖关系，比如，等价关系、上下位关系、整体和部分关系等。一般意义上讲，共指可以脱离上下文存在。而指代是在一个小的范围内存在。

例如：下面一段文字中，“北极熊”、“白熊”、“肉食动物”和“熊”都指向同一个实体，属于共指现象。而“那里”和“北极”是指代关系。

北极熊又称**白熊**，是陆上最庞大的**肉食动物**。这种**熊**在**北极**生活，**那里**非常寒冷。

近年来，共指消解 (Coreference Resolution) 和指代消解 (anaphora resolution) 的研究受到了格外的关注，2000 年开始的 ACE (Automatic Content Extraction) 评测会议中共指消解也是重要内容之一。中文的共指消解研究开始于二十世纪末。中文共指消解评测开始于 2003 年 ACE 会议。

这里我们主要考虑指代消解问题。王厚峰 [2002]

按照指向，可以分为回指和预指。回指就是代词的先行语在代词前面，预指就是代词的先行语在代词后面。按照指代的类型可以分为三类：人称代词、指示代词、有定描述、省略、部分—整体指代、普通名词短语。这些类别中前四个都是和语言学息息相关的。

FNLP 提供了指代消解功能。对应的类为 edu.fudan.nlp.cn.anaphora.Anaphora。我们可以通过如下 API 来调用¹：

1.3.4 应用

机器翻译、自动问答、文本摘要、对话系统、信息抽取、信息检索。

¹ 更多使用示例请参考最新版发布包内 example 文件夹里的示例代码 AnaphoraResolution.java

参考文献和深入阅读

关于形式语言的研究成果可以阅读 [Chomsky and Miller, 1963]。

若希望全面了解自然语言处理，可以参考以下几本书或相对应的中文版。

1. J. Allen. Natural language understanding. Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA, 1995
2. C.D. Manning and H. Schütze. Foundations of statistical natural language processing. MIT Press, 1999
3. D. Jurafsky, J.H. Martin, A. Kehler, K. Vander Linden, and N. Ward. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, volume 2. Prentice Hall New Jersey, 2000
4. 宗成庆. 统计自然语言处理（第二版）. 中文信息处理丛书. 清华大学出版社, 2013

2011年初，IBM 超级计算机 Watson 在美国最受欢迎的智力竞猜节目《Jeopardy》中击败了两名人类选手，最终获得胜利。Watson 的目标是建造一个能与人类回答问题能力匹敌的计算系统。在比赛中，参赛者必须要回答一系列的问题，主要涉及历史，文学，政治，电影，流行文化和科学。这要求计算机具有足够的速度、精确度和置信度，并且能使用人类的自然语言回答问题。比赛题目需要分析人类语言中微妙的含义、讽刺口吻、谜语等，这些通常是人类擅长的方面，一直以来计算机在这方面毫无优势可言。Watson 的成功也是人工智能的又一次标志性进展，也使得工业界对自然语言处理有了新的认识。

自然语言处理技术最终还取决于人工智能、语言学、生物学、心理学等多个学科的发展。目前，自然语言处理在很多领域和任务上取得了一定的进展。但是这些方法犹如管中窥豹，盲人摸象，还不足以揭示人们如何理解自然语言处理的机理。因此，我们需要不断的互相交流，最大限度地实现信息和数据共享。

第二章 自然语言的形式化描述

让一只猴子在打字机上随机地按键，当按键时间达到无穷时，几乎必然能够打出任何给定的文字，比如莎士比亚的全套著作

— 无限猴子定理

为了让计算机可以处理和分析自然语言，最理想的方式是把自然语言加以形式化或公式化，转换成计算机可以接受的形式。我们先来总结下自然语言的特点。

首先，语言由**字符**组成。不同语言中，字符的表示和意义是不同的。比如中文有上万个汉字，而英文只有 26 个字母。汉字的大多数字都有特定的含义，而英文字母只是语音单位。然而，不是所有的一连串字符序列都可以构成语言。有一个很著名的**无限猴子定理**：让一只猴子在打字机上随机地按键，当按键时间达到无穷时，几乎必然能够打出任何给定的文字，比如莎士比亚的全套著作。虽然这个定理在统计上成立，但实际上不可能发生。那什么样的字符串才构成语言呢？

最后，自然语言是信息的载体。每个词、短语、句子、篇章都涵盖了一定语义信息。这些语义信息还需要结合一定的背景知识才能理解。比如让一个学计算机的人看一篇医学论文，他估计会看得头疼。

总之，语言是在一个特定的字符集上，通过一定的组合规则产生的字符序列的集合。这些组合规则究竟是什么？不同语言的组合规则是否有共通之处？虽然远早于计算机出现之前，语言学家已经进行了大量的研究，但至今也没有明确的定论。归根到底，语言与语义、文化有着千丝万缕的联系，很难孤立地去语言。但是语言学的研究中有很多成果给自然语言处理构建了一个良好的基础。

掌握了上面的特点，我们就可以在很多场合用计算机来分析自然语言了。我们先从简单的例子上手。比如口语语言的算术运算。在这个应用场景中，输入的文字不是 $1 + 1 = ?$ 或 $10 * 21 = ?$ 这种数学符号，而是“一加一等于几”或“十乘二十一等于多少”这样的自然语言。根据上面我们分析的自然语言的特点，我们需要确定可以产生算术运

算的字符集合，语法规则以及相应的语义。我们接下来一步步分析如何进行这个特定应用场景的自然语言分析、理解和生成。

为了让计算机理解一门语言，我们需要用计算机擅长的方式来描述语言。通过建立语言模型，用数学或逻辑形式来对语言的各种特性进行精确的描述。

我们如何公式化地描述这个规则呢？我们下面介绍两个自然语言处理中常用的两个描述工具：一个是正则表达式，另一个是上下文无关文法。

2.1 字符

2.1.1 字符以及字符编码

在自然语言处理中，我们首先要面对的就是字符在计算机中如何表示。计算机中处理的信息单位是比特 (bit)，只有 0、1 两种取值。

为了表示字符，首先需要定义不同的字符如何用比特进行编码。

常有的例子就是 ASCII 码。ASCII 码将英文的字母、数字和其它符号编号，并用 7 个比特的二进制来表示这个整数。通常会额外使用一个扩充的比特，以便于以 8 位字节的方式存储。但是 ASCII 码只对英文字母进行了编码，它的局限性很大。随着计算机的普及，人们需要计算机要处理不同语言中的字符的，于是人们开发了很多编码方法进行扩展。比如对于中文，就有很多种编码方式，像中国国家标准总局发布的 GB 2312、GB 18030 和非国家标准的 GBK。每个汉字以及汉字的符号都用两个字节来表示。由于不同语言的编码方案之间经常出现不兼容的情况，给人们的使用带来很多不便，特别是在多语言环境下。比如我们要打开一个文本或网页，需要事先知道它的编码方式，不然就会发生乱码¹。因此，人们为了解决传统的字符编码方案的局限，提出了统一编码方案，即 Unicode。在本书中我们都默认都使用 Unicode 编码。

2.1.2 字符串

字符串(String) 是零个或多个字符组成的有限序列。

另外，我们用 ϵ 来表示特殊的空字符串，其长度为 0。

字符串有以下几个属性：

☞ **长度** 字符串中字符的个数。比如 “language” 的长度为 8。 ϵ 的长度为 0。

¹ 目前很多文本编辑器或网页浏览器都支持智能的语言和编码检测，这也是一个重要的自然语言处理技术，我们会在后面谈到。

```
"language".length(); //长度为8
"".length(); //长度为0
```

☞ **子序列** 字符串的子序列是从该字符串通过去除某些元素但不破坏余下元素的相对位置而形成的新序列。比如“lge”“anua”都为“language”的子序列。

☞ **子串** 子串是子序列的子集，区别在于子串必须是有原字符串中的连续字符构成。比如“ngu”“lang”都为“language”的子序列。

```
//String.substring(int i, int j) 子串开始于位置 i，结束于位置 j-1。
```

```
"language".substring(2, 5); //子串"ngu"
"language".substring(0, 4); //子串"lang"
```

一个字符串包含它的所有子串。

```
"language".contains("lan"); //true
"language".contains("lana"); //false
```

☞ **前缀** 和 **后缀** 前缀是指从字符串开始位置开始到某个位置结束的一个特殊子串，而后缀是指从某个位置开始到整个字符串结尾的一个特殊子串。比如“lan”、“age”分别为“language”的前缀和后缀。

```
"language".substring(0,3); //前缀"lan"
"language".substring(5,8); //后缀"age"
```



Java 对字符串操作的支持已经非常完善了，字符串操作主要为 **String**、**StringBuffer**、**StringBuilder**、**CharSequence** 类。除了 Java 自身的类，还有很多第三方的类库也提供功能更加强大的字符串操作。比如流行的 Apache 的 **StringUtils** 类¹。

2.1.3 字符串运算

字符串的一个最基本的运算是**连接**操作。如果 x 和 y 是两个字符串，则 x 和 y 的连接为 xy ，是将 y 附加在 x 的后面。例如， x 为 ba， y 为 na，则 xy 为 bana， xyy 为 banana。

对于一个字符串和它自身的连接，我们可以用指数运算来表示。比如 xx 可以表示为

¹ <http://commons.apache.org/proper/commons-lang/>

表 2.1: 口语语言算术运算的语言字符集

字符集	字符
数字	一、二、三、四、五、六、七、八、九、零
单位	十、百、千、万、亿
运算符	加、减、乘、除
等式	等、于
其他	几、多、少

x 的 2 次方 x^2 , xxx 可以表示为 x 的 3 次方 x^3 。另外, 我们定义 $x^0 = \epsilon$, $x^1 = x$ 。 xy^2 为 banana.

2.2 语言

语言就是指在一个有限的**字符集**上, 产生的符合一定规则的字符串集合。这里的语言不限于自然语言, 也包括数学语言、程序语言等人工语言。

2.2.1 字符集

字符集是字符的有限集合。最常见的字符集是 ASCII 字符集, 其包含了来表示所有的大写和小写字母, 数字、标点符号以及在美式英语中使用的特殊控制字符。

不同语言的字符集也都不相同。英语的字符集包含了 26 个字母的大小写和英语标点等, 而汉语的字符集包含了上万个汉字、中文标点以及一些外来符号等。

在自然语言中, 字符集会随着语言的演化为逐渐改变。比如汉语的字符集从以前的繁体字改进为简体字, 以及一些外来字符 (罗马数字, 字母等)。

比如在口语语言算术运算的语言中, 字符集见表 2.1 所示。其中, 并不是字符集中的所有单个字符都有语义信息。有些字符必须组合成一个字符串作为一个语义单位。比如“等”和“于”必须组合成“等于”才有明确的语义。

2.2.2 字符串集合运算

语言是由字符集中字符产生的字符串集合, 那么我们如何通过字符集生成一个字符串集合呢。我们先来了解下字符串集合的基本运算。

字符串集合的运算包括连接和合并等操作。

假定有限的字符集为 \mathcal{A} ， \mathcal{A} 中的字符变量用 α 表示。 α 也相当于长度为 1 的字符串。 \mathcal{R} ， \mathcal{S} 为两个字符串的集合。

则两个集合的连接 \mathcal{RS} 定义为 \mathcal{R} 中的所有元素和 \mathcal{S} 中的所有元素进行字符串连接得到的集合。

$$\mathcal{RS} = \{\alpha\beta | \alpha \in \mathcal{R}, \beta \in \mathcal{S}\} \quad (2.1)$$

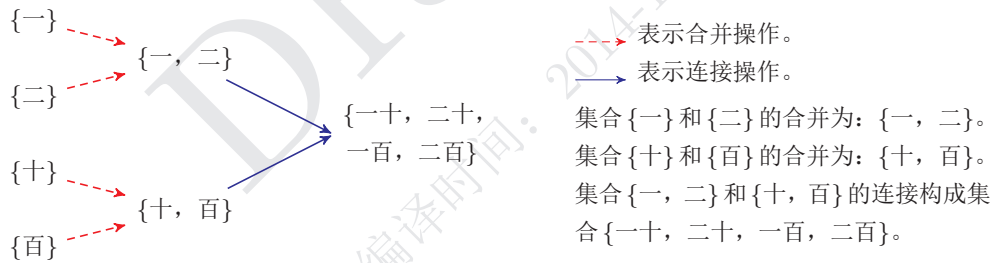
则两个集合的合并 $\mathcal{R}|\mathcal{S}$ 定义为 \mathcal{R} 和 \mathcal{S} 的并集，即这两个集合的所有元素构成的集合。

$$\mathcal{R}|\mathcal{S} = \{\alpha | \alpha \in \mathcal{R} \text{ or } \alpha \in \mathcal{S}\} \quad (2.2)$$

字符集中的每个字符都可以形成一个包含自身的集合，集合的元素个数为一。这样通过字符串集合的运算我们就可以得到无限大的字符串集合。

对于字符集 2.1 的字符，集合 $\{\text{一}\}$ ， $\{\text{二}\}$ ， $\{\text{十}\}$ ， $\{\text{百}\}$ ，通过合并和连接操作，我们可以得到一个集合 $\{\text{一十}, \text{二十}, \text{一百}, \text{二百}\}$ 。

构造过程如下所示：



同样地，对于一个集合和它自身的连接，我们用指数运算来表示。比如 $\mathcal{R}^2 = \mathcal{R}\mathcal{R}$ ， $\mathcal{R}^3 = \mathcal{R}\mathcal{R}\mathcal{R}$ ，以此类推。

Kleene 星号，或称 **Kleene 闭包**，表示为 \mathcal{R}^* ，是可以通过 \mathcal{R} 中的零或多个字符串的串接得到所有字符串的集合。

$$\mathcal{R}^* = \{\mathcal{R}^0 | \mathcal{R}^1 | \mathcal{R}^2 | \dots\} \quad (2.3)$$

其中， $\mathcal{R}^0 = \{\epsilon\}$ 。

例如， $\{ab, c\}^* = \{\epsilon, ab, c, abab, abc, cab, cc, ababab, \dots\}$ 。

2.3 正则表达式

在表2.1所包含的字符集中，并不是任意两个字符都能组合在一起都可以形成一个正确的算术表达式。算术语言有一定的规则，必须符合规则的字符串才是合格的。因此，我们需要定义这个规则，并形式化这些规则，以便让计算机理解和执行。

正则表达式(Regular Expression) 就是一个非常简单的语言规则描述方法，用来描述一系列符合某个句法规则的字符串的集合。最初的正则表达式出现于理论计算机科学的自动控制理论和形式语言理论中。此后，正则表达式被广泛地使用于各种编辑工具。

正则表达式由字符集中的字符和一些特定的操作符组成，来表示字符串的集合和在这些集合上的运算。一个正则表达式通常也被称为一个**模式** (Pattern)。每个正则表达式都对应一个字符串集合。因此，正则表达式也必须支持我们之前介绍的集合运算。

完整的正则表达式由两种字符构成：元字符（有特殊含义的字符,metacharacters）和普通文本字符。正则表达式采用如下递归定义：

1. 字符集中的每个字符都是一个正则表达式。
2. 空字符串和空集也是正则表达式。
3. 如果X 和Y 是正则表达式，那么下面的式子也是正则表达式。

☞ **连接：XY**

表示Y跟在X后面

☞ **或：X|Y**

|表示X或Y。例如 color|colour 可以匹配"color"或"colour"。

☞ **Kleene 星号：X***

零个或多个X

这三个基本操作的优先级由高到低依次为 Kleene 星号、连接、或。比如：abc* 表示的字符串集合是 ab 然后跟随 0 个或多个 c，而不是 0 个或多个 abc。


为了提高正则表达式的表示能力，大部分高级程序语言中都支持下面的扩展的操作符。

☞ **分组 (X)**


圆括号可以用来改变操作的范围和优先度，也可以用作捕获组。比如 (abc)* 表示要匹配 0 个或多个 abc。

☞ **量词**：除了 Kleene 星号外，还支持如下量词：

- **X?** 问号表示X出现0次或一次，相当于 $X|\epsilon$ 。例如，`colou?r` 可以匹配可以匹配"color" 或"colour"。
- **X+** 加号代表前面的字符必须至少出现一次，相当于 XX^* 。
- **X{n}** 表示X出现n次。
- **X{n,}** 表示X至少出现n次。
- **X{n,m}** 表示X至少出现n次，但是不超过m次。


 **字符类**：在默认的字符集上，可以用下面的字符类来简化正则表达式。一般正则表达式是不支持交集或补集操作的，原因是正则表达式对应的集合一般都是无穷大的，而交集或补集一般针对有限大小的集合。但是在字符类上，正则表达式可以有交集和补集操作。

- **[abc]** 简单类，表示a、b或c。
- **[^abc]** 补集，表示除了a、b或c之外的任何字符
- **[a-zA-Z]** 范围，表示a到z或A到Z，两头的字母包括在内
- **[a-d[m-p]]** 并集，表示a到d或m到p：[a-dm-p]
- **[a-z&&[def]]** 交集，表示d、e或f
- **[a-z&&[^bc]]** 减去，表示a到z，除了b和c：[ad-z]
- **[a-z&&[^m-p]]** 减去，表示a到z，而非m到p：[a-lq-z]

 **预定义字符类**：提供了常用正则表达式的简写形式。无论何时都有可能使用预定义字符类，它可以使代码更易阅读，更易从难看的字符类中排除错误。

- **.** 任意字符
- **\d** 数字：[0-9]
- **\D** 非数字：[^0-9]
- **\s** 空白字符：[\t\n\x0B\f\r]
- **\S** 非空白字符：[^ \s]
- **\w** 单词字符：[a-zA-Z_0-9]
- **\W** 非单词字符：[^ \w]

上述的扩展可以让我们更灵活多变的构造正则表达式，比如 `color|colour` 可以写为 `colou?r` 或 `col(or|our)`。为了避免过多的分组，如果没有歧义则可以省略括号。例如，`(ab)c` 可以写为 `abc`，而 `a|(b(c*))` 可以写为 `a|bc*`。

 电话号码：`\d{11}|\d{3,4}-\d{8}`

 Email地址：：`\w+([-+.] \w+)*@\w+([-.] \w+)*\.\. \w+([-.] \w+)*`

☞ 无符号数字: $\backslash d+(\backslash.\backslash d+)?(E(+|-)?\backslash d)?$

是不是很简单？但是，如果把这些正则表达式用在 Java 代码的字符串中，还需要进行转义构造。以反斜线（‘\’）开始的构造称为**转义构造**。例如：

```
String REGEX = "\\d";           // 单个数字
```

在使用正则表达式编译器编译这些正则表达式之前，Java 编译器需要先处理这些字符串。根据 Java 的语言规范，源代码的字符串中的反斜线被解释为 Unicode 转义或其他字符转义。因此正则表达式中的每一个反斜线前必须再增加一个反斜用于字符串的编译。



本书提到的元字符仅仅是 Java 支持的元字符集合中的一部分，更多的元字符定义可以参考 JDK API 文档



反斜线字符（‘\’）用于引用转义构造，同时还用于引用其他将被解释为非转义构造的字符。因此，表达式\\与单个反斜线匹配，而\\{与左括号匹配。在不表示转义构造的任何字母字符前使用反斜线都是错误的。可以在非字母字符前使用反斜线，不管该字符是否非转义构造的一部分。例如，当解释为正则表达式时，字符串字面值\b与单个退格字符匹配，而\\b与单词边界匹配。字符串字面值\\(hello\\)是非法的，将导致编译时错误；要与字符串“(hello)”匹配，必须使用字符串字面值\\(hello\\)。

2.3.1 匹配

给定一个正则表达式和另一个字符串，我们可以达到如下的目的：1. 给定的字符串是否符合正则表达式的过滤逻辑（称作“匹配”）；2. 可以通过正则表达式，从字符串中获取我们想要的特定部分。

Java 中用于匹配字符序列与正则表达式指定模式软件包 `java.util.regex` 中的两个类：**Pattern**类和**Matcher**类。

Pattern类的实例采用类似于 Perl 的语法来表示以字符串形式指定的正则表达式。

Matcher类的实例用于匹配字符序列与给定模式。通过 `CharSequence` 接口将输入提供给匹配器，以支持从多种输入源到字符的匹配。

练习 2-1 构造一个正则表达式用于验证“密码字符串”，依次满足如下要求：

1. 只能由小写字母、数字和横线（-）组成；
2. 满足条件 1，并且开头和结尾不允许是横线；
3. 满足条件 2，并且不允许有连续（超过一个）的横线。
4. 满足条件 3，并且不允许全部是数字；

答案 2-1

1. `[\d\w-]+`
2. `[\d\w]+ [\d\w-]+[\d\w]+`
3. `[\d\w]+ ([\d\w]+ - [\d\w]+)+[\d\w]+`
3. `[\d\w]+ ([\d\w]+ - [\d\w]+)+[\d\w]+\w[\d\w]+ ([\d\w]+ - [\d\w]+)+[\d\w]+`

2.3.2 Unicode 支持

上面我们提到的字符类中只包含了非常简单的类别：英文字母以及数字。但是在目前的文本处理中，这些字符类远远不够。比如如果我们如果想过滤下面字符串中的夹杂各种标点符号，怎么处理呢？

"测试 <> 《》!*(^% !@#\$...&¥—+=、。、; “ ”: • 文本"

当然，我们穷举出所有的标点符号，然后用正则表达式描述出来。在 Java 中，我们可以借助 Unicode 标准中的字符属性。

```
String string="测试<>《》!*(^% !@#$...&¥—+=、。、; “ ”: • 文本";
System.out.println(string.replaceAll("\\pP|\\pS",
""));
//输出:
"测试文本"
```

这里 `\p` 表示 Unicode 属性，用于 Unicode 正表达式的前缀。大写 `P` 表示 Unicode 字符集七个基本字符属性之一：标点字符，`S` 表示符号（比如数学符号、货币符号等）。

- > L: 字母;
- > M: 标记符号（一般不会单独出现）;

- Z: 分隔符 (比如空格、换行等);
- S: 符号 (比如数学符号、货币符号等);
- N: 数字 (比如阿拉伯数字、罗马数字等);
- Z: 分隔符 (比如空格、换行等);
- C: 其他字符

上面这七个是基本属性, 每个基本属性还有若干个子属性, 用于更进一步地进行细分。比如Sc表示货币符号。Unicode 的标准可以参考<http://www.unicode.org/reports/tr18/>, 可以找到所有的子属性。



Java 中用于 Unicode 的正则表达式数据都是由 Unicode 组织提供的。Unicode 编码并不只是为某个字符简单定义了一个编码, 而且还将其进行了归类。具体每个 Unicode 字符属性的定义可以参考<http://www.unicode.org/Public/UNIDATA/UnicodeData.txt>。这个文本文档一行是一个字符, 第一列是 Unicode 编码, 第二列是字符名, 第三列是 Unicode 属性, 以及其他一些字符信息。

2.3.3 正则表达式的不足

正则表达式的特点是: 1. 灵活性、逻辑性和功能性非常的强; 2. 可以迅速地用极简单的方式达到字符串的复杂控制。

虽然正则表达式提供了非常便捷的语言描述方法, 但是依然存在一些不足。

1. 可读性不高。当描述一个复杂的语言集合时, 正则表达式的可读性就变得很差。
2. 描述能力有限¹。比如对与语言集合 a^nb^n (n 为变量)。正则表达式并不能描述这个集合。正则表达式的运算只有闭包运算 $*$ 一种, 但 a^*ba^* 能保证 b 前后的 a 的数量是否相等。不能描述层次结构
3. 正则表达式只适合匹配文本字面, 不适合匹配文本意义: 像匹配 url, email 这种纯文本的字符就很好, 但比如匹配多少范围到多少范围的数字, 如果你这个范围很复杂的话用正则就很麻烦。或者匹配 html, 这个是很多人经常遇到的, 写一个复杂匹配 html 的正则很麻烦, 不如使用针对特定意义的处理器来处理 (比如写语法分析器, dom 分析器等)
4. 容易引起性能问题: 像 $*$ 这种贪婪匹配符号很容易造成大量的回溯, 性能有时候会有上百万倍的下降, 编写好的正则表达式要对正则引擎执行方式有很清楚的理解才可以。

¹ 在自动机理论中的数学论证是著名的泵引理 (Pumping lemma) <http://zh.wikipedia.org/wiki/>

5. 正则的替换功能较差：甚至没有基本的截取字符串或者把首字母改变大小写的功能，这对于 url 重写引擎有时候是致命的影响

总之，只要熟练应用正则表达式，而且匹配的目标是纯文本，那么相比于写分析器来说，正则可以更快速的完成工作。还有在捕获字符串的能力，正则也可以很好的完成工作，比如截取 url 的域名或者其他的内容等等

正则表达式的定义非常精简，避免多余的量词？和 +，它们可以被表达为： $a+ = aa^*$ 和 $a? = (a|\epsilon)$ 。有时增加补算子 \bar{R} ； \bar{R} 指示在 R 上的不在 R 中的所有字符串的集合。补算子是多余的，因为它使用其他算子来表达（尽管计算这种表示的过程是复杂的，而结果可能以指数增大）。

中文算术表达式

只要熟练应用正则表达式，而且匹配的目标是纯文本

2.4 上下文无关文法

上下文无关文法（Context-Free Grammar, CFG），也叫短语结构文法（Phrase-Structure Grammar）。

在计算机科学中，若一个形式文法 $G = (N, \Sigma, P, S)$ 的上下文无关文法产生式规则都取如下的形式： $V \rightarrow w$ ，则称之为上下文无关的，其中 $V \in N$ ， $w \in \Sigma^*$ 。上下文无关文法取名为“上下文无关”的原因就是因为字符 V 总可以被字符串 w 自由替换，而无需考虑字符 V 出现的上下文。一个形式语言是上下文无关的，如果它是由上下文无关文法生成的（条目上下文无关语言）。上下文无关文法重要的原因在于它们拥有足够强的表达力来表示大多数程序设计语言的语法；实际上，几乎所有程序设计语言都是通上下文无关文法过上下文无关文法来定义的。另一方面，上下文无关文法又足够简单，使得我们可以构造有效的分析算法来检验一个给定字符串是否是由某个上下文无关文法产生的。例子可以参见 LR 分析器和 LL 分析器。BNF（巴克斯-诺尔范式）经常用来表达上下文无关文法。文法规则使用相似的表示法。名字用斜体表示（但它是一种不同的上下文无关文法字体，所以可与正则表达式相区分）。竖线仍表示作为选择的元符号。并置也用作一种标准运算。但是这里没有重复的元符号（如正则表达式中的星号*），稍后还会再讲到它。表示法中的另一个差别是现在用箭头符号“ \rightarrow ”代替了等号来表示名字的定义。这是由于现在的名字不能简单地由其定义取代，而需要更为复杂的定义过程来表示，这是由定义的递归本质决定的。同正则表达式类似，文法规则是定义在一个字母表或上下文无关文法符号集之上。在正则表达式中，这些符号通常就是字符，而在文法规则中，符号通常是表示字符串的记号。我们利用 C 中的枚举类型定义了扫描程序中的记号；为了避免涉及到特定实现语言（例如 C）中表示记号的细节，就使用了正则表达式本身来表示记

号。此时的记号就是一个固定的符号，如同在保留字 `while` 中或诸如 `+` 或 `=` 这样的特殊符号一样，对于作为表示多于一个串的标识符和数的记号来说，代码字体为斜体，这就同假设这个记号是正则表达式的名字（这是它经常的表示）一样。上下文无关文法的却利用了与正则表达式中极为类似的命名惯例和运算，二者的主要区别在于上下文无关文法的规则是递归的（recursive）。

2.4.1 应用

Facebook 的 graph search 如何进行自然语言？答案是上下文无关文法。算是特定场合下的 NLP 了。解码用的是 A* 算法。

```

start → users
users → user
users → user-headuser-filter+
user-head → my friends
user-filter+ → who go to schools
schools → {college}
schools → school attended by users

```

参考文献和深入阅读

有关正则表达式构造行为更准确的描述，请参见 Jeffrey Friedl. Mastering regular expressions. O'Reilly Media, Inc., 2006。正则表达式的匹配都是通过正则表达式引擎实现的。正则表达式引擎分为两类：基于 NFA (Nondeterministic Finite Automata, 非确定型有穷状态自动机) 和基于 DFA (Deterministic Finite Automaton, 确定型有穷状态自动机) 的引擎。关于自动机理论，可以参考 John E Hopcroft. 自动机理论, 语言和计算导论. 清华大学出版社, 2002。DFA 和 NFA 的区别在于，DFA 对于一个状态和一个输入，一定会有一个唯一的后续状态，而 NFA 可能有多个状态，也可能没有。一般来说，DFA 正则编译的时候花的时间会多一点，但是在匹配的时候会更快一点。不同的程序语言对正则表达式有不同的改进，使用的自动机也不尽相同。在 Java 中，使用 NFA 和 DFA 结合方法。

虽然通过形式语言的方法可以处理很多问题，但是这种方法有几点不足。一是这些语法规则都是根据专家知识人工撰写的，费时费力，维护成本很高；二是形式语言的内在本质都是确定性的，不能处理自然语言的歧义现象；三是这个语言规则只能描述部分语言现象，而无法涵盖各种语言现象，因此只能在单一、封闭的任务中使用。随着计算

机计算能力的不断提高，基于大规模的训练数据的机器学习方法被引入到自然语言处理中。和之前的基于规则的方法相比，这种方法更具鲁棒性。

<http://www.regular-expressions.info/unicode.html>

DRAFT
编译时间: 2014-10-31 16:03

第三章 自然语言的概率模型

让一只猴子在打字机上随机地按键，当按键时间达到无穷时，几乎必然能够打出任何给定的文字，比如莎士比亚的全套著作。

— 无限猴子定理

我们来看一个完形填空的例子。完形填空在语言水平测试中经常出现，是从一个语义连贯的句子中有去掉一些词语，形成空格，要求从给出的候选答案中，选出一个正确的或最佳的答案，使句子的恢复完整。

例 3.1. 在下面的句子，有三个空格，用“书”，“通俗易懂”和“实用性”三个候选词进行填充以使句子完整。

这本 _____ 写得 _____，_____ 很强。

对于母语是中文的人来说，这是非常容易的。我们靠语感就可以轻松完成。但是如果让我们如何让计算机来完成呢？什么是“语感”呢？又如何建立形式化的语言模型呢？是否可以用我们之前介绍的正则表达式或上下文无关文法呢？好像不是那么容易。但幸运的是，我们可以用一个很简单的方法获得一个语言模型来判断一个句子是否合理。这就是**统计语言模型**（Statistical Language Model）。因为统计语言模型的广泛成功应用，在统计自然语言处理中，一般可以直接简称为**语言模型**。

3.1 概率论基础

在介绍统计语言模型之前，我们先介绍一些概率论的基本概念。概率论是研究大量随机现象中数量规律的数学分支。

数学小知识 | 笛卡儿乘积

在数学中，两个集合 X 和 Y 的笛卡儿乘积 (Cartesian product)，又称直积，在集合论中表示为 $X \times Y$ ，是所有可能的有序对组成的集合，其中有序对的第一个对象是 X 的成员，第二个对象是 Y 的成员。

$$X \times Y = \{\langle x, y \rangle \mid x \in X \wedge y \in Y\}.$$

举个实例，如果集合 X 是 13 个元素的点数集合 $\{A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2\}$ ，而集合 Y 是 4 个元素的花色集合 $\{\heartsuit, \spadesuit, \clubsuit, \diamondsuit\}$ ，则这两个集合的笛卡儿积是有 52 个元素的标准扑克牌的集合 $\{(A, \heartsuit), (K, \heartsuit), \dots, (2, \heartsuit), (A, \spadesuit), \dots, (3, \spadesuit), (2, \spadesuit)\}$ 。

3.1.1 样本空间

样本空间是一个实验或随机试验所有可能结果的集合，而随机试验中的每个可能结果称为样本点。通常用 Ω 表示。例如，如果抛掷一枚硬币，那么样本空间就是集合 $\{\text{正面}, \text{反面}\}$ 。如果投掷一个骰子，那么样本空间就是 $\{1, 2, 3, 4, 5, 6\}$ 。

有些实验有两个或多个可能的样本空间。例如，从 52 张扑克牌中随机抽出一张，一个可能的样本空间是数字 (A 到 K)，另外一个可能的样本空间是花色 (黑桃，红桃，梅花，方块)。如果要完整地描述一张牌，就需要同时给出数字和花色，这时的样本空间可以通过构建上述两个样本空间的笛卡儿乘积来得到。

3.1.2 事件和概率

随机事件 (或简称事件) 指的是一个被赋予机率的事物集合，也就是样本空间中的一个子集。**概率**表示对一个随机事件发生的可能性大小，为 0 到 1 之间的一个非负实数。

若样本空间是关于一个机会均等的抛硬币动作，则样本输出为“正面”或“反面”。事件为：

- $\{\text{正面}\}$ ，其概率为 0.5；
- $\{\text{反面}\}$ ，其概率为 0.5；
- $\{\} = \emptyset$ ，不是正面也不是反面，其概率为 0；
- $\{\text{正面}, \text{反面}\}$ ，不是正面就是反面，这是，其概率为 1。

3.1.3 随机变量

在随机实验中，很多问题是与数值发生关系，试验的结果可以用一个数 X 来表示，这个数 X 是随着试验结果的不同而变化的，是样本点的一个函数。我们把这种数称为**随机变量**。

例如，随机掷一个骰子，得到的点数就可以看成一个随机变量 X ， X 的取值为 $\{1, 2, 3, 4, 5, 6\}$ 。

如果随机掷两个骰子，整个事件空间可以由 36 个元素组成：

$$\Omega = \{(i, j) | i = 1, \dots, 6, j = 1, \dots, 6\} \quad (3.1)$$

这里可以构成多个随机变量，比如随机变量 X （获得的两个骰子的点数和）或者随机变量 Y （获得的两个骰子的点数差），随机变量 X 可以有 11 个整数值，而随机变量 Y 只有 6 个。

$$X(i, j) := i + j, \quad x = 2, 3, \dots, 12 \quad (3.2)$$

$$Y(i, j) := |i - j|, \quad y = 0, 1, 2, 3, 4, 5. \quad (3.3)$$

又比如，在一次扔硬币事件中，如果把获得的背面的次数作为随机变量 X ，则 X 可以取两个值，分别是 0 和 1。

离散随机变量

如果**随机变量** X 所可能取的值为有限可列举的，假设有 n 个有限取值 $\{x_1, \dots, x_n\}$ 。 X 称为**离散随机变量**。我们一般用大写的字母表示一个随机变量，用小写字母表示该变量的某一个具体的取值。

要了解 X 的统计规律，就必须知道它取每种可能值 x_i 的概率，即

$$P(X = x_i) = p(x_i), i = 1 \dots, n \quad (3.4)$$

$p(x_i), i = 1 \dots, n$ 称为离散型随机变量 X 的**概率分布**或**分布**，并且满足

$$\sum_{i=1}^n P(x_i) = 1 \quad (3.5)$$

$$P(x_i) \geq 0, \quad i = 1 \dots, n \quad (3.6)$$

下面看些常见的离散随机变量以及其概率分布的例子。

数学小知识 | 排列组合

排列组合是组合学最基本的概念。所谓**排列**，就是指从给定个数的元素中取出指定个数的元素进行排序。**组合**则是指从给定个数的元素中仅仅取出指定个数的元素，不考虑排序。排列组合的中心问题是研究给定要求的排列和组合可能出现的情况总数。

排列的任务是确定 n 个不同的元素的排序的可能性。 n 个不同的元素可以有 $n!$ 种不同的排列方式，即 n 的阶乘。

$$n! = n(n-1) \cdots 3 \cdot 2 \cdot 1.$$

如果从 n 个元素中取出 k 个元素，这 k 个元素的排列总数为

$$P_n^k = \frac{n!}{(n-k)!}.$$

从 n 个元素中取出 k 个元素，这 k 个元素可能出现的组合数为

$$C_n^k = \binom{n}{k} = \frac{P_n^k}{k!} = \frac{n!}{k!(n-k)!}$$

区分排列与组合的关键是“有序”与“无序”。

[伯努利分布] 在一次试验中，事件 A 出现的概率为 p ，不出现的概率为 $q = 1 - p$ 。若用变量 X 表示事件 A 出现的次数，则 X 的取值为 0 和 1，其相应的分布为： $P(X = 0) = q, P(X = 1) = p$ 。这个分布称为伯努利分布（Bernoulli distribution），又名两点分布或者 0-1 分布。

[二项分布] 在 n 次伯努利分布中，若以变量 X 表示事件 A 出现的次数，则 X 的取值为 $\{0, \dots, n\}$ ，其相应的分布为：

$$P(X = k) = \binom{n}{k} p^k q^{n-k}, \quad k = 1 \cdots, n \quad (3.7)$$

这里， $\binom{n}{k}$ 为二项式系数（这就是二项分布的名称的由来），表示从 n 个元素中取出 k 个元素而不考虑其顺序的**组合**的总数。

连续随机变量

与离散随机变量不同，一些随机变量 X 的取值是不可列举的，由全部实数或者由一部分区间组成，比如

$$X = \{x | a \leq x \leq b\}, \quad -\infty < a < b < \infty$$

则称 X 为**连续随机变量**，连续随机变量的值是不可数及无穷尽的。

对于连续随机变量 X ，它取一个具体值 x_i 的概率为 0，这个离散随机变量截然不同。因此用列举连续随机变量取某个值的概率来描述这种随机变量不但做不到，也毫无意义。

我们一般用**密度函数** $p(x)$ 来描述连续随机变量 X 的概率分布。 $p(x)$ 为可积函数，并满足

$$\int_{-\infty}^{+\infty} p(x) dx = 1 \quad (3.8)$$

$$p(x) \geq 0. \quad (3.9)$$

给定密度函数 $p(x)$ ，便可以计算出随机变量落入某一个区间的概率，而 $p(x)$ 本身反映了随机变量取落入 x 的非常小的邻近区间中的概率大小。

下面看些常见的连续随机变量以及其概率分布的例子。

[**均匀分布**] 若 a, b 为有限数，有下面密度函数定义的分布称为 $[a, b]$ 上的均匀分布 (Uniform Distribution)。

$$p(x) = \begin{cases} \frac{1}{b-a} & , \quad a \leq x \leq b \\ 0 & , \quad x < a \text{ 或 } x > b \end{cases} \quad (3.10)$$

[**正态分布**] 正态分布 (Normal Distribution)，又名高斯分布，是自然界最常见的一种分布，并且具有很多良好的性质，在很多领域都有非常重要的影响力，其概率密度函数为

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (3.11)$$

其中， $\sigma > 0$ ， μ 和 σ 均为常数。若随机变量 X 服从一个位置参数为 μ 和 σ 的概率分布，简记为

$$X \sim N(\mu, \sigma^2). \quad (3.12)$$

当 $\mu = 0$ ， $\sigma = 1$ 时，称为**标准正态分布**。

图14.1a和14.1b分别显示了均匀分布和正态分布的密度函数。

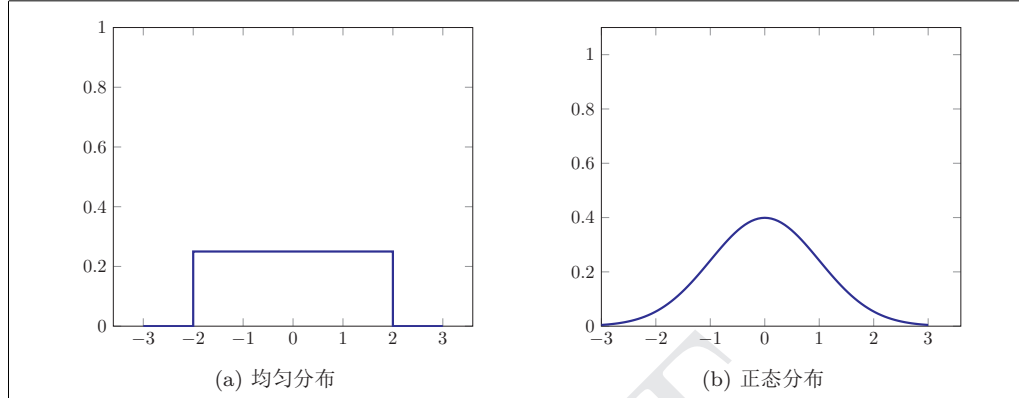


图 3.1: 连续随机变量的密度函数

3.1.4 随机向量

随机向量是指一组随机变量构成的向量。如果 X_1, X_2, \dots, X_n 为 n 个随机变量, 那么称 (X_1, X_2, \dots, X_n) 为一个 n 维随机向量。一维随机向量称为随机变量。

随机向量也分为离散随机向量和连续随机向量。

一般, 离散随机向量的(联合)概率分布为

$$P(X_1 = x_{i_1}, X_2 = x_{i_2}, \dots, X_n = x_{i_n}) = p(x_{i_1}, x_{i_2}, \dots, x_{i_n}), \quad (3.13)$$

$$x_{i_1}, x_{i_2}, \dots, x_{i_n} = 1, 2, \dots$$

和离散随机变量类似, 离散随机向量的概率分布满足

$$p(x_{i_1}, x_{i_2}, \dots, x_{i_n}) \geq 0, \quad x_{i_1}, x_{i_2}, \dots, x_{i_n} = 1, 2, \dots \quad (3.14)$$

$$\sum_{i_1} \sum_{i_2} \dots \sum_{i_n} p(x_{i_1}, x_{i_2}, \dots, x_{i_n}) = 1. \quad (3.15)$$

对连续型随机向量, 其(联合)密度函数满足

$$p(x_1, \dots, x_n) \geq 0, \quad (3.16)$$

$$\int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} p(x_1, \dots, x_n) dx_1 \dots dx_n = 1. \quad (3.17)$$

[多元正态分布] 若 n 维随机向量 $X = [X_1, \dots, X_n]^T$ 服从 n 元正态分布, 其密度函数为

$$p(x_1, \dots, x_n) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right), \quad (3.18)$$

其中, Σ 为正定对称矩阵; Σ^{-1} 表示 Σ 的逆阵; $|\Sigma|$ 表示 Σ 的行列式; 向量 $\mu = [\mu_1, \dots, \mu_N]^T$ 为列向量。

[多项分布] 在 n 次伯努利分布中, 若以变量 X 表示事件 A 出现的次数, 则 X 的取值为 $\{0, \dots, n\}$, 其相应的分布为:

$$P(X = k) = \binom{n}{k} p^k q^{n-k}, \quad k = 1 \dots, n \quad (3.19)$$

这里, $\binom{n}{k}$ 为二项系数 (这就是二项分布的名称的由来), 表示从 n 个元素中取出 k 个元素而不考虑其顺序的组的总数。

把二项分布推广到随机向量, 就得到了多项分布。假设一个袋子中装了很多球, 总共有 k 个不同的颜色。我们从袋子中取出 n 个球。每次取出一个球时, 就在袋子中放入一个同样颜色的球。这样保证每次取到同颜色的球的概率是相等的。我们定义一个 k 维随机向量 X , X_i 定义为取出的 n 个球中颜色为 i 的球的数量 ($i = 1, \dots, k$)。 θ_i 定义为每次抽取的球颜色为 i 的概率。 X 服从多项分布, 其概率分布为:

$$p(x_1, \dots, x_k | \theta) = P(X_1 = x_1, \dots, X_k = x_k | \theta_1, \dots, \theta_k) \quad (3.20)$$

$$= \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}, \quad (3.21)$$

其中, x_1, \dots, x_k 为非负整数, 并且满足 $\sum_{i=1}^k x_i = n$ 。

多项分布的概率分布也可以用 gamma 函数表示:

$$p(x_1, \dots, x_k | \theta) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \prod_{i=1}^k \theta_i^{x_i}. \quad (3.22)$$

从这种表示形式和 Dirichlet 分布类似, 而 Dirichlet 分布可以作为多项分布的共轭先验。

3.1.5 边际分布

为了方便起见, 我们下面以二维随机向量进行讨论, 多维时这些结论依然成立。对于二维离散随机向量 (X, Y) , 假设 X 取值为 x_1, x_2, \dots ; Y 取值为 y_1, y_2, \dots 。其联合概率分布满足

$$p(x_i, y_j) \geq 0, \quad \sum_{i,j} p(x_i, y_j) = 1. \quad (3.23)$$

对于联合概率分布 $p(x_i, y_j)$, 我们可以分别对 i 和 j 进行求和。

(1) 对于固定的 i ,

$$\sum_j p(x_i, y_j) = P(X = x_i) = p(x_i). \quad (3.24)$$

(2) 对于固定的 j ,

$$\sum_i p(x_i, y_j) = P(Y = y_j) = p(y_j). \quad (3.25)$$

也就是说, 由离散随机向量 (X, Y) 的联合概率分布, 对 Y 的所有取值进行求和得到 X 的概率分布; 而对 X 的所有取值进行求和得到 Y 的概率分布。这里 $p(x_i)$ 和 $p(y_j)$ 就称为 $p(x_i, y_j)$ 的**边际分布**。

对于二维连续随机向量 (X, Y) , 其边际分布为:

$$p(x) = \int_{-\infty}^{+\infty} p(x, y) dy \quad (3.26)$$

$$p(y) = \int_{-\infty}^{+\infty} p(x, y) dx \quad (3.27)$$

对于二元正态分布, 其边际分布仍为正态分布。

3.1.6 条件分布

对于离散随机向量 (X, Y) , 已知 $X = x_i$ 的条件下, 随机变量 $Y = y_j$ 的**条件概率**为:

$$P(Y = y_j | X = x_i) = \frac{P(X = x_i, Y = y_j)}{P(X = x_i)} = \frac{p(x_i, y_j)}{p(x_i)}. \quad (3.28)$$

这个公式定义了随机变量 Y 关于随机变量 X 的**条件分布**。

对于二维连续随机向量 (X, Y) , 已知 $X = x$ 的条件下, 随机变量 $Y = y$ 的**条件密度函数**为

$$p(y|x) = \frac{p(x, y)}{p(x)}. \quad (3.29)$$

同理, 已知 $Y = y$ 的条件下, 随机变量 $X = x$ 的**条件密度函数**为

$$p(x|y) = \frac{p(x, y)}{p(y)}. \quad (3.30)$$

通过公式3.29和3.29, 我们可以得到两个条件概率 $p(y|x)$ 和 $p(x|y)$ 之间的关系。

$$p(y|x) = \frac{p(y|x)p(y)}{p(x)} = \frac{p(y|x)p(y)}{\sum_y p(y|x)p(y)} \quad (3.31)$$

这个公式称为**贝叶斯公式**

3.1.7 随机变量的独立性

对于离散随机向量 (X, Y) ，如果其联合概率 $p(x_i, y_j)$ 满足

$$p(x_i, y_j) = p(x_i)p(y_j), \quad i, j = 1, 2, \dots \quad (3.32)$$

则称 X 和 Y 是相互独立的。

对于二维连续随机向量 (X, Y) ，如果其联合分布概率密度函数 $p(x, y)$ 满足

$$p(x, y) = p(x)p(y), \quad (3.33)$$

则称 X 和 Y 是相互独立的。

3.1.8 均值和方差

对于离散变量 X ，取值为 x_1, x_2, \dots ，对应的概率为 $p(x_1), p(x_2), \dots$ ，如果级数

$$\sum_{i=1}^{\infty} x_i p(x_i) \quad (3.34)$$

绝对收敛，则把它称为 X 的数学期望或均值，记为 $E(X)$ 。

如果 $\sum_{i=1}^{\infty} x_i p(x_i)$ 发散时，则说 X 的均值不存在。

若 $E((X - E(X))^2)$ 存在，则称它为随机变量 X 的方差，记为 $D(X)$ ， $\sqrt{D(X)}$ 则称为 X 的根方差或标准差。

3.2 统计语言模型

虽然语言的内在机制是建立的语法规则的基础上，但是对于自然语言（比如中文，英文），到目前为止还没有人能总结出一个完整的语法规则集合。统计语言模型将语言看出一个随机现象，然后借助概率统计的方法，来获取一种简化的语言规律。

自然语言在某种程度上也确实有很多随机因素。比如，当我们称赞一个人漂亮时，可以说“美丽”，“帅”或者“好看”等。当不指定使用场合时，这几个词可以交替使用。使用哪个词可以看作一个随机事件。假设其概率分别为：

- $p(\text{美丽}) = 0.1$
- $p(\text{帅}) = 0.1$
- $p(\text{好看}) = 0.1$

• ...

但是当称赞男性时，说“美丽”就不太合适了。这时就是要用到条件概率了。假设另外一个随机变量为称赞对象，取值为{男，女}。当给定称赞对象时，“美丽”，“帅”和“好看”的条件概率分别为：

- $p(\text{美丽}|\text{男}) = 0.01$
- $p(\text{帅}|\text{男}) = 0.2$
- $p(\text{好看}|\text{男}) = 0.12$
- $p(\text{美丽}|\text{女}) = 0.2$
- $p(\text{帅}|\text{女}) = 0.05$
- $p(\text{好看}|\text{女}) = 0.12$
- ...

同样地，我们也可以把一个句子（或任意字符串） s 看作一个随机事件，并赋予相应的概率来描述其属于某种语言集合的可能性。

但是，以句子为基本单位来讨论概率时，其样本空间为无限大（整个语言集合），很难去估计每个句子的概率。如果以词为基本单位，其样本空间为整个词典大小，是有限的，可以相对容易地估计每个词的概率。因此，我们需要对这个问题做下转换。

假设句子 s 由 m 个词（也可以是字、短语等）构成的一个序列 $s = w_1 w_2 \cdots, w_m = w_{1:m}$ 。这里并不要求句子 s 在语法上是完备的，我们需要对任意的字符串 s 都给出一个概率值。句子的概率可以看出是 m 维随机向量的概率。

$$P(s) = P(w_{1:m}) = P(w_1, \cdots, w_m) \quad (3.35)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \cdots P(w_m|w_{1:(m-1)}) \quad (3.36)$$

$$= P(w_1) \prod_{i=2}^m P(w_i|w_{1:(i-1)}) \quad (3.37)$$

$$= \prod_{i=1}^m P(w_i|w_{1:(i-1)}), \quad (3.38)$$

为了形式上简洁，公式3.38引进一个虚拟词 w_0 ，并且假定 $P(w_1|w_0) = p(w_1)$ 。这里， $P(s)$ 的定义不是很严格， $P(s)$ 应该是和句子的长度相关的。假设句子长度的概率分布为 $P(m)$ ，那么完整的分布函数为：

$$P(s) = P(n)P(w_{1:n}). \quad (3.39)$$

数学小知识 | 马尔可夫性质

马尔可夫性质是概率论中的一个概念。当一个随机过程在给定现在状态及所有过去状态情况下，其未来状态的条件概率分布仅依赖于当前状态；换句话说，在给定现在状态时，它与过去状态（即该过程的历史路径）是条件独立的，那么此随机过程即具有马尔可夫性质。

假设随机变量 X_1, \dots, X_3 构成一个数列。这些随机变量的所有可能取值的集合被称为“状态空间”，而 X_n 的值则是在时间 n 的状态。如果 X_{n+1} 对于过去状态的条件概率分布仅是 X_n 的一个函数，则 $P(X_{n+1} = x | X_0, X_1, X_2, \dots, X_n) = P(X_{n+1} = x | X_n)$ 。这里 x 为过程中的某个状态。上面这个恒等式可以被看作是马尔可夫性质。

马尔可夫链是数学中具有马尔可夫性质的离散时间随机过程。该过程中，在给定当前知识或信息的情况下，只有当前的状态用来预测将来，过去（即当前以前的历史状态）对于预测将来（即当前以后的未来状态）是无关的。

对于一种特定的语言，句子中的每个词 w_i 的样本空间为该语言的词典 V ，大小标记为 V 。这对于长度为 m 的句子 $w_{1:m}$ ，其样本空间为 V^m 。

从公式3.38可以看出，我们需要估计句子中每个词 w_i 在给定前面词序列 $w_{1:(i-1)}$ 时的条件概率。假如我们有一个很大的文档集合，就可以去估计这些概率。但是由于数据稀疏问题，我们很难估计所有的词序列。一个解决方法是我们假设一个词的概率只依赖于其前面的 $n-1$ 个词（ n 阶马尔可夫性质），即

$$P(w_i | w_{1:(i-1)}) = P(w_i | w_{(i-n+1):(i-1)}) \quad (3.40)$$

这就是 N 元（ N -gram）语言模型。当 $n=1$ 时，称为一元（unigram）语言模型，当 $n=2$ 时，称为二元（bigram）语言模型，以此类推。

利用统计语言模型，可以确定哪个词序列的可能性更大，或者给定若干个词的序列，可以预测下一个最可能出现的词语。

3.2.1 一元语言模型

在 N 元语言模型中，当 $n=1$ 时，相当于假设句子 $w_{1:m}$ 中每个位置上的词都和其他词是独立的，和它出现的上下文（邻居词）无关。也就是说，每个位置上的词都是从多项式分布 θ 独立生成的。这个多项式分布 θ 的维数为词典的大小 V 。

数学小知识 | 拉格朗日乘数

在数学最优化问题中，拉格朗日乘数法（以数学家约瑟夫·拉格朗日命名）是一种寻找变量受一个或多个条件所限制的多元函数的极值的方法。这种方法将一个有 n 个变量与 k 个约束条件的最优化问题转换为一个有 $n + k$ 个变量的方程组的极值问题，其变量不受任何约束。这种方法引入了一种新的标量未知数，即拉格朗日乘数：约束方程的梯度（gradient）的线性组合里每个向量的系数。如 f 定义为在 R^n 上的方程，约束为 $g_k(x) = c_k$ （或将约束左移得到 $g_k(x) - c_k = 0$ ）。定义拉格朗日函数为

$$\Lambda(\mathbf{x}, \boldsymbol{\lambda}) = f + \sum_k \lambda_k (g_k - c_k). \quad (3.45)$$

注意极值的条件和约束现在就都被记录到一个式子里了。

因此，句子 $w_{1:m}$ 的概率可以表示为：

$$P(w_{1:m} | \boldsymbol{\theta}) = \prod_{i=1}^m P(w_i) \quad (3.41)$$

$$= \prod_{k=1}^V \theta_v^{c_v}, \quad (3.42)$$

$$= P(c_{1:V} | \boldsymbol{\theta}) \quad (3.43)$$

其中， c_v 为字典中第 v 个词在句子中出现的次数。公式3.42可以看成是 V 维随机向量 c 的多项式分布，但是和一般多项式分布的区别是公式3.42 没有多项式系数，因为这里词的顺序是给定的。

一元语言模型可以看作是多个一元状态机的组合。

在公式3.42中， $\boldsymbol{\theta}$ 是需要实现估计出来的。我们需要从一个大量的文档集合中进行估计。

根据 Bayes 公式，

$$P(\boldsymbol{\theta} | c_{1:V}) = \frac{P(c_{1:V} | \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(c_{1:V})} \quad (3.44)$$

最大似然估计

假设先验 $P(\theta)$ 为均匀分布, θ 的最大似然估计为:

$$\theta^{ML} = \arg \max_{\theta} P(c_{1:V} | \theta) \quad (3.46)$$

$$= \arg \max_{\theta} \prod_{v=1}^V \theta_v^{c_v} \quad (3.47)$$

$$= \arg \max_{\theta} \sum_{v=1}^V c_v \log \theta_v \quad (3.48)$$

$$(3.49)$$

约束优化问题:

$$\max_{\theta_{1:V}} \sum_{k=1}^V c_k \log \theta_k \quad (3.50)$$

$$\text{subject to } \sum_{k=1}^V \theta_k = 1 \quad (3.51)$$

引入拉格朗日乘子 (Lagrange multiplier) λ , 定义拉格朗日函数 G 为

$$G(\theta_{1:V}, \lambda) = \sum_{k=1}^V c_k \log \theta_k - \lambda \left(\sum_{k=1}^V \theta_k - 1 \right) \quad (3.52)$$

这样, 原问题转换为无约束优化问题。

$$\frac{\partial G}{\partial \theta_k} = \frac{c_k}{\theta_k} - \lambda = 0 \quad (3.53)$$

$$\frac{\partial G}{\partial \lambda} = \sum_{k=1}^V \theta_k - 1 = 0 \quad (3.54)$$

进一步得到

$$\theta_k^{ML} = \frac{c_k}{\sum_{k=1}^V c_k} = \frac{c_k}{|C|}, \quad (3.55)$$

其中, $|C| = \sum_{k=1}^V c_k$ 为文档集合的长度。

由此可见, 在这里, 最大似然估计等价于频率估计。

然而，最大似然估计存在一个问题，**数据稀疏问题**：如果一个词 w 在语料库里不存在（假设 w 词典里的第 k 个词），那么 $c_k = 0$ ，并且相应的频率 $\theta_k = 0$ 。这就会导致任何包含 w 的句子的概率都为 0。

为了避免这个问题，一般可以通过下面途径改进语言模型：

1. 增加语料库规模；
2. 平滑技术；
3. 采用其他参数估计方法，比如下一个小节中介绍的最大后验估计方法。

最大后验估计

最大似然估计假设参数 θ 的先验分布是均匀分布，从而忽略了参数的先验分布。

最大后验估计（Maximum A Posteriori, MAP）是根据经验数据获得对难以观察的量的点估计。根据公式 3.44，

$$\theta^{MAP} = \arg \max_{\theta} P(\theta | c_{1:V}) \quad (3.56)$$

$$= \arg \max_{\theta} \frac{P(c_{1:V} | \theta) P(\theta)}{P(c_{1:V})} \quad (3.57)$$

$$= \arg \max_{\theta} P(c_{1:V} | \theta) P(\theta) \quad (3.58)$$

这里， $P(\theta)$ 为先验分布。为了方便计算，我们假设 θ 的先验分布为多项分布的共轭分布：Dirichlet 分布。

$$P(\theta | \alpha_{1:V}) = \frac{\Gamma(\alpha_1) \cdots \Gamma(\alpha_V)}{\Gamma(\alpha_1 + \cdots + \alpha_V)} \theta_1^{\alpha_1-1} \cdots \theta_V^{\alpha_V-1}, \quad (3.59)$$

其中， $\alpha_{1:V}$ 为正的超参数，函数 $\Gamma(\alpha)$ 为：

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx. \quad (3.60)$$

最大后验估计的参数为：

$$\theta_k^{MAP} = \frac{c_k + \alpha_k - 1}{|C| + \sum_{k=1}^V (\alpha_k - 1)}. \quad (3.61)$$

如果超参数 $\alpha_{1:V}$ 都不少于 1，即使第 k 个词在语料库中没有出现（ $c_k = 0$ ），该词的概率 θ_k^{MAP} 也不等于 0。假设我们设定 $\alpha_k = 2, k = 1, \dots, V$ ，我们得到所谓的加一平滑：

$$\theta_k^{MAP} = \frac{c_k + 1}{|C| + V}. \quad (3.62)$$

MAP 与 MLE 最大区别是 MAP 中加入了模型参数本身的概率分布，或者说。MLE 中认为模型参数本身的概率的是均匀的，即该概率为一个固定值。

3.2.2 n 元语言模型

一元语法模型在信息检索等领域有很广泛的应用，但是它忽略了词的顺序以及词之间的关系，假设每个词和句子中其他词是独立的。这个假设不符合自然语言的语法习惯，因此我们需要把这个假设弱化一下，一个词出现的概率只和它前面的 $n - 1$ 个词相关（ n 阶马尔可夫性质），即

$$P(w_i | w_{1:(i-1)}) = P(w_i | w_{(i-n+1):(i-1)}) \quad (3.63)$$

这样，一个句子 $w_{1:m}$ 的概率近似等于：

$$P(w_{1:m}) = \prod_{i=1}^m P(w_i | w_{1:(i-1)}) \quad (3.64)$$

$$\approx \prod_{i=1}^m P(w_i | w_{(i-n+1):(i-1)}) \quad (3.65)$$

公式3.65称为 n 元语言模型。

同样，通过最大似然估计，条件概率 $P(w_i | w_{(i-n+1):(i-1)})$ 可以从 n 元统计频率得到，

$$P(w_i | w_{(i-n+1):(i-1)}) = \frac{\text{count}(w_{(i-n+1):i})}{\text{count}(w_{(i-n+1):(i-1)})}, \quad (3.66)$$

其中， $\text{count}(w_{i:j})$ 为 $w_{i:j}$ 出现的次数。

公式??中， n 元模型的条件概率由频率估计得到，因此 n 元模型也会遇到采用最大似然估计的一元模型中同样的问题：数据稀疏问题。当一个 n 元组合在训练语言里不存在时，其概率为0。因此，为了避免这种情况发生，需要给一些没有出现的词组合赋予一定概率，这就统计语言模型的平滑技术，我们将在下节进行介绍。

3.3 平滑技术

3.4 统计语言模型的应用

统计语言模型广泛应用于各种自然语言处理问题，如语音识别、机器翻译、拼音输入法，字符识别 (OCR)，等等。简单地说，统计语言模型就是用来计算一个句子的概率的模型。

借助统计语言模型的概率参数，可以估计出自然语言中每个句子出现的可能性，而不是简单的判断该句子是否符合语法。在本章开头的例子3.1中，我们只需要计算将候选词随机的填写到每个空格中，形成很多填充后的句子，比如：

- s_1 = 这本书写得通俗易懂，实用性很强。
- s_2 = 这本通俗易懂写得书，实用性很强。
- s_3 = 这本书写得实用性，通俗易懂很强。
- ...

然后计算这些句子的概率 $p(s_1), p(s_2), \dots$ 。概率最大的句子就是正确的答案。

3.4.1 拼音输入法

简体中文大约有 400 个合法的拼音，GBK 汉字大约有 18000 个汉字，那么平均每个拼音对应 45 个汉字。

由于每个拼音都有很多个转换结果，那么对于上面的拼音序列（8 个拼音）约有 45^8 种转换结果，我们希望给出一个对用户来说最好的转换结果。举个音字转换的例子来说，输入拼音串为 “*xie'zuo'ye*”，对应的输出可以有多种形式，如“写作业”、“写昨夜”等等，那么到底哪个才是正确的转换结果呢，利用语言模型，我们知道前者的概率大于后者，因此转换成前者在多数情况下比较合理。

语言模型是根据你的输入习惯自动更新的，也就是说，你的输入是按字打，还是按词打，会严重的影响到语言模型的更新。

3.4.2 统计机器翻译

再举一个机器翻译的例子，给定一个汉语句为“我正在家里看电视。”，按照词或短语进行对位翻译可以得到：

- I am watching TV at home.
- I am at home and watching TV.
- I am seeing TV at home.

等等。同样根据语言模型，我们知道第一句的概率大于后几句，所以最终翻译成 “I am watching TV at home.” 比较合理。

3.4.3 编码识别

3.5 概率统计语言模型的不足

那么如何计算一个句子的概率呢？给定句子（词语序列）

他很喜欢榴莲。当他看到榴莲就很——他很讨厌榴莲。当他看到榴莲就很——

1. 高兴 2. 难过。

参考文献和深入阅读

Beyond n-gram type language models, a variety of models have been proposed to try to “put language back into language model”, including class-based LM, tree LM, grammar-based LM, Maximum entropy LM, whole sentence LM and so on. However, trigram language model is surprisingly hard to beat.

第四章 文本分类： 机器学习方法的简单实践

机器学习是对能通过经验自动改进的计算机算法的研究。

— Mitchell [1997]

文本分类，也叫文档分类，是指在给定一个分类体系下，根据文本内容来判断文本的相应类别。文本分类是文本挖掘的一个重要内容。20世纪90年代以前，文本分类由专业人员手工进行分类，非常费时，效率也很低。之后，机器学习方法被大量应用于自动文本分类，并在信息检索、网页分类、数字图书馆等很多涉及文档组织和管理的领域中得到了广泛的应用。

本章通过简单的文本分类任务来描述机器学习算法的基本概念以及在自然语言处理中的实践。

4.1 机器学习的基本概念

机器学习是从观测数据（样本）中寻找规律，或模拟人类的学习行为，并利用学习到的规律（模型）对未知或无法观测的数据进行预测。

狭义地讲，机器学习是给定一些观测样本 $(x_i, y_i), 1 \leq i \leq N$ （其中 x_i 是样本， y_i 是相应的类别），让计算机自动寻找一个模型 $\hat{y} = f(\phi(x))$ ，对于所有已知或未知的 (x, y) ，使得 \hat{y} 和 y 尽可能地一致。这里 $\phi(x)$ 表示样本 x 对应的特征表示。机器学习系统的示例见图4.1。

机器学习可以分为下面几类：

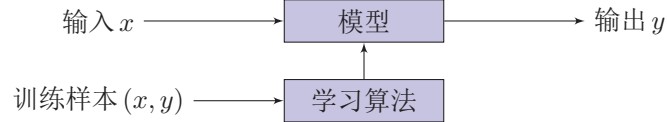


图 4.1: 机器学习系统示例

有监督学习 有监督学习是利用一组已知输出标记的数据来学习模型的参数，使得模型预测的输出标记和真实标记尽可能的一致。如果输出 y 是离散的类别标记，就是分类问题。通过学习得到的模型也叫**分类器**。如果 y 是连续的值，就是回归问题。

无监督学习 无监督学习是用来学习的数据不包含输出标记，需要学习算法自动学习到一些有价值的信息。聚类就是一个典型的无监督学习问题。关于聚类问题，我们将在下一章进行介绍。

有监督的学习方法需要每个数据记录都有类标号，而无监督的学习方法则不考虑任何指导性信息。一般而言，一个监督学习模型需要大量的有标记数据集，而这些数据集是需要人工标注的。因此，也出现了很多弱监督学习和半监督学习的方法，希望从大规模的未标记数据中充分挖掘有用的信息，降低对标记数据数量的要求。

上述的关于机器学习的介绍中，提及了一些基本概念，比如“数据”，“样本”，“特征”，“数据集”，“分类器”，“学习算法”，“评价方法”等。我们首先来解释下这些概念。

4.1.1 数据

在计算机科学中，**数据**是指所有能计算机程序处理的对象的总称，可以是数字、字母和符号等。在不同的任务中，表现形式不一样。比如在文本分类中，数据可以是一篇文档，也可以是一个句子。

样本是按照一定的抽样规则从全部数据中取出的一部分数据，是实际观测得到的数据。我们用 x 表示抽象概念的样本，可以有多种表示形式，比如字符串、数组、集合或者特征空间中一个点等。

在有监督学习中，需要提供一组有类别标记的样本用来学习模型以及检验模型的好坏。这组样本集合就称为**数据集**。数据集用 X 表示， $X = \{(x_1, y_1), \dots, (x_N, y_N)\}$ ， N 为样本个数。 y_i 为样本实例 x_i 的类别标记。在自然语言处理中，数据集也经常称为**语料库**。

对于类别数为 2 的数据，类别可以表示为 $\{+1, -1\}$ ，或者直接用正负号表示。因此，常用**正例**和**负例**来分别表示属于不同类别的样本。

为了检验机器学习算法的好坏，一般将数据集分为两部分：训练集和测试集。训练集用来进行模型学习，测试集用来进行模型验证。

机器学习算法是在给定训练集 $(x_i, y_i), 1 \leq i \leq N$ 上, 让计算机自动寻找一个模型 $\hat{y} = f(\phi(x))$, 使得对于每一个样本 x_i , \hat{y}_i 和 y_i 尽可能地一致。

通过学习算法, 在训练集得到一个模型, 这个模型可以对测试集上样本 x 预测一个类别标签 \hat{y} 。假设测试集为 T , 模型的正确率为:

$$Acc = \frac{1}{|T|} \sum_{(x_i, y_i) \in T} |\hat{y}_i = y_i|, \quad (4.1)$$

其中 $|T|$ 为测试集的大小。第4.3节中会介绍更多的评价方法。

4.1.2 特征

在机器学习算法中, 样本实例一般是以连续变量或离散变量的形式存在的 (也称为特征), 而在自然语言处理中一般都是文字形式。因此, 在进行学习模型之前, 需要进行特征提取, 将样本的原始表示转换为特征向量。

在机器学习中, 为了更好地表示样本的属性, 一般将样本表示成代数形式, 称为样本的**特征**, 我们用 $\phi(x)$ 。样本特征可以是一维或多维向量, $\phi(x) \in \mathbb{R}^m$, m 是向量维数。

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_k(x) \end{bmatrix} \quad (4.2)$$

在自然语言处理中, 数据都是以字符形式存在的。样本的原始表示一般是字符串序列。为了便于使用机器学习方法, 首先要把样本表示为向量形式。下面我们介绍几种常用的特征表示方法。自然语言处理的, 在构造了样本和样本集合之后, 为了和后面的机器学习算法相结合, 我们将样本 x 转变成向量 $\phi(x)$ 。在将字符表示转换成向量表示的过程中需要很多中间步骤, 我们把这些中间步骤都成为**数据处理**, 并且尽可能的模块化。

词袋模型

一种简单的方法是简单假设文本 (如一个句子或一个文档) 是由字、词组成的无序多重集合, 不考虑语法甚至词序¹。这就是在自然语言处理和信息检索中常用的**词袋模型** (Bag of Words, BOW)²。

¹ 多重集合是数学中的一个概念, 是集合概念的推广。在一个集合中, 相同的元素只能出现一次, 因此只能显示出有或无的属性。在多重集之中, 同一个元素可以出现多次。

² 词袋模型可以看成一种以词为基本单位的**向量空间模型** (Vector Space Model, VSM)。

数学小知识 | 向量

在线性代数中，向量是指 n 个实数组成的有序数组，称为 n 维向量。如果没有特别说明，一个 n 维向量一般表示列向量，即大小为 $n \times 1$ 的矩阵。

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix},$$

其中， a_i 称为向量 \mathbf{a} 的第 i 个分量，或第 i 维。

为简化书写、方便排版起见，有时会以加上转置符号 T 的行向量（大小为 $1 \times n$ 的矩阵）表示列向量。

$$\mathbf{a} = [a_1, a_2, \dots, a_n]^T$$

向量符号一般用黑体小写字母 $\mathbf{a}, \mathbf{b}, \mathbf{c}$ 或希腊字母 α, β, γ 等来表示。

向量 \mathbf{a} 的模 $\|\mathbf{a}\|$ 为

$$\|\mathbf{a}\| = \sqrt{\sum_{i=1}^n a_i^2}.$$

给定一组文档 $D = \{D_1, D_2, \dots, D_N\}$ ，假设这组文档里总共包含 M 个不同的词 t_1, \dots, t_M 。每一个文档 D_i 可以表示为一个 M 维的向量：

$$D_i = [d_{i1}, d_{i2}, \dots, d_{iM}]^T, \quad (4.3)$$

其中 $d_{im} \in \{0, 1\}$ 表示第 m 个词在文档 D_i 是否出现。

比如，两个文本 D_1 和 D_2 分别为：

D_1 ：“我喜欢读书”

D_2 ：“我讨厌读书”

这两个文本总共包含 4 个不同的词，用 0 或 1 表示这 4 个词在 D_1 和 D_2 是否出现， D_1 和 D_2 可以分别表示为 $[1, 1, 0, 1]^T$ 和 $[1, 0, 1, 1]^T$ 。如下所示。

	我	喜欢	讨厌	读书
D_1	1	1	0	1
D_2	1	0	1	1

除了用是否出现来表示 d_{ij} 的值外，还可以有其他方法来对一个文本中不同词的权重进行赋值。比较常用的赋值为**词频-逆向文档频率**（Term Frequency - Inverse Document Frequency, TF-IDF）。TF-IDF 是一种统计方法，用以评估词（Term）对于文本集中的某一个文档的重要程度。词的权重跟它在文本中出现的次数成正比，与包含该词的文档数量成反比。TF-IDF 加权的各种形式常被搜索引擎、文本分类等应用，可以用来衡量两个文档之间相关程度。

TF-IDF 其主要思想是，如果某个词在一个文档中出现的频率高，并且在其他文档中很少出现，则认为该项具有很好的类别区分能力，适合用来分类。

词频（Term Frequency, TF） $tf_{i,m}$ 表示文档 i 中词 m 出现的频率，计算公式如下：

$$tf_{i,m} = \frac{n_{i,m}}{\sum_k n_{k,m}} \quad (4.4)$$

其中， $n_{i,m}$ 表示词 m 在文档 i 中出现的次数，分母则是所有文档中该词出现的次数之和。

逆向文档频率（Inverse Document Frequency, IDF） idf_i 是衡量一个词的区分性，由下面的式子计算：

$$idf_m = \log \frac{|D|}{|d : d \ni t_m|} \quad (4.5)$$

其中， $|D|$ 是文档总数，分母是包含词 t_m 的文档数目。

在文档 D_i 中，第 m 个词的权重为：

$$tf-idf_{i,m} = tf_{i,m} * idf_m. \quad (4.6)$$

文档 D_i 的向量可以表示为：

$$D_i = [tf-idf_{i,1}, \dots, tf-idf_{i,M}]^T. \quad (4.7)$$

通过词袋模型，每一文档都被映射为由 M 维的向量。这种表示的优点在于将非结构化和半结构化的文本表示为向量形式，使得后续可以更便捷地使用各种机器学习方法。但是，词袋模型将文本仅仅看作词的集合，这种假设虽然对文本表示进行了简化，但在不需要深层分析的应用（比如信息检索和文本分类）来说，也有一定的合理性，也便于模型化。比如在文本分类中，如果我们观察“体育”和“科技”两个类别的新闻网页，会发现这两类网页中词汇的分布是有明显区别的。在“体育”类的网页中经常会出现“比赛”、“胜利”、“冠军”等词，而“科技”类网页中经常会出现“电脑”、“手机”、“产品”等词。

在词袋模型中，这种向量表示的特征通常都是非常稀疏的。一般一篇文档中包含的词只占全部词典的一小部分，其对应的向量表示中大部分词条对应的维数上的值都为0。因此，这些向量一般都是用稀疏向量来表示。

N 元特征

词袋模型在需要深层分析的场合就会显得太过简化了。例如在语义分析里，“你打了我”和“我打了你”，意思是相反的，但用词袋模型表示后，这两句话是向量表示的等价的，这显然是不合理的。

N 元特征（N-gram 特征），顾名思义，就是由 N 个字或词组成的字符串，单元可以是字或词。这里 N 是大于等于1的任意整数。如果 N 为2，就叫做二元特征，如果 N 为3，就叫做三元特征以此类推。

N 元特征可以看出是对词袋模型的一种改进方法。与 N 元特征相关的概念是 **N 元语法模型**。N 元语法模型 (N-gram Model) 一种统计方法，假定一个字只与前面的 $N - 1$ 个字有关，相当于 $N - 1$ 阶的马尔可夫模型，在信息检索、语音识别和机器翻译中被广泛使用。

以中文句子“机器学习算法”为例，以字为基本单位的二元特征集合为： $\{\text{机器, 器学, 学习, 习算, 算法}\}$ 。集合中每一项都是由二个相邻的字组成的子串，长度为2。这些子串可以是有意义的词（例如：“学习”、“算法”），也可以是无任何意义的字符串（例如：“器学”，“习算”）。但是这些无意义的子串也有可能分类中起到很关键的作用。一个长度为 L 的句子，可以提取出 $L - 1$ 个二元特征。

在很多时候，一个字在句子中的位置信息对某些分类任务来讲也非常重要，特别是一个句子的开始或结束位置。我们可以增加两个特殊的标记 #, \$ 分别来表示句子的开始的结束。这样就增加了两个新的二元特征 “# 机”，“算 \$”。

由于在中文里字和字之间没有空格，因此以字为基本单位的 N 元特征很好地回避了中文分词问题。当然，也可以先进行分词，然后构建以词为基本单位的 N 元特征。这样，上面的句子先分词为“机器 学习 算法”，对应的以词为基本单位的二元特征集合为：{机器学习，学习算法}。

有了 N 元特征集合，就可以利用词袋模型将文本表示为向量形式。随着 N 的增加，可以抽取的特征就会越多，特征空间也会呈指数增加。这些高阶的特征出现的频率也会相对较低，对分类不但没有太多帮助，还会直接影响着后续处理的效率与复杂度。因此在一般的文本分类任务中， N 取 3 就足够了，并且同时也使用一元和二元特征，防止出现过拟合（过拟合的定义见第 54 页）。

4.1.3 判别函数

经过特征抽取后，一个样本可以表示为 k 维特征空间中的一个点。为了对这个特征空间中的点进行区分，就需要寻找一些超平面来将这个特征空间分为一些互不重叠的子区域，使得不同类别的点分布在不同的子区域中，这些超平面就成为判别界面。

为了定义这些用来进行空间分割的超平面，就需要引入判别函数的概念。假设变量 $\mathbf{z} \in \mathbb{R}^m$ 为特征空间中的点，这个超平面由所有满足函数 $f(\mathbf{z}) = 0$ 的点组成。这里的 $f(\mathbf{z})$ 就称为判别函数。

有了判别函数，分类就变得很简单，就是看一个样本在特征空间中位于哪个区域，从而确定这个样本的类别。

判别函数的形式多种多样，在自然语言处理中，最为常用的判别函数为线性函数。

两类分类问题

首先对于两类分类问题，假设类别 $y \in \{1, -1\}$ ，相应的线性判别函数为

$$f(\mathbf{z}) = \boldsymbol{\theta}^T \mathbf{z} + \theta_0, \quad (4.8)$$

其中， $\boldsymbol{\theta}$ 为权重向量， θ_0 为偏置或阈值权重。

公式 4.8 定义了一个两类分类问题的线性判别函数。在高维的特征空间中，所有满足 $f(\mathbf{z}) = 0$ 的点组成用一个超平面，这个超平面将特征空间一分为二，划分成两个区域。这两个区域分别对应两个类别。

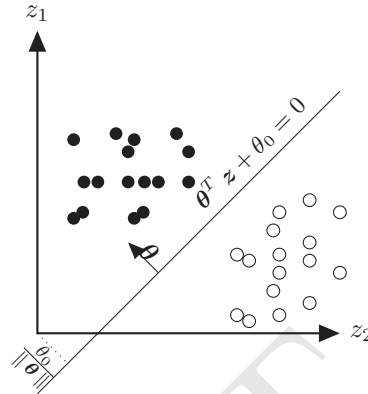


图 4.2: 二维特征空间中两类分类的分类超平面示例

图4.2中给了一个两维数据的判别函数以及对应的判别界面。在二维空间中，分类界面为一个直线。在三维空间中，分类界面为一个平面。在高维空间中，分类界面为一个超平面。对于线性函数来说，权重向量在线性空间中垂直于分类界面的向量。

判别函数还不是分类器。在判别函数的基础上，比如可以使用这样的决策：如果 $f(z) > 0$ ，类别 $y = +1$ ；如果 $f(z) < 0$ ，类别为 $y = -1$ 。

$$\hat{y} = \begin{cases} +1 & \text{if } f(z) > 0 \\ -1 & \text{if } f(z) < 0 \end{cases}, \quad (4.9)$$

这里， \hat{y} 为根据判别函数预测的类别。

公式4.9定义了一个两类分类问题的线性分类器。也可以写为

$$\hat{y} = \text{sign}(f(z)) = \text{sign}(\theta^T z + \theta_0), \quad (4.10)$$

这里，**sign** 是符号函数，取判别函数 $f(z)$ 的正负号。

为了表示方便，我们简写判别函数4.8为：

$$f(z) = \theta^T z + \theta_0 = \sum_{i=1}^k \theta_i z_i + \theta_0 = \sum_{i=0}^k \theta_i z_i = \hat{\theta}^T \hat{z}, \quad (4.11)$$

其中， $z_0 = 1$ ， $\hat{\theta}$ 和 \hat{z} 分别称为增广权重向量和增广特征向量。

$\hat{\mathbf{z}}$ 为

$$\hat{\mathbf{z}} = \begin{bmatrix} 1 \\ z_1 \\ \vdots \\ z_k \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{z} \end{bmatrix}, \quad (4.12)$$

$\hat{\boldsymbol{\theta}}$ 为

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_k \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \boldsymbol{\theta} \end{bmatrix}. \quad (4.13)$$

在后面的分类器描述中，我们都采用简化的表示方法，并直接用 $\boldsymbol{\theta}$ 和 \mathbf{z} 来表示增广权重向量和增广特征向量。

$$f(\mathbf{z}) = \boldsymbol{\theta}^T \mathbf{z}. \quad (4.14)$$

多类分类问题

对于多类分类问题（假设类别数为 $C(C > 2)$ ），有很多种利用判别函数的方法。比较常用的是把多类分类问题转换为两类分类问题，在得到两类分类结果后，还需要通过投票方法进一步确定多类的分类结果。一般有两种多类转两类的转换方式：

1. 把多类分类问题转换为 C 个两类分类问题，构建 C 个一对多的分类器。每个两类分类问题都是把某一类和其他类用一个超平面分开。
2. 把多类分类问题转换为 $C(C-1)/2$ 个两类分类问题，构建 $C(C-1)/2$ 个两两分类器。每个两类分类问题都是把 C 类中某两类用一个超平面分开。

当对一个样本进行分类时，首先用多个两个分类器进行分类，然后进行投票，选择一个得分最高的类别。

但是上面两种转换方法都存在一个缺陷：空间中的存在一些区域，这些区域中点的类别是不能区分确定的。因为如果用多个两个分类器对这些点进行分，然后进行投票时，会发现有两个或更多个类别的得分是一样的。

为了避免上述缺陷，可以使用一个更加有效的决策规则，直接建立多类分类器。假设 $y = \{1, \dots, C\}$ 共 C 个类别，首先定义 C 个判别函数：

$$f_c(\mathbf{z}) = \boldsymbol{\theta}_c^T \mathbf{z}, \quad c = 1, \dots, C, \quad (4.15)$$

这里 θ_i 为类 i 的增广权重向量。

这样，对于空间中的一个点 z ，如果存在类别 i ，对于所有的其他类别 $j(j \neq i)$ 都满足 $f_i(z) > f_j(z)$ ，那么 z 属于类别 i 。相应的分类函数可以表示为：

$$\hat{y} = \arg \max_{c=1}^C f_c(z). \quad (4.16)$$

两个分类为多类分类的特殊情况。

$$\begin{aligned} \hat{y} &= \arg \max_{y \in \{+1, -1\}} f_y(z) \\ &= \text{sign}(f_{+1}(z) - f_{-1}(z)) \\ &= \text{sign}\left(\theta_{+1}^T z - \theta_{-1}^T z\right) \\ &= \text{sign}\left((\theta_{+1} - \theta_{-1})^T z\right) \\ &= \text{sign}(f'(z)), \end{aligned} \quad (4.17)$$

这里， $f'(z) = (\theta_{+1} - \theta_{-1})^T z$ 为相应的判别函数。对比两类分类的判别函数4.14，可以发现 $\theta = \theta_{+1} - \theta_{-1}$ 。

其他判别函数形式

判别函数的形式也可以为其他复杂的函数，比如可以表示为概率形式，

$$f(\phi(x)) = \arg \max_{y=1}^C p(y | \phi(x)). \quad (4.18)$$

4.1.4 学习算法

为了进行分类，就需要知道不同类别的判别界面。然而，这些界面是事先不知道的。机器学习的本质就是从很多已经有标记类别的样本中，自动学习出这个判别界面。更具体地讲，就是在给定判别函数（比如公式4.8和4.15）后，自动学习参数 $\theta_i (i = 1, \dots, C)$ 的值。

不同机器学习算法的区别在于其判别函数和学习算法的差异。这些方法是可以是相同的判别函数，但是学习算法不同。比如对于判别函数4.8，也就是线性分类器，其参数的学习算法可以是感知器、支持向量机以及最小平方误差等。

通过一个学习算法进行自动学习参数的过程也叫作训练过程。

4.2 感知器

本节以感知器为例来讲解分类器的原理以及训练过程。这里，我们直接考虑多类分类问题。

感知器是一个经典的线性分类器学习算法，也是最简单的神经网络（只有一层）。

感知器的判别函数为线性函数。对于一个 C 类分类问题，假设样本为 x ， $\phi(x)$ 为增广特征向量，类别标签集合为 $\{1, \dots, C\}$ ，感知器的分类函数可以写为

$$\hat{y} = \arg \max_{c=1}^C \theta_c^T \phi(x). \quad (4.19)$$

分类函数4.19的权重向量是未知的，需要从给定的训练数据集中学习得到。这样，当新的数据到来时，可以根据这个函数预测结果。

接下来我们看下感知器是如何学习的。

4.2.1 感知器学习算法

给定 N 个样本的训练集: $(x_i, y_i), i = 1, \dots, N$ 。多类感知器的学习过程如算法4.1 所示。

4.2.2 多类感知器的另一种表示形式

在上面几节中，判别函数都是在特征空间定义的。对于 C 类分类问题，需要定义 C 个判别函数。但是这种表示一般适用于类别 y 为离散变量的情况。在自然语言处理的很多学习任务，类别 y 可以是更复杂的结构，比如多标签、层次化以及结构化等形式。为了更好的描述这些情况，公式4.19 可以改写为如下形式：

$$\hat{y} = \arg \max_{y=1}^C \theta^T \phi(x, y), \quad (4.20)$$

这里 $\phi(x, y)$ 是包含了样本 x 和类别 y 混合信息的特征向量， $\theta = [\theta_1; \dots, \theta_C]$ 。

算法 4.1: 多类感知器算法

输入: 训练集: $(x_i, y_i), i = 1, \dots, N$, 迭代次数: T

输出: θ

```

1  $\theta_c \leftarrow 0, c = 1, \dots, C$ ;
2  $k = 1$ ;
3 for  $t = 1 \dots T$  do
4   for  $i = 1 \dots N$  do
5     选取一个样本  $(x_i, y_i)$ , 其特征为  $\phi(x_i)$ ;
6     通过公式4.19计算出预测的类别  $\hat{y}_i$ ;
7     if  $\hat{y}_i \neq y_i$  then
8        $\theta_{y,k} = \theta_{y,k-1} + \phi(x_i)$ ;
9        $\theta_{\hat{y},k} = \theta_{\hat{y},k-1} - \phi(x_i)$ ;
10       $k = k + 1$ ;
11   end
12 end
13 end
14 return  $\theta_k$ ;
```

当 y 为离散变量时 ($y = \{1, \dots, C\}$), 类别也可以表示为向量:

$$\phi(y = c) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boxed{1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{第 } c \text{ 行}$$

如果每个样本可以有多个类别, 标签分类 $y = \{c, k\}$ 时, 类别可以表示为向量:

数学小知识 | 向量化算子

设 $A = [a_{ij}]_{m \times n}$, 则

$$\text{vec}(A) = [a_{11}, a_{21}, \dots, a_{m1}, a_{12}, a_{22}, \dots, a_{m2}, \dots, a_{1n}, a_{2n}, \dots, a_{mn}]^T.$$

$$\phi(y = \{c, k\}) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{array}{l} \leftarrow \text{第 } c \text{ 行} \\ \leftarrow \text{第 } k \text{ 行} \end{array}$$

$\phi(x, y)$ 可以看成是 $\phi(x)$ 和 $\phi(y)^T$ 的乘积得到矩阵的向量化。

$$\phi(x, y = c) = \text{vec}(\phi(x) \phi(y)^T) \quad (4.21)$$

$$\phi(x, y = c) = \begin{bmatrix} \vdots \\ 0 \\ \phi(x) \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ \phi_1(x) \\ \vdots \\ \phi_m(x) \\ 0 \\ \vdots \end{bmatrix} \quad \begin{array}{l} \leftarrow \text{第 } c * m + 1 \text{ 行} \\ \leftarrow \text{第 } c * m + m \text{ 行} \end{array} \quad (4.22)$$

多类感知器算法的训练过程如算法4.3所示。

算法 4.2: 多类感知器算法的另一种表示形式

输入: 训练集: $(x_i, y_i), i = 1, \dots, N$, 最大迭代次数: T

输出: θ_k

```

1  $\theta_0 = 0$  ;
2  $k = 1$  ;
3 for  $t = 1 \dots T$  do
4   for  $i = 1 \dots N$  do
5     选取一个样本  $(x_i, y_i)$ ;
6     用公式4.20计算预测类别  $\hat{y}_t$ ;
7     if  $\hat{y}_t \neq y_t$  then
8        $\theta_k = \theta_{k-1} + (\phi(x_t, y_t) - \phi(x_t, \hat{y}_t))$ ;
9        $k = k + 1$  ;
10    end
11  end
12 end
13 return  $\theta_T$  ;
```

4.2.3 收敛性证明

定义 4.1 – 两类线性可分: 对于训练集 $\mathcal{D} = \{(x_i, y_i) \mid y_i \in \{-1, 1\}\}_{i=1}^n$, 如果存在一个正的常数 $\gamma (\gamma > 0)$ 和权重向量 θ^* , 并且 $\|\theta^*\| = 1$, 对所有 i 都满足 $y_i \langle \theta^*, \phi(x_i) \rangle > \gamma$ ($\phi(x_i) \in \mathbb{R}^m$ 为样本 x_i 的增广特征向量), 那么训练集 \mathcal{D} 是线性可分的。

Novikoff 证明对于两类问题, 如果训练集是线性可分的, 那么感知器算法可以在有限次迭代后收敛, 迭代次数最多 $\left(\frac{2R}{\gamma}\right)^2$, 其中 R 为输入向量的最大平均值。然而, 如果训练集不是线性分隔的, 那么这个算法则不能确保会收敛。

下面我们来证明多类分类时, 感知器的收敛情况。

定义 4.2 – 多类线性可分: 对于训练集 $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, 如果存在一个正的常数 $\gamma (\gamma > 0)$ 和权重向量 θ^* , 并且 $\|\theta^*\| = 1$, 对所有 i 都满足 $\langle \theta^*, \phi(x_i, y_i) \rangle - \langle \theta^*, \phi(x_i, y) \rangle > \gamma, y \neq y_i$ ($\phi(x_i, y_i) \in \mathbb{R}^m$ 为样本 x_i 的增广特征向量), 那么训练集 \mathcal{D} 是线性可分的。

定理 4.1 – 感知器收敛性： 对于任何线性可分的训练集 $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ ，假设 R 是所有样本中错误类别和真实类别在特征空间 $\phi(x, y)$ 最远的距离。

$$R = \max_i \max_{z \neq y_i} \|\phi(x_i, y_i) - \phi(x_i, z)\|$$

那么在感知器学习算法4.3中，总共的预测错误次数 $K < \frac{R^2}{\gamma^2}$ 。

证明：

权重向量的更新公式为：

$$\theta_k = \theta_{k-1} + \left(\phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right), \quad (4.23)$$

这里， $(x^{(k)}, y^{(k)})$ 为第 k 个错误分类的样本。

因为初始权重向量为0，因此在第 K 次更新时，

$$\theta_K = \sum_{k=1}^K \left(\phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right), \quad (4.24)$$

$\|\theta_K\|^2$ 的上界为：

$$\begin{aligned} \|\theta_K\|^2 &= \left\| \sum_{k=1}^K \left(\phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right) \right\|^2 \\ &\leq \sum_{k=1}^K \left\| \phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right\|^2 \\ &\leq \sum_{k=1}^K \max_{k=1}^K \left\| \phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right\|^2 \\ &\leq K \times R^2 \end{aligned} \quad (4.25)$$

下面计算 $\|\theta_K\|^2$ 的下界。

首先，因为两个向量内积的平方一定小于等于这两个向量的模的乘积。

$$\|\theta^* \times \theta_K\|^2 \leq \|\theta^*\|^2 \times \|\theta_K\|^2 = \|\theta_K\|^2 \quad (4.26)$$

$$\|\theta^* \times \theta_K\|^2 = \left\| \theta^* \times \sum_{k=1}^K \left(\phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right) \right\|^2$$

机器学习 | 泛化错误、过拟合

泛化错误是衡量一个机器学习模型是否可以很好地泛化到未知数据。泛化错误一般表现为一个模型在训练集和测试集上错误率的差距。泛化错误产生的原因是用来训练样本的训练集是真实数据的一个很小的子集或者存在噪声数据，不能很好地反映全部数据的真实分布。

过拟合是为在训练集上得到更好的模型而使模型变得过度复杂，导致泛化能力比较差。过拟合问题往往是由于训练数据少和噪声等原因造成的。过拟合的标准定义为：给定一个假设空间 H ，一个假设 h 属于 H ，如果存在其他的假设 h' 属于 H ，使得在训练样例上 h' 的错误率比 h 小，但在整个实例分布上 h 比 h' 的错误率小，那么就说明假设 h 过度拟合训练数据。—[Mitchell, 1997]

$$\begin{aligned} &= \left\| \sum_{k=1}^K \left(\theta^* \times \left(\phi(x^{(k)}, y^{(k)}) - \phi(x^{(k)}, \hat{y}^{(k)}) \right) \right) \right\|^2 \\ &\geq K^2 \gamma^2 \end{aligned} \quad (4.27)$$

由公式4.26和4.27，得到

$$\|\theta_K\|^2 \geq \|\theta^* \times \theta_K\|^2 \geq K^2 \gamma^2 \quad (4.28)$$

由公式4.25和4.28，得到

$$K^2 \gamma^2 \leq \|\theta_K\|^2 \leq K \times R^2 \quad (4.29)$$

取最左和最右的两项，进一步得到， $K^2 \gamma^2 \leq K \times R^2$ 。然后两边都除 K ，最终得到

$$K \leq \frac{R^2}{\gamma^2}. \quad (4.30)$$

4.2.4 权重平均化的感知器

从定理4.1可以看出，如果训练数据是线性可分的，那么感知器可以找到一个判别函数来分割不同类的的数据。并且如果边际距离越大，收敛越快。但是，感知器并不能保证找到的判别函数是最优的（比如泛化能力高），这样可能导致过拟合。

此外，从感知器的迭代算法可以看出：在迭代次序上排在后面的错误点，比前面的错误点对最终的权重向量影响更大。比如有1,000个训练样本，在迭代100个样本后，感知器已经学习到一个很好的权重向量。在接下来的899个样本上都预测正确，也没有更新权重向量。但是在最后第1,000个样本时预测错误，并更新了权重。这次更新可能反而使得权重向量变差了。

为了这种情况，可以“参数平均”的策略。

假设 $\theta^{t,i}$ 是在第 t 轮更新到第 i 个样本时权重向量的值，平均化的权重向量为

$$\bar{\theta} = \frac{\sum_{t=1}^T \sum_{i=1}^n \theta^{t,i}}{nT} \quad (4.31)$$

这个方法非常简单，只需要在算法4.3中增加一个 $\bar{\theta}$ ，并且在处理每一个样本后，更新

$$\bar{\theta} = \bar{\theta} + \theta^{t,i} \quad (4.32)$$

这里要注意的是， $\bar{\theta}$ 需要在处理每一个样本时都需要更新，并且 $\bar{\theta}$ 和 $\theta^{t,i}$ 都是稠密向量。因此，这个操作比较费时。为了提高迭代速度，有很多改进的方法，让这个更新只需要在错误预测发生时才进行更新。

算法4.3给出了一个改进的平均感知器算法的训练过程。

4.3 评价方法

为了衡量一个分类算法好坏，需要给定一个测试集，用分类器对测试集中的每一个样本进行分类，并根据分类结果计算评价分数。常见的评价标准有正确率、准确率、召回率和F值等。

给定测试集 $T = (x_1, y_1), \dots, (x_N, y_N)$ ，对于所有的 $y_i \in \{\omega_1, \dots, \omega_C\}$ 。假设分类结果为 $Y = \hat{y}_1, \dots, \hat{y}_N$ 。

则**正确率**（Accuracy, Correct Rate）为：

$$Acc = \frac{\sum_{i=1}^N |y_i = \hat{y}_i|}{N} \quad (4.33)$$

其中， $|\cdot|$ 为指示函数，若条件为真， $|\cdot| = 1$ ；否则 $|\cdot| = 0$ 。

和正确率相对应的就是**错误率**（Error Rate）。

$$Err = \frac{\sum_{i=1}^N |y_i \neq \hat{y}_i|}{N} \quad (4.34)$$

算法 4.3: 平均感知器算法

输入: 训练集: $(x_i, y_i), i = 1, \dots, N$, 最大迭代次数: T

输出: $\bar{\theta}$

```

1  $\theta = 0$  ;
2  $u = 0$  ;
3  $c = 0$  ;
4 for  $t = 1 \dots T$  do
5   for  $i = 1 \dots N$  do
6     选取一个样本  $(x_i, y_i)$ ;
7     用公式4.20计算预测类别  $\hat{y}_t$ ;
8     if  $\hat{y}_t \neq y_t$  then
9        $\theta = \theta + (\phi(x_t, y_t) - \phi(x_t, \hat{y}_t))$ ;
10       $u = u + c \cdot (\phi(x_t, y_t) - \phi(x_t, \hat{y}_t))$ ;
11    end
12     $c = c + 1$  ;
13  end
14 end
15  $\bar{\theta} = \theta_T - \frac{1}{c}u$  ;
16 return  $\bar{\theta}$  ;
    
```

正确率是平均的整体性能。

在很多情况下，我们需要对每个类都进行性能估计，这就需要计算准确率和召回率。正确率和召回率是广泛用于信息检索和统计学分类领域的两个度量值，在机器学习的评价中也被大量使用。

准确率 (Precision, P)，也叫查准率，精确率或精度，是识别出的个体总数中正确识别的个体总数的比例。对于类 c 来说，

$$P_c = \frac{\sum_{i=1}^N |y_i = \hat{y}_i|}{\sum_{i=1}^N 1} \quad (4.35)$$

召回率 (Recall, R)，也叫查全率，是测试集中存在的个体总数中正确识别的个体

总数的比例。

$$R_c = \frac{\sum_{i=1, y_i=c}^N |y_i = \hat{y}_i|}{\sum_{i=1, y_i=c}^N 1} \quad (4.36)$$

F1 值是根据正确率和召回率二者给出的一个综合的评价指标，具体定义如下：

$$F1_c = \frac{P_c * R_c * 2}{(P_c + R_c)} \quad (4.37)$$

为了计算分类算法在整个数据集上的总体准确率、召回率和 F1 值，经常使用两种平均方法，分别称为**宏平均**（macro average）和**微平均**（micro average）。

宏平均是每一个类的性能指标的算术平均值，

$$R_{macro} = \sum_{i=1}^C R_c / C, \quad (4.38)$$

$$P_{macro} = \sum_{i=1}^C P_c / C, \quad (4.39)$$

$$F1_{macro} = \frac{P_{macro} * R_{macro} * 2}{(P_{macro} + R_{macro})}. \quad (4.40)$$

而微平均是每一个样本的性能指标的算术平均。对于单个样本而言，它的准确率和召回率是相同的（要么都是 1，要么都是 0）因此准确率和召回率的微平均是相同的，根据 F1 值公式，对于同一个数据集它的准确率、召回率和 F1 的微平均指标是相同的。

4.4 文本分类实践

在了解机器学习的基本原理之后，我们可以尝试搭建一个简单的文本分类器。本节的示例代码可以在 fnlp-demo 项目中的找到。

当我们在网站上浏览新闻时，会发现这些新闻都是按不同类别进行组织的。当网站编辑发布一篇文章之前，需要将这篇文章归到某个类别下，比如“体育”，“经济”等。这个过程可以用文本分类器来自动处理来提高效率。

4.4.1 准备数据

首先，我们需要准备有类别标记的训练数据。为了简单起见，我们收集一些新闻的标题，然后根据这些标题进行分类。这些数据放在 **example-data-classification** 目录

下。每个文件表示一类，文件中每一行为一篇新闻的标题。

我们这里使用的示例数据是十分简单的。为了更好地验证不同分类方法的效果，通常会使用比较大的数据集。表4.1列出了一些常用的数据集。

表 4.1: 文本分类数据集

数据集	网址
20 Newsgroup	http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html
TechTC	http://techtc.cs.technion.ac.il/
Reuters 21578	http://www.daviddlewis.com/resources/testcollections/reuters21578/
搜狗文本分类语料库	http://www.sogou.com/labs/dl/c.html

4.4.2 样本表示

FNLP 中样本用 `org.fnlp.ml.types.Instance` 类（简称 **Instance** 类）描述。每一样本都是 **Instance** 类的实例化对象。**Instance** 类有两个主要的成员变量 `data` 和 `target`，它们分别表示样本的属性 x 和类别 y 。

Instance 类描述：

```

1 //表示单个样本 (x,y)。x,y 分别对应 data,target.
2 public class Instance {
3     // 样本属性，相当于 x
4     protected Object data;
5     // 标签或类别，相当于 y
6     protected Object target;
7 }

```

样本集合用 `org.fnlp.ml.types.InstanceSet` 类（简称 **InstanceSet** 类）描述。样本集合定义为 **InstanceSet** 类的实例化对象。

InstanceSet 类描述：

```

1 // 样本集合
2 public class InstanceSet extends ArrayList<Instance> {
3     // 本样本集合默认的数据类型转换管道
4     private Pipe pipes = null;
5     // 本样本集合对应的特征和标签索引字典管理器

```

```

6     private AlphabetFactory factory = null;
7     //批量读取训练数据     public void loadThruPipes(Reader reader);
8 }

```

其中，**Pipe** 和 **AlphabetFactory** 分别是特征转换管道和字典管理器。

4.4.3 特征抽取

FNLP 中特征抽取通过**管道**来进行，定义为 **org.fnlp.nlp.pipe.Pipe** 类（简称 **Pipe** 类），该类是一个抽象类。

```

1 //特征转换管道，将数据从字符串转换为基于向量的特征表示
2 public abstract class Pipe{
3     // 基本的数据类型转换处理操作，继承类需重新定义实现
4     public abstract void addThruPipe(Instance inst);
5 }

```

所有的特征转换器都继承 **Pipe** 类，通过重新定义 **addThruPipe** 方法来实现不同的数据类型转换。

一个特殊的继承类是 **org.fnlp.nlp.pipe.SeriesPipes** 类（简称 **SeriesPipes** 类），该类是一个或多个 **Pipe** 对象的串行组合。

4.4.4 特征向量

特征向量用 **org.fnlp.ml.types.sv.ISparseVector** 接口（简称 **ISparseVector** 接口）描述。

ISparseVector 接口描述：

```

1 // 稀疏向量，并实现各种向量运算
2 public interface ISparseVector extends Serializable{
3     //与数组进行点积运用
4     public float dotProduct(float[] vector);
5     //与另外一个稀疏向量进行点积运用
6     public float dotProduct(HashSparseVector sv);
7 }

```

稀疏向量 **ISparseVector** 接口的具体实现有两种。

一种是值为 0,1 的稀疏向量 **BinarySparseVector** 类。

另一种为一般取值的稀疏向量 **HashSparseVector** 类。

4.4.5 训练分类器

在了解上面基本的数据结构之后，我们可以构建一个简单的文本分类器。

```

1 //建立字典管理器
2 AlphabetFactory af = AlphabetFactory.buildFactory();
3 //使用n元特征
4 Pipe ngrampp = new NGram(new int[] {2,3});
5 //特征转换：将n元特征转换成字典索引
6 Pipe indexpp = new StringArray2IndexArray(af);
7 //类别表示：将类别转换为数字表示
8 Pipe targetpp = new Target2Label(af.DefaultLabelAlphabet());
9
10 //建立pipe组合
11 SeriesPipes pp = new SeriesPipes(new Pipe[]{ngrampp,targetpp,indexpp
    });
12
13 InstanceSet instset = new InstanceSet(pp,af);
14
15 //用不同的Reader读取相应格式的文件
16 Reader reader = new FileReader(trainDataPath,"UTF-8",".data");
17
18 //读入数据，并进行数据处理
19 instset.loadThruStagePipes(reader);
20
21 //训练分类器
22 OnlineTrainer trainer = new OnlineTrainer(af);
23 Linear pclassifier = trainer.train(instset);
24
25 //将分类器保存到模型文件
26 pclassifier.saveTo(modelFile);
27 }

```

4.4.6 使用分类器

再训练好模型之后，我们可以调用训练好的模型来对新闻标题进行自动分类。

```

1 // 待分类的新闻标题
2 String str = "人民币对美元中间价再创新高";
3 // 从模型文件读入分类器
4 Linear cl =Linear.loadFrom(modelFile);
5 // 处理数据
6 Pipe p = cl.getPipe();

```

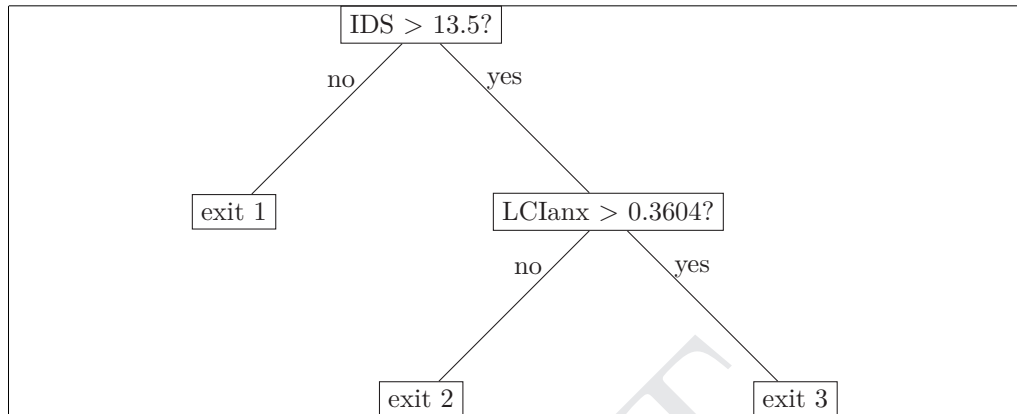



图 4.3: 决策树示例

```

7 Instance inst = new Instance(str);
8 //特征转换p.addThruPipe(inst);
9 //分类
10 String label = cl.getStringLabel(inst);
11 }

```

4.5 其他机器学习算法简介

4.5.1 决策树

决策树（Decision Tree）是一种简单但是广泛使用的分类器。通过训练数据构建决策树，可以高效的对未知的数据进行分类。决策树有两大优点：1）决策树模型可以读性好，具有描述性，有助于人工分析；2）效率高，决策树只需要一次构建，反复使用，每一次预测的最大计算次数不超过决策树的深度。

决策树的典型算法有ID3，C4.5，CART等。相对于其它算法，决策树易于理解和实现，人们在通过解释后都有能力去理解决策树所表达的意义。决策树可以同时处理不同类型的属性,并且在相对短的时间内能够对大型数据源做出可行且效果良好的结果。

4.5.2 朴素贝叶斯分类器

贝叶斯分类器根据在给定样本情况下不同类别的后验概率进行决策分类。其表示形式见公式4.43:

公式4.18可以用贝叶斯公式进行扩展：

$$f(\phi(x)) = \arg \max_{y=1}^C p(y|\phi(x)) \quad (4.41)$$

$$= \arg \max_{y=1}^C \frac{p(\phi(x)|y)p(y)}{\sum_y p(\phi(x)|y)p(y)} \quad (4.42)$$

$$\propto \arg \max_{y=1}^C p(\phi(x)|y)p(y). \quad (4.43)$$

$$f(\phi(x)) \propto \arg \max_{y=1}^C p(\phi(x)|y)p(y). \quad (4.44)$$

这里需要估计概率函数 $p(\phi(x)|y)$ ，当 $\phi(x)$ 为高维向量时，这是比较困难的工作。

在实际应用中，经常使用朴素贝叶斯（Naïve Bayes, NB）算法。在朴素贝叶斯算法中， $p(\phi(x)|y)$ 近似为样本向量中每一维变量概率的乘积。

$$p(\phi(x)|y) = \prod_{i=1}^M p(\phi_i(x)|y), \quad (4.45)$$

这里 M 是特征向量 $\phi(x)$ 的维数， $\phi_i(x)$ 是 $\phi(x)$ 第 i 维的值。

$p(\phi(x)|y)$ 之所以能展开成公式（4.45）的连乘积形式，就是假设样本每一维之间是彼此独立的。但这种情况在实际情况中经常是不成立的，因此其分类准确率可能会下降。比如在文本分类中，一个样本是一篇文档，每一维特征可以看出是一个词。但是词语之间有明显的所谓“共现”关系，在不同主题的文章中，可能共现的次数或频率有变化，但彼此间绝对谈不上独立。

但在许多场合，朴素贝叶斯分类算法的假设依然取得很好的性能，并且十分简单，可以与很多复杂的分类算法相媲美，是自然语言处理中最为常用的算法之一。

4.5.3 k 最近邻算法

k 最近邻算法 (k-Nearest Neighbor, kNN) 是根据待分样本的最相似的几个样本的类别来决定其所属类别，是最简单有效的机器学习算法之一。对于一个样本，我们可以在特征空间中找到和它最相似（即特征空间中最邻近）的 k 个样本，如果 k 个样本中的大多数属于某一个类别，则该样本也属于这个类别。当 $k=1$ 时，也称为**最近邻（NN）算法**。

kNN方法相当于非参数密度估计方法，在决策时只与极少量的相邻样本有关。由于KNN方法主要靠周围有限的邻近的样本，因此对于类域的交叉或重叠较多的非线性可分数据来说，kNN方法较其他方法更为适合，可以很好的克服了线性不可分问题的缺陷。

在kNN算法中，所选择的邻居都是已经有类别标记的样本，这也意味着kNN算法不需要训练阶段。因此，kNN算法也很适用于分类标准随时会产生变化的需求。只要删除旧的标记样本，添加新的标记样本，就改变了分类准则。

但是，kNN算法也存在一些不足之处。

一是在判断一个样本的类别时，需要把它与所有已知类别的样本都比较一遍，这样计算开销是相当大的。比如一个文本分类数据有2万个训练样本，为了判断一个新样本的类别，也要做2万次的向量比较。虽然可以通过对样本空间建立索引来提高找到最近邻的效率，但是效率依然低于其他分类器。

二是当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的k个邻居中大容量类的样本占多数，导致分类错误。

三是kNN算法很容易受到噪声特征的影响，在进行kNN分类之前需要进行特征选择。

4.5.4 支持向量机

支持向量机（Support Vector Machine, SVM）是一个经典的监督学习算法，它在解决很多任务中表现很强的优势。

支持向量机引入**边际距离**（Margin）的概念，给定训练样本 $\{(x_i, y_i)\}_{i=1}^n$ ，我们定义

$$\gamma_i(\theta) = \theta^T \phi(x, y) - \arg \max_{\hat{y} \neq y} \theta^T \phi(x, \hat{y}). \quad (4.46)$$

支持向量机的目标是寻找一个权重向量 θ^* 使得，

$$\theta^* = \arg \max_{\theta: \|\theta\|=1} \min_{i=1}^n \gamma_i(\theta). \quad (4.47)$$

公式4.47可以写为：

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2} \|\theta\|^2 \\ \text{s.t.} \quad & \gamma_i(w) \geq 1, (\forall i) \end{aligned} \quad (4.48)$$

公式4.48的约束条件比较强，为了能够容忍部分不满足约束的样本，可以引入松弛变量 ξ ：

$$\begin{aligned} \min_{\theta, \xi} \quad & \frac{1}{2} \|\theta\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \gamma_i(w) \geq 1 - \xi_i, \xi_i \geq 0, (\forall i) \end{aligned} \quad (4.49)$$

支持向量机的求解可以通过二次优化方法得到全局最优解，这使它有着其他统计学习技术难以比拟的优越性。同时，还使用核函数将原始的样本空间向高维空间进行变换，

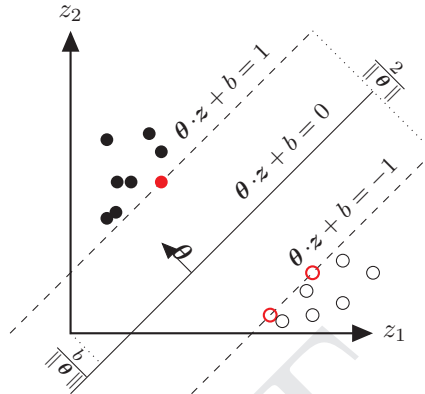


图 4.4: 支持向量机

能够解决原始样本线性不可分的问题。支持向量机的优点在于通用性较好，且分类精度高、分类速度快、分类速度与训练样本个数无关，是最常用的分类器之一。

参考文献和深入阅读

本书并不会深入或全面地介绍机器学习的理论知识。如果需要快速全面地了解机器学习的基本概念可以阅读《Pattern Classification》[Duda et al., 2001]，进一步深入了解可以阅读《The Elements of Statistical Learning》[Hastie et al., 2001] 以及《Learning in Graphical Models》[Jordan, 1998]。

Rosenblatt [1958] 最早提出了两类感知器算法，Freund and Schapire [1999] 提出了改进的投票感知器来提高泛化能力。Collins [2002] 将感知器算法扩展到结构化学习，给出了相应的收敛性证明，并且提出一种更加有效并且实用的参数平均化策略。

文献[Sebastiani, 2002] 对文本分类中的机器学习算法的应用做了比较详细和全面的介绍。关于支持向量机的文本分类方法可以参考[Joachims, 1998]。文本分类中的特征向量表示方法，比如词袋模型，可以看作是向量空间模型[Salton et al., 1975]的一个特例，向量空间模型是文档表示的一个统计模型，在信息检索的相关书籍也会有详细的介绍[Yates and Neto, 1999]。

特征选择在文本分类中也起到很关键的作用。特别是对基于kNN算法的文本分类，特征选择是一个必不可少的步骤。比较有效的特征选择方法有互信息、 κ^2 检验等，相关的实验比较可以参考[Yang and Pedersen, 1997]。关于的

第五章 非监督学习

导言

：词义相似度可以分为两个角度来看：聚合 () 和组合 () 相似性。

DRAFT
编译时间: 2014-10-31 16:03

5.0.5 Brown 聚类

可以不言而喻的是，前缀相似度更高的词就越邻近。什么样的汇词相似呢？一个直觉的想法就是：相似的词出在现相似的置位。更准确的说法就是：相似词的驱前词和后继词的分布相似，也就是它面前的词和面后的词现出得是相似的。

为何 Brown 聚类倾向于寻找固定搭配，即横行组合相似度高的词。我们假设输入序列是 $\mathbf{w} = \{w_1, \dots, w_n\}$ ，函数 Ω 将词 w_i 分到 k 个词类 $\{\omega_1, \dots, \omega_k\}$ 中的一个，即 $\Omega(w_i) \in \{1, \dots, k\}$ 。 w_i 属于字典 \mathcal{C} 。那么如何衡量 Ω 的好坏呢？Brown 聚类采用了基于类的二元语言模型。

$$Q(\Omega) = \frac{1}{n} \log P(w_1, \dots, w_n) \quad (5.1)$$

$$= \frac{1}{n} \log P(w_1, \dots, w_n, \Omega(w_1), \dots, \Omega(w_n)) \quad (5.2)$$

$$= \frac{1}{n} \log \prod_{i=1}^n P(w_i | \Omega(w_i)) P(C(w_i) | \Omega(w_{i-1})) \quad (5.3)$$

$$= \frac{1}{n} \sum_{i=1}^n \log P(w_i | \Omega(w_i)) P(C(w_i) | \Omega(w_{i-1})) \quad (5.4)$$

$$= \sum_{c, c' \in \mathcal{C}} \frac{n(c, c')}{n} \log P(c' | \Omega(c')) P(\Omega(c') | \Omega(c)) \quad (5.5)$$

我们定义 $n(\omega) = \sum_{\Omega(c)=\omega} n(c)$ 为词类 ω 出现的次数，即属于这个词类的所有单词出现次数之和。

$$Q(\Omega) = \sum_{c, c' \in \mathcal{C}} \frac{n(c, c')}{n} \log \frac{n(c')}{n(\Omega(c'))} \frac{n(\Omega(c'), \Omega(c))}{n(\Omega(c))} \quad (5.6)$$

$$= \sum_{c, c' \in \mathcal{C}} \frac{n(c, c')}{n} \log \frac{n(c')}{n} \frac{n(\Omega(c'), \Omega(c)) \times n}{n(\Omega(c)) \times n(\Omega(c'))} \quad (5.7)$$

$$= \sum_{c' \in \mathcal{C}} \frac{n(c')}{n} \log \frac{n(c')}{n} + \sum_{\omega, \omega'} \frac{n(\omega, \omega')}{n} \log \frac{n(\omega, \omega') \times n}{n(\omega) \times n(\omega')} \quad (5.8)$$

$$= \sum_{c' \in \mathcal{C}} P(c') \log P(c') + \sum_{\omega, \omega'} P(\omega, \omega') \log \frac{P(\omega, \omega')}{P(\omega) \times P(\omega')} \quad (5.9)$$

$$= I(\Omega) - H \quad (5.10)$$

$I(\Omega)$ 为两个相邻词类之间的互信息。

非监督学习（Unsupervised Learning）是指直接对输入的无类别标记的数据集进行建模。

监督学习必须要有标记好的训练集，在训练集中找规律，以求对训练集数据达到某种最优，并能推广到新数据。而非监督学习只有无标记的数据，在该组数据集内寻找规律。非监督学习方法只有要分析的数据集本身，没有标号。如果发现数据集呈现某种聚集性，则可按自然的聚集性分类，但并没有预先的分类标号。

样本数据类别未知，需要根据样本间的相似性对样本集进行分类，这个过程可以称为聚类。

聚类（Clustering）就是将数据分组成为多个类（Cluster）。在同一个类内对象之间具有较高的相似度，不同类之间的对象差别较大。

非监督式学习是一种机器学习的方式，并不需要人力来输入标签。它是监督式学习和强化学习等策略之外的一种选择。在监督式学习中，典型的任务是分类和回归分析，且需要使用到人工预先准备好的范例。一个常见的非监督式学习是数据聚类。在人工神经网络中，自我组织映射（SOM）和适应性共振理论（ART）则是最常用的非监督式学习。

5.1 聚类算法

目标是我们不告诉计算机怎么做，而是让它（计算机）自己去学习怎样做一些事情。非监督学习一般有两种思路。第一种思路是在指导 Agent 时不为其指定明确的分类，而是在成功时采用某种形式的激励制度。需要注意的是，这类训练通常会置于决策问题的框架里，因为它的目标不是产生一个分类系统，而是做出最大回报的决定。这种思路很好的概括了现实世界，Agent 可以对那些正确的行为做出激励，并对其他的行为进行处罚。强化学习的一些形式常常可以被用于非监督学习，由于没有必然的途径学习影响世界的那些行为的全部信息，因此 Agent 把它的行为建立在前一次奖惩的基础上。在某种意义上，所有的这些信息都是不必要的，因为通过学习激励函数，Agent 不需要任何处理就可以清楚地知道要做什么，因为它（Agent）知道自己采取的每个动作确切的预期收益。对于防止为了计算每一种可能性而进行的大量计算，以及为此消耗的大量时间（即使所有世界状态的变迁概率都已知），这样的做法是非常有益的。另一方面，在尝试出错上，这也是一种非常耗费时间的学习。不过这一类学习可能会非常强大，因为它假定没有事先分类的样本。在某些情况下，例如，我们的分类方法可能并非最佳选择。在这方面一个突出的例子是 Backgammon（西洋双陆棋）游戏，有一系列计算机程序（例如 neuro-gammon 和 TD-gammon）通过非监督学习自己一遍又一遍的玩这个游戏，变得比最强的人类棋手还要出色。这些程序发现的一些原则甚至令双陆棋专家都感到惊讶，并且它们比那些使用预分类样本训练的双陆棋程序工作得更出色。一种次要的非监督学习类型称之为聚合（clustering）。这类学习类型的目标不是让效用函数最大化，而是找到训练数据中的近似点。聚合常常能发现那些与假设匹配的相当好的直观

分类。例如，基于人口统计的聚合个体可能会在一个群体中形成一个富有的聚合，以及其他的贫穷的聚合。

在机器学习 (Machine learning) 领域，监督学习 (Supervised learning)、非监督学习 (Unsupervised learning) 以及半监督学习 (Semi-supervised learning) 是三类研究比较多，应用比较广的学习技术，wiki 上对这三种学习的简单描述如下：监督学习：通过已有的一部分输入数据与输出数据之间的对应关系，生成一个函数，将输入映射到合适的输出，例如分类。非监督学习：直接对输入数据集进行建模，例如聚类。半监督学习：综合利用有类标的数据和没有类标的数据，来生成合适的分类函数。以上表述是我直接翻译过来的，因为都是一句话，所以说得不是很清楚，下面我用一个例子来具体解释一下。其实很多机器学习都是在解决类别归属的问题，即给定一些数据，判断每条数据属于哪些类，或者和其他哪些数据属于同一类等等。这样，如果我们上来就对这一堆数据进行某种划分 (聚类)，通过数据内在的一些属性和联系，将数据自动整理为某几类，这就属于非监督学习。如果我们一开始就知道了这些数据包含的类别，并且有一部分数据 (训练数据) 已经标上了类标，我们通过对这些已经标好类标的数据进行归纳总结，得出一个“数据 \rightarrow 类别”的映射函数，来对剩余的数据进行分类，这就属于监督学习。而半监督学习指的是在训练数据十分稀少的情况下，通过利用一些没有类标的数据，提高学习准确率的方法。铺垫了那么多，其实我想说的是，在 wiki 上对于半监督学习的解释是有一点点歧义的，这跟下面要介绍的主动学习有关。主动学习 (active learning)，指的是这样一种学习方法：有的时候，有类标的数据比较少而没有类标的数据是相当丰富的，但是对数据进行人工标注又非常昂贵，这时候，学习算法可以主动地提出一些标注请求，将一些经过筛选的数据提交给专家进行标注。这个筛选过程也就是主动学习主要研究的地方了，怎么样筛选数据才能使得请求标注的次数尽量少而最终的结果又尽量好。主动学习的过程大致是这样的，有一个已经标好类标的数据集 K (初始时可能为空)，和还没有标记的数据集 U ，通过 K 集合的信息，找出一个 U 的子集 C ，提出标注请求，待专家将数据集 C 标注完成后加入到 K 集合中，进行下一次迭代。按 wiki 上所描述的看，主动学习也属于半监督学习的范畴了，但实际上是不一样的，半监督学习和直推学习 (transductive learning) 以及主动学习，都属于利用未标记数据的学习技术，但基本思想还是有区别的。如上所述，主动学习的“主动”，指的是主动提出标注请求，也就是说，还是需要一个外在的能够对其请求进行标注的实体 (通常就是相关领域人员)，即主动学习是交互进行的。而半监督学习，特指的是学习算法不需要人工的干预，基于自身对未标记数据加以利用。至于直推学习，它与半监督学习一样不需要人工干预，不同的是，直推学习假设未标记的数据就是最终要用来测试的数据，学习的目的就是在这些数据上取得最佳泛化能力。相对应的，半监督学习在学习时并不知道最终的测试用例是什么。也就是说，直推学习其实类似于半监督学习的一个子问题，或者说是一个特殊化的半监督学习，所以也有人将其归为半监督学习。而主动学习和半监督学习，其基本思想上就不一样了，所以还是

要加以区分的，如果 wiki 上对半监督学习的解释能特别强调一下“是在不需要人工干预的条件下由算法自行完成对无标记数据的利用”，问题就会更清楚一些了。

DRAFT
编译时间: 2014-10-31 16:03

第六章 序列标注模型

序列标注是自然语言处理领域的一个非常常见的问题，从分词、词性标注，到较深层的组块分析以至更为深层的完全句法分析、语义角色标注等任务，都可以看作是典型的序列标注问题。

在自然语言处理应用中，比如分词、词性标注等，不再是简单的离散分类问题，样本的不同元素标记之间有很强的相关性，因此不能孤立地对每个待标注对象进行分类和标记，必须进行联合标记。

序列标注问题是指对于序列形式的输入样本 $x = x_1, \dots, x_n$ ，对序列中每个元素进行标记，输出标记序列 $y = y_1, \dots, y_n$ ， n 是序列的长度。若 y_i 的取值范围定义为 $S = \{s_i\}_{i=1}^C$ ，输出序列的可能组合数为 C^L 。变量 y_i 的不同取值也叫不同的状态。 y_i 所有可能取值的集合 S ，也被称为“状态空间”。因为一个序列状态的组合数非常多，也不能直接用传统的学习方法通过枚举 y 来得到最佳的标记，需要用动态优化的方法来求解 y 。

序列中每个元素间关联紧密，元素标记之间也具有很强的相关性，传统的单点分类器方法难以获得整个序列的最优标记。图6.1是两种线性链序列标注结构，每个元素标记只与相邻的元素相关，构成了线性链式结构。其中，图6.1a是有向图结构，每个元素标记只与前一个元素标记相关，图6.1b是无向图结构，每个元素标记与左右两个相邻元素标记相关。

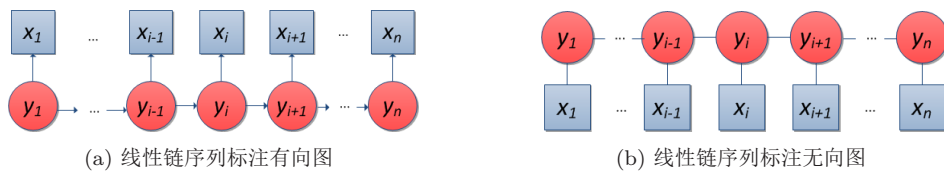


图 6.1: 线性链序列标注

序列标注问题是结构化机器学习的一种特例。**结构化机器学习**就是指处理的样本 (x, y) , y 不再是离散的类别, 而是有结构的, 比如序列、树结构等。序列标注模型都可以和图模型 [Jordan, 1998] 进行对应。

序列标注问题的经典参考文献如下:

1. C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. Introduction to statistical relational learning, page 93, 2007
2. John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, 2001
3. A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 591–598. Citeseer, 2000
4. Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, 2002
5. Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In In proceedings of the 17th Annual Conference on Neural Information Processing Systems, Whistler, B.C., Canada, 2003
6. I. Tsochantaridis, T. Hofmann, T. Joachims, and Y Altun. Support vector machine learning for interdependent and structured output spaces. In Proceedings of the International Conference on Machine Learning(ICML), 2004

6.1 序列标注问题

序列标注问题可以用下面公式表示:

$$\hat{y} = \overbrace{\arg \max_y}^{\text{解码}} \underbrace{f(\underbrace{\Phi(x, y)}_{\text{特征}}, \underbrace{\mathbf{w}}_{\text{模型参数}})}_{\text{模型}} \quad (6.1)$$

其中, x 和 y 分别是输入序列和标记序列, $\Phi(x, y)$ 是在序列上抽取的特征向量, $f(\Phi(x, y), \mathbf{w})$ 是序列标注模型, \mathbf{w} 是模型参数。

从公式6.1可知, 序列标注问题需要解决四个问题:

1. 如何选择合适的序列标注模型？确定标记之间的关联关系。
2. 怎样从序列上抽取特征？
3. 如何进行求解？也就是解码问题。
4. 如何进行参数学习？

下面我们分别来讨论这四个问题：

6.1.1 序列标注模型

序列标注模型就是定义一个函数 $f(\cdot)$ ，用来描述一个序列标注任务，包括模型结构（不同元素标记之间的关联关系），解码方法以及参数学习方法。常见的序列标注模型有：线性模型、隐马尔可夫模型、最大熵马尔可夫模型、条件随机场等。

首先，我们看下序列标注模型的结构，一般用阶数来形容模型结构和复杂度。阶数是从马尔可夫链中借鉴来的一个概念。

马尔可夫链，简称马氏链，是由随机变量组成的一个序列 $x_1, x_2, x_3, \dots, x_t$ 的值是在时间 t 时的状态。如果 x_{t+1} 对于过去状态的条件概率分布仅是 x_t 的一个函数，即

$$P(x_{t+1}|x_1, x_2, \dots, x_t) = P(x_{t+1}|x_t) \quad (6.2)$$

那么，这个序列就称为**一阶马尔可夫链**，简称马尔可夫链。

如果 x_{t+1} 对于过去状态的条件概率分布仅是 x_t, x_{t-1} 的一个函数，即

$$P(x_{t+1}|x_1, x_2, \dots, x_t) = P(x_{t+1}|x_t, x_{t-1}) \quad (6.3)$$

那么，这个序列就称为**二阶马尔可夫链**。

如果 x_{t+1} 对于过去状态的条件概率分布仅是 x_{t-m+1}, \dots, x_t 的一个函数，即

$$P(x_{t+1}|x_1, x_2, \dots, x_t) = P(x_{t+1}|x_{t-m+1}, \dots, x_t) \quad (6.4)$$

那么，这个马尔可夫链的**阶数**就是 m 。

一般阶数大于等于 2 的马尔可夫链，也叫**高阶马尔可夫链**。

图6.2给出了序列标注问题中一阶和二阶的序列标注模型。

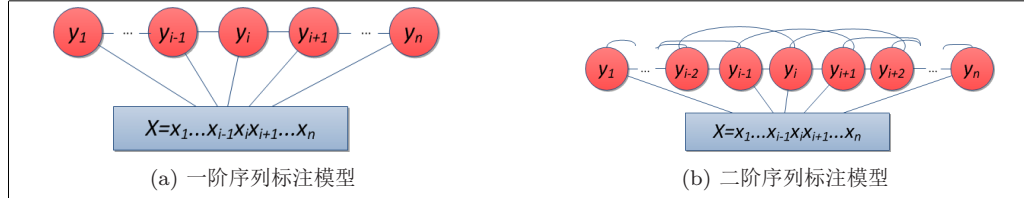


图 6.2: 序列标注模型

序列标注模型可以分为两大类：一种是非统计方法，另一种是统计的方法。

在非统计方法中，最有代表性的是线性分类器：

$$y = \arg \max_{y'} \mathbf{w} \cdot \Phi(x, y), \quad (6.5)$$

在基于统计方法，比较主流的方法是用无向图来表示模型。

在无向图模型中，条件概率可以表示为：

$$P(y|x) = \frac{\exp(\mathbf{w} \cdot \Phi(x, y))}{Z_x}, \quad (6.6)$$

其中， Z_x 是 x 的边际概率， $Z_x = \sum_{y' \in \mathcal{Y}} \exp(\mathbf{w} \cdot \Phi(x, y'))$ 。

基于无向图模型的序列标注可以表示为：

$$y = \arg \max_{y'} P(y'|x) \quad (6.7)$$

$$= \arg \max_{y'} \frac{\exp(\mathbf{w} \cdot \Phi(x, y'))}{Z_x}, \quad (6.8)$$

$$= \arg \max_{y'} \exp(\mathbf{w} \cdot \Phi(x, y')), \quad (6.9)$$

$$= \arg \max_{y'} \mathbf{w} \cdot \Phi(x, y'), \quad (6.10)$$

从公式6.5和6.10看出，基于统计和非统计的方法殊途同归，最后的解码公式是相同的。但这不意味这些模型是相同的，不同的模型的出发点不同，学习参数的目标函数也不同，导致最终的模型也存在一定差异。

6.1.2 特征生成

6.1.3 特征 v.s. 模板

在给定序列标注模型后，接下来一个问题是如何把一个样本 (x, y) 转换为向量表示 $\Phi(x, y)$ ，也就是特征向量。输入 $x \in \mathcal{X}$ 是观察序列，可以是一篇文档，一个句子，有字或

词为单位构成的序列。 $x = x_0, x_1, \dots, x_n$ 。输出 $y \in \mathcal{Y}$ 是状态序列，可以 x 对应的分词、词性、实体名的标记序列。 $y = y_0, y_1, \dots, y_n$ 。 $\Phi(x, y)$ 是将 (x, y) 映射为特征向量的函数：

$$\Phi(x, y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m \quad (6.11)$$

根据序列标注模型的关联性假设，我们可以对 $\Phi(x, y)$ 进行分解（factorization）。

对于一阶模型， $\Phi(x, y)$ 可以分解为：

$$\Phi(x, y) = \sum_{i=1}^{|y|} \phi(x, y_{i-1}, y_i) \quad (6.12)$$

对于 k 阶模型， $\Phi(x, y)$ 可以分解为：

$$\Phi(x, y) = \sum_{i=1}^{|y|} \phi(x, y_{i-k}, \dots, y_{i-1}, y_i) \quad (6.13)$$

下面通过一个例子来说明如何构造特征。

对于中文分词的序列标注问题，可以定义 $y \in \{B, O\}$ ，这里 B 表示把当前字作为一个新词的开始， O 表示当前字与前面的字构成一个词。例如：句子“他 / 说 / 的 / 确实 / 在理”转化为下面以字为基本元素构成的序列。

x	=	他	说	的	确	实	在	理
y	=	B	B	B	B	O	B	O

对于一阶序列标注模型， $\phi(x, y_{i-1}, y_i)$ 可以定义在 x 中的任何元素以及当前位置两个相邻的标记上。比如对于下面两个特征： ϕ_j 和 ϕ_k ：

$$\phi_j(x, y_{i-1}, y_i) = \begin{cases} 1 & \text{当 } x_i = \text{“在”}, x_{i+1} = \text{“理”}, y_{i-1} = \text{“O”}, y_i = \text{“B”} \\ 0 & \text{otherwise} \end{cases} \quad (6.14)$$

$$\phi_k(x, y_{i-1}, y_i) = \begin{cases} 1 & \text{当 } x_i = \text{“在”}, x_{i+1} = \text{“理”}, y_{i-1} = \text{“O”}, y_i = \text{“O”} \\ 0 & \text{otherwise} \end{cases} \quad (6.15)$$

显然，对于“在”后面跟着“理”，并且“在”之前的字已经构成了词的一部分，那么在“在”就是一个新词的开始。特征 ϕ_j 比 ϕ_k 更符合中文分词习惯，它们对应的特征权重 w_j 应该大于 w_k 。这样，在解码时，“在”在这种情况下更倾向被标记为“B”。

6.1.4 解码问题

对于未知标记的样本 x , \hat{y} 可以通过一个得分函数求得,

$$\hat{y} = \arg \max_y f(\mathbf{w}, \Phi(x, y)), \quad (6.16)$$

这里, \mathbf{w} 是函数 $f(\cdot)$ 的参数.

从公式6.5和6.10可知, 解码问题的一般形式为:

$$\hat{y} = \arg \max_y \mathbf{w} \cdot \Phi(x, y), \quad (6.17)$$

假设当前模型为一阶模型, 公式6.21可以转换为:

$$\hat{y} = \arg \max_y \mathbf{w} \cdot \Phi(x, y) \quad (6.18)$$

$$= \arg \max_y \mathbf{w} \cdot \sum_{i=1}^{|y|} \phi(x, y_{i-1}, y_i) \quad (6.19)$$

$$= \arg \max_y \sum_{i=1}^{|y|} \mathbf{w} \cdot \phi(x, y_{i-1}, y_i) \quad (6.20)$$

假设状态空间大小为 C , 对于长度为 n 的 y , 其可能的组合数为 C^n . 因此, 穷举不同的 y 已获得最佳序列是不可行的. 通过观察公式6.20, 我们可以用动态优化方法来快速的求解.

我们首先定义 $\alpha_{s,i}$ 是输入序列 x_0, \dots, x_i 且 $y_i = s$ 的最佳标记序列. 公式6.20可以写为:

$$\hat{y} = \max_s \alpha_{s,n}, \quad (6.21)$$

$\alpha_{s,i}$ 可以通过下面两个递归公式来计算:

$$\alpha_{s,0} = 0, \forall s \in \mathcal{S} \quad (6.22)$$

$$\alpha_{s,i} = \max_{s'} \alpha_{s',i-1} + \mathbf{w} \cdot \phi(x, s', s) \quad (6.23)$$

这个方法也叫 **Viterbi 算法**, 可以保证找到得分最高的标记序列.

6.1.5 参数学习

最后一个问题是如何学习函数 $f(\cdot)$ 的参数 \mathbf{w} . 根据 $F(\cdot)$ 的形式不同, 学习参数的方法也不同, 一般为最大似然估计、最大边际距离或最小均方误差等. 我们可以用传统的分类器训练算法, 比如感知器、SVM、kNN等.

这里我们介绍一种简单有效的参数学习方法，**Passive-Aggressive (PA) 算法** Crammer et al. [2006]。PA 算法是一种在线学习算法，它结合了感知器和 SVM 的优点，学习速度快，效果可以和批量学习算法近似。

PA 算法

给定一个样本 (x, y) ， \hat{y} 定义为错误标签中得分最高的标签。

$$\hat{y} = \arg \max_{z \neq y} \mathbf{w} \cdot \Phi(x, z). \quad (6.24)$$

边际距离 $\gamma(\mathbf{w}; (x, y))$ 定义为：

$$\gamma(\mathbf{w}; (x, y)) = \mathbf{w} \cdot \Phi(x, y) - \mathbf{w} \cdot \Phi(x, \hat{y}). \quad (6.25)$$

损失函数定义为：**hinge loss**.

$$\ell(\mathbf{w}; (x, y)) = \begin{cases} 0, & \gamma(\mathbf{w}; (x, y)) > 1 \\ 1 - \gamma(\mathbf{w}; (x, y)), & \text{otherwise} \end{cases} \quad (6.26)$$

PA 算法用在线的方式计算更新参数。在第 t 轮，通过下面公式计算 \mathbf{w}_{t+1} ：

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg \min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C \cdot \xi, \\ \text{s.t. } \ell(\mathbf{w}; (x_t, y_t)) &\leq \xi \text{ and } \xi \geq 0 \end{aligned} \quad (6.27)$$

这里， C 是正的参数来控制松弛变量的影响。

我们用 ℓ_t 来表示 $\ell(\mathbf{w}_t; (x, y))$ 。如果 $\ell_t = 0$ ，那么 \mathbf{w}_t 满足公式 6.1.5。因此我们只关心 $\ell_t > 0$ 的情况。当 $\ell_t > 0$ 时，我们通过公式求解新的参数 \mathbf{w} ，即寻找一个 \mathbf{w} 使得下面公式最小化：

$$\begin{aligned} \min \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C \cdot \xi, \\ \text{s.t. } \ell(\mathbf{w}; (x_t, y_t)) &\leq \xi \text{ and } \xi \geq 0 \end{aligned} \quad (6.28)$$

我们通过拉格朗日优化方法求解，公式 6.28 转换为对偶形式：

$$\begin{aligned} \mathcal{L}(\mathbf{w}, (x_t, y_t), \alpha, \beta) &= \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C \cdot \xi + \alpha(\ell(\mathbf{w}; (x_t, y_t)) - \xi) - \beta\xi, \\ \text{s.t. } \alpha &\geq 0 \text{ and } \beta \geq 0 \end{aligned} \quad (6.29)$$

将 \mathcal{L} 对 \mathbf{w} 求导，并令其等于 0 得：

$$0 = \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \mathbf{w}_t - \alpha(\Phi(x, y) - \Phi(x, \hat{y})). \quad (6.30)$$

即,

$$\mathbf{w} = \mathbf{w}_t + \alpha(\Phi(x, y) - \Phi(x, \hat{y})). \quad (6.31)$$

将 \mathcal{L} 对 ξ 求导, 并令其等于0得:

$$0 = \nabla_{\xi} \mathcal{L} = \mathcal{C} - \alpha - \beta. \quad (6.32)$$

即

$$\beta = \mathcal{C} - \alpha. \quad (6.33)$$

将公式6.31和6.33代入公式6.29得到:

$$\mathcal{L}(\alpha) = \frac{1}{2} \|\alpha(\Phi(x, y) - \Phi(x, \hat{y}))\|^2 + \alpha \mathbf{w}^T (\Phi(x, y) - \Phi(x, \hat{y})) - \alpha. \quad (6.34)$$

将公式6.34对 α 求导, 并令其等于0得:

$$\alpha = \frac{1 - \mathbf{w}_t^T (\Phi(x, y) - \Phi(x, \hat{y}))}{\|\Phi(x, y) - \Phi(x, \hat{y})\|^2}. \quad (6.35)$$

由公式6.33可知,

$$\alpha \leq \mathcal{C}. \quad (6.36)$$

最终我们得到更新策略为:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha^* (\Phi(x, y) - \Phi(x, \hat{y})). \quad (6.37)$$

其中,

$$\alpha^* = \min(\mathcal{C}, \alpha). \quad (6.38)$$

算法如6.1所示。为了避免过拟合，这里使用了平均的策略。

算法 6.1: PA 算法

输入: 训练集: $(x_n, y_n), n = 1, \dots, N$, 参数: C, K
 输出: w

```

1  初始化:  $cw \leftarrow 0$ ;
2  for  $k = 0 \dots K - 1$  do
3       $w_0 \leftarrow 0$ ;
4      for  $t = 0 \dots T - 1$  do
5          挑一个样本  $(x_t, y_t)$ ;
6          预测:  $\hat{y}_t = \arg \max_{z \neq y_t} \langle w_t, \Phi(x_t, z) \rangle$ ;
7          计算  $\ell(w; (x, y))$ ;
8          用 Eq.(6.37) 更新  $w_{t+1}$ ;
9      end
10      $cw = cw + w_T$ ;
11 end
12  $w = cw / K$ ;
```

6.2 常见的序列标注模型

下面我们介绍几种常见的序列标注模型。

6.2.1 线性模型

线性模型的表示如下:

$$y = \arg \max_{y'} w \cdot \Phi(x, y), \quad (6.39)$$

不同线性模型的区别是如何计算参数 w 。常见的训练方法有: 感知器模型、PA 模型、结构化 SVM 等。

6.2.2 隐马尔可夫模型

隐马尔可夫模型 (Hidden Markov Model, HMM) [Rabiner, 1989] 是最早的序列标注模型, 自 20 世纪 80 年代以来被应用于语音识别, 取得重大成功, 成为信号处理的一个重要方向。

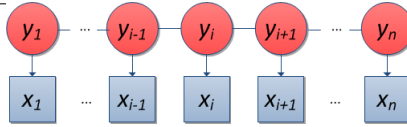


图 6.3: 隐马尔可夫模型

HMM 有三个典型问题:

- 1 已知模型参数, 计算某一特定输出序列的概率. 通常使用 forward 算法解决.
- 2 已知模型参数, 寻找最可能的能产生某一特定输出序列的隐含状态的序列. 通常使用 Viterbi 算法解决.
- 3 已知输出序列, 寻找最可能的状态转移以及输出概率. 通常使用 Baum-Welch 算法以及 Reversed Viterbi 算法解决.

HMM 的三个问题可以通过动态规划方法解决, 比如前向算法、后向算法、Viterbi 算法和 Baum-Welch 算法。

6.2.3 最大熵马尔可夫模型

隐马尔可夫模型是一个生成式模型, 要对 $p(x, y)$ 进行建模。并且样本序列的观测值只与当前状态（标记）有关, 这就限制了模型的能力。最大熵马尔可夫模型是一个判别式模型, 直接对 $p(y|x)$ 进行建模, 这样可以利用大量的冗余特征提高模型性能。最大熵马尔可夫模型如图6.4所示。

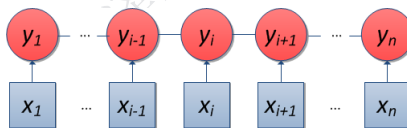


图 6.4: 最大熵马尔可夫模型

最大熵马尔可夫模型可以用 Generalized Iterative Scaling (GIS)

6.2.4 条件随机场

最大熵马尔可夫模型是用局部信息去优化全局, 会有标注偏置 (Label Bias) 的问题。条件随机场 (Conditional Random Fields, CRF) [Lafferty et al., 2001] 是一种无向

图模型，它是在给定需要标记的观察序列 x 的条件下计算整个标记序列 y 的联合概率分布，而不是在给定当前状态条件下定义下一个状态的分布。即

$$P(y|x) = \frac{\exp(\mathbf{w} \cdot \Phi(x, y))}{Z_x}, \quad (6.40)$$

隐马尔可夫模型中存在两个假设：输出独立性假设和马尔可夫性假设。其中，输出独立性假设要求序列数据严格相互独立才能保证推导的正确性，而事实上大多数序列数据不能被表示成一系列独立事件。而条件随机场则使用一种概率图模型，条件随机场没有隐马尔可夫模型那样严格的独立性假设条件，因而可以容纳任意的上下文信息，可以灵活地设计特征。同时，条件随机场具有表达长距离依赖性和交叠性特征的能力，而且所有特征可以进行全局归一化，能够求得全局的最优解，还克服了最大熵马尔可夫模型标记偏置的缺点。

条件随机场模型作为一个整句联合标定的判别式概率模型，同时具有很强的特征融入能力，是目前解决自然语言序列标注问题最好的统计模型之一。条件随机场的缺点是训练的时间比较长。

6.2.5 最大边际距离马尔科夫网络

最大边际距离马尔科夫网络 (Maximum margin Markov networks, M^3N) Taskar et al. [2003] 定义了一个对数线性马尔科夫网络。该模型的参数通过基于边际距离的优化问题进行求解。

$$\begin{aligned} \min \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{\mathbf{x}} \xi_{\mathbf{x}} \\ \text{s.t.} \quad & \mathbf{w}^T \Delta \mathbf{f}_{\mathbf{x}}(\mathbf{y}) \geq \Delta t_{\mathbf{x}}(\mathbf{y}) - \xi_{\mathbf{x}}, \forall \mathbf{x}, \mathbf{y} \end{aligned} \quad (6.41)$$

这里， ξ 是一个非负的松弛变量。

对于容易进行三角剖分的马尔科夫网络，可以讲二次优化问题转换为等价的多项式方程来进行高效的计算。

第七章 中文分词

词是最小的能够独立活动的有意义的语言成分，英文单词之间是以空格作为自然分界符的，而汉语是以字为基本的书写单位，词语之间没有明显的区分标记，因此，中文分词是中文信息处理的基础与关键。

中文句子是由字¹组成的连续字符串。为了理解中文语义，首先需要将句子划分成以词为基本单位的词串，这就是中文分词。

中文分词最大的难点在于分词标准不明确。根据《信息处理用现代汉语分词规范 GB/T 13715-92》定义，词是最小的能独立运用的语言单位。汉语分词是从信息处理需要出发，按照特定的规范，对汉语按分词单位进行划分的过程。分词的基本原则是结合紧密、使用稳定。

从这个规范可以看出，词的定义非常灵活，不仅仅和词法、语义相关，在很大程度上也和应用场景、使用频率等其它因素相关。比如：“吃饭”是词，而“吃鱼”不是一个词。另外，像“不管三七二十一”、“由此可见”等一些常用的短语也看成一个词。

分词的方法可以概括为以下三类：

模板	特征
基于规则的方法	字典匹配等方法
无监督学习方法	比如词频统计、关联度分析等方法
监督学习方法	分类器、序列标注等方法

其中，由于基于规则的方法，尤其是字典匹配，易于理解和实现，成为分词方法中最流行的。因为字典匹配的方法不考虑具体的语言环境和语义，它最大的缺点就是不能处理多词冲突和新词情况。

¹ “字”不只限于汉字。目前中文表述中不可避免地会包含少量的非汉字字符，本文所说的“字”，也包括外文字母、阿拉伯数字和标点符号等字符。所有这些字符都是构词的基本单元。

多词冲突，也称为歧义分词¹，是指一个字本身，或者与左右的相邻字组合，都可以匹配上词典中的词。表7.1 给出了多词冲突的示例。当然，词典匹配方法也有很多算法来处理这种情况，包括最大切分（包括向前、向后、以及前后相结合）、最少切分、全切分等。但这些算法都在一定程度上存在不足。

表 7.1: 多词冲突示例

字典	示例句子
的、的确、确实、实在、在理	他说的确实在理
两、个、个人、人	两个人、个人问题
马、上、马上	坐在马上、马上来
大、大学、学生、大学生、生源	大学生、大学生源

新词，也成为未登录词，是指在字典或训练语料中都没有出现过的词。常见新词包括：人名、机构名、地名、产品名、商标名、简称、省略语、专业术语、网络新词等。字典匹配的方法只能靠不断在字典中增加新词来解决这个问题。

此外，在中文分词中，不用应用场景对分词性能的需求不同，比如在搜索引擎的应用中，对中文的处理是基于自动切分的单字切分，或者二元切分。

7.1 基于两类分类器的中文分词

中文句子中的相邻字符是否可以组成一个词，不能单独看这几个相邻的字符，还要看其出现的上下文环境。不同的上下文环境，同样的相邻字符串，有的可以组成一个词，有的就不能组成一个词。这也是基于字典匹配的分词方法的不足之处。

为了充分利用上下文信息，我们可以从机器学习的角度来看待中文分词问题。给定一个字符序列，中文分词问题可以看成是在每两个连续字符之间判断是否切分。如下面例子所示，我们用 $:$ 表示一个潜在切分，其取值为 $\{0, 1\}$ ，0表示不切分，1表示切分。

自	:	然	:	语	:	言	:	处	:	理
	0		1		0		1		0	

对于每一个“ $:$ ”，我们用一个固定大小的窗口取得上下文信息，作为样本 x ，“ $:$ ”的取值作为类别标签 y 。这样我们可以中文分词问题看成是两类分类问题。

¹ 这里的“歧义”仅仅是针对字典匹配方法而言，在人的认知上不存在歧义。因此不建议使用“歧义分词”这个定义。对于人也不能区分的真歧义（比如：“乒乓球拍卖完了”、等），这已经超出了中文分词的范畴。

窗口大小	样本 x	类别标签 y
2	“自:然”	0
	“然:语”	1
	“语:言”	0
4	“自然:语言”	1
	“然语:言处”	0
	“语言:处理”	1

但这样的方法有一个缺陷：在一个句子中，每个潜在切分是独立判定的。这个独立性假设并不符合人们的直观感觉。

7.2 基于字标记的中文分词

近年来，序列标注方法成为主流的中文分词方法，该方法将中文分词转化为基于字的序列标注问题，不依赖词典，极大地改进了歧义分词和新词识别的性能。Xue [2003] 将分词问题转换为基于字的序列标注问题，根据每个字在词中的位置赋予其一个标记。标记共四类：左、中、右、单字词，并用最大熵模型，获得很好的效果。Peng et al. [2004] 也将分词问题看成序列标记问题，但是使用条件随机场模型，取得了比最大熵模型更好的性能。

首先，我们需要将分词问题转换为序列标记问题。我们用 $\{B, M, E, S\}$ 分别表示当前字是词的开始，中间，结尾和单字词。例如：句子“FNL P主要是为中文自然语言处理而开发的工具包，也包含为实现这些任务的机器学习算法和数据集。”转化为下面以字为基本元素构成的序列。

FNL P	主	要	是	为	中	文	自	然	语	言	处	理	而
S	B	E	S	S	B	E	B	E	B	E	B	E	S
开	发	的	工	具	包	，	也	包	含	为	实	现	这
B	E	S	B	M	E	S	S	B	E	S	B	E	B
任	务	的	机	器	学	习	算	法	和	数	据	集	。
B	E	S	B	E	B	E	B	E	S	B	M	E	S

7.2.1 特征模板

特征的好坏直接会影响学习的性能。在中文分词中，特征一般通过特征模板进行抽取。表7.2中列出了在中文分词常用的特征模板。

表 7.2: 特征模板

单字符特征	$x_{-2}y_0, x_{-1}y_0, x_0y_0, x_1y_0, x_2y_0$ ^a
双字符特征	$x_{-1}x_0y_0, x_0x_1y_0, x_{-1}x_1y_0,$
三字符特征	$x_{-1}x_0x_1y_0$
马氏链特征	$y_{-1}y_0$

^a 下标 $\{-2, -1, 0, 1, 2\}$ 表示该字符位置到当前分析字符位置的相对位移。

用特征模板按先后次序在文本中每个位置上进行特征抽取，就可以抽取出所有的特征。

例如，对于句子“自然语言处理”和对应的标记序列“BEBEBE”，若当前分析字符是“语”，则 x_0y_0 抽取的特征为“语|B”， $x_{-1}x_0x_1y_0$ 对应的特征为“然语言|B”。表7.3给出了不同特征模板抽取的相应特征。

表 7.3: 不同模板抽取特征示例

模板	特征
x_0y_0	“自 B”，“然 E”，“语 B”，“言 E”，“处 B”，“理 E”
$x_{-1}x_0y_0$	“SOS自 B” ^a ，“自然 E”，“然语 B”，“语言 E”，“言处 B”，“处理 E”
$x_0x_1y_0$	“自然 B”，“然语 E”，“语言 B”，“言处 E”，“处理 B”，“理EOS E” ^b
$y_{-1}y_0$	“BE”，“EB”，“SOS-B”，“E-EOS”

^a SOS表示句子的开始。

^b EOS表示句子的结束。

7.3 基于无监督学习的中文分词

从词的定义上看，词是稳定的字的组合，因此在上下文中，相邻的字同时出现的次数越多，就越有可能构成一个词。因此相邻字的共现频率能够较好的反映其组成词的概率。可以对语料中相邻共现的各个字的组合的频度进行统计，计算它们的互现信息。定义两个字的互现信息，计算两个汉字的相邻共现概率。互现信息体现了汉字之间结合关系的紧密程度。当紧密程度高于某一个阈值时，便可认为此字组可能构成了一个词。这种方法只需对语料中的字组频度进行统计，不需要词典，也不需要标注语料。但这种方法也有一定的局限性，会经常抽出一些共现频度高、但并不是词的常用字组，例如“这一”、“之一”、“有的”、“我的”、“许多的”等，并且对常用词的识别精度差，时空开销大。

7.4 小结

实际应用的中文分词系统应该是利用结合上下文进行分词，同时可以结合少量的人工规则进行错误修正。

第八章 词性标注

词性标注对句子中的每一个词附上相应的词性。

8.1 词性

词性（Part-of-Speech, POS）指作为划分词类的根据的词的特点。现代汉语的词可以粗分为12类。

实词 名词、动词、形容词、数词、量词和代词

虚词 副词、介词、连词、助词、拟声词和叹词

但这样的划分对于后续的语义分析远远不够。因此不同的语料构建者都会对词性进行更细的划分。比如“名词”可以再分为“一般名词”、“专用名词”、“抽象名词”和“方位名词”等。“专用名词”可以再分为“人名”、“地名”、“机构名”等。

词性标注（POS Tagging）是给句子中每个词标记出最合适的词性。中文和英语的一个重要区别是中文没有词性变化。中文的每个词会有多种词性（比如“希望”即是名词又是动词）。但特定的使用场合下，比如一个句子中，每个词都有唯一确定的词性。即在若干词性候选项中选择一个合适的词性。

8.2 词性标注规范

中文词性标注目前还没有统一明确的标注规范。不同研究者都提出了自己的标注规范以及相应的数据集。

在众多词性标注数据集中，Penn Chinese TreeBank (CTB)¹是广泛使用的数据集。CTB数据集中定义的词性分为11个大类和33个小类，如表8.1所示。具体规范可参考文献Xia [2000]。

表 8.1: Chinese TreeBank 词性标记集合

类别	词性标记
动词，形容词 (4)	VA, VC, VE, VV.
名词 (3)	NR, NT, NN.
方位词 (1)	LC.
代词 (1)	PN.
限定词和数字 (3)	DT, CD, OD.
量词 (1)	M.
副词 (1)	AD.
介词 (1)	P.
连词 (2)	CC, CS.
虚词 (8)	DEC, DEG, DER, DEV, SP, AS, ETC, SP, MSP.
其他 (8)	IJ, ON, PU, JJ, FW, LB, SB, BA.

比如下面例子中，“DT”，“PN”，“VV”，“AS”，“JJ”，“DEG”，“NN”和“PU”都是CTB定义的词性标签。

去年/时间短语他/人称代词取得/动词了/时态词可喜/形容词的/结构助词进步/名词。/标点

在FNL P中，词性分为：动词、能愿动词、趋向动词、把动词、被动词、形谓词、形容词、副词、名词、方位词、人名、地名、机构名、时间短语、邮件、网址、型号名、实体名、疑问代词、指示代词、人称代词、量词、介词、数词、惯用词、限定词、连词、叹词、序数词、省略词、语气词、结构助词、时态词、标点、拟声词、表情词等。

8.3 词性标注

中文的词缺乏形态变化，不能直接从词的形态变化上来判别词的类别。并且大多数词是多义的，兼类现象严重。中文词性标注要更多的依赖语义，相同词在表达不同义项

¹ <http://www.cis.upenn.edu/~chinese/ctb.html>

时，其词性往往不一致的。因此通过查词典等简单的词性标注方法效果会比较差。

目前，有效的中文词性标注方法可以分为基于规则的方法和基于统计的方法两大类。基于规则的方法的局限性在于自然语言的复杂性，建立规则库需要大量的专家知识和很高的人工成本。基于统计学习的方法的局限性在于其严重依赖数据集的规模和质量。在近几年，由于人们可以通过较低成本获得高质量的数据集，基于统计学习的词性标注方法取得了较好的效果，并成为主流方法，常用的学习算法有隐马尔科夫模型（HMM）、最大熵模型（ME）、条件随机场（CRFs）等。

8.4 基于统计学习的词性标注方法

当一个词具有多个义项时，它相应的词性和特定场合下该词的实际语义密切相关。比如：“爱”这个词，可以有“动词”和“名词”两种词性。在下面两个句子里，“爱”分别具有不同的词性。

我/人称代词爱/动词 你/人称代词。/标点

你/人称代词给/动词我/人称代词的/结构助词爱/名词。/标点

8.5 基于序列标注的词性标注方法

8.6 中文分词和词性联合标注方法

第九章 命名实体识别

命名实体识别任务是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。在当今世界，随着计算机的普及以及互联网的迅猛发展，大量的信息以电子文档的形式呈现在人们面前。为了应对信息爆炸带来的严重挑战，人们迫切需要一些自动化的工具帮助他们在海量的信息源中迅速找到真正重要的信息。

命名实体识别是信息提取、问答系统、句法分析、机器翻译等应用领域的重要基础工具，在自然语言处理技术走向实用化的过程中占有重要地位。

一般来说，命名实体识别的任务就是识别出待处理文本中三大类（实体类、时间类和数字类）、七小类（人名、机构名、地名、时间、日期、货币和百分比）命名实体。

命名实体识别的过程通常包括两部分：（1）实体边界识别；（2）确定实体类别（人名、地名、机构名或其他）。

英语中的命名实体具有比较明显的形式标志（即实体中的每个词的第一个字母要大写），所以实体边界识别相对容易，任务的重点是确定实体的类别。和英语相比，汉语命名实体识别任务更加复杂，而且相对于实体类别标注子任务，实体边界的识别更加困难。汉语命名实体识别的难点主要存在于：

1. 汉语文本没有类似英文文本中空格之类的显式标示词的边界标示符，命名实体识别的第一步就是确定词的边界，即分词；
2. 汉语分词和命名实体识别互相影响；
3. 除了英语中定义的实体，外国人名译名和地名译名是存在于汉语中的两类特殊实体类型；
4. 现代汉语文本，尤其是网络汉语文本，常出现中英文交替使用，这时汉语命名实体识别的任务还包括识别其中的英文命名实体；

5. 不同的命名实体具有不同的内部特征，不可能用一个统一的模型来刻画所有的实体内部特征。

在命名实体中，时间词和数量词的识别相对容易，现行通用的是基于规则的方法；实体名（人名、地名和机构名）识别是研究的焦点。本软件工具主要对汉语人名（包括人名简称）、地名（包括地名简称）和机构名识别展开专项研究，利用大颗粒度特征（词性特征）和小颗粒度特征（词类[1]特征）相结合、统计模型和专家知识相结合的汉语命名实体识别模型。

第十章 句法分析概述

一种语言的语法是一系列规则的集合，人们通过这些规则来使用这种语言。

句法分析（Syntactic Analysis，或 Parsing）是指根据一种语言的语法，对句子（词串）进行分析，来判断这个句子是否符合该语言的语法，同时给出它的语法结构。对文本的处理过程中，句法分析处于一个十分重要的位置。基于句法分析的很多方法、模型在很多应用系统中被广泛的使用。

句法分析可以分为两个部分：一是如何定义一套符合语言学的基本原则，并且可以在计算机中表示、处理的句法描述规范。通过抽象的符号系统，对语言进行理论上的分析和表示。二是根据给定的句法描述规范，自动分析自然语言的句子。事实上，我们对句法分析并不陌生。句法分析不仅仅面向自然语言，也是我们各种人工语言（比如 XML 语言、HTML 语言以及各种程序语言）中不可缺少的一部分。人工语言和自然语言的一个重要区别是人工语言必须是无歧义的，而自然语言中充满大量的歧义。

语法（Grammar）是语言学的一个分支，主要研究语言的构成规则，包括词法（morphology）和句法（syntax）。词法指词的构成及变化规律；句法指短语和句子的组织规律，词之间的相互关系以及词在句中的功能和关系。

一般狭义上讲，句法是语法的一个组成部分，主要指句子或短语的构造规则，而语法则包含词、词性、句子、篇章、修辞等一切现象。但是在自然语言处理领域，人们一般不特别区分这两者的区别。很多文献中也用语法来指代句法。

10.1 语法理论

句法分析首先要明确自然语言的语法结构，对自然语言的语法结构进行形式化的定义。根据句子结构中最基本单元的不同划分，句法可分为两大类：

1. 短语结构语法 (Phrase Structure Grammar)

短语结构语法也叫**成分语法** (Constituency Grammar)，这种语法基于短语以及短语之间的成分关系来描述句子结构。

2. 依存语法 (Dependency Grammar)

依存语法是基于句子中词之间的依存关系来描述句子结构。

10.1.1 短语结构语法

短语结构语法的主要特点是使用短语和他们之间的成分关系来描述句子结构。对于一个句子来说，最基础的结构是“主谓”结构。一个句子可以分为两个成分构成：主语（名词短语）和谓语（动词短语）。每个成分又可以由一个或多个子成分构成。这样一个句子可以用树状结构来表示。

比如下面的句子：

这是一颗语法树。

在这个例句中，“这”是主语（名词短语），而“是一颗语法树”是谓语（动词短语）。作为主语的名词短语进一步可以由代词构成。作为谓语的动词短语可以进一步由动词 + 名词短语构成。图10.1给出了整个句子的短语结构语法树。在短语结构语法树中，叶子节点为句子中词，非叶子节点为短语或句子成分。图10.1中非叶子节点的标签使用了CTB语料中的定义[Xue et al., 2005]。比如，NP为名词短语，VP为动词短语等。

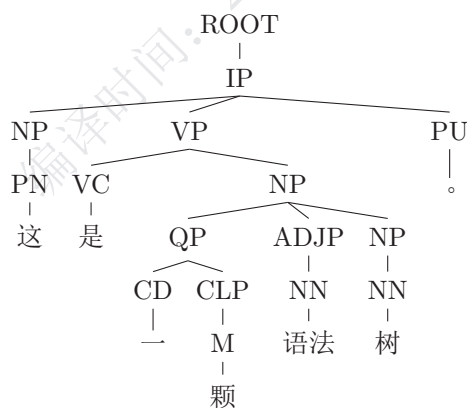


图 10.1: 短语结构语法示例

在汉语中，一般的语法成分包括：主语、谓语、宾语、定语、状语、补语等。

在自然语言的计算机处理中，成分句法主要是使用Chomsky的上下文无关语法。

10.1.2 依存语法

依存语法是基于依存关系来描述句子的语法结构。依存关系是句子中词与词之间的二元非对称关系 [Tesnière, 1959]。对存在依存关系的两个词而言，一个称为**中心词**，另一个称为**修饰词**。依存关系可以用图来表示。图中两个节点代表两个词，一条有向边从中心词指向修饰词，如下图所示。



对于一个句子，依存语法通常认为存在一个核心动词，这个动词作为整个句子的结构中心，也叫根节点。其他词都直接或间接地依赖于核心动词。这样，一个句子的依存结构也构成了一颗树，也是可以说是有向连通树，或有向非循环连通图。和短语结构语法不同，依存语法不包含短语和成分的语法单位，依存结构由词和它的修饰词构成。因此，依存结构比短语结构更加扁平化。

语言学家们对于依存语法的观点也有很多分歧，但一般都遵循下面四条公理 [Robinson, 1970]：

- **根节点唯一性**：一个句子中有且只有一个独立元素；
- **连通性**：所有其他元素都依赖别的元素；
- **非循环性**：任何一个元素都不能直接从属于两个或两个以上的元素；
- **投影性**：如果元素 A 直接从属于元素 B，那么 A 和 B 之间的元素 C 只能从属于 A、B 或 A 和 B 之间的其他元素。

通过这四条公理的限制，一个句子可以用一颗依存结构树来表示。但是，依存语法的投影性要求并不适合于所有语言。在德语、捷克语中就存在很多非投影的依存句法结构。

对于句子“这是一颗语法树。”，图10.2给出了它相应的依存结构语法树。其中核心动词依赖到 ROOT 节点，其他词都直接或间接的依赖到核心动词上。

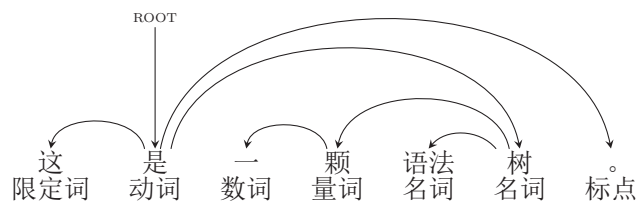


图 10.2: 依存语法示例

依存结构也可以用树状结构图来可视化，如图10.3所示。但是这种图不能很好的反映词的顺序，因此较少使用。通过对比图10.1和10.3，我们可以发现，依存树中的每一个节点都与句中的词相对应，没有短语和成分这些语法节点。因此，依存树的节点数和层数都比成分树要少。

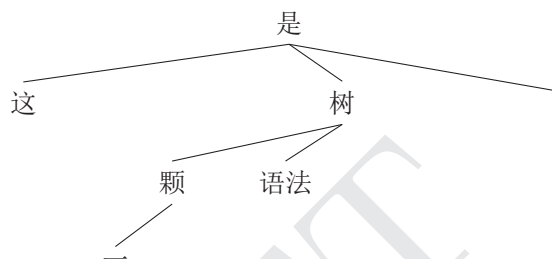
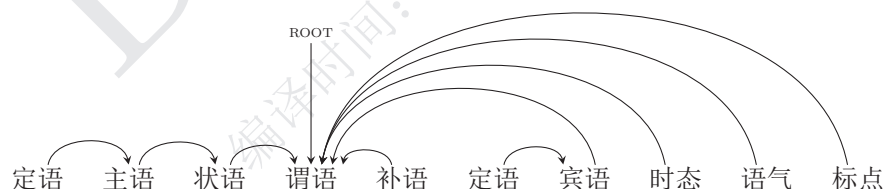


图 10.3: 树状结构表示的依存语法示例

中文依存句法

在现代汉语中，句子的一般结构为：（定）主+[状]谓<补>+（定）宾。用依存树表示为：



依存关系类型

依存关系类型也是依存语法的一个重要组成部分。对于依存结构树中的每一个依存关系，都隐含着一定的语法功能或语法关系。一般的语法功能包括主语、宾语、状语等。在不同的语言和依存语法中，这些功能也都不尽相同。我们把这些语法功能称为依存关系类型。

这些功能可以被标记在依存树中的每一条依存关系上，如图10.4所示。

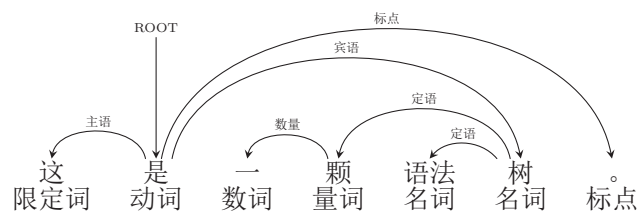


图 10.4: 带依存关系类型的依存语法树示例

和短语结构语法不同，依存语法中的语法功能是词和词之间的二元关系。而短语结构语法中语法功能是成分和成分之间的多元关系。

依存语法的缺陷

对于依存文法的语言学研究中，始终没有非常明确的定义什么是依存关系。因而在一个语言结构中讨论依存关系以及如何确定中心词和修饰词时，不同研究者都给出自己的一些标准。

Hudson [1990] 提出了一个比较有代表性的标准。

在一个依存关系结构 C 中，确定支配词 H 和从属词 M 的标准如下：

1. H 决定了 M 的语法类型，H 常常可以取代 M。
2. H 决定了 M 的语义类型，M 在语义上对 H 作了限制。
3. H 是必需的，M 是省略的。
4. H 决定了 M 是可选的还是省略的。
5. H 决定了 M 的形式。从属词的形式包括指示代词的单复数 (this, these)，人称代词的单复数、格等。
6. H 决定了 M 和 H 在句子中的相对位置。

从这些准则可以看出，依存关系涉及语法和语义这两个不同的层面，因的确很难给出一种符合所有标准的依存关系定义。基于这点，许多理论研究者主张需要对不同类型的依存关系区别对待。对于大多数依存结构而言，他们的依存关系是比较明显的。但也有些结构，在各种依存语法研究中如何定义他们的依存关系存在一定的分歧。

Melčuk [1988] 将依存关系分为词法的 (morphological)，句法的 (syntactic) 和语义的 (semantic)。

Nikula [1986] 则认为需要区分依存关系的向心结构 (endocentric construction) 和离心结构 (exocentric construction)¹。在向心结构中，删去从属词并不影响句子结构，即满足第一条准则，而在离心结构中，支配词无法直接取代整个结构，从属词是不可缺少的。比如，主谓、谓宾所构成的依存关系显然是以谓语词作为支配词的离心结构，而形容词修饰名词构成的依存关系显然是向心结构。

此外，下面两种结构也很难用依存结构描述：

1. 包含功能词的结构，这里功能词包括介词、连词、结构助词等。比如对于助动词，有的理论将助动词作为支配词，实际动词作为从属词，而有的理论则恰恰相反。
2. 并列结构。并列结构的每个元素在语义上是并列的，因而并没有对于任意两个元素，并没有显然的依存关系存在。对于这些结构，不同的依存文法理论会使用不同的假设和处理策略。

针对这些缺点，在 Tesnière 之后，很多语言学家对依存语法进行了很多改进，并演化出众多的语法理论，包括词语法 (Word Grammar) 等等。尽管这些语法各有侧重，相互之间有着比较大的差异，但也包含着共同的假设，其中最核心的是：句子结构有词汇节点构成，两个词汇节点之间可以通过依存关系这种二元非对称关系建立联系。

10.2 短语结构到依存结构的转换

很多逻辑家和语言学家通过研究表明，在一定的条件限制下，依存语法弱等价于上下文无关的短语结构语法 [Gaifman, 1965]。但是依存语法并不是短语结构语法的简单变种，两种语法还是存在很明显的区别。

- 短语在依存结构中是一个隐式的概念，很难从依存结构中找出短语。
- 依存语法中所有的结构都是向心结构，而短语结构语法中有向心结构和离心结构的区别。
- 依存结构中没有短语和成分的中间节点，因此它很难反映短语结构语法中的一元分支。比如名词短语仅由名词组成。

¹ Leonard Bloomfield 在《语言论》一书中提出著名的“向心结构”和“离心结构”概念。如果一个整体的组成部分中，只有一个部分跟整体的语法功能相同的结构，叫向心结构。如果没有一个部分跟整体的语法功能相同的结构，叫离心结构。

10.3 句法分析

语言的集合是无限的，而语法规则通常是有限的。人们通过这些语法规则来产生无限组合的语言。句法分析是语言生成的拟过程，是指根据一种语言的语法，对句子（词串）进行分析，来判断这个句子是否符合该语言的语法，同时给出它的语法结构。

人工语言一般都是建立在上下文无关语法的基础上的。有很多成熟的句法分析方法，比如自顶向下分析法、自底向上分析法。人工语言和自然语言的一个重要区别是人工语言必须是无歧义的，而自然语言中充满大量的歧义。这些分析方法在处理自然语言的歧义时都显得无能为力。在自然语言句法分析中，也存在很多语法规则。但是要人工来总结和整理这些规则是不可行的。因此，目前主流的句法分析都是使用统计学习的方法从大量的语料库中自动学习规则，同时给不同的规则赋予不同的权重。然后通过不同的策略来生成语法树。

目前，已有很多成熟的基于统计学习的句法分析方法，这些方法会在后续的章节中陆续介绍。

第十一章 依存句法分析

理解一个句子，就是找出句子中各个词之间的所有联系。

—Lucien Tesnière

依存文法在语言学上的研究一般认为开始于法国语言学家 Lucien Tesnière 提出的“图式”（stemma）结构。Tesnière 认为每个句子都拥有一个有组织的完整的内在结构，构成这个结构的基本成分是句子中的词，和词与词之间的联系。词和词之间的联系构成了句子的结构。词与词之间的联系是一种结构性的依存关系，即**支配词**和**从属词**。在很多文献中，支配词又被称为中心词、管辖词等，从属词又被称为修饰词、被支配词等¹。

依存句法是描述句子内部各个词之间的**依存关系**。与成分句法不同，结构没有非终结点，词与词之间直接发生依存关系，构成一个依存对，其中一个是**支配词**，另一个是**从属词**，也叫**修饰词**。依存关系是非对称的，可以用有向边来表示。由从属词指向支配词。一个从属词只能依赖到一个支配词。而一个支配词可以支配多个从属词。一个从属词可以是其它词的支配词，一个支配词也可以是其它词的从属词。一个句子中不能含有循环。这样依存句法结构也可以看出是一颗依存句法树。

此外，如果约束依存树中不存在交叉边，这颗树称为**投影依存树**。否则，若依存树中存在交叉边，这颗树称为**非投影依存树**。

一般一个句子包含一个**核心词**，也就是句子中的核心谓语动词、名词或形谓词，对主语加以陈述，说明主语“做什么”、“怎么样”或者“是什么”。核心词作为依存树的根节点。

每个依存对可以属于不同的**关系类型**，表示该依存对中的两个词之间存在什么样的依存关系。关系类型包括：主语、定语、补语等。

¹ 在英文文献中，也存在不同的术语表示。支配词可写为 governor、head 和 regent 等，从属词写为 dependent、modifier、subordinate 等。

目前，在基于统计的句法分析方法中，依存句法有着下面几个优点，并逐渐受到研究人员的重视。

- 形式简单，易于理解；
- 易于构造基于统计模型的高效、准确的分析模型，适合大规模的文本处理；
- 易于理解，具有更好的心理对照，标注难度相对较低；
- 更容易进行从句法到语义的转换。

这些特点使得依存句法分析方式在很多实际应用中被大量使用。尤其在处理互联网级别的大规模数据时，依存句法分析方法显示出很强的优势。

目前主流的基于统计的依存句法分析方法中可以分为两种类型：

- 基于转换的方法 [Nivre, 2003, Yamada and Matsumoto, 2003]
- 基于图的方法 [McDonald et al., 2005]

基于转换的方法将句法分析过程分解成一系列的动作（比如移进、规约）。当一个动作完成后，通过一个分类器来决定下一个动作。最终形成一个依存树。这种方法采用贪婪的决策来选择下一个动作。因此，最终形成的依存树不一定是最好的。

基于图的方法将依存树分解为一系列边（或其他子结构）的组合，然后选择一个最高得分的组合。基于图的方法通常有两个阶段组成。一是打分阶段，使用一个打分函数对所有可能的边或其他子结构进行打分；二是解码阶段，通过组合优化算法（比如最小生成树算法 [Chu and Liu, 1965, Edmonds, 1967]）从所有的组合中选择一个最高得分的树。

对于依存文法分析，也有这两种类型的方法：句法驱动的分析方法和数据驱动的分析方法。句法驱动的依存文法分析方法需要在给定文法的基础上对句子进行分析，因而这些方法经常伴随着对于依存文法形式化定义的研究。

然而依存概念的模糊性造成了依存文法理论的多样性，因而现在的依存文法分析研究主要以数据驱动。数据驱动的依存文法分析可以概括为下面3步：

1. 建模：如何计算输入句子的一个依存结构的分数
2. 参数估计：利用标注数据或者来标注数据估计模型中的参数
3. 推断：搜索对于当前的输入句子，分数最大的一个依存结构

目前，基于统计的依存句法分析主要分为两类：

目前基于数据驱动的依存语法分析方法又分为两类：基于图的依存句法分析 [McDonald et al., 2005] 和基于转换的依存句法分析 [Nivre and Nilsson, 2005]。其代表性成果分别为：MaltParser 和 MSTParser。

在基于图的依存句法分析中，通过动态规划或近似算法，得到一棵满足依存关系约束的全局最优的依存树。时间复杂至少为 $O(n^3)$ 。

在基于转换的依存句法分析中，词汇之间的依存关系是用贪婪的策略逐步决定的。由输入句子构造初始状态，在当前状态的基础上，执行某一动作，转换到一个新的状态。最终的状态包括了一棵完整的依存分析树。通过定义不同的状态和转换动作，分别由不同的分析算法。由于使用了贪婪的策略，时间复杂度一般为 $O(n)$ 或 $O(n^2)$ 。

11.1 基于转换的依存句法分析

在基于转换的分析方法中，依存分析被看作是对输入句子执行若干动作，由这些动作建立起句子中词与词之间的联系。每一个动作都将当前的分析状态转换到新的状态。基于转换的分析方法并不搜索全局最优的动作序列，而是采用贪婪的策略，根据当前状态选择局部最优的动作，一个动作一旦执行就不会在更改，因而又称确定性分析方法。

11.1.1 Yamada 算法

Yamada 算法 [Yamada and Matsumoto, 2003] 是基于移进-规约算法的确定性分析算法，它从左到右多遍地扫描检查输入的句子（或已部分构建的依存结构序列）来构建局部的依存结构，最终形成一颗完整的依存树。

在从左到右的每一步分析过程中，Yamada 算法把当前分析位置的相邻词对成为焦点词¹。相应地，把当前分析的位置称为焦点位置。然后通过分类器来预测一个分析动作：① 为两个焦点词建立一条依存边，或者②不构建依存边而仅仅将焦点位置右移一位。

初始状态是输入句子对应的词序列，并且设置焦点词为序列最左边的第一个和第二个词。Yamada 算法通过一系列的分析动作来构建一颗完整的依存树。因为依存边是有方向的，构建依存边的动作可以细分为左右两个方向。因此，在 Yamada 算法中有三个基本动作，表示为：Shift、Left 和 Right。具体定义如下：

Shift 两个焦点词之间不建立依存边，将焦点位置往右移动一位。如下图所示，方框里的词为焦点词。

¹ 这里的“词”称为“节点”更合适些。一是因为待处理的输入序列中每一个元素可以是词，也可以是部分依存结构，二是因为输入序列不但包含词，也经常包含词性等信息。

我 正在 吃苹果。 \Rightarrow 我 正在 吃 苹果。

Right 在两个焦点词之间建立一条依存边，方向是从右到左。右焦点词作为左焦点词的父节点。并将左焦点词从序列中删去，并更新左焦点词。

我 正在 吃 苹果。 \Rightarrow 我 吃 苹果。

\nwarrow
正在

Left 在两个焦点词之间建立一条依存边，方向是从左到右。左焦点词作为右焦点词的父节点。将右焦点词从序列中删去，并更新右焦点词。需要指出的是，当原右焦点词为句子的末尾时，焦点词重新返回到序列最左边的第一第二个词。

正在 \nearrow

吃 苹果 。 \Rightarrow

吃

\nwarrow

。

正在 \nwarrow

吃

\searrow

苹果

Yamada 算法的流程可以概况为不断重复下面两个步骤：

1. 根据焦点词的上下文信息，提取特征，用分类器来估计一个适合的分析动作；
2. 执行分析动作来构造依存树。如果在一遍扫描中没有 Left 或 Right 动作，那么选择一个最大可能的候选的 Left 或 Right 动作来执行。

Yamada 算法的伪代码在算法11.1。其中， T 表示长度为 m 的序列，其中每一个词 $t_m (m \leq n)$ 表示在分析过程中部分构造子树的根节点。初始化时，每个词对应输入句子的一个词，整个序列的长度为 n 。

在分析过程中，对于一个焦点词对 $\langle t_i, t_{i+1} \rangle$ ，根据其上下文特征，用分类器预测一个分析动作 $y_i \in \{Right, Left, Shift\}$ 。同时，如果 $y_i = Shift$ ，用 y'_i 记录一个候选动作，这个候选动作作为分类器预测的第二好的结果，其相应的得分为 s'_i 。

在一遍扫描中，如果每一个动作 $y_i = Shift$ for $i = 1, \dots, T-1$ ，也就是说没有构建

邱锡鹏：《自然语言处理原理与实现（草稿）》

<http://www.fnlp.org/book/>

任何依存边。我们需要选择一个最大可能的候选动作 y'_j ，其中 $j = \arg \max_i s'_i$ 。

算法 11.1: Yamada 依存句法分析算法

输入: 句子: $(w_1, p_1), (w_2, p_2), \dots, (w_n, p_n)$

输出: 依存树 T , ($|T| = 1$)

```

1  初始化:
2   $i = 1$ ;
3   $T = \{(w_1, p_1), (w_2, p_2), \dots, (w_n, p_n)\}$ ;
4   $\text{no\_construction} = \text{true}$ ;
5  while  $|T| \geq 1$  do
6      if  $i == |T|$  then
7          if  $\text{no\_construction} == \text{true}$  then
8              找到候选动作  $y'_j$ , 其中  $j = \arg \max_i s'_i$ ;
9              在位置  $j$  执行动作  $y'_j$ ;
10              $\text{no\_construction} = \text{false}$ ;
11              $i = j$ ;
12         end
13     end
14     构建上下文特征  $x$ ;
15     预测分析动作  $y$ ;
16     在位置  $i$  执行动作  $y$ ;
17     if  $y == \text{left or right}$  then
18          $\text{no\_construction} = \text{false}$ ;
19     else
20         记录第二大得分的动作  $y'_i$  和它相应的得分  $s'_i$ ;
21     end
22 end
23 返回  $T$ ;
```

对于长度为 n 的句子, Yamada 算法在最坏的情况下需要 $n1$ 遍扫描才能构建一颗完整的依存树, 复杂度为 $O(n^2)$ 。但是在实际应用中, 该算法一般是线性时间复杂度。

上下文特征

在 Yamada 算法中, 在某个状态下如何预测下一步的动作是十分关键的。这个预测是通过一个分类器完成的, 如何使得预测尽可能准确就依赖两个因素。一是分类器选取,

不同的分类器（比如SVM、感知器等）会对分类准确率有很大影响。二是特征选取，特征的好坏是最终预测效果的关键因素所在。

和其他任务相似，这里的特征也是从一个上下文窗口中抽取。假设在序列 T 中当前的左右焦点词的位置分别为 i 和 $i+1$ 。我们分别从左右各取长度为 (l, r) 的窗口中词。

这样一个特征可以表示为一个三元组 (p, k, v) ，这里 p 是相对焦点词的位置， k 是特征类型， v 特征值。如果 $p < 0$ ，这个特征是焦点词左边的词， $p = 0-$ ； $0+$ 表示左右两个焦点词。 $p > 0$ 表示焦点词右边的词。特征类型 k 如表11.1所示。

表 11.1: Yamada 算法中的特征

类型	说明
pos	词性
lex	词
ch-L-pos	左子节点的词性
ch-L-lex	左子节点的词
ch-R-pos	右子节点的词性
ch-R-lex	右子节点的词

对于图11.1中的例句，方框中的词为焦点词。根据表11.1，我们可以抽取如下特征： $(-2, \text{pos}, \text{代词})$ ， $(-2, \text{lex}, \text{我})$ ， $(-1, \text{pos}, \text{动词})$ ， $(-1, \text{lex}, \text{喜欢})$ ， $(-1, \text{ch-R-pos}, \text{副词})$ ， $(-1, \text{ch-R-lex}, \text{非常})$ ， $(0-, \text{pos}, \text{动词})$ ， $(0-, \text{lex}, \text{看})$ ， $(0+, \text{pos}, \text{名词})$ ， $(0+, \text{lex}, \text{科幻})$ ， $(+1, \text{pos}, \text{名词})$ ， $(+1, \text{lex}, \text{电影})$ ， $(+2, \text{pos}, \text{标点})$ ， $(+2, \text{lex}, \text{。})$ 。

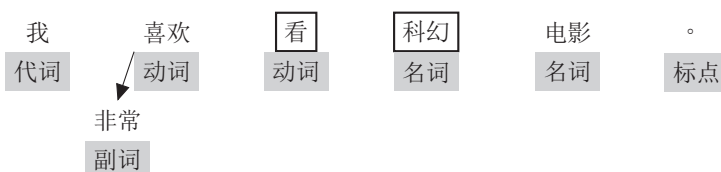


图 11.1: Yamada 特征抽取示例

这些三元组特征可以用词袋模型转换为向量表示，并作为分类器的输入。表11.1所列举的特征都可以认为是一元特征，这些特征也可以进行组合来形成更复杂的特征。

11.1.2 Nivre 算法

11.2 基于图的方法

11.3 评价指标

为了比较不同算法的性能，在依存句法分析中通常使用下面的评价指标：

1. 依存准确率 (Dependency Accuracy, DA): 在所有非根词 (排除标点) 中，被指定正确的中心词的词所占的比例。

$$DA = \frac{\text{正确识别的支配词个数}}{\text{识别的支配词个数}} \quad (11.1)$$

2. 根节点准确率 (Root Accuracy, RA): 在所有根节点中，被正确识别的比例。

$$RA = \frac{\text{正确识别的根节点个数}}{\text{根节点总数}} \quad (11.2)$$

3. 整句准确率 (Complete Accuracy, CA): 整句的依存关系全部正确的句子所占的比例。

$$CA = \frac{\text{整句的依存关系全部正确的句子数}}{\text{句子总数}} \quad (11.3)$$

4. 有标记依存关系的准确率 (Labeled Attachment Score, LAS)

$$LAS = \frac{\text{正确识别其支配词和关系类型的词的个数}}{\text{总词数}} \quad (11.4)$$

5. 无标记依存关系的准确率 (Unlabeled Attachment Score, UAS):

$$UAS = \frac{\text{正确识别其支配词的词的个数}}{\text{总词数}} \quad (11.5)$$

在同时识别依赖关系类型的评测是，一般使用和两种评测指标。这两种评测指标都不区分根词和非根词。

11.4 小结

基于图的方法，如果使用一阶模型和最小有向生成树算法 [Chu and Liu, 1965, Edmonds, 1967]，时间复杂度为 $O(n^2)$ 。

虽然基于转换的方法通常比基于图的方法更快，但是这类方法的一个问题就是错误传播。因此，他们的准确率一般略低于基于图的方法 [Buchholz and Marsi, 2006, Nilsson et al., 2007]。但是最近的一些研究工作通过不同的方法来改进基于转换的方法。通过这些改进，基于转换的方法在很多语言上已经可以达到最先进的水平。

第十二章 词的语义表示与知识库

上下文相同的词，其意义也相似。—Harris

自然语言处理的目标是让计算机理解语言所表示的意义。

知网（HowNet）是一个以汉语和英语的词语所代表的概念为描述对象，以揭示概念与概念之间以及概念所具有的属性之间的关系为基本内容的常识知识库。

HowNet 中的语义关系

1. 上下位关系
2. 同义关系
3. 反义关系
4. 对义关系
5. 部件-整体关系
6. 属性-宿主关系
7. 材料-成品关系
8. 角色-事件关系

第十三章 指代消解

指代是人类语言中带有普遍性的现象。任何语言都有一些特定的语言手段用来指示或者代替语境中的某一成分，使话语与一定的人物、事物、时间、空间、事件等发生直接联系。指代消解是自然语言处理中的关键难题之一。

指代是一个语言学现象，指为了语言的简洁性，经常使用代词或是其他名词来代替已经出现过（或隐含）的语言实体（对象或事件），以避免语句结构过于冗余和语意不够清晰。

我们先来看个例子：

李明给李明的妈妈打电话说李明爱李明的妈妈。

读起来是不是觉得绕口？这句话中有两个人物，“李明”和“李明的妈妈”，在特定的场合下，我们都可以用代词来替换以减少冗余性。这句话可以改写为：

李明给他的妈妈打电话说他爱她。

如果这句话出现的情景中没有更多的人物出现，“他”就指代“李明”，而“她”就“他的妈妈”，更进一步指代“李明的妈妈”。这虽然给语言带来简洁性，但同时也增加了一定的模糊成分。因此，不是所有的对象都可以进行指代。

1. 李明给他的爸爸打电话说他爱她。
2. 李明给他的爸爸打电话说他爱他。
3. 李明给他的爸爸打电话说他爱他的妈妈。
4. 李明给他的爸爸打电话说他爱他自己。
5. 李明给他的爸爸打电话说他爱他的爸爸。

这个例子中，第1、2句都有一定的模糊成分。

在生活中也有很多故意使用指代引起的模糊性。比如商家进行使用“买一送一”的营销策略，其实所赠送的商品与消费者购买的商品根本不相同。

13.1 基本概念

下面先介绍几个基本概念。

指代表达式 (referring expression) 是指标识一个特定人、地方、其他物体或事件的词或短语，一般是名词、名词短语或代词。在语言学中，识别指代以及相应的指代对象是语用学研究的范畴。语用学是研究如何通过语境来理解和使用语言，涉及到哲学、认知学以及心理学等多个领域。

在语法角度，不是所有的名词都是指代表达式。比如“白马非马”，这里“白马”和“马”都指一个种类，而不是一个具体的对象。因此它们都不是指代表达式。“那匹马是白马”，这里“那匹马”是一个指代表达式，指一个特定的对象。

在英文中，指代表达式经常包含定冠词“the”或指示代词“that”、“those”等。在中文中，指代表达式经常包含“这”、“那”等。

如果要理解文本，就必须先弄明白文本中出现的指代表达式之间的关系。指代表达式之间的关系主要分为三种：

- **共指**(coreference) 如果两个指代表达式a和b指向真实世界中的同一参照体，并且这种指代脱离上下文仍然成立。我们就说a和b是共指关系。共指关系等价关系。
- **回指**(anaphora)，也称指示性指代或照应是指当前的照应语与上文出现的词、短语或句子(句群)存在密切的语义关联性。指代依存于上下文语义中，在不同的语言环境中可能指代不同的实体，具有非对称性和非传递性。回指就是代词的先行语在代词前面，预指就是代词的先行语在代词后面。

•

回指和共指存在很大的交集，又不互相包含。

如果指代表达式a和b指向同一个对象，

众所周知，人们为了避免重复，习惯用代词、称谓和缩略语来指代前面提到的实体全称。例如，在文章开始处会写“哈尔滨工业大学”，后面可能会说“哈工大”、“工大”等，还会提到“这所大学”、“她”等。这种现象称为共指现象。虽然人们可以毫无困难的区分文章中实体的不同称谓，所谓共指消解就是将现实世界中同一实体的不同描述合

并到一起的过程。如上几个描述都是现实世界中“哈尔滨工业大学”的不同体现。在某种意义上说，共指在自然语言中起到了超链接的作用。一方面，它使得文章作者在撰写文章时体现了一定的风格和篇章的连贯性。另一方面，共指使得自然语言理解机制中增加了一种新的模糊成分。

照应语(Anaphor)

是消解过程中当前考察的用于指向的表述对象。先行语(Antecedent)是被指向的表述对象。以上几个概念主要的区别就在于照应语、先行语的类型以及二者之间的位置关系话语中提及了某个事物后，当再次论及这个事物的时候会采用各种方式来进行上下文的照应。

一般情况下，指代可以分为两种：回指(cataphora)和共指。

在语言学，照应是一个类型的表达式，其参考取决于另一个参照元素。例如，在句子中的“萨莉首选自己的公司”，“她”是一个照应的表达，它是共指的表达主体地位。通常，一个照应表达形式或一些其他类型的指示词例如，代词指其前身表示。长期照应语，英语的奇异变种，有时也被用于指定一人使用：“照应语是一种语言的实体，这表明一些其他语言的实体在同一文本中引用领带。”[1] 照应是一个重要的概念，不同的原因，在不同的层次上。首先，照应表示：“话语是如何建造和维护”。其次，上档次的句子，指代不同的语法元素绑定在一起。第三，计算语言学照应提出了挑战自然语言处理，可能会很困难，因为鉴定的参考。四，照应“告诉我们一些关于语言是如何理解和处理”，[2] 这是指代语言学认知心理学感兴趣的领域有关

预指与回指相反。是对下文将要出现的内容或者词项的指代。它总是替代项在前，被替代项在后。

1. 人称代词 人称代词的作用在于使语言形式简洁明了、行文紧凑。名词和人称代词之间的照应又可以使文章前后呼应、意义连贯。第三人称代词，在汉语中有“他、她、它、他们、它们、她们”，在英语中有“he / she / it / they / him / her / it / them”
2. 指示代词
3. 普通名词短语
4. 间接指代 (indirect anaphor) 指一个短语和它的先行语并不是严格意义上的指代关系。但是根据人们的常识可以推断出该短语的实际指代对象。

[这个手机]很好，[屏幕]很大，[分辨率]也很高。

这里，“屏幕”是“这个手机”的部件，二者是整体与部分的关系，实际上指“这个手机的屏幕”。“分辨率”是指“这个手机的屏幕的分辨率”。此时“屏幕”成为先行语，二者是对象与属性的关系。

5. **零指代** (zero anaphor) 当回指在语流上没有任何的形式体现时, 就是零指代或零形回指。

有一个笑话:

老婆给当程序员的老公打电话: 下班顺路买一斤包子带回来, 如果看到卖西瓜的, 买一个。当晚, 程序员老公手捧一个包子进了家门...老婆怒道: 你怎么就买了一个包子?! 老公答曰: 因为看到了卖西瓜的。

这里, 这里“一个”没有明确的修饰对象, 可以是一个西瓜, 一个包子, 一个卖西瓜的, 或者一个任意对象。

在自然语言处理中, 涉及指代的任务有两大类: 一类是从文本中识别出指代表达式所指代的对象, 称为指代消解; 另一类是在自然语言生成中, 创建指代表达式使得生成的句子更加通顺和接近人类所说的语言, 称为指代生成 (referring expression generation)。指代生成是指代消解的逆过程。

每种类型的指代词都有自己的特点, 很难采用单一方法解决各种指代。比如: 从距离上分析, 代名词和其先行语的距离不会太远, 而有定描述先行语和指代词的距离则较远。A

13.2 指代消解

指代消解 (coreference resolution) 在信息抽取中起着重要作用, 在自然语言接口、机器翻译、文本摘要和问答系统等应用以及篇章理解中也很关键。通常, 相同信息会在同一文本中出现若干次, 为了保证, 文本的概念关联性往往通过指代关系来刻画。因此, 有必要把这些指代互相联系起来, 实现相关信息的融合, 以获得相应信息在该文本中的完整描述。

如果让计算机来理解时。但是对于计算机来说, 这仍旧是一项非常困难的问题。

目前, 大部分指代消解算法都有如下流程:

1. 找到指代表达式 a, 并在邻近的段落或句子中寻找候选先行词。
2. 通过一系列的一致性检查, 去除不满足要求的候选词。
3. 通过一定的规则或算法对每个候选词进行打分。
4. 挑选最高得到的候选词作为 a 的先行词。

要进行指代消解, 必须进行词性标注和实体名识别等词法分析和浅层句法分析 (名词短语识别), 有些方法还需要依赖句法分析和知识库。

13.2.1 Hobbs 算法

最早最简单的指代消解算法是Hobbs 算法 [Hobbs, 1978], 也叫朴素算法。Hobbs 算法主要利用句法

Hobbs 算法的要点如下:

- 对于两个候选的先行词 a 和 b, 如果在搜索空间中 a 比 b 出现得早, 那么优先选择 a。
- 所有先行词的显著性都不相同。
- 不能处理歧义词

Converse [2005] Ge et al. [1998] Mitkov [1999]

句子“我们把香蕉给猴子, 因为它们饿了”和“我们把香蕉给猴子, 因为它们熟透了”有同样的结构。但是代词“它们”在第一句中指的是“猴子”, 在第二句中指的是“香蕉”。如果不了解猴子和香蕉的属性, 无法区分。

13.2.2 Lappin & Leass 算法

Lappin and Leass [1994] 提出了一种经典的RAP 算法 (Resolution of Anaphora Procedure)。

13.2.3 Kennedy & Boguraev 算法

由于Lappin and Leass [1994] 依赖句法分析的结果, 而目前 Kennedy and Boguraev [1996]

13.2.4 特征

在基于机器学习的指代消解方法中, 特征的选择对分类效果影响显著, 因此应选择能够有效判断两个名词短语间是否具有指代关系的特征。

不同语言的指代规则不同, 因此相应的特征也要针对不同语言进行定制。常见的特征如下:

词汇特征

中心词匹配 (STR—MATCH): 可能的取值为0或1。如果I和J的中心词完全匹配, 特征值取1, 否则取0。名词短语的中心词是名词短语的主体, 它只被其他成分修饰, 不修饰其他名词。例如名词短语“这名嫌犯”与“其中一名嫌犯”的中心词都为“嫌犯”, 它们相匹配, 对应特征值为1。(b) 别名 (ALIAS): 可能的取值为0或1。如果J是I的别名, 特征值取1, 否则取0。为了避免重复和描述的方便, 中文中别名的使用相当普遍, 别名的形式也灵活多变。有的别名是原名的一个子串, 如“广州市妇联”和“市妇联”; 有的别名是原名的抽取匹配, 如“朝鲜中央通讯社”和“朝中社”; 有的别名只有部分词存在于原名中, 如“哥伦比亚”和“哥国”。本文认为, 如果两个名词短语都为专有名词, 且一个是另一个的子串或抽取子串, 则认为是别名关系。所谓抽取子串是形如串A=“ab—ede”, 串B=“bd”, 则B是A的抽取子串。b) 语法特征 (a) I为人称代词 (I—PRONOUN): 可能的取值为0或1。如果I是人称代词, 特征值取1, 否则取0。汉语中主要的人称代词有“我、你、您、他、她、它”及其复数形式。此外, 还有非常特殊的反身代词“自己”“自我”“本人”和常用于指人的“大家”“各位”“其”等。(b) J为人称代词 (J—PRONOUN): 可能的取值为0或1。定义如I—PRONOUN。(c) I为专有名词 (I—PROPER): I如果是专有名词, 该特征值取1, 否则取0。专有名词包括人名、地名、组织名等专用名词。这个特征值可直接根据命名实体识别的结果来确定。(d) J为专有名词 (J—PROPER): 同I—PROPER。(e) 指示性名词 (DEN—NP): 可能的取值为0或1。J是指示性名词, 该特征值取1, 否则取0。指示性名词即为“J形这(些) / 那(些) / 谢其f也 / 有些 / (+量词)+名词”类型。(f) 数量一致 (NUMBER): 可能的取值为0、1或0.5。若I和J满足单复数一致, 取1, 不一致, 取0; 若两者有一个为单复数未知, 则取0.5。这一属性值可根据明显的单复数搭配词语进行判别, 若“各、(一大)群、许多、两者、全体、所有的+名词”或“大于1的数词(+量词)+名词”结构, 判断名词为复数; 若“1 / 一 / 每 / 这+(量词)+名词”结构, 判断为单数; 其他情况为单数未知。

性别一致 (GENDER)

可能的取值为0、1或0.5。‘如果I和J的性别一致, 特征值取1, 不一致, 特征值取0; 当其中一个的性别特征未知时, 特征值取0.5。对于普通名词, 主要通过明显的性别指示词进行识别。如含“先生、男(士)、丈夫、父、兄、儿子”等明显标志的认为是男性, 含“小姐、女(士)、母、姐、妈、妇、婆”等明显标志的认为是女性。对于人称代词, “他(们)”对应男性, 而“她(们)”则对应女性, 其他人称代词为性别未知。

对于中文人名, 很难通过名字确定地推测出性别。但是可以通过统计来发现的用字绝大多数遵循一定的规律, 喜欢使用具有性别指示性的字。对于两字的姓名, 性别往往与最后一字有关; 对于三字或复姓的四字姓名, 性别可能与最后一字或后两字有关, 有

的还与倒数第二字有关。因此，对于二字姓名，本文提取最后一字作为特征；对于三字或四字姓名，本文提取最后一字、最后两字和倒数第二字这三个特征。把35万人名中的32万作为训练数据，使用最大熵算法，生成训练模型，使用剩下的3万(左右)人名作为测试。对于测试中的每一个人名，首先查阅男性名用字字典和女性名用字字典，如果在字典中查到，则判断为相应性别；否则，生成特征向量，根据生成的最大熵模型作性别预测。实验的准确率为89.12%(其中测试总数为29 732个，判断正确的数为26 497个)。

(h) 同位语 (APPOSITIVE): 可能的取值为0或1。如果J是I的同位语，特征值取1，否则取0。在中文当中，如果以一个普通名词短语开头，后面紧跟一个专有名词短语，或者，如果以一个专有名词短语开头，后面紧跟一个普通名称短语，中间没有空格和标点符号，那么这样的两个名词短语是同位语关系。例如，“[美国总统][克林顿]上星期到朝鲜访问”，那么“美国总统”和“克林顿”就是一对同位语。c) 语义特征语义类别一致 (SEMCLASS): 可能的取值为0或1。若I和J满足语义类别一致，取1，不一致，取0；若两者有一个语义类别未知，也取1。d) 距离特征距离 (DIST): 该特征的值为I与J相隔的句子数。对文档中的每一个句子，从第一句开始依次顺序编号。对于I和J，分别得到它们所在的句子编号，句子号之差的绝对值即为该特征的值。距离是指代消解中的一个很重要的特征。根据人的思维特点，照应语一般会指向刚刚提及的某一事物，尤其是人称代词和指示代词的指代，它们之间的距离更近。本文对ACE2005中 broadest news 的指代距离进行了统计，统计结果如表1所示。从表中可以看出，对于代词的指代，4句以内占97.7%，对于名词短语指代，8句以内的占98.0%。因此在构建照应语的候选先行语集时，不必一直搜寻至篇章开头，可以对搜寻距离作限制，从而减小候选先行语集的元素数。对代词和非代词指代，距离分别限制在4句和8句内。

参考文献和深入阅读

共指消解是传统的研究方向，见著于二十世纪三十年代，是自然语言处理、机器翻译、信息抽取、信息检索等领域的关键技术之一。经过起初的蓬勃发展，于七十年代达到高潮，经历八十年代的低谷后，重新在九十年代初复兴。

近20年来，这方面的研究受到了格外的关注，许多重要的会议都设立了共指消解的专题会议，2001年 Computational Linguistics 学报还出版了指代消解的专辑，在1996、1997年的 MUC (Message Understanding Conference) 评测会议上被列为评测内容之一。DAARC从96年到2006年共举行了五次，专门讨论指代消解。2000年开始的 ACE (Automatic Content Extraction) 评测中共指消解也是重要内容之一。2006年11月到2007年3月，英国伍尔佛汉普敦大学发起了一个名为指代消解练习 ARE (Anaphora Resolution Exercise) 的共指消解评测。

http://en.wikipedia.org/wiki/Referring_expression

A lot of progress in supervised coreference modeling the mention-pair model is theoretically unappealing it makes coreference decisions based on only two NPs

The cluster-ranking model resembles Lappin & Leass’ s (1994) heuristic pronoun resolver narrows the gap between the sophistication of heuristic-based coref models and the simplicity of learning-based coref models

Ng [2010]

<http://plato.stanford.edu/entries/anaphora/> [http://en.wikipedia.org/wiki/Anaphora_\(linguistics\)](http://en.wikipedia.org/wiki/Anaphora_(linguistics))

第十四章 概率主题模型

概率主题模型是一系列旨在发现隐藏在大规模文档中的主题结构的算法。

在第三章（第22页）中，我们介绍了一元语言模型。给定了词的概率分布之后，我们可以通过如下的方法产生一篇长度为 N 的文档：

1. 根据词的概率分布，随机抽取一个词；
2. 重复（1），直到产生 N 个词。

这种产生文档的方法也叫作**生成式模型**。在一元语言模型中，产生的每一个词概率都是独立的，和这个词出现在文档的位置无关，也和这个词的前面和后面的词无关。

通过这方法我们可以产生很多篇文档。但是，每一篇文档都是从相同的词分布中产生的，即使在不考虑词序的情况下，这样生成的文档也是杂乱无章的，这和人们“创作”一篇文章的过程是不相符的。当一个作者“创作”一篇文章时，他首先会确定一个或几个主题，然后用和文章主题相关的词来写一篇文章。比如，在一篇讨论足球的文章里，会出现很多“比赛”、“战术”等词，而很少会出现“股票”以及“银行”等词；相反在讨论经济的文章里，出现“股票”以及“银行”等词的可能性要比“比赛”、“战术”等词大很多。

一个改进的方法是引入一个表示主题的离散随机变量，每个主题有一个不同的词分布。我们可以通过如下的方法产生一篇长度为 N 的文档：

1. 根据主题的概率分布，随机抽取一个主题；
2. 根据该主题对应的词分布，随机抽取一个词；
3. 重复（2）直到产生 N 个词。

这个模型称为**一元混合语言模型**。一元语言模型可以看作是只有一个主题的一元混合语言模型。

14.1 概率主题模型

对于语料库中的每篇文档，LDA 定义了如下生成过程（generative process）：

1. 对每一篇文档，从主题分布中抽取一个主题；
2. 从上述被抽到的主题所对应的单词分布中抽取一个单词；
3. 重复上述过程直至遍历文档中的每一个单词。

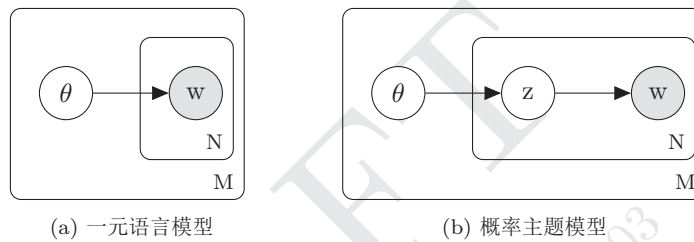


图 14.1: 一元语言模型和概率主题模型对比

这个图模型表示法也称作“盘子表示法”（plate notation）。图中的阴影圆圈表示可观测变量（observed variable），非阴影圆圈表示潜在变量（latent variable），箭头表示两变量间的条件依赖性（conditional dependency），方框表示重复抽样，重复次数在方框的右下角。

LDA 是一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。由于 Dirichlet 分布随机向量各分量间的弱相关性（之所以还有点“相关”，是因为各分量之和必须为 1），使得我们假想的潜在主题之间也几乎是不相关的，这与很多实际问题并不相符，从而造成了 LDA 的又一个遗留问题。

该模型有两个参数需要推断（infer）：一个是“文档-主题”分布，另外是 T 个“主题-单词”分布。通过学习（learn）这两个参数，我们可以知道文档作者感兴趣的主题，以及每篇文档所涵盖的主题比例等。推断方法主要有 LDA 模型作者提出的变分-EM 算法，还有现在常用的 Gibbs 抽样法。

更形式化一点说，语料库中的每一篇文档与 T （通过反复试验等方法事先给定）个主题的一个多项分布相对应，将该多项分布记为 θ 。每个主题又与词汇表（vocabulary）

中的 V 个单词的一个多项分布相对应，将这个多项分布记为 θ 。上述词汇表是由语料库中所有文档中的所有互异单词组成，但实际建模的时候要剔除一些停用词（stopword），还要进行一些词干化（stemming）处理等。和 θ 分别有一个带有超参数（hyperparameter） α 的 Dirichlet 先验分布。对于一篇文档 d 中的每一个单词，我们从该文档所对应的多项分布中抽取一个主题 z ，然后我们再从主题 z 所对应的多项分布中抽取一个单词 w 。将这个过程重复 N_d 次，就产生了文档 d ，这里的 N_d 是文档 d 的单词总数。这个生成过程可以用如下的图模型表示：

LDA 的论文，我们必须先明白两个分布：多项式分布和 Dirichlet 分布。多项式分布：一般的多项式分布是把分布看着是组合的形式，见公式。

14.2 预备知识

14.2.1 Dirichlet 分布

Γ 函数和 Γ 分布

Γ 函数：

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx \quad (14.1)$$

性质：

$$\Gamma(\alpha + 1) = \alpha \Gamma(\alpha) \quad (14.2)$$

特殊值：

$$\Gamma(1) = 1 \quad (14.3)$$

$$\Gamma(n + 1) = n! \quad (14.4)$$

$$\Gamma\left(\frac{1}{2}\right) = \pi \quad (14.5)$$

Γ 分布：

$$\Gamma(x|\alpha) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, x > 0, \alpha > 0 \quad (14.6)$$

均值：

$$\mu(x) = \alpha \quad (14.7)$$

方差：

$$\sigma^2(x) = \alpha^2 \quad (14.8)$$

B 函数和 B 分布

B 函数:

$$\mathbf{B}(\alpha, \beta) = \int_0^1 x^{\alpha-1} (1-x)^{\beta-1} dx, \alpha > 0, \beta > 0 \quad (14.9)$$

$$\mathbf{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \quad (14.10)$$

B 分布:

$$\mathbf{B}(x|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}, \quad (14.11)$$

均值:

$$\mu(x) = \frac{\alpha}{\alpha+\beta} \quad (14.12)$$

方差:

$$\sigma^2(x) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \quad (14.13)$$

性质: 如果 $x_1 \sim \Gamma(\alpha)$ 和 $x_2 \sim \Gamma(\beta)$, 那么

$$Y = \frac{x_1}{x_1 + x_2} \sim \mathbf{B}(x|\alpha, \beta) \quad (14.14)$$

Dirichlet 函数和 Dirichlet 分布

Dirichlet 函数

$$\mathbf{Dir}(\alpha_1, \dots, \alpha_K) = \int \dots \int_{x_1, \dots, x_K} x_1^{\alpha_1-1} \dots x_K^{\alpha_K-1} dx_1 \dots dx_K, \quad (14.15)$$

其中, $0 < x_1, \dots, x_K < 1$, $\sum_{i=1}^K x_i = 1$ 。

Dirichlet 函数是多参数的 **B** 函数。 $K = 2$ 时, $\mathbf{Dir} = \mathbf{B}$ 。

性质:

$$\mathbf{Dir}(\alpha_1, \dots, \alpha_K) = \frac{\Gamma(\alpha_1) \dots \Gamma(\alpha_K)}{\Gamma(\alpha_1 + \dots + \alpha_K)} \quad (14.16)$$

Dirichlet 分布

$$\mathbf{Dir}(x_1, \dots, x_K|\alpha_1, \dots, \alpha_K) = \frac{\Gamma(\alpha_1) \dots \Gamma(\alpha_K)}{\Gamma(\alpha_1 + \dots + \alpha_K)} x_1^{\alpha_1-1} \dots x_K^{\alpha_K-1}, \quad (14.17)$$

$$\mu(x_i) = \frac{\alpha_i}{\alpha_1 + \cdots + \alpha_K} \quad (14.18)$$

$$\sigma^2(x_i) = \frac{\alpha_i(\alpha_1 + \cdots + \alpha_K - \alpha_i)}{(\alpha_1 + \cdots + \alpha_K)^2(\alpha_1 + \cdots + \alpha_K + 1)} \quad (14.19)$$

$$\sigma(x_i, x_j) = \frac{\alpha_i \alpha_j}{(\alpha_1 + \cdots + \alpha_K)^2(\alpha_1 + \cdots + \alpha_K + 1)} \quad (14.20)$$

性质 1:

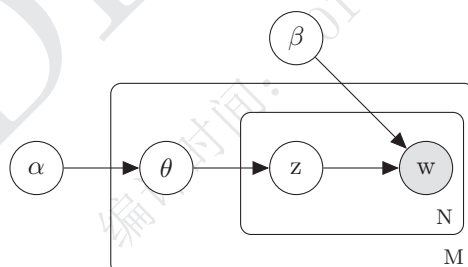
$x_1 \sim \Gamma(\alpha_1), \cdots, x_K \sim \Gamma(\alpha_K)$, 那么

$$y_i = \frac{x_i}{x_1 + \cdots + x_K} \sim \text{Dir}() \quad (14.21)$$

性质 2: 边际分布

性质 3: 分组

14.3 LDA 模型



参考文献和深入阅读

To understand and interpret topic models, it is important to have a solid understanding of how topic models work. Topic models have been described from a variety of perspectives, ranging from the metaphorical, like Jocker’ s LDA Buffet,[2] to the rigorously mathematical, like Blei, Ng, and Jordan’ s article introducing LDA in the Journal of Machine Research,[3] to the pragmatic, like Brett’ s introduction in the Journal of Digital Humanities.[4] My goal is to describe topic modeling by complementing existing

introductions to topic modeling and filling some important bits of information they have left out.

概率主题模型是一种统计语言模型，是用统计方法对自然语言进行建模。Blei et al. [2003] 最早提出了 LDA 模型，使用的是变分法推导，用 EM 算法求解，得到一个局部最优解。由于现在基本都是用 Gibbs Sampling 的方法求解，我也主要阅读的是 Gibbs 方面的论文。

M. Blei 在 2003 年（准确地说应该是 2002 年）提出的 LDA（Latent Dirichlet Allocation）模型（翻译成中文就是——潜在狄利克雷分配模型。Griffiths and Steyvers [2004] 采用了 Gibbs 采样的方法来估计 LDA 模型中参数。因为基于 Gibbs 采样的 LDA 模型非常容易实现，训练过程只需要简单的计算就可以得到很好的结果，因此在信息检索、文本挖掘等领域非常流行。LDA 主题模型涉及到贝叶斯理论、Dirichlet 分布、多项分布、图模型、变分推断、EM 算法、Gibbs 采样等知识。

Hofmann [1999]

共轭先验

Anthes [2010]

关于概率图模型和参数估计的知识可以参考 [Bishop, 2006]。关于 Gibbs 采样以及更基本的 Markov chain Monte Carlo (MCMC) 方法可以参考 Andrieu et al. [2003]。

直接读 Steyvers 的 Probabilistic Topic Models，没必要一定去读 Blei 的论文。

目前 LDA 的 C, java, matlab 实现都可以在网络上找到。LDA 的提出者 Blei 的 C 版本，可以在他的主页¹下载。

概率图模型，在 PRML 第九章有很好的介绍

¹ <http://www.cs.princeton.edu/~blei/lda-c/>

第十五章 关键词抽取

得到

关键词（Keyword）是可以反映一篇文档主要内容或主题的词或词组。通常一篇文档需要用多个关键词来描述其主题。关键词的主要作用有以下两点：

1. 关键词能鲜明而直观地表述文献论述或表达的主题，使读者在未看学术论文的文摘和正文之前便能一目了然地知道论文论述的主题，从而作出是否要花费时间阅读正文的判断。
2. 在一个文档数据库中，关键词可以使人们能很快查到符合要求的文献。检索工具中的关键词索引和数据库文档中的按关键词编制的倒排文档就。关键词索引和倒排文档提供了快速检索文献的途径。关键词具有重要的检索意义。

(2)

要切实使关键词的逻辑组合能准确起到提示论文主题的内容的作用，就应使所选的关键词确实能准确提示该文主题内容，但还是会遇到诸如：研究、使用方法、分析、问题、服务、质量等形式的关键词。由于这些关键词几乎在大多数学术论文中都可使用，使其在提示某一论文主题内容的专指性方面的作用就大大降低，失去该关键词应起的基本作用。

目前主流的搜索引擎也是根据关键词进行搜索的。

关键词抽取是从一段文本中抽取若干有意义和代表性的词或词组。关键词抽取是自然语言处理中的一个非常重要的任务，在自动文摘、信息检索、文本分类、文本聚类等方面具有十分重要的作用。

传统的通过人工抽取关键词方式不仅费时费力，而且在信息资源极大丰富的今天几乎是不可能完成任务，所以通过计算机对输入的文本自动抽取相应的关键词就显得特

别重要。

FNLP 中的关键词抽取主要参考下面文献：

R. Mihalcea and P. Tarau. Texttrank: Bringing order into texts. In Proceedings of EMNLP, volume 4, pages 404–411. Barcelona: ACL, 2004

参考文献

- 王厚峰. 指代消解的基本方法和实现技术. 中文信息学报, 16(006):9–17, 2002.
- 宗成庆. 统计自然语言处理（第二版）. 中文信息处理丛书. 清华大学出版社, 2013.
- J. Allen. Natural language understanding. Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA, 1995.
- Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. Machine learning, 50(1-2):5–43, 2003.
- Gary Anthes. Topic models vs. unstructured data. Commun. ACM, 53(12):16–18, December 2010. ISSN 0001-0782.
- C.M. Bishop. Pattern recognition and machine learning. Springer New York., 2006.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, March 2003. ISSN 1532-4435. doi: <http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993>. URL <http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993>.
- Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In Proceedings of the Tenth Conference on Computational Natural Language Learning, pages 149–164. Association for Computational Linguistics, 2006.
- N. Chomsky and G.A. Miller. Introduction to the formal analysis of natural languages. Wiley, 1963.
- Yoeng-Jin Chu and Tseng-Hong Liu. On shortest arborescence of a directed graph. Scientia Sinica, 14(10):1396, 1965.
- Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, 2002.
- Susan P Converse. Resolving pronominal references in chinese with the hobbs algorithm. In Proceedings of the 4th SIGHAN workshop on Chinese language processing, pages 116–122, 2005.
- Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. Journal of Machine Learning Research, 7:551–585, 2006.

- R. Duda, P. Hart, and D. Stork. Pattern Classification. Wiley, New York, 2nd edition, 2001. ISBN 0471056693.
- Jack Edmonds. Optimum branchings. Journal of Research of the National Bureau of Standards B, 71(233-240):160, 1967.
- Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. Machine learning, 37(3):277–296, 1999.
- Jeffrey Friedl. Mastering regular expressions. O’Reilly Media, Inc., 2006.
- Haim Gaifman. Dependency Systems and Phrase-Structure Systems. Information and Computation/information and Control, 8:304–337, 1965. doi: 10.1016/S0019-9958(65)90232-9.
- Niyu Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. In Proceedings of the sixth workshop on very large corpora, pages 161–170, 1998.
- Thomas L Griffiths and Mark Steyvers. Finding scientific topics. Proceedings of the National academy of Sciences of the United States of America, 101(Suppl 1):5228–5235, 2004.
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. Springer, New York, 2001.
- Jerry R Hobbs. Resolving pronoun references. Lingua, 44(4):311–338, 1978.
- T. Hofmann. Probabilistic latent semantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 50–57, 1999.
- John E Hopcroft. 自动机理论, 语言和计算导论. 清华大学出版社, 2002.
- R. Hudson. English Word Grammar. Basil Blackwell, Oxford, 1990.
- T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. Proc. of Euro. Conf. on Mach. Learn. (ECML), pages 137–142, 1998.
- M.I. Jordan. Learning in Graphical Models. Kluwer Academic Publishers, 1998.
- D. Jurafsky, J.H. Martin, A. Kehler, K. Vander Linden, and N. Ward. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, volume 2. Prentice Hall New Jersey, 2000.
- Christopher Kennedy and Branimir Boguraev. Anaphora for everyone: pronominal anaphora resolution without a parser. In Proceedings of the 16th conference on Computational linguistics, pages 113–118. Association for Computational Linguistics, 1996.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

- Shalom Lappin and Herbert J Leass. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561, 1994.
- C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.
- A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598. Citeseer, 2000.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05*, pages 91–98, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: <http://dx.doi.org/10.3115/1219840.1219852>. URL <http://dx.doi.org/10.3115/1219840.1219852>.
- I.A. Melčuk. *Dependency syntax: theory and practice*. State University of New York Press, 1988.
- R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4, pages 404–411. Barcelona: ACL, 2004.
- T.M. Mitchell. *Machine learning*. Burr Ridge, IL: McGraw Hill, 1997.
- Ruslan Mitkov. *Anaphora resolution: the state of the art*. Citeseer, 1999.
- Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, 2010.
- H. Nikula. *Dependensgrammatik*. LiberFörlag, 1986.
- Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932. sn, 2007.
- Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. Citeseer, 2003.
- Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P/P05/P05-1013>.
- F. Peng, F. Feng, and A. McCallum. Chinese segmentation and new word detection using conditional random fields. *Proceedings of the 20th international conference on Computational Linguistics*, 2004.
- L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

- Jane J Robinson. Dependency structures and transformational rules. Language, pages 259–285, 1970.
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386–408, 1958.
- G. Salton, A. Wong, and CS Yang. A vector space model for automatic indexing. Communications of the ACM, 18(11):613–620, 1975.
- F. Sebastiani. Machine learning in automated text categorization. ACM computing surveys, 34(1):1–47, 2002.
- C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. Introduction to statistical relational learning, page 93, 2007.
- Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In Proceedings of the 17th Annual Conference on Neural Information Processing Systems, Whistler, B.C., Canada, 2003.
- Lucien Tesnière. Éléments de syntaxe structurale. Librairie C. Klincksieck, 1959.
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y Altun. Support vector machine learning for interdependent and structured output spaces. In Proceedings of the International Conference on Machine Learning(ICML), 2004.
- F. Xia. The part-of-speech tagging guidelines for the penn Chinese treebank (3.0), 2000.
- N. Xue. Chinese word segmentation as character tagging. Computational Linguistics and Chinese Language Processing, 8(1):29–48, 2003.
- Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. Natural language engineering, 11(2): 207–238, 2005.
- H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In Proceedings of the International Workshop on Parsing Technologies (IWPT), volume 3, 2003.
- Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In Proc. of Int. Conf. on Mach. Learn. (ICML), volume 97, 1997.
- R.B. Yates and B.R Neto. Modern Information Retrieval. New York: ACM Press Series/Addison Wesley, 1999.