

自然语言处理

理论与实战

唐聃 白宁超 冯暄 卿鸿宾 文俊 著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

自然语言处理是什么？谁需要学习自然语言处理？自然语言处理在哪些地方应用？相关问题一直困扰着不少初学者。针对这一情况，作者结合教学经验和工程应用编写此书。本书讲述自然语言处理相关学科知识和理论基础，并介绍使用这些知识的应用和工具，以及如何在实际环境中使用它们。由于自然语言处理的特殊性，其是一门多学科交叉的学科，初学者难以把握知识的广度和宽度，对侧重点不能全面掌握。本书针对以上情况，经过科学调研分析，选择以理论结合实例的方式将内容呈现出来。其中涉及开发工具、Python 语言、线性代数、概率论、统计学、语言学等工程上常用的知识介绍，然后介绍自然语言处理的核心理论和案例解析，最后通过几个综合性的例子完成自然语言处理的学习和深入。本书旨在帮助读者快速、高效地学习自然语言处理和人工智能技术。

本书适用于具备一定编程基础的计算机专业、软件工程专业、通信专业、电子技术专业和自动化专业的大学二年级以上的学生、科研工作者和相关技术人员。一些做工程应用的自然语言处理工程师，也可以通过阅读本书补充理论知识，理论知识的魅力在于遇到工程难题时，可以知道其背后的原因，快速、准确地解决问题。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

自然语言处理理论与实战 / 唐聃等著. —北京：电子工业出版社，2018.6
ISBN 978-7-121-34390-2

I. ①自… II. ①唐… III. ①自然语言处理—研究 IV. ①TP391

中国版本图书馆 CIP 数据核字（2018）第 122905 号

责任编辑：陈晓猛

印 刷：三河市君旺印务有限公司

装 订：三河市君旺印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×980 1/16 印张：22.5

字数：432 千字

版 次：2018 年 6 月第 1 版

印 次：2018 年 6 月第 1 次印刷

定 价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前言

本书讲述自然语言处理重要的相关学科知识和理论基础，并介绍使用这些知识的应用和工具，以及如何在实际环境中使用它们。市面上出版的自然语言处理书籍不多，且大多数讨论的是其背后的深奥原理，很少涉及基础知识和编程实现。自然语言处理是一门多学科交叉的学科，初入门的读者难以把握知识的广度和宽度，尤其对侧重点不能全面掌握。本书针对以上情况，经过科学调研分析，选择以理论结合实例的方式呈现知识点。首先介绍开发工具、Python 语言、线性代数、概率论、统计学、语言学等工程上常用的知识，然后介绍自然语言处理的核心理论和案例解析，最后通过几个综合性的例子完成自然语言处理的学习和深入。本书旨在帮助读者快速高效地学习自然语言处理和人工智能技术。

读者对象

自然语言处理是什么？谁需要学习自然语言处理？自然语言处理在哪些地方应用？本书就是对这几个问题的回答。自然语言处理领域主要探讨：如何处理及运用自然语言；自然语言认知（让计算机“懂”人类的语言）；自然语言生成系统（将计算机数据转化为自然语言）和自然语言理解系统（将自然语言转化为计算机程序更易于处理的形式）。自然语言处理在我们身边应用得非常广泛，其中包括：语音的自动合成与识别、机器翻译、自然语言理解、人机对话、信息检索、文本分类、自动文摘，等等。此外，自然语言处理也是人工智能、机器学习、深度学习的基础，重要程度不言而喻。如果读者有一定的编程基础，那么将有助于本书的阅读。如果读者不具备线性代数、概率论、统计学、语言学的知识，则可从本书中快速学习常见的工程应用知识；如果读者具备线性代数、概率论、统计学、语言学的知识，则更利于本书的阅读，可以对知识进行查全补充。此外，本身使用 Python 语言进行编程，假设读者具备 Python 知识，则可以跳过第 2 章，也更有利于本书的阅读。本书对于具备一定编程基础的计算机专业、软件工程专业、通信专业、电子技术专业和自动化专业的大学二年级以上的学生都是适宜的。一些做工程应用的自然语言处理工程师，也可以通过阅读本书补充理论知识。理论知识的最大魅力在于遇到工程难题时，可以知道其背后的原因，快速准确地解决问题。本书整体难度适宜，适合作为自学用书或课程教材。

本书结构

本书共四大部分 15 章，第一部分为基础部分，从第 1 章至第 6 章，主要介绍在自然语言交叉学科中，工程应用常用的学科知识，包括自然语言处理概述、Python 基础知识和环境搭建、线性代数、概率论、统计学、语言学。第二部分为理论部分，从第 7 章至第 14 章，主要介绍自然语言处理常用的理论知识，包括自然语言处理任务限制、技术范畴、语料库、中文自动分词、数据预处理、马尔可夫模型、条件随机场、模型评估和命名实体识别。第三部分为实战部分，第 15 章通过 GitHub 数据提取与可视化分析、微博话题爬取与存储分析，综合介绍网络爬虫、中文分词、数据处理、模型选择、数据分析、自然语言处理工具和数据可视化等技术点，这些技术也适用于以机器学习为代表的人工智能领域。本书各章节的具体内容介绍如下。

- ◎ 第 1 章基础入门：随着人工智能的快速发展，自然语言处理和机器学习技术的应用愈加广泛。然而身为初学者，要想快速入门这些前沿技术总是存在着各种各样的困难。为使读者对该领域的整体概况有一个系统明晰的认识，本章主要从发展历程、研究现状、应用前景等角度概要介绍自然语言处理及相关的机器学习技术。
- ◎ 第 2 章快速上手 Python：Python 作为一门简洁优美且功能强大的语言，越来越受到编程人员的青睐，在工业界和学术界也非常受欢迎。本书的全部代码都是通过 Python 实现的，之所以选择 Python 语言，是因为其可以跨平台跨应用开发，因此本章旨在帮助读者快速领略 Python 的概貌。如果读者已经具备 Python 基础，则可略过此章。
- ◎ 第 3 章线性代数：机器学习是计算机科学、统计学、数学和信息论等多个领域交叉的学科。线性代数又是数学的一个重要分支，对机器学习有着直接的影响。诸如算法建模、参数设置、验证策略、识别欠拟合和过拟合，等等。读者往往知道线性代数很有用，常常全书通读，造成时间不足和效率较低，归因于对线性代数在机器学习中的重点和用途不明。本章主要以简明的方式介绍常用的线性代数知识，并使读者知道线性代数常用于哪些方面。
- ◎ 第 4 章概率论：机器学习与深度学习是多学科交叉的科学技术，其中数学尤为重要，是很多形式化模型向数学建模的必经过程。继线性代数核心知识的介绍之后，本章着重介绍概率论的相关知识。
- ◎ 第 5 章统计学：在数据科学中，统计学的地位尤为显著。这是一门在数据分析的基础上，研究如何测定、收集、整理、归纳和分析数据规律，以便给出正确消息的学科。通过揭示数据背后的规律和隐藏信息，给相关角色提供参照价值，以做出相应的决策。其在数据挖掘、自然语言处理、机器学习中都被广泛应用。本章首先介绍常见的图形可视化的概念和使用，继而介绍数据度量标准、概率分布、统计假设检验、相关和回归，

以短小精悍的篇章使读者掌握基本的统计知识。

- ◎ 第6章语言学：本章主要从语音、词汇、语法三个角度对现代汉语进行一个简单概要的勾勒，在以往传统的语言学教材中一般还有“文字”“修辞”两节内容，因篇幅有限、与全书关联不强，在此删繁就简，未给读者一一呈现。需要注意的是，语言学本身是一门十分庞杂的学科，知识体系与研究方法或因语言不同而有区别，或因派别主义不同而有区别。但无论是何种语言，或是何门何派，在进行自然语言处理时我们要面临的永远是一个个真实的语料和具体的语言现象。理论是用来指导实践、拓宽我们研究思路的，究竟最后采用何种理论，这只是一个“白猫黑猫”的问题。
- ◎ 第7章自然语言处理：本章开篇直击要点，即自然语言处理的任务和限制。进而介绍其所涉及的主要技术范畴，并对这些技术方向进行介绍。在针对当前自然语言处理的难点进行详细剖析后，最终对2017年以后自然语言处理的发展进行展望。
- ◎ 第8章语料库：大数据发展的基石就是数据量的快速增加，无论是自然语言处理、数据挖掘、文本处理，还是机器学习领域，都是在此基础上通过规则或统计方法进行模型构建的。但是不是数据量足够大就叫大数据了呢？是不是数据量足够多就构成语料库了呢？带着这些疑问，本章将带你走进语料库的世界，对语料知识进行一次全面而深入的了解。
- ◎ 第9章中文自动分词：中文分词技术属于自然语言处理的技术范畴，中文分词是其他中文信息处理的基础，搜索引擎只是中文分词的一个应用。诸如机器翻译（MT）、语音合成、自动分类、自动摘要、自动校对，等等。
- ◎ 第10章数据预处理：数据预处理的整个步骤流程在自然语言处理的工程中要比其在机器学习的工程中精简一些，最大的区别就在于数据清洗和特征构造这两个至关重要的过程。在自然语言处理中特征构造是否良好，很大程度上取决于所构造的特征数据集的数据特性与文本内容语义吻合程度的高低。比如，文本情感分类和文本内容分类都属于分类范畴，但对于同一种算法（参数都调整到最优），在两个不同分类的业务下，得到的结果可能会相差很大。通过仔细分析，我们不难发现造成这种差异的根本原因就是构造出来的特征数据集的数据模式没有很好地契合文本的真实语义，这也是自然语言处理的最大难点。
- ◎ 第11章马尔可夫模型：笔者最早接触马尔可夫模型的定义源于吴军先生的《数学之美》一书，起初觉得深奥难懂且没什么用处。直到学习自然语言处理时，才真正使用到马尔可夫模型，并体会到此模型的奇妙之处。马尔可夫模型在处理序列分类时具有强大的功能，解决诸如词类标注、语音识别、句子切分、字素音位转换、局部句法剖

析、语块分析、命名实体识别、信息抽取等问题。此外它还广泛应用于自然科学、工程技术、生物科技、公用事业、信道编码等多个领域。

- ◎ 第 12 章条件随机场：条件随机场常用于序列标注、数据分割等自然语言处理任务中，此外在中文分词、中文人名识别和歧义消解等任务中也有应用。本书基于笔者在做语句识别序列标注过程中对条件随机场产生的了解。主要内容源于自然语言处理、机器学习、统计学习方法和部分网上资料对 CRF 的相关介绍，最后由笔者进行大量研究整理后汇总成知识体系。本章首先介绍条件随机场的相关概念，然后结合实例以期让读者深入理解条件随机场的应用。
- ◎ 第 13 章模型评估：本章源于基于 HMM 模型序列标注的一个实验，在实验完成之后，迫切想知道采用的序列标注模型好坏，有哪些指标可以度量。于是就产生了对这一专题进度的学习总结，这样也便于其他人参考。本章依旧简明扼要地梳理出模型评估核心指标，以期达到实用的目的。
- ◎ 第 14 章命名实体识别：命名实体识别在自然语言处理中占据着非常重要的地位，也是不可逾越的学术问题。命名实体识别的学术理论和研究方法众多，本章侧重整体介绍。首先阐述命名实体识别的背景知识和研究概况，介绍中文命名实体识别的特点与难点，辅以案例加深理解；然后对命名实体识别当前的研究方法和核心技术进行详细介绍；最后展望其在未来人工智能方面的发展前景。
- ◎ 第 15 章自然语言处理实战：自然语言处理技术是理论与实践相结合的一门学科，通过前面基础理论知识介绍，读者对其理论有所认识，但其究竟有何用、怎么用却不深刻。本章通过实例演练，一方面对前面几章的知识进行复习回顾，另一方面利于加深理解研发的相关工作。本章的第一个案例以 GitHub 为例，实现数据提取和可视化；第二个案例以微博话题为例，实现数据采集、提取、存储与分析。

勘误

由于笔者能力有限，时间仓促，书中难免有错漏，欢迎读者批评指正。

联系方式：nlpjiaocheng@sina.com。

作者介绍

唐聃 教授，中科院工学博士。现工作于成都信息工程大学软件工程学院。研究方向包括自然语言处理、信息安全、数据分析。曾参与多项国家 863 项目和中科院知识创新工程项

目、省科技厅和教育厅项目；2016 年入选中国科学院西部之光人才计划（中国科学院西部青年学者）。

白宁超 四川省计算机研究院软件开发工程师，曾参与多项四川省科技厅项目。其自然语言处理系列博文曾被 CSDN、博客园、阿里云栖等多个技术社区转载。

冯暄 高级工程师，硕士学位。现任四川省计算机研究院信息化工程研究所所长。研究方向包括物联网、多源信息融合、软件工程。主持或参与国家级、省级科研项目 16 项。获得四川省科技进步奖二等奖 2 项、四川省科技进步奖三等奖 1 项。

卿鸿宾 四川大学中文系在校生。研究方向包括应用语言学、计算语言学、韵律句法学等。常年从事文学创作与文字工作，2017 年作品《黄昏速写》发表于《子曰书院》微信公众号，取得了不错的反响。

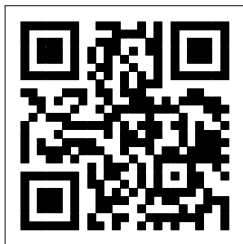
文俊 硕士学位，现工作于成都广播电视台橙视传媒大数据中心，大数据算法工程师。研究方向主要包括数据挖掘、机器学习、自然语言处理、深度学习及云计算。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- ◎ **提交勘误**：您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- ◎ **交流互动**：在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/34390>



目录

- 第 1 章 基础入门..... 1
 - 1.1 什么是自然语言处理..... 1
 - 1.1.1 自然语言处理概述..... 1
 - 1.1.2 自然语言处理的发展历史..... 3
 - 1.1.3 自然语言处理的工作原理..... 6
 - 1.1.4 自然语言处理的应用前景..... 7
 - 1.2 开发工具与环境..... 7
 - 1.2.1 Sublime Text 和 Anaconda 介绍..... 7
 - 1.2.2 开发环境的安装与配置..... 8
 - 1.3 实战：第一个小程序的诞生..... 13
 - 1.3.1 实例介绍..... 13
 - 1.3.2 源码实现..... 13
- 第 2 章 快速上手 Python..... 15
 - 2.1 初识 Python 编程语言..... 15
 - 2.1.1 Python 概述..... 15
 - 2.1.2 Python 能做什么..... 17
 - 2.1.3 Python 的语法和特点..... 19
 - 2.2 Python 进阶..... 24
 - 2.2.1 Hello World..... 24
 - 2.2.2 语句和控制流..... 24
 - 2.2.3 函数..... 27
 - 2.2.4 List 列表..... 29
 - 2.2.5 元组..... 32
 - 2.2.6 set 集合..... 33
 - 2.2.7 字典..... 33

2.2.8 面向对象编程：类.....	34
2.2.9 标准库.....	36
2.3 Python 深入——第三方库.....	36
2.3.1 Web 框架.....	36
2.3.2 科学计算.....	37
2.3.3 GUI.....	37
2.3.4 其他库.....	37
第 3 章 线性代数.....	39
3.1 线性代数介绍.....	39
3.2 向量.....	40
3.2.1 向量定义.....	40
3.2.2 向量表示.....	42
3.2.3 向量定理.....	42
3.2.4 向量运算.....	43
3.3 矩阵.....	47
3.3.1 矩阵定义.....	47
3.3.2 矩阵表示.....	48
3.3.3 矩阵运算.....	48
3.3.4 线性方程组.....	51
3.3.5 行列式.....	51
3.3.6 特征值和特征向量.....	55
3.4 距离计算.....	56
3.4.1 余弦距离.....	56
3.4.2 欧氏距离.....	57
3.4.3 曼哈顿距离.....	58
3.4.4 明可夫斯基距离.....	59
3.4.5 切比雪夫距离.....	61
3.4.6 杰卡德距离.....	62
3.4.7 汉明距离.....	63
3.4.8 标准化欧式距离.....	64
3.4.9 皮尔逊相关系数.....	65

第 4 章 概率论	67
4.1 概率论介绍	67
4.2 事件	68
4.2.1 随机试验	68
4.2.2 随机事件和样本空间	69
4.2.3 事件的计算	70
4.3 概率	71
4.4 概率公理	73
4.5 条件概率和全概率	76
4.5.1 条件概率	76
4.5.2 全概率	77
4.6 贝叶斯定理	78
4.7 信息论	79
4.7.1 信息论的基本概念	79
4.7.2 信息度量	80
第 5 章 统计学	85
5.1 图形可视化	85
5.1.1 饼图	85
5.1.2 条形图	88
5.1.3 热力图	91
5.1.4 折线图	93
5.1.5 箱线图	96
5.1.6 散点图	99
5.1.7 雷达图	102
5.1.8 仪表盘	104
5.1.9 可视化图表用法	106

5.2 数据度量标准	108
5.2.1 平均值	108
5.2.2 中位数	108
5.2.3 众数	110
5.2.4 期望	111
5.2.5 方差	112
5.2.6 标准差	113
5.2.7 标准分	114
5.3 概率分布	115
5.3.1 几何分布	115
5.3.2 二项分布	116
5.3.3 正态分布	118
5.3.4 泊松分布	121
5.4 统计假设检验	123
5.5 相关和回归	125
5.5.1 相关	125
5.5.2 回归	127
5.5.3 相关和回归的联系	130
第 6 章 语言学	132
6.1 语音	132
6.1.1 什么是语音	132
6.1.2 语音的三大属性	133
6.1.3 语音单位	134
6.1.4 记音符号	135
6.1.5 共时语流音变	136
6.2 词汇	137
6.2.1 什么是词汇	137
6.2.2 词汇单位	137
6.2.3 词的构造	138

6.2.4	词义及其分类	140
6.2.5	义项与义素	141
6.2.6	语义场	142
6.2.7	词汇的构成	143
6.3	语法	143
6.3.1	什么是语法	143
6.3.2	词类	144
6.3.3	短语	148
6.3.4	单句	150
6.3.5	复句	152
第 7 章	自然语言处理	155
7.1	自然语言处理的任务和限制	155
7.2	自然语言处理的主要技术范畴	156
7.2.1	语音合成	156
7.2.2	语音识别	156
7.2.3	中文自动分词	157
7.2.4	词性标注	158
7.2.5	句法分析	158
7.2.6	文本分类	159
7.2.7	文本挖掘	160
7.2.8	信息抽取	161
7.2.9	问答系统	161
7.2.10	机器翻译	162
7.2.11	文本情感分析	163
7.2.12	自动摘要	164
7.2.13	文字蕴涵	165

7.3 自然语言处理的难点	165
7.3.1 语言环境复杂	165
7.3.2 文本结构形式多样	166
7.3.3 边界识别限制	166
7.3.4 词义消歧	167
7.3.5 指代消解	168
7.4 自然语言处理展望	169
第 8 章 语料库	173
8.1 语料库浅谈	173
8.2 语料库深入	174
8.3 自然语言处理工具包：NLTK	176
8.3.1 NLTK 简介	176
8.3.2 安装 NLTK	177
8.3.3 使用 NLTK	180
8.3.4 在 Python NLTK 下使用 Stanford NLP	186
8.4 获取语料库	194
8.4.1 国内外著名语料库	195
8.4.2 网络数据获取	197
8.4.3 NLTK 获取语料库	200
8.5 综合案例：走进大秦帝国	208
8.5.1 数据采集和预处理	208
8.5.2 构建本地语料库	208
8.5.3 大秦帝国语料操作	209
第 9 章 中文自动分词	216
9.1 中文分词简介	216
9.2 中文分词的特点和难点	218
9.3 常见中文分词方法	219
9.4 典型中文分词工具	220
9.4.1 HanLP 中文分词	220

9.4.2 其他中文分词工具.....	223
9.5 结巴中文分词	224
9.5.1 基于 Python 的结巴中文分词.....	224
9.5.2 结巴分词工具详解.....	227
9.5.3 结巴分词核心内容.....	230
9.5.4 结巴分词基本用法.....	233
第 10 章 数据预处理	241
10.1 数据清洗.....	241
10.2 分词处理.....	242
10.3 特征构造.....	242
10.4 特征降维与选择.....	243
10.4.1 特征降维.....	243
10.4.2 特征选择.....	243
10.5 简单实例.....	244
10.6 本章小结.....	249
第 11 章 马尔可夫模型	250
11.1 马尔可夫链	250
11.1.1 马尔可夫简介	250
11.1.2 马尔可夫链的基本概念	251
11.2 隐马尔可夫模型.....	253
11.2.1 形式化描述	253
11.2.2 数学形式描述	255
11.3 向前算法解决 HMM 似然度.....	256
11.3.1 向前算法定义	256
11.3.2 向前算法原理	256
11.3.3 现实应用：预测成都天气的冷热	258
11.4 文本序列标注案例：Viterbi 算法	259

第 12 章 条件随机场	263
12.1 条件随机场介绍	263
12.2 简单易懂的条件随机场	265
12.2.1 CRF 的形式化表示	265
12.2.2 CRF 的公式化表示	266
12.2.3 深度理解条件随机场	268
第 13 章 模型评估	269
13.1 从统计角度介绍模型概念	269
13.1.1 算法模型	269
13.1.2 模型评估和模型选择	270
13.1.3 过拟合与欠拟合的模型选择	272
13.2 模型评估与选择	275
13.2.1 模型评估的概念	275
13.2.2 模型评估的评测指标	275
13.2.3 以词性标注为例分析模型评估	276
13.2.4 模型评估的几种方法	278
13.3 ROC 曲线比较学习器模型	279
第 14 章 命名实体识别	281
14.1 命名实体识别概述	281
14.2 命名实体识别的特点与难点	284
14.3 命名实体识别方法	284
14.4 中文命名实体识别的核心技术	286
14.5 展望	295
第 15 章 自然语言处理实战	296
15.1 GitHub 数据提取与可视化分析	296
15.1.1 了解 GitHub 的 API	296
15.1.2 使用 NetworkX 作图	299
15.1.3 使用 NetworkX 构建兴趣图	301

15.1.4 NetWorkX 部分统计指标.....	304
15.1.5 构建 GitHub 的兴趣图	305
15.1.6 可视化.....	318
15.2 微博话题爬取与存储分析	320
15.2.1 数据采集.....	320
15.2.2 数据提取.....	329
15.2.3 数据存储.....	332
15.2.4 项目运行与分析.....	333
附录 A Python 与其他语言调用.....	337
附录 B Git 项目上传简易教程.....	339
参考文献.....	341

第 1 章

基础入门

导读：随着人工智能的快速发展，自然语言处理技术的应用越来越广泛。身为初学者，要想快速入门这些前沿技术总是存在各种各样的困难。为使读者对该领域整体概况有一个系统明晰的认识，本章首先介绍自然语言处理发展历程、研究现状、应用前景等。古语说“工欲善其事，必先利其器”，本书的“器”就是开发环境的部署，接着介绍 Sublime 的安装部署与使用。最后用一个简单的实战案例让读者领略编程之美。

1.1 什么是自然语言处理

1.1.1 自然语言处理概述

自然语言处理的概念

自然语言处理（Natural Language Processing，简称 NLP）是人工智能和语言学交叉领域下的分支学科。该领域主要探讨如何处理及运用自然语言、自然语言认知（即让计算机“懂”人类的语言）、自然语言生成系统（将计算机数据转化为自然语言），以及自然语言理解系统（将自然语言转化为计算机程序更易于处理的形式）。

所谓“自然语言”，其实就是我们日常生活中使用的语言（在这里还包括书面文字和语音视频等），人们熟知的汉语、日语、韩语、英语、法语等语言都属于此范畴。至于“自然语言处理”，则是对自然语言进行数字化处理的一种技术；是通过语音文字等形式与计算机进行通信，从而实现“人机交互”的技术。

自然语言处理的学科领域

多交叉学科：自然语言处理是一门多学科交叉的技术，其中包括语言学、计算机科学（提供模型表示、算法设计、计算机实现）、数学（数学模型）、心理学（人类言语心理模型和理论）、哲学（提供人类思维和语言的更深层次理论）、统计学（提供样本数据的预测统计技术）、电子工程（信息论基础和语言信号处理技术）、生物学（人类言语行为机制理论）。

自然语言处理的研究方向

自然语言处理热门的研究方向包括语音的自动合成与识别、机器翻译、自然语言理解、人机对话、信息检索、文本分类、自动文摘，等等。分为 4 大方向：

- ◎ 语言学方向；
- ◎ 数据处理方向；
- ◎ 语言工程方向；
- ◎ 人工智能和认知科学方向。

细分为 13 个方面。

- （1）口语输入：语音识别、信号表示、鲁棒的语音识别、语音识别中的隐马尔可夫模型方法、语言模型、说话人识别、口语理解。
- （2）书面语输入：文献格式识别、光学字符识别（OCR）（印刷体识别/手写体识别、手写界面、手写文字分析）。
- （3）语言分析理解：小于句子单位的处理、语法的形式化、针对基于约束的语法编写的词表、计算语义学、句子建模和剖析技术、鲁棒的剖析技术。
- （4）语言生成：句法生成、深层生成。
- （5）口语输入技术：合成语音技术、语音合成的文本解释、口语生成。
- （6）话语分析与对话：对话建模、话语建模口语对话系统。
- （7）文献自动处理：文献检索、文本解释（信息抽取、文本内容自动归纳、文本写作和编辑的计算机支持、工业和企业中使用的受限语言）。
- （8）多语问题的计算机处理：机器翻译、人助机译、机助人译、多语言信息检索、多语言语音识别、自动语种验证。

- (9) 多模态的计算机处理：空间和时间表示方法、文本与图像处理、口语与手势的模态结合、口语与面部信息的模态结合（面部运动和语音识别）。
- (10) 信息传输和信息存储：语音压缩、语音品质的提升。
- (11) 自然语言处理中的数学方法：统计建模和分类的数学理论、数字信号处理技术、剖析算法的数学基础研究、神经网络、有限状态分析技术、语音/语言处理中的最优化技术和搜索技术。
- (12) 语言资源：书面语料库、口语语料库、机器词典与词网的建设、术语编撰和术语数据库、网络数据挖掘和信息提取。
- (13) 自然语言处理系统的评测：面向任务的文本分析评测、机器翻译系统和翻译工具的评测、大覆盖面的自然语言剖析器的评测、语音识别（评估和评测、语音合成评测、系统的可用性和界面的评测、语音通信质量的评测、文字识别系统的评测）。

1.1.2 自然语言处理的发展历史

机器翻译的发展历程

自然语言处理的相关研究最早是从机器翻译系统的研究开始的。20 世纪 60 年代，国外对机器翻译曾有过大规模的研究工作，投入了大量的人力物力财力。但是，受到客观历史因素的限制，当时人们低估了自然语言的复杂性，语言处理的理论和技术均不成熟，所以进展并不大。其主要的做法是存储两种语言的单词、短语对应译法的大辞典，翻译时一一对应，技术上只是调整语言的同条顺序。但日常生活中语言的翻译远不是如此简单，很多时候还要参考某句话前后的意思。机器翻译研究的发展大致可分为三个时期。

- (1) 初创期（1947 ~ 1970）：计算机问世（1946）的第二年，英国工程师布斯（A.D.Booth）和美国工程师威弗（W.Weaver）最早提出了利用计算机进行自动翻译。1952 年，在洛克菲勒基金会的大力支持下，一些英美学者在美国麻省理工学院召开了第一次机器翻译会议。1954 年，《机械翻译》（*Mechanical Translation*）杂志开始公开发行人。同年，成功地进行了世界上第一次机器翻译试验。尽管这次试验用的机器词汇仅仅包含了 250 个俄语单词和 6 条机器语法规则。但是，它第一次向公众和科学界展示了机器翻译的可行性，并且激发了美国政府部门在随后十年对机器翻译进行大量资助的兴趣。随着研究的深入，人们看到不是机器翻译的成功，而是一个又一个它无法克服的局限。第一代机器翻译系统设计上的粗糙所带来的翻译质量的低劣，最终导致了一些人对机器

翻译的研究失去信心。有些人甚至错误地认为机器翻译追求全自动质量目标是不可能实现的。标志着机器翻译的研究就此陷入低谷。

- (2) 复苏期 (1970 ~ 1976): 尽管机器翻译的研究困难重重, 但是法国、日本、加拿大等国仍然坚持机器翻译的研究。在 20 世纪 70 年代初期, 机器翻译又出现了复苏的局面。在这个时期, 研究者们普遍认识到, 原语和译语两种语言的差异, 不仅表现在词汇上的不同, 而且还表现在句法结构上的不同, 为了得到可读性强的译文, 必须在自动句法分析上多下功夫。通过大量的科学实验, 机器翻译的研究者逐渐认识到机器翻译过程本身必须保持原语和译语在语义上的一致, 一个好的机器翻译系统应该把原语的语义准确无误地在译语中表现出来。于是, 语义分析在机器翻译中越来越受到重视。美国斯坦福大学的威尔克斯 (Y.A.Wilks) 提出了“优选语义学”, 并在此基础上设计了英法机器翻译系统。由于这个系统的语义表示方法比较细致, 能够解决仅用句法分析难于解决的歧义现象、代词所指等困难问题, 译文质量较高, 受到专家学者的一致肯定。
- (3) 繁荣期 (1976 ~ 至今): 繁荣期最突出的特点是机器翻译研究走上了实用化的道路, 出现了一大批实用化的机器翻译系统, 机器翻译产品开始进入市场, 逐渐由实用化步入商业化。第二代机器翻译系统以基于转换的方法为代表, 普遍采用以句法分析为主、语义分析为辅的基于规则的方法, 采用由抽象的转换表示的分层次实现策略。比如加拿大蒙特利尔大学开发研制的实用性机器翻译系统 TAUM-METEO 就是采用了典型的转换方法, 整个翻译过程分为 5 个阶段 (英—法翻译): 英语形态分析、英语句法分析、转换、法语句法生成和法语形态生成。这个翻译系统投入使用之后, 每小时可以翻译 6 万 ~ 30 万个词, 每天可以翻译 1500 ~ 2000 篇天气预报的资料, 并能够通过电视、报纸立即公布。TAUM-METEO 系统是“机器翻译发展史上的一个里程碑, 它标志着机器翻译由复苏走向了繁荣”。

我国机器翻译的发展历程

我国机器翻译的起步并不算太晚, 是继美国、苏联、英国之后世界上第四个开展机器翻译研究的国家。早在 20 世纪 50 年代机器翻译就被列入我国科学研究的发展规划。一些研究人员还进行了俄汉机器翻译实验, 取得了一定的研究成果。我国机器翻译研究的全面开展始于 20 世纪 80 年代中期, 特别是 20 世纪 90 年代以来, 一批机器翻译系统相继问世, 其中影响力较大的有: 中软总公司开发的汉英—汉日翻译系统 (1993), 中科院计算所研制的 IMTEC 英汉翻译系统 (1992) 等。

在自然语言处理的形成和发展进程中, 除机器翻译外, 自然语言理解 (Natural Language

Understanding) 所起到的作用也是不可忽视的。自然语言理解的发展始于 20 世纪 60 年代中期机器翻译处于举步维艰之时,到了 20 世纪 70 年代初,它的研究已获得了累累硕果。自然语言理解又称“人机对话”,就是“让计算机理解自然语言,使计算机获得人类理解自然语言的智能,并对人给计算机提出的问题,通过对话的方式,用自然语言进行回答”。20 世纪 60 年代中期,人们开始由“词对词”的翻译方式逐步转入对自然语言的语法、语义和语用等基本问题的研究,并尝试让计算机来理解自然语言。许多学者认为,判断计算机是否理解了自然语言的最直观方法,就是让人类同计算机对话,如果计算机对人提出的问题能做出有意义的回答,那就证明计算机已经理解了自然语言。最初的“人机对话”系统(或“自然语言理解”系统)的研究工作主要在美国。冯志伟教授把第一代自然语言理解系统分为四种类型:

- (1) 特殊格式系统。根据人机对话内容的特点,采用特定的格式来进行人机对话。
- (2) 以文本为基础的系统。某些研究者不满意在特殊格式系统中种种格式的限制,因为就一个专门领域来说,最方便的还是使用不受特殊格式结构限制的系统来进行人机对话。
- (3) 有限逻辑系统。在这种系统中,自然语言的句子以某种更加形式化的记号来替代,这些记号自成一个有限逻辑系统,可以进行某些推理。
- (4) 一般演绎系统。它使用某些标准数学符号来表达信息,可以表达那些在有限逻辑系统中不容易表达出来的复杂信息,从而进一步提高自然语言理解系统的能力。随着研究的进一步深入,第二代自然语言理解系统应运而生。这些系统绝大多数是程序演绎系统,大量地进行语义、语境以至语用的分析,输入/输出都使用书面文字。口头的自然语言理解系统还涉及语音识别、语音合成等复杂的技术,发展速度比较缓慢。

自然语言处理的发展历程

- ◎ 1948 年,香农(Shannon)把离散马尔可夫过程的概率模型应用于描述语言的自动机;同时又把“熵”(entropy)的概念引用到语言处理中。而克莱尼(Kleene)在同一时期研究了有限自动机和正则表达式。
- ◎ 1956 年,乔姆斯基(Chomsky)提出了上下文无关语法。这一工作导致了基于规则和基于概率两种不同的自然语言处理方法的诞生,使得该领域的研究分成了采用规则方法的符号派(symbolic)和采用概率方法的随机派(stochastic)两大阵营,进而引发了数十年有关这两种方法孰优孰劣的争执。同年,人工智能诞生以后,自然语言处理迅速融入了人工智能的研究中。随机派学者在这一时期利用贝叶斯方法等统计学原理取得了一定的进步;而以 Chomsky 为代表的符号派也进行了形式语言理论生成句法和形

式逻辑系统的研究。由于这一时期多数学者注重研究推理和逻辑问题，只有少数学者在研究统计方法和神经网络，因此符号派的势头明显强于随机派的势头。

- ◎ 1967 年，美国心理学家 Neisser 提出了认知心理学，从而把自然语言处理与人类的认知联系起来。
- ◎ 20 世纪 70 年代初，由于自然语言处理研究中的一些问题未能在短时间内得到解决，而新的问题又不断涌现，许多人因此丧失了信心，自然语言处理的研究进入了低谷时期。尽管如此，一些发达国家的学者依旧没有停止。基于隐马尔可夫模型（Hidden Markov Model, HMM）的统计方法和话语分析（Discourse Analysis）在这一时期取得了重大进展。
- ◎ 20 世纪 80 年代，在人们对过去的工作进行反思之后，有限状态模型和经验主义的研究方法开始复苏。
- ◎ 20 世纪 90 年代以后，随着计算机的速度和存储量大幅提高，自然语言处理的物质基础大幅改善，语音和语言处理的商品化开发成为可能。同时，网络技术的发展和 Internet 的逐步商业化使得基于自然语言的信息检索和信息抽取的需求变得更加突出，自然语言处理的应用面渐渐不再局限于机器翻译、语音控制等早期研究领域。
- ◎ 从 20 世纪 90 年代末到 21 世纪初，人们逐渐认识到仅用基于规则的方法或仅用基于统计的方法，都是无法成功进行自然语言处理的。基于统计、基于实例和基于规则的语料库技术在这一时期开始蓬勃发展，各种处理技术开始融合，自然语言处理的研究又开始兴旺起来。

1.1.3 自然语言处理的工作原理

计算机处理自然语言的过程：形式化描述——数学模型算法化——程序化——实用化。具体步骤如下。

- （1）形式化描述：把需要研究的问题在语言上建立形式化模型，使其可以以数学形式表示出来。
- （2）数学模型算法化：把数学模型表示为算法的过程称为“算法化”。
- （3）程序化：计算机根据算法进行实现，建立各种自然语言处理系统，这个过程是“程序化”。
- （4）实用化：对系统进行评测和改进最终满足现实需求，这个过程是“实用化”。

1.1.4 自然语言处理的应用前景

随着自然语言处理的蓬勃发展和深入研究，新的应用方向会不断呈现出来。自然语言处理发展前景十分广阔，主要研究领域如下。

- ◎ 文本方面：基于自然语言理解的智能搜索引擎和智能检索、智能机器翻译、自动摘要与文本综合、文本分类与文件整理、智能自动作文系统、自动判卷系统、信息过滤与垃圾邮件处理、文学研究与古文研究、语法校对、文本数据挖掘与智能决策、基于自然语言的计算机程序设计等。
- ◎ 语音方面：机器同声传译、智能远程教学与答疑、语音控制、智能客户服务、机器聊天与智能参谋、智能交通信息服务、智能解说与体育新闻实时解说、语音挖掘与多媒体挖掘、多媒体信息提取与文本转化、残疾人智能帮助系统等。

1.2 开发工具与环境

1.2.1 Sublime Text 和 Anaconda 介绍

Sublime Text 简介

Sublime Text 是一套跨平台的文本编辑器，支持基于 Python 的插件。Sublime Text 是专有软件，可通过 Package 包扩充本身的功能。大多数的包使用自由软件授权发布，并由社区建设维护。Sublime Text 是由程序员 Jon Skinner 于 2008 年 1 月份开发出来的，它最初被设计为一个具有丰富扩展功能的 Vim，具有漂亮的用户界面和强大的功能。例如，代码缩略图、Python 的插件、代码段等，还可自定义键绑定、菜单和工具栏。Sublime Text 的主要功能包括拼写检查、书签、完整的 Python API、Goto 功能、即时项目切换、多选择、多窗口，等等。Sublime Text 是一个跨平台的编辑器，同时支持 Windows、Linux、Mac OS X 等操作系统。

Sublime Text 支持众多编程语言，并支持语法上色。内置支持的编程语言包含：ActionScript、AppleScript、ASP、batch files、C、C++、C#、Clojure、CSS、D、Diff、Erlang、Go、Graphviz (DOT)、Groovy、Haskell、HTML、Java、JSP、JavaScript、JSON、LaTeX、Lisp、Lua、Makefiles、Markdown、MATLAB、Objective-C、OCaml、Perl、PHP、Python、R、Rails、Regular Expressions、reStructuredText、Ruby、Scala、shell scripts (Bash)、SQL、Tcl、Textile、XML、XSL 和 YAML。用户可通过下载外挂支持更多的编程语言。

Sublime Text 的优点

- ◎ 主流前端开发编辑器；
- ◎ 体积较小，运行速度快；
- ◎ 文本功能强大；
- ◎ 支持编译功能且可在控制台看到输出；
- ◎ 内嵌 Python 解释器支持插件开发以达到可扩展目的；
- ◎ Package Control：ST 支持的大量插件可通过其进行管理。

Anaconda 简介

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux、macOS、Windows 系统，提供了包管理与环境管理的功能，可以很方便地解决多版本 Python 并存、切换，以及各种第三方包安装问题。Anaconda 利用工具/命令 conda 来进行 package 和 environment 的管理，并且已经包含了 Python 和相关的配套工具。这里先解释下 conda、anaconda 这些概念的差别。conda 可以理解为一个工具，也是一个可执行命令，其核心功能是包管理与环境管理。包管理与 pip 的使用类似，环境管理则允许用户方便地安装不同版本的 Python 并可以快速切换。Anaconda 则是一个打包的集合，里面预装好了 conda、某个版本的 Python、众多 packages、科学计算工具等，所以也称为 Python 的一种发行版。其实还有 Miniconda，顾名思义，它只包含最基本的内容——Python 与 conda，以及相关的必须依赖项，对于空间要求严格的用户，Miniconda 是一个不错的选择。

1.2.2 开发环境的安装与配置

工具包的准备

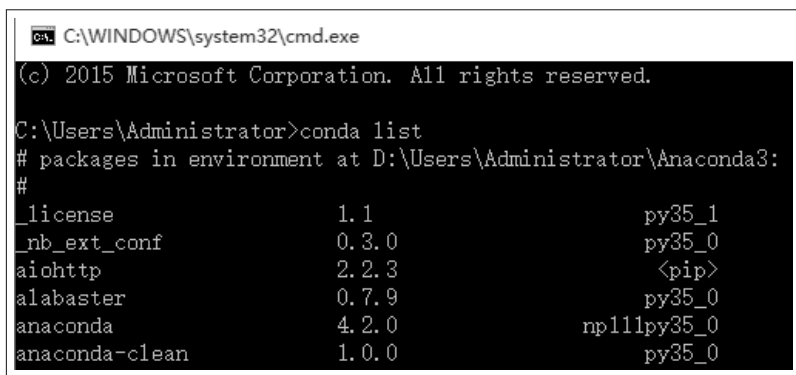
本书主要介绍 Windows 下的安装配置，关于 Linux 和 macOS 下的安装，后文给出了扩展文章，需要的读者可自行在线查看。Sublime Text 3 安装包下载地址：<http://www.sublimetext.com/3>。单击该网址进入 Sublime 主页，根据本机操作系统选择相关工具包下载。本书示范下载 Windows64 bit。

Anaconda 安装包下载地址：<https://www.anaconda.com/download/>。进入下载页面，显示 Python 3.0 以上版本和 Python 2.0 以上版本。关于 Python 3 和 Python 2 的区别请访

访问<http://www.runoob.com/Python/Python-2x-3x.html>查看，在此不再赘述。一般推荐下载 Python 3.0 以上版本。

安装 Anaconda

- (1) 安装 Anaconda 集成环境，将下载后的 Anaconda 包双击打开。
- (2) 一直单击“Next”，直到完成配置（环境变量自动配置）。配置完成后，查看是否成功。可以打开“主菜单”→“所有应用”查看安装是否成功。
- (3) 打开 cmd 进入 DOS 命令系统，输入 conda list 查看集成的 Python 包，如图1-1所示。



```

C:\WINDOWS\system32\cmd.exe
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>conda list
# packages in environment at D:\Users\Administrator\Anaconda3:
#
_license                  1.1                      py35_1
_nb_ext_conf              0.3.0                   py35_0
aiohttp                   2.2.3                   <pip>
alabaster                 0.7.9                   py35_0
anaconda                  4.2.0                   np111py35_0
anaconda-clean            1.0.0                   py35_0
  
```

图 1-1 查看 Python 集成包

- (4) 如果想添加新的 Python 包，则可以打开 Anaconda 官网 <https://anaconda.org/search> 进行查找，比如想找到机器学习工具包 scikit-learn，如图1-2所示。

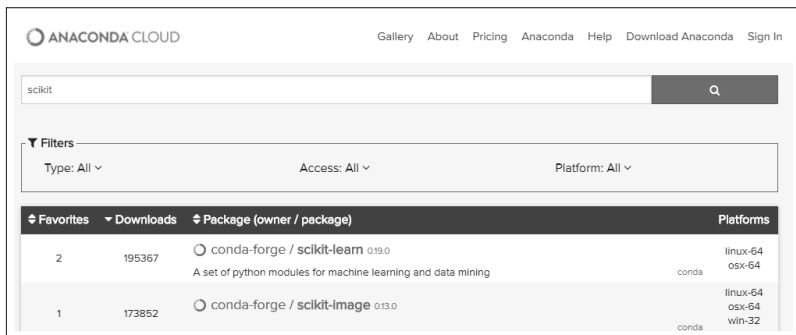


图 1-2 下载 scikit-learn 包

至此我们就完成了 Anaconda 的安装配置工作，以及对包文件的自定义下载安装。需要注意的是，Anaconda 自身集成了 Python、pip、nltk、numpy、matplotlib 等一系列常用包。现在，我们已经可以对 Python 进行操作了。考虑到熟悉 Python 开发的人员常用 Pycharm 开发工具，熟悉 Java 的开发人员常用 Eclipse 开发工具，熟悉 C# 的开发人员常用 VS 开发工具。我们将 Anaconda 集成到 PyDev、Pycharm、Eclipse、VS 等编译环境即可，诸如此类就不一一列举了。考虑到新学一种语言要重新学习一种编程环境，这样极其不方便。那么能不能找到一款编程工具可以通用以上语言？或许这样还不够，如果它还能跨 Linux、Windows、macOS 那就更好了。本书强烈推荐的 Sublime 跨平台跨语言编辑器事实上就是这样一款强大的工具。我们接下来唯一要做的，就是将 Anaconda 集成到 Sublime 中就可以了。扩展：Linux 和 macOS 安装教程请访问 <https://docs.continuum.io/anaconda/install/>。

安装 Sublime Text 3

- (1) 双击下载好的 Sublime Text 3 工具包。
- (2) 一直执行“Next”安装即可，中间的保存路径可以自定义。最终安装成功，如图1-3所示。
- (3) 安装插件 Package Control。打开 <https://packagecontrol.io/installation> 复制 Sublime Text 3 中的代码，如图1-4所示。
- (4) 单击“Ctrl+`”，将图 1-4 中的文本代码内容复制粘贴到文本框中，按“Enter”键即可，如图1-5所示。
- (5) 成功安装后，在 Sublime Text 3 中同时按住“Ctrl+Shift+P”键，最终 Package Control 安装成功，如图1-6所示。

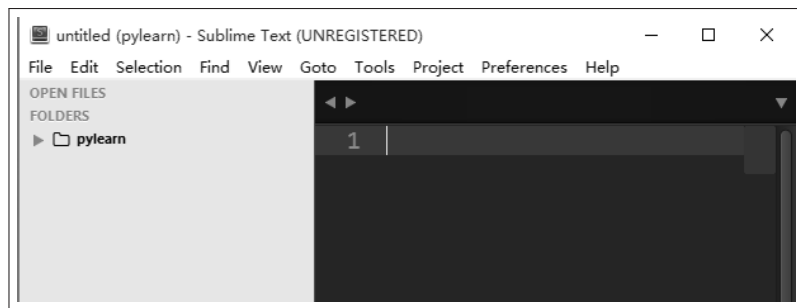


图 1-3 Sublime Text 3 安装成功

```

SUBLIME TEXT 3  SUBLIME TEXT 2
import urllib.request,os,hashlib; h =
'df21e130d211cfc94d9b0905775a7c0f' +
'1e3d39e33b79698005270310898eea76'; pf = 'Package Control.sublime-
package'; ipp = sublime.installed_packages_path();
urllib.request.install_opener( urllib.request.build_opener(
urllib.request.ProxyHandler()) ); by = urllib.request.urlopen(
'http://packagecontrol.io/' + pf.replace(' ', '%20')).read(); dh =
hashlib.sha256(by).hexdigest(); print('Error validating download
(got %s instead of %s), please try manual install' % (dh, h)) if
dh != h else open(os.path.join( ipp, pf), 'wb' ).write(by)

```

图 1-4 Package Control 代码

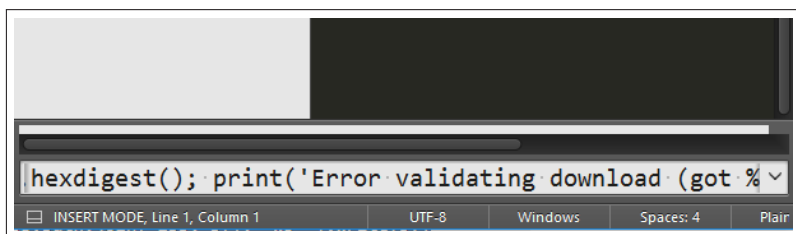


图 1-5 安装 Package Control

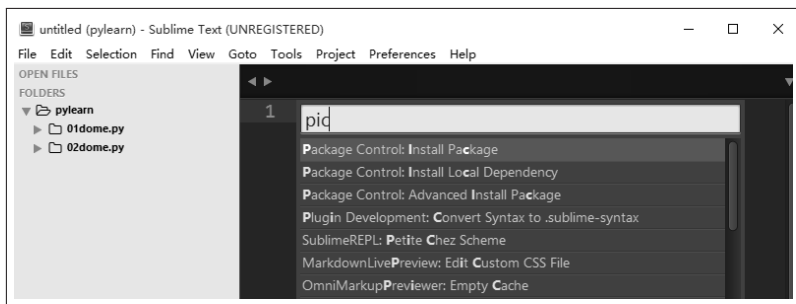


图 1-6 成功安装 Package Control

- (6) 单击“Package Control:Install Package”进入并查找 Python 环境配置插件“SublimeREPL”，下载安装完成后，单击“Preferences”→“Browse Package...”查看安装的包，如图1-7所示。

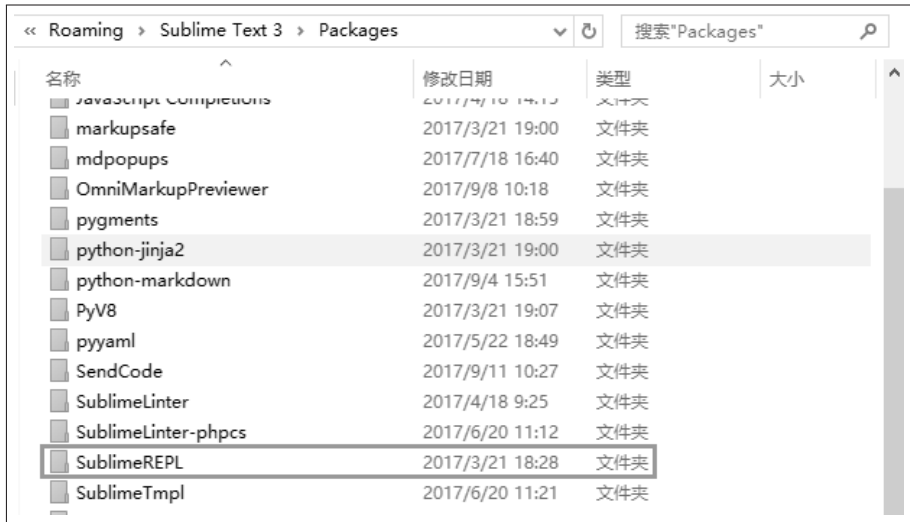


图 1-7 查看安装插件

- (7) 自定义快捷键配置：打开“Preferences”→“Key Bindings”输入如下代码，按 F5 运行程序，按 F6 切换 IDEL 工具，按“Ctrl+D”自定义删除行，其他快捷键是通用的，网上有很多快捷键的资料，这里不赘述了。

```
[
{
  "keys": ["f5"],
  "caption": "SublimeREPL: Python - RUN current file",
  "command": "run_existing_window_command",
  "args": {
    "id": "repl_python_run",
    "file": "config/Python/Main.sublime-menu"
  }
}, {
  "keys": ["f6"],
  "caption": "SublimeREPL: Python",
  "command": "run_existing_window_command",
  "args": {
    "id": "repl_python",
    "file": "config/Python/Main.sublime-menu"
  }
}
```

```

}, {
    "keys": ["ctrl+d"],
    "command": "run_macro_file",
    "args": {"file": "res://Packages/Default/Delete Line.sublime-macro"}
}
]

```

至此完成了 Sublime Text3 的安装配置工作,详细插件安装参考 <http://www.open-open.com/news/view/26d731>, 快捷键使用请查看 <https://segmentfault.com/a/1190000004463984>。其余问题读者可以自行上网检索,由于资料较多且比较容易实现,本书不再详写。

1.3 实战：第一个小程序的诞生

1.3.1 实例介绍

编写一个可以智能数据计算的小程序,用户输入公式如“10/(2+3)”,自动提示计算结果。

1.3.2 源码实现

本实例设计思路如下:

- ◎ a 是采用 def 定义函数名,Python 不采用花括号,而是用冒号代替代码块,形参中 param 是一个自动识别类型的参数。
- ◎ b 是基本的计算公式,记住结尾没有分号。
- ◎ c 是对结果的输入。
- ◎ d 类似于 C、C#、Java 中的主函数,后面章节会介绍。
- ◎ e 是对函数名的调用,并且直接传递列表参数,暂时不理解也没有关系,详见第 2 章。

```

def countNum(param):                                a
    result = param[0]/(param[1]+param[2])           b

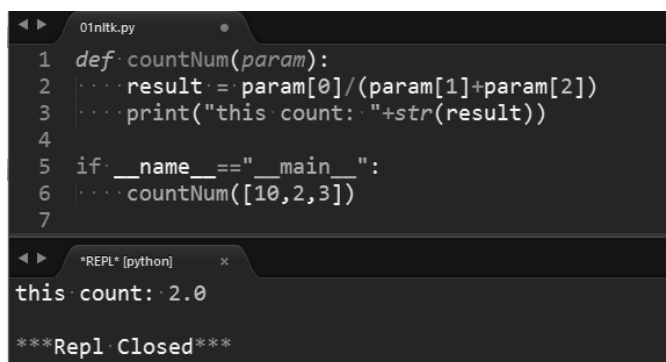
```

```
print("this count: "+str(result))      c

if __name__=="__main__":              d
    countNum([10,2,3])                  e
```

源码下载见 <https://github.com/BaiNingchao/NLP&ML/01chapter/1.1.py>。

运行结果如图1-8所示。



```
01nltk.py
1 def countNum(param):
2     ... result = param[0]/(param[1]+param[2])
3     ... print("this count: "+str(result))
4
5 if __name__=="__main__":
6     ... countNum([10,2,3])
7

*REPL* [python] x
this count: 2.0

***Repl Closed***
```

图 1-8 第一个小程序

第 2 章

快速上手 Python

导读：Python 作为一门简洁优美且功能强大的语言，越来越受编程人员的喜欢，在工业界和学术界都非常受欢迎。本书的全部代码都是通过 Python 实现的，之所以选择 Python 语言，是因为其可以跨平台、跨应用开发，因此本章旨在帮助读者快速了解 Python 的概貌。如果读者已经具备 Python 基础，可略过此章。本章首先介绍 Python 语言及其用途。然后介绍 Python 进阶内容，以实际案例演示常用的语句和控制流、表达式、函数、数据结构、标准库等知识点。最后扩展到 Python 第三方库，使读者对 Python 有全面的理解和认识。

2.1 初识 Python 编程语言

2.1.1 Python 概述

Python 介绍

Python 是一门面向对象、直译式的计算机程序语言。它包含了一组功能完备的标准库，能够轻松完成很多常见的任务。它的语法简单，与大多数程序设计语言使用大括号不一样，它通过缩进来定义语句块。Python 同样是一门动态语言，具备垃圾回收功能，能够自动管理内存使用。Python 经常被当作脚本语言用于处理系统管理任务和网络程序编写，它也非常适合完成各种高级任务。Python 支持命令式程序设计、面向对象程序设计、函数式编程、面向侧面的程序设计、泛型编程等多种编程范式。Python 是完全面向对象的语言。函数、模块、数字、字符串都是对象。Python 完全支持继承、重载、派生、多重继承，有益于增强源代码的复用性。Python 支持重载运算符，因此 Python 也支持泛型设计。

Python 的发展历史

Python 的创始人是 Guido van Rossum。1989 年的圣诞节期间，Guido van Rossum 为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序，作为 ABC 语言的一种继承。之所以选中 Python 作为程序的名字，是因为 Guido 是 BBC 电视剧《蒙提·派森的飞行马戏团》（Monty Python's Flying Circus）的爱好者。ABC 是由 Guido 参加设计的一种教学语言。在 Guido 本人看来，ABC 这种语言非常优美和强大，是专门为非专业程序员设计的。但是 ABC 语言并没有成功，究其原因，Guido 认为是非开放造成的。Guido 决心在 Python 中避免这一错误，并获取了非常好的效果，完美结合了 C 和一些其他语言。就这样，Python 在 Guido 手中诞生了。目前，Guido 仍然是 Python 的主要开发者，决定整个 Python 语言的发展方向。Python 社区经常称呼他是仁慈的独裁者。

Python 标准库的主要功能

- ◎ 文本处理，包含文本格式化、正则表达式匹配、文本差异计算与合并、Unicode 支持、二进制数据处理等功能。
- ◎ 文件处理，包含文件操作、创建临时文件、文件压缩与归档、操作配置文件等功能。
- ◎ 操作系统功能，包含线程与进程支持、I/O 复用、日期与时间处理、调用系统函数、日志（logging）等功能。
- ◎ 网络通信，包含网络套接字、SSL 加密通信、异步网络通信等功能。
- ◎ 网络协议，支持 HTTP、FTP、SMTP、POP、IMAP、NNTP、XMLRPC 等多种网络协议，并提供了编写网络服务器的框架。
- ◎ W3C 格式支持，包含 HTML、SGML、XML 的处理。
- ◎ 其他功能，包括国际化支持、数学运算、Hash、Tkinter 等。

Python 的优缺点

优点

- ◎ 简单、易学、免费、开源、高层语言（无须考虑如何管理程序使用内存等细节问题）。
- ◎ 可移植性（这些平台包括 Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE、PocketPC、Symbian，以及 Google 基于 Linux 开发的 Android 平台）。

- ◎ 解释性、面向对象、可扩展性（可以部分程序用 C 或 C++ 编写，然后在 Python 程序中使用它们）。
- ◎ 可嵌入性（可以把 Python 嵌入 C/C++ 程序，从而向程序用户提供脚本功能）、丰富的库、规范的代码。

缺点

- ◎ 单行语句和命令行输出问题，独特的语法，运行速度慢（与 C 和 C++ 相比）。

Python 的开发环境

通用 IDE / 文本编辑器，很多并非集成开发环境软件的文本编辑器，也对 Python 有不同程度的支持。本书默认开发环境均集成在 Anaconda 中，第 1 章已经详细介绍过。此外，还有如下开发环境：

- ◎ Eclipse + pydev 插件，目前对 Python 3.X 只支持到 3.0。
- ◎ emacs + 插件。
- ◎ NetBeans + 插件。
- ◎ SlickEdit。
- ◎ TextMate。
- ◎ Python Tools for Visual Studio。
- ◎ Vim + 插件。
- ◎ Sublime Text + 插件。
- ◎ EditPlus。
- ◎ UltraEdit。
- ◎ PSPad。
- ◎ Editra，由 Python 开发的程序编辑器。
- ◎ Notepad++。

2.1.2 Python 能做什么

Python 的用途

- ◎ 系统编程：提供 API，能方便地进行系统维护和管理，很多系统管理员理想的编程工具。
- ◎ 图形处理：有 PIL、Tkinter 等图形库支持，能方便地进行图形处理。
- ◎ 数学处理：NumPy 扩展提供大量与许多标准数学库的接口。
- ◎ 文本处理：Python 提供的 re 模块能支持正则表达式，还提供 SGML、XML 分析模块，许多程序员利用 Python 进行 XML 程序的开发。
- ◎ 数据库编程：程序员可通过遵循 Python DB-API（数据库应用程序编程接口）规范的模块与 Microsoft SQL Server、Oracle、Sybase、DB2、MySQL、SQLite 等数据库通信。Python 自带一个 Gadget 模块，提供了一个完整的 SQL 环境。
- ◎ 网络编程：提供丰富的模块支持 Sockets 编程，能方便快速地开发分布式应用程序。
- ◎ Web 编程：应用的开发语言，支持最新的 XML 技术。
- ◎ 多媒体应用：能进行二维和三维图像处理。PyGame 模块可用于编写游戏软件。
- ◎ 黑客编程：Python 有一个 hack 的库，内置多个函数。

Python 开发的应用案例

- ◎ Reddit：社交分享网站。
- ◎ Dropbox：文件分享服务。
- ◎ 豆瓣网：图书、唱片、电影等文化产品的资料数据库网站。
- ◎ Django：鼓励快速开发的 Web 应用框架。
- ◎ Pylons：Web 应用框架。
- ◎ Zope：应用服务器。
- ◎ Plone：内容管理系统。
- ◎ TurboGears：另一个 Web 应用快速开发框架。
- ◎ Twisted：Python 的网络应用程序框架。
- ◎ Fabric：用于管理成百上千台 Linux 主机的程序库。
- ◎ Python Wikipedia Robot Framework：MediaWiki 的机器人程序。
- ◎ MoinMoinWiki：Python 写成的 Wiki 程序。

- ◎ Trac : 使用 Python 编写的 Bug 管理系统。
- ◎ Mailman : 使用 Python 编写的邮件列表软件。
- ◎ Mezzanine : 基于 Django 编写的内容管理系统系统。
- ◎ Flask : Python 微 Web 框架。
- ◎ Webpy : Python 微 Web 框架。
- ◎ Bottle : Python 微 Web 框架。
- ◎ EVE : 网络游戏 EVE 大量使用 Python 进行开发。
- ◎ Blender : 使用 Python 作为建模工具与 GUI 语言的开源 3D 绘图软件。
- ◎ Inkscape : 一个开源的 SVG 矢量图形编辑器。
- ◎ 知乎: 一个问答网站。
- ◎ 果壳: 一个泛科技主题网站。

2.1.3 Python 的语法和特点

◎ 编码

默认情况下, Python 3 源码文件以 UTF-8 编码, 所有字符串都是 Unicode 字符串。当然也可以为源码文件指定不同的编码:

```
# -*- coding: cp-1252 -*-
```

◎ 标识符

标识符书写规则。

在 Python 中, 标识符由字母、数字、下画线组成。

在 Python 中, 所有标识符可以包括英文、数字及下画线 (_), 但不能以数字开头。

Python 中的标识符是区分大小写的。

以下画线开头的标识符是有特殊意义的:

以单下画线开头 (_foo) 的代表不能直接访问的类属性, 需通过类提供的接口进行访问, 不能用 "from xxx import *" 导入;

以双下画线开头的 (__foo) 代表类的私有成员; 以双下画线开头和结尾的 (__foo__) 代表 Python 中特殊方法专用的标识。

◎ Python 保留字

保留字即关键字，不能把它们用作任何标识符名称。

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
 'return', 'try', 'while', 'with', 'yield']
```

◎ 注释

Python 中单行注释以“#”开头，实例如下。

```
#!/usr/bin/python3

# 第一个注释
print ("Hello, Python!") # 第二个注释
```

◎ 行与缩进

Python 最具特色的就是使用缩进来表示代码块，不需要使用大括号 ({}). 用四个空格或者 Tab 进行缩进。

```
if True:
    print ("True")
else:
    print ("False")
```

◎ 多行语句

Python 通常是一行写完一条语句，但如果语句很长，我们可以使用反斜杠 (\) 来实现多行语句。例如：

```
total = item_one + \
        item_two + \
        item_three
```

在 []、{} 或 () 中的多行语句，不需要使用反斜杠 (\)，例如：

```
total = ['item_one', 'item_two', 'item_three',
        'item_four', 'item_five']
```

◎ 引号

Python 接收单引号 (')、双引号 (") 或三引号 (' ' ') 来表示字符串，引号的开始与结束必须是相同类型的。

```
word = 'word'
sentence = "这是一个句子。"
paragraph = """这是一个段落。
包含了多个语句"""
```

◎ 字符串

Python 中单引号和双引号的使用完全相同。

使用三引号 (' ' ' 或 " " ") 可以指定一个多行字符串。

转义符 '\ '。

自然字符串，通过在字符串前加 r 或 R 实现。如 r "this is a line with \n" ， \n 会显示，而不是换行。

Python 允许处理 Unicode 字符串，加前缀 u 或 U，如 u "this is an unicode string"。

字符串是不可变的。

按字面意义级联字符串，如 "this " "is " "string" 会被自动转换为 this is string。

◎ 空行

函数之间或类的方法之间用空行分隔，表示一段新的代码的开始。类和函数入口之间也用一行空行分隔，以突出函数入口的开始。

◎ print（输出）

print 默认输出是换行的，如果要实现不换行，则需要变量末尾加上 end= " "：

```
#!/usr/bin/python3
```

```
x="a"
y="b"
# 换行输出
print( x )
```

```

print( y )

print('-----')
# 不换行输出
print( x, end=" " )
print( y, end=" " )
print()

```

运行结果:

```

a
b
-----
a b

```

◎ import 与 from...import

在 Python 中用 import 或者 from...import 导入相应的模块。

- (1) 将整个模块 (somemodule) 导入, 格式为 import somemodule。
- (2) 从某个模块中导入某个函数, 格式为 from somemodule import somefunction。
- (3) 从某个模块中导入多个函数, 格式为 from somemodule import firstfunc, secondfunc, thirdfunc。
- (4) 将某个模块中的全部函数导入, 格式为 from somemodule import *。

```

导入sys模块
import sys
print('=====Python import mode=====');
print ('命令行参数为:')
for i in sys.argv:
    print (i)
print ('\n Python 路径为',sys.path)

```

```

导入sys模块的argv,path成员
from sys import argv,path # 导入特定的成员

```

```
print('=====Python from import
=====')
print('path:',path) # 因为已经导入path成员，所以引用时不需要加sys.path
```

数据类型

Python 的数据类型如表 2-1 所示。

表 2-1 Python 数据类型

类 型	描 述	例 子
str	一个由字符组成的不可更改的有序列。在 Python 3.x 里，字符串由 Unicode 字符组成	Wikipedia、Wikipedia、Spanning、multiple、lines
bytes	一个由字节组成的不可更改的有序列	b'Some ASCII' b''Some ASCII ''
list	可以包含多种类型的可改变的有序列	[4.0, 'string', True]
tuple	可以包含多种类型的不可改变的有序列	(4.0, 'string', True)
set/frozenset	与数学中集合的概念类似。无序的、每个元素唯一	{4.0, 'string', True}/frozenset([4.0, 'string', True])
dict 或 map	一个可改变的由键值对组成的无序列	{'key1': 1.0, 3: False}
int	精度不限的整数	42
float	浮点数。精度与系统相关	3.1415927
complex	复数	3+2.7j
bool	逻辑值。只有两个值：真、假	True、False

2.2 Python 进阶

2.2.1 Hello World

编程第一步，打印“Hello World”的输入结果：

```
def callHello():  
    print("hello world! ")  
  
callHello()
```

运行结果：

```
hello world!
```

2.2.2 语句和控制流

◎ if 语句：根据成绩评判等级

```
# if 语句：根据成绩评判等级  
def CallLevel(score):  
    if score >= 90 :  
        print("优秀")  
    elif score >= 60:  
        print("及格")  
    else:  
        print("不及格")  
CallLevel(score=60)
```

运行结果：

```
及格
```

◎ for 语句：循环输出所有评分指标

```
# for语句:循环输出所有评分指标
def GetLevel(score):
    for lev in score:
        print(lev,end=" ") # 设置不换行
GetLevel(score=["优秀","及格","不及格"]) # 列表参数
```

运行结果:

优秀 及格 不及格

④ while 语句: 循环输出所有评分

```
# while语句:循环输出所有评分
def GetLevel2(score):
    countlen=len(score)
    while countlen > 0:
        print(score[countlen-1],end=" ") # 设置不换行
        countlen -= 1

GetLevel2(score=[90,30,100,98,60])
```

运行结果:

60 98 100 30 90

④ range() 函数: 循环输出所有评分指标的下标

```
# range()函数:循环输出所有评分指标的下标
def GetValue(score):
    for lev in range(len(score)):
        print(lev,end=" ")
GetValue(score=["优秀","及格","不及格"])
```

运行结果:

0 1 2

④ break 语句: 统计成绩优秀学生的人数

```
# break语句:不及格的跳过
def GetHighLev(score):
    result=0
    for lev in score:
        if lev < 90:
            break
        else:
            result += 1
    print("成绩优秀的学生有: "+str(result)+"位。")

GetHighLev(score=[90,30,100,98,60])
```

运行结果:

成绩优秀的学生有: 1位。

分析: 实际优秀者是90、100、98共3位, 输出结果却是1位。造成这种结果的原因是, 当循环输入列表90时, 满足条件自动加1。继续输入30, 不满足条件, 直接跳出整个程序, 输出最后一条语句。

◎ continue 语句: 统计优秀成绩的个数

```
# continue语句:统计优秀成绩的个数
def GetHighLev2(score):
    result=0
    for lev in score:
        if lev < 90:
            continue
        else:
            result += 1
    print("成绩优秀的学生有: "+str(result)+"位。")

GetHighLev2(score=[90,30,100,98,60])
```

运行结果:

成绩优秀的学生有: 3位。

◎ pass 语句：什么也不做，占位符

```
# pass语句：什么也不做，占位符
def callPass():
    pass
print(callPass())
```

运行结果：

None

2.2.3 函数

◎ 定义函数：斐波那契数列

```
# 输出指定数的斐波那契数列
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
fib(100)
```

运行结果：

0 1 1 2 3 5 8 13 21 34 55 89

结果分析：关键字 `def` 引入了一个函数定义。在其后必须跟函数名和包括形式参数的圆括号。函数体语句从下一行开始，必须是缩进的。一个函数定义会在当前符号表内引入函数名。函数名指代的值（即函数体）有一个被 Python 解释器认定为用户自定义函数的类型。

◎ 函数默认参数

```
# 函数默认参数
def sayhello(name="Tom"):
    print("Hello,"+name)

sayhello()
```

运行结果:

```
Hello,Tom
```

◎ 关键字参数

```
# 关键字参数
def person(name, age, **kw): #前两个是必须有的参数，最后一个是可选可变参数
    print('name:', name, 'age:', age, 'other:', kw)

person("Tome",30) # 只调用必须参数
person("Tom",30,city="ChengDu",sex="man") # 自定义关键字参数
```

运行结果:

```
name: Tome age: 30 other: {}
name: Tom age: 30 other: {'sex': 'man', 'city': 'ChengDu'}
```

◎ 可变参数

```
# 可变参数
def concat(*args, sep="/"):
    print(sep.join(args))
concat('我','是','可变','参数')
```

运行结果:

我/是/可变/参数

◎ Lambda 形式

```
# Lambda 形式
def Lambda(nums):
    nums.sort(key=lambda num: num[0])
    print(nums)

Lambda(nums = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')])
```

运行结果:

```
[(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]
```

2.2.4 List 列表

◎ list 常用方法

<code>list.append(x)</code>	添加元素到列表尾部
<code>list.extend(L)</code>	列表合并
<code>list.insert(i, x)</code>	在指定位置插入一个元素
<code>list.remove(x)</code>	删除列表中值为 <code>x</code> 的第一个元素
<code>list.pop([i])</code>	从列表的指定位置删除元素, 并将其返回
<code>list.clear()</code>	从列表中删除所有元素, 相当于 <code>del a[:]</code>
<code>list.index(x)</code>	返回列表中第一个值为 <code>x</code> 的元素的索引
<code>list.count(x)</code>	返回 <code>x</code> 在列表中出现的次数
<code>list.sort()</code>	对列表中的元素就地进行排序
<code>list.reverse()</code>	就地倒排列表中的元素
<code>list.copy()</code>	返回列表的一个浅拷贝, 等同于 <code>a[:]</code>

◎ 列表的切分

```
# 列表的切分
def calllist(names):
    print(names[-1:]) # 输出列表最后一个值
    print(names[:3])  # 输出列表前3个值
calllist(names=['this','is','a','list'])
```

运行结果:

```
['list']
['this', 'is', 'a']
```

◎ List：列表作堆栈（先进先出）

```
# 把列表当作堆栈使用
def SomeList(stack):
    print("原始列表（栈）:",end=' ')
    print(stack)
    stack.append('贺知章')
    stack.append('杜牧')
    print("追加后列表（栈）:",end=' ')
    print(stack)
    stack.pop()
    stack.pop()
    print("出栈后的数据:",end=' ')
    print(stack)

SomeList(stack=['李白','杜甫','白居易'])
```

运行结果（先进先出）:

```
原始列表（栈）:  ['李白', '杜甫', '白居易']
追加后列表（栈）:  ['李白', '杜甫', '白居易', '贺知章', '杜牧']
出栈后的数据:    ['李白', '杜甫', '白居易']
```

◎ List：列表作队列（先进后出）

把列表当作队列使用：使用队列时调用`collections.deque`，它为在首尾两端快速插入和删除而设计。

```
from collections import deque

def SomeList2(queue):
    print("原始列表: ",end=' ')
    print(queue)
    queue.append("李商隐")
    queue.append("杜牧")
    print("入队的列表: ",end=' ')
    print(queue)
    queue.popleft()
    queue.popleft()
    print("出队后列表: ",end=' ')
    print(queue)

SomeList2(queue = deque(['李白','杜甫','白居易']))
```

运行结果（先进后出）:

```
原始列表: deque(['李白', '杜甫', '白居易'])
入队的列表: deque(['李白', '杜甫', '白居易', '李商隐', '杜牧'])
出队后列表: deque(['白居易', '李商隐', '杜牧'])
```

◎ 列表推导式

```
# 列表推导式
def callList(nums):
    squares = [n**2 for n in nums]
    print(squares)
callList(nums=[2,4,6,8])
```

运行结果:

```
[4, 16, 36, 64]
```

◎ 列表的矩阵转秩

```
# 矩阵转秩
def countList(matrix):
    result = [[row[i] for row in matrix] for i in range(4)]
    print(result)

matrix = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12],
]
countList(matrix)
```

运行结果:

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

2.2.5 元组

虽然元组和列表很类似，但它们经常用在不同的情况和不同的用途上。元组有很多用途，例如， (x,y) 坐标对、数据库中的员工记录，等等。元组就像字符串，是不可变的；列表是可变的，它们的元素通常是相同类型的，需要通过迭代访问。

◎ 操作元组

```
def calltuple(tuples):
    for t in tuples:
        print(t+"\t",end=" ")
    print()
calltuple(tuples=('百度','阿里巴巴','腾讯')) # 元组参数
```

运行结果:

```
百度    阿里巴巴    腾讯
```

2.2.6 set 集合

集合是一个无序不重复元素的集。基本功能包括关系测试和消除重复元素。集合对象还支持 union（联合）、intersection（交）、difference（差）和 symmetric difference（对称差集）等数学运算。

◎ 操作 set 集合

```
def callset(basket):
    result= set(basket)
    print(result)

callset(basket={'apple', 'orange', 'apple', 'pear', 'orange', 'banana'})
```

运行结果:

```
{'pear', 'orange', 'apple', 'banana'}
```

2.2.7 字典

理解字典的最佳方式是把它看作无序的键: 值对（key:value 对）集合，键必须是互不相同的（在同一个字典之内）。一对大括号创建一个空的字典：{}。初始化列表时，在大括号内放置一组逗号分隔的键: 值对，这也是字典输出的方式。

◎ 操作字典

```
def calldict(dict):
    print("原始字典",end=' ')
    print(dict)    # 原始字典
    dict['欧阳修'] = '宋朝' # 追加字典
    print("追加后的字典",end=' ')
    print(dict)    # 追加后的字典
    print("字典键的集合",end=' ')
    print(list(dict.keys())) # 字典键的集合
```

```

print("字典键的排序",end=' ')
print(sorted(dicts.keys())) # 字典键的排序
print("字典值的集合",end=' ')
print(list(dicts.values())) # 字典值的集合

calldict(dicts = {'李白': '唐朝', '杜甫': '唐朝'})

```

运行结果:

```

原始字典 {'杜甫': '唐朝', '李白': '唐朝'}
追加后的字典 {'杜甫': '唐朝', '李白': '唐朝', '欧阳修': '宋朝'}
字典键的集合 ['杜甫', '李白', '欧阳修']
字典键的排序 ['李白', '杜甫', '欧阳修']
字典值的集合 ['唐朝', '唐朝', '宋朝']

```

2.2.8 面向对象编程：类

◎ 通过案例理解 Python 中的类

父类是动物类，有初始化函数，且有动物讲话的方法；子类是一个狗类，继承父类的所有属性，并扩展自己的方法，调用子类讲话方法，并直接调用父类讲话方法。

运行结果:

```

"""
# 欢迎进入我的主页: http://www.cnblogs.com/baiboy/
"""

class BaseAnimal: # 父类: 动物
    def __init__(self,name,age): # 初始化方法: 括号里是形参
        self.name=name
        self.age=age
    def speak(self): # 父类的行为方法
        print("我的名字是[ %s ],今年[ %d ]岁" %(self.name,self.age))

```

```

class SubDog(BaseAnimal): # 子类: 小狗
    def __init__(self,name,age,say): # 初始化方法: 括号里是形参
        BaseAnimal.__init__(self,name,age)
        self.say=say
        print("这是子类[ %s ]."%(self.name))
        print('_'*20+'调用子函数方法'+_'*20)
    def talk(self):          # 子类的行为方法
        # BaseAnimal.speak(self) # 调用父类的行为方法
        print("我的名字是[ %s ],今年[ %d ]岁,我想说:  %s" %(self.name,
            self.age,self.say))

ani=SubDog('dog',12,'汪汪...')
print(ani.talk())
print('_'*20+'直接调用父函数方法'+_'*20)
BaseAnimal('tom',13).speak()

```

运行结果:

这是子类[dog].

-----调用子函数方法-----

我的名字是[dog],今年[12]岁,我想说: 汪汪...

-----直接调用父函数方法-----

我的名字是[tom],今年[13]岁

__init__ 方法 (双下画线): __init__ 方法在类的一个对象被建立时马上运行。这个方法可以用来初始化对象。注意, 这个名称的开始和结尾都是双下画线。我们把__init__方法定义为取一个参数 name (以及普通的参数 self)。在 __init__ 方法里, 我们只是创建一个新的域, 也称为 name。注意: 它们是两个不同的变量, 尽管它们有相同的名字。点号使我们能够区分它们。最重要的是, 在创建一个类的新实例时, 把参数包括在圆括号内跟在类名后面, 从而传递给 __init__ 方法。这是这种方法的重要之处。现在, 我们能够在方法中使用 self.name 域。注释 __init__ 方法类似于 C++、C# 和 Java 中的 constructor。

2.2.9 标准库

Python 拥有一个强大的标准库。Python 语言的核心只包含数字、字符串、列表、字典、文件等常见类型和函数，而由 Python 标准库提供了系统管理、网络通信、文本处理、数据库接口、图形系统、XML 处理等额外的功能。

Python 社区提供了大量的第三方模块，使用方式与标准库类似。它们的功能覆盖科学计算、Web 开发、数据库接口、图形系统多个领域。第三方模块可以使用 Python 或者 C 语言编写。SWIG、SIP 常用于将 C 语言编写的程序库转化为 Python 模块。Boost C++ Libraries 包含了一组库（Boost.Python），使得以 Python 或 C++ 编写的程序能互相调用。Python 常被称为其他语言与工具之间的“胶水”语言。

2.3 Python 深入——第三方库

2.3.1 Web 框架

- ◎ Django：开源 Web 开发框架，它鼓励快速开发并遵循 MVC 设计，开发周期短。
- ◎ Flask：轻量级的 Web 框架。
- ◎ Pyramid：轻量，同时有可以规模化的 Web 框架，Pylon projects 的一部分。
- ◎ ActiveGrid：企业级的 Web 2.0 解决方案。
- ◎ Karrigell：简单的 Web 框架，自身包含了 Web 服务，py 脚本引擎和纯 Python 的数据库 PyDBLite。
- ◎ Tornado：一个轻量级的 Web 框架，内置非阻塞式服务器，而且速度相当快。
- ◎ webpy：一个小巧灵活的 Web 框架，虽然简单但功能强大。
- ◎ CherryPy：基于 Python 的 Web 应用程序开发框架。
- ◎ Pylons：基于 Python 的一个极其高效和可靠的 Web 开发框架。
- ◎ Zope：开源的 Web 应用服务器。
- ◎ TurboGears：基于 Python 的 MVC 风格的 Web 应用程序框架。
- ◎ Twisted：流行的网络编程库，大型 Web 框架。
- ◎ Quixote：Web 开发框架。

2.3.2 科学计算

- ◎ Matplotlib: 用 Python 实现的类 MATLAB 的第三方库, 用来绘制一些高质量的数学二维图形。
- ◎ Pandas: 用于数据分析、数据建模、数据可视化的第三方库。
- ◎ SciPy: 基于 Python 的 MATLAB 实现, 旨在实现 MATLAB 的所有功能。
- ◎ NumPy: 基于 Python 的科学计算第三方库, 提供了矩阵、线性代数、傅立叶变换等解决方案。

2.3.3 GUI

- ◎ PyGtk: 基于 Python 的 GUI 程序开发 GTK+ 库。
- ◎ PyQt: 用于 Python 的 QT 开发库。
- ◎ WxPython: Python 下的 GUI 编程框架, 与 MFC 的架构相似。

2.3.4 其他库

- ◎ BeautifulSoup: 基于 Python 的 HTML/XML 解析器, 简单易用。
- ◎ gevent: Python 的一个高性能并发框架, 使用了 epoll 事件监听、协程等机制将异步调用封装为同步调用。
- ◎ PIL: 基于 Python 的图像处理库, 功能强大, 对图形文件的格式支持广泛。目前已无维护, 另一个第三方库 Pillow 实现了对 PIL 库的支持和维护。
- ◎ PyGame: 基于 Python 的多媒体开发和游戏软件开发模块。
- ◎ Py2exe: 将 Python 脚本转换为 Windows 上可以独立运行的可执行程序。
- ◎ Requests: 适合人们使用的 HTTP 库, 封装了许多烦琐的 HTTP 功能, 极大地简化了 HTTP 请求所需要的代码量。
- ◎ scikit-learn: 机器学习第三方库, 实现许多知名的机器学习算法。
- ◎ TensorFlow: Google 开发维护的开源机器学习库。

- ◎ Keras: 基于 TensorFlow, Theano 与 CNTK 的高级神经网络 API。
- ◎ SQLAlchemy: 关系型数据库的对象关系映射 (ORM) 工具。

第 3 章

线性代数

导读：自然语言处理是计算机科学、统计学、数学和信息论等多个领域交叉的学科。线性代数又是数学的一个重要分支，对自然语言处理有着直接的影响。诸如算法建模、参数设置、验证策略、识别欠拟合和过拟合，等等。读者往往知道线性代数很有用，常常全书通读，造成时间不足和效率较低，归因于对线性代数在机器学习中的重点和用途不明确。本章以简明的方式介绍常用的线性代数知识，介绍线性代数常用于哪些方面。

3.1 线性代数介绍

线性代数是数学的一个分支，它的研究对象是向量、向量空间（或称线性空间）、线性变换和有限维的线性方程组。由于科学研究中的非线性模型通常可以近似为线性模型，使得线性代数广泛地应用于自然科学和社会科学领域。

在计算机广泛应用的今天，线性代数早已不是大学课堂的理论指导，有人甚至说“线性代数是 21 世纪的数学”。在机器学习中对算法主题的理解和优化很有帮助，比如主成分分析（PCA）、奇异值分解（SVD）、矩阵特征分解、因式分解、对称矩阵、正交化/标准正交化、矩阵运算、投影、特征值和特征矢量、矢量空间和范数等。由于现在很多机器学习技术人员常常只会调用机器学习的工具包，以至于有人戏言现在的机器学习是“调参机器学习”，侧面解释了线性代数对于了解算法模型和优化很强的指导作用。

3.2 向量

3.2.1 向量定义

向量

向量也称为欧几里得向量、几何向量、矢量，指具有大小和方向的量。它可以形象化地表示为带箭头的线段。箭头所指：代表向量的方向；线段长度：代表向量的大小。与向量对应的只有大小，没有方向的量叫作标量。

零向量

始点与终点重合，也就是重合点的向量。 $\vec{0} = \vec{AA} = \vec{BB} \cdots$ ，具有方向性，但方向不定。因此，零向量与任一向量平行。

等向量

两向量长度、方向相等，即为等向量。

有向线段

有向线段的概念建构于向量的方向与长度，差别在于多定义了始点与终点。在文字描述时，如果已知某有向线段的起点和终点分别是 A 和 B，则此线段的长度可以记为 $|\vec{AB}|$ ，即

$$|\vec{AB}| = |\overline{AB}|$$

向量的记法

向量由方向和长度两个因素组成，可以记为 $\vec{a}$ 。

数量积

数量积也叫点积，它是向量与向量的乘积，其结果为一个标量（非向量）。几何上，数量积可以定义如下：设 $\vec{A}$ 、 $\vec{B}$ 为两个任意向量，它们的夹角为 θ ，则它们的数量积为

$$\vec{A} \cdot \vec{B} = |\vec{A}| |\vec{B}| \cos \theta$$

即 $\vec{A}$ 向量在 $\vec{B}$ 向量方向上的投影长度（同方向为正，反方向为负号）与向量 $\vec{B}$ 长度的乘积。数量积被广泛应用于物理中，如做功就是用力的向量乘以位移的向量，即 $W = \vec{F} \cdot \vec{s}$ 。

向量积

向量积也叫叉积、外积，它也是向量与向量的乘积，它的结果是个向量。它的几何意义是所得的向量与被乘向量所在平面垂直，方向由右手定则规定，大小是两个被乘向量张成的平行四边形的面积。所以向量积不满足交换律。举例来说：

$$(1, 0, 0) \times (0, 1, 0) = (0, 0, 1) \quad (0, 1, 0) \times (1, 0, 0) = (0, 0, -1)$$

设有向量

$$\vec{A} = (A_x \vec{i}, A_y \vec{j}, A_z \vec{k}) \quad \vec{B} = (B_x \vec{i}, B_y \vec{j}, B_z \vec{k})$$

则其向量积的矩阵表达式可表示为

$$\vec{A} \times \vec{B} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix}$$

线性相关性

对于 m 个向量 $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ ，如果存在一组不全为零的 m 个数 $a_2, a_1, \dots, a_m$ ，使得 $\sum_{i=1}^m a_i \vec{v}_i = \vec{0}$ ，那么称 m 个向量 $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ 线性相关。如果这样不全为零的个数不存在，即上述向量等式仅当 $a_1 = a_2 = \dots = a_m = 0$ 时才能成立，则称向量 $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ 线性无关。

例子解析

某人家门口是一条由南北向的道路。他散步时先向南行走 100 米，那么他位置的移动就可以用一个大小为 100 米、方向为南的向量来表示。之后他再向北走 300 米，这一次的移动可以用一个大小为 300 米、方向为北的向量来表示。散步的人总共相对于他家的位移则可以用大小为 200 米、方向为北的向量来表示。从几何学上看，这些向量都在同一条一维的直线上，只有两个互相平行的方向。

3.2.2 向量表示

代数表示

一般计算机上采用加粗小写英文字母如 **a**、**b**、**c** 等表示向量；手写时在 **a**、**b**、**c** 等字母上加一箭头 ($\rightarrow$) 表示，如 $\vec{a}$ 、 $\vec{b}$ 、 $\vec{c}$ 。

几何表示

向量可以用有向线段来表示。有向线段的长度表示向量的大小，向量的大小也就是向量的长度。长度为 0 的向量叫作零向量，记作长度等于 1 个单位的向量叫作单位向量。箭头所指的方向表示向量的方向，如图3-1所示。

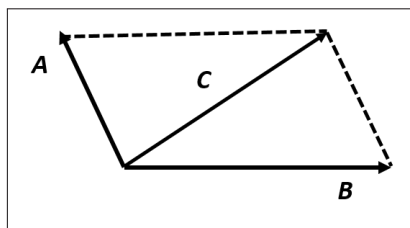


图 3-1 向量表示

3.2.3 向量定理

共线定理

若 $b \neq 0$, 则 $a \parallel b$ 的充要条件是存在唯一实数 λ , 使 $\vec{a} = \lambda \vec{b}$ 。若设 $a = (x_1, y_1)$, $b = (x_2, y_2)$, 则有 $x_1 y_2 = x_2 y_1$, 与平行概念相同。 $\vec{0}$ 平行于任何向量。

垂直定理

$a \perp b$ 的充要条件是 $\mathbf{a} \cdot \mathbf{b} = 0$ ，即 $x_1x_2 + y_1y_2 = 0$ 。

3.2.4 向量运算

加法

向量的加法满足平行四边形法则和三角形法则。两个向量 $\vec{a}$ 和 $\vec{b}$ 相加，得到的是另一个向量。这个向量可以表示为 $\vec{a}$ 和 $\vec{b}$ 的起点重合后，以它们为邻边构成的平行四边形的一条对角线（以共同的起点为起点的那一条，如图3-2所示），或者表示为将 $\vec{a}$ 的终点和 $\vec{b}$ 的起点重合后，从 $\vec{a}$ 的起点指向 $\vec{b}$ 的终点的向量。

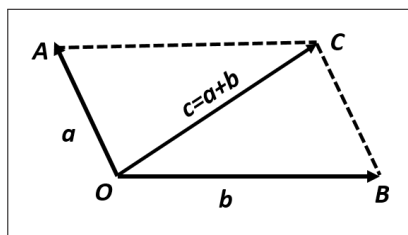


图 3-2 向量的加法

例子：

$$\vec{a} = (x_1, y_1), \vec{b} = (x_2, y_2)$$

则

$$\vec{a} + \vec{b} = (x_1 + x_2, y_1 + y_2)$$

向量加法的运算律：

$$\vec{a} + \vec{0} = \vec{0} + \vec{a} = \vec{a}$$

交换律：

$$\vec{a} + \vec{b} = \vec{b} + \vec{a}$$

结合律：

$$(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$$

减法

两个向量 $\vec{a}$ 和 $\vec{b}$ 的相减可以看作向量加上一个与大小相等、方向相反的向量。换言之， $\vec{a}$ 和 $\vec{b}$ 的相减得到的向量可以表示为 $\vec{a}$ 和 $\vec{b}$ 的起点重合后，从 $\vec{b}$ 的终点指向 $\vec{a}$ 的终点的向量，如图3-3所示。

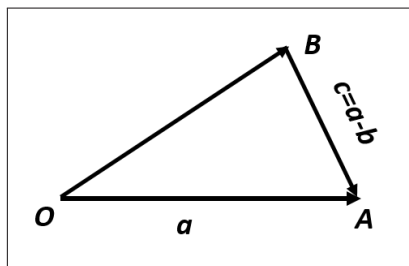


图 3-3 向量的减法

例子：

$$\vec{a} = (x_1, y_1), \vec{b} = (x_2, y_2)$$

则：

$$\vec{a} - \vec{b} = (x_1 - x_2, y_1 - y_2)$$

加减变换律：

$$\vec{a} + (-\vec{b}) = \vec{a} - \vec{b}$$

数乘

一个标量 k 和一个向量 $\vec{a}$ 之间可以做乘法，得出的结果是另一个与 $\vec{a}$ 方向相同或相反，大小为 $\vec{a}$ 的大小的 $|k|$ 倍的向量，可以记为 $k\vec{a}$ 。

- ⊙ 当 $k > 0$ 时， $k\vec{a}$ 的方向与 $\vec{a}$ 的方向相同。
- ⊙ 当 $k < 0$ 时， $k\vec{a}$ 的方向与 $\vec{a}$ 的方向相反。
- ⊙ 当 $k = 0$ 时， $k\vec{a} = \vec{0}$ ，方向任意。当 $\vec{a} = \vec{0}$ 时，对于任意实数 k ，都有 $k\vec{a} = \vec{0}$ 。

-1 乘以任意向量会得到它的反向量，0 乘以任何向量都会得到零向量 $\vec{0}$ 。

数与向量满足运算律

结合律:

$$(k\vec{a}) \times \vec{b} = k(\vec{a} \times \vec{b}) = (\vec{a} \times k\vec{b})$$

向量对于数的分配律 (第一分配律):

$$(k+m)\vec{a} = k\vec{a} + m\vec{a}$$

数对于向量的分配律 (第二分配律):

$$k(\vec{a} + \vec{b}) = k\vec{a} + k\vec{b}$$

数乘向量的消去律:

- (1) 如果实数 $k \neq 0$ 且 $k\vec{a} = k\vec{b}$, 那么 $\vec{a} = \vec{b}$ 。
- (2) 如果 $\vec{a} \neq \vec{b}$ 且 $k\vec{a} = m\vec{a}$, 那么 $k = m$ 。

向量积的运算律

交换律:

$$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$$

结合律:

$$(k\vec{a}) \times \vec{b} = k(\vec{a} \times \vec{b})$$

分配律:

$$(\vec{a} + \vec{b}) \times \vec{c} = \vec{a} \times \vec{c} + \vec{b} \times \vec{c}$$

向量积

定义: 两个向量 $\vec{a}$ 和 $\vec{b}$ 的向量积 (外积、叉积) 是一个向量。记作 $\vec{a} \wedge \vec{b}$ 。

- ◎ 若 $\vec{a}$ 、 $\vec{b}$ 不共线, 则 $\vec{a} \wedge \vec{b}$ 的模是: $|\vec{a} \wedge \vec{b}| = |\vec{a}| \times |\vec{b}| \times \sin \angle \vec{a}, \vec{b}$; $\vec{a} \wedge \vec{b}$ 的方向是: 垂直于 $\vec{a}$ 和 $\vec{b}$, 且 $\vec{a}$ 、 $\vec{b}$ 和 $\vec{a} \wedge \vec{b}$ 按这个次序构成右手系。

- ⊙ 若 $\vec{a}$ 、 $\vec{b}$ 垂直，则 $|\vec{a} \wedge \vec{b}| = |\vec{a}| \times |\vec{b}|$ （此处与数量积不同，请注意），若 $|\vec{a} \wedge \vec{b}| = 0$ ，则 $\vec{a}$ 、 $\vec{b}$ 平行。

向量积即两个不共线非零向量所在平面的一组法向量。

运算法则：运用三阶行列式

设 $\vec{a}$ 、 $\vec{b}$ 、 $\vec{c}$ 分别为沿 x 、 y 、 z 轴的单位向量 $\mathbf{A} = (x_1, y_1, z_1)$ ， $\mathbf{B} = (x_2, y_2, z_2)$ ，则

$$\mathbf{A} \times \mathbf{B} = \begin{vmatrix} \vec{a} & \vec{b} & \vec{c} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

向量积性质

向量积 $|\vec{a} \wedge \vec{b}|$ 是以 $\vec{a}$ 和 $\vec{b}$ 为边的平行四边形面积。

- ⊙ $|\vec{a} \wedge \vec{b}| = 0$
- ⊙ $\vec{a} \parallel \vec{b} = \vec{a} \wedge \vec{a} = 0$

向量积运算律

- ⊙ $|\vec{a} \wedge \vec{b}| = (-\vec{b}) \wedge \vec{a}$
- ⊙ $(k\vec{a}) \wedge \vec{b} = k(\vec{a} \wedge \vec{b}) = \vec{a}$
- ⊙ $\vec{a} \wedge (\vec{b} + \vec{c}) = \vec{a} \wedge \vec{b} + \vec{a} \wedge \vec{c}$
- ⊙ $(\vec{a} + \vec{b}) \wedge \vec{c} = \vec{a} \wedge \vec{c} + \vec{b} \wedge \vec{c}$

$a \times (b + c) = a \times b + a \times c$ ， $(a + b) \times c = a \times c + b \times c$ 。这两个分配律分别称为左分配律和右分配律。在演算中应注意不能交换“ $\times$ ”号两侧向量的次序。注：向量没有除法，“向量 $\mathbf{AB}$ /向量 $\mathbf{CD}$ ”是没有意义的。

向量的模长（范数）

向量的大小也叫作范数或者模长，在有限维空间中，已知向量的坐标，就可以知道它的模长。设向量 $\vec{v} = (v_1, v_2, \dots, v_n)$ ，范数记作 $\|\vec{v}\|$ ，模长记作 $|\vec{v}|$ 。

计算表达式由弗罗贝尼乌斯范数（一种同时适用于向量和矩阵的范数计算方法）给出：

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

或

$$|\vec{v}| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

3.3 矩阵

3.3.1 矩阵定义

矩阵定义

数学上，一个 $m \times n$ 的矩阵是一个由 m 行 n 列元素排列成的矩形阵列。矩阵里的元素可以是数字、符号或数学式。以下是一个由 6 个数字元素构成的 2 行 3 列的矩阵：

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -6 \end{bmatrix}$$

大小相同（行数列数都相同）的矩阵之间可以相互加减，具体是对每个位置上的元素做加减法。矩阵的乘法则较为复杂。两个矩阵可以相乘——当且仅当第一个矩阵的列数等于第二个矩阵的行数。矩阵的乘法满足结合律和分配律，但不满足交换律。

矩阵的理解

将一些元素排列成若干行，每行放上相同数量的元素，就是一个矩阵。这里说的元素可以是数字，如以下的矩阵：

$$A = \begin{bmatrix} 9 & 13 & 5 \\ 1 & 11 & 7 \\ 3 & 9 & 2 \\ 6 & 0 & 7 \end{bmatrix}$$

排列成的形状是矩形，所以称为矩阵。矩阵一般用大写拉丁字母表示，一般用方括号或圆括号括起。以上的矩阵 A 是一个 4 行 3 列的矩阵。

行数是 1 或列数是 1 的矩阵又可分别称为行向量和列向量。矩阵的任一行（列）都是一个行（列）向量，例如矩阵 A 的第一行 [9 13 5] 就是一个行向量。行（列）向量可以看作一个向量，因此可以称矩阵的两行（列）相等，或者某一行等于某一列，表示其对应的向量相等。

增广矩阵

矩阵的一个重要用途是解线性方程组。线性方程组中未知量的系数可以排成一个矩阵，加上常数项，称为增广矩阵。

3.3.2 矩阵表示

一个矩阵 A 从左上角数起的第 i 行第 j 列上的元素称为第 ij 项，通常记为 A_{ij} 、 A_{ij} 、 a_{ij} 或 $A_{[i,j]}$ ，在上述例子中 $A_{[4,3]} = 7$ 。

3.3.3 矩阵运算

矩阵加法

定义： $m \times n$ 矩阵 A 和 B 的和， $A + B$ 为一个 $m \times n$ 矩阵，其中每个元素是 A 和 B 相应元素的和。

$$(A + B)_{ij} = A_{i,j} + B_{i,j}$$

其中， $1 \leq i \leq m, 1 \leq j \leq n$ 。

例子：

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{bmatrix}$$

运算律（ A, B, C 都是同型矩阵）：

$$\begin{aligned} \mathbf{A} + \mathbf{B} &= \mathbf{B} + \mathbf{A} \\ (\mathbf{A} + \mathbf{B}) + \mathbf{C} &= \mathbf{A} + (\mathbf{B} + \mathbf{C}) \end{aligned}$$

需要注意的是，只有同型矩阵之间才可以进行加法。

矩阵减法

定义： $m \times n$ 矩阵 $\mathbf{A}$ 和 $\mathbf{B}$ 的差， $\mathbf{A} - \mathbf{B}$ 为一个 $m \times n$ 矩阵，其中每个元素是 $\mathbf{A}$ 和 $\mathbf{B}$ 相应元素的差。

$$(\mathbf{A} - \mathbf{B})_{ij} = \mathbf{A}_{i,j} - \mathbf{B}_{i,j}$$

其中， $1 \leq i \leq m, 1 \leq j \leq n$ 。

例子：

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1-0 & 3-0 & 1-5 \\ 1-7 & 0-5 & 0-0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -4 \\ -6 & -5 & 0 \end{bmatrix}$$

矩阵数乘

定义：标量 c 与矩阵 $\mathbf{A}$ 的数乘。 $c\mathbf{A}$ 的每个元素是 $\mathbf{A}$ 的相应元素与 c 的乘积。

$$(c\mathbf{A})_{ij} = c\mathbf{A}_{i,j}$$

例子：

$$2 \times \begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 \times 1 & 2 \times 3 & 2 \times 1 \\ 2 \times 1 & 2 \times 0 & 2 \times 0 \end{bmatrix} = \begin{bmatrix} 2 & 6 & 2 \\ 2 & 0 & 0 \end{bmatrix}$$

运算律：

- ⊙ $(\lambda\mu)\mathbf{A} = \lambda(\mu\mathbf{A})$
- ⊙ $(\lambda + \mu)\mathbf{A} = \lambda\mathbf{A} + \mu\mathbf{A}$
- ⊙ $\lambda(\mathbf{A} + \mathbf{B}) = \lambda\mathbf{A} + \lambda\mathbf{B}$

矩阵转置

定义： $m \times n$ 矩阵 A 的转置是一个 $n \times m$ 的矩阵，记为 A^T （其中第 i 个行向量是原矩阵 A 的第 i 个列向量；或者说，转置矩阵 A^T 的第 i 行第 j 列的元素是原矩阵 A 第 j 行第 i 列的元素）。

$$(A^T)_{i,j} = A_{j,i}$$

例子：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

运算律：

$$\odot (A^T)^T = A$$

$$\odot (\lambda A)^T = \lambda A^T$$

$$\odot (A + B)^T = B^T + A^T$$

共轭矩阵

定义：矩阵的共轭定义为 $(A)_{i,j} = \overline{A_{ij}}$ 。

例子：

一个 2×2 复数矩阵的共轭如下。

$$A = \begin{bmatrix} 3+i & 5 \\ 2-2i & i \end{bmatrix}$$

则 A 的共轭矩阵是：

$$\overline{A} = \begin{bmatrix} 3-i & 5 \\ 2+2i & -i \end{bmatrix}$$

3.3.4 线性方程组

矩阵乘法的一个基本应用是在线性方程组上。线性方程组是方程组的一种，它符合以下的形式：

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \cdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m \end{cases}$$

其中的 $a_{1,1}$ 、 $a_{1,2}$ 和 b_1 、 b_2 等是已知的常数，而 x_1 、 x_2 等则是要求的未知数。运用矩阵的方式，可以将线性方程组写成一个向量方程：

$$Ax = b$$

其中， A 是由方程组里未知量的系数排成的 $m \times n$ 矩阵， x 是含有 n 个元素的行向量， b 是含有 m 个元素的行向量。

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} \cdots & a_{2,n} \\ \cdots & \cdots & \cdots \\ a_{m,1} & a_{m,2} \cdots & a_{m,n} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}$$

这个写法下，将原来的多个方程转化成一个向量方程，在已知矩阵 A 和向量 b 的情况下，求未知向量 x 。

3.3.5 行列式

方块矩阵

方块矩阵是行数与列数相同的矩阵称为方块矩阵，简称方阵，表示如下：

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

逆矩阵

方阵 $\mathbf{A}$ 称为可逆或非奇异的，如果存在另一个方阵 $\mathbf{B}$ ，使得

$$\mathbf{AB} = \mathbf{I}_n$$

成立。这时可以证明也有 $\mathbf{BA} = \mathbf{I}_n$ 成立，可将矩阵 $\mathbf{B}$ 称为 $\mathbf{A}$ 的逆矩阵。一个矩阵 $\mathbf{A}$ 的逆矩阵如果存在，则是唯一的，通常记作 $\mathbf{A}^{-1}$ 。

矩阵的迹

矩阵 $\mathbf{A}$ 的元素 A_i ， i 称为其主对角线上的元素。方块矩阵 $\mathbf{A}$ 的所有主对角线元素之和称为它的迹，写作 $\text{tr}(\mathbf{A})$ 。

例子：

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

$\mathbf{A}$ 的迹是 $\text{tr}(\mathbf{A}) = a_{1,1} + a_{2,2}$

矩阵的迹具备以下特征：

尽管矩阵的乘法不满足交换律，方阵相乘时交换顺序会导致乘积变化，但它们的迹不会变，即 $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$ 。

矩阵转置的迹等于其自身的迹，即 $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^T)$ 。

对角矩阵

如果一个方阵只有主对角线上的元素不是 0，其他都是 0，那么称其为对角矩阵，如下所示。

$$\begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix}$$

上三角矩阵

如果主对角线下方的元素都是 0，那么称其为上三角矩阵，如下所示。

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

下三角矩阵

如果主对角线上方的元素都是 0，那么称其为下三角矩阵，如下所示。

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

行列式

◎ 1×1 方阵的行列式为该元素本身。

$$\mathbf{A} = (a_{11})$$

$$|\mathbf{A}| = a_{11}$$

◎ 2×2 方阵的行列式用主对角线元素乘积减去次对角线元素的乘积。

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

◎ 3×3 阶方阵：

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

三阶矩阵发现 a_{12} 的对角线少一部分（也就是 a_{23} 的右下部分缺失）。一种方法是复制三个完全一样的矩阵做补充。

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{11} & a_{12} & a_{13} & a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} & a_{23} & a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} & a_{33} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$|A| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31}$$

另一种方式就是利用代数余子式来计算。

在一个 n 阶行列式 A 中, 把 (i, j) 元素 a_{ij} 所在的第 i 行和第 j 列划去后, 留下的 $n-1$ 阶方阵的行列式叫作元素 a_{ij} 的余子式, 记作 M_{ij} 。

代数余子式: $A_{IH} = (-1)^{i+j} M_{ij}$ 。

注意: 代数余子式是个数值!

下面方框里计算的值便是 $a_{11}a_{12}$ 的代数余子式 $M_{11}M_{12}$ 。

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

◎ n 阶的行列式等于它的任意一行 (或列) 的各元素与其对应的代数余子式乘积之和。

对于任意一列:

$$|A| = \sum_{i=1}^n a_{ij} (-1)^{i+j} M_{ij}, \quad \forall j \leq n$$

对于任意一行:

$$|A| = \sum_{j=1}^n a_{ij} (-1)^{i+j} M_{ij}, \quad \forall i \leq n$$

所以上面三阶方阵的行列式 A 就是:

$$|A| = a_{11}(a_{22}a_{33} - a_{23}a_{32}) + a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{23}a_{31})$$

3.3.6 特征值和特征向量

$n \times n$ 的方块矩阵 $\mathbf{A}$ 的一个特征值和对应特征向量是满足 $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ 的标量 λ ，以及非零向量 $\mathbf{v}$ 。特征值和特征向量的概念对研究线性变换很有帮助。一个线性变换可以通过它对应的矩阵在向量上的作用来进行可视化。一般来说，一个向量在经过映射之后可以变为任何可能的向量，而特征向量具有更好的性质。假设在给定的基底下，一个线性变换对应某个矩阵 $\mathbf{A}$ ，如果一个向量 $\mathbf{x}$ 可以写成矩阵的几个特征向量的线性组合：

$$\mathbf{x} = c_1\mathbf{x}_{\lambda_1} + c_2\mathbf{x}_{\lambda_2} + \dots + c_k\mathbf{x}_{\lambda_k}$$

其中的 $\mathbf{x}_{\lambda_i}$ 表示此向量对应的特征值是 λ_i ，那么向量 $\mathbf{x}$ 经过线性变换后会变成：

$$\mathbf{A}\mathbf{x} = c_1\lambda_1\mathbf{x}_1 + c_2\lambda_2\mathbf{x}_2 + \dots + c_k\lambda_k\mathbf{x}_k$$

可以清楚地知道变换后向量的结构。

另一个等价的特征值定义是：标量为特征值，如果矩阵 $\mathbf{A} - \lambda \mathbf{I}_n$ 是不可逆矩阵。根据不可逆矩阵的性质，这个定义也可以用行列式方程描述： λ 为特征值，如果

$$\det(\lambda \mathbf{I}_n - \mathbf{A}) = 0$$

这个定义中的行列式可以展开成一个关于的 n 阶多项式，叫作矩阵 $\mathbf{A}$ 的特征多项式，记为 $P_{\mathbf{A}}$ 。特征多项式是一个首一多项式（最高次项系数是 1 的多项式）。它的根就是矩阵 $\mathbf{A}$ 特征值。哈密尔顿—凯莱定理说明，如果用矩阵 $\mathbf{A}$ 本身代替多项式中的不定元，那么多项式的值是零矩阵：

$$P_{\mathbf{A}}(\mathbf{A}) = \mathbf{0}$$

3.4 距离计算

3.4.1 余弦距离

形式化描述

余弦夹角也可以叫余弦相似度。几何中夹角余弦可用来衡量两个向量方向的差异，机器学习中借用这一概念来衡量样本向量之间的差异。

余弦取值范围为 $[-1, 1]$ 。求得两个向量的夹角，并得出夹角对应的余弦值，此余弦值就可以用来表征这两个向量的相似性。夹角越小，趋近于 0 度，余弦值越接近于 1，它们的方向更加吻合，则越相似。当两个向量的方向完全相反时，夹角余弦取最小值 -1 。当余弦值为 0 时，两向量正交，夹角为 90 度。因此可以看出，余弦相似度与向量的幅值无关，只与向量的方向相关。

公式化描述

$$\cos \theta = \frac{\vec{a} \times \vec{b}}{|\vec{a}| \times |\vec{b}|}$$

在二维空间中向量 $A(x_1, y_1)$ 与向量 $B(x_2, y_2)$ 的夹角余弦公式为

$$\cos \theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \sqrt{x_2^2 + y_2^2}}$$

Python 实现

```
import numpy as np
vec1 = [1,2,3,4]
vec2 = [5,6,7,8]

#方法一：根据公式求解
dist1=np.dot(vec1,vec2)/(np.linalg.norm(vec1)*np.linalg.norm(vec2))
print("余弦距离测试结果是：\t"+str(dist1))
```



```
#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=1-pdist(Vec,'cosine')
print("余弦距离测试结果是：\t"+str(dist2))
```

3.4.2 欧氏距离

形式化描述

在数学中，欧几里得距离或欧几里得度量是欧几里得空间中两点间“普通”（即直线）距离。使用这个距离，欧氏空间成为度量空间。相关联的范数称为欧几里得范数。较早的文献称之为毕达哥拉斯度量。

公式化描述

- ◎ 维平面上两点 $a(x_1, y_1)$ 与 $b(x_2, y_2)$ 间的欧氏距离

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- ◎ 三维空间两点 $a(x_1, y_1, z_1)$ 与 $b(x_2, y_2, z_2)$ 间的欧氏距离

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

- ◎ 两个 n 维向量 $\mathbf{a}$ 与 $\mathbf{b}$ 之间的欧氏距离

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + \dots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

表示成向量运算的形式

$$d_{12} = \sqrt{(\mathbf{a} - \mathbf{b})(\mathbf{a} - \mathbf{b})^T}$$

Python 实现

```
import numpy as np
vec1 =np.mat([1,2,3,4])
vec2 =np.mat([5,6,7,8])

#方法一：根据公式求解
dist1=np.sqrt(np.sum(np.square(vec1-vec2)))
print("欧氏距离测试结果是：\t"+str(dist1))

#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec)
print("欧氏距离测试结果是：\t"+str(dist2))

# 方法三：根据公式求解
from numpy import *
dist3 = sqrt((vec1-vec2)*(vec1-vec2).T)
print("欧氏距离测试结果是：\t"+str(dist3))
```

3.4.3 曼哈顿距离

形式化描述

计程车几何或曼哈顿距离或方格线距离是由 19 世纪的赫尔曼·闵可夫斯基所创的辞汇，为欧几里得几何度量空间的几何学之用语，用来标明两个点在标准坐标系上的绝对轴距之总和。简言之，曼哈顿从一个十字路口开车到另外一个十字路口，驾驶距离不是两点之间的直线距离。实际驾驶距离就是这个“曼哈顿距离”。而这也是曼哈顿距离名称的来源，曼哈顿距离也称为城市街区距离（City Block distance）。

公式化描述

◎ 二维平面两点 $a(x_1, y_1)$ 与 $b(x_2, y_2)$ 之间的曼哈顿距离

$$d_{12} = |(x_1 - x_2)| + |(y_1 - y_2)|$$

◎ 两个 n 维向量 $a(x_1, x_2, \dots, x_n)$ 与 $b(y_1, y_2, \dots, y_n)$ 之间的曼哈顿距离

$$d_{12} = \sum_{i=1}^n |x_i - y_i|$$

Python 实现

```
import numpy as np
vec1 = np.mat([1,2,3,4])
vec2 = np.mat([5,6,7,8])

#方法一：根据公式求解
dist1=np.sum(np.abs(vec1-vec2))
print("曼哈顿距离测试结果是：\t"+str(dist1))

#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec,'cityblock')
print("曼哈顿距离测试结果是：\t"+str(dist2))

from numpy import *
dist3 = sum(abs(vec1-vec2))
print("曼哈顿距离测试结果是：\t"+str(dist3))
```

3.4.4 明可夫斯基距离

形式化描述

明氏距离又叫作明可夫斯基距离，是欧氏空间中的一种测度，被看作欧氏距离和曼哈顿距离的一种推广。

公式化描述

◎ 两个 n 维向量 $a(x_1, x_2, \dots, x_n)$ 与 $b(y_1, y_2, \dots, y_n)$ 之间的曼哈顿距离

$$d_{12} = p \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$

也可以写成

$$d_{12} = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

其中 p 是一个变参数。

- (1) 当 $p = 1$ 时，就是曼哈顿距离。
- (2) 当 $p = 2$ 时，就是欧氏距离。
- (3) 当 $p \rightarrow \infty$ 时，就是切比雪夫距离。

Python 实现

```
import numpy as np
vec1 = np.mat([1,2,3,4])
vec2 = np.mat([5,6,7,8])

#方法一：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec,'minkowski',p=1)
print("当p=1时就是曼哈顿距离,结果是: \t"+str(dist2))

#方法二：根据公式求解,p=2
dist1=np.sqrt(np.sum(np.square(vec1-vec2)))
print("当p=2时就是欧式距离,结果是: \t"+str(dist1))

from numpy import *
#方法三：根据公式求解,p=1
dist3 = sum(abs(vec1-vec2))
print("当p=1时就是曼哈顿距离,结果是: \t"+str(dist3))

#方法四：根据公式求解,p=2
```

```
dist4 = sqrt((vec1-vec2)*(vec1-vec2).T)
print("当p=2时就是欧式距离,测试结果是: \t"+str(dist4))
```

3.4.5 切比雪夫距离

形式化描述

数学上, 切比雪夫距离 (Chebyshev distance) 是向量空间中的一种度量, 二个点之间的距离定义为其各坐标数值差的最大值。以 (x_1, y_1) 和 (x_2, y_2) 二点为例, 其切比雪夫距离为 $\max(|x_2 - x_1|, |y_2 - y_1|)$ 。切比雪夫距离得名自俄罗斯数学家切比雪夫。

公式化描述

◎ 二维平面两点 (x_1, y_1) 与 $b(x_2, y_2)$ 之间的切比雪夫距离

$$d_{12} = \max(|x_1 - x_2|, |y_1 - y_2|)$$

◎ 两个 n 维向量 $a(x_1, x_2, \dots, x_n)$ 与 $b(y_1, y_2, \dots, y_n)$ 之间的切比雪夫距离

$$d_{12} = \max_i (|x_i - y_i|)$$

还可以表示为

$$d_{12} = \lim_{k \rightarrow \infty} \left(\sum_{i=1}^n |x_i - y_i|^k \right)^{\frac{1}{k}}$$

Python 实现

```
import numpy as np
vec1 = np.mat([1,2,3,4])
vec2 = np.mat([5,6,7,8])

#方法一: 根据公式求解
dist1=np.max(np.abs(vec1-vec2))
print("切比雪夫距离测试结果是: \t"+str(dist1))
```

```
#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec,'chebyshev')
print("切比雪夫距离测试结果是：\t"+str(dist2))
```

3.4.6 杰卡德距离

形式化描述

杰卡德相似系数：两个集合 A 和 B 的交集元素在 A 、 B 的并集中所占的比例，称为两个集合的杰卡德相似系数，用符号 $J(A,B)$ 表示。

杰卡德距离：与杰卡德相似系数相反的概念，即杰卡德距离用两个集合中不同元素占所有元素的比例来衡量两个集合的区分度。这是杰卡德距离（Jaccard distance）。

杰卡德相似系数与杰卡德距离的应用：可将杰卡德相似系数用在衡量样本的相似度上。样本 A 与样本 B 是两个 n 维向量，而且所有维度的取值都是 0 或 1。例如： $A(0111)$ 和 $B(1011)$ 。我们将样本看作一个集合，1 表示集合包含该元素，0 表示集合不包含该元素。

公式化描述

杰卡德相似系数用公式表示：

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

杰卡德距离可用如下公式表示：

$$J_d(A,B) = 1 - J(A,B) = \frac{A \cup B - A \cap B}{A \cup B}$$

Python 实现

```
import numpy as np
v1=np.random.random(10)>0.5
```

```

v2=np.random.random(10)>0.5

vec1=np.asarray(v1,np.int32)
vec2=np.asarray(v2,np.int32)

#方法一：根据公式求解
up=np.double(np.bitwise_and((vec1 != vec2),np.bitwise_or(vec1 != 0, vec2
    != 0)).sum())
down=np.double(np.bitwise_or(vec1 != 0, vec2 != 0).sum())
dist1=(up/down)
print("杰卡德距离测试结果是：\t"+str(dist1))

#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec,'jaccard')
print("杰卡德距离测试结果是：\t"+str(dist2))

```

3.4.7 汉明距离

形式化描述

在信息论中，两个等长字符串之间的汉明距离是两个字符串对应位置的不同字符的个数。换句话说，它就是将一个字符串变换成另外一个字符串所需要替换的字符个数。

范例公式化描述

- ◎ 1011101 与 1001001 之间的汉明距离是 2。
- ◎ 2143896 与 2233796 之间的汉明距离是 3。
- ◎ “toned” 与 “roses” 之间的汉明距离是 3。

Python 实现

```

v1=np.random.random(10)>0.5
v2=np.random.random(10)>0.5

vec1=np.asarray(v1,np.int32)
vec2=np.asarray(v2,np.int32)

#方法一：根据公式求解
dist1=np.mean(vec1!=vec2)
print("汉明距离测试结果是：\t"+str(dist1))

#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
Vec=np.vstack([vec1,vec2])
dist2=pdist(Vec,'hamming')
print("汉明距离测试结果是：\t"+str(dist2))

```

3.4.8 标准化欧式距离

形式化描述

标准化欧氏距离是针对简单欧氏距离的缺点而做的一种改进方案。标准欧氏距离的思路：既然数据各维分量的分布不一样，那先将各个分量都“标准化”到均值、方差相等。均值和方差标准化到多少呢？这里先复习一些统计学知识，假设样本集 X 的均值（mean）为 m ，标准差（standard deviation）为 s ，那么 X 的“标准化变量”表示为

$$X^* = \frac{X - m}{s}$$

标准化后的值 = （标准化前的值 - 分量的均值）/分量的标准差

公式化描述

两个 n 维向量 $a(x_1, x_2, \dots, x_n)$ 与 $b(y_1, y_2, \dots, y_n)$ 之间的标准化欧氏距离的公式:

$$d_{12} = \sqrt{\sum_{k=1}^n \left(\frac{x_i - y_i}{S_k} \right)^2}$$

Python 实现

```
import numpy as np
vec1 = np.array([1,2,3,4])
vec2 = np.array([5,6,7,8])

Vec=np.vstack([vec1,vec2])
#方法一：根据公式求解
sk=np.var(Vec,axis=0,ddof=1)
dist1=np.sqrt(((vec1 - vec2) ** 2 /sk).sum())
print("标准化欧氏距离测试结果是: \t"+str(dist1))

#方法二：根据scipy库求解
from scipy.spatial.distance import pdist
dist2=pdist(Vec,'seuclidean')
print("标准化欧氏距离测试结果是: \t"+str(dist2))
```

3.4.9 皮尔逊相关系数

形式化描述

在统计学中, 皮尔逊积矩相关系数 (Pearson product-moment correlation coefficient, 常用 r 表示) 用于度量两个变量 X 和 Y 之间的相关程度 (线性相关), 其值介于 -1 与 1 之间。在自然科学领域中, 该系数广泛用于度量两个变量之间的相关程度。

公式化描述

计算相关系数:

$$\text{Corr}(x, y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}} = \frac{\langle x - \bar{x}, y - \bar{y} \rangle}{\|x - \bar{x}\| \|y - \bar{y}\|}$$

Python 实现

```
import numpy as np
vec1 = np.array([1,2,3,4])
vec2 = np.array([5,6,7,8])

#方法一：根据公式求解
vec1_=vec1-np.mean(vec1)
vec2_=vec2-np.mean(vec2)
dist1=np.dot(vec1_,vec2_)/(np.linalg.norm(vec1_)*np.linalg.norm(vec2_))
print("皮尔逊相关系数测试结果是：\t"+str(dist1))

#方法二：根据numpy库求解
Vec=np.vstack([vec1,vec2])
dist2=np.corrcoef(Vec)[0][1]
print("皮尔逊相关系数测试结果是：\t"+str(dist2))
```

源码请访问<https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>。

第 4 章

概率论

导读：机器学习与深度学习是多学科交叉的科学技术，其中数学尤为重要，是很多形式化模型向数学建模的必经过程。继上一章线性代数核心知识的介绍之后，本章着重介绍概率论的相关知识。由于基于规则方法向基于统计方法的转型，概率就显得尤为重要，诸如一些随机事件、独立假设、条件概率、完全概率，等等。然后对贝叶斯模型进行案例式介绍，旨在让读者加深理解。最后重点介绍信息论相关概念和信息度量的常用方法。

4.1 概率论介绍

概率论（Probability Theory）是集中研究概率和随机现象的数学分支，是研究随机性或不确定性等现象的学科。概率论主要研究对象为随机事件、随机变量和随机过程。随机事件是不可能准确预测其结果的，然而对于一系列的独立随机事件——例如，掷骰子、扔硬币、抽扑克牌及轮盘等，会呈现出一定的、可以被用于研究及预测的规律，两个用来描述这些规律的最具代表性的数学结论分别是大数定律和中心极限定理。

概率的生活案例之六合彩

买 5, 17, 19, 24, 33, 49 的中奖概率高还是买 1,2,3,4,5,6 的中奖概率高？

古典概率论说：一样。但实际上机械或彩球制造上都有些微小的差异，所以每组概率不一定完全相同，但必须累积多期开奖结果后才看得出来。

概率的生活案例之生日悖论

在一个足球场上有 23 个人（ 2×11 个运动员和 1 个主裁判员），不可思议的是，在这 23 人当中至少有两个人的生日是在同一天的概率要大于 50%。如果这 23 人都没有相同的生日，也不违反概率，只是概率小于 50%。

概率的生活案例之轮盘游戏

在游戏中玩家可能认为，在连续出现多次红色后，出现黑色的概率会越来越大。这种判断也是错误的，即出现黑色的概率每次是相等的，因为球本身并没有“记忆”，它不会意识到以前都发生了什么，其概率始终是 $18/37$ 。但轮盘的前后期开奖数字形成了时间序列（可能存在自回归模型）。

概率的生活案例之赢取名车

赢取电视节目里的名车：在参赛者面前有三扇关闭的门，其中只有一扇后面有名车，而其余门的后面是山羊。

游戏规则是，参赛者先选取一扇门，但在他打开之前，主持人在其余两扇门中打开了一扇有山羊的门，并询问参赛者是否改变主意选择另一扇门，以使赢得名车的概率变大。

正确的分析结果是，假如不管开始哪一扇门被选，主持人都打开其余两扇门中有山羊的那一扇并询问参赛者是否改变主意，则改变主意会使赢得汽车的概率增加一倍（“标准”的三门问题情况）。

假如主持人只在有名车那扇门被选中时劝诱参赛者打开其他门，改变主意则必输（资讯不对称）。

4.2 事件

4.2.1 随机试验

随机试验的定义

我们将对自然现象进行的一次观察或一次科学试验称为试验。

随机试验的例子

举例 1：硬币试验。

- ◎ E1：抛一枚硬币，观察正（H）反（T）面的情况。
- ◎ E2：将一枚硬币抛三次，观察正反面出现的情况。
- ◎ E3：将一枚硬币抛三次，观察出现正面的情况。
- ◎ E4：电话交换台一分钟内接到的呼唤次数。
- ◎ E5：在一批灯泡中任取一只，测试它的寿命。

举例 2：数学家去赌场。

新闻：数学家 3 年赌赢 156 亿人民币，数学家在赌场里有什么优势？

令 19 名数学家惊喜的是，虽然他们所掌握的那些高深数学知识在现实生活中似乎派不上多大用场，但竟然出人意料地在赌场上显现出了巨大的威力！据悉，19 名数学家参与的大多是赛马、赛狗及 21 点之类的赌博项目。而每次下注之前，他们会利用自己所精通的专业数学方法对各种中奖的概率进行推理演算，从而研究出某种“逢赌必赢”的秘籍！因为它的形态看起来合乎理想。在现实生活中，遇到测量之类的大量连续数据时，在“正常情况下”会期望看到这种形态。

4.2.2 随机事件和样本空间

基本事件或单位事件

定义：在一次随机试验中可能发生且不能再细分的结果被称为基本事件，或者称为单位事件，用 E 表示。

样本空间

定义：在随机试验中可能发生的所有单位事件的集合称为事件空间，用 S 来表示。

例如：在一次掷骰子的随机试验中，如果用获得的点数表示单位事件，那么一共可能出现 6 个单位事件，则事件空间可以表示为 $S = \{1, 2, 3, 4, 5, 6\}$ 。

上面的事件空间是由可数有限单位事件组成的，事实上还存在由可数无限及不可数单位事件组成的事件空间，比如在一次获得正面朝上就停止的随机掷硬币试验中，其事件空间由可数无限单位事件组成，表示为 $S = \{\text{正}, \text{反正}, \text{反反正}, \text{反反反正}, \text{反反反反正}, \dots\}$ ，

注意到在这个例子中，“反反反正”是单位事件。将两根筷子随意扔向桌面，其静止后所形成的交角假设为 α ，这个随机试验的事件空间的组成可以表示为 $S = \{\alpha | 0^\circ \leq \alpha \leq 180^\circ\}$ 。

随机事件

随机事件是事件空间 S 的子集，它由事件空间 S 中的单位元素构成，用大写字母 A 、 B 、 C ...表示。例如，在掷两个骰子的随机试验中，设随机事件 $A = \text{“获得的点数和大于 } 10\text{”}$ ，则可以由下面 3 个单位事件组成： $A = \{(5, 6), (6, 5), (6, 6)\}$ 。

必然事件和不可能事件

如果在随机试验中事件空间中的所有可能的单位事件都发生，则这个事件被称为必然事件；相应的，如果事件空间里不包含任何一个单位事件，则称为不可能事件。

4.2.3 事件的计算

因为事件在一定程度上是以集合的含义定义的，因此可以把集合计算方法直接应用于事件的计算。也就是说，在计算过程中，可以把事件当作集合来对待，如图 4-1 所示。

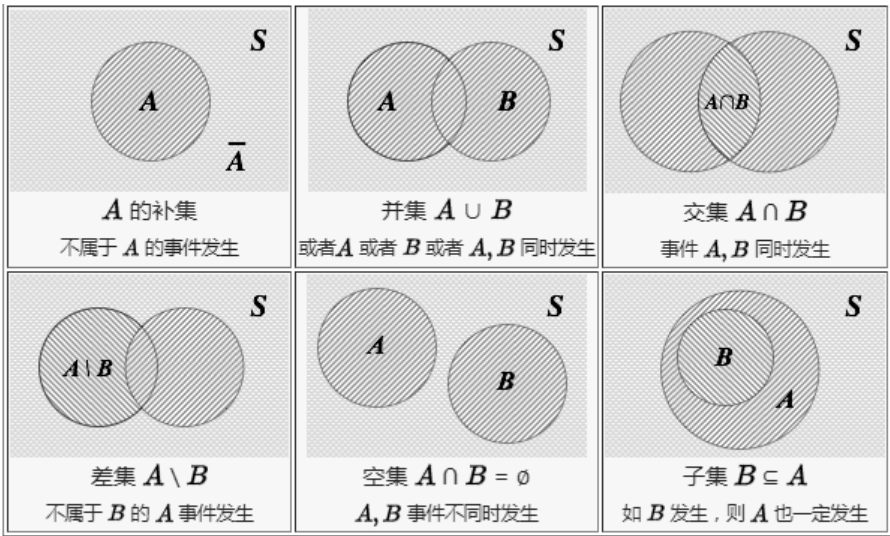


图 4-1 集合计算

在轮盘游戏中假设 A 代表事件“球落在红色区域”， B 代表事件“球落在黑色区域”，因为事件 A 和 B 没有共同的单位事件，因此可表示为 $A \cap B = \phi$ 。

注意到事件 A 和 B 并不是互补的关系，因为在整个事件空间 S 中还有一个单位事件“零”，其既不是红色，也不是黑色，而是绿色。

4.3 概率

古典概率

古典概率又叫传统概率或拉普拉斯概率，古典概率的定义是由法国数学家拉普拉斯（Laplace）提出的。如果一个随机试验所包含的单位事件是有限的，且每个单位事件发生的可能性均相等，则这个随机试验叫作拉普拉斯试验。在拉普拉斯试验中，事件 A 在事件空间 S 中的概率 $P(A)$ 为：

$$P(A) = \text{构成事件} A \text{的元素数目} / \text{构成事件空间} S \text{的所有元素数目}$$

例如，在一次同时掷一个硬币和一个骰子的随机试验中，假设事件 A 为获得国徽面且点数大于 4，那么事件 A 的概率应该有如下计算方法： $S = \{ (\text{国徽}, 1 \text{点}), (\text{数字}, 1 \text{点}), (\text{国徽}, 2 \text{点}), (\text{数字}, 2 \text{点}), (\text{国徽}, 3 \text{点}), (\text{数字}, 3 \text{点}), (\text{国徽}, 4 \text{点}), (\text{数字}, 4 \text{点}), (\text{国徽}, 5 \text{点}), (\text{数字}, 5 \text{点}), (\text{国徽}, 6 \text{点}), (\text{数字}, 6 \text{点}) \}$ ， $A = \{ (\text{国徽}, 5 \text{点}), (\text{国徽}, 6 \text{点}) \}$ ，按照拉普拉斯定义， A 的概率为

$$P(A) = \frac{2}{12} = \frac{1}{6}$$

注意到在拉普拉斯试验中存在着若干的疑问，在现实中是否存在其单位事件的概率具有精确相同的概率值的试验？因为我们不知道，硬币和骰子是否完美，即骰子制造的是否均匀，其重心是否位于正中心，以及轮盘是否倾向于某一个数字。尽管如此，传统概率在实践中被广泛应用于确定事件的概率值，其理论根据是：如果没有足够的论据来证明一个事件的概率大于另一个事件的概率，那么可以认为这两个事件的概率值相等。

如果仔细观察这个定义，则会发现拉普拉斯用概率解释了概率，定义中用了相同的可能性（原文是 *également possible*）一词，其实指的就是“相同的概率”。这个定义也并没有说

出到底什么是概率，以及如何用数字来确定概率。在现实生活中也有一系列问题，无论如何不能用传统概率定义来解释，比如，人寿保险公司无法确定一个 50 岁的人在下一年将死去的概率。

古典概率的两个特点

- (1) 样本空间的元素只有有限个。
- (2) 实验中每个基本事件发生的可能性相同。

统计概率：大数定律

继传统概率论之后，英国逻辑学家约翰·维恩和奥地利数学家理查德提出了建立在频率理论基础上的统计概率。他们认为，获得一个事件的概率值的唯一方法是通过对该事件进行 100 次、1000 次或者甚至 10000 次的前后相互独立的 n 次随机试验，针对每次试验均记录下绝对频率值 $h_n(A)$ 和相对频率值 f_n ，随着试验次数 n 的增加，会出现如下事实，即相对频率值会趋于稳定，它在一个特定的值上下浮动，也就是说存在一个极限值 $P(A)$ ，相对频率值趋向于这个极限值。这个极限值被称为统计概率，表示为

$$P(A) = \lim_{n \rightarrow \infty} f_n(A)$$

例如，若想知道在一次掷骰子的随机试验中获得 6 点的概率值，则可以对其进行 3000 次前后独立的扔掷试验（如表 4-1 所示），在每一次试验后记录下出现 6 点的次数，然后通过计算相对频率值可以得到趋向于某一个数的统计概率值。

表 4-1 掷骰子随机试验统计结果

扔 掷 数	获得 6 点的绝对频率	获得 6 点的相对频率
1	1	1.00000
2	1	0.50000
3	1	0.33333
4	1	0.25000
5	2	0.40000
10	2	0.20000

(续表)

20	5	0.25000
100	12	0.12000
200	39	0.19500
300	46	0.15333
400	72	0.18000
500	76	0.15200
1000	170	0.17000
2000	343	0.17150
3000	506	0.16867

上面提到的有关相对频率的经验规律是大数定律在现实生活中的反映，大数定律是初等概率论的基础。统计概率在今天的实践中依然具有重要意义，特别是在初等概率论及数理统计等学科中。

4.4 概率公理

公理 1：事件 A 的概率 $P(A)$ 是一个 0 与 1 之间（包含 0 与 1）的非负实数。

$$0 \leq P(A) \leq 1 (A \in S)$$

公理 2：事件空间的概率值为 1。

$$P(S) = 1$$

公理 3：互斥事件的加法法则。这里需要注意：公理 3 可以推广到可数个互斥事件的联集。

$$P(A \cup B) = P(A) + P(B) \quad \text{如果: } A \cap B = \emptyset$$

定理 1（互补法则）：与 A 互补事件的概率始终是

$$P(\overline{A}) = 1 - P(A), A \in S$$

定理 2：不可能事件的概率为零。

$$P(\varnothing) = 0$$

定理 3：如果若干事件 $A_1, A_2, \dots, A_n \in S$ 每两两之间是空集关系，那么这些所有事件集合的概率等于单个事件的概率的和。

$$P(A_1 \cup \dots \cup A_n) = \sum_{j=1}^n P(A_j)$$

注意，针对这一定理有效性的决定因素是 $A_1 \cdots A_n$ 事件不能同时发生。例如，在一次掷骰子中，得到 5 点或者 6 点的概率是：

$$P = P(A_5) + P(A_6) = \frac{2}{6} = \frac{1}{3}$$

定理 4：如果事件 A, B 是差集关系，则有

$$P\left(\frac{A}{B}\right) = P(A) - P(A \cap B)$$

定理 5（任意事件加法法则）：对于事件空间 S 中的任意两个事件 A 和 B ，有如下定理。

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

例如，在由一共 32 张牌构成的斯卡特扑克牌中随机抽出一张，其或者是“方片”或者是“A”的概率是多少？

事件 A, B 是或者的关系，且可同时发生，就是说抽出的这张牌即可以是“方片”，又可以是“A”， $A \cap B$ （既发生 A 又发生 B ）的值是 $1/32$ ，因此有如下结果：

$$P(A \cup B) = \frac{8}{32} + \frac{4}{32} - \frac{1}{32} = \frac{11}{32}$$

定理 6（乘法法则）：事件 A, B 同时发生的概率是

$$P(A \cap B) = P(A) \cdot P(B|A) = P(B) \cdot P(A|B)$$

公式中的 $P(A|B)$ 是指在 B 条件下 A 发生的概率，又称作条件概率。回到上面的斯卡特游戏中，在 32 张牌中随机抽出一张，即是“方片”又是“A”的概率是多少呢？现用 $P(A)$ 代表抽出“方片”的概率，用 $P(B)$ 代表抽出“A”的概率，很明显， A 、 B 之间有一定联系，即 A 里包含 B ， B 里又包含 A ，在 A 的条件下发生 B 的概率是 $P(B|A) = 1/8$ ，则有：

$$P(A \cap B) = P(A) \cdot P(B|A) = \frac{8}{32} \times \frac{1}{8} = \frac{1}{32}$$

或者

$$P(A \cap B) = P(B) \cdot P(A|B) = \frac{4}{32} \times \frac{1}{4} = \frac{1}{32}$$

从上面的公式中也可以看出，符合条件的只有一张牌，即方片 A。另一个例子，在 32 张斯卡特牌里连续抽两张 A （第一次抽出的牌不放回去），连续得到两个的概率是多少呢？

设 A 、 B 分别为连续发生的这两次事件，我们看到， A 、 B 之间有一定联系，即 B 的概率由于 A 发生了变化，属于条件概率，按照公式有：

$$P(A \cap B) = P(A) \cdot P(B|A) = \frac{4}{32} \times \frac{3}{31} = \frac{3}{248}$$

定理 7（无关事件乘法法则）：两个不相关联的事件 A 、 B 同时发生的概率是

$$P(A \cap B) = P(A) \cdot P(B)$$

注意到这个定理实际上是定理 6（乘法法则）的特殊情况，如果事件 A 、 B 没有联系，则有 $P(A|B)=P(A)$ ，以及 $P(B|A)=P(B)$ 。现在观察一下轮盘游戏中两次连续的旋转过程， $P(A)$ 代表第一次出现红色的概率， $P(B)$ 代表第二次出现红色的概率，可以看出， A 与 B 没有关联，利用上面提到的公式，连续两次出现红色的概率为

$$P(A \cap B) = \frac{18}{37} \times \frac{18}{37} = 0.2367$$

4.5 条件概率和全概率

4.5.1 条件概率

条件概率的描述

设试验 E 的样本空间为 S ，事件包括 A 、 B ，要考虑在 A 已经发生的条件下 B 发生的概率，这就是条件概率问题。

条件概率的定义

设 A 、 B 是两个事件，且 $P(A) > 0$ ，称： $P(A|B) = \frac{P(AB)}{P(A)}$ (AB 不独立)

设 A 、 B 是两个事件，且 $P(A) > 0$ ，称： $P(A|B) = P(A)$ (AB 独立)

条件概率的性质

性质 1：对于每一个事件 B ，有 $0 \leq P(B|A) \leq 1$

性质 2： $P(S|A) = 1$

性质 3：设 $B_1, B_2, \dots, B_n$ 两两互不相容，则 $P(UB_i|A) = \sum P(B_i|A)$

条件概率的计算方法

(1) 公式法。

先计算 $P(A)$ 、 $P(AB)$ ，然后按公式计算 $P(B|A) = P(AB)/P(A)$ 。

(2) 图解法：利用概率树求解。

例如：图圈饼店正在调查客户购买圈饼和咖啡的概率，下面是一些线索，画出概率树并求解相应概率（如图 4-2 所示）。以下是已知条件：

$$\odot P(\text{圈饼}) = 3/4$$

$$\odot P(\text{咖啡} | \text{圈饼}) = 1/3$$

$$\odot P(\text{圈饼} \cap \text{咖啡}) = 9/20$$

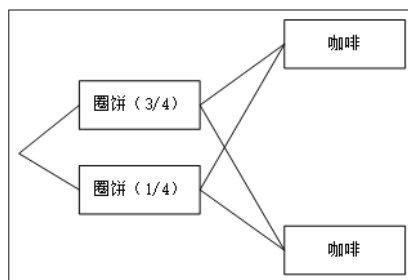


图 4-2 圈饼概率树

计算过程：

$$P(\text{咖啡} | \text{圈饼}) = P(\text{圈饼} \cap \text{咖啡}) / P(\text{圈饼}) = 3/5$$

$$P(\text{咖啡} | \text{圈饼}) = P(\text{圈饼} \cap \text{咖啡}) / P(\text{圈饼}) = 1/3$$

$$P(\text{咖啡} | \text{圈饼}) = P(\text{圈饼} \cap \text{咖啡}) / P(\text{圈饼}) = 2/5$$

$$P(\text{咖啡} | \text{圈饼}) = P(\text{圈饼} \cap \text{咖啡}) / P(\text{圈饼}) = 2/3$$

使用概率树求解问题的优缺点

- ◎ 优点：能够以图形体现条件概率，同时帮助计算概率，利用分支结构，条理清楚，不易算错。
- ◎ 不足：画概率树很浪费时间。

4.5.2 全概率

概念介绍

n 个事件 $H_1, H_2, \dots, H_n$ 互相独立，且共同组成整个事件空间 S ，即

$$H_i \cap H_j = \emptyset, (i \neq j)$$

以及

$$H_1 \cup H_2 \cup \dots \cup H_n = S$$

这时 A 的概率可以表示为

$$P(A) = \sum_{j=1}^n P(A|H_j) \cdot P(H_j)$$

举例解析

例如，一个随机试验工具由一个骰子和一个柜子中的三个抽屉组成，抽屉 1 里有 14 个白球和 6 个黑球，抽屉 2 里有 2 个白球和 8 个黑球，抽屉 3 里有 3 个白球和 7 个黑球，试验规则是首先掷骰子，如果获得小于 4 点，则抽屉 1 被选择，如果获得 4 点或者 5 点，则抽屉 2 被选择，其他情况选择抽屉 3。然后在选择的抽屉里随机抽出一个球，最后抽出的这个球是白球的概率是：

$$\begin{aligned} P(\text{白}) &= P(\text{白} | \text{抽} 1) \times P(\text{抽} 1) + P(\text{白} | \text{抽} 2) \times P(\text{抽} 2) + P(\text{白} | \text{抽} 3) \times P(\text{抽} 3) \\ &= (14/20) \times (3/6) + (2/10) \times (2/6) + (3/10) \times (1/6) \\ &= 28/60 = 0.4667 \end{aligned}$$

从例子中可看出，全概率特别适合于分析具有多层结构的随机试验的情况。

4.6 贝叶斯定理

贝叶斯公式

贝叶斯定理用来描述两个条件概率之间的关系，比如 $P(A|B)$ 和 $P(B|A)$ 。按照定理 6 的乘法法则， $P(A \cap B) = P(A) \cdot P(B|A) = P(B) \cdot P(A|B)$ ，可以立刻导出贝叶斯定理：

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

案例 1：狗叫抓贼

例如：一座别墅在过去的 20 年里一共发生过 2 次被盗事件，别墅的主人有一条狗，狗平均每周晚上叫 3 次，在盗贼入侵时狗叫的概率被估计为 0.9。问题是：在狗叫的时候发生入侵的概率是多少？

我们假设 A 事件为狗在晚上叫, B 为盗贼入侵, 则

$$P(A) = \frac{3}{7}, P(B) = \frac{2}{(20 \times 365.25)} = \frac{2}{7305}, P(A|B) = 0.9$$

按照公式很容易得出结果:

$$P(B|A) = 0.9 \times \frac{2}{7305} \times \frac{7}{3} = 0.0005749486653 \cdots$$

案例 2: 追踪红球

现分别有 A、B 两个容器, 在容器 A 里分别有 7 个红球和 3 个白球, 在容器 B 里有 1 个红球和 9 个白球, 现已知从这两个容器里任意抽出了一个球, 且是红球, 问这个红球是来自容器 A 的概率是多少?

假设已经抽出红球为事件 B , 从容器 A 里抽出球为事件 A , 则有:

$$P(B) = \frac{8}{20}, P(A) = \frac{1}{2}, P(B|A) = \frac{7}{10}$$

按照公式, 则有:

$$P(A|B) = \frac{7}{10} \times \frac{1}{2} \times \frac{20}{8} = \frac{7}{8}$$

4.7 信息论

4.7.1 信息论的基本概念

信息论

信息论 (information theory) 是应用数学、电机工程学和计算机科学的一个分支, 涉及信息的量化、存储和通信等。信息论是由香农发展的, 用来找出信号处理与通信操作的基本限制, 如数据压缩、可靠的存储和数据传输等。自创立以来, 它已拓展应用到许多其他领域, 包括统计推断、自然语言处理、密码学、神经生物学、进化论和分子编码的功能、生态学的模式选择、热物理、量子计算、语言学、剽窃检测、模式识别、异常检测和其他形式的数据分析。

信息熵

熵是信息的一个关键度量，通常用一条消息中需要存储或传输一个符号的平均比特数来表示。熵衡量了预测随机变量的值时涉及的不确定度的量。例如，指定掷硬币的结果（两个等可能的结果）比指定掷骰子的结果（六个等可能的结果）所提供的信息量更少（熵更少）。

1948 年，香农引入信息熵，将其定义为离散随机事件的出现概率。一个系统越有序，信息熵就越低；反之，一个系统越混乱，信息熵就越高。所以说，信息熵可以被认为是系统有序化程度的一个度量。

4.7.2 信息度量

信息熵

如果一个随机变量 X 的可能取值为 $X = \{x_1, x_2, \dots, x_n\}$ ，其概率分布为 $P(X = x_i) = p_i, i = 1, 2, \dots, n$ ，则随机变量 X 的熵定义为 $H(X)$ 。

$$H(X) = - \sum_{i=1}^n P(x_i) \log P(x_i) = \sum_{i=1}^n P(x_i) \frac{1}{\log P(x_i)}$$

其中 x 是定义在 X 上的随机变量。信息熵是随机事件不确定性的度量。

例子：若 S 为一个三个面的骰子。

$$\odot P(\text{面一}) = 1/5$$

$$\odot P(\text{面二}) = 2/5$$

$$\odot P(\text{面三}) = 2/5$$

$$H(X) = \frac{1}{5} \log_2(5) + \frac{2}{5} \log_2\left(\frac{5}{2}\right) + \frac{2}{5} \log_2\left(\frac{5}{2}\right)$$

联合熵

两个随机变量 X 和 Y 的联合分布可以形成联合熵，定义为联合自信息的数学期望，它是二维随机变量 X 、 Y 的不确定性的度量，用 $H(X, Y)$ 表示：

$$H(X, Y) = - \sum_{i=1}^n \sum_{j=1}^n P(x_i, y_j) \log P(x_i, y_j)$$

条件熵

在随机变量 X 发生的前提下，随机变量 Y 发生新带来的熵，定义为 Y 的条件熵，用 $H(Y|X)$ 表示：

$$H(Y|X) = - \sum_{x,y} P(x, y) \log P(y|x)$$

条件熵用来衡量在已知随机变量 X 的条件下，随机变量 Y 的不确定性。

由贝氏定理，我们有 $p(x, y) = p(y|x)p(x)$ ，代入联合熵的定义，可以分离出条件熵，于是得到联合熵与条件熵的关系式：

$$H(Y|X) = H(X, Y) - H(X)$$

推导过程如下：

$$\begin{aligned} & H(X, Y) - H(X) \\ &= - \sum_{x,y} p(x, y) \log p(x, y) + \sum_x p(x) \log p(x) \\ &= - \sum_{x,y} p(x, y) \log p(x, y) + \sum_x \left(\sum_y p(x, y) \right) \log p(x) \\ &= - \sum_{x,y} p(x, y) \log p(x, y) + \sum_{x,y} p(x, y) \log p(x) \\ &= - \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)} \\ &= - \sum_{x,y} p(x, y) \log p(y|x) \end{aligned}$$

上式中：

- ◎ 第二行推到第三行的依据是边缘分布 $p(x)$ 等于联合分布 $p(x,y)$ 的和；
- ◎ 第三行推到第四行的依据是把公因子 $\log p(x)$ 乘进去，然后把 x 、 y 写在一起；
- ◎ 第四行推到第五行的依据是：因为两个 σ 都有 $p(x,y)$ ，故提取公因子 $p(x,y)$ 放到外边，然后把里边的 $-(\log p(x,y) \log p(x))$ 写成 $\log(p(x,y)/p(x))$ ；
- ◎ 第五行推到第六行的依据是： $p(x,y) = p(x) \times p(y|x)$ ，故 $p(x,y)/p(x) = p(y|x)$ 。

相对熵：信息增益

相对熵又称互熵、交叉熵、KL 散度、信息增益，是描述两个概率分布 P 和 Q 差异的一种方法，记为 $D(P||Q)$ 。在信息论中， $D(P||Q)$ 表示当用概率分布 Q 来拟合真实分布 P 时，产生的信息损耗，其中 P 表示真实分布， Q 表示 P 的拟合分布。

对于一个离散随机变量的两个概率分布 P 和 Q 来说，它们的相对熵定义为

$$D(P||Q) = \sum_{i=1}^n P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

注意： $D(P||Q) \neq D(Q||P)$ 。

互信息

两个随机变量 X 、 Y 的互信息： X 、 Y 的联合分布和各自独立分布乘积的相对熵称为互信息，用 $I(X,Y)$ 表示。互信息是信息论里一种有用的信息度量方式，它可以看作一个随机变量中包含的关于另一个随机变量的信息量，或者说是一个随机变量由于已知另一个随机变量而减少的不肯定性。互信息是两个事件集合之间的相关性。

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

互信息、熵和条件熵之间存在以下关系：

$$H(Y|X) = H(Y) - I(X,Y)$$

推导过程如下：

$$\begin{aligned}
& H(Y) - I(X, Y) \\
&= - \sum_y p(y) \log p(y) - \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\
&= - \sum_y \left(\sum_x p(x, y) \right) \log p(y) - \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\
&= - \sum_{x,y} p(x, y) \log p(y) - \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\
&= - \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)} \\
&= - \sum_{x,y} p(x, y) \log p(y|x) \\
&= H(Y|X)
\end{aligned}$$

通过上面的计算过程发现 $H(Y|X) = H(Y) - I(X, Y)$ ，又由前面条件熵的定义 $H(Y|X) = H(X, Y) - H(X)$ ，于是有 $I(X, Y) = H(X) + H(Y) - H(X, Y)$ ，此结论被多数文献作为互信息的定义。

最大熵

最大熵原理是概率模型学习的一个准则，它认为：学习概率模型时，在所有可能的概率分布中，熵最大的模型是最好的模型。通常用约束条件来确定模型的集合，所以最大熵模型原理也可以表述为：在满足约束条件的模型集合中选取熵最大的模型。

前面我们知道，若随机变量 X 的概率分布是 $P(x_i)$ ，则其熵定义如下：

$$H(X) = - \sum_{i=1}^n P(x_i) \log P(x_i) = \sum_{i=1}^n P(x_i) \frac{1}{\log P(x_i)}$$

熵满足下列不等式：

$$0 \leq H(X) \leq \log |X|$$

式中， $|X|$ 是 X 的取值个数，当且仅当 X 的分布是均匀分布时右边的等号成立。也就是说，当 X 服从均匀分布时，熵最大。

直观地看，最大熵原理认为：要选择概率模型，首先必须满足已有的事实，即约束条件；在没有更多信息的情况下，那些不确定的部分都是“等可能的”。最大熵原理通过熵的最大化来表示等可能性；“等可能”不易操作，而熵则是一个可优化的指标。

第 5 章

统计学

导读：在数据科学中，统计的地位尤为显著。这是一门在数据分析的基础上，研究如何测定、收集、整理、归纳和分析数据规律，以便给出正确消息的学科。通过揭示数据背后的规律和隐藏信息，给相关角色提供参照价值，以做出相应的决策。其在数据挖掘、自然语言处理、机器学习中都被广泛应用。本章首先介绍常见的图形可视化的概念和使用，继而介绍数据度量标准、概率分布、统计假设检验、相关和回归。以短小精悍的篇章来使读者掌握基本的统计知识。

5.1 图形可视化

5.1.1 饼图

饼图广泛地应用在各个领域，用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块（圆弧）表示该分类占总体的比例大小，所有区块（圆弧）的加和等于 100%。

饼图的优缺点

优点：

- ◎ 饼图可以很好地帮助用户快速了解数据的占比分配

缺点：

- ◎ 饼图不适用于分类多的数据，原则上一张饼图不可多于 9 个分类，因为随着分类的增多，每个切片就会变小，最后导致大小区分不明显，每个切片看上去都差不多大小，这样对于数据的对比是没有什么意义的。所以饼图不适合用于数据量大且分类很多的场景。
- ◎ 相比于具备同样功能的其他图表（比如百分比柱状图、环图），饼图需要占据更大的画布空间。
- ◎ 很难进行多个饼图之间的数值比较。

应用场景

适合的场景：

- ◎ 展示 2 个分类的占比情况。比如一个班级的男女生的占比情况。
- ◎ 多个但不超过 9 个分类的占比情况。比如一个游戏公司的销售情况。

不适合的场景：

- ◎ 分类过多的场景。比如各个省的人口的占比情况，很难清晰对比各个省份的人口数据占比情况。所以这种情况下，推荐使用横向柱状图。
- ◎ 分类占比差别不明显的场景。比如游戏公司的不同种类的游戏的销售量相近，所以不太适合使用饼图，此时可以使用柱状图来呈现。

饼图与其他图表的对比

饼图和柱状图

- ◎ 饼图主要展示分类之间的占比情况。
- ◎ 柱状图主要展示各个分类数量大小的对比。

饼图案例

某站点用户访问来源饼状图统计如图 5-1 所示（此图引自百度 ECharts）：

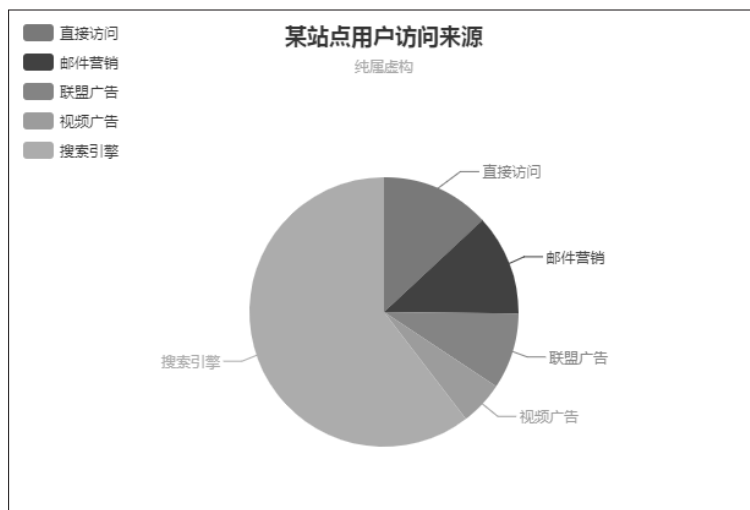


图 5-1 某站点用户访问来源

代码实现

```
series :
[
  {
    name:'访问来源',
    type:'pie',
    radius : '55%',
    center: ['50%', '60%'],
    data:[
      {value:335, name:'直接访问'},
      {value:310, name:'邮件营销'},
      {value:234, name:'联盟广告'},
      {value:135, name:'视频广告'},
      {value:1548, name:'搜索引擎'}
    ]
  }
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>, 查看 5.1.1.html 页面。

5.1.2 条形图

定义

典型的条形图（又名柱状图）使用垂直或水平的柱子显示类别之间的数值比较。其中一个轴表示需要对比的分类维度，另一个轴代表相应的数值。

柱状图有别于直方图，柱状图无法显示数据在一个区间内的连续变化趋势。柱状图描述的是分类数据，回答的是每一个分类中“有多少”的问题。需要注意的是，当柱状图显示的分类很多时，往往会导致分类名层叠等显示问题，下面我们会举例说明。

条形图的优缺点

优点：

- ◎ 可以很清晰地看出每个类的总和及各个属性的比例。

缺点：

- ◎ 不容易看出各个属性的频数。

应用场景

适合的场景：

- ◎ 适合应用到分类数据对比。比如一个游戏销量的图表，展示不同游戏类型的销量对比。

不适合的场景：

- ◎ 分类太多不适合使用纵向柱状图，如对比不同省份的人口数量。分类情况过多时，柱状图的文本为了排布合理，需要进行旋转，不利于阅读；相比于纵向柱状图，横向柱状图更适用于此类分类较多的场景。
- ◎ 不适合表示趋势。柱状图使用矩形的长度（宽度）来对比分类数据的大小，非常方便临近的数据进行大小的对比，不适合展示连续数据的趋势。比如展示 ACME 这只股票在 2015 年 9 月份整个月的每日价格走势，可是效果却不尽人意。

条形图与其他图表的对比

柱状图和折线图、饼图：

- ◎ 柱状图主要用于多个分类间的数据（大小、数值）对比。
- ◎ 折线图主要用于展示连续数值（例如，时间）或者有序分类的变化趋势。
- ◎ 饼图主要用于展示分类之间的占比情况。

条形图案例

- ◎ 水平条形图：世界人口总量条形图统计如图 5-2 所示（此图引用百度 ECharts）。

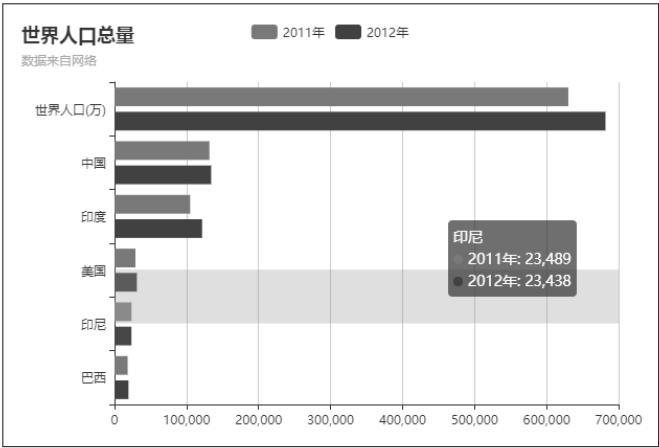


图 5-2 世界人口总量

- ◎ 垂直条形图：某地区蒸发量和降水量统计，如图 5-3 所示（此图引用百度 ECharts）。

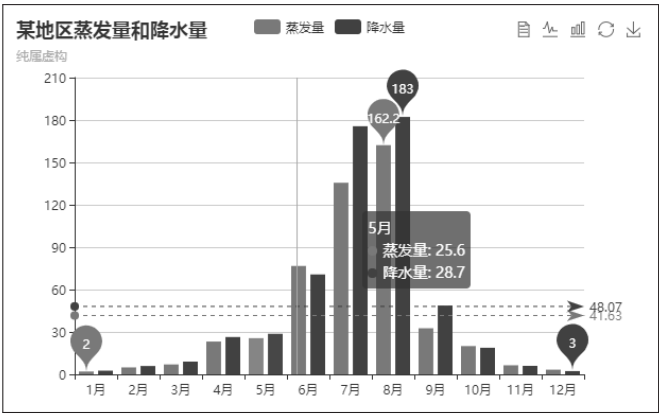


图 5-3 某地区蒸发量和降水量

◎ 堆叠条形图：某站点用户访问统计如图 5-4 所示（此图引用百度 ECharts）。

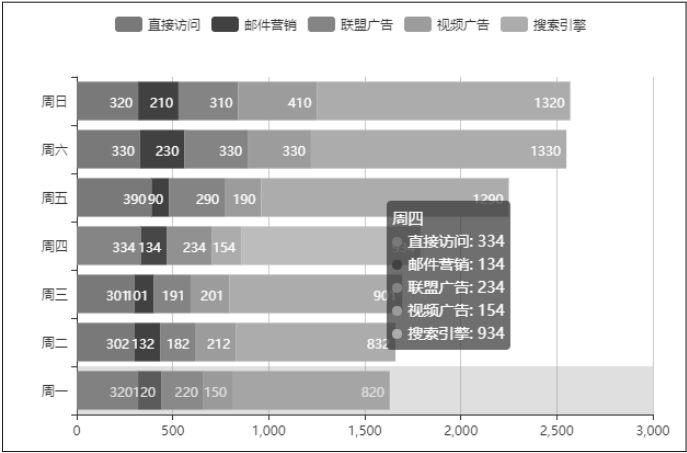


图 5-4 某站点用户访问

水平条形图代码实现

```
series:
[
  {
    name: '2011年',
    type: 'bar',
    data: [18203, 23489, 29034, 104970, 131744, 630230]
  },
  {
    name: '2012年',
    type: 'bar',
    data: [19325, 23438, 31000, 121594, 134141, 681807]
  }
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>，查看 5.1.2.html 页面。

5.1.3 热力图

定义

热力图（Heat Map）一词最初是由软件设计师 Cormac Kinney 于 1991 年提出并创造的，用来描述一个 2D 显示的实时金融市场信息。最初的热力图是矩形色块加上颜色编码。经过多年的演化，习语上的“热力图”如今更规范，更被大多数人理解的是这种经过平滑模糊过的热力图谱。

热力图的特点

- ◎ 热力图尤其关注分布。
- ◎ 热力图可以不需要坐标轴，其背景常常是图片或地图。
- ◎ 热力图一般情况用其专有的彩虹色系（rainbow）。
- ◎ 热力图能告诉你，页面的哪些部分吸引了大多数访客的注意。
- ◎ 热力图可以直观清楚地看到页面上每一个区域的访客兴趣焦点。

热力图应用场景

适合的场景：

- ◎ 连续数值数据分布。比如城市房租热力图用于显示城市房租价格分布。
- ◎ 数据的统计预测。比如钻石克拉数和价格的关系。通过已有的钻石数据，对未知区域的钻石数据进行预测。

热力图案例

笛卡儿坐标系上的热力图如图 5-5 所示（此图引用百度 ECharts）。

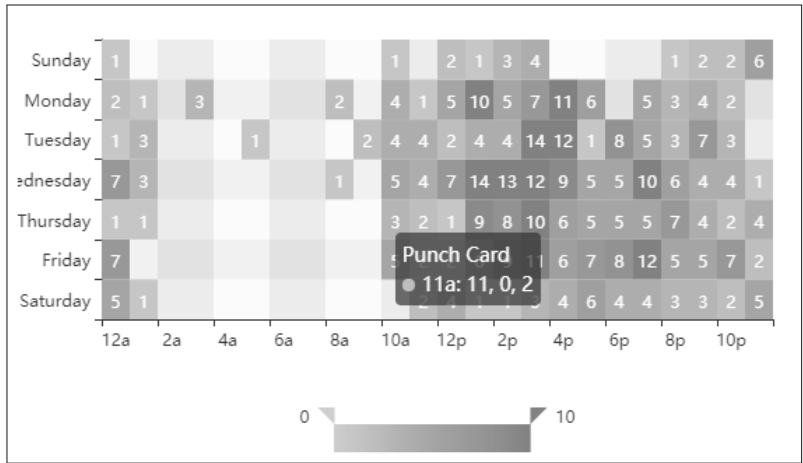


图 5-5 笛卡儿坐标系上的热力图

热力图代码实现

```
series:
[
  {
    name: 'Punch Card',
    type: 'heatmap',
    data: data,
    label: {
      normal: {
        show: true
      }
    },
    itemStyle: {
      emphasis: {
        shadowBlur: 10,
        shadowColor: 'rgba(0, 0, 0, 0.5)'
      }
    }
  }
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>，查看 5.1.3.html 页面。

5.1.4 折线图

定义

折线图用于显示数据在一个连续的时间间隔或者时间跨度上的变化，它的特点是反映事物随时间或有序类别而变化的趋势。在折线图中，数据是递增还是递减、增减的速率、增减的规律（周期性、螺旋性等）、峰值等特征都可以清晰地反映出来。所以，折线图常用来分析数据随时间变化的趋势，也可用来分析多组数据随时间变化的相互作用和相互影响。例如，可用来分析某类商品或是某几类相关的商品随时间变化的销售情况，从而进一步预测未来的销售情况。在折线图中，一般水平轴（X 轴）用来表示时间的推移，并且间隔相同；而垂直轴（Y 轴）则代表不同时刻的数据大小。

折线图应用场景

如果分类标签是文本并且代表均匀分布的数值（如月、季度或财政年度），则应该使用折线图。当有多个系列时，尤其适合使用折线图；对于一个系列，应该考虑使用类别图。如果有几个均匀分布的数值标签（尤其是年），也应该使用折线图。如果拥有的数值标签多于十个，则改用散点图。另外，折线图是支持多数据进行对比的。

适合的场景：

- ◎ 有序的因变量，比如时间。某监控系统的折线图表，显示了请求次数和响应时间随时间的变化趋势。
- ◎ 不同月份的温度。

不适合的场景：

- ◎ 当水平轴的数据类型为无序的分类或者垂直轴的数据类型为连续时间时，不适合使用折线图。
- ◎ 当折线的条数过多时不建议将多条线绘制在一张图上。

折线图与其他图表的对比

折线图和柱状图：

- ◎ 柱状图主要用于多个分类间的数据（大小、数量）对比，折线图主要用于描述时间或者连续数据上的趋势。

◎ 分类间的数据比较，如果分类不存在顺序，那么不要使用折线图。

折线图和面积图：

◎ 折线图和面积图都可以表示一段时间（或者有序分类）的趋势，相比之下面积图的表现力更强一些。

◎ 面积图还可以表示数据的上下限，例如，可以表示温度的最小值、最大值。

折线图案例

未来一周气温变化折线图如图 5-6 所示（此图引用百度 ECharts）。

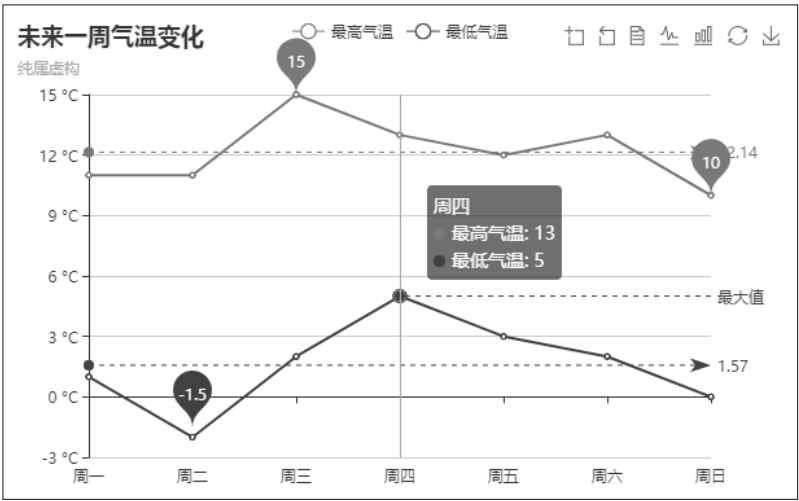


图 5-6 未来一周气温变化

折线图代码实现

```
series:
[
  {
    name: '最高气温',
    type: 'line',
    data: [11, 11, 15, 13, 12, 13, 10],
    markPoint: {
      data: [
```

```

        {type: 'max', name: '最大值'},
        {type: 'min', name: '最小值'}
    ]
},
markLine: {
    data: [
        {type: 'average', name: '平均值'}
    ]
}
},
{
    name: '最低气温',
    type: 'line',
    data: [1, -2, 2, 5, 3, 2, 0],
    markPoint: {
        data: [
            {name: '周最低', value: -2, xAxis: 1, yAxis: -1.5}
        ]
    },
    markLine: {
        data: [
            {type: 'average', name: '平均值'},
            [{
                symbol: 'none',
                x: '90%',
                yAxis: 'max'
            }, {
                symbol: 'circle',
                label: {
                    normal: {
                        position: 'start',
                        formatter: '最大值'
                    }
                }
            }, {
                type: 'max',
                name: '最高点'
            }
        ]
    }
}]

```

```
    ]  
  }  
}  
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>，查看 5.1.4. html 页面。

5.1.5 箱线图

定义

箱形图又称盒须图、盒式图或箱线图，是一种用于显示一组数据分布情况的统计图。

如果一个数据集中包含了一个分类变量和大于等于一个的连续变量，那么你可能会想知道连续变量是如何随着分类变量的水平变化而变化的。箱形图就可以提供这种方法，它只用了 5 个数字对分布进行概括，即一组数据的最大值、最小值、中位数、下四分位数及上四分位数。对于数据集中的异常值，通常会以散点的形式绘制。箱形图可以水平或者垂直绘制。

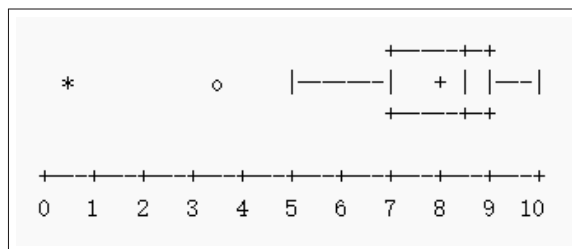
箱形图多用于数值统计。虽然相比于直方图和密度曲线更为原始简单，但是它不需要占据过多的画布空间，空间利用率高，非常适用于描述多组数据的分布情况。

从箱形图中我们可以观察到：

- ⊙ 一组数据的关键值——中位数、最大值、最小值等。
- ⊙ 数据集中是否存在异常值，以及异常值的具体数值。
- ⊙ 数据是否是对称的。
- ⊙ 这组数据的分布是否密集。
- ⊙ 数据是否扭曲，即是否有偏向性。

箱线图特点分析

以下是箱形图的具体例子：



这组数据显示出：

- ⊙ 最小值 (minimum) = 5。
- ⊙ 下四分位数 (Q1) = 7。
- ⊙ 中位数 (Med, 也就是 Q2) = 8.5。
- ⊙ 上四分位数 (Q3) = 9。
- ⊙ 最大值 (maximum) = 10。
- ⊙ 平均值 = 8。
- ⊙ 四分位间距 (interquartile range) = $Q3 - Q1 = 2$ (即 ΔQ) 在区间 $Q3 + 3\Delta Q$, $Q1 - 3\Delta Q$ 之外的值被视为应忽略 (farout)。
- ⊙ farout: 在图上不予显示, 仅标注一个符号 ∇ 。
- ⊙ 最大值区间: $Q3 + 1.5\Delta Q$ 。
- ⊙ 最小值区间: $Q1 - 1.5\Delta Q$, 最大值与最小值产生于这个区间。区间外的值被视为 outlier 显示在图上。
- ⊙ mild outlier = 3.5。
- ⊙ extreme outlier = 0.5。

箱线图应用场景

- ⊙ 关注于一组数据的分布情况。
- ⊙ 分组箱形图。为了更清晰地比较不同品种间相同属性数值的区别。
- ⊙ 一维箱形图。箱形图有多种变换, 这里介绍一维箱形图, 在一维坐标系中, 通过添加某一属性, 可以为该一维箱形图再增加一个展示维度, 即分类。

箱线图案例

两种性别的三种基因含量表箱线图如图 5-7 所示（此图引用百度 ECharts）。

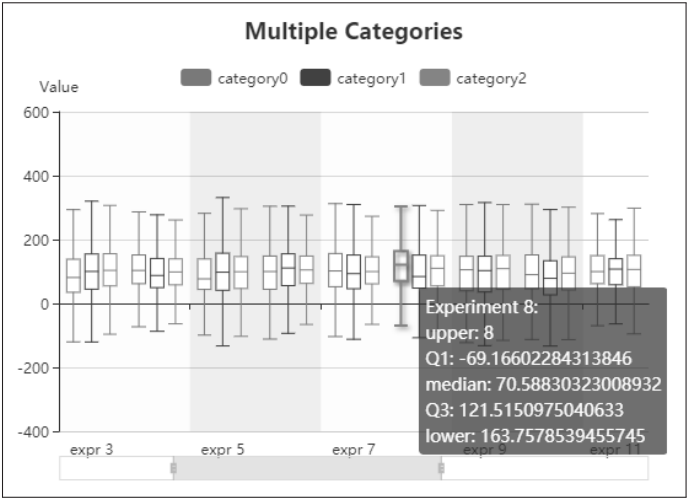


图 5-7 两种性别的三种基因含量表

箱线图代码实现

```
series:
[
  {
    name: 'category0',
    type: 'boxplot',
    data: data[0].boxData,
    tooltip: {formatter: formatter}
  },
  {
    name: 'category1',
    type: 'boxplot',
    data: data[1].boxData,
    tooltip: {formatter: formatter}
  },
  {
    name: 'category2',
```

```
        type: 'boxplot',
        data: data[2].boxData,
        tooltip: {formatter: formatter}
    }
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>, 查看 5.1.5. html 页面。

5.1.6 散点图

定义

散点图也叫 X-Y 图，它将所有的数据以点的形式展现在直角坐标系上，以显示变量之间的相互影响程度，点的位置由变量的数值决定。

通过观察散点图上数据点的分布情况，我们可以推断出变量间的相关性。如果变量之间不存在相互关系，那么在散点图上就会表现为随机分布的离散的点，如果存在某种相关性，那么大部分的数据点就会相对密集并以某种趋势呈现。数据的相关关系主要分为：正相关（两个变量值同时增长）、负相关（一个变量值增加另一个变量值下降）、不相关、线性相关、指数相关等。那些离点集群较远的点称为离群点或者异常点。

散点图经常与回归线（就是最准确地贯穿所有点的线）结合使用，归纳分析现有数据以进行预测分析。

对于那些存在密切关系，但是这些关系又不像数学公式和物理公式那样能够精确表达的变量，散点图是一种很好的图形工具。但是在分析过程中需要注意，这两个变量之间的相关性并不等同于确定的因果关系，也可能需要考虑其他的影响因素。

散点图的特点

散点图通常用于显示和比较数值。不仅可以显示趋势，还能显示数据集群的形状，以及在数据云团中各数据点的关系。

散点图应用场景

- ◎ 男女身高和体重的例子来展示上述所描述的散点图的功能。

散点图与其他图表的对比

散点图和折线图：

- ◎ 折线图可以显示随单位（如单位时间）而变化的连续数据，因此非常适用于显示在相等时间间隔下数据的趋势。
- ◎ 散点图显示若干数据系列中各数值之间的关系，或者将两组数绘制为 X、Y 坐标的一个系列。
- ◎ 在折线图中，类别数据沿水平轴均匀分布，所有值数据沿垂直轴均匀分布，即折线图只有一个数据轴（即垂直轴）。
- ◎ 散点图有两个数值轴，沿水平轴（X 轴）方向显示一组数值数据，沿垂直轴（Y 轴）方向显示另一组数值数据。散点图将这些数值合并到单一数据点并以不均匀间隔或簇显示它们。散点图通常用于显示和比较数值，例如，科学数据、统计数据 and 工程数据。

散点图和气泡图：

- ◎ 散点图和气泡图都会将两个字段映射到 X、Y 轴的位置上。散点图侧重于展示点之间的分布规律，而气泡图将数值映射到气泡的大小上，增加了一个维度的数据展示。
- ◎ 散点图可以展示成千上万个点的数据，而气泡图为了防止气泡的互相遮挡，需要根据画布的大小控制数据的规模。

散点图案例

男性女性身高体重分布散点图如图 5-8 所示（此图引用百度 ECharts）。

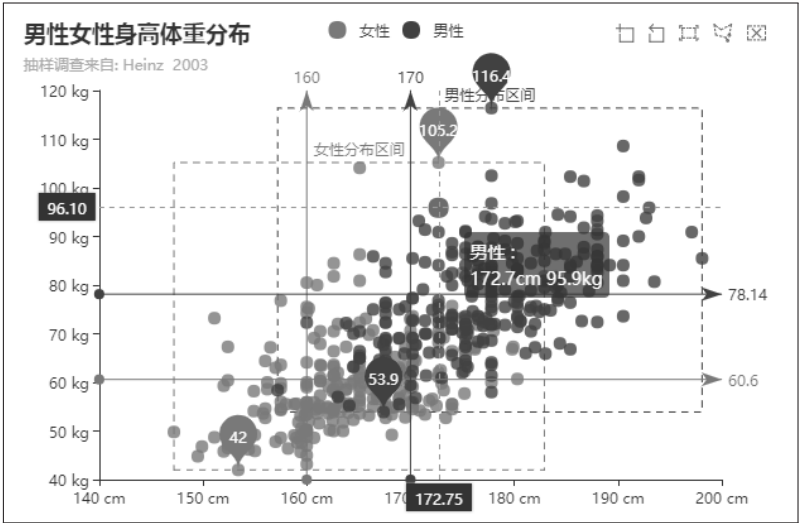


图 5-8 男性女性身高体重分布

散点图代码实现

```
data:
[[
  {
    name: '女性分布区间',
    xAxis: 'min',
    yAxis: 'min'
  },
  {
    xAxis: 'max',
    yAxis: 'max'
  }
]]

data: [[
  {
    name: '男性分布区间',
    xAxis: 'min',
    yAxis: 'min'
  },
  {
    xAxis: 'max',
    yAxis: 'max'
  }
]]
```

```
{  
    xAxis: 'max',  
    yAxis: 'max'  
}  
]]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>, 查看 5.1.6. html 页面。

5.1.7 雷达图

定义

雷达又叫戴布拉图、蜘蛛网图。传统的雷达图被认为是一种表现多维（4 维以上）数据的图表。它将多个维度的数据量映射到坐标轴上，这些坐标轴起始于同一个圆心点，通常结束于圆周边缘，将同一组的点使用线连接起来就成为雷达图。它可以将多维数据进行展示，但是点的相对位置和坐标轴之间的夹角是没有任何信息量的。在坐标轴设置恰当的情况下雷达图所围面积能表现出一些信息量。

每一个维度的数据都分别对应一个坐标轴，这些坐标轴具有相同的圆心，以相同的间距沿着径向排列，并且各个坐标轴的刻度相同。连接各个坐标轴的网格线通常只作为辅助元素。将各个坐标轴上的数据点用线连接起来就形成了一个多边形。坐标轴、点、线、多边形共同组成了雷达图。

雷达图的优缺点

优点：

- ◎ 雷达图可以展示出数据集中各个变量的权重高低情况，非常适用于展示性能数据。

缺点：

- ◎ 如果雷达图上多边形过多则会使可读性下降，使整体图形过于混乱。特别是有颜色填充的多边形的情况，上层会遮挡覆盖下层多边形。

- ④ 如果变量过多，也会造成可读性下降。因为一个变量对应一个坐标轴，这样会使坐标轴过于密集，使图表给人的感觉很复杂。所以最佳实践就是尽可能控制变量的数量以使雷达图保持简洁清晰。

雷达图应用场景

- ④ 世界经济论坛不久前发布了全球竞争力指数报告，通过基本要求、效率增强器、创新与成熟因素等三个大方面对全球国家和地区进行竞争力评估。中国排名第 28，得分为 4.89。通过雷达图，我们可以清晰看出中国在各个因素下的得分情况，进而进行分析。
- ④ 常常表示由多个维度组成的能力衡量。比如展示了华为 Mate 和中兴 Grand Memo 两款手机的综合表现雷达图，分别从易用性、功能、拍照、跑分、续航五个维度进行考核，可以看出两款手机在这个维度方面的性能都比较平衡，同时也可逐项对比。

雷达图案例

浏览器占比变化雷达图如图 5-9 所示（此图引用百度 ECharts）。

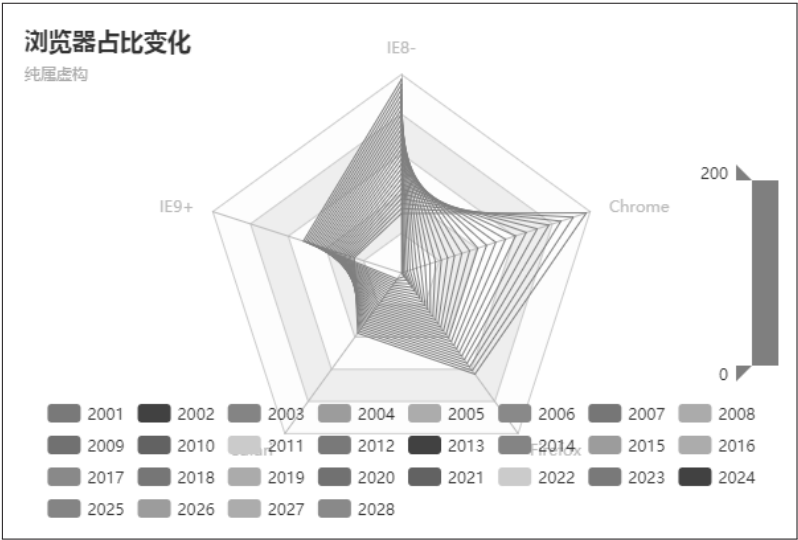


图 5-9 浏览器占比变化

雷达图代码实现

```
series.push({
  name: '浏览器（数据纯属虚构）',
  type: 'radar',
  symbol: 'none',
  itemStyle: {
    normal: {
      lineStyle: {
        width: 1
      }
    },
    emphasis: {
      areaStyle: {color: 'rgba(0,250,0,0.3)'}
    }
  },
  data: [
    {
      value: [
        (40 - i) * 10,
        (38 - i) * 4 + 60,
        i * 5 + 10,
        i * 9,
        i * i / 2
      ],
      name: i + 2000 + ''
    }
  ]
})
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>，查看 5.1.7. html 页面。

5.1.8 仪表盘

定义

仪表盘（Gauge）是一种拟物化的图表，刻度表示度量，指针表示维度，指针角度表示数值。仪表盘图表就像汽车的速度表一样，有一个圆形的表盘及相应的刻度，有一个指针指向当前数值。目前很多的管理报表或报告上用的都是这种图表，可以直观地表现出某个指标的进度或实际情况。

仪表盘的优缺点

优点：

- ◎ 仪表盘的好处在于它能跟人们的常识结合，使大家马上能理解看什么、怎么看。拟物化的方式使图标变得更友好、更人性化，正确使用可以提升用户体验。
- ◎ 仪表盘的圆形结构可以更有效地利用空间。

仪表盘应用场景

适合仪表盘的场景：

- ◎ 时钟与表；
- ◎ 投资收益率。

仪表盘案例

业务指标完成度仪表盘如图 5-10 所示（此图引用百度 ECharts）。

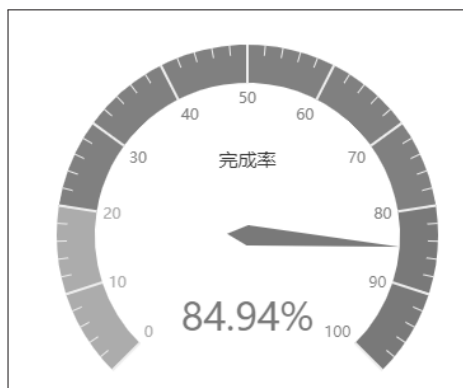


图 5-10 业务指标完成度仪表盘

仪表图代码实现

```
series:
[
  {
    name: '业务指标',
    type: 'gauge',
    detail: {formatter: '{value}%'},
    data: [{value: 50, name: '完成率'}]
  }
]
```

源码请访问 <https://github.com/BaiNingchao/NLPDome/blob/master/05chapter.zip>，查看 5.1.8. html 页面。

5.1.9 可视化图表用法

可视化比较图

可视化的方法显示值与值之间的不同和相似之处。使用图形的长度、宽度、位置、面积、角度和颜色来比较数值的大小，通常用于展示不同分类间的数值对比、不同时间点的数据对比。包括柱状图、气泡图、子弹图、色块图、漏斗图、直方图、K线图、马赛克图、雷达图、玉块图、螺旋图、层叠面积图、层叠柱状图、矩形树图、词云。

可视化分布图

可视化的方法显示频率、数据分散在一个区间或分组。使用图形的位置、大小、颜色的渐变程度来表现数据的分布情况，通常用于展示连续数据上数值的分布情况。包括箱形图、气泡图、色块图、等高线、分布曲线图、点描法地图、热力图、直方图、散点图、茎叶图。

可视化流程图

可视化的方法显示流程流转和流程流量。一般流程都会呈现出多个环节，每个环节之间会有相应的流量关系，这类图形可以很好地表示这些关系。包括漏斗图、桑基图。

可视化占比图

可视化的方法显示同一维度上的占比关系。包括环图、马赛克图、饼图、层叠面积图、层叠柱状图、矩形树图。

可视化区间图

可视化的方法显示同一维度上值的上下限之间的差异。使用图形的大小和位置表示数值的上限和下限，通常用于表示数据在某一个分类（时间点）上的最大值和最小值。包括仪表盘、层叠面积图。

可视化关联图

可视化的方法显示数据之间的相互关系。使用图形的嵌套和位置表示数据之间的关系，通常用于表示数据之间的前后顺序、父子关系及相关性。包括弧长链接图、和弦图、桑基图、矩形树图、韦恩图。

可视化趋势图

可视化的方法分析数据的变化趋势。使用图形的位置表现出数据在连续区域上的分布，通常展示数据在连续区域上的大小变化的规律。包括面积图、K线图、卡吉图、折线图、回归曲线图、层叠面积图。

可视化时间图

可视化的方法显示以时间为特定维度的数据。使用图形的位置表现出数据在时间上的分布，通常用于表现数据在时间维度上的趋势和变化。包括面积图、K线图、卡吉图、折线图、螺旋图、层叠面积图。

可视化地理图

可视化的方法显示地理区域上的数据。使用地图作为背景，通过图形的位置来表现数据的地理位置，通常来展示数据在不同地理区域上的分布情况。包括带气泡的地图、分级统计地图、点描法地图。

Echarts 扩展学习：<http://echarts.baidu.com/index.html>。

AntV 扩展学习：<https://antv.alipay.com/index.html>。

5.2 数据度量标准

5.2.1 平均值

定义

均值是统计中的一个重要概念，为集中趋势的最常用测度值，目的是确定一组数据的均衡点。

均值的计算

将所有的数字加起来，然后除以数字的个数。可以记为

$$\mu = \frac{\sum_i^n x_i}{n}$$

应用实例

寒假英语兴趣班的总人数为 28，总共有 7 个小组。知道了总的人数和总共有多少个小
组，即可求出每组人数的均值 μ 。

$$\mu = \frac{28}{7} = 4$$

均值的优缺点

优点：可以用它来反映一组数据的一般情况，也可以用它进行不同组数据的比较，以看出组与组之间的差别。

缺点：只能应用于数值型数据，不能用于分类数据和顺序数据。

5.2.2 中位数

定义

中位数又称中值，是统计学中的专有名词。代表一个样本、种群或概率分布中的一个数值，其可将数值集合划分为相等的上下两部分。在 n 个数据由大到小排序后，位置处在中间的数字。

中位数的计算

- (1) 按顺序排列数字：从最小值排列到最大值。
- (2) 如果有奇数 n 个数值，则中位数为位于中间的数值。中间数的位置为 $\frac{n+1}{2}$ 。
- (3) 如果有偶数 n 个数值，则将两个中间数相加，然后除以 2。中间位置的算法是 $\frac{n}{2}$ 。

应用实例

有一组数据如表 5-1 所示，求出该组数据的中位数。

表 5-1 年龄信息统计表

年 龄	频 数
19	3
20	6
21	3
147	1
145	1

- (1) 按顺序排列数字：从最小值排列到最大值。

19 19 19 20 20 20 20 20 20 21 21 21 145 147

- (2) 统计总的个数为 $3 + 6 + 3 + 1 + 1 = 14$ 个。
- (3) 因为是偶数，所以选择算法 $\frac{n}{2} = \frac{14}{2} = 7$ 。
- (4) 找到第七个位置，则中位数是 20（注意，计算机中下标从 0 开始，即寻找 7-1 的位置）。

中位数的特点

一个数集中最多有一半的数值小于中位数，最多有一半的数值大于中位数。如果大于和小于中位数的数值个数均少于一半，那么数集中必有若干值等同于中位数。

5.2.3 众数

定义

众数指一组数据中出现次数最多的数据值。例如，在 {2,3,3,3} 中，出现最多的是 3，因此众数是 3，众数可能是一个数，也可能是多个数。众数主要用于分类数据，也可用于顺序数据和数值型数据。

众数的计算

- (1) 把数据中的不同类别或数值全部找出来。
- (2) 写出每个数值或类别的频数。
- (3) 挑出具有最高频数的一个或几个数值，得出众数。

应用实例

有一组数据：19 19 19 20 20 20 20 20 20 21 21 21 147 145，求出该组数据的中位数。

- (1) 把数据中的不同类别或数值全部找出来：19,20,21,147,145。
- (2) 写出每个数值或类别的频数，如表 5-2 所示。

表 5-2 年龄数值或类别的频数

年 龄	频 数
19	3
20	6
21	3
147	1
145	1

- (3) 挑出具有最高频数的一个或几个数值，得出众数。观察易知是 20，总共出现 6 次。

扩展：如果将上面一组数字再多加 3 个 19 和 3 个 21，则众数就变成三个，即 19、20 和 21。

众数的特点

- ③ 在离散概率分布中，众数是指概率质量函数有最大值的数据，也就是最容易取样到的数据。在连续概率分布中，众数是指概率密度函数有最大值的数据，也就是概率密度函数的峰值。
- ③ 在高斯分布（正态分布）中，众数位于峰值，和平均数、中位数相同。若分布是高度偏斜分布，则众数可能会和平均数、中位数有很大的差异。
- ③ 用众数代表一组数据，适合在数据量较多时使用，且众数不受极端数据的影响，并且求法简便。在一组数据中，如果个别数据有很大的变动，则选择中位数表示这组数据的“集中趋势”就比较适合。

5.2.4 期望

定义

在概率论和统计学中，“期望”为期望值的简称，是指在一个离散型随机变量试验中每次可能结果的概率乘以其结果的总和。随机变量 X 的期望通常写作 $E(X)$ ，但有时也会写作 μ ，也就是均值的符号。下面是 $E(X)$ 的计算式：

$$E(X) = \sum xP(X = x)$$

应用案例

假设表 5-3 为游戏机的概率分布。

表 5-3 游戏机的概率分布

组 合	无	柠 檬	樱 桃	美元/樱桃	美 元
x	-1	4	9	14	19
$P(X=x)$	0.977	0.008	0.008	0.006	0.001

游戏机收益的期望：

$$\mu = E(X) = \sum xP(X = x) = -1 \times 0.977 + 4 \times 0.008 + 14 \times 0.006 + 19 \times 0.001 = -0.77$$

游戏机收益的方差（方差见 5.2.5 节）：

$$\begin{aligned}
 \text{Var}(X) &= E(X - \mu)^2 \\
 &= \sum (x - \mu)^2 P(X = x) \\
 &= (-1 + 0.77)^2 \times 0.977 + \dots + (19 + 0.77)^2 \times 0.001 \\
 &= 2.6971
 \end{aligned}$$

游戏机收益的期望（标准差见 5.2.6 节）：

$$\sqrt{\text{Var}(X)} = 1.642$$

5.2.5 方差

定义

形式化描述：

在概率论和统计学中，一个随机变量的方差描述的是它的离散程度，也就是该变量离其期望值的距离。方差是度量数据分散性的一种方法，是数据与均值间距离的平方数的平值。

数学化描述：

设 X 为服从分布 F 的随机变量，如果 $E[X]$ 是随机变数 X 的期望值（平均数 $\mu = E[X]$ ），则随机变量 X 或者分布 F 的方差为

$$\text{Var}(X) = \sigma^2 = \frac{\sum E(X - \mu)^2}{N}$$

计算方法

连续随机变数：

如果随机变数 X 是连续分布，并对应概率密度函数 $f(x)$ ，则其方差为

$$\text{Var}(X) = \sigma^2 = \int (x - \mu)^2 f(x) dx = \int x^2 f(x) dx - \mu^2$$

此处 μ 是期望值:

$$\mu = \int x f(x) dx$$

离散随机变数:

如果随机变数 X 是具有概率质量函数的离散概率分布 $x_1 \rightarrow p_1, \dots, x_n \rightarrow p_n$, 则:

$$\text{Var}(X) = \sum_{i=1}^n p_i \cdot (x_i - \mu)^2 = \sum_{i=1}^n (p_i \cdot x_i^2) - \mu^2$$

此处 μ 是其期望值:

$$\mu = \sum_{i=1}^n (p_i \cdot x_i)$$

特点

◎ 方差不会是负的, 因为次方计算为正或为零。

5.2.6 标准差

定义

形式化描述: 标准差是描述典型值与均值距离的一种方法。标准差越小, 数值离均值越近。

数学化描述: 标准差 $\sigma = \text{方差开方}$ 。

注: 标准差也有可能为 0, 如果每个数值与均值的距离都是为 0, 则标准差为 0。

计算方法:

- (1) 计算出该组数据的均值 u 。
- (2) 统计该组数据的个数 n 。
- (3) 利用方差的公式计算出方差。
- (4) 利用标准差的公式计算出标准差。

5.2.7 标准分

定义

标准分数也叫 z 分数，是一种具有相等单位的量数。它是将原始分数与团体的平均数之差除以标准差所得的商数，是以标准差为单位度量原始分数离开其平均数的分数之上多少个标准差，或是在平均数之下多少个标准差。它是一个抽象值，不受原始测量单位的影响，并可接受进一步的统计处理。

计算公式

用公式表示为 $z = (x - \mu) / \sigma$ ；其中 z 为标准分数； x 为某一具体分数， μ 为平均数， σ 为标准差。

z 值的量代表着原始分数和母体平均值之间的距离，以标准差为单位计算。在原始分数低于平均值时， z 为负数，反之则为正数。

特点

标准分数是一种不受原始测量单位影响的数值。其除了能够表明原数据在其分布中的位置，还能对未来不能直接比较的各种不同单位的数据进行比较。例如，比较各个学生的成绩在班级成绩中的位置，或比较某个学生在两门或多门测验中所得分数的优劣。

应用实例

例如，有两名考生的高考入学考试成绩，乙考生的总分是 400 分，而甲只有 382 分，按总分录取则录取乙，若按标准分数录取则应录取甲，因为甲的所有成绩都不低于平均分数，而乙却在数学、外语二门学科上低于平均分数，可见把分数标准化（转换为标准分数）是有好处的。

其应用在数据预处理的数据归一化中，比如在数据集中，列向量显示英语、数学等成绩。其中一个列向量可能是绩点，比如 3.2。这个 3.2 与成绩 90 分根本不在一个数据量级上，这种情况下会用到标准分进行数据归一化处理。

5.3 概率分布

5.3.1 几何分布

定义

几何分布是离散型概率分布，如图 5-11 所示。在 n 次伯努利试验中，试验 k 次才得到第一次成功的概率。

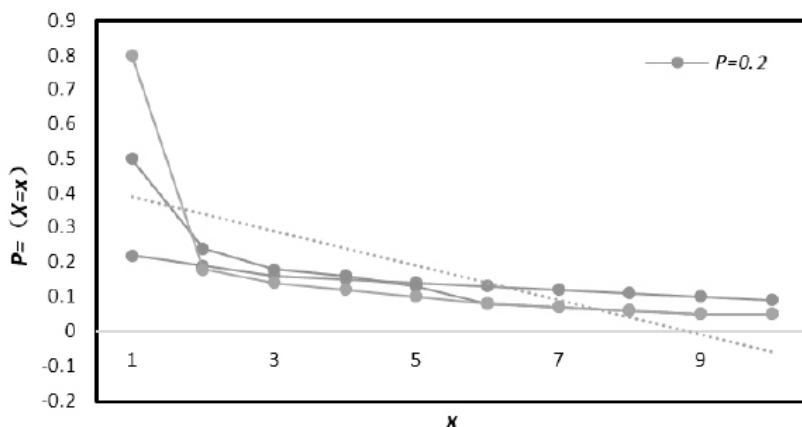


图 5-11 离散型概率分布

详细地说，是前 $k-1$ 次皆失败，第 k 次成功的概率。几何分布公式如下：

$$P(X = r) = q^{r-1}p$$

计算公式

成功概率为 p ，失败概率为 q ，试验次数为 r ，则有如下情况。

- ◎ 第 r 次试验第一次成功： $P(X = r) = pq^{r-1}$ 。
- ◎ 需要试验 r 次以上才第一次成功： $P(X > r) = q^r$ 。
- ◎ 试验 r 次或者不到 r 次才第一次成功： $P(X < r) = 1 - q^r$ 。

几何分布的条件

- (1) 进行一系列相互独立的实验。
- (2) 每一次实验既有成功，又有失败的可能，且单次实验成功的概率相等。
- (3) 为了取得第一次成功需要进行多少次实验。

几何分布的期望

期望特点：随着 x 变大，累计总数和越来越接近一个特定值。

$$P(X) = \frac{1}{p}$$

几何分布的方差

方差特点：随着 x 变大，方差越来越接近特定值。

$$\text{Var}(X) = \frac{q}{p^2}$$

应用范围

- ◎ 应用科学：数学及相关领域。
- ◎ 适用领域范围：自然数学、应用数学、高等数学、概率论。
- ◎ 射击比赛等。

5.3.2 二项分布

定义

二项分布即重复 n 次独立的伯努利试验。在每次试验中只有两种可能的结果，而且两种结果发生与否互相对立，并且相互独立，与其他各次试验结果无关。事件发生与否的概率在每一次独立试验中都保持不变，二项分布图如图 5-12 所示。

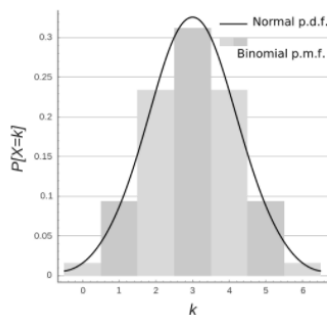


图 5-12 二项分布

计算公式

在相互独立事件中，每道题答对的概率为 p ，答错的概率为 q 。在 n 个问题中答对 r 个问题的概率为二项分布，表达式是 $X \sim B(n, p)$ ， x 表示 n 次随机变量 ξ 次成功数， p 表示成功的概率。计算公式：

$$P(X = r) = {}^n C_r \times p^r \times q^{n-r}, (r = 0, 1, \cdots, n)$$

二项分布的条件

- (1) 正在进行一系列独立试验。
- (2) 每次试验都可能失败和成功，每一次成功概率相同。
- (3) 试验次数有限。

二项分布形状特点如图 5-13 所示。

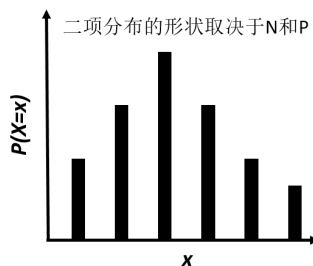


图 5-13 二项分布特点

$p < 0.5$ 时，图形向右偏移；当 $p > 0.5$ 时，图形向左偏移。

优缺点

- ◎ 优点：在试验次数一定，求成功次数时，几何分布显示不适合的情况下，二项分布能更好地解决这类问题。
- ◎ 缺点：面对试验次数不固定，发生事件概率的情况下，显然几何分布与二项分布都不能解决问题，这里也体现出泊松分布的优势。

二项分布的期望

$$E(X) = np$$

几何分布的方差

$$\text{Var}(X) = npq, (q = 1 - p)$$

应用范围

- ◎ 应用科学：数学及相关领域。
- ◎ 适用领域范围：自然数学、应用数学、高等数学、概率论。
- ◎ 射击比赛等。

5.3.3 正态分布

正态分布描述

正态分布又名高斯分布，以德国数学家卡尔·弗里德里希·高斯的姓冠名，是一个在数学、物理及工程等领域都非常重要的概率分布。由于这个分布函数具有很多非常漂亮的性质，使得其在诸多涉及统计科学、离散科学的领域都有重大的影响力。比如图像处理中最常用的正态分布函数，图 5-14 是正态分布示意图。

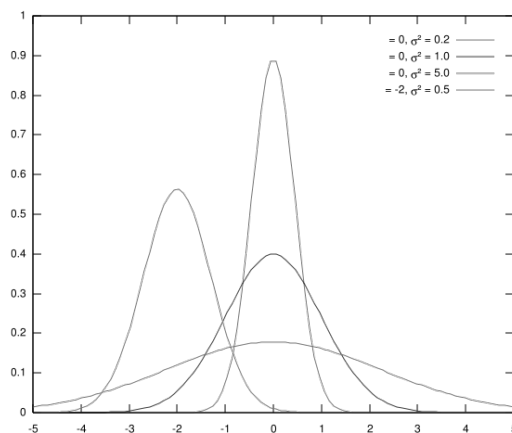


图 5-14 正态分布

若随机变量 X 服从一个位置参数为 μ 、尺度参数为 σ 的概率分布，记为

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

则其概率密度函数为

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

正态分布的数学期望值或期望值 μ 等于位置参数，决定了分布的位置；其方差 σ^2 的平方或标准差 σ 等于尺度参数，决定了分布的幅度。

定义

正态分布概率函数密度曲线可以表示为

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

称服从正态分布，记为 $X \sim N(m, s_2)$ ，其中 μ 为均值， σ 为标准差。标准正态分布令正态分布： $\mu = 0, \sigma = 1$ ，公式简化为

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

正态分布的特点

- ◎ 正态分布函数密度曲线在横轴上方均数处最高。
- ◎ 正态分布函数密度曲线以均数为中心，左右对称。
- ◎ 正态分布函数密度曲线有两个参数，即均数 (μ) 和标准差 (s)。 μ 是位置参数，当 s 固定不变时， μ 越大，曲线沿横轴，越向右移动；反之， μ 越小，则曲线沿横轴，越向左移动。 s 是形状参数，当 μ 固定不变时， s 越大，曲线越平阔； s 越小，曲线越尖峭。通常用 $N(\mu, s)$ 表示均数为 μ 、方差为 s 的正态分布。用 $N(0, 1)$ 表示标准正态分布。
- ◎ 正态分布函数密度曲线下面积的总和为 1。

二项分布的期望

$$E(X) = \mu$$

几何分布的方差

$$\text{Var}(X) = \sigma^2$$

正态概率计算步骤

第一步：首先确定数据是否符合正态分布，确定正态分布的均值和方差。对一些不符合正态分布的数据进行取对数，或者样本重新排列成符合正态分布的标准后，再确定均值和方差。

第二步：标准化（平移，收放）。对一般正态分布进行标准化，标准化的过程为先平移，平移过程用公式表达即 $x - \mu$ ，再对结果进行收放，收放过程即 $\frac{y}{\sigma}$ ，其中 $y = x - \mu$ 。则标准化公式： $Z = \frac{(x - \mu)}{\sigma}$ ；其中 Z 为标准分， x 为随机变量， μ 为均值， σ 为标准差。

第三步：使用概率表。通过标准分，查表（标准正态分布概率表），得到具体的概率。

正态分布的优缺点

- ◎ 优点：对于社会上遇到的大部分问题，其概率分布规律基本都满足正态分布，为了计算某种概率，我们就可以利用正态分布建立数学模型来解决问题。

- ◎ 缺点：无法近似估算符合几何分布的问题，无法精确解决离散数据概率。

应用场景

- ◎ 适用场景：连续型数据或者数据离散性小，数据基本符合正态分布特点，对不符合的数据进行取对数或者样本重新排序达到正态分布特点，有具体的均数（期望）和标准差。
- ◎ 不适合应用场景：数据离散性太大，数据不符合正态分布，通过对数据进行取对数或者重新排序也无法达到正态分布特点，无法得出均数（期望）和标准差。

中心极限定理

正态分布有一个非常重要的性质：在特定条件下，大量统计独立的随机变量的平均值的分布趋于正态分布，这就是中心极限定理。中心极限定理的重要意义在于，根据这一定理的结论，其他概率分布可以用正态分布作为近似。

- ◎ 参数为 n 和 p 的二项分布，在 n 相当大且 p 接近 0.5 时近似于正态分布。近似正态分布平均数为 $\mu = np$ ，且方差为 $\sigma^2 = np(1 - p)$ 。
- ◎ 泊松分布带有参数 λ ，当取样样本数很大时将近似正态分布 λ 。近似正态分布平均数为 $\mu = \lambda$ 且方差为 $\sigma^2 = \lambda$ 。

5.3.4 泊松分布

定义

泊松分布适合描述单位时间内随机事件发生次数的概率分布。如某一服务设施在一定时间内受到服务请求的次数，电话交换机接到呼叫的次数、车站台的候客人数、机器出现的故障数、自然灾害发生的次数、DNA 序列的变异数、放射性原子核的衰变数、激光的光子数分布等。泊松分布的概率质量函数为

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}$$

泊松分布的参数 λ 是单位时间（或单位面积）内随机事件的平均发生率。

计算公式

X 服从参数为 λ 的泊松分布，记为 $X \sim P(\lambda)$ 。单独事件在给定区间随机独立发生，已知事件平均发生数且有限次数，通过以下公式计算：

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}, (k = 0, 1, \dots, n)$$

泊松分布的条件

- (1) 单独事件在给定区间内随机独立地发生，给定区间可以是时间或者空间（一周、一英里）。
- (2) 已知该区间内的事件平均发生次数（发生率），且为有限数值。该事件平均发生次数用 λ 表示。

泊松分布形状特点

- ◎ 不需要一系列试验，描述事件特定区间发生次数。
- ◎ 两个独立的泊松分布相加也符合泊松分布（即 $n > 50$ 且 $p < 0.1$ 时，或 np 近似等于 npq 时）。
- ◎ 特定条件下可以用来近似代替二项分布。

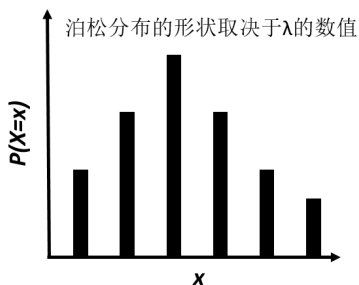


图 5-15 泊松分布特点

泊松分布形状特点：当 λ 小时，分布向右偏斜；当 λ 大时，分布逐渐对称。

优缺点：不需要一系列试验，描述事件特定区间发生的次数时特别适用。另外，一定条件下替换二项分布带来了简便的运算。

泊松分布与二项分布关系

当二项分布 $X \sim B(n, p)$ 的 n 很大而 p 很小时，泊松分布可作为二项分布的近似，其中 λ 为 np 。通常当 $n \geq 10$, $p \leq 0.1$, $np \leq 5$ 时，就可以用泊松公式近似计算， X 可以近似表示 $X \sim P_0(np)$ 。

问题：为什么 n 要足够大， p 要足够小？

因为在分时间窗口的时候有个假设：每个时间窗口最多只有一个乘客到达（时间区间乘客问题）。

二项分布的期望

$$E(X) = \lambda$$

几何分布的方差

$$\text{Var}(X) = \lambda$$

应用范围

- ◎ 应用学科：概率论。
- ◎ 某一服务设施在一定时间内的到达人数，电话交换机接到呼叫的次数，汽车站台的候客人数，机器出现的故障次数，自然灾害发生次数，一块产品的缺陷，显微镜下单位分区内的细菌分布数等。
- ◎ 在交通工程中的应用、非典流行与传播服从泊松分布。
- ◎ 自然现象普遍存在泊松分布现象，主要指大量重复实验中稀有事件发生的次数。

5.4 统计假设检验

定义

假设检验是推论统计中用于检验统计假设的一种方法。而“统计假设”是可通过观察一组随机变量的模型进行检验的科学假说。一旦能估计未知参数，就会希望根据结果对未知的真正参数值做出适当的推论。假设检验的种类包括： t 检验、 Z 检验、卡方检验、 F 检验，等等。

假设检验的过程

假设检验的过程可以用法庭的审理来说明。先想象现在法庭上有一名被告，假设该被告是清白的，而检察官必须要提出足够的证据去证明被告的确有罪。在证明被告有罪前，被告是被假设为清白的。

◎ 假设被告清白的假设，就相当于零假设。

◎ 假设被告有罪的假设，则是备择假设。

而检察官提出的证据，是否足以确定该被告有罪，则要经过检验。这样的检验过程就相当于用 t 检验或 Z 检验去检视研究者所搜集到的统计资料。

检验过程

在统计学的文献中，假设检验发挥了重要作用。假设检验大致有如下步骤：

- (1) 最初研究假设为真相不明。
- (2) 第一步是提出相关的零假设和备择假设。这是很重要的，因为错误陈述假设会导致后面的过程变得混乱。
- (3) 第二步是考虑检验中对样本做出的统计假设。例如，关于独立性的假设或关于观测数据的分布形式的假设。这个步骤也同样重要，因为无效的假设将意味着试验的结果是无效的。
- (4) 决定哪个检测是合适的，并确定相关检验统计量 T 。
- (5) 在零假设下推导检验统计量的分布。在标准情况下应该会得出一个熟知的结果。比如检验统计量可能会符合学生 t -分布或正态分布。
- (6) 选择一个显著性水平 (α)，若低于这个概率阈值，就会拒绝零假设。常用的是 5% 和 1%。
- (7) 根据在零假设成立时的检验统计量 T 分布，找到数值最接近备择假设，且概率为显著性水平 (α) 的区域，此区域称为“拒绝域”，意思是在零假设成立的前提下，落在拒绝域的概率只有 α 。

- (8) 针对检验统计量 T ，根据样本计算其估计值 $tobs$ 。
- (9) 若估计值 $tobs$ 未落在“拒绝域”，则接受零假设。若估计值 $tobs$ 落在“拒绝域”，拒绝零假设，则接受备择假设。

应用实例

淑女品茶是一个有关假设检验的著名例子，费雪的一个女同事声称可以判断在奶茶中，是先加入茶还是先加入牛奶。费雪提议给她八杯奶茶，四杯先加茶，四杯先加牛奶，但随机排列，而女同事要说出这八杯奶茶中，哪些先加牛奶，哪些先加茶，检验统计量是确认正确的次数。零假设是女同事无法判断奶茶中的茶先加入还是牛奶先加入，备择假设为女同事有此能力。若单纯以概率考虑（即女同事没有判断的能力）下，则八杯都正确的概率为 $1/70$ ，约为 1.4%，因此“拒绝域”为八杯的结果都正确。而测试结果为女同事八杯的结果都正确，在统计上是相当显著的结果。

5.5 相关和回归

5.5.1 相关

定义

线性回归示意图如图 5-16 所示。

在概率论和统计学中，相关称为相关系数或关联系数，表示两个随机变量之间线性关系的强度和方向。在统计学中，相关的意义用来衡量两个变量相对于其相互独立的距离。在这个广义的定义下，有许多根据数据特点而定义的用来衡量数据相关的系数。

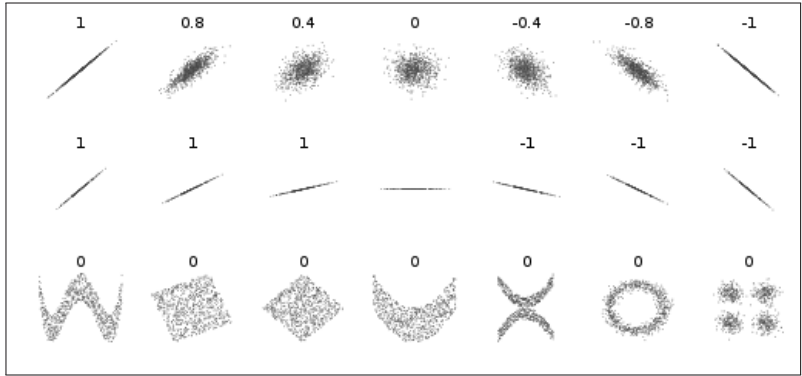


图 5-16 线性回归

相关性

相关性即变量之间的数学关系，通过散点图上的点的独特构成模式，可以识别出散点图上的各种相关性。如果散点图上的点几乎呈直线分布，则相关性为线性。

◎ 正线性相关（见图 5-17）

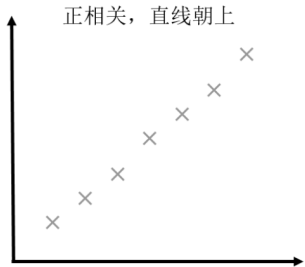


图 5-17 正线性相关

当 x 轴上的低端值对应 y 轴上的低端值，同时 x 轴的高端值对应 y 轴上的高端值且呈直线分布时，为正线性相关。即随着 x 增长， y 也呈现增长趋势。

◎ 负线性相关（见图 5-18）

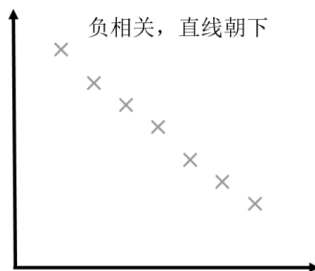


图 5-18 负线性相关

当 x 轴上的低端值对应 y 轴上的高端值，同时 x 轴的高端值对应 y 轴上的低端值且呈直线分布时，为负线性相关。即随着 x 增长， y 呈现下降趋势。

◎ 不相关（见图 5-19）



图 5-19 不相关

如果 x 和 y 呈现一种随机模式，则二者不相关。

5.5.2 回归

最佳拟合线预测

（1）假设最佳拟合线的方程为

$$y = ax + b$$

（2）计算自变量 x 和因变量 y 的均值。

$$\bar{x} = \frac{\sum_i^n x_i}{n}, \bar{y} = \frac{\sum_i^n y_i}{n}$$

(3) 利用最小二乘法回归法求最佳拟合线的斜率。

$$b = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2}$$

(4) 计算最佳拟合线的切距。

$$a = \bar{y} - b\bar{x}$$

(5) 由求得的斜率和切距得出最佳拟合线的方程。

(6) 计算自变量 x 和因变量 y 的标准差。

$$S_x = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n - 1}}, S_y = \sqrt{\frac{\sum_i^n (y_i - \bar{y})^2}{n - 1}}$$

(7) 计算相关系数。

$$r = \frac{b_{s_x}}{s_y}$$

(8) 通过相关系数判断所求最佳拟合线与数据的拟合度，规则如下。

- ⊙ 如果相关系数的绝对值越接近 1，则所求最佳拟合线的拟合度越高，可用于数据预测。
- ⊙ 如果相关系数的绝对值越接近 0，则所求最佳拟合线的拟合度越低，不推荐用于进行预测（预测的结果可能不准确）。

应用案例

案例定义：有一个二变量数据同时给出预计天晴时数和音乐会听众人数（其中：天晴时数表示自变量，听众人数表示因变量），如表 5-4 所示。

表 5-4 音乐会听众人数和天晴时数关系

音乐会听众人数（百人）	天晴时数（小时）
22	1.9
33	2.5
30	3.2
42	3.8
38	4.7

(续表)

49	5.5
42	5.9
55	7.2

如果音乐会当天预计天晴时数可能为 4.3 小时, 请问音乐会听众人数可能会有多少人?

(1) 假设最佳拟合线的方程为

$$y = ax + b$$

(2) 计算天晴时数和听众人数的均值。

$$\bar{x} = \frac{\sum_i^n x_i}{n} = \frac{34.7}{8} = 4.3375, \bar{y} = \frac{\sum_i^n y_i}{n} = \frac{311}{8} = 38.875$$

(3) 利用最小二乘法回归法求最佳拟合线的斜率。

$$b = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2} = \frac{122.53}{23.02} = 5.32$$

(4) 计算最佳拟合线的切距。

$$a = \bar{y} - b\bar{x} = 38.875 - 5.32 \times 4.3375 = 15.8$$

(5) 由求得的斜率和切距得出最佳拟合线的方程。

(6) 计算天晴时数和听众人数的标准差。

$$S_x = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n - 1}} = \sqrt{\frac{23.02}{8 - 1}} = 1.81$$

$$S_y = \sqrt{\frac{\sum_i^n (y_i - \bar{y})^2}{n - 1}} = \sqrt{\frac{780.875}{8 - 1}} = 10.56$$

(7) 计算相关系数。

$$r = \frac{b_{s_x}}{S_y} = \frac{5.32 \times 1.81}{10.56} = 0.91$$

(8) 通过相关系数判断所求最佳拟合线与数据的拟合度并得出预测结果。

由于 r 接近 1，说明音乐会听众人数和预计天晴时数之间有很强的正相关。换句话说，根据已有的数据，利用最佳拟合线预计天晴时数给出了期望音乐会听总人数的合理的良好估计。

当音乐会当天预计天晴时数可能为 4.3 小时，利用最佳拟合线方程，那么就可估计当天音乐会听众人数大约是 3868 人。

线性回归和逻辑回归

在线性回归中，结果（因变量）是连续的。它可以是无限数量的可能值中的任何一个。在逻辑回归中，结果（因变量）只有有限数量的可能值。

例如，如果 X 包含房屋平方英尺的面积，并且 Y 包含这些房屋的相应销售价格，则可以通过线性回归的方法，利用面积来预测房价。虽然售价可能不是实际的，但是有很多可能的值可以选择线性回归模型。

相反，如果想根据大小来预测房屋的出售金额是否超过 20 万美元，将使用逻辑回归。可能的输出是“是的”，房子将卖出超过 20 万美元；否则房子“不会”。

5.5.3 相关和回归的联系

两者区别

回归和相关都是研究两个变量相互关系的分析方法。相关分析研究两个变量之间相关的方向和密切程度。但是相关分析不能指出两个变量相互关系的具体形式，也无法从一个变量的变化来推测另一个变量的变化关系。回归方程则是通过一定的数学方程来反映变量之间相互关系的具体形式，以便从一个已知量来推测另一个未知量。为估算预测提供一个重要的方法。具体区别有：

- (1) 相关分析中变量之间处于平等的地位；回归分析中，因变量处在被解释的地位，自变量用于预测因变量的变化。
- (2) 相关分析中不必确定自变量和因变量，所涉及的变量可以都是随机变量；而回归分析则必须事先确定具有相关关系的变量中，哪个是因变量，哪个是自变量。一般来说，回归分析中因变量是随机变量，而把自变量作为研究时给定的非随机变量。

- (3) 相关分析研究变量之间相关的方向和程度，但相关分析不能根据一个变量的变化来推测另一个变量的变化情况；回归分析是研究变量之间相互关系的具体表现形式，根据变量之间的联系确定一个相关的数学表达式，从而可以从已知量来推测未知量。
- (4) 对两个变量来说，相关分析只能计算出一个相关系数；而回归分析有时可以根据研究目的的不同建立两个不同的回归方程。

两者联系

相关分析与回归分析是广义相关分析的两个阶段，两者有着密切的联系：

- (1) 相关分析是回归分析的基础和前提，回归分析则是相关分析的深入和继续。相关分析需要依靠回归分析来表现变量之间数量相关的具体形式，而回归分析则需要依靠相关分析来表现变量之间数量变化的相关程度。只有当变量之间高度相关时，进行回归分析寻求其相关的具体形式才有意义。如果在没有对变量之间是否相关，以及相关方向和程度做出正确判断之前，就进行回归分析，则很容易造成“虚假回归”。
- (2) 由于相关分析只研究变量之间相关的方向和程度，不能推断变量之间相互关系的具体形式，也无法从一个变量的变化来推测另一个变量的变化情况。因此在具体应用过程中，只有把相关分析和回归分析结合起来，才能达到研究和分析的目的。

第 6 章

语言学

本章导读：本章主要从语音、词汇、语法三个角度对现代汉语进行了一个简单的介绍，在以往传统的语言学教材中一般还有“文字”“修辞”两节内容，因篇幅有限、与全书关联不强，在此就删繁就简，未能给读者一一呈现。需要注意的是，语言学本身是一门十分庞杂的学科，知识体系与研究方法或因语言不同而有区别，或因派别主义不同而有区别。但无论是何种语言，或是何门何派，在进行自然语言处理时我们要面临的永远是一个个真实的语料和具体的语言现象。理论是用来指导实践、拓宽我们研究思路的，究竟最后采用何种理论，这只是一个“白猫黑猫”的问题。

6.1 语音

6.1.1 什么是语音

语音是通过人类发音器官发出的、有意义内容并用于社会交际的声音。它既是语言的物质外壳，又是语义的重要载体。因此，自然界中的虫鸣鸟叫不是语音，甚至我们日常发出的诸如咳嗽、哭笑、打鼾、轻哼等声响，由于其本身并不能传达确定的语义而应用于交际，也不能算作真正的语音。有一个简单的判断标准，在普通话中有字可以表示的音就是语音。“他呼呼大睡”中的“呼呼”模拟了“他”睡觉时发出的声音，是语音（注意：“呼呼”并不就是人类睡觉时实际发出的声音，它是抽象的、符号化后的拟声词）。另外，孩童们在玩闹时用手作枪模拟子弹发射的 [biubiu] 声不是语音，虽然它的表意在这里似乎很明确，但缺乏字形的现状表明，这个音目前还未被大众所认可，不能用于社会交际。

6.1.2 语音的三大属性

物理属性

语音的本质是音波，由发声体振动而产生。因此语音同自然界其他声音一样，拥有音高、音强、音长、音色四种要素。

- ◎ 音高指的是声音的高低，它是由发声体振动的频率（单位 Hz）决定的。单位时间内发声体振动的次数越多，其声音就越高，反之亦然。一般说来，大的、粗的、厚的、长的、松的物体振动慢；小的、细的、薄的、短的、紧的物体振动快。反映到人声上，一般来说成年男性声带长且厚，所以音高低；而儿童与妇女之所以常被认为“嗓子尖”则是缘于他们的声带常短而薄。音高的区别，在汉语里就构成了声调与语调的区别。
- ◎ 音强指的是声音的强弱，它是由发声体的振幅（单位为 dB）决定的。发声体振动时摆动幅度越大，其声音就越强，反之亦然。反映到人声上，一般来说用力越大，肺部呼出气流越强，声带振动越剧烈，声音就越响亮；反之如窃窃私语时，音强就较弱。音强的区别，在汉语里就构成了轻重音的区别。
- ◎ 音长指的是声音的长短，它是由发声体的振动时长决定的。发声体振动时间越长，其声音就越长，反之亦然。音强的区别，在汉语普通话里虽未用来区别意义，但在一些外语和方言中，其区别词义的作用却较为明显。
- ◎ 音色（音质、音品）指的是声音的特色，它是由发声体振动时产生的不同音波波纹的曲折形式决定的。它的影响因素有发音体材质、发音方法及共鸣器的形状。反映到人声上，千人千语就是音色影响声音的最好例子。

以上物理四要素在自然语音中是难以分割的，它们在不同语言或方言中分别起着不同的区别作用。其中，音色是任何语言中最基础也是最重要的区别因素，如“足球”与“篮球”两个词的第一个字“足”[zú]与“篮”[lán]就是音色截然不同的两个语音。另外在汉语普通话中，音高带来的所谓“阴平、阳平、上声、去声”四调与音强所带来的轻重音（如“东西”的两个音[dōng xī]和[dōng xi]）也起着十分重要的区别作用。

生理属性

语音是由人的发音器官发出的，后者根据参与发音过程的先后顺序可分为三个部分：肺、气管、胸腔、横膈膜等呼吸器官，它们主要通过呼出气流为发音提供原动力；喉头、声带等

发声器官，黏附在喉头的软骨上，由两片薄膜构成的声带在气流冲击下发生振动，形成音波；咽腔、鼻腔、口腔处的调音器官，它们将把从声带传出的音波进行加工，从而产生我们人耳听到的各种语音。

社会属性

语言是一套符号系统，语音作为其重要组成部分也是如此。语音的社会性质主要有以下三个方面：

- ◎ 语音与相应语义之间的关系是非必然的，约定俗成的。汉语普通话中的 [dà lù] 可以用来指代陆地，也可以用来指代宽广的道路。不同语言间的音义规定差别则更为明显，同样是苹果，英国人就会说成是 “apple”。
- ◎ 语音具有地域性、民族性。上面说到音义之间的对应关系是人为规定的，但是这种规定如果不能得到一个区域内特定群体的广泛认同，则人与人之间就无法进行有效的交流。如果进一步放任音义关系的随意扩展，那么就有可能产生《圣经》中“巴别塔”般的悲剧。为了避免上述现象的发生，不同民族和地区的人们在地理文化的差异下便产生了成百上千种各具特色的语言。
- ◎ 语音具有系统性。这就意味着不同语言或方言拥有着不同的语音元素以及不同的内部关系。同样是说“牛奶”和“男人”这两个词，北京人和四川人就会有两种不同的发音结果（四川方言存在 [n][l] 不分的情况）。

6.1.3 语音单位

音素

音素是音色角度划分而出的最小语音单位，分为元音、辅音两大类。例如，“看” [kàn] 就可以划分出“k、a、n”三个各有特色的音素。元音音素发音时气流振动声带，在口腔、咽腔等处不受阻碍，辅音相反。元音又叫母音，在现代汉语普通话中共有 10 个，分别是 7 个舌面元音：a[A]、o[]、e[ɤ]、ê[ɛ]、i[i]、u[u]、ü[y]；2 个舌尖元音：-i[]、-i[]；以及一个卷舌元音：er[]。辅音又叫子音，在现代汉语普通话中共有 22 个：b[]p[]m[]f[]d[]t[]n[]l[]g[]k[]h[]j[]q[]x[]zh[]ch[]sh[]r[]z[]c[]s[]ng[]，其中前 21 个都可以作为声母，而 ng[] 只可做韵尾。

音节

音节是由音素构成的，在交谈时自然感到的语音单位。一般情况下一个汉字就表示一个音节，儿化音节诸如“花儿”[huār]等则是用两个汉字代表一个音节的特例。

声母、韵母、声调

- ◎ 声母在普通话中共 21 个，是主要由辅音构成的音节前段。例如，在“表”[biǎo] 这个音节里，辅音 b 就是它的声母。特别注意，有些音节不以辅音开头，我们习惯上将元音前头那部分看作零，叫作“零声母”。例如，“乌”[wū] 开头没有辅音，就算是零声母音节。而 ū 前的 w 在拼音书写时既是一种隔音符号，也提示了 ū 前面实际发音中半元音 [w] 的存在。因此，“零声母”未必就是零，但其不以辅音开头的特点却是肯定的。
- ◎ 韵母在普通话中共 39 个，是由元音或元音加辅音构成的音节后段，可分为韵头、韵腹、韵尾三个轻重长短不一的部分。韵头只有 i、u、ü 三个；韵腹是韵母的主干，是其必不可少的部分且声音在三部分中最为清晰响亮；韵尾只能由元音 i、u 和鼻辅音 n、ng 四个充当。来看一例，在“壮”[zhuàng] 这个音节里，辅音“zh”是它的声母，“u”是它的韵头，“a”是它最为响亮的韵腹，“ng”是它的后鼻韵尾。另外请注意，并不是所有音节都有声母、韵头、韵尾，如上文所举“乌”[wū] 的例子，它就只有韵腹 u。
- ◎ 声调贯穿于一个音节、具有区别意义作用的音高变化。调类是声调的种类，普通话中共有四种：阴平、阳平、上声和去声，它们的调值分别为 55、35、214 和 51（此处采用赵元任的“五度标记法”）。

音位

音位是一个具体的语言系统中能够区别意义的最小语音单位，也就是按语音的辨义作用归纳出的音类。在具体的某种语言或方言里，人们可以发出的音素很多，但音位的数量却是有限的。例如，“妈”[m] 与“怕”[p] 的声母 [m][p] 就是两个音位，它们区别意义；而如“巴”[b] “班”[bn] “帮”[bng] 中的 α 实际上分别读作 [A]、[a]、[ɑ]（称作“音位变体”），是三个不同的元音，只不过因为三者的音色不会因读错而导致人们的误解，因此我们常把它们归纳到一个音位里去。

6.1.4 记音符号

历史上汉语的注音方法很多，诸如反切法、“注音符”（至今台湾等地区还在使用）标记等。现在国内通行的是 1958 年审议通过的《汉语拼音方案》，相信绝大多数读者在小学语

文课堂上就学习了相应的知识，在此就不再赘述。另外需要重点介绍的，也是目前国际上通用的记音符号就是国际音标（International Phonetic Alphabet，简称 IPA）。它采用一符一音原则，大部分符号采用拉丁小写字母及其变体（如倒写、反写、合写、添加附加符号等）。至于国际音标里部分普通话中没有的符号发音敬请有兴趣的读者自行上网检索，这方面的语音资料网上很多，尤以赵元任和瞿霭堂两位先生的发音最为经典，可供参阅。

6.1.5 共时语流音变

变调

- ◎ 轻声。四声在一定条件下念得比原调短而弱的特殊音变现象。具有区别词义、区别词性等作用，如曾举过的“东西”，又如“对头”。
- ◎ 上声的变调。两个上声相连，前一个调值从 214 变成 35（上上相连变阳平），例如，“蒙古”；上声后接其余三调或轻声，这个上声变为半上 21，例如“好的”；三个及以上上声音节相连的变读还请读者自行尝试总结规律，当然也可参阅相应的现代汉语教材，尤以黄伯荣、廖序东主编的《现代汉语》（上下册）最为经典。
- ◎ “一、不的变调”。“一、不”在单念、词句末及“一”用在序数中时读作原调；去声前二者一律变为上声 35，例如，“一个”“不去”；在非去声前，“一、不”皆读去声 51，例如，“一天”“不开心”；“一、不”嵌在相同的动词、形容词间皆读作轻声，例如，“瞅一瞅”“行不行”；“不”处于可能补语中变读轻声 bu，例如，“起不来”。
- ◎ “七、八的变调”。“七、八”在去声前可变为阳平 35，也可不变。例如，“七岁”“八路”。
- ◎ 其他变调。除上述三类外，还有一些变调可供有兴趣的读者自行总结，此处举上几例，如“远远儿的”（“AA 儿的”式，第二个“远”常读阴平 55）、“乱蓬蓬”（“ABB”式，“蓬蓬”常读阴平 55）。

儿化

一个音节中韵母带上卷舌音色的特殊音变现象。与轻声一样，儿化也具有区别词义、词性的作用，如“头”“头儿”，“盖”“盖儿”；此外，儿化后的字词常带有细小、轻松、喜爱、亲切的感情色彩，如“小猫儿”“脸蛋儿”。

“啊”的音变语气词：

“啊”受前字末尾音素的影响往往产生音变，共有6种，举例如下：“吃呀喝呀”“困难哪”“还爬楼哇”（此处记“啊”音变后的字）。此外几种读者可自行参阅相关教材。

6.2 词汇

6.2.1 什么是词汇

词汇又叫语汇，是某种特定语言里所有词和固定短语的总和。

6.2.2 词汇单位

语素

构词的单位，语言中最小的音义结合体。例如，“笔”就是一个语素，它的语音形式是[sh]，它的词汇意义是“写字、画图的工具”，语法意义是“名词、量词”及相关的语法作用。

从音节角度看，语素有单复音节之分。现代汉语中绝大部分是由一个音节构成的语素，如“天、地、人、山、海、的、啥、啊”等；而复音节语素是指由两个及其以上音节构成的语素，如“蝴蝶、鸳鸯、玻璃、抠门、布宜诺斯艾利斯”等；另外还有小于一个音节的特殊语素，如“鸟儿”的“儿”字因是儿化标记，只算半个语素。

有一个简单的判别语素的方法——“替代法”，即用一个新语素替代尚未确定为语素的语言单位，其间还要注意替代前后另外一个未被替代的语素，其意义不能发生变化。以“抠门”为例，该词虽可被替换为“铁门”“抠鼻子”等，然而“抠门”合在一起表示的“小气、不大方”义，并非拆开后各自的“用手指或细小的东西挖”“建筑物的出入口”等义简单组合就可得到。对于复音节语素，其语义常常是确定的，其组成元素也往往是不容随意更改的。从表意虚实看，语素还分为实语素和虚语素。前者有具体的词汇意义，如“天、人、喜”等；后者则只表示抽象的语法意义，如“吗、的、老（公）”等。从构词能力看，还有成词语素与不成词语素之分。前者又叫自由语素，指既能单独成词又可以与其他语素组合成词的语素，如“马跑了。”这句话中的成词语素“马”既可以单用，也可以组合成“白马、骏马、老马”等。而不成词语素则是指不能独立成词，必须和别的语素组合在一起的语素，也称黏着语

素。不成词语素又可以分为两类，一类位置自由、承担了所成词部分乃至全部的基本意义，如“卉、健、民、丽”等，它与成词语素合在一起被称作词根，是词义的承担者；另一类位置固定只表示一些附加语义，被称作词缀，依所在位置被进一步分为前缀、中缀、后缀，如“者、第、化（白热化）、里（稀里哗啦）”等。

词

词由语素构成，是语言中最小的、能够独立运用的音义结合体。例如，“打水”可被拓展成“打热水”，因此是短语，不是最小的词。再如上文的不成词语素“卉”，生活中我们从不单说这个字，相反语义相近的“花”就可以，因此也不能算作能够独立运用的词。

固定短语

词语之间的固定搭配，一般不能随意增减或改换其中的元素；与之相对应的是自由短语，它可以依据表达的需要临时组合词语。我们常说的“短语”其实是后一种自由短语，它十分能产。例如，“看报、看电视、看球赛”等。固定短语又可分为专名与熟语两类。前者以企事业单位名居多，如“四川大学”“全国计算语言学学术会议”“国营长虹机器厂”等；后者包括成语（“七上八下、东倒西歪”等）、惯用语（“炒鱿鱼、穿小鞋”等）、歇后语（“哑巴吃黄连——有苦说不出”等）。

缩略语

语言中被压缩或省略的词语，主要有简称、数词略语两种形式。前者如“彩色电视机——彩电”、“工人联合会——工会”、“马萨诸塞州——麻省”等；后者如“三好学生”“五讲四美”等。

6.2.3 词的构造

单纯词：只由一个语素构成的词。

最简单的单纯词主要是单音节词，如“花、山、人、好、吗”等，另有一部分是多音节的，主要有以下几类。

- ◎ 联绵词：由两个不同音节连缀而成，不能拆分为两个语素解释其义的单纯词，又具体分为三种。
 - (1) 双声：指两个音节声母相同的联绵词，如“琵琶、参差、蹊跷、仓促”等。
 - (2) 叠韵：指两个音节韵母（或只是韵腹韵尾）相同的联绵词，如“骆驼、琢磨、叮咛”等。
 - (3) 其他：指两个音节声韵母皆不同的联绵词，如“鸳鸯、峥嵘、马虎”等。
- ◎ 叠音词：指两个相同音节重叠构成的词，其中任何一个音节都只有音而没有义（或没有组合后的新义），它们只有合在一起构成一个复音节语素才有意义，如“孜孜、娘娘（单说表母亲，合说后意义发生变化）、草草、狒狒”等。
- ◎ 拟声词：指用来模拟声音的词，如“叮叮当、扑哧、叽里咕噜”等。
- ◎ 音译外来词：指对音翻译外民族语言而得的词，其中任何一个音节都只表音而不表原义，如“麦克风、朱古力”等。

合成词：由两个及以上语素构成的词，主要由以下几类。

- ◎ 复合词：指由两个及以上不同词根构成的合成词，从词根间关系看，又可细分为五类。
 - (1) 联合式：也称并列式，词根语素之间地位平等，意义相同、相近、相关或相反，例如，“建造、车马、迟早、忘记”等。其中“忘记”一词，实际承担语义的只有“忘”这一词根语素，另一词根并不表意，我们把这类词也称作“偏义词”。
 - (2) 偏正式：前一词根修饰、限制后一词根。例如，“莲子、南瓜、嫩绿、风行”等。其中前两个例子中心词根是名词性语素，也称为定中式；后两个例子中心词根是动词、形容词性语素，也称状中式。
 - (3) 述宾式：前一词根表示动作行为，后一词根表示前者所支配关涉的对象。例如，“司机、主席、化石”等。
 - (4) 主谓式：前一词根表示被陈述主体，后一词根陈述前一词根，也称“陈述式”。例如，“月食、自强、胆小”等。
 - (5) 补充式：后一词根补充说明前一词根的动作结果、事物单位等，例如，“打倒、课本、改正”等。

此外还有介宾式（如“从前、当”等）、连动式（如“走访、病故”等）、兼语式（如“讨（人）厌，请（人）教”）等，这三者因数量较少，在有的论著中前者被归入述宾式，而后二者则被归入联合式中。

- ◎ 重叠词：由相同词根语素重叠构成的合成词，例如，“妹妹、弟弟、常常”等。
- ◎ 附加词：由词根词缀构成的合成词，“前缀+词根”的叫前加式，“词根+后缀”的叫后加式，另有加中缀的情况。例如，“老虎、老公、石头、馒头、稀里糊涂、古里古怪”等。

另外，“词汇单位”一节所讲之“缩略语”中有部分词汇诸如“北大、社科”等因组成元素间关系密切，社会认可度高、使用广泛，有时我们也将其作是一种特殊的合成词。现代汉语中双语素合成词是占绝大多数的，当然还有更多数量的语素构成的词，篇幅有限难以展开介绍，但还请读者在分析这些词时一定要留意它们内部的多层性。如“老虎机”中外层关系“老虎”修饰“机”，属偏正式；内层关系“老”又做“虎”的前缀，属附加式。

6.2.4 词义及其分类

什么是词义

词义是词的意义，广义上包括词汇意义和语法意，本章中所讲“词义”通常指词汇意义。

词义的分类

- ◎ 理性义：也称概念义，是词义中反应事物概念的意义部分。例如，“货币：政府法律规定强制使用，可充当交易的媒介、价值的标准、记账的单位及延期支付的工具”“飞船：运送东西的飞行器”等。一般来说，词典里对各条词目的解释就是该词的理性义。另外，理性义还可以分为通用义与专门义。同样是“水”，生活中人们只会把它解释为“一种无色、无臭、透明的液体”，而在化学实验室里“水”就变成了“氢、氧两种元素组成的无机物”。
- ◎ 色彩义：也称附属义，是附着在理性义之上旨在表达评价、形象等内容的词义部分。可细分为以下几类。
 - (1) 感情色彩有褒义、贬义、中性词之分，例如，“英雄、伟人、汉奸、抠门、大海、学习”等。
 - (2) 语体色彩有书面语和口语之分，例如，“进食——吃饭，交谈——唠嗑”等。
 - (3) 形象色彩能使人产生关于事物形态、颜色、声音等具体形象联想的词义部分。例如，“蝴蝶兰、迎客松、鹅黄、啦啦队”等。

- (4) 文化义一些词出自经典、诗文、民间传说等，本身就具有丰富的历史文化内涵，引人联想。例如，“松龟——长寿，莲花——洁净，乌鸦——噩讯”等。

6.2.5 义项与义素

义项

词的理性意义的分项说明。例如，“经过”就有两个义项：一是“通过”，二是“经历的过程”。一个词若同时拥有几个义项，那其中必有一个义项是最为基本、常用的，我们把它称作基本义；其他的一些义项一般都是由它直接或间接转变而来的，我们称之为转义。根据转义产生的方式，又可以细分为两种：引申义与比喻义。前者是基本义经过推演而形成的意义，如“锤”基本义是“敲打物件的器具”，后来也引申为“用锤敲打”这一动作；后者是基本义被用来比喻为另一事物后逐渐固定下来的新意，如“手足”原指手脚，后也用它来比喻“兄弟”。

根据词义项的多少我们可以将其分为单义词和多义词。值得注意的是，日常生活中我们很容易把多义词与一些特殊的同音词混淆在一起。多义词就是有两个及以上义项的词，而同音词则是语音相同而意义毫无联系的一组词。试看一对例子：“广”是一个多义词，既可以在“宽广”中指“面积、范围宽阔”；又能在“广开言路”中作动词，表“扩大、扩充”义。而“花”则是由一组同音词构成的，一组以“花朵”为基本义，另一组如“花钱、花销”等，则是词义与前者毫不相干的另一组词。

义素

和语音学部分所讲的“音素”“音位”类似，词义部分也有义素与义位两组概念。前者是最小的不能独立运用的意义单位，也称词的语义特征；后者则是最小的能独立运用的意义单位，略等于之前所讲的义项。有了这样一组概念，接下来我们就可以使用一种名为“义素分析法”的手段来深入分析词的内部意义构成，举例如下。

父亲——[+ 男性 + 长辈 + 血亲] 伯父——[+ 男性 + 长辈 - 血亲]

妹妹——[+ 女性 - 长辈 + 血亲] 姨妈——[+ 女性 + 长辈 - 血亲]

要注意，只有相关的词才可以进行义素分析，反之如“大海”与“狗”，因二者词义相差过大而很难总结出较为短小的几条区别特征。另外，义素的选取也尤为重要，要仔细分析出一组可以区别目标词的特征，然后用“[]”将其标注出来，在前面用“+”“-”号进行分析。

6.2.6 语义场

物理学上把物理量在空间中某个特定区域内的分布称为场，有温度场、引力场、电场、磁场等。语义学中也引入了“场”的概念，从而形成了语义场，它是根据词义的共同特点或关系划分出来的类。同场词有着共同的语义区别特征，同时其各个义素前正负号的差异又将它们在同一个语义场中做出区别。

例如：
词共同义素区别义素

	运动项目	陆上项目	评分	计时
游泳	+	-	-	+
跳水	+	-	+	-
体操	+	+	+	-
跨栏	+	+	-	+

有些读者在阅读到上文“只有相关的词才可以进行义素分析”时可能会有些疑惑，不知具体什么样的联系才可以叫作“相关”。在引入语义场概念后，我们就可以对这个问题做出一定程度上的解答——只要词在同一语义场内，就可以认为其具有相关性，从而进行义素分析。那么语义场到底有哪些呢？根据场中各成员间的关系不同，语义场可以粗分为以下几类。

- (1)类属义场 场内成员同属于一个更高层的类，如“诗歌 散文 小说”都属于文学体裁类，其中“诗歌”“散文”等我们称其为下位词，“文学体裁”我们称其为上位词。上下位的概念是相对的、不断转换的，如“诗歌”就可以进一步细分为“古诗、近体诗、现代诗歌”等。
- (2)顺序义场 场内成员之间存在某种顺序排列，如“状元 榜眼 探花”“立春 雨水 惊蛰”等。
- (3)同义义场 场内成员意义相同或相近，如“高兴 开心 喜悦”“集合 汇合 聚集”等。

(4)反义义场 与同义义场相反，场内成员意义相反或相对，如“高贵 卑贱”“干净 邋遢”等。

(5)其他关系义场 除上述义场外，仍有部分语义关系无法得以充分归纳，所以将其单列一项，一齐算入“其他关系义场”。例如：“父母 子女”“前边 后边”等。

6.2.7 词汇的构成

现代汉语词汇分为基本词汇与一般词汇两类。前者具有稳固性、能产性、全民通用性等特征，是机器翻译训练集的理想词源；后者包括古语词、方言词、外来词、行业语、熟语及新兴的网络用语等，这些词数量多、适用范围窄、更新速度快，是专语语料库的主要面向对象，但并不适合作为机器自动学习的词汇语法教科书。

6.3 语法

6.3.1 什么是语法

语法

语法是语言三要素之一（另有语音、语义），专指一种语言中语素、词、短语和句子等有意义的语言单位由小到大组合所依据的规则。在术语层面，“语法”除了指代上述的语法规律，还可以指代研究这些语法规律及其系统的科学——语法学。与语音和词汇不同的是，语法一般比较抽象（抽象性），变化相对缓慢（稳固性），因地区民族不同而存在着明显差异（民族地域性）。语法单位主要有四级，从小到大依次是：语素、词、短语、句子。前两级我们已在之前的篇章中有所涉及，它们在语法中的具体组合规则及新出现的后两级我们将在随后一一予以系统性介绍。

句法成分

即句法结构的组成成分，根据成分间存在的陈述、支配等关系可细分为五组：主语和谓语、述语和宾语、定语和中心语、状语和中心语、中心语和补语。主语是被陈述对象；而谓

语则是用来陈述主语的，二者是陈述关系。例如：“她的新发型非常漂亮”，大主语是“她的新发型”，大谓语是“非常漂亮”。述语又叫动语，表示发生的动作行为；而宾语表客观事物，二者之间是支配、涉及关系。例如：“小明吃完了一大碗饭”，大动语是“吃完了”，大宾语是“一大碗饭”。

修饰语位于中心语之前，用来描写或限制中心语。根据后接中心语的性质又分为定语和状语两种。定语修饰名词性短语里的中心语，而状语则修饰谓词（动词、形容词）性短语里的中心语。在“她的新发型非常漂亮”中，“她的”和“新”作定语修饰中心语“发型”；“非常”作状语修饰中心语“漂亮”。补语是跟在谓词性短语里的中心语后面的补充成分，起补充说明的作用。例如：“今天超市里的苹果便宜得很”中的“很”就作为补语来补充说明中心语“便宜”的程度。此外还有一种独立于八大配对成分之外的特殊句子成分——独立语。它不与句中其他成分产生结构关系，只是出于语用或表达的需要，在句中起特定的表意功能。可分为插入语、称呼语、感叹语、拟声语四类，例如：“事情已然如此，你说，我还能怎么办？”“老王，你去哪儿？”“啊，我马上就来。”“咚咚咚，响起了敲门声。”。

6.3.2 词类

词类是词在语法性质方面划分出的类别，主要依据三个标准：语法功能、形态和意义。其中尤以语法功能标准最为重要，它主要指词的组合能力及充当句法成分的能力。在三条标准的指导下，词可先粗分为两类：实词与虚词。前者可以单独充当句法成分，意义较为具体；而后者不能充当句法成分，只能伴随实词发挥语法意义。二者还可进一步细分，具体如下。

实词

◎ 名词——表示人、事、物、时、地等的名称，又分：

（1）专有名词——巴金、美国、社会主义。

（2）普通名词——母亲（个体名词）、大众（集合名词）、经济（抽象名词）、烟花（物质名词）。

◎ 时间名词：盛夏、清晨、曾经。

◎ 处所名词：周围、城郊、厨房（上海、成都等地名既是专有名词又是处所名词）。

- ◎ 方位名词：上、左、东、后面、以上。语法特征：常作主语、宾语和定语，一般不作状语（时间名词例外）；一般可被数量短语修饰，却不能加副词“不”；大多能跟在介词后头构成介词短语；一般不能重叠；单复数同形。

◎ 动词——表示动作行为、心理活动及存现等，又分：

- (1) 动作动词——听、说、读、写、打击、革新。
- (2) 心理动词——爱、恨、害怕、焦虑、希冀。
- (3) 存（变）现动词——有、在、灭亡、变化、发展。
- (4) 使令动词——叫、让、给、请、令、要求。
- (5) 判断动词——是、称、叫、等于。
- (6) 能愿动词——能、会、敢、要、可以、应该。
- (7) 趋向动词——去、来、上、过、出、进去、起开。
- (8) 形式动词——进行、给予、加以。

语法特征：大多可作谓语（中心语）、动语；能被否定副词“不、没、没有”修饰；除少数心理动词和部分能愿动词外，一般前面不能加程度副词；多数可以后接“着、了、过”；有些动作动词可以重叠，且表意发生一定变化（表短促、尝试、轻松等义）。

◎ 形容词——表示事物的形状、性质和状态等，又分：

- (1) 性质形容词——好、英勇、聪慧、大、高、瘦弱。
- (2) 状态形容词——冷清、乌黑、干巴巴、滑不溜秋。

语法特征：常作定语和谓语（中心语）；不带宾语（部分兼类词除外，如“严肃纪律”中的“严肃”既可作形容词又可作动词）；多数可被程度副词“很、太”等修饰；部分可重叠，且重叠后意义常趋向加深或适中，不能再被“很”修饰；单音形容词可附加叠音词缀或其他词缀（如“硬邦邦”，也不能再加“很”修饰）。

◎ 区别词——一种特殊的形容词（非谓形容词），表示事物的区别性特征，往往成对出现。例如，“男：女、土：洋、国营：民营、小号：中号：大号”等。

语法特征：单用只能作定语，多数可以扩充为“的”字短语（如“假的、坏的”等）；组成联合短语（如“的”字短语）后可充当主谓宾（如“我要中杯”）；单用时否定需用“非”而不能“不”。

◎ 数词——分基数词和序数词以分别表示数目和次序。前者包括“零”至“九”共10个系数词，以及“十”到“兆”共6个位数词；后者一般则是在基数前加“第、初”等词构成，当然也有用天干地支、拉丁字母等表次序的情况。

语法特征：常需和量词组合成数量短语入句，充当定状补；“俩”、“仨”可以看作“两个”“三个”的合体词，后不能再加“个”，其意义与功能和数量短语同。

◎ 量词——又称“单位词”，表示计算单位，又分：

- (1) 名量词——用以计算人或事物的数量，有个体量词（“根、条、把”等）、集体量词（“群、批、副”等）、度量词（“寸、升、吨”等）之分。这三种也叫“专用名量词”，另有从名词、动词处借用来的名量词，如“一瓶水、一抔土”等。
- (2) 动量词——用以计算动作次数多少及持续时间长短，如“下、次、场、番”等。另有借用的动量词，如“噉一嗓子、打一拳、算了一算”等。

语法特征：常位于名词和数词之间，数量短语可作为定状补宾；许多单音节量词可以重叠，且重叠后意义常趋向每一、逐一、多等义，不能再作状语、宾语；有时也可单独入句，但常常是对数词“一”的省略（如“我有个朋友”等）；量词与名词的搭配关系并不固定，随方言习惯而各有差异。

◎ 副词——表示范围、程度、时间等义，常用以修饰动词、形容词性词语，又分：

- (1) 表示范围——全、都、皆、均、总、共、齐、就、只、单、光。
- (2) 表示程度——很、非常、极、最、十分、太、越发。
- (3) 表示时频——立即、曾经、刚刚、一向、再次、偶尔。
- (4) 表示肯否——必须、一定、别、未、莫、没有、勿。
- (5) 表示语气——竟然、岂非、偏、难怪、未免、只好。
- (6) 表示处所——到处、处处、随处、四处。
- (7) 表示关联——遂、就、再、又、仍。
- (8) 表示情态方式——匆匆、一味、亲自、暗自、随意。

语法特征：皆可作状语，例如，“极”“很”等还可作补语；一般不单独成句，只有“别、不、当然、何苦”等可以单用于省略句；部分副词可以起关联作用（如“越跑越快、又蹦又跳”等）。

◎ 代词——起代替、指示作用的词，语法功能与所代指语言单位大体相当，又分：

- (1) 代名词——有一般代名词（包括人称、疑问、指示）、处所代词、时间代词、数量代共4种，如“我、什么、这、哪里、这会儿、多少”等。
- (2) 代谓词——怎么、这样。
- (3) 代副词——多么、那么。

(4) 其他指示代词——各、每、旁的、其他。

语法特征：一般单用，不被别的词所修饰；使用灵活，有任指、虚指等用法（如“明天吃什么呢”“多少钱他都不卖”等）。

◎ 拟声词——也称“象声词”，用以模拟事物的声音，如“轰、哗、叮咚、呼啦啦”等。

语法特征：常在句中作为状语，有时加“地”、“一声”等（有人把能加上“一声”的称作“单纯拟声词”，不能的称作“合成拟声词”）；还可作为定谓补，也能单独成句。

◎ 叹词——表示感叹、应答或呼唤的词，如“唉、哎、啊、喂、哦、嗯”等。

语法特征：常作句子独立语，也可单独成句。

虚词

◎ 介词——也称“前置词”，常在实词、短语前组成介词短语，用以修饰或补充谓词性词语。可分为：

(1) 表示施受关系——让、被、叫、把、给。

(2) 表示原因、目的——因为、为了、由于。

(3) 表示方式——据、按、照、依、靠、以。

(4) 表示关涉对象——和、跟、同、比、对、关于。

(5) 表示时间、处所、方向——自、从、往、到、至、趁着、当着。

语法特征：主要作状语，少数也能作补语、定语；多由动词虚化而来（如“比、给、叫”等），有些介词还处于过渡阶段，判断标准是介词不可重叠、不能单独作谓语（中心语）、不能加动态助词“着了过”。

◎ 连词——也称“连接词”，用以连接词、短语、分句和句子等成分并表示其间逻辑语义关系的虚词，又分：

(1) 主要连接词和短语——和、跟、同、与、及、或。

(2) 主要连接词语或分句——而、并、且、或者。

(3) 主要连接复句中的分句——虽然、然而、与其、因此、只要。

语法特征：不能单独充当句子成分，需连接词语才可表达语法意义；常与副词配合成关联词语；有些特殊兼类词（如“和、同、跟”等）既是介词也是连词，需加以区分。

◎ 助词——附着在实词、短语或句子前后表示一定的结构关系或语法意义，又分：

(1) 结构助词——的、地、得、之（定语后接“的”、状语后接“地”、补语前加“得”）

- (2) 动态助词——着、了、过。
- (3) 尝试助词——看（轻声）。
- (4) 时间助词——来着、的（如“饭后是我洗的碗”等）。
- (5) 约数助词——把、来、多、左右。
- (6) 比况助词——似的、一般、一样。
- (7) 其他助词——所、连、给、等、们。

语法特征：必须跟在别的词语前后，且后接“的”都读轻声，前加“的”都读原调。

◎ 语气词也称“语（气）助词”，主要附在句末或句中停顿处表示语气，念作轻声。可分为：

- (1) 陈述语气词——的、了、吧、啊、呢、嘛、嘞。
- (2) 疑问语气词——吗、吧、么、呢。
- (3) 祈使语气词——吧、了、啊。
- (4) 感叹语气词——啊。

语法特征：语气词常跟语调一起表达语气，因此同一个语气词可能会因语调不同而出现在好几种不同的语气中（如“啊”）；语气词可以连用，并具有一定的层次性（第一层为“的”，第二层为“了”，第三层为“啊、吗、吧、呢”），一般全局的基本语气由最后一个语气词决定。

6.3.3 短语

什么是短语

短语也叫“词组”，是词和词按照一定的语法规则和语义搭配关系组合起来的，没有句调的语言单位。词组成短语的主要语法手段有语序和虚词两类，前者主要体现在词组的直接组合中，后者主要体现在间接组合里，例如，“努力工作/工作努力、老人与海/老人的狗”等。如何区别词和短语呢？一个简单的方法就是看这个语言单位是否可以插入其他成分进行扩展而自身意义还不发生较大的变化。例如，“黑板”原指一种教具，若是强行拓展为“黑色的板”，就会变成一种泛指概念；而网络热词“心累”中间加入程度副词拓展为“心好累”，词义本身并无改变，只是程度加深了。

短语的类型

◎ 组合分类

- (1) 主谓短语：前主后谓，说明主语是什么或怎么样，二者间是陈述说明关系。例如：时光 || 飞逝（名 || 动）、空气 || 清新（名 || 形）、明天 || 周二（名 || 名）、他的话 || 我不相信（名 || 主 || 谓）。
- (2) 动宾短语：前动后宾，表示做什么、有什么、是什么，二者间是支配、涉及关系。例如：提高 | 免疫力（动 || 名）、喜欢 | 热闹（动 || 形）、开始 | 学习（动 || 动）、喜欢 | 你（动 || 代）、买 | 两袋（动 || 数量短语）。
- (3) 偏正短语：由修饰语和中心语组成，二者间是修饰与被修饰的关系，又分为定中（有时以结构助词“的”为标志）、状中两类（有时以结构助词“地”为标志）。例如：海上钢琴师、雨果的秘密、爱乐之城、海边的曼彻斯特（定中）、不停地说、电话联系、这么做、一米长（状中）。
- (4) 中补短语：也称补充短语，由两个谓词组成，前为中心语后为补语，二者间是补充说明关系，有时补语前有结构助词“得”作为标志。例如：讲得（动 · 形）、学（动 · 动）、跑了（动 · 数量短语）、高兴了（动 · 副）。
- (5) 联合短语：由语法地位平等的两个及其以上部分组成，其间是联合关系（可细分为并列、选择、递进等关系），有时用“与、和、或”等词连接。例如：罗密欧与朱丽叶（并列）、生存还是毁灭（选择）、讨论并通过（递进）。
- (6) 其他短语：除了以上五种基本短语，另有连谓（多个谓词性成分连用，如“外出见面”）、兼语（动宾和主谓短语套用，如“祝你幸福”）、同位（前后两项不同词语指向同一事物，如“美国总统特朗普”）、方位（主要表示处所、范围或时间，如“放学后”）、量词（分数量和指量，如“一根、哪款”）、介词（如“[为中华崛起]而读书”）、助词（分“的”字、“所”字等，如“好吃的、傻瓜似的、所了解”）等短语，在此不再一一赘述，感兴趣的读者可以自行查阅相关书籍。

◎ 功能分类，依照短语的句法功能相当于那类词而划分为：

- (1) 名词性短语，包括主谓、偏正、联合、同位、量词、方位、“的”字短语。
- (2) 谓词性短语，分为动词性短语和形容词性短语，包括主谓、偏正、中补、联合等。
- (3) 加词性短语，包括偏正短语（如“高质量”）和介词短语（如“向北走”）。

短语的歧义

我们把意义单一的短语称作“单义短语”，把含有两个及其以上意义的短语称作“多义短语”。对于后者，人们在日常对话或篇章理解中常可通过背景信息或上下文进行语义排歧，然而当计算机处理这些字面上完全相同的短语时，事情就没那么简单了。理清歧义产生背后的原因，有助于我们对目前的机器理解机制进行改进。

◎ 结构关系不同产生的歧义

A. 共享-单车，动宾，动宾短语（相当于“共享什么”）

B. 共享-单车，定中，偏正短语（相当于“什么单车”）

◎ 语义关系不同产生的歧义

A. 鸡-不-吃-了，主-谓-状-中，主谓短语（相当于“不吃鸡了”）

B. 鸡-不吃了，主-谓状中，主谓短语（相当于“鸡不吃食了”）

◎ 结构、语义关系皆不同产生的歧义

A. 咬-死-了-猎人-的-狗，动-谓-中-补-定-中，动宾短语（相当于“猎人的狗被咬死了”）

B. 咬-死-了-猎人-的-狗，定-中-动-宾，偏正短语（相当于“狗咬死了猎人”）

以上分析短语时用到的框式图解方法称作“层次分析法”，它要求分析到词，层层二分（兼语、联合与连动例外），是国内传统现代汉语研究常用来分析各级语法单位的切分工具。层次分析法先是切分，再是定性；既可以由小到大（组合法），也可以由大到小（切分法）。

此外，西方语言学家乔姆斯基还提出了一种“X-bar”理论用来分析句法结构，其形式正是我们常在外文自然语言处理教材中看到的二叉树。应用这套理论，上文提到的三种例外也全部可以进行二分表示，具体内容请有兴趣的读者自行参阅原著 *Lecture on Government Binding*（1981）或外文语言学教材如 *An Introduction to Language*（Eighth Edition）等进行学习。

6.3.4 单句

什么是单句

句子是用以实际交际的基本语言单位，带有一定语调并表达相对完整的意思。而单句则是由短语或词构成的句子，在句子结构上与“复句”相对。

单句的分类

单句大体有两种分类角度：根据句法成分的搭配格局分出的结构类，也叫句型；根据整个句子的语气语调分出的语气类，也叫句类。具体如下。

◎ 句型

- (1) 主谓句：由“主语+谓语”构成的单句，根据谓语核心的词性还可分为三小类：动词性谓语句、形容词性谓语句和名词性谓语句。例如：“故乡的月光不知让多少人魂牵梦绕。”（动谓）、“这块石头硬邦邦的。”（形谓）、“明天礼拜天。”（名谓）。
- (2) 非主谓句：主谓句的补集，常由单个词或短语组成，分不出主谓语。主要分为：名词性非主谓句、动词性非主谓句、形容词性非主谓句、叹词句和拟声词句几类。例如：“妖怪啊！”（名）、“加油！”（动）、“妙极了！”（形）、“哎哟！”（叹）、“砰！”（拟）。

◎ 句类

- (1) 陈述句：给出一定事实信息并带有陈述语气、语调的句子。常用平调、平而略降调，根据陈述内容的不同还可分为叙述句、描写句、判断句三类。例如，“我下班回家了。”（叙述）、“同志们的工作热情都十分高涨。”（描写）、“你就是我失散多年的亲人呐！”（判断）。
- (2) 疑问句：用提问索取相关信息并带有疑问语气、语调的句子。多数用升调，根据疑问标记方式还可分为是非问、选择问、正反问和特指问四类。例如：“你饿了吗？”（是非）、“我们是去公园还是去逛街？”（选择）、“你来不来？”（正反）、“谁动了我的奶酪？”（特指）。
- (3) 祈使句：要求受话人实施某种行为的句子，常用短促的降调。例如：“放下武器！”。
- (4) 感叹句：表达强烈情感的句子，常用舒缓绵长的降调。例如：“人心不古哇！”。

几类特殊句式

句式是根据句子的局部结构特征划分出来的句子类型，体现了语言在形式与表意等方面的特色。现代汉语中常见的、有结构特点的句式主要有主谓谓语句、双宾句、连谓句、兼语句、存现句、“把”“被”诸字句等。因篇幅有限，在此就不一一赘述这些特殊句式的构成与特点，有兴趣的读者自行参阅相关教材，本书举几例供大家思考。例如：“什么大风大浪他没见识过？”（主谓谓）、“老师告诉小华明天比赛。”（双宾）、“这大饼吃起来硌牙。”（连谓）、“你妈妈喊你回家吃饭！”（兼语）、“天墙上挂着一副世界地图。”（存现）、“你可把我气坏了！”（“把”字句）、“小明被人打了。”（“被”字句）。

析句方法

一个单句可以从句法、语义、语用等角度进行分析，其中单句的句法分析仍可以沿用我们之前介绍的层次分析法，在此就不再赘述了。而语义分析最常见的有三种方法，分别是：语义特征、语义角色和语义指向。其中语义特征的表示方法有点类似于之前讲过的义素分析法，都是用 [+/-A] 来进行标注的；语义角色则是根据词语间的相互关系将其划分为“时间/处所”“施/受/与事”“工具”“动作”“结果”等成分；至于语义指向则是利用箭头将句中指向不明的两个成分关联起来。

最后，若是要对句子进行语用分析，则有几组概念必须先做了解。在实际交流过程中，人们常常会出于各种各样的需要故意对句子成分进行简省或调换，这些变化后的句型统一叫作“变式句”。其中简省了成分的叫“省略句”，调换了句子成分的叫“倒装句”。面对以上变式句的语用分析，我们第一步要做的就是通过扩展、移位转换等辅助手段将其变为正常结构及语序的句子来进行分析。

6.3.5 复句

什么是复句

上文已经提到“复句”是结构上与“单句”相对的句子类别。它特指由两个及以上意义相关而结构上互不包含的分句，加上贯通前后的句调后所构成的句子。

复句的意义分类

根据复句分句间的语义关系，我们可以把复句相应地分为联合复句、偏正复句两大类。各分句间意义地位平等且无主从之分的复句叫联合复句（等立复句），反之句类有主从、正偏之分的就叫偏正复句（主从复句）。二者之下又可以进一步细分，具体如下。

◎ 联合复句

- （1）并列复句：前后分句分别对有关联的几件事情或同一事物的不同方面进行叙述，常用意合法或“一方面……一方面”“既……又”等关联词语连接。例如：“读诗使人灵秀，数学使人周密。”（意合法）、“低碳不仅是一种生活方式，也是一种生活态度。”（关联词）。

- (2) 承接复句: 前后分句按时空、逻辑等顺序说明相关的动作或情况, 常用分句顺序或关联词“然后”“接着”“便”等连接。例如: “湖水滋润着湖边的青草, 青草喂肥了羊群, 羊奶哺育着少女的后代子孙。”(顺序)、“他说家里还有事, 就提前走了。”(关联词)。
- (3) 解说复句: 用后一分句解释说明或归纳概括前面的分句, 有总分和解释两种关系, 一般不用关联词, 以分句间语义关系连接。例如: “我从国外给你带回来了一盒巧克力, 榛子味的。”(解释)、“世上有两样东西不可直视, 一是太阳, 二是人心。”(总分)。
- (4) 选择复句: 提出两种或多种可能的情况进行选择, 常用“是……还是”“要么……要么”“宁可……也不”等合用关联词连接。例如: “如果我不在家, 就是在咖啡馆。如果不是在咖啡馆, 就是在往咖啡馆的路上。”
- (5) 递进复句: 用后面分句进一步解释说明前句的意思, 从而构成由浅入深的表达格局, 反之亦可。常用关联词“不但……而且”“尚且……何况”“甚至”等连接。例如: “木犹如此, 人何以堪。”

◎ 偏正复句

- (1) 条件复句: 偏句提出条件, 正句说明这一条件推出的结果。有充足条件、必要条件、无条件三类, 常用关联词“只要……就”“只有……才”“任凭”等连接。例如: “只要人人都献出一点爱, 世界将变成美好的人间。”(充足)、“无论是谁, 都会在不经意间失去什么。”(无条件)。
- (2) 假设复句: 偏句先提出假设, 正句再说明将产生的结果。有一致和相悖两类关系, 常用关联词“如果……那么”“即使……也”等连接。例如: “如果你不说出去, 就没有人会知道。”(一致)、“你要是不方便的话, 也可以明天来。”(相悖)。
- (3) 因果复句: 偏句说原因, 正句表结果。有说明和推论两种关系, 常用关键词“因为……所以”“既然”等连接。例如: “因为一个小小的失误, 这次的行动彻底失败。”(说明)、“小明这题都算错了, 可见他上课没有认真学。”(推论)。
- (4) 目的复句: 偏句说行为, 正句表明目的。有得到和避免两类, 常用关联词“为了”“以免”等连接。例如: “你把这些零食都带上, 路上才好充饥。”(得到)、“一块儿上, 省得我一个个收拾你们!”(避免)。
- (5) 转折复句: 前后分句语义上相对或相反。依转折强度不同, 有重转、轻转、弱转之分。常用关联词“虽然……但是”(重)、“然而”“却”(轻)、“只是”“不过”(弱)等连接。例如: “虽然已是家财万贯, 他却仍郁郁寡欢。”

多重复句与紧缩句

根据结构层次的数量，复句可以划分成“一重复句”和“多重复句”两类。我们之前所举的例句大多都是一重复句，而多重复句则是由两个及以上结构层次嵌套使用得到的句子。分析时常常根据各分句间的关系，第一层复句用|，第二层复句用||，以此类推，将句中组合层次进行切分，并在竖线旁标明结构关系。例如：“老家没有高楼大厦，||递进甚至没有电灯电话，|转折可那儿的人们关系单纯亲切，||并列生活自在悠闲。”

紧缩句是取消了分句间语音停顿，压缩了某些关联词语，结构上更为紧凑的复句。它与单句中的连谓句颇为相像，分析时要仔细辨认，区别主要在于结构上有无关联词，语义上有无各类关系。例如：“他俩见面交谈起来。”（连谓）、“他俩一见面就交谈起来。”（紧缩）。

第 7 章

自然语言处理

本章导读：本章进入自然语言处理篇章，作为开篇，有必要给读者详细介绍自然语言处理的全貌。本章开篇直击要点，即自然语言处理的任务和限制。进而介绍其所涉及的主要技术范畴，并对这些技术方向进行介绍。在针对当前自然语言处理的难点进行详细剖析后，最终对 2017 年以后自然语言处理的发展进行展望。

7.1 自然语言处理的任务和限制

NLP 是一种极具吸引力的人机交互方式。早期的语言处理系统如 SHRDLU，当它们处于一个有限的“积木世界”，即运用有限的词汇表会话时，工作得相当好。这使得研究员们对此系统相当乐观，然而，当把这个系统拓展到充满了含糊与不确定性的现实环境中时，他们很快丧失了信心：原因在于理解自然语言需要关于外在世界的广泛知识，以及运用操作这些知识的能力。自然语言认知，同时也被视为一个人工智能完备的问题，在自然语言处理中，“理解”的定义也变成一个主要的问题。

一些 NLP 面临的问题实例

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，计算机就无法区分（中文里“动物它”和“事物它”没有区别，并且代词在中文里常常被省略，因此需区别属性并且标示出来）。

语音识别（Speech Recognition）技术也被称为语音转文本识别（Speech to Text, STT），其目标是让计算机自动将人类的语音内容转换为相应的文字，如图 7-2 所示。与说话人识别及说话人确认不同，后者尝试识别或确认发出语音的说话人而非其中所包含的词汇内容。



图 7-2 语音识别

语音识别技术的应用包括语音拨号、语音导航、室内设备控制、语音文档检索、简单的听写数据录入等。语音识别技术与其他自然语言处理技术如机器翻译及语音合成技术相结合，可以构建出更加复杂的应用，例如，语音到语音的翻译。

语音识别技术涉及的领域包括信号处理、模式识别、概率论和信息论、发声机理和听觉机理、人工智能，等等。

7.2.3 中文自动分词

中文自动分词（Chinese Word Segmentation）

中文自动分词指的是使用计算机自动对中文文本进行词语的切分，即像英文那样使得中文句子中的词之间有空格以标识。中文自动分词被认为是中文自然语言处理中的一个最基本的环节（图 7-3 引用自 NLPIR 汉语分词系统）。

由联合国环境规划署组织 20 多个国家的 55 名科学家共同完成。报告指出，为了达到控制气温上升幅度的目标，2020 年全球二氧化碳排放量需从 2010 年的约 490 亿吨降至 440 亿吨。但如果各国不采取有力行动，这个数值到 2020 年反而很可能能上升到 580 亿吨；就算把现在世界各国最高程度的减排承诺加在一起，也会达到 520 亿吨，离目标仍然存在 80 亿吨的差距。与联合国环境规划署去年发布的报告相比，这个差距又扩大了 20 亿吨。联合国副秘书长、环境规划署执行主任阿希姆·施泰纳通过远程视频参加了当天的新闻发布会。他说：“严峻的事实说明全球向低碳和绿色经济模式的转变仍然太慢，达到 440 亿吨目标的机会正在逐年减小。”联合国环境规划署首席科学家约瑟夫·阿尔卡莫在接受新华社记者采访时强调，希望仍然存在。他说

图 7-3 中文自动分词

现有方法如下。

- ◎ 基于词典的匹配：前向最大匹配、后向最大匹配。
- ◎ 基于字的标注：最大熵模型、条件随机场模型、感知器模型。
- ◎ 其他方法：与词性标注结合、与句法分析结合。

7.2.4 词性标注

词性标注（Part-of-Speech tagging）

词性标注（Part-of-Speech tagging 或 POS tagging）又称词类标注或者简称标注，是指在词性标记集已确定，并且词典中每个词都有确定词性的基础上，将一个输入词串转换成相应词性标记串的过程（图 7-4 引用自 NLPPIR 汉语分词系统）。



图 7-4 中文词性标注

在汉语中，因为汉语词汇词性多变的情况比较少见，大多词语只有一个词性，或者出现频次最高的词性远远高于第二位的词性，相对比较简单。同时，它也受到一些条件约束。比如：兼类词在具体语境中的词性判定问题、未登录词即新词词性问题、兼类词问题等。

词性标注方法包括概率方法、隐马尔可夫模型的词性标注方法、机器学习规则的方法等。

7.2.5 句法分析

句法分析（Parsing）

句法分析（Parsing）就是指对句子中的词语语法功能进行分析。比如“欢迎大家使用演示平台”（图 7-5 引用自 NLPPIR 汉语分词系统）。

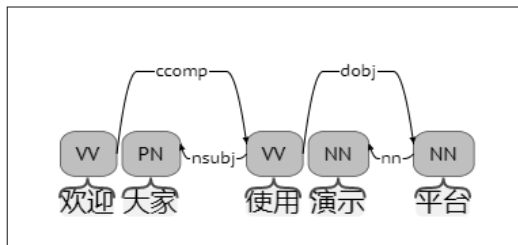


图 7-5 句法分析

句法分析在中文信息处理中的主要应用包括机器翻译、命名实体识别等。

自然语言生成 (Natural Language Generation)

自然语言生成研究使计算机具有人一样的表达和写作功能，即能够根据一些关键信息及其在机器内部的表达形式，经过一个规划过程，自动生成一段高质量的自然语言文本。自然语言处理包括自然语言理解和自然语言生成。自然语言生成是人工智能和计算语言学的分支，相应的语言生成系统是基于语言信息处理的计算机模型，其工作过程与自然语言分析相反，从抽象的概念层次开始，通过选择并执行一定的语义和语法规则来生成文本。

7.2.6 文本分类

文本分类 (Text Categorization)

文本分类用计算机对文本集按照一定的分类器模型进行自动分类标记。文本分类的总体过程如下（图 7-6 引用自 NLPPIR 汉语分词系统）。

- （1）预处理：将原始语料格式化为同一格式，便于后续的统一处理。
- （2）索引：将文档分解为基本处理单元，同时降低后续处理的开销。
- （3）统计：词频统计，项（单词、概念）与分类的相关概率。
- （4）特征抽取：从文档中抽取反映文档主题的特征。
- （5）分类器：分类器的训练。
- （6）评价：分类器的测试结果分析。

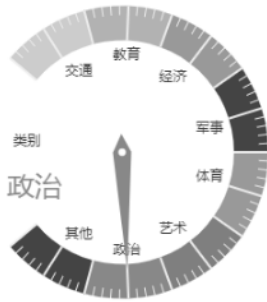


图 7-6 文本分类

文本分类常用算法包括决策树、朴素贝叶斯、神经网络、支持向量机、线性最小平方拟合、KNN、遗传算法、最大熵等，广泛应用于垃圾过滤、新闻分类、词性标注等。

7.2.7 文本挖掘

文本挖掘

文本挖掘一般指在文本处理过程中产生高质量的信息。高质量的信息通常通过分类和预测来产生，如模式识别。文本挖掘通常涉及输入文本的处理过程，产生结构化数据，并最终评价和解释输出（图 7-7 引用自 NLPiR 汉语分词系统）。

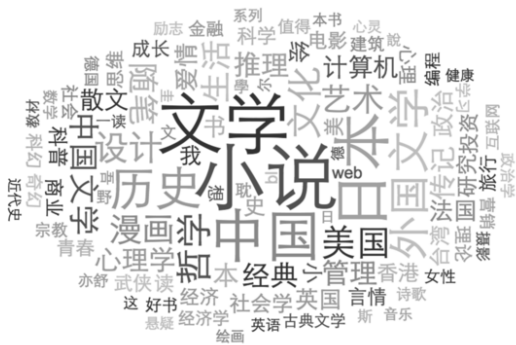


图 7-7 文本挖掘

典型的文本挖掘方法包括文本分类、文本聚类、信息抽取、概念/实体挖掘、情感分析和观点分析等。

7.2.8 信息抽取

信息抽取 (Information Extraction)

信息抽取 (Information Extraction) 是从大量文字数据中自动为访问数据库而抽取特定消息的技术 (图 7-8 引用自 NLPPIR 汉语分词系统)。

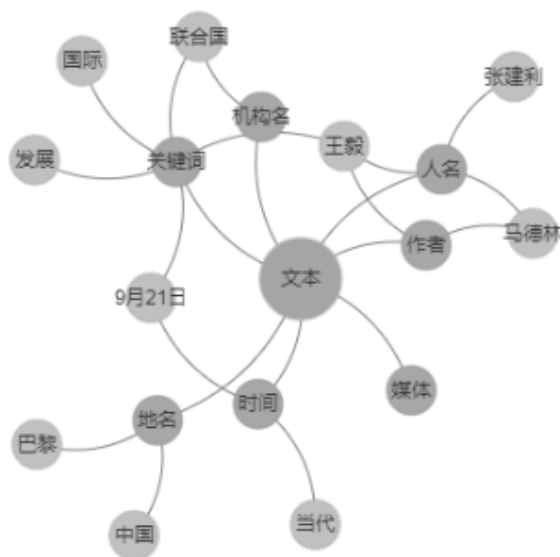


图 7-8 信息抽取

简单可以理解为从给定文本中抽取重要的信息，比如时间、地点、人物、事件、原因、结果、数字、日期、货币、专有名词等。通俗说来，就是要了解谁在什么时候、什么原因、对谁、做了什么事、有什么结果，涉及实体识别、时间抽取、因果关系抽取等关键技术。

7.2.9 问答系统

问答系统 (Question Answering)

问答系统（Question Answering）是当下自然语言处理研究的热点，也是未来自然语言处理的重点问题。从问答系统的外部行为来看，其与目前主流资讯检索技术有两点不同：首先是查询方式为完整而口语化的问句，再者是其回传的为高精度网页结果或明确的答案字串，如图 7-9 所示。



图 7-9 问答系统

以 Ask Jeeves 为例，使用者不需要思考该使用什么样的问法才能够得到理想的答案，只需要用口语化的方式直接提问如“请问谁是美国总统？”即可。而系统在了解使用者问句后，会非常清楚地回答“特朗普是美国总统”。从系统内部来看，问答系统使用了大量有别于传统资讯检索系统的自然语言处理技术，如自然语言剖析（Natural Language Parsing）、问题分类（Question Classification）、专名辨识（Named Entity Recognition）等。

7.2.10 机器翻译

机器翻译（Machine Translation）

机器翻译（Machine Translation，经常简写为 MT）属于计算语言学的范畴，是计算机程序将文字或演说从一种自然语言翻译成另一种自然语言，如图 7-10 所示。

简单来说，机器翻译是通过将一个自然语言的字辞取代成另一个语言的字辞来实现的。借由使用语料库的技术，可达成更加复杂的自动翻译，包括可更佳地处理不同的文法结构、辞汇辨识、惯用语的对应等。



图 7-10 机器翻译

一般而言，大众使用机器翻译的目的只是为了获知原文句子或段落的要旨，而不是精确的翻译。总的来说，机器翻译的效果并没有达到可以取代人工翻译的程度，所以目前还无法成为正式的翻译。

不过现在已有越来越多的公司尝试用机器翻译的技术为公司网站提供多语系的服务支援。例如，微软公司试以机器翻译将其 MSDN 自动翻译成多国语言，如上文所说，知识库作为专业领域，其文法较为制式化，翻译结果往往更符合自然语言。

7.2.11 文本情感分析

文本情感分析

文本情感分析（也称为意见挖掘）是指用自然语言处理、文本挖掘及计算机语言学等方法来识别和提取原素材中的主观信息（图 7-11 引用自 NLPIR 汉语分词系统）。

通常来说，情感分析的目的是为了找出说话者/作者在某些话题上或者针对一个文本两极的观点的态度。这个态度或许是他的个人判断或评估，或许是他当时的情感状态（也就是说，作者在做出这个言论时的情绪状态），或是作者有意向的情感交流（就是作者想要读者所体验的情绪）等。

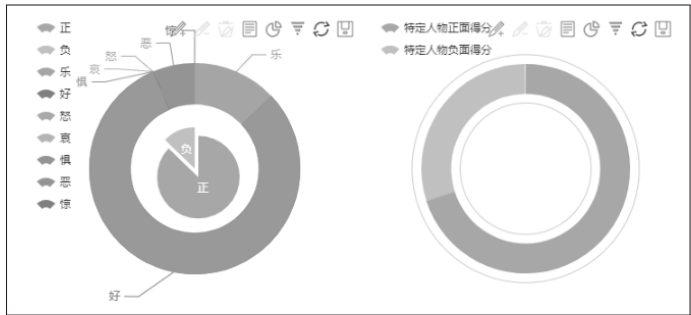


图 7-11 文本情感分析

7.2.12 自动摘要

自动摘要 (Automatic Summarization)

所谓自动摘要就是利用计算机自动地从原始文献中提取文摘，文摘是全面准确地反映某一文献中心内容的连贯短文。常用方法是将文本作为句子的线性序列，将句子视为词的线性序列，如图 7-12 所示。

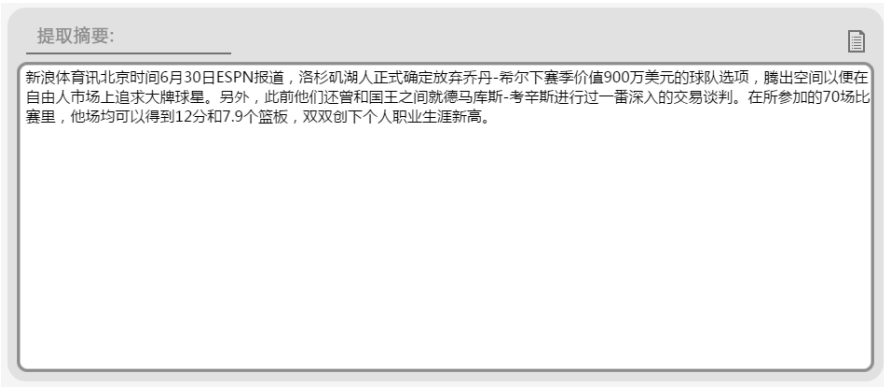


图 7-12 自动摘要

自动摘要可以按照技术类型和信息提取分类。

- ◎ 技术应用类型：自动提取给定文章的摘要信息，自动计算文章中词的权重，自动计算文章中句子的权重。

- ◎ 信息提取：单篇文章的摘要自动提取，大规模文档的摘要自动提取，基于分类的摘要自动提取。

7.2.13 文字蕴涵

文字蕴涵（Textual Entailment, TE）

文字蕴涵在自然语言处理中主要指一个文字片段之间的定向关系。

- ◎ 正向蕴涵

文本 T：日本时间 2011 年 3 日 11 日，日本宫城县发生里氏震级 9.0 强震，造成死伤失踪约 3 万多人。

假设 H：日本时间 2011 年 3 日 11 日，日本宫城县发生里氏震级 9.0 强震。

- ◎ 矛盾蕴涵

文本 T：张学友在 1961 年 7 月 10 日，生于香港，祖籍天津。

假设 H：张学友生于 1960 年。

- ◎ 独立蕴涵

文本 T：黎姿与“残障富豪”马廷强结婚。

假设 H：马廷强为香港“东方报业集团”创办人之一马惜如之子。

7.3 自然语言处理的难点

7.3.1 语言环境复杂

自然语言处理的语言环境较为复杂，以命名实体识别进行分析，对于同一个汉字某些情况下可以看作实体处理，某些情况则不能看作实体。例如：

- ◎ 人名，比如《天龙八部》中“婢子四姊妹一胎孪生，童姥姥给婢子取名为梅剑，这三位妹子是兰剑、竹剑、菊剑。”人物“竹剑”，某些情况下就是指的一种竹子做的剑。
- ◎ 地名，比如《射雕英雄传》中“陆庄主知道此人是湖南铁掌帮的帮主”中地点“湖南”，在某种情况下就指代地理方位“湖的那边”。

- ◎ 机构名，比如《鹿鼎记》中“这位是莲花堂香主蔡德忠蔡伯伯。”组织机构名（帮派名）“莲花堂”，在某种情况就指代种植莲花的一个地方，变成地点名了。

7.3.2 文本结构形式多样

文本内部结构形式多样。还是以自然语言处理中的命名实体识别任务为例子，例如：

- ◎ 人名，人名由姓和名构成。其中姓氏包括单姓和复姓（如赵、钱、孙、李、慕容、东方、西门等），名由若干个汉字组成。姓氏的用字范围相对有限，比较容易识别。然而名就比较灵活，既可以用名、字、号表示，也可以使用职务名和用典。比如：“李白、李十二、李翰林、李供奉、李拾遗、李太白、青莲居士、谪仙人”都是同一个人。
- ◎ 地名，一般由若干个字组成地名，可以为作为后缀关键字或者别名，都是指代一个地方。比如：“成都、蓉城、锦城、芙蓉城、锦官城、天府之国”，其中“蓉城、锦城、芙蓉城、锦官城、天府之国”为别名。除了全称的名称，还有地理位置代表地名的。比如：“河南、河南省、豫”都是指的一个省份，其中“豫”是简称。
- ◎ 组织机构名，组织机构命名方式比较复杂，有些是修饰性的命名，有些表示历史典故，有些表示地理方位，有些表示地名，有些表示风俗习惯和关键字等。例如：组织名“广州恒大淘宝足球俱乐部”中，“广州”表示地名的成分，“恒大”“淘宝”表示公司名称成分，“足球”是一项体育赛事成分，“俱乐部”是关键字的成分。比如：“四川大学附属中学”（四川省成都市第十二中学）中包括另一个机构名“四川大学”。机构名还可以以简称形式表示，比如：“四川大学附属中学”简称“川大附中”，“成都信息工程大学”简称“成信大”。

7.3.3 边界识别限制

在自然语言处理任务中，边界识别最广泛应用于命名识别当中。边界识别可以分解为两大任务：如何去识别实体的边界；如何去判定实体的类别（诸如人名、地名、机构名）。中文命名实体识别要比英文命名实体识别更为复杂，一是受中文自身语言特性的限制，不同于英语文本中词间有空格界定；二是英文中的实体一般首字母大写容易区分，例如：‘Jobs was adopted at birth in San Francisco, and raised in a hotbed of counterculture’中，人名乔布斯 Jobs 的首字母大写，地名旧金山 San Francisco 的首字母也是大写，而中文不具备这样的特征。

7.3.4 词义消歧

词义消歧

词义消歧是一个自然语言处理和本体论的开放问题。歧义与消歧是自然语言理解中最核心的问题，在词义、句义、篇章含义层次都会出现语言根据上下文语义而产生不同含义的现象。消歧即指根据上下文确定对象语义的过程，词义消歧即在词语层次上的语义消歧。语义消歧/词义消歧是自然语言处理任务的一个核心与难点，影响了几乎所有任务的性能，比如搜索引擎、意见挖掘、文本理解与产生、推理等。

词性标注与词义消歧

词性标注与词义消歧是相互关联的两个问题，在语言使用者身上它们往往同时能得到满足。但是目前的计算机系统一般并不能让二者共用参数并同时输出。语义理解包括分词、词性标注、词义消歧、句法解析、语义解析等。它们并不是前馈的，是相互依赖并存在反馈的。

词性标注与语义消歧都要依赖上下文来标注，但是词性标注比语义消歧处理起来要更简单，最终结果也往往较好。主要原因是词性标注的标注集合是确定的，而语义消歧并没有，并且量级上词性标注要大得多；词性标注的上下文依赖比语义消歧要短。

典型例子

许多字词不单只有一个意思，因而我们必须选出使句意最为通顺的解释。看下面歧义的句子，词义消歧就是要分析出特定上下文的词被赋予的到底是哪个意思。

- (1) 川大学生上网成瘾如患绝症。歧义在于“川大学生”（四川大学的学生；四川的大学生）。
- (2) 两代教授，人格不同。歧义：“两代”（两位代理教授；两个时代的教授）。
- (3) 被控私分国有资产，专家总经理成了被告人。歧义：“专家总经理”（专家和总经理；有专家身份的总经理）。
- (4) 新生市场苦熬淡季。歧义：“新生”（新学生的市场；新产生的市场）。
- (5) 朝鲜十年走近国际社会一步。歧义：“十年走近国际社会一步”（每十年就向国际社会走近一步；最近十年间向国际社会走近了一步）。

- (6) 新汽车牌照。歧义：“新”（新的汽车；新的牌照）。
- (7) 咬死了猎人的狗。歧义：（猎人的狗被咬死了；把猎人咬死了的那条狗）。
- (8) 菜不热了。歧义：“热”（指菜凉了；指菜不加热了）。
- (9) 还欠款四万元。歧义：“还”（读 huai；读 hai）。
- (10) 北京人多。歧义：北京/人多；北京人/多。

7.3.5 指代消解

定义

指代消解（Anaphora Resolution）是自然语言处理的重要内容，在信息抽取时就用到了指代消解技术。

中文的三种典型指代

- (1) 人称代词：李明怕高妈妈一个人呆在家里寂寞，【他】便将家里的电视搬了过来。
- (2) 指示代词：很多人都想创造一个美好的世界留给孩子，【这】可以理解，但不完全正确。
- (3) 有定描述：贸易制裁似乎成了美国政府对华关系中惯用的大棒。然而，这【大棒】果真如美国政府所希望的那样灵验吗？

典型指代消解

◎ 显性代词消解

所谓显性代词消解，就是指在篇章中确定显性代词指向哪个名词短语的问题，代词称为指示语或照应语（Anaphor），其所指向的名词短语一般被称为先行语（Antecedent）。根据二者之间的先后位置，可分为回指（Anaphora）与预指（Cataphora），其中：如果先行语出现在指示语之前，则称为回指，反之则称为预指。

◎ 零代词消解

所谓零代词消解，是代词消解中针对零指代（Zero Anaphora）现象的一类特殊的消解。在篇章中，用户能够根据上下文关系推断出的部分经常会省略，而省略的部分（用零代词表示）在句子中承担着相应的句法成分，并且回指前文中的某个语言单位。零指代现象在汉语中更加常见，近几年随着各大评测任务的兴起开始受到学者们的广泛关注。

◎ 共指消解

所谓共指消解，是将篇章中指向同一现实世界客观实体（Entity）的词语划分到同一个等价集的过程，其中被划分的词语称为表述或指称语（Mention），形成的等价集称为共指链（Coreference Chain）。在共指消解中，指称语包含普通名词、专有名词和代词，因此可以将显性代词消解看作共指消解针对代词的子问题。共指消解与显性代词消解不同，它更关注在指称语集合上进行的等价划分，评测方法与显性代词消解也不尽相同，通常使用 MUC、B-CUBED、CEAF 和 BLANC 等评价方法。

指代消解的研究方法大致可以分为基于启发式规则的、基于统计的和基于深度学习的方法。目前看来，基于有监督统计机器学习的消解算法仍然是主流算法。

典型例子

指代消解是解决“谁对谁做了什么”，处理如上所述的自然语言的问题，下面看看例子：

- （1）美国政府表示仍然支持强势美元，但这到底只是嘴上说说还是要采取果断措施，经济学家对此的看法是否定的。
- （2）今天老师又在班会上表扬了自己，但是我觉得还需要继续努力。
- （3）三妹拉着葛姐的手说，她老家在偏远的山区，因为和家里赌气才跑到北京打工的，接着她又哭泣起自己的遭遇来。
- （4）当他把证书发给小钱时，他对他笑了。
- （5）小明和肖华去公园玩，他摔了一跤，他急忙把他扶起来。
- （6）星期天，小雨和小英到田老师家补习功课，她一早就打电话给她约好在红旗饭店吃早餐。

7.4 自然语言处理展望

2017 年，在第三届中国人工智能大会上，哈尔滨工业大学刘挺教授对自然语言处理的发展趋势做了一次精彩的归纳，他把趋势分成了十个方面（下文来源于刘挺教授的报告）。

趋势 1：语义表示——从符号表示到分布表示

自然语言处理一直以来都是比较抽象的，都是直接用词汇和符号来表达概念。但是使用符号存在一个问题，比如两个词，它们的词性相近但词形不匹配，计算机内部就会认为它们是两个词。举个例子，荷兰和苏格兰这两个国家名，如果我们在一个语义的空间里，用词汇与词汇组合的方法，把它表示为连续、低维、稠密的向量，就可以计算不同层次的语言单元之间的相似度。这种方法同时也可以被神经网络直接使用，是这个领域的一个重要的变化。

从词汇间的组合，到短语、句子，一直到篇章，现在有很多人在做这个事，这和以前的思路是完全不一样的。有了这种方法之后，再用深度学习，就带来了一个很大的转变。原来我们认为自然语言处理要分成几个层次，但是就句法分析来说，它是人为定义的层次，那它是不是一定是必要的？这里应该打一个问号。

实际工作中，我们面临着一个课题——信息抽取。我之前和一个单位合作，初衷是我做句法分析，然后他们在我的基础上做信息抽取，相互配合，后来他们发表了一篇文章，与初衷是相悖的，它证明了没有句法分析，也可以直接做端到端的直接的实体关系抽取，这很震撼，不是说现在句法分析没用了，而是我们认为句法分析是人为定义的层次，在端到端的数据量非常充分，可以直接进行信息抽取的时候，那么不用句法分析，也能达到类似的效果。当端到端的数据不充分时，才需要人为划分层次。

趋势 2：学习模式——从浅层学习到深度学习

浅层到深层的学习模式中，浅层是分步骤走，可能每一步都用了深度学习的方法，实际上各个步骤是串接起来的。直接的深度学习是一步到位的端到端，在这个过程中，我们确实可以看到一些人为贡献的知识，包括该分几层，每层的表示形式，一些规则等，但我们所谓的知识在深度学习里所占的比重确实减小了，主要体现在对深度学习网络结构的调整。

◎ 趋势 3：NLP 平台化——从封闭走向开放

以前我们搞研究的，都不是很愿意分享自己的成果，比如程序或是数据，现在这些资料彻底开放了，无论是学校还是大企业，都更多地提供平台。NLP 领域提供的开放平台越来越多，它的门槛也越来越低。

语音和语言其实有很大的差别，我认识的好几位国内外的进入 NLP 的学者，他们发现 NLP 很复杂，因为像语音识别和语音合成等只有有限的问题，而且这些问题的定义非常清晰。但到了自然语言，要处理的问题变得纷繁复杂，尤其是 NLP 和其他领域还会有所结合，所以问题非常琐碎。

趋势 4：语言知识——从人工构建到自动构建

AlphaGo 告诉我们，没有围棋高手介入它的开发过程，到 AlphaGo 最后的版本，它已经不怎么需要看棋谱了。所以 AlphaGo 在学习和使用过程中都有可能会超出人的想象，因为它并不是简单地跟人学习。

美国有一家文艺复兴公司，它做金融领域的预测，但是这个公司不招金融领域的人，只招计算机、物理、数学领域的人。这就给了我们一个启发，计算机不是跟人的顶级高手学，而是用自己已有的算法，去直接解决问题。

但是在自然语言处理领域，还是要有大量的显性知识的，但是构造知识的方式也在产生变化。比如，现在我们开始用自动的方法，自动地去发现词汇与词汇之间的关系，像毛细血管一样渗透到各个方面。

趋势 5：对话机器人——从通用到场景化

最近出现了各种图灵测试的翻版，就是做知识抢答赛来验证人工智能，从产学研应用上来讲就是对话机器人，非常有趣味性和实用价值。

这块的趋势在哪里？我们知道，从 Siri 刚出来，国内就开始做语音助手了，后来语音助手很快下了马，因为它可以听得到但是听不懂，导致后面的服务跟不上。后来国内把难度降低成了聊天，你不是调戏 Siri 吗？我就做小冰就跟你聊。但是难度降低了，实用性却跟不上来，所以在用户的留存率上，还是要打个问号。

现在更多的做法和场景结合，降低难度，然后做任务执行，即希望做特定场景时有用的人机对话。在做人机对话的过程中，大家热情一轮比一轮高涨，但是随后发现，很多问题是由于自然语言的理解没有到位，才难以产生真正的突破。

趋势 6：文本理解与推理——从浅层分析向深度理解迈进

Google 等都已经推出了这样的测试机——以阅读理解作为一个深入探索自然语言理解的平台。也就是说，给计算机一篇文章，让它去理解，然后人问计算机各种问题，看计算机是否能回答，这样做是很有难度的，因为答案就在这文章里面，人会很刁钻地问计算机。所以说阅读理解是现在竞争的一个很重要的点。

趋势 7：文本情感分析——从事实性文本到情感文本

多年以前，很多人都在做新闻领域的事实性文本，而如今，搞情感文本分析的似乎更受群众欢迎，这一块这在商业和政府舆情上也都有很好的应用。

趋势 8：社会媒体处理——从传统媒体到社交媒体

相应的，在社会媒体处理上，从传统媒体到社交媒体的过渡，情感的影响是一方面，大家还会用社交媒体做电影票房的预测，做股票的预测等。但是从长远的角度看，社会、人文等学科与计算机学科的结合是历史性的。比如，在文学、历史学等学科中，有相当一部分新锐学者对本门学科的计算机的大数据非常关心，这两者在碰撞，未来的前景是无限的，而自然语言处理是其中重要的、基础性的技术。

趋势 9：文本生成——从规范文本到自由文本

文本生成这两年很火，从生成古诗词到生成新闻报道再到写作文。这方面的研究价值是很大的，它的趋势是从生成规范性的文本到生成自由文本。比如，我们可以从数据库里面生成一个可以模板化的体育报道，这个模板是很规范的。我们可以再向自由文本过渡，比如写作文。

趋势 10：NLP+ 行业——与领域深度结合，为行业创造价值

最后谈与企业的合作。现在像银行、电器、医药、司法、教育、金融等各个领域对 NLP 的需求都非常多。

我预测 NLP 首先会在信息准备得充分并且服务方式本身就是知识和信息的领域产生突破。还比如司法领域，它的服务本身也有信息，它就会首先使用 NLP。NLP 最主要会用在以下四个领域：医疗、金融、教育和司法。

第 8 章

语料库

本章导读：大数据发展的基石就是数据量的快速增加，无论是自然语言处理、数据挖掘、文本处理，还是机器学习领域，都是在此基础上通过规则或统计方法进行模型构建的。但是不是数据量足够大就叫大数据了呢？是不是数据量足够多就构成语料库了呢？带着这些疑问，本章将带你走进语料库的世界，对语料知识进行一次全面而深入的了解。

8.1 语料库浅谈

自然语言

自然语言（Natural language）通常是指一种自然地随文化演化的语言（如英语、汉语、法语、日语等）。人类使用的语言都会被视为“自然”语言，以相对于如编程语言等为计算机而设的“人造”语言。这种用法可见于自然语言处理一词中。自然语言是人类交流和思维的主要工具。

语料和语料库

语料通常指在统计自然语言处理中实际上不可能观测到大规模的语言实例。所以人们简单地用文本作为替代，并把文本中的上下文关系作为现实世界中语言的上下文关系的替代品。

语料库一词在语言学上意指大量的文本，通常经过整理，具有既定格式与标记。其具备三个显著的特点：

- ◎ 语料库中存放的是在语言的实际使用中真实出现过的语言材料。
- ◎ 语料库以电子计算机为载体承载语言知识的基础资源，但并不等于语言知识。
- ◎ 真实语料需要经过加工（分析和处理），才能成为有用的资源。

语料库语言学

大多数学者普遍认为：“语言学的研究必须基于语言事实的基础，必须详尽大量地占有材料，才有可能在理论上得出比较可靠的结论”。传统语言材料的搜集整理和加工完全以手工进行，费时费力，直到计算机出现并随着计算能力强大之后，原先手工的工作开始转向计算机去做，后来在逐渐的方法完善中，提出一些初步的理论，形成了语料学这样一门语言学与计算机科学交叉的学科。

语料库语言学的研究范畴：主要研究机器可读自然语言文本的采集、存储、检索、统计、语法标注、句法语义分析，以及具有上述功能的语料库在语言教学、语言定量分析、词汇研究、词语搭配研究、词典编制、语法研究、语言文化研究、法律语言研究、作品风格分析、自然语言理解、机器翻译等方面的应用。

建立语料库的意义

语料库是为一个或者多个应用目标而专门收集的，有一定结构的、有代表的、可被计算机程序检索的、具有一定规模的语料集合。本质上讲，语料库实际上是通过自然语言运用的随机抽样，以一定大小的语言样本来代表某一研究中所确定的语言运用的总体。

8.2 语料库深入

语料库划分与种类

语料库的划分一直是标准各异，其中冯志伟教授的语料库划分比较有影响力且在学术界认可度较高。其划分类型如下：

- ◎ 按语料选取的时间划分，可分为历时语料库（diachronic corpus）和共时语料库（synchronic corpus）。
- ◎ 按语料的加工深度划分，可分为标注语料库（annotated corpus）和非标注语料库（non-annotated corpus）。

- ◎ 按语料库的结构划分,可分为平衡结构语料库(balance structure corpus)和自然随机结构的语料库(random structure corpus)。
- ◎ 按语料库的用途划分,可分为通用语料库(general corpus)和专用语料库(specialized corpus)。
- ◎ 按语料库的表达形式划分,可分为口语语料库(spoken corpus)和文本语料库(text corpus)。
- ◎ 按语料库中语料的语种划分,可分为单语种语料库(monolingual corpora)和多语种语料库(multilingual corpora)。多语种语料库又可以再分为比较语料库(comparable corpora)和平行语料库(parallel corpora)。比较语料库的目的侧重于特定语言现象的对比,而平行语料库的目的侧重于获取对应的翻译实例。
- ◎ 按语料库的动态更新程度划分,可分为参考语料库(reference corpus)和监控语料库(monitor corpus)。参考语料库原则上不做动态更新,而监控语料库则需要不断地进行动态更新。

语料库构建原则

语料库应该具有代表性、结构性、平衡性、规模需求并制定语料的元数据规范,各个原则具体介绍如下:

- ◎ 代表性:在应用领域中,不是根据量而划分是否是语料库,而是在一定的抽样框架范围内采集而来的,并且能在特定的抽样框架内做到代表性和普遍性。
- ◎ 结构性:有目的地收集语料的集合,必须以电子形式存在,计算机可读的语料集合结构性体现在语料库中语料记录的代码、元数据项、数据类型、数据宽度、取值范围、完整性约束。
- ◎ 平衡性:主要体现在平缓因子——学科、年代、文体、地域、登载语料的媒体、使用者的年龄、性别、文化背景、阅历、预料用途(私信/广告等),根据实际情况选择其中一个或者几个重要的指标作为平衡因子,最常见的平衡因子有学科、年代、文体、地域等。
- ◎ 规模性:大规模的语料对语言研究特别是对自然语言研究处理很有用,但是随着语料库的增大,垃圾语料越来越多,语料达到一定规模以后,语料库功能不能随之增长,语料库规模应根据实际情况而定。

- ◎ 元数据：元数据对于研究语料库有着重要的意义，我们可以通过元数据了解语料的时间、地域、作者、文本信息等；构建不同的子语料库；对不同的子语料对比；记录语料知识版权、加工信息、管理信息等。

注意：汉语词与词之间没有空隙，不便于计算机处理，一般需要进行切词和词性标注。

语料标注的优缺点

- ◎ 优点：研究方便。可重用、功能多样、分析清晰。
- ◎ 缺点：语料不客观（手工标注准确率高而一致性差，自动或者半自动标注一致性高而准确率差）、标注不一致、准确率低。

8.3 自然语言处理工具包：NLTK

8.3.1 NLTK 简介

NLTK

NLTK (Natural language Toolkit): 自然语言工具包, Python 编程语言实现的统计自然语言处理工具。它是由宾夕法尼亚大学计算机和信息科学的史蒂芬·伯德和爱德华·洛珀编写的。NLTK 支持 NLP 研究和教学相关的领域, 其收集的大量公开数据集、模型上提供了全面易用的接口, 涵盖了分词、词性标注 (Part-of-Speech tag, POS-tag)、命名实体识别 (Named Entity Recognition, NER)、句法分析 (Syntactic Parse) 等各项 NLP 领域的功能。广泛应用在经验语言学、认知科学、人工智能、信息检索和机器学习。在 25 个国家中已有 32 所大学将 NLTK 作为教学工具。

NLTK 模块及功能如表 8-1 所示。

表 8-1 NLTK 模块及功能

任 务	模 块	描 述
获取语料库	nltk.corpus	语料库和词典的标准化接口
字符串处理	nltk.tokenize, nltk.stem	分词、句子分解和提取主干 (不支持中文)

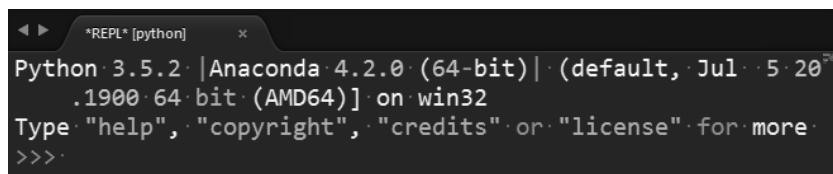
(续表)

搭配研究	nltk.collocations	t 检验、卡方检验和互信息
词性标注	nltk.tag	n-gram、backoff 和 HMM
分类	nltk.classify、nltk.cluster	决策树、最大熵、朴素贝叶斯、EM 和 K-means
分块	nltk.chunk	正则表达式、n-gram 和命名实体
解析	nltk.parse	图标、基于特征、一致性和概率性
语义解释	nltk.sem、nltk.inference	演算、模型检验
指标评测	nltk.metrics	准确率、召回率和协议系数
概率与估计	nltk.probability	频率分布和平滑概率分布
应用	nltk.app、nltk.chat	图形化关键字排序、分析器，wordNet 查看器
语言学领域工作	nltk.toolbox	处理 SIL 工具箱格式的数据

8.3.2 安装 NLTK

NLTK 安装步骤

(1) 查看 Python 版本，如图 8-1 所示。



```
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul 5 2016, 19:00:64 bit (AMD64)) on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
```

图 8-1 查看 Python 版本

(2) Windows 系统下载如下文件 nltk-3.2.1.win32.exe (<https://pan.baidu.com/s/1qYzXFPy>), 并执行 exe 文件, 会自动匹配到 Python 安装路径, 如果没有找到路径则说明 NLTK 版本不正确, 去官网 (<https://pypi.python.org/pypi/nltk/3.2.1>) 选择正确版本号下载, 如图 8-2 所示。

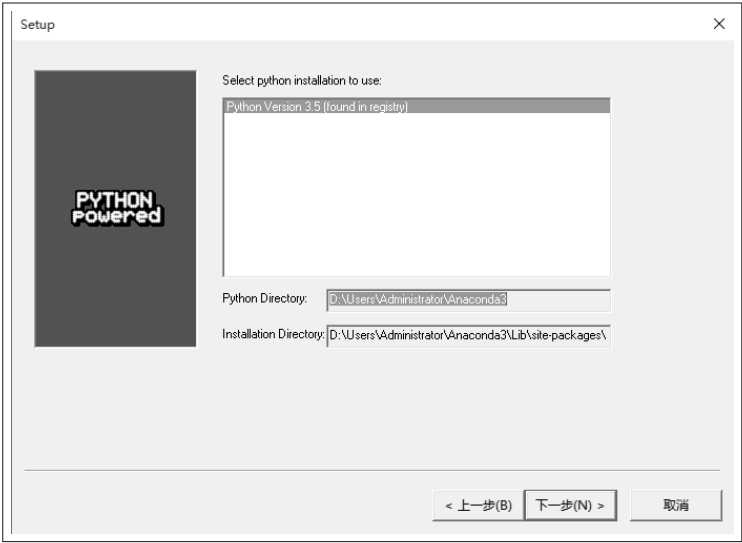


图 8-2 Python 安装根路径

(3) 安装成功后，打开 Python 编辑器，输入“import nltk”和“nltk.download()”下载 NLTK 数据包（如图 8-3 所示）。选中 book，修改下载路径“D:\ Users\ Administrator\ Anaconda3\ nltk\ data”（book 包含了数据案例和内置函数）。

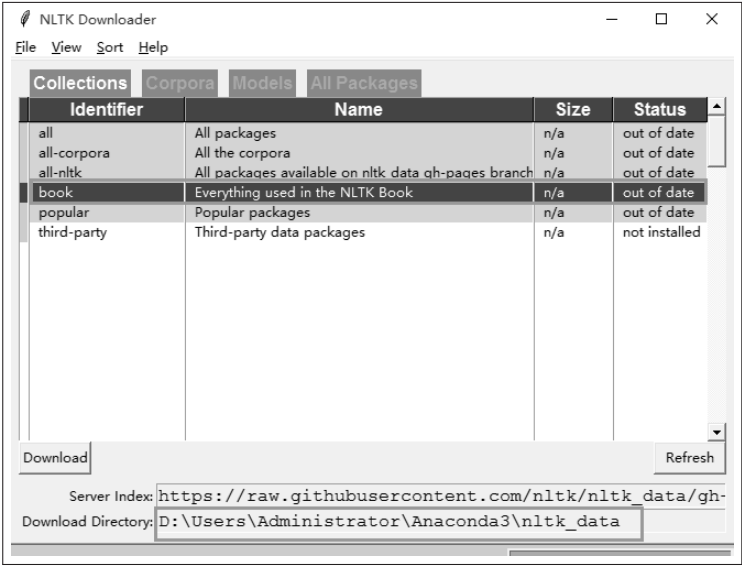


图 8-3 下载 NLTK 的 book 数据包

```
>>> import nltk
>>> nltk.download()
```

- (4) 环境变量配置：计算机 → 属性 → 高级系统设置 → 高级 → 环境变量-系统变量 → path，输入如下路径：

```
D:\Users\Administrator\Anaconda3\nltk_data
```

- (5) 打开 Python 解释器输入如下代码，出现如图 8-4 所示内容则表示安装成功。

```
>>> from nltk.book import *
```

```
>>> import nltk
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

图 8-4 成功安装 NLTK 数据包

NLTK 核心包

NLTK 核心包主要包括如下，在安装 Anaconda 时已经预安装过了。如果读者没有采用 Anaconda 安装方式，则可以单击<http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>下载安装即可。

- ◎ NLTK-Data：分析和处理语言的语料库。
- ◎ NumPy：科学计算库。
- ◎ Matplotlib：数据可视化 2D 绘图库。
- ◎ NetworkX：存储和操作由节点和边组成的网络结构函数库。

8.3.3 使用 NLTK

NLTK 加载 book 模块

如下代码，第一句是导入 NLTK 模块，第二句是导入 book 模块下的全部文件，第三句是显示 text1 文本信息，第四句是 text1 文本书名和字节数。

```
>>> import nltk
>>> from nltk.book import *
>>> text1
```

```
Text: Moby Dick by Herman Melville 1851
```

使用函数 concordance 搜索指定内容

我们想要在 text1 《白鲸记》一文中检索美国（即 “American”），输入如下代码，执行结果首先显示总共出现 12 次，它不仅可以展示全文所有出现 “American” 出现的地方及其上下文，也可以以对齐方式打印出来，便于对比分析。

```
>>> text1.concordance('America')
Displaying 12 of 12 matches:
  of the brain ." -- ULLOA ' S SOUTH AMERICA . " To fifty chosen sylphs
  of speci
, in spite of this , nowhere in all America will you find more patrician
- like
hree pirate powers did Poland . Let America add Mexico to Texas , and
pile Cuba
, how comes it that we whalemen of America now outnumber all the rest
of the b
mocracy in those parts . That great America on the other side of the
sphere , A
f age ; though among the Red Men of America the giving of the white belt
of wam
and fifty leagues from the Main of America , our ship felt a terrible
shock ,
```

```
, in the land - locked heart of our America , had yet been nurtured by
all thos
some Nor ' West Indian long before America was discovered . What other
marvels
d universally applicable . What was America in 1492 but a Loose - Fish ,
in whi
w those noble golden coins of South America are as medals of the sun and
tropic
od of the last one must be grown in America ." " Aye , aye ! a strange
sight th
>>>
```

使用函数 `similar` 查找相似上下文

我们想要在 `text1` 《白鲸记》中检索与 ‘very’ 相似的上下文，输入如下代码即可：

```
>>> text1.similar('very')
a same so last first pretty the too only other one rather as great
entire next white strange long broad
```

使用函数 `common_contexts` 共用多个词汇的上下文

当我们不满足在 `text1` 《白鲸记》中检索某个单词，而是想搜索共用多个词汇的上下文时，可输入如下代码：

```
>>> text1.common_contexts(['a', 'very'])
by_heedful was_good was_clear is_curious had_little of_great was_calm
s_queer
```

使用函数 `dispersion_plot` 离散图表示词汇分布情况

判断词在文本中的位置，从开头算起有多少词出现，可以用离散图表示，每一列代表一个单词，每一行代表有个文本。还是以 `text1` 《白鲸记》为例，代码如下。运行结果如图 8-5 所示。

```
>>> text1.dispersion_plot(["The", "Moby", "Dick", "America"])
```

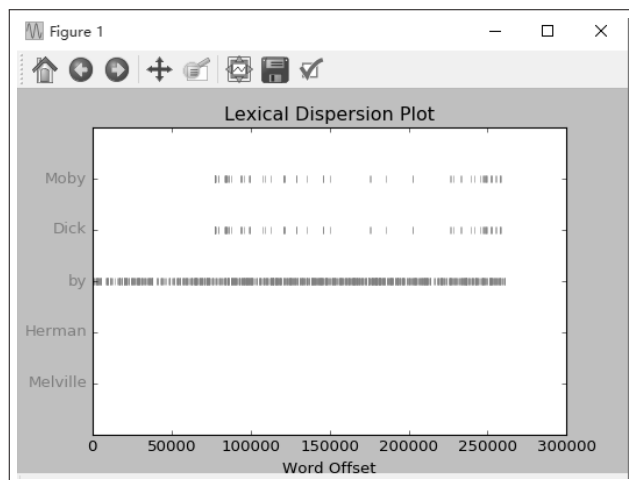


图 8-5 词汇分布情况

函数 `len()` 计数词汇：

```
>>> len(text1)
260819
```

词汇表排序：

```
>>> sorted(set(text1))
```

词汇表大小：

```
>>> len(set(text1))
```

每个词平均使用次数：

```
>>> len(text1)/len(set(text1))
```

特定词在文本中出现的次数:

```
>>> text1.count("smote")
```

特定词在文本中所占的百分比:

```
>>> 100*text1.count('a')/len(text1)
```

NLTK 搜索函数 FreqDist()

- ④ 查询文本 text1 中词汇分布情况, 诸如 the 使用了 13721 次

```
>>> fdist1=FreqDist(text1)
>>> fdist1
FreqDist({' ': 18713, 'the': 13721, '.': 6862, 'of': 6536, 'and': 6024, 'a': 4569, 'to': 4542, ';': 4072, 'in': 3916, 'that': 2982, ...})
```

- ④ 指定查询某个词的使用频率

```
>>> fdist1['whale']
906
```

- ④ 指定常用词累积频率图

fdist1.plot(50,cumulative=True), text1 中 50 个常用词的累积频率图, 这些词占了所有标识的将近一半, 如图 8-6 所示。

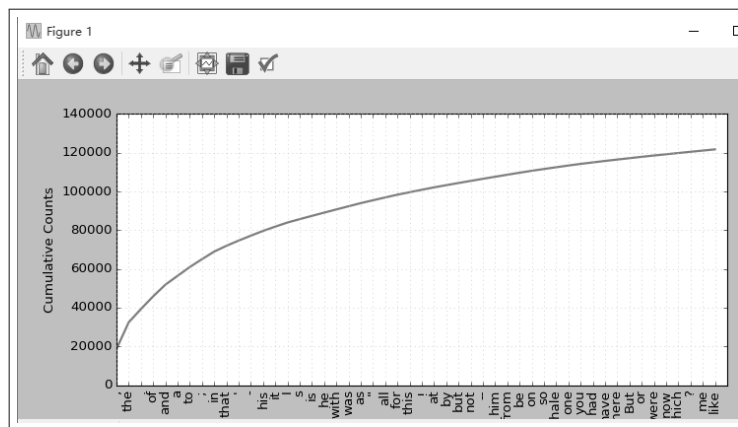


图 8-6 常用词累积频率图

注意：函数 `fdist1.hapaxes()` 低频词出现 1 次查找。

细粒度查询：

```
>>> V=set(text1)
>>> longwords=[w for w in V if len(w) > 15]
>>> sorted(longwords)
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', '
cannibalistically', 'characteristically', 'circumnavigating', '
circumnavigation', 'circumnavigations', 'comprehensiveness', '
hermaphroditical', 'indiscriminately', 'indispensableness', '
irresistibleness', 'physiognomically', 'preternaturalness', '
responsibilities', 'simultaneousness', 'subterraneousness', '
supernaturalness', 'superstitiousness', 'uncomfortableness', '
uncompromisedness', 'undiscriminating', 'uninterpenetratingly']
>>>
```

查询文本中单词长度大于 10 并且出现次数超过 10 次的：

```
>>> sorted(w for w in set(text1) if len(w) > 10 and fdist1[w] > 10)
['Nantucketer', 'Nevertheless', 'circumstance', 'circumstances', '
considerable', 'considering', 'continually', 'countenance', 'disappeared
', 'encountered', 'exceedingly', 'experienced', 'harpooneers', '
immediately', 'indifferent', 'indispensable', 'involuntarily', '
naturalists', 'nevertheless', 'occasionally', 'peculiarities', '
perpendicular', 'significant', 'simultaneously', 'straightway', '
unaccountable']
>>>
```

词语搭配和双连词

搭配：不经常在一起出现的词序列，如 `red wine` 是搭配，而 `the wine` 就不是。另一个特点就是词不能被类似的词置换，如 `maroon wine`（栗色酒）就不行。

`bigrams ()`：获取搭配，提取文本词汇的双连词。

```
>>> from nltk import bigrams
>>> from collections import Counter
>>> b = bigrams('This is a test')
>>> Counter(b)
Counter({'s', ' ': 2, ('i', 's'): 2, ('s', 't'): 1, (' ', 'i'): 1, ('', 'a'): 1, (' ', 't'): 1, ('a', ' '): 1, ('h', 'i'): 1, ('e', 's'): 1, ('t', 'e'): 1, ('T', 'h'): 1})
>>>
```

NLTK 频率分布类中定义的函数

- ⊙ `fdist=FreqDist(Samples)`，创建包含给定样本的频率分布。
- ⊙ `fdist.inc(Sample)`，增加样本。
- ⊙ `fdist['monstrous']`，计数给定样本出现的次数。
- ⊙ `fdist.freq('monstrous')`，给定样本的频率。
- ⊙ `fdist.N()`，样本总数。
- ⊙ `fdist.keys()`，以频率递减顺序排序样本链表。
- ⊙ `for sample in fdist`，以频率递减顺序遍历样本。
- ⊙ `fdist.max()`，数值最大的样本。
- ⊙ `fdist.tabulate()`，绘制频率分布表。
- ⊙ `fdist.plot()`，绘制频率分布图。
- ⊙ `fdist.plot(cumulative=True)`，绘制累积频率分布图。
- ⊙ `fdist1<fdist2`，测试样本在 `fdist1` 中出现的频率是否小于 `fdist2`。

词汇比较运算（s 代表字符串）

- ⊙ `s.startswith(t)`，测试是否以 `t` 开头。
- ⊙ `s.endswith(t)`，测试是否以 `t` 结尾。
- ⊙ `t in s`，测试 `s` 是否包含 `t`。
- ⊙ `s.islower()`，测试 `s` 所有字符是否都是小写字母。

- ◎ `s.isupper()`，测试 `s` 所有字符是否都是大写字母。
- ◎ `s.isalpha()`，测试 `s` 所有字符是否都是字母。
- ◎ `s.isalnum()`，测试 `s` 所有字符是否都是字母或数字。
- ◎ `s.isdigit()`，测试 `s` 所有字符是否都是数字。
- ◎ `s.istitle()`，测试 `s` 所有词首字母都是大写。

8.3.4 在 Python NLTK 下使用 Stanford NLP

Stanford NLP 简介

Stanford NLP：由斯坦福大学的 NLP 小组开源的 Java 实现的 NLP 工具包，同样对 NLP 领域的各个问题提供了解决办法。斯坦福大学的 NLP 小组是世界知名的研究小组，能将 NLTK 和 Stanford NLP 两个工具包结合起来使用，对于自然语言开发者再好不过了。2004 年 Steve Bird 在 NLTK 中加上了对 Stanford NLP 工具包的支持，通过调用外部的 jar 文件来使用 Stanford NLP 工具包的功能，这样一来就变得更为方便好用。

本书主要介绍 NLTK 提供的 Stanford NLP 中的以下几个功能。

- ◎ 中英文分词：StanfordTokenizer。
- ◎ 中英文词性标注：StanfordPOSTagger。
- ◎ 中英文命名实体识别：StanfordNERTagger。
- ◎ 中英文句法分析：StanfordParser。
- ◎ 中英文依存句法分析：StanfordDependencyParser。

安装配置过程中的注意事项

本书以 Python 3.5.2 和 Java version “1.8.0_111” 版本进行配置，具体安装需要注意以下几点：

- ◎ Stanford NLP 工具包需要 Java 8 及之后的版本，如果出错则检查 Java 版本。
- ◎ 本书的配置以 Stanford NLP 3.6.0 为例，如果使用的是其他版本，则注意替换相应的文件名。

- ◎ 本书的配置过程以 NLTK 3.2 为例，如果使用 NLTK 3.1，则需要注意该旧版本中 StanfordSegmenter 未实现，其余大致相同。
- ◎ 下面的配置过程是具体细节可以参照 <http://nlp.stanford.edu/software/>。

Stanford NLP 必要工具包下载

必要包下载：下载以下 3 个文件就够了，stanford NLTK 文件里面就是 Stanford NLP 工具包在 NLTK 中所依赖的 jar 包和相关文件。

- ◎ stanford NLTK (<https://pan.baidu.com/s/1nvEYdfj>)：将所有需要的包和相关文件已经打包在一起了，下面有具体讲解。
- ◎ Jar1.8 (<http://pan.baidu.com/s/1miubwq0>)：如果本机是 Java 8 以上版本，可以不用下载了。
- ◎ NLTK (<https://pan.baidu.com/s/1pKA9XuN>)：这个工具包提供 Stanford NLP 接口。

下载以上文件后，jar 如果是 1.8 的版本可以不用下载，另外两个压缩包下载到本地，解压后复制文件夹到 Python 安装主路径下，然后进入 NLTK 下通过 `python setup.py install` 命令安装即可。后面操作将路径简单修改即可（如果不能正常分词等操作，则查看 Python 是否是 3.2 以上版本，Java 是否是 8 以后版本，jar 环境变量是否配置正确）。

Stanford NLTK 目录结构如图 8-7 所示。

classifiers	2016/11/6 18:40	文件夹	
data	2016/11/3 21:56	文件夹	
models	2016/11/3 21:56	文件夹	
slf4j-api.jar	2016/1/23 15:20	JAR 文件	26 KB
stanford-corenlp-3.6.0.jar	2016/1/20 17:56	JAR 文件	7,101 KB
stanford-corenlp-3.6.0-models.jar	2016/1/20 17:56	JAR 文件	369,255 KB
stanford-ner.jar	2016/1/23 15:17	JAR 文件	4,268 KB
stanford-parser.jar	2016/1/20 18:13	JAR 文件	4,032 KB
stanford-parser-3.6.0-models.jar	2016/1/20 18:14	JAR 文件	336,581 KB
stanford-postagger.jar	2016/1/23 14:38	JAR 文件	3,338 KB
stanford-segmenter.jar	2016/1/23 15:20	JAR 文件	3,749 KB

图 8-7 Stanford NLTK 源码解析

- ◎ 分词依赖：stanford-segmenter.jar、slf4j-api.jar、data 文件夹相关子文件。
- ◎ 命名实体识别依赖：classifiers、stanford-ner.jar。
- ◎ 词性标注依赖：models、stanford-postagger.jar。

- ◎ 句法分析依赖: stanford-parser.jar、stanford-parser-3.6.0-models.jar、classifiers。
- ◎ 依存语法分析依赖: stanford-parser.jar、stanford-parser-3.6.0-models.jar、classifiers。

压缩包下载和源码分析

- ◎ 分词压缩包 StanfordSegmenter 和 StanfordTokenizer: 下载 stanford-segmenter-2015-12-09.zip (<https://pan.baidu.com/s/1kVc20ib>), 解压获取目录中的 stanford-segmenter-3.6.0.jar 复制为 stanford-segmenter.jar 和 slf4j-api.jar。
- ◎ 词性标注压缩包: 下载 stanford-postagger-full-2015-12-09.zip (<https://pan.baidu.com/s/1hrVMSE4>), 解压获取 stanford-postagger.jar。
- ◎ 命名实体识别压缩包: 下载 stanford-ner-2015-12-09.zip (<https://pan.baidu.com/s/1sk0Jb5r>), 解压获取 stanford-ner.jar 和 classifiers 文件。
- ◎ 句法分析、句法依存分析: 下载 stanford-parser-full-2015-12-09.zip (<http://pan.baidu.com/s/1nv6Q2bZ>), 解压获取 stanford-parser.jar 和 stanford-parser-3.6.0-models.jar。

Stanford NLP 的应用

(1) 分词。

StanfordSegmenter 中文分词: 下载 52nlp 改过的 NLTK 包 nltk-develop (<https://pan.baidu.com/s/1misFxnA>), 解压后将其复制到你的 Python 目录下, 进入 E:\Python\nltk-develop, 用 Python 编辑器打开 setup.py 文件, 按 F5 运行, 输入以下代码:

```
>>> from nltk.tokenize.stanford_segmenter import StanfordSegmenter
>>> segmenter = StanfordSegmenter(
    path_to_jar=r"E:\tools\stanfordNLTK\jar\stanford-segmenter.jar",
    path_to_slf4j=r"E:\tools\stanfordNLTK\jar\slf4j-api.jar",
    path_to_sihan_corpora_dict=r"E:\tools\stanfordNLTK\jar\data",
    path_to_model=r"E:\tools\stanfordNLTK\jar\data\pku.gz",
    path_to_dict=r"E:\tools\stanfordNLTK\jar\data\dict-chris6.ser.gz"
)
>>> str="我在博客园开了一个博客，我的博客名叫伏草惟存，写了一些自然语言
处理的文章。"
>>> result = segmenter.segment(str)
>>> result
```

执行结果：

[我在博客园开了一个博客，我的博客名叫伏草惟存，写了一些自然语言处理的文章。]

程序解读：StanfordSegmenter 的初始化参数说明。

- ④ path_to_jar: 用来定位 jar 包，本程序分词依赖 stanford-segmenter.jar（注：其他所有 Stanford NLP 接口都有 path_to_jar 参数）。
- ④ path_to_slf4j: 用来定位 slf4j-api.jar，作用于分词。
- ④ path_to_sihan_corpora_dict: 设定为 stanford-segmenter-2015-12-09.zip 解压后目录中的 data 目录，data 目录下有两个可用模型 pkg.gz 和 ctb.gz。需要注意的是，使用 Stanford-Segmenter 进行中文分词后，其返回结果并不是 list，而是一个字符串，各个汉语词汇在其中被空格分隔开。

StanfordTokenizer 英文分词：

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32
  bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> from nltk.tokenize import StanfordTokenizer
>>> tokenizer = StanfordTokenizer(path_to_jar=r"E:\tools\stanfordNLTK\
jar\stanford-parser.jar")
>>> sent = "Good muffins cost $3.88\nin New York. Please buy me\ntwo of
them.\nThanks."
>>> print(tokenizer.tokenize(sent))
```

运行结果：

```
{[]}`Good', `muffins', `cost', `\'$', `3.88', `in', `New', `York', `.',
`Please', `buy', `me', `two', `of', `them', `.', `Thanks', `.'{[]}
```

（2）命名实体识别。

StanfordNERTagger 英文命名实体识别：

```
>>> from nltk.tag import StanfordNERTagger
>>> eng_tagger = StanfordNERTagger(model_filename=r'E:\tools\
stanfordNLTK\jar\classifiers\english.all.3class.distsim.crf.ser.gz',
path_to_jar=r'E:\tools\stanfordNLTK\jar\stanford-ner.jar')
>>> print(eng_tagger.tag('Rami Eid is studying at Stony Brook University
in NY'.split()))
```

运行结果:

```
[('Rami', 'PERSON'), ('Eid', 'PERSON'), ('is', 'O'), ('studying', 'O'),
('at', 'O'), ('Stony', 'ORGANIZATION'), ('Brook', 'ORGANIZATION'), ('
University', 'ORGANIZATION'), ('in', 'O'), ('NY', 'O')]
```

StanfordNERTagger 中文命名实体识别:

```
>>> result
'四川省 成都 信息 工程 大学 我 在 博客 园 开 了 一 个 博 客 , 我 的 博 客
名 叫 伏 草 惟 存 , 写 了 一 些 自 然 语 言 处 理 的 文 章 。 \r\n'
>>> from nltk.tag import StanfordNERTagger
>>> chi_tagger = StanfordNERTagger(model_filename=r'E:\tools\
stanfordNLTK\jar\classifiers\chinese.misc.distsim.crf.ser.gz',
path_to_jar=r'E:\tools\stanfordNLTK\jar\stanford-ner.jar')
>>> for word, tag in chi_tagger.tag(result.split()):
    print(word,tag)
```

运行结果:

四川省	ORG
成都	ORG
信息	ORG
工程	ORG
大学	ORG
我	O
在	O
博客	O

园	0
开	0
了	0
一个	0
博客	0
,	0
我	0
的	0
博客	0
...	...

(3) 词性标注。

StanfordPOSTagger 英文词性标注:

```
>>> from nltk.tag import StanfordPOSTagger
>>> eng_tagger = StanfordPOSTagger(model_filename=r'E:\tools\
stanfordNLTK\jar\models\english-bidirectional-distsim.tagger',
path_to_jar=r'E:\tools\stanfordNLTK\jar\stanford-postagger.jar')
>>> print(eng_tagger.tag('What is the airspeed of an unladen swallow ?'.
split()))
```

运行结果:

```
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('airspeed', 'NN'), ('of', 'IN')
, ('an', 'DT'), ('unladen', 'JJ'), ('swallow', 'VB'), ('?', '.')]

```

StanfordPOSTagger 中文词性标注:

```
>>> from nltk.tag import StanfordPOSTagger
>>> chi_tagger = StanfordPOSTagger(model_filename=r'E:\tools\
stanfordNLTK\jar\models\chinese-distsim.tagger', path_to_jar=r'E:\tools\
stanfordNLTK\jar\stanford-postagger.jar')
>>> result
'四川省 成都 信息 工程 大学 我 在 博客 园 开 了 一 个 博 客 , 我 的 博 客
名 叫 伏 草 惟 存 , 写 了 一 些 自 然 语 言 处 理 的 文 章 。 \r\n'
>>> print(chi_tagger.tag(result.split()))
```

运行结果：

四川省	NR
成都	NR
信息	NN
工程	NN
大学	NN
我	PN
在	P
博客	NN
园	NN
开	VV
了	AS
一个	CD
博客	NN
,	PU
我	PN
的	DEG
博客	NN
...	...

(4) 句法分析。

StanfordParser 英文句法分析：

```
>>> from nltk.parse.stanford import StanfordParser
>>> eng_parser = StanfordParser(r"E:\tools\stanfordNLTK\jar\stanford-
parser.jar",r"E:\tools\stanfordNLTK\jar\stanford-parser-3.6.0-models.jar
",r"E:\tools\stanfordNLTK\jar\classifiers\englishPCFG.ser.gz")
>>> print(list(eng_parser.parse("the quick brown fox jumps over the lazy
dog".split())))
```

运行结果：

```
[Tree('ROOT',[Tree('NP',[Tree('DT',['the'])],Tree('JJ',['quick'])],Tree('
JJ',['brown'])],Tree('NN',['for']))],Tree('NP',[Tree('NP',[Tree('NNS',['
jumps'])])],Tree('PP',[Tree('IN',['over'])],Tree('NP',[Tree('DT',['the'])],
Tree('JJ',['lazy']),Tree('NN',['dog'])])))))]
```

```
((('for', 'NN'), 'dep', ('jumps', 'NNS'))
(('jumps', 'NNS'), 'nmod', ('dog', 'NN'))
(('dog', 'NN'), 'case', ('over', 'IN'))
(('dog', 'NN'), 'det', ('the', 'DT'))
(('dog', 'NN'), 'amod', ('lazy', 'JJ'))
```

StanfordDependencyParser 中文依存句法分析:

```
>>> from nltk.parse.stanford import StanfordDependencyParser
>>> chi_parser = StanfordDependencyParser(r"E:\tools\stanfordNLTK\jar\
stanford-parser.jar", r"E:\tools\stanfordNLTK\jar\stanford-parser-3.6.0-
models.jar", r"E:\tools\stanfordNLTK\jar\classifiers\chinesePCFG.ser.gz")
>>> res = list(chi_parser.parse(u'四川 已 成为 中国 西部 对外开放 中 升
起 的 一 颗 明星'.split()))
>>> for row in res[0].triples():
    print(row)
```

运行结果:

```
((('成为', 'VV'), 'nsubj', ('四川', 'NR'))
(('成为', 'VV'), 'advmod', ('已', 'AD'))
(('成为', 'VV'), 'dobj', ('明星', 'NN'))
(('明星', 'NN'), 'dep', ('对外开放', 'NN'))
(('对外开放', 'NN'), 'nn', ('中国', 'NR'))
(('对外开放', 'NN'), 'nn', ('西部', 'NN'))
(('对外开放', 'NN'), 'case', ('中', 'LC'))
(('明星', 'NN'), 'relcl', ('升起', 'VV'))
(('升起', 'VV'), 'mark', ('的', 'DEC'))
(('明星', 'NN'), 'clf', ('颗', 'M'))
(('颗', 'M'), 'nummod', ('一', 'CD'))
```

8.4 获取语料库

语料库是语料库语言学研究的基础资源,也是经验主义语言研究的主要资源。应用于词典编纂、语言教学、传统语言研究、自然语言处理中基于统计或实例的研究等方面。

8.4.1 国内外著名语料库

多语料库

- ◎ 点通多语言语音语料库: <https://archive.is/20121208123647/http://www.dmcbbc.com.cn/>。
- ◎ 宾州大学语料库: <https://www ldc.upenn.edu/>。
- ◎ Wikipedia XML 语料库: <http://www-connex.lip6.fr/~denoyer/wikipediaXML/>。
- ◎ 中英双语知识本体词网 (<http://bow.sinica.edu.tw/>): 结合词网、知识本体与领域标记的词汇知识库。

英文语料库

- ◎ 古滕堡语料库: <http://www.gutenberg.org/>。
- ◎ 语料库在线: <http://www.aihanyu.org/cncorpus/index.aspx#P0>。

中文语料库

- ◎ 搜狗实验室新闻 | 互联网数据: <http://www.sogou.com/labs/>。
- ◎ 北京大学语言研究中心: <http://ccl.pku.edu.cn/term.asp>。
- ◎ 计算机语言研究所: http://www.icl.pku.edu.cn/icl_res/。
- ◎ 数据堂: <http://www.datatang.com/>。
- ◎ 中央研究院平衡语料库 (<https://www.sinica.edu.tw/SinicaCorpus/>): 专门针对语言分析而设计的, 每个文句都依词断开并标示词类。语料的搜集也尽量做到现代汉语分配在不同的主题和句式上, 是现代汉语无穷多的语句中一个代表性的样本。现有语料库主要针对语言分析而设计, 由中央研究院信息所、语言所词库小组完成, 内含有简介、使用说明, 现行的语料库是 4.0 版本。
- ◎ LIVAC 汉语共时语料库: <http://www.livac.org/index.php?lang=tc>。
- ◎ 兰开斯特大学汉语平衡语料库: <http://www.lancaster.ac.uk/fass/projects/corpus/>。
- ◎ 兰开斯特—洛杉矶汉语口语语料库: <http://www.lancaster.ac.uk/fass/projects/corpus/>。
- ◎ 语料库语言学在线: <http://www.corpus4u.org/>。

- ◎ 北京森林工作室汉语句义结构标注语料库：<http://www.isclab.org.cn/csa/bfs-ctc.htm>。
- ◎ 国家语委现代汉语语料库（<http://www.cncorpus.org/>）：现代汉语通用平衡语料库现在重新开放网络查询了。重开后的在线检索速度更快，功能更强，同时提供检索结果下载。现代汉语语料库在线提供免费检索的语料约 2000 万字，为分词和词性标注语料。
- ◎ 古代汉语语料库（<http://www.cncorpus.org/login.aspx>）：网站现在增加了一亿字的古代汉语生语料，研究古代汉语的也可以去查询和下载。网站同时还提供了分词、词性标注软件，词频统计、字频统计软件。基于国家语委语料库的字频词频统计结果和发布的词表等进行建库，以供学习研究语言文字的同学和老师使用。
- ◎ 《人民日报》标注语料库（http://www.icl.pku.edu.cn/icl_res/）：《人民日报》标注语料库中一半的语料（1998 年上半年）共 1300 万字，已经通过《人民日报》新闻信息中心公开并提供许可使用权。其中一个月的语料（1998 年 1 月）近 200 万字在互联网上公布，可自由下载。
- ◎ 古汉语语料库（<http://www.sinica.edu.tw/ftms-bin/ftmsw>）：古汉语语料库包含以下五个语料库——上古汉语、中古汉语（含大藏经）、近代汉语、出土文献、其他。部分数据取自史语所汉籍全文数据库，故两者间内容略有重叠。此语料库之出土文献语料库，全部取自史语所汉简小组所制作的数据库。
- ◎ 近代汉语标记语料库（http://www.sinica.edu.tw/Early_Mandarin/）：为应对汉语史研究需求而建构的语料库。目前语料库所搜集的语料已涵盖上古汉语（先秦至西汉）、中古汉语（东汉魏晋南北朝）、近代汉语（唐五代以后）大部分的重要语料，并陆续开放使用；在标记语料库方面，上古汉语及近代汉语都已有部分语料完成标注的工作，并视结果逐步提供上线检索。
- ◎ 树图数据库（<http://treebank.sinica.edu.tw/>）。
- ◎ 搜文解字（<http://words.sinica.edu.tw/>）：包含「搜词寻字」、「文学之美」、「游戏解惑」、「古文字的世界」四个单元，可由部件、部首、字、音、词互查，并可查询在四书、老、庄、唐诗中的出处，以及直接链接到出处并阅读原文。
- ◎ 文国寻宝记（<http://www.sinica.edu.tw/wen/>）：在搜文解字的基础之上，以华语文学习者对象，进一步将字、词、音的检索功能与国编、华康、南一等三种版本的国小国语课本结合。与唐诗三百首、宋词三百首、红楼梦、水浒传等文学典籍结合，提供网络上国语文学习的素材。
- ◎ 唐诗三百首（<http://cls.admin.yzu.edu.tw/300/>）：以中小学学生为主要使用对象，提供吟唱、绘画、书法等多媒体数据，文字数据包含作者生平、读音标注、翻译、批注、评

注、典故出处等资料；检索点包含作者、诗题、诗句、综合资料、体裁分类等；检索结果可以列出全文，并选择标示相关之文字及多媒体数据。并提供了一套可以自动检查格律、韵脚、批改的「依韵入诗格律自动检测索引教学系统」，协助孩子们依韵作诗，协助教师批改习作。

- ◎ 汉籍电子文献 (<http://www.sinica.edu.tw/tdbproj/handy1/>)：包含整部 25 史整部阮刻 13 经、超过 2000 万字的台湾史料、1000 万字的大正藏及其他典籍。
- ◎ 红楼梦网络教学研究数据中心 (<http://cls.hs.yzu.edu.tw/HLM/home.htm>)：元智大学中国文学网络系统研究室所开发的「网络展书读—中国文学网络系统」，为研究中心负责人罗凤珠老师主持。红楼梦是其中一个子系统，其他还包括善本书、诗经、唐宋诗词、作诗填词等子系统。此网站为国内最大的在线中国文学研究数据库，提供用户最完整的中国文学研究数据。
- ◎ 中国传媒大学文本语料库检索系统 (<http://ling.cuc.edu.cn/RawPub/>)。
- ◎ 在线分词标注系统 (<http://ling.cuc.edu.cn/cucseg/>)。
- ◎ 新词语研究资源库 (<http://ling.cuc.edu.cn/newword/web/index.asp>)。
- ◎ 音视频语料检索系统 (<http://ling.cuc.edu.cn/mmcpub>)。
- ◎ 哈工大信息检索研究室对外共享语料库资源 (http://ir.hit.edu.cn/demo/ltpl/Sharing_Plan.htm)：该语料库为汉英双语语料库，10 万对齐双语句对，文本书件格式，同义词词林扩展版，77343 条词语，秉承《同义词词林》的编撰风格。同时采用五级编码体系，多文档自动文摘语料库，40 个主题，文本书件格式，同一主题下是同一事件的不同报道。汉语依存树库，不带关系 5 万句，带关系 1 万句；LTM 化，分词、词性、句法部分人工标注，可以图形化查看，问答系统问题集，6264 句；已标注问题类型，LTM 化，分词、词性、句法、词义、浅层语义等程序处理得到，单文档自动文摘语料库共 211 篇。
- ◎ 清华大学汉语均衡语料库 TH-ACorpus (<http://www.lits.tsinghua.edu.cn/ainlp/source.htm>)。
- ◎ 中国科学院计算技术研究所，跨语言语料库 (<http://mtgroup.ict.ac.cn/new/resource/index.php>) 目前的双语句对数据库中有约 180000 对已对齐的中英文句子。本数据库支持简单的中英文查询服务。查询结果包括句对编号、中文句子、英文句子、句对来源等。

8.4.2 网络数据获取

从网络和硬盘访问文本（在线获取伤寒杂病论）：

```
>>> from __future__ import division
>>> import nltk,re,pprint
>>> from urllib.request import urlopen
>>> url=r'http://www.gutenberg.org/files/24272/24272-0.txt'
>>> raw=urlopen(url).read()
>>> raw = raw.decode('utf-8')
>>> len(raw)
70306
>>> raw[2000:2500]
```

运行结果:

'。\\r\\n陽脈浮大而濡，陰脈浮大而濡，陰脈與陽脈同等者，名曰緩也。脈浮而緊者，名\\r\\n曰弦也。弦者狀如弓弦，按之不移也。脈緊者，如轉索無常也。\\r\\n脈弦而大，弦則為減，大則為芤。減則為寒，芤則為虛。寒虛相搏，此名為革。\\r\\n婦人則半產、漏下，男子則亡血、失精。\\r\\n問曰：病有戰而汗出，因得解者，何也？答曰：脈浮而緊，按之反芤，此為本虛，\\r\\n故當戰而汗出也。其人本虛，是以發戰。以脈浮，故當汗出而解也。\\r\\n若脈浮而數，按之不芤，此人本不虛；若欲自解，但汗出耳，不發戰也。\\r\\n問曰：病有不戰而汗出解者，何也？答曰：脈大而浮數，故知不戰汗出而解也。\\r\\n問曰：病有不戰，不汗出而解者，何也？答曰：其脈自微，此以曾經發汗、若吐、\\r\\n若下、若亡血，以內無津液，此陰陽自和，必自愈，故不戰、不汗出而解也。\\r\\n問曰：傷寒三日，脈浮數而微，病人身涼和者，何也？答曰：此為欲解也。解以\\r\\n夜半。脈浮而解者，濺然汗出也；脈數而解者，必能食也；脈微而解者，必大汗\\r\\n出也。\\r\\n問曰：病脈，欲知愈未愈者，何以別之？答曰：寸口、關上、尺中三處，大小、\\r\\n浮沉、遲數同等，雖有寒熱不解者，此脈陰陽為和平，雖劇當愈。\\r\\n立夏得洪（一作浮）'

在线获取处理 HTML 文本（红楼梦）:

```
>>> import re,nltk
>>> from urllib.request import urlopen
>>> url='http://www.gutenberg.org/cache/epub/24264/pg24264-images.html'
>>> html=urlopen(url).read()
>>> html=html.decode('utf-8')
>>> html[5000:5500]
```

运行结果:

'五次，纂成目錄，分出章回，則題曰《金陵十二釵》。并題一絕云：\r\n\u3000\u3000滿紙荒唐言，一把辛酸淚！\r\n\u3000\u3000都云作者痴，誰解其中味？\r\n\u3000\u3000出則既明，且看石上是何故事。按那石上書云：\r\n\u3000\u3000當日地陷東南，這東南一隅有處曰姑蘇，有城曰閶門者，最是紅塵中一\r\n\u3000二等富貴風流之地。這閶門外有個十里街，街內有個仁清巷，巷內有個古廟\r\n\u3000，因地方窄狹，人皆呼作葫蘆廟。廟旁住著一家鄉宦，姓甄，名費，字士隱\r\n\u3000。嫡妻封氏，情性賢淑，深明禮義。家中雖不甚富貴，然本地便也推他為望\r\n\u3000族了。因這甄士隱稟性恬淡，不以功名為念，每日只以觀花修竹，酌酒吟詩\r\n\u3000為樂，倒是神仙一流人品。只是一件不足：如今年已半百，膝下無兒，只有\r\n\u3000一女，乳名喚作英蓮，年方三歲。 \r\n\u3000\u3000一日，炎夏永晝，士隱于書房間坐，至手倦拋書，伏几少憩，不覺朦朧\r\n\u3000睡去。夢至一處，不辨是何地方。忽見那廟來了一僧一道，且行且談。只听\r\n\u3000道人問道：“你攜了這蠢物，意欲何往？”那僧笑道：“你放心，如今現有\r\n\u3000一段風流公案正該了結，這一干風流冤家，尚未投胎入世。趁此機會，就將\r\n\u3000\r\n\u3000此蠢物夾帶于中，使他去經歷經歷。”那道人道：“原來近日風流冤孽又將\r\n\u3000\r\n\u3000造劫歷世去不成？但'

处理 RSS 订阅:

```
>>> import feedparser #feedparser需要在python库中下载,或者在cmd中'pip
install feedparser'
>>> llog=feedparser.parse(url)
```

相关正则知识

- ◎ \d，匹配一个数字。
- ◎ \w，匹配一个字母或者数字。
 - ◎ *，任意字符（包括 0 个）。
 - ◎ +，至少一个字符。
- ◎ ?，0 个或 1 个字符。
- ◎ {n}，n 个字符。
- ◎ {n,m}，n ~ m 个字符。

- ◎ `\s`，匹配一个空格。
- ◎ `\s+`，至少有一个空格。
- ◎ `\d{3,8}`，表示 3 ~ 8 个数字，例如，`' 1234567 '`。
- ◎ `[0-9a-zA-Z_]`，匹配一个数字、字母或者下画线。
- ◎ `[0-9a-zA-Z_]+`，匹配至少由一个数字、字母或者下画线组成的字符串，比如 `' a100 '`，`' 0_Z '`，`' Py3000 '`，等等。
- ◎ `[a-zA-Z_][0-9a-zA-Z_]*`，可以匹配由字母或下画线开头，后接任意由一个数字、字母或者下画线组成的字符串，也就是 Python 合法的变量。
- ◎ `[a-zA-Z_][0-9a-zA-Z_]{0, 19}`，更精确地限制了变量的长度是 1 ~ 20 个字符（前面 1 个字符 + 后面最多 19 个字符）。
- ◎ `A|B`，可以匹配 A 或 B，所以 `(P|p)ython` 可以匹配 `' Python '` 或者 `' python '`。
- ◎ `^`，表示行的开头，`\d` 表示必须以数字开头。
- ◎ `-`，表示行的结束，`\d` 表示必须以数字结束。

8.4.3 NLTK 获取语料库

古腾堡语料库

- (1) 直接获取语料库的所有文本：`nlk.corpus.gutenberg.fileids()`。

```
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
```

运行结果：

```
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-
kjbv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.
txt', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt
', 'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-
moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt', '
shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt
']
```

(2) 导入包获取语料库的所有文本。

```
>>> from nltk.corpus import gutenberg
>>> gutenberg.fileids()
```

运行结果:

```
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt']
```

(3) 查找某个文本。

```
>>> persuasion=nltk.corpus.gutenberg.words("austen-persuasion.txt")
>>> len(persuasion)
98171
>>> persuasion[:200]
```

运行结果:

```
['[', 'Persuasion', 'by', 'Jane', 'Austen', '1818', ...]
```

网络和聊天文本

(1) 获取网络聊天文本。

```
>>> from nltk.corpus import webtext
>>> for fileid in webtext.fileids():
    print(fileid,webtext.raw(fileid))
```

(2) 查看网络聊天文本信息。

```
>>> for fileid in webtext.fileids():
    print(fileid,len(webtext.words(fileid)),len(webtext.raw(fileid)),len(
        webtext.sents(fileid)),webtext.encoding(fileid))
```

运行结果：

```
firefox.txt 102457 564601 1142 ISO-8859-2
grail.txt 16967 65003 1881 ISO-8859-2
overheard.txt 218413 830118 17936 ISO-8859-2
pirates.txt 22679 95368 1469 ISO-8859-2
singles.txt 4867 21302 316 ISO-8859-2
wine.txt 31350 149772 2984 ISO-8859-2
```

(3) 即时消息聊天会话语料库。

```
>>> from nltk.corpus import nps_chat
>>> chatroom = nps_chat.posts('10-19-20s_706posts.xml')
>>> chatroom[123]
```

运行结果：

```
['i', 'do', 'n't', 'want', 'hot', 'pics', 'of', 'a', 'female', ',', 'I',
 'can', 'look', 'in', 'a', 'mirror', '.']
```

布朗语料库

(1) 查看语料信息。

```
>>> from nltk.corpus import brown
>>> brown.categories()
```

运行结果：

```
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', '
hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', '
reviews', 'romance', 'science_fiction']
```

(2) 比较文体中情态动词的用法。

```
>>> import nltk
>>> from nltk.corpus import brown
>>> new_texts=brown.words(categories='news')
>>> fdist=nltk.FreqDist([w.lower() for w in new_texts])
>>> modals=['can','could','may','might','must','will']
>>> for m in modals:
    print(m + ': ',fdist[m])
```

运行结果:

```
can: 94
could: 87
may: 93
might: 38
must: 53
will: 389
```

(3) NLTK 条件概率分布函数。

```
>>> cfd=nltk.ConditionalFreqDist((genre,word) for genre in brown.categories()
    for word in brown.words(categories=genre))
>>> genres=['news','religion','hobbies','science_fiction','romance','humor']
>>> modals=['can','could','may','might','must','will']
>>> cfd.tabulate(condition=genres,samples=modals)
```

运行结果:

	can	could	may	might	must	will
adventure	46	151	5	58	27	50
belles_lettres	246	213	207	113	170	236
editorial	121	56	74	39	53	233
fiction	37	166	8	44	55	52
government	117	38	153	13	102	244
hobbies	268	58	131	22	83	264
humor	16	30	8	8	9	13
learned	365	159	324	128	202	340

lore	170	141	165	49	96	175
mystery	42	141	13	57	30	20
news	93	86	66	38	50	389
religion	82	59	78	12	54	71
reviews	45	40	45	26	19	58
romance	74	193	11	51	45	43
science_fiction	16	49	4	12	8	16

路透社语料库

- (1) 包括 10788 个新闻文档，共计 130 万字，这些文档分 90 个主题，安装训练集和测试分组，编号为 'test/14826' 的文档属于测试文档。

```
>>> from nltk.corpus import reuters
>>> print(reuters.fileids()[:50])
```

运行结果：

```
['test/14826', 'test/14828', 'test/14829', 'test/14832', 'test/14833', 'test/14839', 'test/14840', 'test/14841', 'test/14842', 'test/14843', 'test/14844', 'test/14849', 'test/14852', 'test/14854', 'test/14858', 'test/14859', 'test/14860', 'test/14861', 'test/14862', 'test/14863', 'test/14865', 'test/14867', 'test/14872', 'test/14873', 'test/14875', 'test/14876', 'test/14877', 'test/14881', 'test/14882', 'test/14885', 'test/14886', 'test/14888', 'test/14890', 'test/14891', 'test/14892', 'test/14899', 'test/14900', 'test/14903', 'test/14904', 'test/14907', 'test/14909', 'test/14911', 'test/14912', 'test/14913', 'test/14918', 'test/14919', 'test/14921', 'test/14922', 'test/14923', 'test/14926']
```

- (2) 查看语料包括的前 100 个类别。

```
print(reuters.categories()[:100])
```

运行结果：

```
[ 'acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconut', 'coconut-oil', 'coffee', 'copper', 'copra-cake', 'corn', 'cotton', 'cotton-oil', 'cpi', 'cpu', 'crude', 'dfl', 'dlr', 'dmk', 'earn', 'fuel', 'gas', 'gnp', 'gold', 'grain', 'groundnut', 'groundnut-oil', 'heat', 'hog', 'housing', 'income', 'instal-debt', 'interest', 'ipi', 'iron-steel', 'jet', 'jobs', 'l-cattle', 'lead', 'lei', 'lin-oil', 'livestock', 'lumber', 'meal-feed', 'money-fx', 'money-supply', 'naphtha', 'nat-gas', 'nickel', 'nkr', 'nzdlr', 'oat', 'oilseed', 'orange', 'palladium', 'palm-oil', 'palmkernel', 'pet-chem', 'platinum', 'potato', 'propane', 'rand', 'rape-oil', 'rapeseed', 'reserves', 'retail', 'rice', 'rubber', 'rye', 'ship', 'silver', 'sorghum', 'soy-meal', 'soy-oil', 'soybean', 'strategic-metal', 'sugar', 'sun-meal', 'sun-oil', 'sunseed', 'tea', 'tin', 'trade', 'veg-oil', 'wheat', 'wpi', 'yen', 'zinc']
```

- (3) 查看某个编号的语料下类别尺寸。

```
>>> reuters.categories('training/9865')
```

运行结果:

```
['barley', 'corn', 'grain', 'wheat']
```

- (4) 查看某几个联合编号下语料的类别尺寸。

```
>>> reuters.categories(['training/9865', 'training/9880'])
```

运行结果:

```
['barley', 'corn', 'grain', 'money-fx', 'wheat']
```

- (5) 查看哪些编号的文件属于指定的类别。

```
>>> reuters.fileids('barley')
```

运行结果:

```
['test/15618', 'test/15649', 'test/15676', 'test/15728', 'test/15871', 'test/15875', 'test/15952', 'test/17767', 'test/17769', 'test/18024', 'test/18263', 'test/18908', 'test/19275', 'test/19668', 'training/10175', 'training/1067', 'training/11208', 'training/11316', 'training/11885', 'training/12428', 'training/13099', 'training/13744', 'training/13795', 'training/13852', 'training/13856', 'training/1652', 'training/1970', 'training/2044', 'training/2171', 'training/2172', 'training/2191', 'training/2217', 'training/2232', 'training/3132', 'training/3324', 'training/395', 'training/4280', 'training/4296', 'training/5', 'training/501', 'training/5467', 'training/5610', 'training/5640', 'training/6626', 'training/7205', 'training/7579', 'training/8213', 'training/8257', 'training/8759', 'training/9865', 'training/9958']
```

就职演说语料库

(1) 查看语料信息。

```
>>> from nltk.corpus import inaugural
>>> len(inaugural.fileids())
56
>>> inaugural.fileids()
```

运行结果:

```
['1789-Washington.txt', '1793-Washington.txt', '1797-Adams.txt', '1801-Jefferson.txt', '1805-Jefferson.txt', '1809-Madison.txt', '1813-Madison.txt', '1817-Monroe.txt', '1821-Monroe.txt', '1825-Adams.txt', '1829-Jackson.txt', '1833-Jackson.txt', '1837-VanBuren.txt', '1841-Harrison.txt', '1845-Polk.txt', '1849-Taylor.txt', '1853-Pierce.txt', '1857-Buchanan.txt', '1861-Lincoln.txt', '1865-Lincoln.txt', '1869-Grant.txt', '1873-Grant.txt', '1877-Hayes.txt', '1881-Garfield.txt', '1885-Cleveland.txt', '1889-Harrison.txt', '1893-Cleveland.txt', '1897-McKinley.txt', '1901-McKinley.txt', '1905-Roosevelt.txt', '1909-Taft.txt', '1913-Wilson.txt', '1917-Wilson.txt', '1921-Harding.txt', '1925-
```

```
Coolidge.txt', '1929-Hoover.txt', '1933-Roosevelt.txt', '1937-Roosevelt.
txt', '1941-Roosevelt.txt', '1945-Roosevelt.txt', '1949-Truman.txt',
'1953-Eisenhower.txt', '1957-Eisenhower.txt', '1961-Kennedy.txt', '1965-
Johnson.txt', '1969-Nixon.txt', '1973-Nixon.txt', '1977-Carter.txt',
'1981-Reagan.txt', '1985-Reagan.txt', '1989-Bush.txt', '1993-Clinton.txt
', '1997-Clinton.txt', '2001-Bush.txt', '2005-Bush.txt', '2009-Obama.txt
']
```

(2) 查看演说语料的年份。

```
>>> [fileid[:4] for fileid in inaugural.fileids()]
```

运行结果：

```
['1789', '1793', '1797', '1801', '1805', '1809', '1813', '1817', '1821',
'1825', '1829', '1833', '1837', '1841', '1845', '1849', '1853', '1857',
'1861', '1865', '1869', '1873', '1877', '1881', '1885', '1889', '1893',
'1897', '1901', '1905', '1909', '1913', '1917', '1921', '1925', '1929',
'1933', '1937', '1941', '1945', '1949', '1953', '1957', '1961', '1965',
'1969', '1973', '1977', '1981', '1985', '1989', '1993', '1997', '2001',
'2005', '2009']
```

(3) 条件概率分布。

```
>>> import nltk
>>> cfd=nltk.ConditionalFreqDist((target,fileid[:4]) for fileid in
inaugural.fileids() for w in inaugural.words(fileid) for target in ['
america','citizen'] if w.lower().startswith(target))
>>> cfd.plot()
```

运行结果如图 8-8 所示。

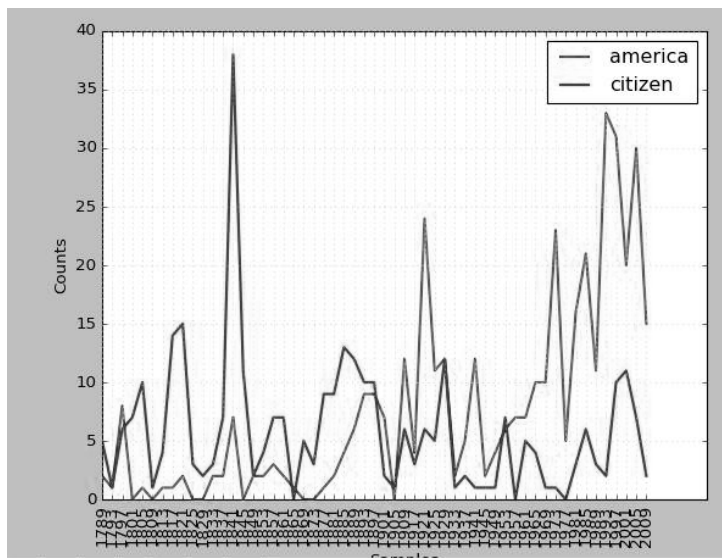


图 8-8 运行结果

8.5 综合案例：走进大秦帝国

8.5.1 数据采集和预处理

下载孙皓晖先生的《大秦帝国》.zip 文件，里面包含 5 个文件，分别是 30852 词的 p1.txt、70046 词的 p2.txt、111970 词的 p3.txt、1182769 词的 p5.txt、419275 词的 p10.txt。本书节选了大秦帝国第一部 673167 字的 dqdg.txt。

8.5.2 构建本地语料库

构建自己语料库：

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root=r'E:\dict'
>>> wordlists=PlaintextCorpusReader(corpus_root,'.*')
```



```
>>> wordlists.fileids()
['dqdg.txt', 'q0.txt', 'q1.txt', 'q10.txt', 'q2.txt', 'q3.txt', 'q5.txt',
 'text.txt']
>>> len(wordlists.words('text.txt')) #如果输入错误或者格式不正确,
notepad++转换下编码格式即可
152389
```

语料库信息如图 8-9 所示。

名称	修改日期	类型
dqdg.txt	2016/10/20 17:02	文本文档
q0.txt	2016/10/24 16:15	文本文档
q1.txt	2016/10/17 15:10	文本文档
q2.txt	2016/10/17 15:11	文本文档
q3.txt	2016/10/17 15:12	文本文档
q5.txt	2016/10/17 15:27	文本文档
q10.txt	2016/10/17 15:21	文本文档
text.txt	2016/10/20 20:17	文本文档

图 8-9 语料库信息

构建完成自己语料库之后，利用 Python NLTK 内置函数都可以完成对应的操作。换言之，其他语料库的方法在自己语料库中通用。唯一的问题是，部分方法 NLTK 是针对英文语料的，中文语料不通用（典型的的就是分词）。这个问题的解决方法很多，诸如通过插件等在 NLTK 工具包内完成对中文的支持。另外也可以在 NLTK 中利用 Stanford NLP 工具包完成对自己语料的操作，这部分知识上节讲解过。

8.5.3 大秦帝国语料操作

打开 Python 编辑器，导出 NLTK，并统计大秦帝国第一部共计多少字（注：在读取文本时，Python 3.5 IDLE 执行起来比较慢，采用 pycharm 效率就高很多了）。

```
>>> with open(r"E:\dict\dqdg.txt", "r+") as f:
    str=f.read()
```

查看大秦帝国第一部总共有多大的用字量，即不重复词和符合的尺寸：

```
>>> len(set(str))
4053

>>> len(str)/len(set(str))
166.09104367135456
```

实验可知用了 4053 个尺寸的词汇表，平均每个词使用了 166 次，那么常用词分布如何呢？既然是大秦帝国，那么秦字使用了多少次呢？

```
>>> str.count("秦")
3538

>>> str.count("大秦")
14

>>> str.count("国")
6536
```

可以知道，秦用词 3538 次，大秦用了 14 次，因为讲的各国之间的事情，“国”也是高频词（6536 次）。如上所述，大秦帝国第一部总词汇表 673167，那么整个词汇累积分布如何？

```
>>> fdist=FreqDist(str)
>>> fdist.plot()
```

运行结果如图 8-10 所示。整本书的累积分布情况如图 8-11 所示。

图 8-10 的横坐标表示词的序列，纵坐标表示词频。表说明词频大于 5000 的非常少，说明高频词不多，低频词特别多。后面进一步探究。

分析图 8-11 我们不难发现，3 万以下时低频词大于 30%，高频词大于 1.4%，中频占 68.6%（偏低中频 2 万左右占 29.85%，偏高中频占 8.96%）。

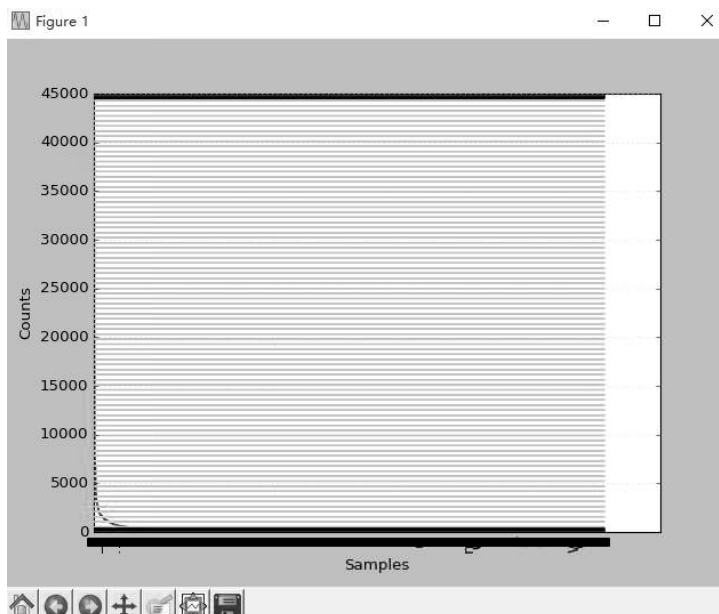


图 8-10 运行结果

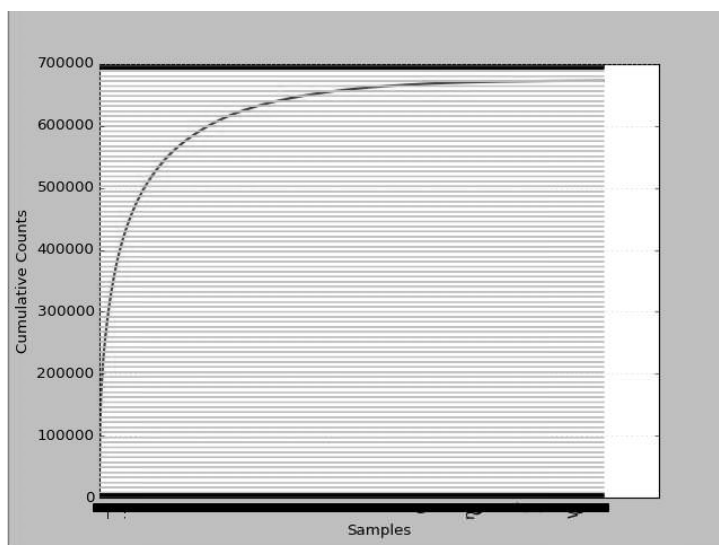


图 8-11 整本书的累积分布

高频率的 1000 个词如图 8-12 所示。

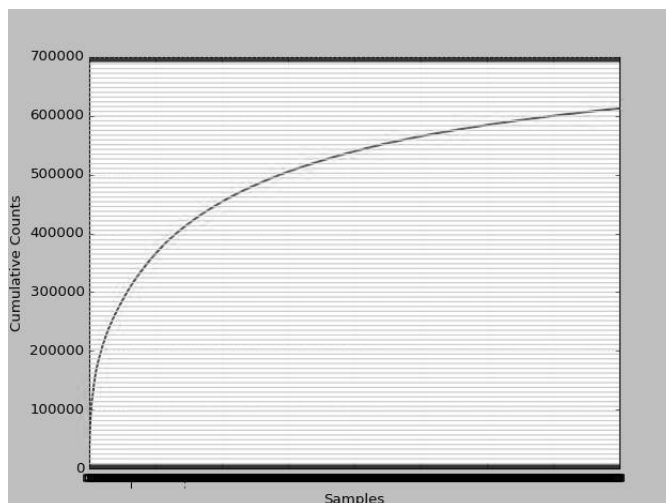


图 8-14 1000 个高频词的累积分布

粗略估计下大约占了 80% 以上。频率最高的前 100 个词的分布如图 8-15 所示。

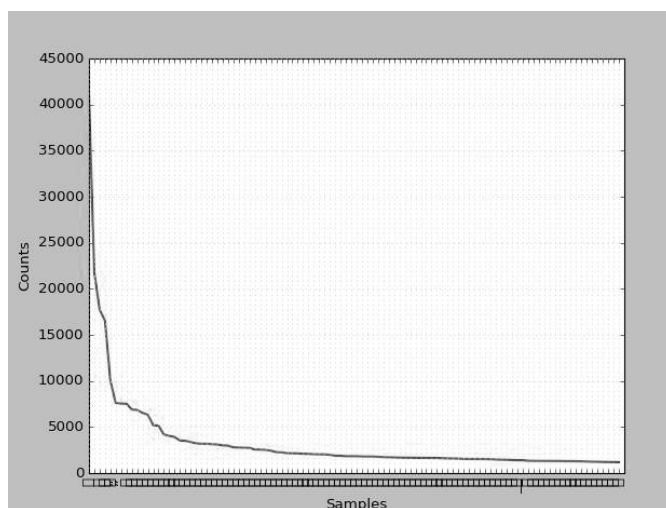


图 8-15 频率最高的前 100 个词的分布

前 100 个词也就是大约 0.02% 的词在本书的累积分布情况如图 8-16 所示。

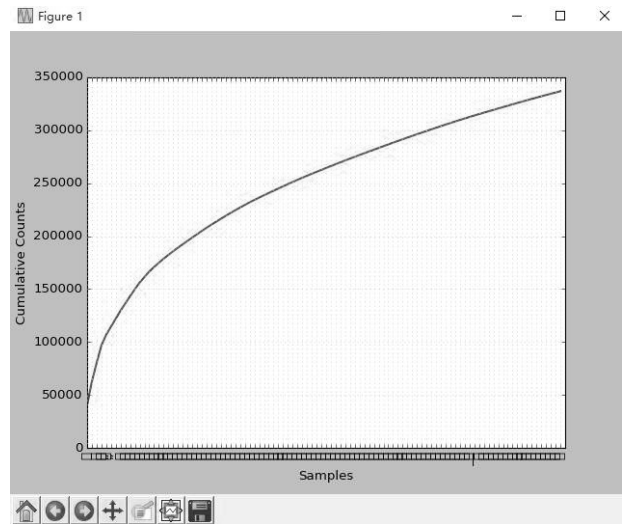


图 8-16 前 100 个词的累积分布

```
>>> fdist=FreqDist(str)
>>> fdist.plot(100,cumulative=True)
```

图 8-16 可知，前 0.2% 词汇占据整本书的 50% 以上的比例。“国、旗、秦、魏、队、阅”等跟战争相关词汇使用较多。那么低频词如何呢？有时候低频词也具有其特殊的研究价值，如图 8-17 所示。

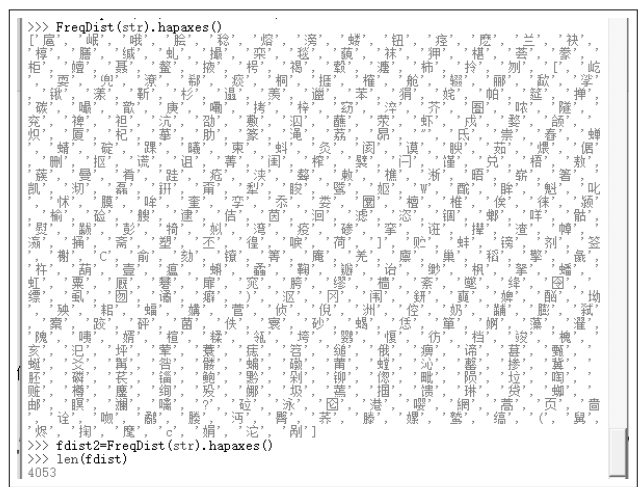


图 8-17 低频词的情况

统计可知大约有 4053 个词出现一次，占比为 0.6%。词语内部搭配如图 8-18 所示。

```
>>> from collections import Counter
>>> V=Counter(str)
```

File	Edit	Shell	Debug	Options	Window	Help
25	25	25	25	25	25	25
24	24	24	24	24	24	24
23	23	23	23	23	23	23
22	22	22	22	22	22	22
21	21	21	21	21	21	21
20	20	20	20	20	20	20
19	19	19	19	19	19	19
18	18	18	18	18	18	18
17	17	17	17	17	17	17
16	16	16	16	16	16	16
15	15	15	15	15	15	15
14	14	14	14	14	14	14
13	13	13	13	13	13	13
12	12	12	12	12	12	12

图 8-18 词语内部搭配

查询词频在 [0 ~ 100] 的词有多少？

```
>>> len([w for w in V.values() if w<100])
3103
```

查询词频在 [100 ~ 1000] 的词有多少？

```
>>> len([w for w in V.values() if w>100 and w<1000])
819
```

查询词频在 [1000 ~ 5000] 的词有多少？

```
>>> len([w for w in V.values() if w>1000 and w<5000])
113
```

查询词频在 [5000 以上] 的词有多少？

```
>>> len([w for w in V.values() if w>5000])
14
```

第 9 章

中文自动分词

本章导读：中文分词技术属于自然语言处理技术范畴，中文分词是其他中文信息处理的基础，搜索引擎只是中文分词的一个应用。诸如机器翻译（MT）、语音合成、自动分类、自动摘要、自动校对，等等。本章首先介绍中文分词及其特点和难点，其次对常用的中文分词方法进行阐述；紧接着介绍几个典型的中文分词工具，有兴趣的读者还可对文中所列出的其他工具自行深入研究。最后，本章将对结巴中文分词进行详细介绍，从原理到使用逐渐深入，力求让读者快速掌握其思想及原理。

9.1 中文分词简介

中文分词

中文分词指的是将一个汉字序列切分成一个个单独的词。分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。我们知道，在英文的行文中，单词之间是以空格作为自然分界符的，而中文只是句段能通过明显的分界符来简单划界，而词是没有一个形式上的分界符的。虽然英文也同样存在短语的划分问题，不过在词这一层上，中文比之英文要复杂得多、困难得多。

例如：

英文句子：I am a student.

中文意思：我是一名学生。

由于英文的语言使用习惯，通过空格我们很容易拆分出单词；而中文字词接线模糊，往往不容易区别哪些是“字”、哪些是“词”。这也是为什么我们想把中文的词语进行切分的原因。

中文分词的发展

与以英文为代表的印欧语系语言相比，中文由于继承自古代汉语的传统，词语之间常常没有分隔。古代汉语中除了联绵词和人名地名等，词通常就是单个汉字，所以当时没有分词书写的必要。而现代汉语中双字或多字词逐渐增多，一个字已经不再等同于一个词了。

在中文里，“词”和“词组”的边界模糊。现代汉语的基本表达单元虽然为“词”，且以双字或者多字词居多，但由于人们认知水平的不同，对词和短语的边界还很难区分。

◎ 例如：“对随地吐痰者给予处罚”，“随地吐痰者”本身是一个词还是一个短语，不同的人会有不同的标准，同样的“海上”“酒厂”等，即使是同一个人也可能做出不同的判断，如果汉语真的要分词书写，则必然会出现混乱，难度很大。

中文分词的方法其实不局限于中文应用，也被应用到英文处理。例如，手写识别，英文单词之间的空格就不是很清楚，中文分词方法可以反过来帮助判别英文单词的边界。

中文分词的用途

中文分词是文本处理的基础，对于输入的一段中文，成功地进行中文分词，可以达到计算机自动识别语句含义的效果。中文分词技术属于自然语言处理技术范畴，目前在自然语言处理技术中，中文处理技术比西文处理技术要落后很大一截，而许多西文的处理方法中文却不能直接采用，就是因为中文必须有分词这道工序。中文分词是其他中文信息处理的基础，搜索引擎只是中文分词的一个应用。其他的比如机器翻译（MT）、语音合成、自动分类、自动摘要、自动校对等，都需要用到分词。因为中文需要分词，可能会影响一些研究，但同时也为企业带来机会，因为国外的计算机处理技术要想进入中国市场，首先要解决中文分词问题。在中文研究方面，相比外国人来说，中国人有十分明显的优势。

中文分词对于搜索引擎来说，最重要的并不是找到所有结果，因为在上百亿的网页中找到所有结果没有太多的意义，没有人能看得完；相反，最重要的是把最相关的结果排在最前面，这也称为相关度排序。中文分词的准确与否，常常直接影响对搜索结果的相关度排序。从定性分析来说，搜索引擎的分词算法不同、词库的不同都会影响页面的返回结果。

9.2 中文分词的特点和难点

中文分词就是让计算机在词与词之间加上边界标记。当前研究所面临的问题和困难主要体现在三个方面：分词的规范、歧义词的切分和未登录词识别。

（1）分词的规范。

中文因其自身语言特性的局限，字（词）的界限往往很模糊，关于字（词）的抽象定义和词边界的划定尚没有一个公认的、权威的标准。曾经有专家对母语是汉语者进行了调查，结果显示，对汉语文本中“词”的认同率仅有 70% 左右。正是由于这种不同的主观分词差异，给汉语分词造成了极大的困难。尽管在 1992 年国家颁布了《信息处理用现代词汉语分词规范》，但是这种规范很容易受主观因素影响，在处理现实问题时也不免相形见绌。

（2）歧义词切分。

中文中的歧义词是很普遍的，歧义词即同一个词有多种切分方式，该如何处理这种问题呢？普遍认为中文歧义词有三种类型。

- ◎ 交集型切分歧义，汉语词如 AJB 类型，满足 AJ 和 JB 分别成词。如“大学生”一种切分方式“大学/生”，另一种切分方式“大/学生”。你很难去判定哪种切分正确，即使是人工切分也只能依据上下文，类似的有“结合成”“美国会”等。
- ◎ 组合型切分歧义，汉语词如 AB，满足 A、B、AB 分别成词。如“郭靖有武功高超的才能”中的“才能”，一种切分为“郭靖/有/武功/高超/的/才能”，另一种切分“中国/什么时候/才/能/达到/发达/国家/水平”显示是不同的切分方式。
- ◎ 混合型切分歧义，汉语词包含如上两种共存情况。如“郭靖说这把剑太重了”，其中“太重了”是交集型字段，“太重”是组合型字段。

（3）未登录词（新词）识别。

未登录词又称新词。这类词通常指两个方面，一是词库中没有收录的词，二是训练语料没有出现过的词。未登录词主要体现在以下几种。

- ◎ 新出现的网络用词。如“蓝牙”“蓝瘦香菇”“房姐”“奥特”“累觉不爱”等。
- ◎ 研究领域名称：特定领域和新出现领域的专有名词。如“苏丹红”“禽流感”“埃博拉”“三聚氰胺”等。
- ◎ 其他专有名词：诸如城市名、公司企业、职称名、电影、书籍、专业术语、缩写词等。如“成都”“阿里巴巴”“三少爷的剑”“NLP”“川大”等。

综上所述，处理汉语词边界、歧义词切分和未登录词切分问题比较复杂，其中未登录词的影响大大超过了歧义词的影响，所以如何处理未登录词是关键问题。

9.3 常见中文分词方法

早在 20 世纪 80 年代就有中文分词的研究工作，曾有人提出“正向最大匹配法”“逆向最大匹配法”“双向扫描匹配法”“逐词遍历法”等方法，共计多达 16 种之多。由于这些分词方法多是基于规则和词表的方法，随着统计方法的发展，不少学者提出很多关于统计模型的中文分词方法。关于规则的中文自动方法主要有以下几种。

(1) 基于字符串匹配的分词方法。

基本思想是基于词典匹配，将待分词的中文文本根据一定规则进行切分和调整，然后跟词典中的词语进行匹配，匹配成功则按照词典的词分词，匹配失败则通过调整或者重新选择，如此反复循环即可。代表方法有基于正向最大匹配和基于逆向最大匹配及双向匹配法。

(2) 基于理解的分词方法。

基本思想是通过专家系统或者机器学习神经网络方法模拟人的理解能力。前者是通过专家对分词规则的逻辑推理并总结形成特征规则，不断迭代完善规则，其受到资源消耗大和算法复杂度高的制约。后者通过机器模拟人类理解的方式，虽然可以取得不错的效果，但是依旧受训练时间长和过拟合等因素困扰。

(3) 基于统计的分词方法。

关于统计的中文分词方法的基本思想本书整理如下：

- ◎ 基于隐马尔可夫模型的中文分词方法。基本思想是通过文本作为观测序列去确定隐藏序列的过程。该方法采用 Viterbi 算法对新词识别，效果不错，但具有生成式模型的缺点，需要计算联合概率，因此随着文本增大，存在计算量大的问题。
- ◎ 基于最大熵模型的中文分词方法。基本思想是学习概率模型时，在可能的概率分布模型中，认为熵最大的进行切分。该方法可以避免生成模型的不足，但是存在偏移量的问题。
- ◎ 基于条件随机场模型的中文分词方法。基本思想主要来源于最大熵马尔可夫模型，主要关注的字跟上下文标记位置有关，进而通过解码找到词边界。因此需要大量训练语料，而训练和解码又非常耗时。

综上所述，基于词典和规则的方法其分词速度较快，但是在不同领域取得的效果差异很大，还存在构造费时费力、算法复杂度高、移植性差等缺点。基于统计的中文分词，虽然其相较于规则的方法取得了不错的效果，但也存在模型训练时间长、分词速度慢等问题。针对这些问题，本书提出基于隐马尔可夫统计模型和自定义词典结合的方法，其在分词速度、歧义分析、新词发现和准确率方面都取得了不错效果。

9.4 典型中文分词工具

9.4.1 HanLP 中文分词

HanLP

HanLP 是由一系列模型与算法组成的 Java 工具包，目标是普及自然语言处理在生产环境中的应用。HanLP 具备功能完善、性能高效、架构清晰、语料时新、可自定义等特点。在提供丰富功能的同时，HanLP 内部模块坚持低耦合、模型坚持惰性加载、服务坚持静态提供、词典坚持明文发布，使用起来非常方便，同时自带一些语料处理工具，帮助用户训练自己的语料。

Python 调用 HanLP 进行中文分词

- (1) 下载 HanLP 的 jar 包 hanlp.jar (<http://hanlp.linrunsoft.com/services.html>)。
- (2) 安装配置 JRE1.7+，本书省略具体安装步骤。
- (3) 在“.py”后缀的 Python 文件中启动 JVM。

```
from jpy import *
```

```
startJVM(getDefaultJVMPath(), "-Djava.class.path=C:\\hanlp\\hanlp-1.3.2.jar;
```

```
C:\\hanlp", "-Xms1g", "-Xmx1g") # 启动JVM, Linux需替换分号;为冒号:
```

此处是HanLP的具体调用方法

```
shutdownJVM()
```

Python 调用 HanLP 分词

◎ 默认分词

```
paraStr1='中国科学院计算技术研究所的宗成庆教授正在教授自然语言处理课程'
print("="*30+"HanLP分词"+"="*30)
HanLP = JClass('com.hankcs.hanlp.HanLP')
print(HanLP.segment(paraStr1))
```

运行结果:

```
=====HanLP分词=====
[中国科学院计算技术研究所/nt, 的/ude1, 宗成庆/nr, 教授/nnt, 正在/d, 教授/nnt, 自然语言处理/nz, 课程/n]
```

◎ 标准分词

```
print("="*30+"标准分词"+"="*30)
StandardTokenizer = JClass('com.hankcs.hanlp.tokenizer.StandardTokenizer')
print(StandardTokenizer.segment(paraStr1))
```

运行结果:

```
=====标准分词=====
[中国科学院计算技术研究所/nt, 的/ude1, 宗成庆/nr, 教授/nnt, 正在/d, 教授/nnt, 自然语言处理/nz, 课程/n]
```

◎ NLP 分词

```
print("="*30+"NLP分词"+"="*30)
NLPTokenizer = JClass('com.hankcs.hanlp.tokenizer.NLPTokenizer')
print(NLPTokenizer.segment(paraStr1))
```

运行结果:

```
=====NLP分词=====
[中国科学院计算技术研究所/nt, 的/ude1, 宗成庆/nr, 教授/nnt, 正在/d, 教授/v, 自然语言处理/nz, 课程/n]
```

◎ 索引分词

```
print("="*30+" 索引分词"+"="*30)
IndexTokenizer = JClass('com.hankcs.hanlp.tokenizer.IndexTokenizer')
termList= IndexTokenizer.segment(paraStr1);
for term in termList :
    print(str(term) + " [" + str(term.offset) + ":" + str(term.offset + len(
        term.word)) + "]"")
```

运行结果:

```
=====索引分词=====
中国科学院计算技术研究所/nt [0:12]
中国/ns [0:2]
中国科学院/nt [0:5]
科学/n [2:4]
科学院/nis [2:5]
学院/nis [3:5]
计算/v [5:7]
技术/n [7:9]
研究/vn [9:11]
研究所/nis [9:12]
的/ude1 [12:13]
宗成庆/nr [13:16]
自然语言/gm [13:17]
自然语言处理/nz [13:19]
教授/nnt [16:18]
正在/d [18:20]
教授/nnt [20:22]
自然语言处理/nz [22:28]
自然/n [22:24]
语言/n [24:26]
处理/vn [26:28]
课程/n [28:30]
```

◎ 极速词典分词

```
print("="*30+" 极速词典分词"+"="*30)
```

```
SpeedTokenizer = JClass('com.hankcs.hanlp.tokenizer.SpeedTokenizer')
print(NLPTokenizer.segment(paraStr1))
```

运行结果:

```
===== 极速词典分词=====
[中国科学院计算技术研究所/nt, 的/ude1, 宗成庆/nr, 教授/nnt, 正在/d, 教授/v, 自然语言处理/nz, 课程/n]
```

◎ 自定义分词

```
paraStr2 = '攻城狮逆袭单身狗，迎娶白富美，走上人生巅峰'
print("="*30+" 自定义分词"+"="*30)
CustomDictionary = JClass('com.hankcs.hanlp.dictionary.CustomDictionary')
CustomDictionary.add('攻城狮')
CustomDictionary.add('单身狗')
HanLP = JClass('com.hankcs.hanlp.HanLP')
print(HanLP.segment(paraStr2))
```

运行结果:

```
===== 自定义分词=====
[攻城狮/nz, 逆袭/nz, 单身狗/nz, , /w, 迎娶/v, 白富美/nr, , /w, 走上/v, 人生/n, 巅峰/n]
```

9.4.2 其他中文分词工具

- ◎ BosonNLP: 玻森实验室开发的一款分词工具。
- ◎ 语言云: 以哈工大社会计算与信息检索研究中心研发的“语言技术平台 (LTP)”为基础, 为用户提供高效精准的中文自然语言处理云服务。
- ◎ NLPIR: 中科院分词系统。
- ◎ 新浪云。
- ◎ 搜狗分词。

- ◎ 结巴分词。
- ◎ SCWS: 简易中文分词系统缩写。SCWS 由 hightman 开发, 并以 BSD 许可协议开源发布, 源码托管在 GitHub。
- ◎ 腾讯文智。
- ◎ 盘古分词。
- ◎ IKAnalyzer: 一个开源的、基于 Java 语言开发的轻量级的中文分词工具包。

中文分词工具可分为基于规则的分词方法和基于统计的分词方法两种。本节所给出的相关常见分词工具, 读者感兴趣可以自行深入研究。特别说明的是, 随着深度学习的快速发展, 深度学习在中文分词的应用也越来越流行。由于开发过程中多数开发人员及其相关研究者使用结巴中文分词的比较多, 所以本书将其单独立为一节进行深入学习。

9.5 结巴中文分词

9.5.1 基于 Python 的结巴中文分词

结巴中文分词的特点

- (1) 支持三种分词模式。
 - ◎ 精确模式, 试图将句子最精确地切开, 适合文本分析;
 - ◎ 全模式, 把句子中所有的可以成词的词语都扫描出来, 速度非常快, 但是不能解决歧义;
 - ◎ 搜索引擎模式, 在精确模式的基础上, 对长词再次切分, 提高召回率, 适合用于搜索引擎分词。
- (2) 支持繁体分词。
- (3) 支持自定义词典。
- (4) MIT 授权协议。
 - ◎ 在线演示: <http://jiebademo.ap01.aws.af.cm/>。
 - ◎ 网站代码: <https://github.com/fxsjy/jiebademo>。

安装说明：代码对 Python 2/3 均兼容

- ◎ 全自动安装：easy_install jieba 或者 pip install jieba / pip3 install jieba。
- ◎ 半自动安装：先下载文件（<http://pypi.python.org/pypi/jieba/>），解压后运行 Python setup.py install。
- ◎ 手动安装：将 jieba 目录放置于当前目录或者 site-packages 目录下。
- ◎ 通过 import jieba 来引用。

结巴分词工具下载

- ◎ hanlp jar 包（<http://download.csdn.net/download/lb521200200/9686915>）；
- ◎ ik 分词 5.0.0 版本 jar 包（<http://download.csdn.net/download/youyao816/9676084>）；
- ◎ ik 分词 1.10.1 版本 jar 包（<http://download.csdn.net/download/youyao816/9676082>）；
- ◎ IKAnalyzer 所需的 jar 包（<http://download.csdn.net/download/jingjingchen1014/9659225>）；
- ◎ jieba 分词包（<http://download.csdn.net/download/u014018025/9652341>）。

主要分词功能

- ◎ jieba.cut 方法接受三个输入参数：需要分词的字符串；cut_all 参数用来控制是否采用全模式；HMM 参数用来控制是否使用 HMM 模型。
- ◎ jieba.cut_for_search 方法接受两个参数：需要分词的字符串；是否使用 HMM 模型。该方法适合用于搜索引擎构建倒排索引的分词，粒度比较细。
- ◎ 待分词的字符串可以是 Unicode 或 UTF-8 字符串、GBK 字符串。注意：不建议直接输入 GBK 字符串，可能无法预料地错了解码成 UTF-8。
- ◎ jieba.cut 和 jieba.cut_for_search 返回的结构都是一个可迭代的 generator，可以使用 for 循环来获得分词后得到的每一个词语（Unicode）。
- ◎ jieba.lcut 和 jieba.lcut_for_search 直接返回 list。
- ◎ jieba.Tokenizer(dictionary=DEFAULT_DICT) 新建自定义分词器，可用于同时使用不同词典。jieba.dt 为默认分词器，所有全局分词相关函数都是该分词器的映射。

代码示例：

```
# encoding=utf-8
import jieba
seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + " ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + " ".join(seg_list))  # 精确模式

seg_list = jieba.cut("他来到了网易杭研大厦")  # 默认是精确模式
print(" ", ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print(" ", ".join(seg_list))
```

输出结果：

【全模式】：我 / 来到 / 北京 / 清华 / 清华大学 / 华大 / 大学
【精确模式】：我 / 来到 / 北京 / 清华大学
【新词识别】：他，来到，了，网易，杭研，大厦 （此处，“杭研”并没有在词典中，但是也被Viterbi算法识别出来了）
【搜索引擎模式】： 小明，硕士，毕业，于，中国，科学，学院，科学院，中国科学院，计算，计算所，后，在，日本，京都，大学，日本京都大学，深造

添加自定义词典

载入词典

- ◎ 开发者可以指定自己自定义的词典，以便包含 jieba 词库里没有的词。虽然 jieba 有新词识别能力，但是自行添加新词可以保证更高的正确率。
- ◎ 用法：jieba.load_userdict(file_name) # file_name 为文件类对象或自定义词典的路径。
- ◎ 词典格式和 dict.txt 一样，一个词占一行；每一行分词语、词频（可省略）和词性（可省略）三部分，用空格隔开，顺序不可颠倒。file_name 若为路径或二进制方式打开的文件，则文件必须为 UTF-8 编码。

- ◎ 词频省略时使用自动计算的能保证分出该词的词频。

例如：

```
创新办 3 i
云计算 5
凯特琳 nz
台中
```

自定义分词前后对比：

```
之前： 李小福 / 是 / 创新 / 办 / 主任 / 也 / 是 / 云 / 计算 / 方面 / 的
/ 专家 /
加载自定义词库后： 李小福 / 是 / 创新办 / 主任 / 也 / 是 / 云计算 / 方
面 / 的 / 专家 /
```

9.5.2 结巴分词工具详解

结巴分词的算法策略

- (1) 基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG）。
- (2) 采用了动态规划查找最大概率路径，找出基于词频的最大切分组合。
- (3) 对于未登录词，采用基于汉字成词能力的 HMM 模型，使用 Viterbi 算法。

结巴源码组织形式

```
jieba
|-- Changelog
|-- extra_dict
| |-- dict.txt.big
| |-- dict.txt.small
| |-- idf.txt.big
| `-- stop_words.txt
```

```
|-- jieba
| |-- analyse
| | |-- analyzer.py
| | |-- idf.txt
| | |-- __init__.py
| | |-- textrank.py
| | `-- tfidf.py
| |-- _compat.py
| |-- dict.txt
| |-- finalseg
| | |-- __init__.py
| | |-- prob_emit.p
| | |-- prob_emit.py
| | |-- prob_start.p
| | |-- prob_start.py
| | |-- prob_trans.p
| | `-- prob_trans.py
| |-- __init__.py
| |-- __main__.py
| `-- posseg
| |-- char_state_tab.p
| |-- char_state_tab.py
| |-- __init__.py
| |-- prob_emit.p
| |-- prob_emit.py
| |-- prob_start.p
| |-- prob_start.py
| |-- prob_trans.p
| |-- prob_trans.py
| `-- viterbi.py
|-- LICENSE
|-- setup.py `-- test
|-- *.py
|-- parallel
| |-- extract_tags.py
| `-- test*.py `-- userdict.txt
```

算法实现分词

(1) 基于前缀词典实现高效的词图扫描, 生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)。

生成句子中汉字所有可能成词情况所构成的有向无环图。DAG 根据我们生成的前缀字典来构造一个这样的 DAG, 对一个 sentence DAG 是以 $\{\text{key}:\text{list}[i,j\dots], \dots\}$ 的字典结构存储, 其中 key 是词在 sentence 中的位置, list 存放的是在 sentence 中以 key 开始且词 sentence[key:i+1] 在前缀词典中以 key 开始 i 结尾的词的末位置 i 的列表, 即 list 存放的是 sentence 中以位置 key 开始的可能词语的结束位置, 这样通过查字典得到词, 以及开始位置 + 结束位置列表。

例如: 句子“抗日战争”生成的 DAG 中 $\{0:[0,1,3]\}$ 这样一个简单的 DAG, 就是表示 0 位置开始, 在 0,1,3 位置都是词。
就是说 0~0, 0~1, 0~3 即“抗”, “抗日”, “抗日战争”这三个词在 dict.txt 中是词。

(2) 采用动态规划查找最大概率路径, 找出基于词频的最大切分组合。基于上面的 DAG 利用动态规划查找最大概率路径, 理解 DP 算法 (动态规划算法) 很容易就能明白了。根据动态规划查找最大概率路径的基本思路就是对句子从右往左反向计算最大概率。依次类推, 最后得到最大概率路径, 得到最大概率的切分组合 (这里满足最优子结构性质, 可以利用反证法进行证明)。代码实现中有个小诀窍, 即概率对数 (可以让概率相乘的计算变成对数相加, 防止相乘造成下溢, 因为在语料、词库中, 每个词的出现概率平均下来还是很小的浮点数)。

(3) 对于未登录词, 采用基于汉字成词能力的 HMM 模型, 使用 Viterbi 算法; 未登录词其实就是词典 dict.txt 中没有记录的词。这里采用了 HMM 模型, HMM 是一个简单强大的模型 HMM 在实际应用中主要用来解决 3 类问题。

- ◎ 评估问题 (概率计算问题): 即给定观测序列 $O=O_1, O_2, O_3 \dots O_t$ 和模型参数 $\lambda=(A, B, \pi)$, 怎样有效计算这一观测序列出现的概率 (Forward-backward 算法)。
- ◎ 解码问题 (预测问题): 即给定观测序列 $O=O_1, O_2, O_3 \dots O_t$ 和模型参数 $\lambda=(A, B, \pi)$, 怎样寻找满足这种观察序列意义上最优的隐含状态序列 S (Viterbi 算法, 近似算法)。
- ◎ 学习问题: 即 HMM 的模型参数 $\lambda=(A, B, \pi)$ 未知, 如何求出这 3 个参数以使观测序列 $O=O_1, O_2, O_3 \dots O_t$ 的概率尽可能大 (即用极大似然估计的方法估计参数, Baum-Welch, EM 算法)。

模型的关键相应参数 $\lambda=(\mathbf{A},\mathbf{B},\pi)$ ，经过作者对大量语料的训练，得到了 `finalseg` 目录下的三个文件（初始化状态概率（ π ）即词语以某种状态开头的概率，其实只有两种，要么是 B，要么是 S。这个就是起始向量，就是 HMM 系统的最初模型状态，对应文件 `prob_start.py`。隐含状态概率转移矩 $\mathbf{A}$ 即字的几种位置状态（用 BEMS 四个状态来标记，B 是开始 `begin` 位置；E 是 `end`，是结束位置；M 是 `middle`，是中间位置；S 是 `single`，单独成词的位置）的转换概率，对应文件 `prob_trans.py`；观测状态发射概率矩阵 $\mathbf{B}$ 即位置状态到单字的发射概率，比如 $P(\text{“狗”}|\text{M})$ 表示一个词的中间出现“狗”这个字的概率，对应文件 `prob_emit.py`）。

9.5.3 结巴分词核心内容

结巴分词的算法策略

在这个版本中使用前缀字典实现了词库的存储（即 `dict.txt` 文件中的内容），而弃用之前版本的 `trie` 树存储词库。Python 中实现的 `trie` 树是基于 `dict` 类型的数据结构，而且 `dict` 中又嵌套 `dict` 类型，这样嵌套很深，导致内存耗费严重，具体实现见 `gen_pfdict(self, f_name)`。

下面是 `@gumblxcommit` 的内容：

对于 `get_DAG()` 函数来说，用 `Trie` 数据结构，特别是在 Python 环境中，内存使用量过大。经实验，可构造一个前缀集合解决问题。

该集合储存词语及其前缀，如 `set(['数', '数据', '数据结', '数据结构'])`。

在句子中按字正向查找词语，在前缀列表中就继续查找，直到不在前缀列表中或超出句子范围。大约比原词库增加 40% 词条。

该版本通过各项测试，与原版本分词结果相同。测试：一本 5.7MB 的小说，用默认字典，64 位 Ubuntu, Python 2.7.6。

Trie: 第一次加载为 2.8 秒，缓存加载为 1.1 秒；内存为 277.4MB，平均速率为 724KB/s

前缀字典: 第一次加载为 2.1 秒，缓存加载为 0.4 秒；内存为 99.0MB，平均速率为 781KB/s

此方法解决纯 Python 中 `Trie` 空间效率低下的问题。

jieba 0.37 版本中实际使用的是前缀字典（对应代码中 `Tokenizer.FREQ` 字典），即利用 Python 中的 `dict` 把 `dict.txt` 中出现的词作为 `key`，出现频次作为 `value`，比如 `sentece`：“北京

大学”，处理后的结果为 {u '北':17860,u '北京':34488,u '北京大':0,u '北京大学':2053}，具体详情见代码 `def gen_pfdict(self, f_name)`。

DAG

例如，句子“去北京大学玩”对应的 DAG 为 {0:[0], 1:[1, 2, 4], 2:[2], 3:[3, 4], 4:[4], 5:[5]}。

例如，DAG 中 {0:[0]} 这样一个简单的 DAG，就是表示 0 位置对应的是词，就是说 0 ~ 0，即“去”这个词在 dict.txt 中是词条。DAG 中 {1:[1,2,4]}，就是表示从 1 位置开始，在 1、2、4 位置都是词，就是说 1 ~ 1、1 ~ 2、1 ~ 4 即“北”“北京”“北京大学”这三个词在 dict.txt 对应文件的词库中。

基于词频最大切分组合

通过上面内容可以得知，我们已经有了词库 (dict.txt) 的前缀字典和待分词句子 sentence 的 DAG，基于词频的最大切分要在所有的路径中找出一条概率得分最大的路径，该怎么做呢？

jieba 中的思路就是使用动态规划方法，从后往前遍历，选择一个频度得分最大的一个切分组合。具体实现如下代码（已给详细注释）。

```
#动态规划，计算最大概率的切分组合
def calc(self, sentence, DAG, route):
    N = len(sentence)
    route[N] = (0, 0)
    # 对概率值取对数之后的结果
    logtotal = log(self.total)
    # 从后往前遍历句子，反向计算最大概率
    for idx in xrange(N - 1, -1, -1):
        # [x+1][0]即表示取句子x+1位置对应元组(概率对数，词语末字位置)的概率对数
        route[idx] = max((log(self.FREQ.get(sentence[idx:x + 1]) or 1) -
                           logtotal + route[x + 1][0], x) for x in DAG[idx])
```

从代码中可以看出 `calc` 是一个自底向上的动态规划（重叠子问题、最优子结构），它从 `sentence` 的最后一个字（ $N-1$ ）开始倒序遍历 `sentence` 的字（`idx`），计算子句 `sentence[isdx~N-1]` 概率对数得分（这里利用 DAG 及历史计算结果 `route` 实现，同时使用概率对数以有效防止下溢问题）。然后将概率对数得分最高的情况以（概率对数，词语最后一个字的位置）这样的 tuple 保存在 `route` 中。根据上面的结果写了如下的测试。输出结果为：

```
“去北京大学玩”的前缀字典：
去 123402
去北 0
去北京 0
去北京大 0
去北京大学 0
去北京大学玩 0
“去北京大学玩”的DAG：
0 : [0]
1 : [1, 2, 4]
2 : [2]
3 : [3, 4]
4 : [4]
5 : [5]
route：
{0: (-26.039894284878688, 0), 1: (-19.851543754900984, 4), 2:
(-26.6931716802707, 2),
 3: (-17.573864399983357, 4), 4: (-17.709674112779485, 4), 5:
(-9.567048044164698, 5), 6: (0, 0)}
去/北京大学/玩
```

中文分词的未登录词

- ◎ 分词规范，词的定义还不明确（《统计自然语言处理》：宗成庆）。
- ◎ 歧义切分问题、交集型切分问题、多义组合型切分歧义等。结婚的和尚未结婚的=>结婚/的/和/尚未/结婚/的结婚/的/和尚/未/结婚/的。
- ◎ 未登录词问题有两种解释：一是已有的词表中没有收录的词，二是已有的训练语料中未曾出现过的词，第二种含义中未登录词又称 OOV（Out of Vocabulary）。对于大规模

真实文本来说，未登录词对于分词的精度的影响远超歧义切分。一些网络新词、自造词一般都属于这些词。

因此可以看到，未登录词是分词中的一个重要问题，jieba 分词中对于 OOV 的解决方法是：采用基于汉字成词能力的 HMM 模型，使用 Viterbi 算法。

9.5.4 结巴分词基本用法

本机是 Win10 64 位系统，已经安装了 pip 工具，按 Win+R 键，输入 pip install jieba，效果如图 9-1 所示。

```
E:\tools\jieba-master>pip install jieba
Collecting jieba
  Downloading jieba-0.38.zip (7.4MB)
    100% | 7.4MB 97kB/s
Installing collected packages: jieba
  Running setup.py install for jieba... done
Successfully installed jieba-0.38
You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
E:\tools\jieba-master>
```

图 9-1 结巴分词安装成功

结巴几种模式下的分词操作（以下默认已导入 import jieba）：

全模式分词

```
>>> import jieba
>>> str="我是王小二来自技术社区"
>>> seg_list=jieba.cut(str,cut_all=True)
>>> print("Full Mode: " + "/ ".join(seg_list)) # 全模式
Full Mode: 我/ 是/ 王/ 小/ 二/ 来自/ 技术/ 社区
```

结果分析：

显然我的名字：王小二，没有正确分词，这是因为全模式把句子中所有可以成词的词语都扫描出来，速度非常快，但是不能解决歧义。

精确模式分词

```
>>> seg_list=jieba.cut(str,cut_all=False)
>>> print("Default Mode: " + "/" .join(seg_list)) # 精确模式
Default Mode: 我/ 是/ 王小二/ 来自/ 技术社区
>>> seg_list=jieba.cut(str)
>>> print("Default Mode: " + "/" .join(seg_list)) # 默认模式
Default Mode: 我/ 是/ 王小二/ 来自/ 技术社区
```

结果分析:

首先默认模式就是精确模式，即`cut_all=False`。这里很好地将“白宁超”划分为一个词。与全模式分词是有区别的。精确模式适合文本分析。

默认精确模式分词

```
>>> seg_list = jieba.cut("他来到了网易杭研大厦") # 默认是精确模式
>>> print("【新词发现】\t"+", ".join(seg_list))
【新词发现】 他, 来到, 了, 网易, 杭研, 大厦
```

结果分析:

此处杭研并没有在词典中，但是也被Viterbi算法识别出来了。实际上是基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法可以发现新词。当然也可以到自定义字典中去收集新词。

搜索引擎模式分词

```
>>> seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造") # 搜索引擎模式
>>> print("搜索引擎模式: \t"+", ".join(seg_list))
搜索引擎模式: 小明, 硕士, 毕业, 于, 中国, 科学, 学院, 科学院, 中国科学院, 计算, 计算所, , , 后, 在, 日本, 京都, 大学, 日本京都大学, 深造
```

结果分析：

在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

繁体分词

```
>>> str=''此開卷第一回也。作者自云：因曾歷過一番夢幻之后，故將真事隱去，而借"通靈"之說，撰此《石頭記》一書也。故曰"甄士隱"云云。但書中所記何事何人？自又云：“今風塵碌碌，一事無成，忽念及當日所有之女子，一一細考較去，覺其行止見識，皆出于我之上。何我堂堂須眉，誠不若彼裙釵哉？實愧則有余，悔又無益之大無可如何之日也！'''
>>> str=jieba.cut(str)
>>> print('/ '.join(str))
```

```
此開卷/ 第一回/ 也/ . / 作者/ 自云/ : / 因曾/ 歷過/ 一番/ 夢/ 幻之后/ ,
/ 故將/ 真事/ 隱去/ , /
/ 而/ 借/ "/ 通靈/ "/ 之/ 說/ , / 撰此/ 《/ 石頭記/ 》/ 一書/ 也/ . / 故
/ 曰/ "/ 甄士/ 隱/ "/ 云云/ . / 但書中/ 所記/
/ 何事何/ 人/ ? / 自又云/ : / “/ 今風/ 塵碌碌/ , / 一事/ 無成/ , / 忽念
及/ 當日/ 所/ 有/ 之/ 女子/ , / 一/
/ 一細/ 考較/ 去/ , / 覺其/ 行止/ 見識/ , / 皆/ 出于/ 我/ 之/ 上/ . / 何
/ 我堂/ 堂須/ 眉/ , / 誠不若/ 彼/ 裙釵/
/ 哉/ ? / 實愧則/ 有/ 余/ , / 悔/ 又/ 無益/ 之/ 大/ 無/ 可/ 如何/ 之/ 日
/ 也/ !
>>>
```

自定义分词器

```
#encoding=utf-8
from __future__ import print_function, unicode_literals
import sys
sys.path.append("../")
import jieba
jieba.load_userdict("userdict.txt")
```

```

import jieba.posseg as pseg

jieba.add_word('凯特琳')
jieba.del_word('自定义词')

test_sent = (
    "李小福和李铁军是创新办主任也是云计算方面的专家；什么是八一双鹿\n"
    "例如我输入一个带“韩玉赏鉴”的标题，在自定义词库中也增加了此词为N类\n"
    "「台中」正确应该不会被切开。mac上可分出「石墨烯」；此时又可以分出来凯特琳了。"
)

words = jieba.cut(test_sent)
print('/'.join(words))

print("="*40)

result = pseg.cut(test_sent)

for w in result:
    print(w.word, "/", w.flag, ", ", end=' ')

print("\n" + "="*40)

terms = jieba.cut('easy_install is great')
print('/'.join(terms))
terms = jieba.cut('Python 的正则表达式是好用的')
print('/'.join(terms))

print("="*40)
# test frequency tune
testlist = [
    ('今天天气不错', ('今天', '天气')),
    ('如果放到post中将出错。', ('中', '将')),
    ('我们中出了一个叛徒', ('中', '出')),
]

for sent, seg in testlist:

```

```

print('/'.join(jieba.cut(sent, HMM=False)))
word = ''.join(seg)
print('%s Before: %s, After: %s' % (word, jieba.get_FREQ(word),
    jieba.suggest_freq(seg, True)))
print('/'.join(jieba.cut(sent, HMM=False)))
print("-"*40

```

结果分析:

首先对一段话进行分词处理,此处“李小福”和“李铁军”都是人名,结果却分词“李小福”和“李铁”,而“军是”当作一个词处理,显然不对。我们可以将“李铁军”当作一个词加入自定义文本中,处理后结果显然经过自定义分词有所好转。而石墨/烯分词错误。

词性标注

```

print("-"*40)
result = pseg.cut(test_sent)
for w in result:
    print(w.word, "/", w.flag, ", ", end=' ')
print("\n" + "-"*40)
terms = jieba.cut('easy_install is great')
print('/'.join(terms))
terms = jieba.cut('Python 的正则表达式是好用的')
print('/'.join(terms))
print("-"*40)
# 结果
=====
李小福 / nr , 和 / c , 李铁军 / x , 是 / v , 创新办 / i , 主任 / b
, 也 / d , 是 / v , 云计算 / x ,
方面 / n , 的 / uj , 专家 / n , ; / x , / x , 什么 / r , 是 / v
, 八一双鹿 / nz , / x , 例如 / v ,
我 / r , 输入 / v , 一个 / m , 带 / v , “ / x , 韩玉赏鉴 / nz ,
” / x , 的 / uj , 标题 / n , , / x ,
在 / p , 自定义词 / n , 库中 / nrt , 也 / d , 增加 / v , 了 / ul ,
此 / r , 词 / n , 为 / p , N / eng ,

```

```

类 / q , / x , 「 / x , 台中 / s , 」 / x , 正确 / ad , 应该 / v ,
不 / d , 会 / v , 被 / p , 切开
/ ad , 。 / x , mac / eng , 上 / f , 可 / v , 分出 / v , 「 / x ,
石墨烯 / x , 」 / x , ; / x ,
此时 / c , 又 / d , 可以 / c , 分出 / v , 来 / zg , 凯特琳 / x ,
了 / ul , 。 / x ,

```

```
=====
```

```

easy_install/ /is/ /great
Python/ /的/正则表达式/是/好用/的

```

```
=====
```

结果分析：李小福 / nr ， 李铁军 / x 都是名字，属于名词，而李铁军 / x 显然词性不对，这是由于刚刚 jieba.add_word(‘李铁军’) 时候，没有进行词性参数输入，我们看看 jieba.add_word(‘李铁军’) 源码：

```

def add_word(self, word, freq=None, tag=None)
jieba.add_word('李铁军',tag='nr')修改后结果

```

再次查看结果：

```
=====
```

```

李小福 / nr , 和 / c , 李铁军 / nr , 是 / v , 创新办 / i , 主任 / b
, 也 / d , 是 / v , 云计算 / x , 方面 / n ,
的 / uj , 专家 / n , ; / x , / x , 什么 / r , 是 / v , 八一双鹿
/ nz , / x , 例如 / v , 我 / r , 输入 / v ,
一个 / m , 带 / v , “ / x , 韩玉赏鉴 / nz , ” / x , 的 / uj , 标
题 / n , , / x , 在 / p , 自定义词 / n ,
库中 / nrt , 也 / d , 增加 / v , 了 / ul , 此 / r , 词 / n , 为 /
p , N / eng , 类 / q , / x , 「 / x ,
台中 / s , 」 / x , 正确 / ad , 应该 / v , 不 / d , 会 / v , 被 /
p , 切开 / ad , 。 / x , mac / eng ,
上 / f , 可 / v , 分出 / v , 「 / x , 石墨烯 / x , 」 / x , ; / x
, 此时 / c , 又 / d , 可以 / c ,
分出 / v , 来 / zg , 凯特琳 / x , 了 / ul , 。 / x ,

```

```
=====
```

自定义调整词典

```
# test frequency tune
testlist = [
('今天天气不错', ('今天', '天气')),
('如果放到post中将出错。', ('中', '将')),
('我们中出了一个叛徒', ('一', '个')),
]
for sent, seg in testlist:
    print('/'.join(jieba.cut(sent, HMM=False)))
    word = ''.join(seg)
    print('%s Before: %s, After: %s' % (word, jieba.get_FREQ(word),
    jieba.suggest_freq(seg, True)))
    print('/'.join(jieba.cut(sent, HMM=False)))
    print("-"*40)
```

结果:

=====

今天天气/不错

今天天气 Before: 3, After: 0

今天天气/不错

如果/放到/post/中将/出错/。

中将 Before: 763, After: 494

如果/放到/post/中/将/出错/。

我们/中/出/了/一个/叛徒

一个 Before: 142747, After: 454

我们/中/出/了/一/个/叛徒

结果分析: 列表中的每一条数据如 ('今天天气不错', ('今天', '天气')), 其中 ('今天', '天气') 是调整分词颗粒精度的。如第三句正常分词: 我们/中/出/了/一个/叛徒。我们假设某些情况下 “一” 和 “个” 分别分词, 可以做如上处理。

使用 `add _ word(word, freq=None, tag=None)` 和 `del _ word(word)` 可在程序中动态修改词典。使用 `suggest_freq(segment, tune=True)` 可调节单个词语的词频, 使其能 (或不能) 被分出来。注意: 自动计算的词频在使用 HMM 新词发现功能时可能无效。

自定义调节词典解决歧义分词问题

```
>>> import jieba
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\cuitbnc\AppData\Local\Temp\jieba.cache
Loading model cost 1.069 seconds.
Prefix dict has been built succesfully.
如果/放到/post/中将/出错/。
>>> jieba.suggest_freq(('中', '将'), True)
494
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
如果/放到/post/中/将/出错/。
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台/中/」/正确/应该/不会/被/切开
>>> jieba.suggest_freq('台中', True)
69
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台中/」/正确/应该/不会/被/切开
```

词性标注

`jieba.posseg.POSTokenizer(tokenizer=None)` 为新建自定义分词器，`tokenizer` 参数可指定内部使用的 `jieba.Tokenizer` 分词器。`jieba.posseg.dt` 为默认词性标注分词器，标注句子分词后每个词的词性，采用和 `ictclas` 兼容的标记法。用法示例如下：

```
>>> import jieba.posseg as pseg
>>> words = pseg.cut("我爱北京天安门")
>>> for word, flag in words:
...     print('%s %s' % (word, flag))
...
我 r
爱 v
北京 ns
天安门 ns
```

第 10 章

数据预处理

本章导读：无论在机器学习还是在自然语言处理的过程中，数据预处理都是举足轻重的关键步骤，它常决定一个好模型的训练难易程度，甚至决定一个项目的成败。依笔者愚见，相比于机器学习，数据预处理对自然语言处理而言更为重要。为什么这样说？因为自然语言对真实信息进行了额外的外表包装，只有剥开这层包装，才能更容易地挖掘到真实信息。在自然语言处理过程中，数据预处理过程大致包括四个步骤：数据清洗、分词与数据转换、特征提取与构造、特征降维。

10.1 数据清洗

在自然语言处理的过程中，通常我们得到的数据并非是完全“干净”的，因此就需要将语言信息中的干扰信息除去，这样才能更有效地挖掘出其中包含的重要信息。文本内容形式多种多样，常见的包括 TXT 文本、HTML 文本、XML 文本、word 文档、excel 文档等。在机器学习过程中，数据清洗的内容步骤是相当复杂的；但对文本来说，清洗的目的却很明确，即排除非关键信息。我们只需要保留文本内容所阐述的文字信息即可，并同时尽可能减小这些信息对算法模型构建的影响。以 HTML 文本为例，HTML 文本中有很多 HTML 标签，如“body”、“title”、“p”等，毫无疑问将这些标签作为有效信息来训练模型是不可取的，因为它们与文本内容所要表达的主题没有任何关联。那么清洗这些标签需要自己写很多代码吗？答案是当然不用！因为在 Java 语言中，有 Jsoup、HtmlParser、Apache tika、HtmlCleaner 及 XPath 等功能包，它们可以来帮助你完成对 HTML 文本的清洗工作。而在 Python 语言中，有 BeautifulSoup、SGMLParser、HTMLParaer 等功能包也能完成清洗工作。所以无论是用 Java 语言，还是 Python 语言来进行自然语言处理工程，行业先驱早已给后辈们种下了无数的“乘

凉树”。当然在实际情况中我们可能还会遇到一些特殊的情况，以至于这些功能包并不能很好地帮我们完成所有的清洗工作；但是不用担心，正则表达式会满足你的需求。

10.2 分词处理

在对文本进行清洗之后，需要给文本分词，这样能更容易挖掘出文本内容中的一些特征。在上一章中我们提到了一些常用的分词工作包，如 Stanford NLP 分词、jieba 分词、中科院 NLPPIR 汉语分词等。这些分词工具的分词效果都很不错，具体使用起来也很简单。比如有这样一句话：“自然语言处理（NLP）是计算机科学、人工智能、语言学关注计算机和人类（自然）语言之间的相互作用的领域！”，采用 jieba 分词后的效果：“自然语言处理/n (/x NLP/eng)/n 是/v 计算机科学/n ,/x 人工智能/n ,/x 语言学/n 关注/v 计算机/n 和/c 人类/n (/x 自然/d)/x 语言/n 之间/f 的/uj 相互作用/l 的/uj 领域/n !/x”。在分词时，一些停词和标点符号通常是需要排除的，如“是”“和”“的”“,”“(”、“)”等。但在某些业务情况下，有些停词和标点符号却对业务的建模有一定的帮助，比如对微博内容的情感判断。

10.3 特征构造

在做文本特征构造的时候，需要先了解“向量空间模型”（VSM: Vector Space Model）这个概念。向量空间模型把文本内容的处理简化为向量空间中的向量运算，并且以空间上的相似度表达语义的相似度，直观易懂。自然语言处理中，几乎所有的特征构造方法都基于这个概念。

（1）词袋模型。

在传统的词袋模型中，对于每一个词采用 one-hot 稀疏编码的形式。假设目标语料中共有 N 个唯一确认的词，那么需要一个长度为 N 的词典，词典的每一个位置表达了文本中出现的某一个词。在某一种特征表达下，比如词频、binary、tf-idf 等，我们可以将任意词或者文本表达在一个 N 维的向量空间里。

（2）N-gram 模型。

N-gram 是一种统计语言模型，其作用是根据前 $n-1$ 个 item 来预测第 n 个 item。N-gram 被广泛地应用于语音识别、输入法、分词等任务，当 n 分别为 1、2、3 时，又分别称为一元语法（unigram）、二元语法（bigram）与三元语法（trigram）。现在一些学者用 N-gram 模型

来构造分类任务的数据特征，笔者也利用 N-gram 模型来构造过新闻突发事件判断的数据特征，其判断的效果还是相当不错的。

10.4 特征降维与选择

在数据特征构造完成后，通常需要对特征数据集进行特征降维和特征选择。二者的目标都是要使数据集的特征维数减少，但二者又存在一定的区别。数据降维，一般说的是维数约简（Dimensionality reduction）。它的思路是：将原始高维特征空间里的点向一个低维空间投影，从而使新空间的维度低于原特征空间，从而达到减少维数的目的。在这个过程中，特征发生了根本性的变化，原始的特征消失了（虽然新的特征也保持了原特征的一些性质）。而特征选择，是从 n 个特征中选择 d ($d < n$) 个出来，其他的 $n-d$ 个特征舍弃，因此新的特征只是原来特征的一个子集，没有被舍弃的 d 个特征没有发生任何变化。

10.4.1 特征降维

当特征数据集构造完成后，可能会出现特征矩阵过大，从而导致计算量大、训练时间长等一系列问题，因此降低特征矩阵维度也是必不可少的。机器学习中常见的特征降维方法：L1 惩罚项的模型、主成分分析法（PCA）、线性判别分析（LDA）。PCA 和 LDA 有很多的相似点，它们的共同原理是将原始样本映射到维度更低的样本空间中，PCA 是一种无监督的降维方法，而 LDA 是一种有监督的降维方法。在自然语言处理中，常用的方法就是主题模型。主题模型同时具备了降维和语义表达的效果，比如 LSI、LDA、PLSA、HDP 等统计主题模型。这些模型寻求文本在低维空间（不同主题上）的表达，在降低维度的同时，尽可能保留原有文本的语义信息，主题模型在处理中长度文本分类任务时特别有效。

10.4.2 特征选择

当特征数据集特征过多时，选择相对更有意义的特征来进行建模是很有必要的。特征选择是去掉无关特征、保留相关特征的过程，换句话说，是从所有的特征中选择一个最好特征子集的过程。特征选择本质上可以认为是降维的过程。

(1) Filter (过滤法): 按照发散性或者相关性对各个特征进行评分, 根据设定阈值或者待选择阈值的个数来选择特征, 如方差选择法、相关系数法、卡方检验法、互信息法。方差选择法: 使用方差选择法, 先要计算各个特征的方差, 然后根据阈值, 选择方差大于阈值的特征。相关系数法: 使用相关系数法, 先要计算各个特征对目标值的相关系数及相关系数的 P 值。卡方检验法: 经典的卡方检验主要是检验定性自变量对定性因变量的相关性。假设自变量有 N 种取值, 因变量有 M 种取值, 考虑自变量等于 i 且因变量等于 j 时样本频数的观察值与期望的差距, 从而构建统计量。互信息法: 经典的互信息也是评价定性自变量对定性因变量的相关性的。

(2) Wrapper (包装法): 根据目标函数 (通常是预测效果评分), 每次选择若干特征或者排除若干特征, 如递归特征消除法。递归特征消除法: 递归消除特征法使用一个基模型来进行多轮训练, 每轮训练后, 消除若干权值系数的特征, 再基于新的特征集进行下一轮训练。

(3) Embedded (嵌入法): 先使用某些机器学习的算法模型进行训练, 得到各个特征的权值系数, 再根据系数从大到小选择特征。类似于 Filter 方法, 但是通过训练来确定特征的优劣。基于惩罚项的特征选择法: 使用带惩罚项的基模型, 除了筛选出特征, 同时进行了降维。使用 `feature_selection` 库的 `SelectFromModel` 类结合带 L1 惩罚项的逻辑回归模型, 如基于惩罚项的特征选择法、基于树模型的特征选择法。

基于树模型的特征选择法: 树模型 (随机森林、GBDT 和 Xgboost) 也可用来作为基模型进行特征选择, 使用 `feature_selection` 库的 `SelectFromModel` 类来结合树模型。

(4) 深度学习方法: 从深度学习模型中选择某一神经层的特征, 用来进行最终目标模型的训练。

10.5 简单实例

前面几个小节依次描述了自然语言处理中的数据预处理需要的流程步骤, 为了让大家能更清楚如何进行实际处理, 接下来用一个简单的实例场景来实战一下。

场景介绍: 假设你是一家国内做资讯业务的新媒体公司的研发人员, 你的直接领导交给你一个研发任务, 希望你完成新闻自动分类这个功能, 并复制了一份已分类的新闻资讯数据给你, 这些数据是利用爬虫工具在互联网上的多个新闻网站上爬取下来的。这时, 你该如何完成这个任务呢? 不用着急, 按照前面几个小节知识就可以基本帮你完成这个任务。

数据清洗

一般通过爬虫爬取的文本都包含 HTML 标签，我们需要把这些标签清洗掉，保证文本内容的正确性。例如，以下一段文本：

```
<div class="video" mode="player" site="qiniu" src="https://dqdvideofile.
qnssl.com/QYibRDG7_7101932251.mp4?sign=6fdec7415956db29a3f54a4fa5a82eb8&
t=5abf5925" title="" thumb="https://o6yh618n9.qnssl.com/
QYibRDG7_7101932251.mp4?vframe/jpg/offset/1" hash="
b0b0748feeb14182bd16da14155d0cad"></div>
</p><p>北京时间2013年4月23日英超第34轮的比赛中，曼联3-0大胜阿斯顿维拉提
前4轮锁定球队历史第20个联赛冠军。比赛中范佩西上演了精彩的帽子戏法，可谓
是居功至伟。下面就一起回顾一下这场经典的比赛吧。</p></div>
```

清洗标签的方法很多，可以采用别人提供的功能包，也可以采用正则表达式自定义处理。个人比较偏爱使用后者来清洗文本中的 HTML 标签。利用正则表达式对 HTML 文本处理，Python 代码如下：

```
# 正则表达式过滤处理HTML中的标签
# @param htmlstr HTML字符串
def filter_tags(htmlstr):
    # 先过滤CDATA
    re_cdata = re.compile('/*!CDATA\[ [ ]* /\] > ', re.I) #匹配CDATA
    re_script = re.compile('<\s*script[ ^>]*>[ ^<]*<\s*/\s*script\s*>', re
.I)
    # 过滤Script
    re_style = re.compile('<\s*style[ ^>]*>[ ^<]*<\s*/\s*style\s*>', re.I)
    # 过滤style
    re_br = re.compile('<br\s*/?>')
    # 处理换行
    re_h = re.compile('</?\w+[ ^>]*>')
    # HTML标签
    re_comment = re.compile('<!--[ ^>]*-->')
    # HTML注释
    s = re_cdata.sub('', htmlstr)
    # 去掉CDATA
    s = re_script.sub('', s) # 去掉SCRIPT
    s = re_style.sub('', s)
```

```

# 去掉style
s = re_br.sub('', s)
# 将br转换为换行
s = re_h.sub('', s) # 去掉HTML 标签
s = re_comment.sub('', s)
# 去掉HTML注释
# 去掉多余的空行
blank_line = re.compile('\n+')
s = blank_line.sub('', s)
blank_line_l = re.compile('\n')
s = blank_line_l.sub('', s)
blank_kon = re.compile('\t')
s = blank_kon.sub('', s)
blank_one = re.compile('\r\n')
s = blank_one.sub('', s)
blank_two = re.compile('\r')
s = blank_two.sub('', s)
blank_three = re.compile(' ')
s = blank_three.sub('', s)
s = replaceCharEntity(s) # 替换实体
return s

# 使用正常的字符替换HTML中特殊的字符实体。可以添加新的实体字符到
CHAR_ENTITIES中，处理更多HTML字符实体。
# @param htmlstr HTML字符串
def replaceCharEntity(htmlstr):
    CHAR_ENTITIES = {'nbsp': ' ', '160': ' ',
                      'lt': '<', '60': '<',
                      'gt': '>', '62': '>',
                      'amp': '&', '38': '&',
                      'quot': '"', '34': '"', }

    re_charEntity = re.compile(r'&#?(?P<name>\w+);')
    sz = re_charEntity.search(htmlstr)
    while sz:
        entity = sz.group() # entity全称，如>
        key = sz.group('name') # 去除&;后entity,如>为gt

```

```

try:
    htmlstr = re_charEntity.sub(CHAR_ENTITIES[key], htmlstr, 1)
    sz = re_charEntity.search(htmlstr)
except KeyError:
    # 以空串代替
    htmlstr = re_charEntity.sub('', htmlstr, 1)
    sz = re_charEntity.search(htmlstr)
return htmlstr

```

清洗过后，基本上就是纯文本的文字描述了。

北京时间2013年4月23日英超第34轮的比赛中，曼联3-0大胜阿斯顿维拉提前4轮锁定球队历史第20个联赛冠军。比赛中范佩西上演了精彩的帽子戏法，可谓居功至伟。下面就一起回顾一下这场经典的比赛吧。

分词处理

分词工具很多，常用的就是 jieba 分词，安装和使用也很简单，这里不再过多描述，来看看 jieba 的分词效果：

```

北京/ns 时间/n 2013/m 年/m 4/m 月/m 23/m 日/m 英超/nr 第/m 34/m 轮/q
的/uj 比赛/vn 中/f , /x 曼联/ns 3/x -/x 0/x 大胜/nr 阿斯顿维拉/ns 提前/v
4/m 轮/n 锁定/v 球队/n 历史/n 第/m 20/m 个/m 联赛/vn 冠军/n 。/x 比赛/vn
中/f 范佩西/nr 上演/v 了/ul 精彩/n 的/uj 帽子戏法/n , /x 可谓/v
居功至伟/nr 。/x 下面/f 就/d 一起/m 回顾/v 一下/m 这场/mq 经典/n 的/uj
比赛/vn 吧/y 。/x

```

分词之后的格式为：单词/词性。通常在某些业务建模情况下，词性可能用不上，但是词性对分析特征提供了参考。例如，要判断这篇文章的情感色彩，那么根据词性就可以将“这场”“下面”等干扰词排除。

分析分词之后的数据不难发现，有些词和标点符号其实在文本中都是很常见的，用专业的一点术语来说，这些词和标点符号对分析处理问题没有多大用处或根本没有作用，甚至会产生一定的负面干扰。这不需要什么公式、定理来验证说明，经验就告诉我们这些词和标点应该剔除，所以就有了停用词和去除停用词的说法。当然，为了尽可能地排除干扰数据，通常在分词之后都要再进行停用词处理，处理后的结果如下：

北京/ns 时间/n 英超/nr 比赛/vn 曼联/ns 大胜/nr 阿斯顿维拉/ns 提前/v
锁定/v 球队/n 历史/n 联赛/vn 冠军/n 比赛/vn 范佩西/nr 上演/v 精彩/n
帽子戏法/n 可谓/v 居功至伟/nr 下面/f 一起/m 回顾/v 一下/m 这场/mq 经典/n
比赛/vn

特征构造

分词并进行停用词处理之后,就需要对数据进行特征构造,也可以说成特征转换。在处理文本数据时,基于向量空间模型这一概念,可以通过词袋模型、N-gram 模型、词向量来进行特征构造。例如,通过求每一个单词的 tf-idf 值,就可以用 tf-idf 值代替单词,将文本转换成一个向量。具体步骤如下:

(1) 将所有分词后的文本,按单词为最小单位去重,构成一个词汇表。

(2) 计算词汇表中的每一个单词的 tf-idf 值,可以将 tf-idf 值较小词直接剔除,这样可以减少向量的维度。

(3) 用词汇表将每一个文本转化成维度一样的向量,并且非零值即为单词所对应的 tf-idf 值。

除此之外,也可以用 N-gram 模型和词向量来讲单词转化成对应的数值向量。但是,实验证明,针对新闻分类业务,tf-idf 和词向量方法更适合。两者的区别在于:tf-idf 方法是直接值替换单词,而词向量方法需要构建一个词向量模型,通过模型将单词映射在多个维度上。简单来说,就是用一个 n 维向量来表示一个单词。比如,“曼联”这个词,tf-idf 值为 0.113,那么其在整个文本中被表示为 0.113;而词向量方法可能会使“曼联”这个词被表示为 (0.103, -0.011, 0.232, ..., 0.145)。值得一提的是:针对新闻分类问题可以直接采用基于概率分布的主题模型来完成文本的特征构造,这也是常用且效果很好的方法,由于主题模型涉及的知识点比较多,这里就不再细说了。

特征降维与选择

在文本的数据特征构造完成以后,一篇文本可能会被表示成一个成百上千维的向量,通常情况下会出现数据集稀疏的情况,那么这时就需要对数据集进行降维或选择。方法有很多,基于模型的特征降维或特征选择效果通常会更好。例如,上文中领导提供的数据,通过清洗、分词、特征构造之后,形成了一个 (M, N) 的特征数据集,采用随机森林算法可以将特征数据集将到 (m, N) , 其中 $m \ll M$ 。这样就可以直接基于这个 (m, N) 的特征数据集来构建分类模型了。

10.6 本章小结

数据预处理的整个步骤流程在自然语言处理的工程中要比机器学习的工程中精简一些，最大的区别就是数据清洗和特征构造，这两个过程也是至关重要的。而且在自然语言处理中特征构造是否良好，很大程度取决于所构造的特征数据集的数据特性与文本内容语义吻合程度的高低。比如，文本情感分类和文本内容分类都是属于分类范畴，但是对于同一种算法（参数都调整到最优），在两个不同分类的业务下，得到的结果可能会相差很大，通过仔细分析，不难发现造成这种差异其实根本的原因就是构造出来的特征数据集的数据模式没有很好地契合文本的真实语义，这也是自然语言处理的最大难点之处。

第 11 章

马尔可夫模型

本章导读：笔者最早接触马尔可夫模型的定义是在阅读吴军先生的《数学之美》时，当时觉得它深奥难懂且没什么用处。后来笔者学习了自然语言处理，才真正使用到隐马尔可夫模型，并体会到它的奇妙之处。马尔可夫模型在处理序列分类时具有强大的功能，可以解决词类标注、语音识别、句子切分、字素音位转换、局部句法剖析、语块分析、命名实体识别、信息抽取等问题。此外，它还广泛应用于自然科学、工程技术、生物科技、公用事业、信道编码等多个领域。

11.1 马尔可夫链

11.1.1 马尔可夫简介

安德烈·马尔可夫，俄罗斯人，物理-数学博士、圣彼得堡科学院院士、彼得堡数学学派的代表人物，以数论和概率论方面的工作著称，他的主要著作有《概率演算》等。1878 年荣获金质奖章，1905 年被授予功勋教授称号。在数论方面，他研究了连分数和二次不定式理论，还解决了很多难题。在概率论中，他发展了矩阵法，扩大了大数律和中心极限定理的应用范围。马尔可夫最重要的工作是在 1906—1912，提出并研究了一种能用数学分析方法研究自然过程的一般图式——马尔可夫链，同时开创了对一种无后效性的随机过程——马尔可夫过程的研究。马尔可夫经多次观察试验发现，一个系统的状态转换过程中第 n 次转换获得的状态常取决于前一次（第 $n-1$ 次）试验的结果。马尔可夫进行深入研究后指出：对于一个系统，在由一个状态转至另一个状态的过程中存在着转移概率；并且这种转移概率可

以依据其紧接的前一种状态推算出来，与该系统的原始状态和此次转移前的马尔可夫过程无关。马尔可夫链的理论与方法现已经广泛应用于自然科学、工程技术和公用事业中。

11.1.2 马尔可夫链的基本概念

序列分类器

序列分类器（又称序列标号器）是给序列中的某个单元指派类或者标号的模型。马尔可夫模型（又称显马尔可夫模型 VMM）和隐马尔可夫模型（HMM）都是序列分类器。词类标注、语音识别、句子切分、字素音位转换、局部句法剖析、语块分析、命名实体识别、信息抽取等都属于序列分类。

随机过程的两层含义

随机过程是一个时间函数，其随着时间变化而变化；随机过程在每个时刻上的函数值是不确定的、随机的，即每个时刻上函数值按照一定的概率进行分布。

独立链

若随机过程中各语言符号或者词是独立的、不相互影响，则称这种链是独立链；若各语言符号或者词彼此有关，则是非独立链。

等概率独立链与非等概率独立链

在独立链中，若各语言符号或者词是等概率出现的，则是等概率独立链；若各语言符号或者词是非等概率出现的，则为非等概率独立链。

马尔可夫链

马尔可夫过程：在独立链中，前面的语言符号对后面的语言符号无影响，是无记忆、没有后效的随机过程。在已知当前状态下，过程的未来状态与它的过去状态无关，这种形式就是马尔可夫过程。

马尔可夫链：在随机过程中，每个语言符号的出现概率不相互独立，每个随机试验的当前状态依赖于此前状态，这种链就是马尔可夫链。链的解析：链可以当作一种观察序列，诸

如：“2016 年是第 31 届夏季奥运会的举办之年”，就可以看作一个字符串链。如果上述字符串中每个字符的出现是随机、独立的，就是独立链；如果每个字符的出现与前面字符相关，即不独立并具有依赖性，就称之为马尔可夫链。

N 元马尔可夫链：考虑前一个语言符号对后一个语言符号出现概率的影响，这样得出的语言成分的链叫作一重马尔可夫链，也是二元语法。考虑前两个语言符号对后一个语言符号出现概率的影响，这样得出的语言成分的链叫作二重马尔可夫链，也是三元语法。考虑前三个语言符号对后一个语言符号出现概率的影响，这样得出的语言成分的链叫作三重马尔可夫链，也是四元语法。类似的，考虑前 $(4, 5, \dots, N-1)$ 个语言符号对后一个语言符号出现概率的影响，这样得出的语言成分的链叫作 $(4, 5, \dots, N-1)$ 重马尔可夫链，也是 $(5, 6, \dots, N)$ 元语法。

有限自动机

马尔可夫链在数学上描述了自然语言句子的生成过程，是一个自然语言形式的早期模型。后来 N 元语法的研究，都是建立在马尔可夫模型的基础之上的。马尔可夫链和隐马尔可夫模型都是有限自动机（状态集合状态之间的转移集）的扩充。

加权有限状态机

加权有限状态机中每个弧与一个概率有关，这个概率说明通过这个弧的可能性，且某一个点出发的弧具有归一化的性质，即某点出发的弧概率之和为 1。

注意：马尔可夫链不能表示固有歧义的问题，当概率指派没有歧义时，马尔可夫链才有用。

马尔可夫链描述

(1) 具有初始状态和终结状态的马尔可夫链描述如表 11-1 所示。

表 11-1 具有初始状态和终结状态的马尔可夫链

$Q = q_1 q_2 \dots q_N$	状态 N 是集合
$Q = a_{01} a_{02} \dots a_{n1} \dots a_{nm}$	转移概率矩阵 A ，每一个 a_{ij} 表示从状态 i 转移到状态 j 的概率，对于任意的 i ，有 $\sum a_{ij} = 1$
q_0, q_F	初始化状态和终结状态。它们与观察值没有关系

(2) 没有初始状态和终结状态的马尔可夫链描述如表 11-2 所示。

表 11-2 没有初始状态和终结状态的马尔可夫链

$\pi = \pi_1 \pi_2 \dots \pi_N$	在状态上初始化概率分布, π_i 表示马尔可夫链在状态 i 开始的概率, 有 $\sum \pi_i = 1$
$Q_A = \{q_x, q_y, \dots\}$	合法接受状态的集合

在一个一阶马尔可夫链中, 我们假设一个特定的概率只与它前面一个状态有关, 马尔可夫假设可以表示为

$$p(q_i | q_1 \dots q_{n-1}) = p(q_i | q_{i-1})$$

从一个状态 i 出发的所有弧的概率之和为 1, 即

$$\sum_{j=1}^n a_{ij} = 1 (\forall i)$$

11.2 隐马尔可夫模型

11.2.1 形式化描述

爱依斯讷 (Jason Eisner) 对隐马尔可夫模型的描述

隐马尔可夫模型即一个隐藏马尔可夫链随机生成不可观察的随机序列, 再由各个状态生成一个观测随机序列的过程。隐藏马尔可夫链随机生成的状态序列称为状态序列; 每个状态生成一个观测的随机序列, 称为观测序列。序列的每个位置可以看作一个时刻, 它可以被形式化地描述为一个五元组 $HMM = \langle Q, A, O, B, \pi \rangle$ 。Jason Eisner 对隐马尔可夫模型的描述如表 11-3 所示。

表 11-3 HMM 形式化描述

五 元 组	描 述
$Q = q_1 q_2 \dots q_N$	状态 N 是集合
$A = a_{01} a_{02} \dots a_{n1} \dots a_{mn}$	转移概率矩阵 A ，每一个 a_{ij} 表示从状态 i 转移到状态 j 的概率，对于任意的 i ，有 $\sum a_{ij} = 1$
$O = o_1 o_2 \dots o_T$	观察 T 的序列，也叫发射概率，每一个观察从词汇 $V = v_1 v_2 \dots v_v$ 中取值
$B = b_i(o_t)$	发射概率，每一个观察似然度表示从状态 i 生成观察 o_t 的概率
q_0, q_F	初始化状态和终结状态

拉宾纳（Rabiner）关于隐马尔可夫模型思想的三个问题

问题 1（似然度问题）：给定一个 HMM $\lambda = (A, B)$ 和一个观察序列 O ，确定观察序列的似然度问题 $P(O|\lambda)$ 。

问题 2（解码问题）：给定一个观察序列 O 和一个 HMM $\lambda = (A, B)$ ，找出最好的隐藏状态序列 Q 。

问题 3（学习问题）：给定一个观察序列 O 和一个 HMM 中的状态集合，自动学习 HMM 的参数 A 和 B 。

根据隐马尔可夫模型的描述，可以将一个长度为 T 的观测序列的生成过程描述如下所示。

输入：HMM $\lambda = (A, B, \pi)$
输出：观察序列 $O = (o_1 o_2 \dots o_T)$
初始状态分布 π 产生初始状态 i_1
设置 $t = 1$
状态 i_t 下的观测概率分布 $b_i(k)$ 生成 O_t
状态 i_t 下的转移概率分布 $a_{i_t i_{t+1}}$ ，生成状态 $i_{t+1}, i_{t+1} = 1, 2, \dots, N$
当 $t = t + 1$ 时，如果 $t < T$ ，则程序继续执行；否则终止程序

综上所述，一个已知的 HMM 模型由参数 (N, T) 、观测序列和三个概率 (A, B, π) 构成，下文皆用 $\lambda = (A, B, \pi)$ 表示完整的 HMM 模型参数。

11.2.2 数学形式描述

在命名实体识别过程中，序列标注是一个重要的子任务。利用 HMM 模型去解决类似序列标注这样的问题，给定一个观测序列 $X = \{x_1, x_2, x_3, \dots, x_n\}$ ，为求得最佳序列标注为 $Y = y_1, y_2, y_3, \dots, y_n$ ，要求条件概率 $P(Y|X)$ 最大。由朴素贝叶斯公式可知：

$$p(Y|X) = \frac{P(X, Y)}{P(X)} = \frac{P(Y)P(X|Y)}{P(X)}$$

在命名实体识别中， X 是给定的一个文本序列，文本粒度可以是一个句子或者一段话，待观测的值 x_1 到 x_n 为词，而 $P(X)$ 对于所有类别都是一样的，可以看作一个常量从而忽略不考虑。则上式可以简化为

$$p(Y|X) = P(X, Y) = P(Y)P(X|Y)$$

隐马尔可夫模型本质上就是求解联合概率。则由上式可得：

$$P(Y|X) = P(x_{1,n}, y_{1,n}) = \prod_{t=1}^n p(x_t, y_t | x_{1,t-1}, y_{1,t-1}) = \prod_{t=1}^n p(x_t | x_{1,t-1}, y_{1,t}) p(y_t | x_{1,t-1}, y_{1,t-1})$$

其中 $x_{1,t} = x_1, x_2, x_3, \dots, x_t$ ， $y_{1,t} = y_1, y_2, y_3, \dots, y_t$ ， $1 \leq i \leq n$ 。给出了非独立假设情况下的求解命名实体识别的概率模型，命名实体识别任务就转化为求解 Y^* ，期望 $P(Y|X)$ 值最大。

公式假设理想状态下的命名实体识别的概率模型，实际上这种参数估计是不科学的。故而我们给出下面两个独立假设。

(1) 马尔可夫假设。

假设求解状态只是跟它前面的状态 $N-1$ 有关，即：

$$p(y_t | x_{1,t-1}, y_{1,t-1}) \approx p(y_t | y_{t,t-1}) \approx p(y_t | y_{t-N+1}, y_{t-N+2}, \dots, y_{t-1})$$

(2) 输出独立假设。

一个输出观察的概率只与产生该观察的状态有关，与其他状态无关，即：

$$p(x_t|x_{1,t-1}, y_{1,t}) \approx p(x_t|y_t)$$

可知一阶 HMM 公式可以化简为

$$P(Y|X) = P(x_{1,n}, y_{1,n}) = \prod_{t=1}^n p(x_t, y_t|x_{1,t-1}, y_{1,t-1}) = \prod_{t=1}^n p(y_t|y_{t-1})p(x_t|y_t)$$

其中, $p(x_t|y_t)$ 为发射概率, $p(y_t|y_{t-1})$ 为转移概率。

11.3 向前算法解决 HMM 似然度

11.3.1 向前算法定义

向前算法的递归定义如下所示。

1. 初始化

$$\alpha_1(j) = a_{0,j}b_j(o_1); 1 \leq j \leq N$$

2. 递归 (由于状态 0 和状态 F 没有发射概率)

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i)a_{ij}b_i(o_t); 1 \leq j \leq N, 1 \leq t \leq T$$

3. 结束

$$P(O|\lambda) = \alpha_T(q_E) = \sum_{i=1}^N \alpha_T(i)a_{iF}$$

11.3.2 向前算法原理

向前算法解决问题 1 (似然度问题): 给定一个 HMM $\lambda = (A, B)$ 和一个观察序列 O , 确定观察序列的似然度问题 $P(O|\lambda)$ 。对于马尔可夫链, 表面观察和实际隐藏是相同的, 只需要记录观察的状态, 并把加权 (弧上) 的对应概率相乘即可。而隐马尔可夫模型就不那么简单了, 因为状态是隐藏的, 所以我们并不知道隐藏的状态序列是什么。

简化下问题：假如我们知道天气的冷热状况，并且知道小明吃冰激凌的数量，然后观察序列似然度。例如，对于给定的隐藏状态序列“hot hot cold”，我们来计算观察序列“3 1 3”的输出似然度。

如何进行计算？首先，隐马尔可夫模型中，每个隐藏状态只产生一个单独的观察，即一一映射。隐藏状态序列与观察序列长度相同，即给定这种一对一的映射及马尔可夫假设，对于一个特定隐藏状态序列 $Q = q_0q_1\dots q_T$ 及一个观察序列 $o = o_1o_2\dots o_T$ ，观察序列的似然度为

$$P(O|Q) = \sum_{i=1}^T P(o_i|q_i)$$

故从隐藏状态“hot hot cold”到所吃冰激凌观察序列“3 1 3”的向前概率为

$$P(3 \ 1 \ 3|\text{hot hot cold}) = P(3|\text{hot})P(1|\text{hot})P(3|\text{cold}) = 0.4 \times 0.2 \times 0.1 = 0.008$$

实际上，隐藏状态序列“hot hot cold”是我们的假设，因为我们并不知道隐藏状态序列，我们要考虑的是所有可能的天气序列。如此一来，我们将计算所有可能的联合概率，但这样的计算特别复杂。我们来计算天气序列 Q 生产一个特定的冰激凌事件序列 O 的联合概率：

$$P(O, Q) = P(O, Q) \times P(Q) = \sum_{i=1}^n p(o_i|q_i) \times \sum_{i=1}^n P(q_i|q_{i-1})$$

如果隐藏序列只有一个是“hot hot cold”，那么我们的冰激凌观察“3 1 3”和一个可能的隐藏状态“hot hot cold”的联合概率为

$$P(3 \ 1 \ 3|\text{hot hot cold}) = P(\text{hot}|\text{start})P(\text{hot}|\text{hot})P(\text{hot}|\text{cold})P(3|\text{hot})P(1|\text{hot})P(3|\text{cold}) = \\ 0.8 \times 0.7 \times 0.3 \times 0.4 \times 0.2 \times 0.1 = 0.001344$$

$$P(3 \ 1 \ 3) = P(3 \ 1 \ 3|\text{cold cold cold}) + P(3 \ 1 \ 3|\text{cold cold hot}) + P(3 \ 1 \ 3|\text{hot hot cold}) + \\ P(3 \ 1 \ 3|\text{cold hot cold}) + P(3 \ 1 \ 3|\text{hot cold cold}) + P(3 \ 1 \ 3|\text{hot hot hot}) + \\ P(3 \ 1 \ 3|\text{hot cold hot}) + P(3 \ 1 \ 3|\text{cold hot hot})$$

如果我们有 N 个隐藏状态和 T 个观察序列，那么之后我们将得到 N^T 个可能隐藏序列。在实际中 T 往往很大，比如文本处理中可能有几十万个词汇，计算量将是指数上升的。在隐含马尔可夫模型中有一种向前算法可以有效代替这种指数增长的复杂算法，大大降低复杂度。实验证明向前算法的复杂度是 $O(N^2T)$ 。

11.3.3 现实应用：预测成都天气的冷热

向前算法本质上属于动态规划算法，向前算法也适用于生成模型。在向前算法中横向表示观察序列，纵向表示状态序列。

每个单元 $\alpha_t(j)$ 表示对于给定的自动机 λ ，在前面 t 个观察之后，在状态 j 的概率： $\alpha_t(j) = P(o_1o_2...o_t, q_t = j|\lambda)$ ，其中 $\alpha_t(j)$ 表示第 t 个状态是状态 j 的概率。例如，表示状态 1 即数 3 时， q_1 的概率。

上面公式的 3 个因素如表 11-4 所示。

表 11-4 公式参数解释

$\alpha_{t-1}(i)$	前一个状态的概率
a_{ij}	从前面 q_i 到 q_j 的转移概率
$b_j(o_t)$	在给定的当前状态下 j ，观察符号 o 的状态似然度

向前网格如图 11-1 所示。

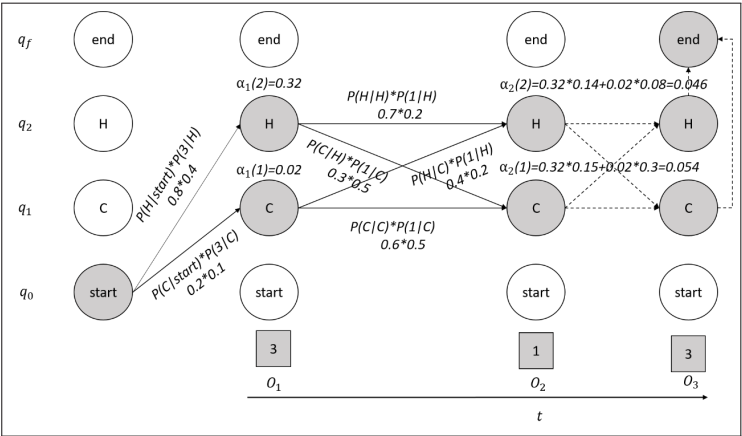


图 11-1 计算冷热事件“3 1 3”的向前网络

(1) 在时间 1 和状态 1 的向前概率：

$$\alpha_1(1) = P(C|start) \times P(3|C) = 0.2 \times 0.1 = 0.02$$

从状态 cold 开始吃 3 根冰激凌的似然度为 0.02。

(2) 在时间 1 和状态 2 的向前概率:

$$\alpha_1(2) = P(H|\text{start}) \times P(3|H) = 0.8 \times 0.4 = 0.32$$

从状态 hot 开始吃 3 根冰激凌的似然度为 0.32。

(3) 在时间 2 和状态 1 的向前概率:

$$\alpha_2(1) = \alpha_1(1) \times P(H|C) \times P(1|C) + \alpha_1(2) \times P(C|H) \times P(1|C) = 0.054$$

从开始到 cold 再到 cold, 以及从开始到 hot 再到 cold 的天气状态, 吃冰激凌 “3 1” 的观察似然度为 0.54。

(4) 在时间 2 和状态 2 的向前概率:

$$\alpha_1(2) = \alpha_1(1) \times P(H|C) \times P(1|H) + \alpha_1(2) \times P(H|H) \times P(1|H) = 0.0464$$

从开始到 cold 再到 hot, 以及从开始到 hot 再到 hot 的天气状态, 吃冰激凌 “3 1” 的观察似然度为 0.0464。

用同样的方法, 我们可以计算时间步 3 和状态步 1 的向前概率, 以及时间步 3 和状态步 2 的概率等, 以此类推, 直到结束。显而易见, 使用向前算法来计算观察似然度可以表示局部观察似然度。这种局部观察似然度比使用联合概率表示的全局观察似然度更有用。

11.4 文本序列标注案例: Viterbi 算法

Viterbi 算法

给定 $\text{HMM}\lambda = (A, B, \pi)$ 和观察序列 $o = (o_1, \dots, o_2, \dots, o_T)$, 求解最大概率的状态序列 $Q = (q_1, \dots, q_2, \dots, q_T)$ 就是 Viterbi 算法解码问题。采用维特比算法求解隐藏序列后面的最大序列, 针对可能的每一个隐藏状态序列, 运用向前算法求解最大似然度的隐藏状态序列, 从而完成解码工作。算法复杂度 $O(N^2T)$ 在状态序列很大时呈指数级别增长, 会造成过大的计算量, 由此采用一种动态规划 Viterbi 算法来降低算法复杂度。

按照观察序列由左向右的顺序, 每个 $V_t(j)$ 表示自动机 λ , HMM 观察了前 t 个状态之后, 每个 $V_t(j)$ 的值递归计算, 并找出最大路径。Viterbi 算法如下所示。

Viterbi 算法

1. 参数: Start 的概率 π 值, 转移矩阵 $\mathbf{B}$, 发射矩阵 $\mathbf{A}$
2. 初始化: 计算开始状态到观察序列 1 的 V 值, $V_j(1) = \pi_i a_{0j} b_j(o_1)$, 其中 $1 \leq j \leq N$
3. 递归: (由于状态 0 和状态 F 没有发射概率) 递归计算状态 2 到状态结束 F 的最大 V 值:
 $V_j(t) = \max V_i(t-1) a_{ij} b_j(o_t)$, 其中 $1 \leq j \leq N, 1 \leq t \leq T$
4. 输入: 初始概率, 发射矩阵 $\mathbf{A}$ 和发射矩阵 $\mathbf{B}$
5. 输出: 最大路径概率
6. 结束: 存储回溯最可能的路径

维特比算法实例解读

例子: 通过吃冰激凌的数量 (观察序列状态) 计算隐藏状态空间的最佳路径 (维特比网络) 如图 11-2 所示。

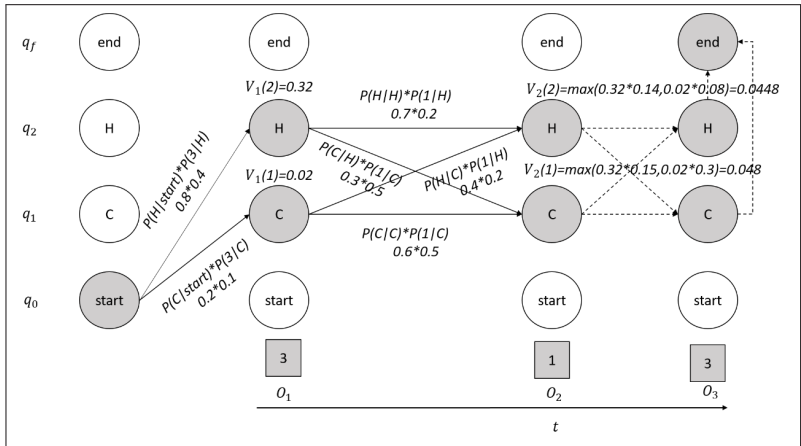


图 11-2 计算隐藏状态空间的最佳路径网络

- 圆圈: 隐藏状态。
- 方框: 观察状态。
- 虚线圆圈: 非法转移。
- 虚线: 计算路径。

◎ q : 隐藏空间状态。

◎ o : 观察时间序列状态。

◎ $V_t(j)$: 在时间步 t 的观察状态下, 隐藏状态 j 的概率。

给定一个 $HMM\lambda = (A, B)$, HMM 把最大似然度指派给观察序列, 算法返回状态路径, 从而找到最优的隐藏状态序列。图 11-2 是小明夏天吃冰激凌 ‘3 1 3’, 据此使用 Viterbi 算法推断最可能出现的天气状态 (天气的热 | 冷)。

(1) 计算时间步 1 的维特比概率。

计算时间步 $t=1$ 和状态 1 的概率:

$$V_1(1) = P(C|\text{start}) \times P(3|C) = 0.2 \times 0.1 = 0.02$$

路径: $\text{start} \rightarrow C$ 的转移概率是 0.02。

计算时间步 $t=1$ 和状态 2 的概率:

$$V_1(2) = P(H|\text{start}) \times P(3|H) = 0.8 \times 0.4 = 0.32$$

路径: $\text{start} \rightarrow H$ 的转移概率是 0.32。

(2) 计算时间步 2 的维特比概率, 在 (1) 的基础计算。

计算时间步 $t=2$ 和状态 1 的概率:

$$V_2(1) = V_1(2) \times P(C|H) \times (1|C) = 0.048$$

路径: $\text{start} \rightarrow H \rightarrow C$ 的转移概率为 0.048。

计算时间步 $t=2$ 和状态 2 的概率:

$$V_2(2) = V_1(2) \times P(H|H) \times (1|H) = 0.0448$$

路径: $\text{start} \rightarrow H \rightarrow H$ 的转移概率为 0.0448。

(3) 计算时间步 3 的维特比概率, 在 (2) 的基础计算。

计算时间步 $t=3$ 和状态 1 的概率:

$$V_3(1) = V_2(1) \times P(C|C) \times (3|C) = 0.00288$$

路径：start→H→C→C 的转移概率为 0.00288。

计算时间步 $t=3$ 和状态 2 的概率：

$$V_3(2) = V_2(1) \times P(H|C) \times (3|H) = 0.048 \times 0.4 \times 0.4 = 0.00768$$

路径：start→H→C→H 的转移概率为 0.00768。

(4) 维特比反向追踪路径。

路径：start→H→C→H。

维特比反向追踪路径图如图 11-3 所示。

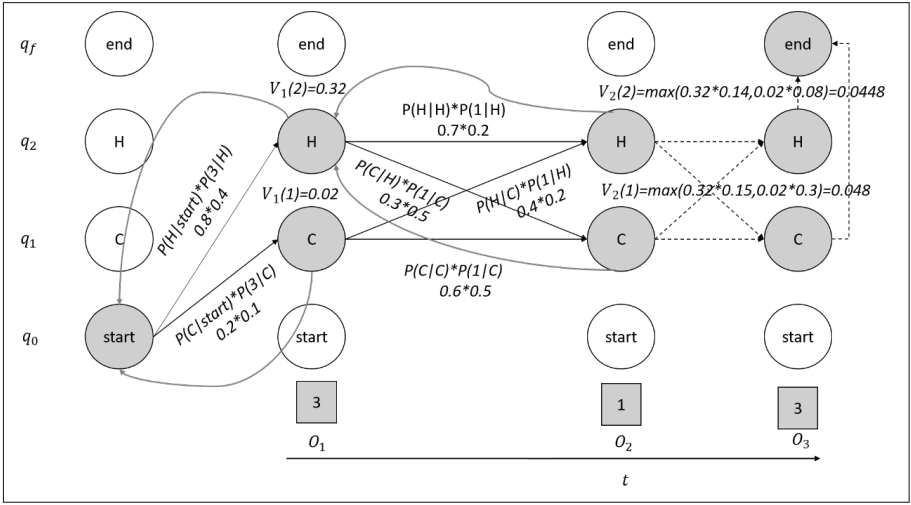


图 11-3 维特比反向追踪路径图

综上所述可知：利用 Viterbi 算法，我们可以通过小明夏季某天吃冰激凌的观察值（3 1 3）推断出天气为（热 冷 热）。其实这个结果是符合客观事实的，人们天热吃 3 根冰激凌，天冷吃 1 根。

第 12 章

条件随机场

本章导读：条件随机场常用于序列标注、数据分割等自然语言处理任务中，此外其在中文分词、中文人名识别和歧义消解等任务中也有应用。本章基于笔者在做语句识别序列标注过程中，对条件随机场产生的了解。内容主要源于自然语言处理、机器学习、统计学习方法和部分网上资料对 CRF 的相关介绍，最后由笔者进行大量研究整理后汇总成体系知识。本章首先介绍条件随机场的相关概念，然后结合实例，以期让读者深入理解条件随机场的应用。

12.1 条件随机场介绍

条件随机场 (Condition Random Fields), 简称 CRF

条件随机场概念：条件随机场就是对给定的输出标识序列 Y 和观察序列 X ，通过定义条件概率 $P(X|Y)$ ，而不是联合概率分布 $P(X,Y)$ 来描述模型。

概念解析：

标注一篇文章中的句子，即语句标注。使用标注方法 BIO 标注，B 代表句子的开始，I 代表句子中间，O 代表句子结束。观察序列 X 就是一个语料库（此处假设有一篇文章， x 代表文章中的每一句， X 是 x 的集合），标识序列 Y 是 BIO，即对应 X 序列的识别。从而我们可以根据条件概率 $P(\text{标注} | \text{句子})$ ，推测出正确的句子标注。显然，这里我们针对的是序列状态，即 CRF 是用来标注或划分序列结构数据的概率化结构模型的。CRF 在自然语言处理和图像处理领域都得到了广泛的应用，我们可以把它看作无向图模型或者马尔可夫随机场。

生产式模型与判别式模型

有监督机器学习方法可以分为生成方法和判别方法。

- ◎ 生成式模型：直接对联合分布进行建模，如混合高斯模型、隐马尔可夫模型、马尔可夫随机场等。
- ◎ 判别式模型：对条件分布进行建模，如条件随机场、支持向量机、逻辑回归等。

生成模型优缺点介绍

优点：

- ◎ 生成给出的是联合分布，不仅能够由联合分布计算条件分布（反之则不行），还可以给出其他信息。如果一个输入样本的边缘分布很小，那么可以认为学习出的这个模型可能不太适合对这个样本进行分类，分类效果可能会不好。
- ◎ 生成模型收敛速度比较快，即当样本数量较多时，生成模型能更快地收敛于真实模型。
- ◎ 生成模型能够应付存在隐变量的情况，比如混合高斯模型就是属于隐变量的生成方法。

缺点：

- ◎ 天下没有免费的午餐，联合分布能提供更多的信息，但也需要更多的样本与计算；尤其是为了更准确地估计类别条件分布，需要增加样本的数目。而且类别条件概率的许多信息是我们做分类所用不到的，因而如果只做分类任务，就浪费了计算资源。
- ◎ 实践中多数情况下判别模型效果更好。

判别模型优缺点介绍

优点：

- ◎ 与生成模型缺点对应，首先是节省计算资源。另外，需要的样本数量也少于生成模型。
- ◎ 准确率往往比生成模型高。
- ◎ 由于直接学习，而不需要求解类别条件概率，所以允许我们对输入进行抽象（比如降维、构造等），从而能够简化学习问题。

缺点：

- ◎ 没有生成模型的上述优点。

寻求最大概率即 $Y_{我B} = 0.5$ ，记录下“我”的标注路径，标注流程如图 12-2 所示。

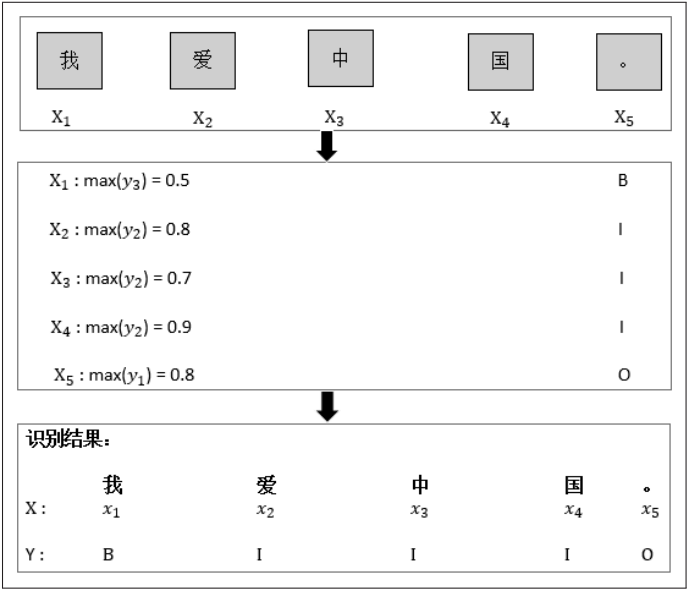


图 12-2 标注流程

如上便是对条件随机场与 Viterbi 算法的综合运用，其中 Viterbi 标注问题本质是隐马尔可夫模型三大问题之解码问题的算法模型。

12.2.2 CRF 的公式化表示

在给定的观察序列为 X 时，某个待定标记序列 Y 的概率可以定义为

$$\exp\left(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i)\right)$$

其中 $t_j(y_{i-1}, y_i, x, i)$ 是转移概率； $s_k(y_i, x, i)$ 是状态函数，表示观察序列 X 中 i 的位置的标记概率； λ_i 和 μ_k 分别是 t 和 s 的权重，需要从训练样本中估计出来。

实例解析：

我爱中国，其中 x_2 是爱字。 $t_j(y_1, y_2, \text{爱}, 2)$ 表示在观察状态 2 中我到爱的转移概率；其中 $j \in \{B, I, O\}$ ，可知 $S_K(y_2, \text{爱}, 2)$ 的生成概率或者发射概率的特征函数。用观察序列 $\{0,1\}$

二值特征 $b(x, i)$ 来表示训练样本中某些分布特征，其中采用 $\{0, 1\}$ 二值特征即符合条件标为 1，反之为 0，如图 12-3 所示。

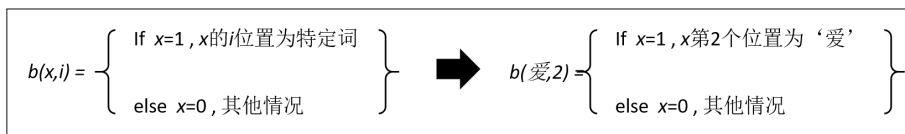


图 12-3 发射函数实例解析图

转移函数定义如图 12-4 所示。

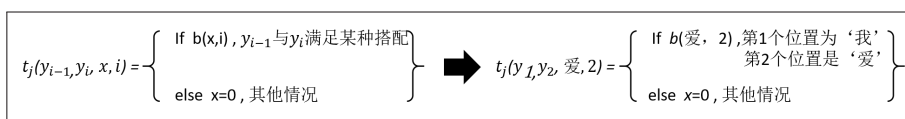


图 12-4 转移函数实例解析

为了便于描述，可以将状态函数写为以下形式：

$$s(y_i, x, i) = s(y_{i-1}, y_i, x, i)$$

特征函数：

$$F_j(Y, X) = \sum_{i=1}^n f_j(y_{i-1}, y_i, x, i)$$

其中每个局部 $f_j(y_{i-1}, y_i, x, i)$ 特征表示状态特征，由此条件随机场定义的条件概率如下：

$$P(Y|X, \lambda) = \frac{1}{Z(x)} \exp(\lambda_j \times F_j(Y, X))$$

其中分母为归一化因子：

$$Z(x) = \sum_i \exp(\lambda_j \times F_j(Y, X))$$

12.2.3 深度理解条件随机场

理论上标记序列描述一定条件的独立性， G 图结构是任意的，对序列进行建模可形成最简单、最普通的链式结构图。节点对应标记序列 X 中的元素，CRF 链式图如 12-5 所示。

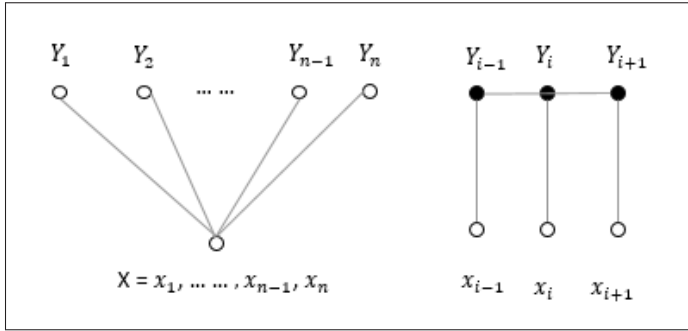


图 12-5 CRF 链式结构

如果图 12-5 中两种表示是一致的，其中图链式句子标注是图链式 2 的实例化，那么有的读者可能会问为什么上面的图是这种的而不是广义的图。其实这是因为观察序列 X 的元素之间并不存在图结构，没有做独立性假设，这点非常容易理解。诸如图 12-1 中“我爱中国”，其中 b 表示反射概率，而 t 是转移概率，线上的数值表示权值即概率值。如图 12-2 所示“我”的发射概率为 0.7，“我到爱”的转移概率为 0.5。通俗地讲，“我”和“爱”两个字是有关联的，而非独立的。

第 13 章

模型评估

本章导读：本章源于基于 HMM 模型序列标注的一个实验，在实验完成之后，迫切想知道采用的序列标注模型好坏，有哪些指标可以度量。于是就产生了对这一专题进度的学习总结，这样也便于其他人参考。本章依旧简明扼要地梳理出模型评估核心指标，以期达到实用的目的。本章首先介绍基于统计角度的模型评估，然后介绍模型评估的方法，最后对模型选择进行介绍。

13.1 从统计角度介绍模型概念

13.1.1 算法模型

概念简述

李航的《统计学习方法》：统计学习方法是由模型、策略和算法构成的，即统计学习方法的三要素构成，简化为方法 = 模型 + 策略 + 算法。

维基百科对数学模型的描述：数学模型是对所描述的对象用数学语言所做出的描述和处理。

百度百科对策略的描述：策略式学习是一项复杂的智能活动，学习过程与推理过程是紧密相连的。按照学习中使用推理的多少，机器学习所采用的策略大体上可分为 4 种——机械学习、通过传授学习、类比学习和通过事例学习。学习中所用的推理越多，系统的能力越强。

单从定义上看，不免让人一头雾水，究竟何为模型？还是没有明确的概念。下文将以数学描述及形式化阐述来解决这个问题。首先解决了什么是模型，才能进行模型好坏指标的评价，进而选择适合的学习模型。

模型：所有学习的条件概率分布或者决策函数。

模型的假设空间：包含所有可能的条件概率分布或者决策函数。

实例解析

假设决策函数是输入变量的线性函数，模型的假设空间就是这些线性函数构成的函数集合，假设空间中的模型一般为无穷多个【现实应用：假设解决序列词性标注的函数模型 M ，模型假设空间中由不同参数构成的 M 模型（不是很严谨，辅助理解）。】

形式化表示：假设空间用 F 表示，空间可以为决策函数的集合。

x 、 y 为在输入空间 X 和输出空间 Y 上的变量，这时 F 通常由一个参数向量决定的函数族表示。

参数向量 θ 取值于 n 维欧氏空间，称为参数空间。

假设空间也可以定义为条件概率集合：

其中 x 和 y 是定义在输入空间 X 和 Y 上的随机变量，这时 F 通常是一个参数向量决定的条件概率分布族。

参数向量 θ 取值于 n 维欧氏空间，称为参数空间。

注意：由决策函数表示的模型为非概率模型（如上述 F 函数），由条件概率表示的模型为概率模型（如上述 $y=f(x)$ 函数）。

13.1.2 模型评估和模型选择

训练误差和测试误差

好的模型的特征：对已知数据和未知数据都有很好的预测能力。学习方法评估标准：基于损失函数的模型训练误差和测试误差为指标。

假设学习到的模型 $Y = \hat{f}(x)$ ，训练误差是模型 $Y = \hat{f}(x)$ 关于训练数据集的平均损失：

$$R_{\text{emp}}(\hat{f}) = \frac{1}{N} \sum_{i=1}^N l(Y_i, \hat{f}(x_i))$$

其中 N 是训练样本容量。

测试误差是模型关于测试数据集的平均损失：

$$e_{\text{mst}}(\hat{f}) = \frac{1}{N'} \sum_{i=1}^{N'} l(Y_i, \hat{f}(x_i))$$

其中 N' 是训练样本容量。

实例

若损失函数是 0-1 损失的时候（0-1 损失参考具体相关知识），则测试误差就变成了常见的测试数据集上的误差率。 $e_{\text{mst}} = \frac{1}{N'} \sum_{i=1}^{N'} I(y_i \neq \hat{f}(x_i))$ ，这里的 I 是指示函数，即 $y_i \neq \hat{f}(x_i)$ 时为 1，否则为 0。

相应的，常见的测试数据集上的准确率是： $r_{\text{mst}} = \frac{1}{N'} \sum_{i=1}^{N'} I(y_i = \hat{f}(x_i))$ ，这里的 I 是指示函数，即 $y_i = \hat{f}(x_i)$ 时为 1，否则为 0。显然： $r_{\text{test}} + e_{\text{test}} = 1$ 。

实例解析

根据误差率和准确率可知，测试误差反映了学习方法对未知测试数据集的预测能力。测试误差小的方法具有很好的预测能力，能更有效地进行预测。通常将学习方法对未知数据的预测能力称为泛化能力。

由此我们可以应用到现实的 NLP 模型中。比如分类模型，当测试误差更小的时候，分类更加准确；在聚类模型中，当测试误差较小时，聚类效果更好，等等。因此，我们为了追求较小的测试误差，就要在训练上下功夫。但是，一味苛求训练效果好、训练误差小，以至于所选择的模型复杂度非常高，这样的低训练误差就一定能换回好的预测吗？事实上这就容易出现过拟合现象，如何避免过拟合选择更好的模型？下节继续。

13.1.3 过拟合与欠拟合的模型选择

何时进行模型选择

当假设空间含有不同复杂度（如：不同的参数个数）的模型时，就要面临模型选择问题，以期表达出的模型与真实的模型（参数个数）相同或相近。

过拟合：一味追求提高对训练数据的预测能力，所选择模型的复杂度往往比真实模型高，此现象就是过拟合。

过拟合指学习时选择的模型包含的参数过多，以至于模型对现有数据预测得好，但是对未知数据预测能力较差。模型选择准则是在避免过拟合的前提下尽可能地去提高模型预测的能力。

实例

给出一个训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ 和一个 M 次多项式的模型

$$f_M(x, w) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

完成下面 10 个数据点的拟合。

- ⊙ $x_i \in R$ 是输入序列集 x 的观察值。
- ⊙ $y_i \in R$ 是输出序列集 y 的观察值。
- ⊙ M 次项式是解决问题的模型。
- ⊙ w 为参数。

解决如上问题按照经验风险最小化策略求解参数即可，即数学表达：

$$L(w) = \frac{1}{2} \sum_{i=1}^N (f(x_i, w) - y_i)^2$$

损失函数为平方损失， $1/2$ 是便于计算。然后将模型公式和训练数据代入风险最小化公式： $L(w) = \frac{1}{2} \sum_{i=1}^N (\sum_{j=0}^M w_j x_i^j - y_i)^2$ ，最后采用最小二乘法求解（过程略）。

实例分析

如上, M 次多项式中, 给出 $M=0,1,3,9$ 的多项式函数的拟合情况。当 $M=0$ 时, 多项式变成了一个常数, 数据拟合表现很差; 当 $M=1$ 时, 多项式曲线为一条直线, 拟合依旧差; 当 $M=9$ 时, 多项式通过每一个点, 训练误差为 0。从训练数据拟合角度分析, $M=9$ 时效果最好, 但是训练数据本身存在很多噪音, 对未来数据预测能力差, 达不到预期的效果。这就是过拟合, 虽然训练数据表现好, 但是未知数据预测差。当 $M=3$ 时, 多项式曲线对训练数据的拟合效果比较好, 对未知数据的拟合也很好, 其模型也简单, 可以选择。总结: 模型选择时, 不仅仅考虑对已知数据的预测能力, 还有考虑对未知数据的预测能力。

训练误差和测试误差与模型复杂度的关系如图 13-1 所示。

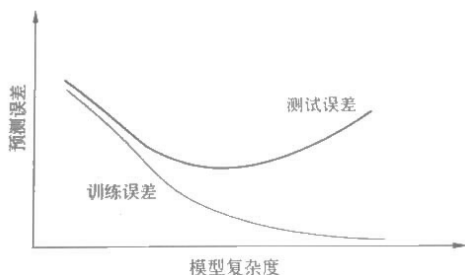


图 13-1 训练误差和测试误差与模型复杂度对比

可知, 当模型复杂度增大时, 训练误差会逐渐减小并趋近于 0, 而测试误差会先减小到最小值后又反向增大。当选择模型复杂度过大时, 过拟合问题就会出现。

补充

以下提供 NLP 序列句子识别标注实例, 用于更好理解本节 (以下实例以帮助读者理解为旨要, 假设可能不是特别严谨)。

训练数据集 $T=\{(\text{南 B}), (\text{海 I}), (\text{是 I}), (\text{中 I}), (\text{国 I}), (\text{领 I}), (\text{土 I}), (\text{。 O})\}$ 未知数据 $P=\{(\text{不 B}), (\text{容 I}), (\text{争 I}), (\text{议 I}), (\text{。 O})\}$ 采用 BIO 标注, B 代表句子开始, I 代表中间连续词, O 代表句子结束。假设采用模型 M 识别, M 次多项式的模型为

$$f_M(x, w) = \sum_{j=0}^M w_j x^j$$

完成下面句子识别。

结果分析

$M=0$ 时，模型为一个常数，效果很差，识别如下：

南	海	是	中	国	领	土	。		不	容	争	议	。
B	B	B	B	B	B	B	B		B	B	B	B	B

$M=1$ 时，模型为一条直线，效果很差，识别如下：

南	海	是	中	国	领	土	。		不	容	争	议	。
B	B	B	I	B	B	I	B		B	I	B	B	I

$M=3$ 时，模型曲线拟合基本合理，且未知数据预测较好，识别如下：

南	海	是	中	国	领	土	。		不	容	争	议	。
B	I	I	I	I	I	O	O		I	I	I	I	O

$M=9$ 时，模型为一条直线，效果很差，识别如下：

南	海	是	中	国	领	土	。		不	容	争	议	。
B	I	I	I	I	I	I	O		B	B	I	O	O

训练数据集：

南	海	是	中	国	领	土	。		不	容	争	议	。
B	I	I	I	I	I	I	O		B	I	I	I	O

实验可知：左侧为训练模型的数据，右侧为测试模型的数据。当 $M=0$ 时，训练误差和测试误差都很大；当 $M=1$ 时，训练误差和测试误差较大；当 $M=3$ 时，训练误差比 $M=9$ 的训练误差大，总体训练误差还好，但是预测误差却小于 $M=9$ 时的预测误差。综合比较，选择 $M=3$ 的模型效果会更好。

13.2 模型评估与选择

13.2.1 模型评估的概念

评估准确率的常用技术：保持和随机子抽样、*K*-折交叉验证、自助方法。

统计显著性检验：评估模型准确率。

ROC 曲线：接收者操作特征曲线比较分类器效果好坏。

13.2.2 模型评估的评测指标

混淆矩阵：正元组和负元组的合计。混淆矩阵如表 13-1 所示。

表 13-1 混淆矩阵表

预 测 的 样 本				
实际的样本		Yes	NO	合计
	Yes	TP	FN	P
	No	FP	TN	N
	合计	P	N	P+N

评估信息度量如表 13-2 所示（其中 *P*：正样本数；*N*：负样本数；*TP*：真正例；*TN*：真负例；*FP*：假正例；*FN*：假负例）。

表 13-2 评估信息度量表

度 量	公 式
准确率、识别率	$\frac{TP+TN}{P+N}$
错误率、误分类率	$\frac{FP+FN}{P+N}$
召回率、敏感度、真正例率	$\frac{TP}{P}$
特效性、真负例率	$\frac{TN}{N}$
精度	$\frac{TP}{TP+FP}$

(续表)

F ，分数精度和召回率的调和均值	$\frac{2 \times \text{精度} \times \text{召回率}}{\text{精度} + \text{召回率}}$
F_{β} ，其中 β 是非负数	$\frac{(1 + \beta^2) \text{精度} \times \text{召回率}}{\beta^2 \times \text{精度} + \text{召回率}}$

注意：学习器的准确率最好在检验集上估计，检验集由训练集模型未使用的含有标记的数据构成。

各参数描述如下。

TP （真正例/真阳性）：指被学习器正确学习的正元组，令 TP 为真正例的个数。

TN （真负例/真阴性）：指被学习器正确学习的负元组，令 TN 为真负例的个数。

FP （假正例/假阳性）：被错误的标记为正元组的负元组，令 FP 为假正例的个数。

FN （假负例/假阴性）：被错误的标记为负元组的正元组，令 FN 为假负例的个数。

高准确率的学习模型：大部分元组应该在混合矩阵的对角线上，而其他为 0 或者接近 0，即 FP 和 FN 为 0。本质上是一个对角矩阵时准确率最高。

准确率：正确识别的元组所占的比例。又叫作识别率，公式如下：

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

13.2.3 以词性标注为例分析模型评估

词性标注为例分析模型评估详细信息如表 13-3 所示。

表 13-3 词性标注为例分析模型评估

标注正确与否	true-words	False-words	countNum	Accuracy (%)
true-words	6954	46	7000	99.34
false-word	412	2588	3000	86.27
Count-Num	7266	2634	10000	95.42

错误率：错误识别元组所占的比例，又叫误识别率。公式如下： $\text{errorrate} = \frac{FP + FN}{P + N}$ 或者 $1 - \text{Accuracy}(M)$ 。

检验时，应采用检验集未加入训练集的数据。当采用训练集估计模型时，会再带入误差，这种称为乐观估计。准确率可以度量正确标注的百分比，但是不能正确度量错误率。诸如样本不平衡时，即负样本稀疏的时候。比如：欺诈、癌症等，这种情况下应使用灵敏性特效性度量。

灵敏度又叫真正识别率：正确识别的正元组的百分比。公式如下：

$$\text{Sensitivity} = \frac{TP}{P}$$

特效性又叫真负例率：正确识别的正元组的百分比。公式如下：

$$\text{Specificity} = \frac{TN}{N}$$

准确率的灵敏度和特效性的函数关系：

$$\text{Accuracy} = \text{Sensitivity} \frac{P}{P+N} + \text{Specificity} \frac{N}{P+N}$$

精度：精确性的度量即标记为正元组，实际为正元组的百分比。公式如下：

$$\text{Precision} = \frac{TP}{TP+FP}$$

召回率：完全性的度量即正元组标记为正的百分比。公式如下：

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{P}$$

精度和召回率之间趋向于逆关系，有可能出现一个指标降低另一个指标提升的情况。为了中和两个指标而引用 F 度量值。

F 度量（又叫 F 分数）：用精度和召回率的方法把它们组合到一个度量中。公式如下：

$$F = \frac{2 \times \text{精度} \times \text{召回率}}{\text{精度} + \text{召回率}}$$

$$F_{\beta} = \frac{(1 + \beta^2) \text{精度} \times \text{召回率}}{\beta^2 \times \text{精度} + \text{召回率}}$$

比较： F 度量是精度和召回率的调和均值，赋予精度和召回率相等的权重， F_{β} 度量是精度

和召回率加权重度量，它赋予召回率权重是精度的 β 倍。诸如中文词汇中常用词的权重比生僻词的权重大，而这也符合实际应用。

注意：当元组属于多个类时，不适合使用准确率。当数据均衡分布即正负元组基本相当时，准确率效果最好，而召回率、特效性、精度、 F 和 F_β 更适合于样本分布不均的情况。

13.2.4 模型评估的几种方法

模型评估包括多种方法，常见的评估流程如图 13-2 所示。

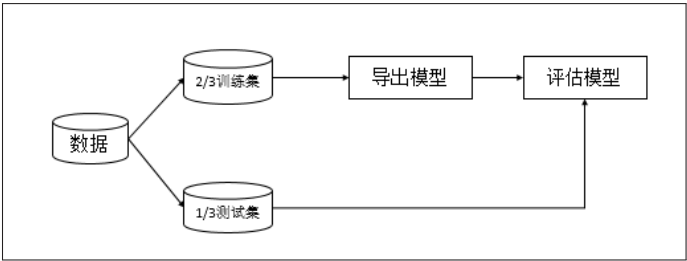


图 13-2 模型评估流程

（1）随机二次抽样评估准确率：保持方法的一种变形，将保持方法重复 K 次，总准确率估计是每次迭代准确率的平均值。

（2） K -折交叉验证评估准确率（建议 10-折）。

K -折交叉验证：将初始的数据随机分为大小大致相同的 K 份，训练和检验进行 K 次，如第 1 次迭代第一份数据作为检查集，则其余 $K-1$ 份作为训练集，进行第 2 次迭代。第二份数据作为检验集，则其余 $K-1$ 份作为训练集，以此类推，直到第 K 份数据作为检验集为止。此方法每份样本用于训练的次数一致且每份样本只作为一次检验集。准确率是 K 次迭代正确元组总数除以初始数据元组总数。一般建议采用 10-折交叉验证估计准确率，因为它的偏移和方差较低。

（3）自助法评估准确率。

自助法是将样本有放回的均匀抽样，常用 632 自助法。即 63.2% 的原数据将出现在自助样本中，而其余 38.8% 的元数据形成检验集。

13.3 ROC 曲线比较学习器模型

成本效益（风险增益）：如错误地预测癌症患者没有患病比将没有患病的病人归类癌症的代价大等事件，据此给予不同的权重。

ROC 曲线又叫接受者操作特征曲线，比较两个学习器模型的可视化工具，横坐标参数为假正例率，纵坐标参数是真正例率。以此汇聚成的曲线，越靠近对角线（随机猜测线），模型越不好。

真正例率（召回率）：

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

假正例率：

$$TFR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

实例解析：以 10 个检验元组的概率分类器为例绘制 ROC 曲线。数据如表 13-4 所示。

表 13-4 检验元素数据表

元 祖	类	概 率	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	1	0	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0

由以上数据绘制的 ROC 曲线如图 13-3 所示。

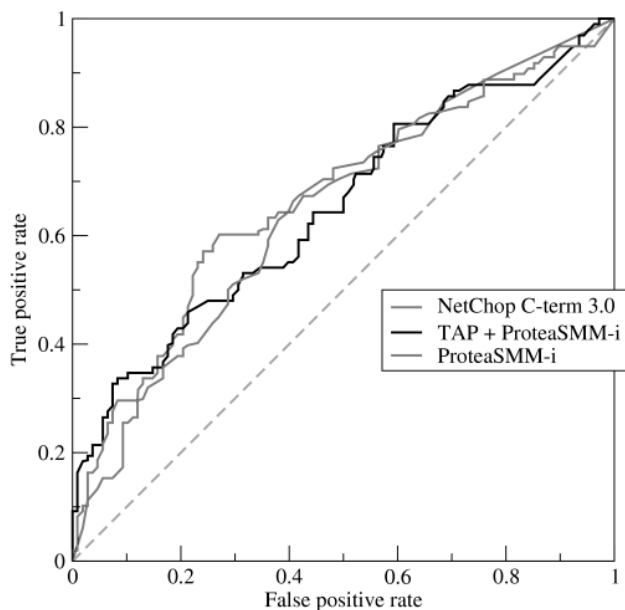


图 13-3 ROC 曲线图

ROC 曲线术语解释。

- ⊙ 真阳性 (TP, true positive): 正确的肯定。
- ⊙ 伪阳性 (FP, false positive): 错误的肯定。
- ⊙ NetChop C-term 3.0: 一种模型方法。
- ⊙ TAP+ProteaSMM-i: 一种模型方法。
- ⊙ ProteaSMM-i: 一种模型方法。

由此可知，对角线为随机猜测线，模型的 ROC 曲线越靠近对角线，模型的准确率越低。如果是很好的模型，真正比例比较多，则曲线应陡峭地从 0 开始上升，后来遇到真正比例越来越少，假正比例元组越来越多，曲线变得更加水平。完全正确的模型面积为 1。

第 14 章

命名实体识别

本章导读：命名实体识别在自然语言处理中占据着非常重要的地位，也是不可逾越的学术问题。命名实体识别的学术理论和研究方法众多，本章侧重整体介绍。首先阐述命名实体识别的背景知识和研究概况；其次介绍中文命名实体识别的特点与难点，辅以案例加深理解；然后对命名实体识别当前研究方法和核心技术进行详细介绍；最后，展望其在未来人工智能方面的发展前景。

14.1 命名实体识别概述

背景介绍

命名实体识别（Named Entity Recognizer, NER）在第六届信息理解会议（MUC-6）上被提出后，人们的视野便聚焦在信息抽取（Information Extraction）问题上（即如何从半结构化、非结构化文本中抽取结构化信息）。此外，命名实体识别也是信息抽取、本体构建、问答系统等自然语言处理任务的基础工作。

命名实体识别旨在识别文本中的角色实体，可以分解成两个子任务即实体边界确定和实体类别划分。当前命名实体识别研究方法主要有：（1）规则和词典相结合的方法，一般适用于精确度较高的情况。但是其存在系统建设周期长、移植性差等问题。（2）统计机器学习的方法，诸如隐马尔可夫模型、条件随机场、最大熵等。本文采用命名实体识别技术主要解决三个问题：一是如何准确分词，因为中文不同于英文（由空格和单词构成），解决中文字词的划分尤为重要；二是分词后如何进行序列标注，标注集合的规约；三是如何准备识别实体边界以及实体名。诸如语料“拖雷与郭靖想起在襄阳城下险些拼个你死我活，都是暗叫惭

愧”。这句话可以看作一个由词构成的序列，那么分词效果尤为重要。下面看看部分分词工具的处理结果。

(1) Stanford NLP 中文分词结果。

【拖雷/与/郭靖/想起/在/襄阳/城下/险些/拼个/你死我活/，/都/是/暗/叫/惭愧/。/】

(2) 结巴分词结果。

【拖雷/与/郭靖/想起/在/襄阳/城下/险些/拼个/你死我活/，/都/是/暗/叫/惭愧/。/】

(3) NLPPIR 汉语分词结果。

【拖/雷与郭/靖想起/在/襄阳/城下/险些/拼/个你死我活/，/都/是/暗/叫/惭愧/。/】

上述分词结果表明，(1)与(2)中“暗叫惭愧”的分词结果不一致；(3)与(1)(2)最大区别就是“拖雷和郭靖”的分词结果不一致。显然(3)出现的问题更为严重，出现错误的人名识别，由此看出分词的好坏直接影响命名实体的识别结果。此外，歧义词、未登录词都是命名实体识别中亟待解决的问题。本节采用基于角色标注的方法对分词结果进行序列标注。角色标注结果如下：

[拖雷/nr,与/cc,郭靖/nr,想起/v,在/p,襄阳/nns,城/n,下/f,险些/d,拼/n,个/q,
你死我活/al,/w,
都/d,是/vshi,暗/a,叫/vi,惭愧/a,。/w]

其中 nr 为人名，ns 为地名。完成实体识别后，采用 BMES 标注方式进行处理。其中 B 表示 Begin 即识别出边界，M 表示 Middle 即识别出实体中间名，E 表示 End 即实体名识别介绍，S 表示 Single 表示独立成词，最后过滤掉独立词即可。其中人名可以细化为译名、日本名等；地名可以细化为国家、省、市县等。综上所述，采用自然语言处理技术手段对文本语料的命名实体识别具有深层次的意义。

国内外研究现状

本节采用自然语言处理技术对中文命名实体识别方法进行研究。命名实体识别在自然语言处理中占据很重要的位置，命名实体识别的评测系统也备受国内外会议重视。主要包括如下会议：

(1) 信息理解研讨。

(2) 文本检索会议。

- (3) 多语种实体评价任务会议。
- (4) 国际中文处理评测。
- (5) 自动内容抽取评测会议。
- (6) ACL 会议。
- (7) 自然语言学习会议。
- (8) 863 评测会议。

国内外关于命名实体识别的主要研究机构和相关工作如下。

(1) 国外研究机构主要是对英语等语言的实体识别，代表机构包括斯坦福研究所人工智能中心、英特尔研究中心、微软研究院、雅虎研究中心、日本东京大学等。

(2) 国内主要解决中文命名实体识别，代表机构包括中科院计算所、微软中国研究院、哈尔滨工业大学自然语言处理实验室、北京语言大学语言信息处理研究所、北京理工大学自然语言处理研究室和复旦大学自然语言处理研究室等。

随着第六届信息理解会议提出的信息抽取相关研究，命名实体识别作为其下的一个子任务而备受关注。名称的自动抽取又称为“命名实体识别”，其任务就是识别出待处理文本中的三大类和七小类命名实体。三大类命名实体包括实体类（人名、地名机构名）、时间类（日期、时间和持续时间）和数字类（货币、度量衡、百分比和基数）；七小类命名实体包括人名、地名、机构名、时间、日期、货币和百分比。其中时间、百分比、日期、货币的构成较为规律，识别起来难度不大；而人名、地名、机构名用字较为灵活，识别难度较大。所以，命名实体识别通常指人名、地名、机构名的识别。在命名实体识别中，中文命名实体存在形式不一、语言环境复杂等现象，其研究方法也呈现出多样性的特点。总体分为三类，分别是基于规则模型命名实体识别、基于统计模型的命名实体识别和基于规则结合统计的命名实体识别。其中基于统计模型的学习方式又划分如下四类。

- (1) 有监督的学习方法：利用人工标注大部分数据集进行模型训练学习。
- (2) 半监督的学习方法：利用人工标注很少的数据集（种子数据）自举学习。
- (3) 无监督的学习方法：不再进行人工标注，而是通过上下文聚类学习。
- (4) 混合方法：几种模型相互结合或利用统计方法和人工总结的知识库。

目前采用自然语言处理工具对命名实体识别具有良好的效果，特别是在结合专业知识领域的情况下。自然语言处理工具主要包括：

- (1) 精准自然语言解析器（SyntaxNet）。

- (2) 中文自然语言处理工具包 (FudanNLP)。
- (3) Java 自然语言处理 (LingPipe)。
- (4) 自然语言处理工具 (OpenNLP)。
- (5) 自然语言工具包 (NLTK)。
- (6) 自然语言工具包 (CRF++)。
- (7) 单词转换成向量形式 (word2vec)。
- (8) 自然语言文本处理库 (spaCy)。

14.2 命名实体识别的特点与难点

由于中文命名实体数量较大，我们还很难构建大而全的名字库、地址库等。还有较长的少数民族人名和译名（比如：扎克伯格、麦当劳、肯德基），没有统一的构词规范。并且人名、地名和组织机构名之间有着交叉和包含现象，组织名称中也常常包含大量的人名、地名、数字。想要正确标注这些实体类型，需要基于上下文内容。

对命名实体的边界识别和类型确定尚没有统一标准。命名实体识别过程常与中文识别等结合，通常分词、语法分析系统的结果也影响命名实体识别的有效性。根据领域和知识本体的需要实体还可以细分，诸如体育名、汽车名、商标名等。随着电商的发展，对品牌名、产品名等商品类的实体识别也有需求。针对这些新的实体类型，最大的瓶颈就是缺少标准的训练数据。歧义现象和不同实体内部特征都是急需解决的问题。

14.3 命名实体识别方法

基于统计的命名实体识别方法

进入 21 世纪之后，基于海量数据的统计方法逐渐成为自然语言处理的主流，同时自然语言处理的各个方面也得到机器学习方法的支持。基于统计的命名实体识别方法具有可移植性好、语言依赖性小、处理速度快等优点。该方法是利用序列标注实现的，本质上就是命名实体识别问题向序列标注问题的转换，其主要处理步骤如下。

- (1) 统计学习策略：基于统计方法的命名实体识别，合适的机器学习方法是很重要的。常用的机器学习方法包括隐马尔可夫模型（HMM）、条件随机场模型（CRF）、最大熵模型（MEM）等。
- (2) 特征选择：特征的选择直接影响命名实体识别的好坏，一般特征包括一些先验知识（如词性、词典、词后缀等）。序列标注：通过基于机器学习策略的文本序列标注来处理训练集和测试集。
- (3) 模型训练：通过对训练集的训练优化算法模型。
- (4) 模型评测：训练出来的算法模型，通过测试集进行测试，反复实验以得到理想的模型结果，并将算法模型应用到命名实体识别中。

以上介绍的机器学习模型中，隐马尔可夫模型是非常重要的统计模型，其本质上是一种马尔可夫随机过程的概率函数。研究者们将隐马尔可夫模型应用到命名实体识别、语音识别、序列标注等领域中都取得了不错的成绩。最大熵模型的原理就是在学习概率模型时，认为概率分布模型中最大熵的模型是最好的，并常以此作为约束条件来确定概率模型集合。最大熵模型也可表示为在满足约束条件的模型集合中选择最大熵的模型。由于马尔可夫模型具备生成式模型的缺点，而最大熵模型受约束条件限制。结合两种模型的最大熵马尔可夫模型是其一种延伸，这种模型具备以上两种模型的优点，成功克服各自模型的缺陷，但是其存在标记偏置的缺陷。

最大熵模型最大的问题就是标记偏置。针对这一问题，Lafferty 等学者提出了条件随机场（CRF）模型。CRF 对给出观测序列的条件下，针对全序列进行联合概率的指数模型，这种方法很好地解决了偏置缺陷的问题。条件随机场不仅适用于命名实体识别，而且在英文 POS 标注、英文词短语识别等方面都取得了不错效果。

综上所述，可以将基于机器学习的命名实体识别方法划分为有监督的学习方法、半监督学习的方法、无监督的学习方法和多种模型混合的方法，具体方法归纳如表 14-1 所示。

表 14-1 检验元素数据表

类 型	实体识别模型或方法	代 表 工 作
有监督学习方法	隐马尔可夫模型	Liu et al.(2005);Zhang et al(2003a)
	最大熵模型	Tsai et l.(2004);Borthwock(1999)
	支持向量机	Yi et al(2004);Asahara and Matsumoto(2003)
	条件随机场	Finkel et al.(2005);McCallum and Li(2003)
	决策树	Isozaki(2001);Paliouras et al.(2000)

(续表)

半监督学习方法	利用小数据集自动学习	Singh et al.(2010);Nadeau.(2007)
无监督学习方法	利用词汇资源上下文聚类	Etzioni et al.(2005);Shinyama(2004)
混合方法	集中模型混合	Liu et al(2011b);Wu et al(2003,2005)

基于规则和统计的命名实体识别方法

在实际应用中，仅仅基于规则或者统计的方法并不能取得期望的结果。规则和统计结合的方法具备两者的优点，也是实际应用最多的方法。通常我们会选择基于统计的方法对训练语料进行模型构建，从而拟合出一个最佳模型。在这个模型的基础上，再加以人工知识库即基于规则的方法辅助命名实体识别，以防止出现过拟合的现象。这样做的好处是一方面降低了命名实体识别模型的语料库规模，提高识别准确率和召回率；另一方面可以保证识别模型的算法效率。事实证明这种方法确切可行且取得了不错的效果。

Seon 等人提出的基于最大熵模型和神经网络模型及规则的方法进行命名实体识别取得了很好的效果，系统可以对中文人名、地名、机构名进行识别，其效果如表 14-2 所示。

表 14-2 人名、地名、机构名实体识别

	人名 (%)	地名 (%)	机构名 (%)
准确率	91.04	82.08	71.01
召回率	95.31	87.13	64.02

张华平等应用隐马尔可夫模型结合词汇表方法进行汉语人名识别，其方法原理是首先对人名进行分类，然后标注语料集进行模型训练，接着采用维特比算法进行人名标注，最后用最大模式匹配进行人名识别。

14.4 中文命名实体识别的核心技术

命名实体角色标注

由于中文实体构成的特点和难点造成实体识别比较困难，针对这些困难本节采用基于角色标注的中文实体识别方法进行处理。分好的词细化为人名内部组成关系、上下文关系、

无关键词，总结为中国人名构成角色表、地名构成角色表、机构名角色表。中文命名实体的识别是将熟语料中的序列标注转换成实体角色标注的过程。其数学描述为：给定一个文本序列串（ $X = x_1, x_2, \dots, x_n$ ，其中表示文本特征项），其目的是构造一个序列标注机器 p ，使其为文本串 x 标注合适的标签串 $y = p(x)$ 。其中 $y = y_1, y_2, \dots, y_n$ ，其中 y 属于人名构成角色表中的标记，然后从所有可能标注的序列中选择最大概率，即：

$$\hat{y} = \arg \max_y P(y|x) \quad (14.1)$$

根据贝叶斯公式，可知

$$\hat{y} = \arg \max_y \frac{p(x|y)p(y)}{p(x)} \quad (14.2)$$

由于 $p(x)$ 是一个常数，由（14.1）（14.2）式可知：

$$\hat{y} = \arg \max_y p(x|y)p(y) \quad (14.3)$$

根据 2.3 节 HMM 相关知识计算 $p(y)p(x|y)$ 可知：

$$\hat{y} = \arg \max_y \prod_{i=1}^n p(x_i|y_i)p(y_i|y_{i-1}) \quad (14.4)$$

（14.4）式可化简为对数形式，则有：

$$\hat{y} = \arg \max_y \left(\sum_{i=1}^n [\ln p(x_i|y_i) + \ln p(y_i|y_{i-1})] \right) \quad (14.5)$$

本节训练数据来源于 2014 年某日报内容，其采用北京大学计算语言学所的词类标注集得到结果如表 14-3 所示。

表 14-3 自动分词标注

自动分词的句子标注
王/nr 小二/nr 力/n 倡 vg “/w 工匠/nnd 精神/n ” /w ,/w
入选/v 2016/m 年/qt 十/m 大/a 流行语/n

结果表明：分词模型根据训练将姓氏和名字进行分割标注采用 nr。针对这种人名的标注不利于姓名的识别，如果姓氏和名字采用不同的标注，则利用上下文识别效果会更好。因此，采用中科院的词类标注集进行实验，其标注形式如表 14-4 所示。

表 14-4 NLPPIR 词类标注

中科院词类句子标注
王/nf 小二/nl]nr 力/n 倡 vg “/we 工匠/nnd 精神/n ” /we ,/we 入选/v 2016/m 年/t 十/m 大/a 流行语/n

分析发现中科院词类标注中对姓和名标注采用 nf、nl 单独标注，nr 标注为人名。这样做的好处是可以通过词位上下文分析，避免出现上文与姓成词或者下文与名成词的现象。

本节针对以上两种情况进行改进，首先对语料采用中科院词类标注集进行处理，再对熟语料进行逐句的读入，判断该句中的词是否为人名相关词即 nf、nl、nr 词。若非人名相关的词性则均标注为独立词即 A，若与人名相关则继续判断其上文 p 与姓 nf 是否成词，不成词标注为 K 即上文，成词标注为 U；继续判断名与下文是否成词，若不成词则标注为 L 即下文，成词则标注为 V。再将姓氏标注为 B，姓与单名标注为 Y，姓与双名标注为 X，双名中间成词标注为 Z。诸如此类讨论人名的各种情况，最后过滤掉独立词，将人名项目的标注进行词频统计，以及不同标注直接转化进行概率统计。再用 Viterbi 算法进行改进后的标注如表 14-5 所示。

表 14-5 基于实体的角色标注

人名实体的句子角色标注
王/B 小/C 二/D 力/A 倡 A “/A 工匠/A 精神/A ” /A ,/A 入选/A 2016/A 年/A 十/A 大/A 流行语/A

关于中文实体标注的问题实际上就是（14.5）式的求解问题了，即转化为隐马尔可夫模型的解码问题，采用 Viterbi 算法便可解决，Viterbi 算法具体介绍见 2.3 节内容。以人名实体识别为例，人名识别还存在以下问题，标记为 U 的词，“公司现任法人为何三立”，这里的人名上文和姓成词了。标记为 V 的词，如“王小二级别过低，不能申请教授”，这里的人名末尾词与下文成词现象。

针对这种情况，解决方式是再次进行细粒度分割，处理成 KB（人名上文 + 姓氏）、DL（双名末尾词 + 人名下文）或者 EL（单名 + 人名下文）；再根据模式集合 {BBCD, BBE,

BBZ, BCD, BE, BG, BXD, BZ, CD, FB, Y, XD} 进行最大模式串的匹配。例如：“王小二在技术社区开了一个博客。”，分词结果是“王/小/二/在/技术社区/开/了/一个/博客/。”采用本文基于角色标注结果是：“BCDLAAAAAA”，可以识别出人名“王小二”，这为下一步地名、机构名的识别减少了很多干扰作用。人名识别算法如表 14-6 所示。

表 14-6 人名识别算法

人名识别算法
输入：任意分词序列（句子） 输出：汉语人名角色 初始化：语料表 src=[[]]，角色表 trg=[[]] 说明：src 是用来存储角色标注后，最大角色概率的数组，trg 是用来记录不同人名角色的数组，便于统计人名词频和角色转化概率 第一步：从句子中任取一个词，执行如下循环，直到句子结束。对每个句子序列进行分词，并采用 Viterbi 算法进行角色标注，求解最大角色概率 将角色 U 即姓与上文成词的 kf 进行拆分： ● 若 f 为姓氏，拆分为 KB（上文和姓） ● 若 f 为双名首字，拆分为 KC（上文和双名首字） ● 若 f 为双名尾字，拆分为 KF（上文和双名尾字） 将角色 V 即名与下文成词的 tn 进行拆分： ● 若 t 为双名尾字，拆分为 DL（双名尾字和下文） ● 若 t 为单名，拆分为 EL（单名和下文） 第二步：对识别的人名进行规则检验，将不符合汉语人名标准的剔除，其他进入人名词典库

上述改进算法的人名识别结果如下所示。

```

龚蕾 nr 3
龚警官 nr 1
... ..
龚铭 nr 2
龚锐云 nr 1

```

通过实体角色标注，再利用 Viterbi 算法求得类别实体名。诸如人名实体识别为“孙中山”，针对南京市的“中山路”该如何识别呢？这就属于包含简单实体的复合实体识别问

题。假设包含简单实体“孙中山”为 w_i ，相应的角色标注为 t_i ，由于 w_i 不在词典中，所以 $p(w_i|t_i)$ 就无法求解了。此时，我们引入基于角色是实体的生成模型，该模型与隐马尔可夫模型映射，其目的就是求解复合命名实体的生成概率。采用 HMM 过程可以得到 14.6 式（其中 w 是待识别的命名实体， c 是类别）：

$$p(w|c) = \prod_{j=1}^n p(w_{i+j}|c_{i+j})p(c_{i+j}|c_{i+j-1}) \tag{14.6}$$

经过语料的角色生成模型之后，问题就简化为对统计角色词频和角色转移概率的求解了。其原理是利用中文切分的熟语料，再结合少量代码对熟语料进行标注。2014 年某日报句子如表 14-7 所示。

表 14-7 2014 年某日报语料

某日报语摘
李某某/nr 来到/v 南京/ns 中山路/ns 某/rz 物业公司/nis 做/v 保安/b

通过少量自动角色标注代码，逐步处理得到如图 14-1 所示的角色标注结果。

```
原始语料:
[未##人/nr, 来自/v, 南京/ns, 的/udel, 中山路/ns, 某/rs, 物业公司/nis, 做/v, 保安/b]
添加首尾:
[始##始/S, 未##人/nr, 来自/v, 南京/ns, 的/udel, 中山路/ns, 某/rs, 物业公司/nis, 做/v, 保安/b,
未##未/A]
标注上文:
[始##始/S, 未##人/nr, 来自/A, 南京/ns, 的/udel, 中山路/ns, 某/rs, 物业公司/nis, 做/v, 保安/b,
未##未/A]
标注下文:
[始##始/S, 未##人/nr, 来自/A, 南京/ns, 的/udel, 中山路/ns, 某/B, 物业公司/nis, 做/v, 保安/b,
未##未/A]
标注中间:
[始##始/S, 未##人/nr, 来自/A, 南京/ns, 的/X, 中山路/ns, 某/B, 物业公司/nis, 做/v, 保安/b, 未##
未/A]
处理整个:
[始##始/S, 未##人/A, 来自/A, 南京/G, 的/X, 中山路/C, 某/B, 物业公司/A, 做/A, 保安/A, 未##未/A]
```

图 14-1 角色标注结果

完成如上操作之后的工作就是统计角色词频和转移矩阵了，这个很容易处理。利用 HMM-Viterbi 算法即可得到复合实体的最大生成概率，然后采用模式匹配方法进行实体识别。将识别出来的实体作为词典数据传输到下一层 HMM 实体中，这不仅可以很好地解决复合实体识别问题，还可以优化模型参数。值得强调的是，本节基于地名角色标注改进了原有

的方法,提出 C (中国地名首部)、D (中国地名中间)、E (中国地名未部),在地名模式识别中 (CDE 地名 + 三字后缀) 可以将地名识别改进到六个汉字的长度,其后便是对命名实体的自动抽取工作。

命名实体自动抽取

层叠隐马尔可夫模型是将中文分词、切分歧义、人名、地名、结构名和词性标注融合在一起的算法模型。由于中文黏稠性的特点,困扰分词结果最大的问题就是未登录词,本质上就是中文命名实体的识别。主要有人名、地名、机构名、时间、数量、单位等。其中人名、地名、机构名最难识别,也是对中文分词影响最大的因素。因此命名实体一体化抽取的整个过程如下:

- (1) 首先对 2014 年某日报语料库进行词频统计,构建出核心分词词典,此时核心词典仅含有极其少量的高频人名、地名、机构名等实体。
- (2) 利用第一步的核心词典,对原始字符串进行粗粒度切分,并利用 N 元最短路径进行歧义切分。利用角色语料进行训练,得到人名角色词典和角色标记间的转移概率,详细参见 4.2 节。
- (3) 利用第二步识别出来的人名粗分结果,基于角色的地名进行识别,得到地名词典。为了识别含有人名的复合地名,人名识别的 HMM 将人名类作为输入参数,进而识别出地名和相应的转移概率。
- (4) 跟人名、地名类似,将人名识别 HMM 和地名识别的 HMM 作为参数,通过与语料库中标注好的机构名进行训练,得到机构名的角色表。
- (5) 将识别出的人名、地名、机构名、歧义切分、登录分词融合一体,进行全局最优概率计算。
- (6) 对实验结果进行测试和模型性能指标评估。

层叠隐马尔可夫模型框架设计过程如下:

- ◎ 构建核心词典本质上就是对词频统计和词之间概率转移的统计;
 - ◎ 进行 N 元最短路径歧义切分,本节采用的是 N 元最短路径的策略,即在初始阶段保全切分概率 $P(X)$ 最大的 N 个结果,并以此作为候选集合,再综合运用最少切分和全切分的方法。具体实验过程如下。
- (1) 对原始语料进行切分标注后如表 14-8 所示。

表 14-8 原始语料切分标注结果

某日报语摘
[参与/v, [北京/ns 电影/n 学院/nis]/nt, 和/cc, [美国/nsf 辛普森/nr 公司/nis]/nt, 的/ude1, 活动/vn, , /w, 由/p, [交通/n 银行/nis 北京/ns 分行/n]/nt, 与/cc, 麦当劳/nt, 赞助/v, , /w, [巴/b 政府/nis]/nt, 和/cc, 指导/vn, , /w, [中央/n 电视台/nis]/nt, 报道/v]

(2) 词性标注向基于角色标注的转换结果如表 14-9 所示:

表 14-9 基于角色标注的结果

基于角色标注过程
添加首尾: [始 ## 始/S, 参与/v, [北京/ns 电影/n 学院/nis]/nt, 和/cc, [美国/nsf 辛普森/nr 公司/nis]/nt, 的/ude1, 活动/vn, , /w, 由/p, [交通/n 银行/nis 北京/ns 分行/n]/nt, 与/cc, 麦当劳 /nt, 赞助/v, , /w, [巴/b 政府/nis]/nt, 和/cc, 指导/vn, , /w, [中央/n 电视台/nis]/nt, 报道/v, 末 ## 末/Z]
标注上文: [始 ## 始/S, 参与/A, [北京/ns 电影/n 学院/nis]/nt, 和/A, [美国/nsf 辛普森/nr 公司/nis]/nt, 的/ude1, 活动/vn, , /w, 由/A, [交通/n 银行/nis 北京/ns 分行/n]/nt, 与/A, 麦当劳 /nt, 赞助/v, , /A, [巴/b 政府/nis]/nt, 和/A, 指导/vn, , /A, [中央/n 电视台/nis]/nt, 报道/v, 末 ## 末/Z]
标注下文: [始 ## 始/S, 参与/A, [北京/ns 电影/n 学院/nis]/nt, 和/B, [美国/nsf 辛普森/nr 公司/nis]/nt, 的/B, 活动/vn, , /w, 由/A, [交通/n 银行/nis 北京/ns 分行/n]/nt, 与/B, 麦当劳/nt, 赞助/B, , /A, [巴/b 政府/nis]/nt, 和/B, 指导/B, , /A, [中央/n 电视台/nis]/nt, 报道/B, 末 ## 末/Z]
标注中间: [始 ## 始/S, 参与/A, [北京/ns 电影/n 学院/nis]/nt, 和/X, [美国/nsf 辛普森/nr 公司/nis]/nt, 的/B, 活动/vn, , /w, 由/A, [交通/n 银行/nis 北京/ns 分行/n]/nt, 与/X, 麦当劳/nt, 赞助/B, , /A, [巴/b 政府/nis]/nt, 和/X, 指导/B, , /A, [中央/n 电视台/nis]/nt, 报道/B, 末 ## 末/Z]
整个过程:

(续表)

[始##始/S,参与/A,未##地/G,电影/C,学院/D,和/X,未##地/G,未##人/F,公司/D,的/B,活动/Z,,/Z,由/A,交通/C,银行/D,未##地/G,分行/D,与/X,麦当劳/K,赞助/B,,/A,巴/J,政府/D,和/X,未##团/K,指导/B,,/A,中央/C,电视台/D,报道/B,未##末/Z]

- (3) 统计词频,在对所有熟语料句子执行自动标注后,即可统计每一个非 Z 词语的各角色词频。然后统计文本序列的发射矩阵和转移矩阵,最终代入公式求最大概率即可。其中发射矩阵如下所示。

```
公司 D 4621 A 24 B 24
公司总部 B 1
公司治理 B 2
公司股票 B 1
公告 B 15 X 1
公告栏 B 2
公园 C 9
公安 C 728 A 19 B 5
公安分局 A 18
公安厅 D 226 A 4
公安处 D 183
公安学 B 4
公安局 D 2251 A 5
公安机关 B 10
.....
```

转移矩阵指的是从一个角色标签转移到另一个角色的频次,利用它和角色词频可以计算出 HMM 中的初始概率、转移概率、发射概率,进而完成求解。这里对某日报 2014 切分语料训练出如下转移矩阵。

```
.,A,B,C,D,F,G,I,J,K,L,M,P,S,W,X,Z
A,0,0,19945,883,2013,58781,3290,1582,19254,1422,282,944,0,0,0,0
B,3013,0,0,0,0,0,0,0,0,0,0,0,0,125708
C,0,0,25949,57230,142,1908,850,1603,53,169,260,834,0,144,0,0
D,0,109511,4389,6473,135,1018,476,229,28,105,177,120,0,59,4586,0
F,0,0,971,2666,138,127,92,163,5,7,5,87,0,48,0,0
G,0,0,24497,42962,1283,5178,3104,2782,182,1415,756,1173,0,140,0,0
I,0,0,2515,5556,34,270,179,181,3,39,56,210,0,11,0,0
J,0,0,2002,4973,69,96,85,707,4,95,47,535,0,23,0,0
K,0,19920,1162,2036,13,184,64,57,16,3,32,25,0,0,1238,0
L,0,0,1289,1476,19,68,58,481,1,18,10,289,0,0,0,0
M,1000,0,569,758,1,7,11,128,0,76,86,143,0,1,0,0
P,0,0,1647,1989,70,54,200,425,0,67,29,433,0,9,0,0
S,9509,0,2911,104,278,12704,476,212,4031,210,23,86,0,0,0,1131650
W,0,0,123,150,23,105,11,9,0,4,0,10,0,0,0,0
X,0,0,1173,50,91,2972,158,77,1173,79,17,34,0,0,0,0
Z,95874,0,0,0,0,0,0,0,0,0,0,0,0,0,19796133
```

经过角色标注和生成模型后，对于实体发射概率 $p(w_i|t_i)$ 和转移概率 $p(t_i|t_{i-1})$ 这两个重要参数进行求解。在大规模语料前提下采用大数定律可知：

$$p(w_i|t_i) \approx n(w_i, w_i) / n(w_i) \tag{14.7}$$

$$p(t_i|t_{i-1}) = n(t_{i-1}, t_i) / n(t_{i-1}) \tag{14.8}$$

其中 $n(t_{i-1}, t_i)$ 是实体角色到下一个角色的次数； $n(w_i, w_i)$ 、 $n(w_i)$ 、 $n(t_{i-1}, t_i)$ 在训练模型中已经得到，由此得到上层实体词典，然后采用 N-Best 策略将词典传输到下一层 HMM 实体中，进而优化层叠隐马尔可夫的参数因子。以“成都张晓明餐饮管理有限公司是由张先生创办的餐饮企业”为例，如表 14-10 所示。

表 14-10 复杂机构名识别

复杂机构名识别
成都/ns, 张晓明/nr, 餐饮/n, 管理/vn, 有限公司/nis, 是/vshi, 由/p, 张先生/nr, 创办/v, 的/ude1, 餐饮/n, 企业/n]

上例中该公司的各个成分被拆散，无法组成完整的机构名角色标注。将其转化为自定义的角色标注后如表 14-11 所示。

表 14-11 复杂机构名角色标注

复杂机构名角色标注
地名角色观察： [S 1162194][成都 G 83472 B 1200 A 470 D 84 X 4][张晓明 F 4309 B 769 A 266 D 254 X 6][餐饮 C 58 B 12][管理 C 706 B 70 A 5][有限公司 D 2861 A 1 B 1][是 A 2340 B 353 X 20 P 2][由 A 1579 B 16 X 11][张先生 F 4309 B 769 A 266 D 254 X 6][创办 A 20 B 5][的 B 7092 A 4185 X 20][餐饮 C 58 B 12][企业 C 86 A 42 B 11 X 6][B 710] 地名角色标注： [/S, 成都/G, 张晓明/F, 餐饮/C, 管理/C, 有限公司/D, 是/B, 由/A, 张先生/F, 创办/A, 的/B, 餐饮/C, 企业/C, /B]

经过模式串匹配得到如下机构名：【成都张晓明餐饮管理有限公司 GFCC】。

14.5 展望

中文命名实体识别是文本信息处理中一个重要的研究分支，也是信息抽取、问答系统、中文文摘、机器翻译等自然语言处理技术的基础工作。目前中文单一命名实体识别研究较多，但是对多种中文命名实体一体化识别研究较少。本章经过大量研究工作提出了基于层叠隐马尔可夫模型的中文命名实体一体化识别，主要工作包括：

- (1) 对命名实体识别工作进行深入研究。针对当前命名实体识别研究现状、特点难点、主要研究方法、评测标准和相关模型进行分析讨论。
- (2) 中文分词中的新词和歧义词。本章提出了基于统计和规则结合的改进中文分词方法，将统计方法在未登录词上的处理优势和词表规则速度优势相结合，很好地解决了歧义和未登录词切分问题。
- (3) 提高了模型的召回率。本章提出了一种细粒度的特征提取方法，首先利用 CRF++ 工具进行特征模板选择。再基于字词不同粒度的特征模型实验对比，从而进行特征模板的优化和改进，形成自定义的特征模板，提高了系统识别的召回率。
- (4) 多种中文命名实体一体化识别模型设计。本章提出了层叠结构，将系统分为低层的隐马尔可夫模型和高层的隐马尔可夫模型。首先采用低层的隐马尔可夫模型对简单人名、简单地名、简单机构名进行识别。然后将识别的结果提供给更高层的隐马尔可夫模型，再进行对复合地名、机构名的识别。其次，低层识别结果可以为高层识别决策提供支持。最后，通过对模型参数的改进与优化完成对多种命名实体的一体化识别。

第 15 章

自然语言处理实战

本章导读：自然语言处理技术是理论与实践相结合的一门学科，通过前面基础理论知识的介绍，读者对其理论有所认识，但其究竟有何用、怎么用却不是很熟悉。本章通过实例演练，一方面对前面几章知识进行复习回顾，另一方面有利于加深理解研发的相关工作。本章第一个案例是以 GitHub 为例，实现数据提取和可视化；第二个案例是以微博话题为例，实现数据采集、提取、存储与分析。

15.1 GitHub 数据提取与可视化分析

GitHub 作为全球最大的代码托管平台，每小时都有成千上万个项目产生，它为开源做出了不可磨灭的贡献。本章使用 NetworkX 对 GitHub 进行图形分析，通过 GitHub 的丰富数据，构建可以在各种不同的方式下使用的数据模型。这里将 GitHub 用户、代码、仓库构建兴趣图。本节包含三个方面：使用 NetworkX 作图、构建 GitHub 的兴趣图和图算法。

15.1.1 了解 GitHub 的 API

同 Twitter 和 Facebook 一样，第一步就是获取 Git 自带的 API。地址为 <https://developer.github.com/v3/>。其中大部分功能其实我们并不需要，我们关注的仅仅是用户和仓库。所有的 API 访问通过 HTTPS，并从 <https://api.par.github.com> 页面进行访问。所有发送和接收的数据都是 JSON。

创建 API 连接

GitHub 实现了 OAuth 接口，在拥有 GitHub 账户后获取 API 通道的方式有两种。一种叫作 Personal access tokens，你可以为自己使用或实现的 Web 流创建一个个人访问令牌，以允许其他用户授权你的应用程序。一种是 OAuth application，所有的开发人员在开始之前都需要注册他们的应用程序。注册 OAuth 应用分配一个唯一的客户 ID 和客户的密钥。为了简单起见，这里采用 Personal access tokens。单击新建即可生成有对应权限的 Token。生成好的 Token 如图 15-1 所示。

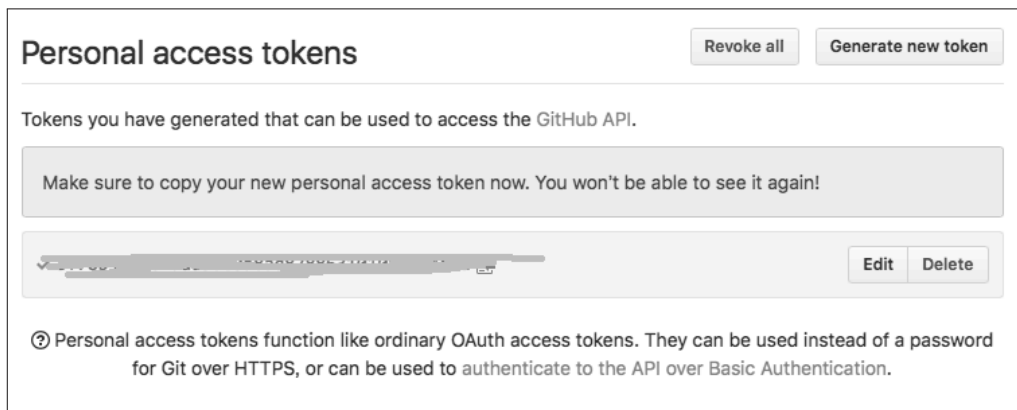


图 15-1 生成对应权限

向 API 根目录发送请求，试一试这个 Token 连接情况：

```
curl https://api.github.com/?access_token=$TOKEN
{
  "current_user_url": "https://api.github.com/user",
  "current_user_authorizations_html_url": "https://github.com/settings/connections/applications{/client_id}",
  "authorizations_url": "https://api.github.com/authorizations",
  "code_search_url": "https://api.github.com/search/code?q={query}{&page,per_page,sort,order}",
  "emails_url": "https://api.github.com/user/emails",
  "emojis_url": "https://api.github.com/emojis",
  "events_url": "https://api.github.com/events",
  "feeds_url": "https://api.github.com/feeds",
```

```

"followers_url": "https://api.github.com/user/followers",
"following_url": "https://api.github.com/user/following{/target}",
"gists_url": "https://api.github.com/gists{/gist_id}",
"hub_url": "https://api.github.com/hub",
"issue_search_url": "https://api.github.com/search/issues?q={query}{&
page,per_page,sort,order}",
"issues_url": "https://api.github.com/issues",
"keys_url": "https://api.github.com/user/keys",
"notifications_url": "https://api.github.com/notifications",
"organization_repositories_url": "https://api.github.com/orgs/{org}/
repos?type,page,per_page,sort}",
"organization_url": "https://api.github.com/orgs/{org}",
"public_gists_url": "https://api.github.com/gists/public",
"rate_limit_url": "https://api.github.com/rate_limit",
"repository_url": "https://api.github.com/repos/{owner}/{repo}",
"repository_search_url": "https://api.github.com/search/repositories?q
={query}{&page,per_page,sort,order}",
"current_user_repositories_url": "https://api.github.com/user/repos{?
type,page,per_page,sort}",
"starred_url": "https://api.github.com/user/starred{/owner}/{repo}",
"starred_gists_url": "https://api.github.com/gists/starred",
"team_url": "https://api.github.com/teams",
"user_url": "https://api.github.com/users/{user}",
"user_organizations_url": "https://api.github.com/user/orgs",
"user_repositories_url": "https://api.github.com/users/{user}/repos{?
type,page,per_page,sort}",
"user_search_url": "https://api.github.com/search/users?q={query}{&
page,per_page,sort,order}"
}

```

GitHub 的 API 符合 HATEOAS 设计, 如果想获取当前用户的信息, 则应该去访问 `api.github.com/user`, 得到如下结果。

```

{
  "login": "luzhijun",
  "id": 15256911,

```

```

"avatar_url": "https://avatars.githubusercontent.com/u/15256911?v=3",
"gravatar_id": "",
"url": "https://api.github.com/users/luzhijun",
"html_url": "https://github.com/luzhijun",
...
}

```

pygithub

你可以通过它方便地用 Python 脚本管理 GitHub，其 API 与 GitHub 对应。举个例子，获取指定用户的所有仓库：

```

from github import Github
# Specify your own access token here
ACCESS_TOKEN = ''
USER = 'luzhijun'
client = Github(ACCESS_TOKEN)
user = client.get_user(USER)
REPOS=user.get_repos()
print(list(REPOS))
[Repository(fullname=" luzhijun/huxblog-boilerplate" ), Repository(
fullname=" luzhijun/leetcode" ), Repository(full_name=" luzhijun/luzhijun
.github.io" ), Repository(fullname=" luzhijun/Optimization" ),
Repository(fullname=" luzhijun/SVDRecommenderSystem" )]

```

15.1.2 使用 NetworkX 作图

了解 NetworkX

NetworkX 是一个用 Python 语言开发的图论与复杂网络建模工具，内置了常用的图与复杂网络分析算法，可以方便地进行复杂网络数据分析、仿真建模等工作。下面创建一个有向图 $x \rightarrow y$ ：

```

import networkx as nx

# 创建有向图
g = nx.DiGraph()
# 加条边x->y
g.add_edge('X', 'Y')
# 打印图的相关数据信息
print (nx.info(g),'\n')

print ("Nodes:", g.nodes())
print ("Edges:", g.edges())
# 节点属性
print ("X props:", g.node['X'])
print ("Y props:", g.node['Y'])
# 边属性
print ("X=>Y props:", g['X']['Y'])
# 更新节点信息
g.node['X'].update({'prop1' : 'value1'})
print ("X props:", g.node['X'])
# 更新边信息
g['X']['Y'].update({'label' : 'label1'})
print ("X=>Y props:", g['X']['Y'])

```

```

Name:
Type: DiGraph #无向图表示为Graph Number of nodes: 2
Number of edges: 1
Average in degree: 0.5000
Average out degree: 0.5000

Nodes: [ 'Y' , 'X' ]
Edges: [( 'X' , 'Y' )]
X props: {}
Y props: {}
X=>Y props: {}
X props: { 'prop1' : 'value1' }
X=>Y props: { 'label' : 'label1' }

```

有向图和无向图都可以给边赋予权重，用到的方法是 `add_weighted edges from`。它接受一个或多个三元组 $[u, v, w]$ 作为参数，其中 u 是起点， v 是终点， w 是权重。例如：

```
g.add_weighted_edges_from([('X','Y',10.0)])
print (g.get_edge_data('X','Y'))
```

```
{ 'weight' : 10.0, 'label' : 'label1' }
```

NetworkX 提供了常用的图论经典算法，如 DFS、BFS、最短路、最小生成树、最大流等，非常丰富。如果不做复杂网络，只做图论方面的工作，也可以应用 NetworkX 作为基本的开发包。

15.1.3 使用 NetworkX 构建兴趣图

兴趣图和社交网络图是有区别的，最大的不同就是兴趣图节点代表的不一定是同一类的东西。

```
from github import Github
import networkx as nx

ACCESS_TOKEN = ''
USER='minrk'
REPO='findspark'
client = Github(ACCESS_TOKEN)
user = client.get_user(USER)
repo=user.get_repo(REPO)
stargazers=list(repo.get_stargazers())\#加星的用户集合

g = nx.DiGraph()
g.add_node(repo.name + '(repo)', type='repo', lang=repo.language, owner=
user.login)

for sg in stargazers:
    g.add_node(sg.login + '(user)', type='user')
```

```

    g.add_edge(sg.login + '(user)', repo.name + '(repo)', type='gazes')
\# 打印图的基本属性
print (nx.info(g),'\n')
\# 打印项目和用户点的基本属性
print (g.node['findspark(repo)'])
print (g.node['luzhijun(user)'],'\n')
\# 打印这条边属性
print (g['luzhijun(user)']['findspark(repo)'])
\# 打印起点为XXX的信息
print (g['luzhijun(user)'])
print (g['findspark(repo)'])
\# 打印用户的出入度信息
print (g.in_edges(['luzhijun(user)']))
print (g.out_edges(['luzhijun(user)']))
\# 打印项目的出入度信息
print (g.in_edges(['findspark(repo)']))
print (g.out_edges(['findspark(repo)']))

```

Name:

Type: DiGraph

Number of nodes: 81

Number of edges: 80

Average in degree: 0.9877 # 80/81 Average out degree: 0.9877 # 80/81

```
{ 'lang': 'Python', 'type': 'repo', 'owner': 'minrk' }
```

```
{ 'type': 'user' }
```

```
{ 'type': 'gazes' }
```

```
{ 'findspark(repo)': { 'type': 'gazes' } }{ }
```

```
[]
```

```
[( 'luzhijun(user)', 'findspark(repo)' )]
```

```
[( 'lendenmc(user)', 'findspark(repo)' ), ( 'nehalecky(user)', '
findspark(repo)' ), ( 'charsmith(user)', 'findspark(repo)' ), ( '
cwharland(user)', 'findspark(repo)' ), ( 'd3borah(user)', 'findspark
(repo)' ), ( 'cimox(user)', 'findspark(repo)' ), ( 'dirmeier(user)',
'findspark(repo)' ), ( 'paulochf(user)', 'findspark(repo)' ), ( '
qingniufly(user)', 'findspark(repo)' ), ( 'hmourit(user)', '

```

```

findspark(repo') ), ( 'ryan-williams(user)', 'findspark(repo') ), ( '
rholder(user)', 'findspark(repo') ), ( 'xysmas(user)', 'findspark(
repo') ), ( 'linkTDP(user)', 'findspark(repo') ), ( 'minimaxir(user)'
, 'findspark(repo') ), ( 'KLXN(user)', 'findspark(repo') ), ( '
quadnix(user)', 'findspark(repo') ), ( 'luzhijun(user)', 'findspark(
repo') ), ( 'chrinide(user)', 'findspark(repo') ), ( '
jaredmichaelsmith(user)', 'findspark(repo') ), ( 'rgbkrk(user)', '
findspark(repo') ), ( 'jiamo(user)', 'findspark(repo') ), ( 'drizham(
user)', 'findspark(repo') ), ( 'assumednormal(user)', 'findspark(
repo') ), ( 'cvincent00(user)', 'findspark(repo') ), ( 'xiaohan2012(
user)', 'findspark(repo') ), ( 'dapurv5(user)', 'findspark(repo') ),
( 'pchalasani(user)', 'findspark(repo') ), ( 'benetka(user)', '
findspark(repo') ), ( 'rohithreddy(user)', 'findspark(repo') ), ( '
seanjensengrey(user)', 'findspark(repo') ), ( 'opikalo(user)', '
findspark(repo') ), ( 'amontalenti(user)', 'findspark(repo') ), ( '
Bekterra(user)', 'findspark(repo') ), ( 'Grillz(user)', 'findspark(
repo') ), ( 'nchammas(user)', 'findspark(repo') ), ( 'markbarks(user)
', 'findspark(repo') ), ( 'jesusjsc(user)', 'findspark(repo') ), ( '
d2207197(user)', 'findspark(repo') ), ( 'Erstwild(user)', 'findspark
(repo') ), ( 'henridf(user)', 'findspark(repo') ), ( 'ranjankumar-gh(
user)', 'findspark(repo') ), ( 'locojay(user)', 'findspark(repo') ),
( 'WilliamQLiu(user)', 'findspark(repo') ), ( 'szinya(user)', '
findspark(repo') ), ( 'seanjh(user)', 'findspark(repo') ), ( 'vherasme
(user)', 'findspark(repo') ), ( 'stared(user)', 'findspark(repo') ),
( 'freeman-lab(user)', 'findspark(repo') ), ( 'wy36101299(user)', '
findspark(repo') ), ( 'esafak(user)', 'findspark(repo') ), ( 'napjon(
user)', 'findspark(repo') ), ( 'richardskim111(user)', 'findspark(
repo') ), ( 'SimonArnu(user)', 'findspark(repo') ), ( 'lmillefiori(
user)', 'findspark(repo') ), ( 'binhe22(user)', 'findspark(repo') ),
( 'robcowie(user)', 'findspark(repo') ), ( 'jazzwang(user)', '
findspark(repo') ), ( 'Dumbris(user)', 'findspark(repo') ), ( '
giulioungaretti(user)', 'findspark(repo') ), ( 'cbouey(user)', '
findspark(repo') ), ( 'andrewiird(user)', 'findspark(repo') ), ( '
DaniGate(user)', 'findspark(repo') ), ( 'rdhyee(user)', 'findspark(
repo') ), ( 'lgautier(user)', 'findspark(repo') ), ( 'sandysnunes(user)
', 'findspark(repo') ), ( 'willcline(user)', 'findspark(repo') ), (
'aliciatb(user)', 'findspark(repo') ), ( 'd18s(user)', 'findspark(

```

```
repo')'), ('he0x(user)', 'findspark(repo)'), ('liulixiang1988(user)', 'findspark(repo)'), ('alexandercbooth(user)', 'findspark(repo)'), ('branning(user)', 'findspark(repo)'), ('bgu0IQ(user)', 'findspark(repo)')])
[]
```

15.1.4 NetWorkX 部分统计指标

度中心性 (Degree Centrality)

这是在网络分析中刻画节点中心性 (Centrality) 的直接度量指标。一个节点的节点度越大, 就意味着这个节点的度中心性越高, 该节点在网络中就越重要。某点的度范围为 [0,1]。

- ◎ `degree centrality(G)`, 计算节点的中心度。
- ◎ `in_degree centrality(G)`, 计算节点的入度中心性。
- ◎ `out_degree centrality(G)`, 计算节点的出度中心性。

中介中心性/中间中心性 (Between Centrality)

以经过某个节点的最短路径数目来刻画节点重要性的指标。如果两个不相邻的参与者 k 和 j 想要与对方互动, 而参与者 i 处在它们的路径上, 那么 i 可能对它们之间的互动拥有一定的控制力。中介性用来度量 i 对于其他结点的控制能力, 即如果 i 处在非常多结点的交互路径上, 那么 i 就是一个重要的参与者。

- ◎ `betweenness centrality(G[, normalized, ...])`, 计算节点的中介中心性。
- ◎ `edge_betweenness centrality(G[, normalized, ...])`, 计算边缘的接近中心性。

接近中心性 (Closeness Centrality)

反映在网络中某一节点与其他节点之间的接近程度。这种中心性的观察视角主要基于接近度或者距离。它的基本思想是如果一个参与者能很容易地与所有其他参与者进行互动, 那么它就是中心的, 即它到其他所有参与者的距离要足够短。于是, 我们就可以使用最短距离来计算这个数值。假设参与者 i 和参与者 j 之间的最短距离记为 $d(i, j)$ (由最短路径上的链接数目度量)。

◎ `closeness centrality(G[, v, weighted_edges])`，计算节点的接近中心性。

```
from operator import itemgetter
kkg = nx.generators.small.krackhardt_kite_graph()
print ("Degree Centrality")
print (sorted(nx.degree_centrality(kkg).items(),
              key=itemgetter(1), reverse=True), '\n')
print ("Betweenness Centrality")
print (sorted(nx.betweenness_centrality(kkg).items(),
              key=itemgetter(1), reverse=True), '\n')
print ("Closeness Centrality")
print (sorted(nx.closeness_centrality(kkg).items(),
              key=itemgetter(1), reverse=True))
```

```
Degree Centrality [(3, 0.6666666666666666), (5, 0.5555555555555556), (6,
0.5555555555555556), (0, 0.4444444444444444), (1, 0.4444444444444444),
(2, 0.3333333333333333), (4, 0.3333333333333333), (7,
0.3333333333333333), (8, 0.2222222222222222), (9, 0.1111111111111111)]
```

```
Betweenness Centrality [(7, 0.38888888888888884), (5,
0.23148148148148148), (6, 0.23148148148148148), (8, 0.2222222222222222),
(3, 0.10185185185185183), (0, 0.023148148148148143), (1,
0.023148148148148143), (2, 0.0), (4, 0.0), (9, 0.0)]
```

```
Closeness Centrality [(5, 0.6428571428571429), (6, 0.6428571428571429),
(3, 0.6), (7, 0.6), (0, 0.5294117647058824), (1, 0.5294117647058824),
(2, 0.5), (4, 0.5), (8, 0.42857142857142855), (9, 0.3103448275862069)]
```

15.1.5 构建 GitHub 的兴趣图

找出“大神”(Closeness Centrality)

GitHub “大神”的关注度很高，在社区中很活跃。除了前面加星，类似微博、Facebook，GitHub 还有“关注”关系可以用。现在对每个加星的用户进行扩展。这里简单起见，关注者

都是现有的节点，不再新增节点。由于 API 每小时限制使用 5000 个请求，打印下剩余请求数看有没有耗尽，耗尽的话就抛出异常。

```
import sys
for i, sg in enumerate(stargazers):
    # 增加关注联系，如果有关注者的话
    try:
        for follower in sg.get_followers():
            if follower.login + '(user)' in g:
                g.add_edge(follower.login + '(user)', sg.login + '(user)',
                           type='follows')
    except Exception: #ssl.SSLError
        sys.stderr.write("Encountered an error fetching followers for",
                        \
                        sg.login, "Skipping.")

print ("Processed", i+1, " stargazers. Num nodes/edges in graph", \
      g.number_of_nodes(), "/", g.number_of_edges())
print ("Rate limit remaining", client.rate_limiting)
```

```
rocessed 1 stargazers. Num nodes/edges in graph 81 / 80
Rate limit remaining (4999, 5000)
Processed 2 stargazers. Num nodes/edges in graph 81 / 83
Rate limit remaining (4987, 5000)
Processed 3 stargazers. Num nodes/edges in graph 81 / 84
Rate limit remaining (4985, 5000)
Processed 4 stargazers. Num nodes/edges in graph 81 / 85
Rate limit remaining (4983, 5000)
Processed 5 stargazers. Num nodes/edges in graph 81 / 86
Rate limit remaining (4981, 5000)
Processed 6 stargazers. Num nodes/edges in graph 81 / 90
Rate limit remaining (4968, 5000)
...
Rate limit remaining (4857, 5000)
Processed 78 stargazers. Num nodes/edges in graph 81 / 95
```

```

Rate limit remaining (4856, 5000)
Processed 79 stargazers. Num nodes/edges in graph 81 / 95
Rate limit remaining (4855, 5000)
Processed 80 stargazers. Num nodes/edges in graph 81 / 95
Rate limit remaining (4854, 5000)

```

接下来，对关联关系进行统计。

```

from operator import itemgetter
from collections import Counter

# 显示更新的图信息
print (nx.info(g), '\n')

# 每个打星用户的关注者数量不同
print (len([e for e in g.edges_iter(data=True) if e[2]['type'] == '
follows'])), '\n')

# 查看某个打星用户有多少关注者
print (len([e
            for e in g.edges_iter(data=True)
            if e[2]['type'] == 'follows' and e[1] == 'freeman-lab(
user)'])), '\n')

# 打印最多的前10个节点
print (list(sorted([n for n in g.degree_iter()], key=itemgetter(1),
reverse=True)[:10])), '\n')

# 对每个打星用户的关注者数目计数
c = Counter([e[1] for e in g.edges_iter(data=True) if e[2]['type'] == '
follows'])
popular_users = [ (u, f) for (u, f) in c.most_common() if f > 0 ]
print ("Number of popular users", len(popular_users))
print ("Top popular users:", popular_users[:10])

```

```
Name: Type: DiGraph Number of nodes: 81 Number of edges: 95 Average in
degree: 1.1728 Average out degree: 1.1728

15

4

[( 'findspark(repo)', 80), ( 'rgbkrk(user)', 6), ( 'freeman-lab(user)
', 5), ( 'esafak(user)', 4), ( 'andrewiiird(user)', 4), ( 'minimaxir(
user)', 3), ( 'nchammas(user)', 3), ( 'rholder(user)', 2), ( '
chrinide(user)', 2), ( 'dapurv5(user)', 2)]

Number of popular users 9 Top popular users: [( 'freeman-lab(user)', 4)
, ( 'rgbkrk(user)', 3), ( 'minimaxir(user)', 2), ( 'rholder(user)',
1), ( 'amontalenti(user)', 1), ( 'rdhyee(user)', 1), ( 'stared(user)'
, 1), ( 'esafak(user)', 1), ( 'nchammas(user)', 1)]
```

(‘freeman-lab(user)’, 4) 指 freeman-lab 在 findspark 项目中的加星者中还有 4 个关注者。为了更清楚地挖掘图信息，用上面介绍的几个图算法来看一下这个图。由于“仓库”节点有大量的度，其他节点相对来说度都很小，因此先删掉“仓库”节点再计算。

```
from operator import itemgetter
h = g.copy()
# 移除中心节点
h.remove_node('findspark(repo)')

dc = sorted(nx.degree_centrality(h).items(),
            key=itemgetter(1), reverse=True)

print ("Degree Centrality")
print (dc[:10], '\n')
bc = sorted(nx.betweenness_centrality(h).items(),
            key=itemgetter(1), reverse=True)

print ("Betweenness Centrality")
```

```

print (bc[:10],'\n')

print ("Closeness Centrality")
cc = sorted(nx.closeness centrality(h).items(),
            key=itemgetter(1), reverse=True)
print (cc[:10])

```

Degree Centrality

```

[( 'rgbkrk(user)' , 0.06329113924050633),
 ( 'freeman-lab(user)' , 0.05063291139240506),
 ( 'esafak(user)' , 0.0379746835443038),
 ( 'andrewiird(user)' , 0.0379746835443038),
 ( 'minimaxir(user)' , 0.02531645569620253),
 ( 'nchammas(user)' , 0.02531645569620253),
 ( 'rholder(user)' , 0.012658227848101266),
 ( 'chrinide(user)' , 0.012658227848101266),
 ( 'dapurv5(user)' , 0.012658227848101266),
 ( 'amontalenti(user)' , 0.012658227848101266)]

```

Betweenness Centrality

```

[( 'rgbkrk(user)' , 0.0009737098344693282),
 ( 'esafak(user)' , 0.0006491398896462187),
 ( 'nchammas(user)' , 0.0001622849724115547),
 ( 'linkTDP(user)' , 0.0),
 ( 'nehalecky(user)' , 0.0),
 ( 'charsmith(user)' , 0.0),
 ( 'cwharland(user)' , 0.0),
 ( 'd3borah(user)' , 0.0),
 ( 'cimox(user)' , 0.0),
 ( 'dirmeier(user)' , 0.0)]

```

Closeness Centrality

```

[( 'andrewiird(user)' , 0.04050632911392405),
 ( 'esafak(user)' , 0.03375527426160337),
 ( 'chrinide(user)' , 0.028768699654775604),
 ( 'rgbkrk(user)' , 0.02531645569620253),
 ( 'alexandercbooth(user)' , 0.02278481012658228),

```

```
( 'dapurv5(user)', 0.012658227848101266),  
( 'Bekterra(user)', 0.012658227848101266),  
( 'nchammas(user)', 0.012658227848101266),  
( 'he0x(user)', 0.012658227848101266),  
( 'Hguimaraes(user)', 0.012658227848101266)]
```

我们来见见两位“大神”，一位是数据显示度中心性和中介中心性最大的 **rgbkrk**，如图 15-2 所示（图片来源于 GitHub）。

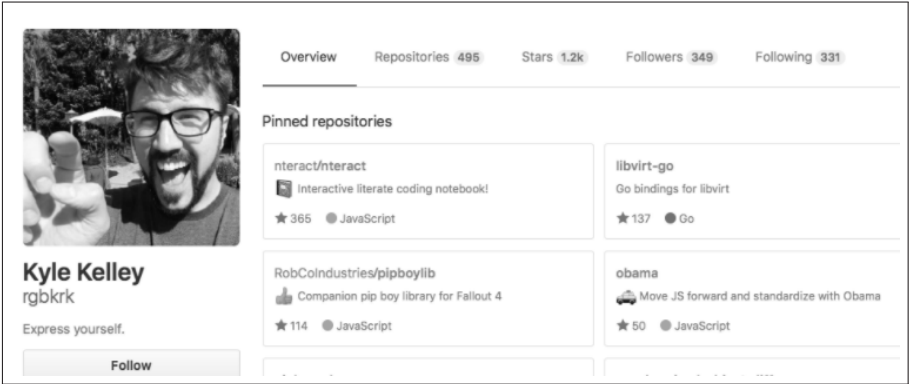


图 15-2 数据显示度中心性和中介中心性最大的 **rgbkrk**

Kyle Kelley 来自硅谷，是 jupyter、ipython、cloudpipe 等大项目的开发组员，影响力很大。另一位是接近中心性最大的 **andrewiird**，如图 15-3 所示（图片来源于 GitHub）。

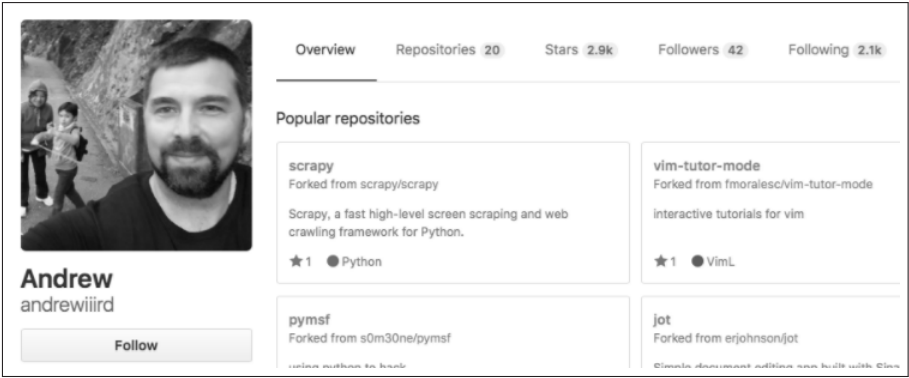


图 15-3 接近中心性最大的 **andrewiird**

Andrew 来自香港，关注与被关注比率几乎有两个数量级的差别，而 Kyle Kelley 的这个比值接近一比一，这也说明 Andrew 的“互动性”并不是很强。

找出活跃项目

现在图中只有一个项目，其余节点都是用户。为了找出活跃项目，遍历每个用户，找到其加星的项目添加到图中，扩充图的项目节点。

```

MAX_REPOS = 500
for i, sg in enumerate(stargazers):
    print (sg.login)
    try:
        for starred in sg.get_starred()[:MAX_REPOS]: # Slice to avoid
            supernodes
                g.add_node(starred.name + '(repo)', type='repo', lang=
                    starred.language, \
                        owner=starred.owner.login)
                g.add_edge(sg.login + '(user)', starred.name + '(repo)',
                    type='gazes')
    except Exception: #ssl.SSLError:
        print ("Encountered an error fetching starred repos for", sg.
            login, "Skipping.")

    print ("Processed", i+1, "stargazers' starred repos")
    print ("Num nodes/edges in graph", g.number_of_nodes(), "/", g.
        number_of_edges())
    print ("Rate limit", client.rate_limiting)

```

```

pchallasani
Processed 1 stargazers' starred repos
Num nodes/edges in graph 176 / 190
Rate limit (4996, 5000)
rgbkrk
Encountered an error fetching starred repos for rgbkrk Skipping.
Processed 2 stargazers' starred repos
Num nodes/edges in graph 266 / 280

```

```

Rate limit (4993, 5000)
napjon
Processed 79 stargazers' starred repos
Num nodes/edges in graph 12692 / 17579
Rate limit (4369, 5000)
luzhijun
Processed 80 stargazers' starred repos
Num nodes/edges in graph 12693 / 17581
Rate limit (4368, 5000)

```

以上操作挺费时间的，节点数由几百个一下子扩充到上万个，可以把 MAX_REPOS 改小一点，然后就可以去抽空泡杯咖啡了。获取所有项目数据后，接下来就是要统计哪个是热点项目，可以查询哪个用户“mark”了哪些项目，他喜欢用什么程序语言，还有那些加星加到“手软”的用户也可以查询出来。

```

print (nx.info(g),'\n')
# 获取图里面的所有项目构成列表
repos = [n for n in g.nodes_iter() if g.node[n]['type'] == 'repo']

# 关注最多的前10个项目
print ("Popular repositories")
print (sorted([(n,d)
                for (n,d) in g.in_degree_iter()
                if g.node[n]['type'] == 'repo'], \
                key=itemgetter(1), reverse=True)[:10]))

print "Respositories that luzhijun has bookmarked"
print ([g.node[n]['lang']
        for n in g['luzhijun(user)']
        if g['luzhijun(user)'][n]['type'] == 'gazes'])

# 用户喜爱的程序语言

print ("Programming languages luzhijun is interested in")
print (list(set([g.node[n]['lang']
                for n in g['luzhijun(user)']

```



```

        if g['luzhijun(user)'][n]['type'] == 'gazes']))
# 查看关注项目最多的用户(超过MAX_REPOS)
print ("Supernode candidates")
print (sorted([(n, len(g.out_edges(n))
                for n in g.nodes_iter()
                if g.node[n]['type'] == 'user' and len(g.out_edges(n))
                > MAX_REPOS], \
                key=itemgetter(1), reverse=True))

```

Name:

Type: DiGraph

Number of nodes: 12693

Number of edges: 17581

Average in degree: 1.3851

Average out degree: 1.3851

Popular repositories

```

[( 'findspark(repo)', 80), ( 'spark(repo)', 27), ( 'tensorflow(repo)',
24), ( 'luigi(repo)', 21), ( 'ipython(repo)', 21), ( 'data-science-
ipython-notebooks(repo)', 20), ( 'awesome-public-datasets(repo)', 20),
( 'spark-notebook(repo)', 19), ( 'docker(repo)', 18), ( '
Probabilistic-Programming-and-Bayesian-Methods-for-Hackers(repo)', 17)]

```

Respositories that ocanbascil has bookmarked

```

[( 'til(repo)', 'VimL' ), ( 'react-joyride(repo)', 'JavaScript' ), (
'django(repo)', 'Python' ), ( 'peewee(repo)', 'Python' ), ( '
select2(repo)', 'JavaScript' ), ( 'requests(repo)', 'Python' ), ( '
benfords-law(repo)', 'JavaScript' ), ( 'daterangepicker(repo)', '
JavaScript' ), ( 'django-zappa(repo)', 'Python' ), ( 'layered(repo)',
'Python' ), ( 'coffeescript(repo)', 'CoffeeScript' ), ( 'awesome-
jekyll(repo)', None), ( 'lektor(repo)', 'Python' ), ( 'aetycoon(repo)
', 'Python' ), ( 'computer-science(repo)', None), ( '
PerformanceEngine(repo)', 'Python' ), ( 'pattern_classification(repo)
', 'Jupyter Notebook' ), ( 'node-v0.x-archive(repo)', None), ( '
django-crispy-forms(repo)', 'Python' ), ( 'hug(repo)', 'Python' ), (
'euler-coffeescript(repo)', 'CoffeeScript' ), ( 'pandas(repo)', '
Python' ), ( 'pachyderm(repo)', 'Go' ), ( 'eulerclash(repo)', None),

```

```
( 'Pipe(repo)', 'Python' ), ( 'data-science-ipython-notebooks(repo)',
    'Python' ), ( 'TweetHit(repo)', 'Python' ), ( 'captionmash(repo)',
    None), ( 'django-tinymce(repo)', 'Python' ), ( 'dataset(repo)', '
    Python' ), ( 'shepherd(repo)', 'CSS' ), ( 'data-science-from-scratch(
    repo)', 'Python' ), ( 'spark(repo)', 'Scala' ), ( 'data-science-
    blogs(repo)', 'Python' ), ( 'specter(repo)', 'Clojure' ), ( '
    tensorflow(repo)', 'R' ), ( 'findspark(repo)', 'Python' ), ( '
    localtunnel(repo)', 'JavaScript' )]
```

```
Programming languages ocanbascil is interested in
[ 'CSS', 'VimL', 'Python', 'JavaScript', 'Clojure', None, '
CoffeeScript', 'R', 'Jupyter Notebook', 'Scala', 'Go' ]
```

```
Supernode candidates [( 'he0x(user)', 501), ( 'esafak(user)', 501)]
```

从结果可以看出,关注 findspark 项目的人大约有三分之一也关注了 Spark 和 Tensorflow。TensorFlow 是谷歌基于 DistBelief 进行研发的第二代人工智能学习系统,最近的“机器学习热”也无愧于这三分之一的编程爱好者。此外,还有大约四分之一的人关注了 Luigi,查了下其主要目的是为了解决需要长期运行的流式批处理任务的管理,可以链接很多个任务,使它们自动化并进行故障管理,估计做 Spark 任务的人也需要 Luigi。

找出热门语言

怎么查询当前图中所最热门的项目语言? 我们可以查找每个项目,然后抽取语言并放入 counter 计数器里面进行统计,这个看起来也不费事。如果要查询有多少用户以某种语言编程呢? 这就需要扫描每个项目的入度用户,最后统计求和,时间复杂度挺大的。一种好的思路就是扩充图,将语言单独作为一个节点,最终的逻辑图如 15-4 所示。

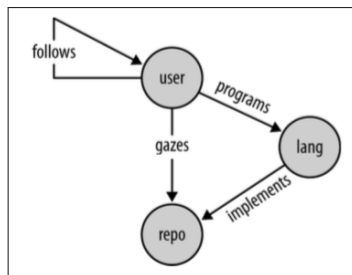


图 15-4 逻辑图

```

repos = [n
          for n in g.nodes_iter()
          if g.node[n]['type'] == 'repo']

for repo in repos:
    # 消除None,有些空项目语言为None
    lang = (g.node[repo]['lang'] or "") + "(lang)"
    # 加星于repo的用户
    stargazers = [u
                  for (u, r, d) in g.in_edges_iter(repo, data=True)
                  if d['type'] == 'gazes'
                  ]
    for sg in stargazers:
        g.add_node(lang, type='lang')
        g.add_edge(sg, lang, type='programs')
        g.add_edge(lang, repo, type='implements')

```

接下来看一下对语言的统计信息:

```

print (nx.info(g),'\n')
# 显示图里有什么语言
print ([n
        for n in g.nodes_iter()
        if g.node[n]['type'] == 'lang'])
# 某个用户使用的语言
print ([n
        for n in g['ocanbascil(user)']
        if g['ocanbascil(user)'][n]['type'] == 'programs'],'\n')
# 查询最热门的语言
print ("Most popular languages")
print (sorted([(n, g.in_degree(n))
               for n in g.nodes_iter()
               if g.node[n]['type'] == 'lang'], key=itemgetter(1), reverse=True)
        [:10])
# 查询用某种语言的有多少人
python_programmers = [u

```

```

        for (u, l) in g.in_edges_iter('Python(lang)')
            if g.node[u]['type'] == 'user']
print ("Number of Python programmers:", len(python_programmers))

javascript_programmers = [u for
    (u, l) in g.in_edges_iter('JavaScript(lang)')
        if g.node[u]['type'] == 'user']
print ("Number of JavaScript programmers:", len(javascript_programmers))

# 两个语言都用的人
print ("Number of programmers who use JavaScript and Python")
print (len(set(python_programmers).intersection(set(
    javascript_programmers))))

# 只用Python不用JS的人
print ("Number of programmers who use JavaScript but not Python")
print (len(set(javascript_programmers).difference(set(python_programmers
))))

# XXX: Can you determine who is the most polyglot programmer?

```

Name: Type: DiGraph Number of nodes: 12807 Number of edges: 31824

Average in degree: 2.4849 Average out degree: 2.4849

```

[ 'LiveScript(lang)', 'APL(lang)', 'Swift(lang)', 'Java(lang)',
  'Nix(lang)', 'Erlang(lang)', 'Common Lisp(lang)', 'Python(lang)',
  , 'GCC Machine Description(lang)', 'Prolog(lang)', 'PigLatin(lang)',
  , 'Vala(lang)', 'SAS(lang)', 'FORTRAN(lang)', 'Cuda(lang)', '
  Matlab(lang)', 'HCL(lang)', 'F#(lang)', 'Verilog(lang)', 'Emacs
  Lisp(lang)', 'Inno Setup(lang)', 'Vue(lang)', 'TeX(lang)', '
  DTrace(lang)', 'Processing(lang)', 'Hack(lang)', 'Lua(lang)', '
  Assembly(lang)', 'Parrot(lang)', 'Shell(lang)', 'Arduino(lang)',
  , 'R(lang)', 'C#(lang)', 'Rust(lang)', 'Standard ML(lang)', '
  Puppet(lang)', 'Gosu(lang)', 'C(lang)', 'PHP(lang)', 'Scala(lang)',
  , 'Gnuplot(lang)', 'Crystal(lang)', 'Objective-J(lang)', '
  COBOL(lang)', 'Cucumber(lang)', 'ApacheConf(lang)', 'Brainfuck(
  lang)', 'Kotlin(lang)', 'Pascal(lang)', 'Pike(lang)', 'RAML(lang)

```

```

)', 'Lean(lang)', 'Logos(lang)', 'Elixir(lang)', 'Dart(lang)',
'C++(lang)', 'CMake(lang)', 'Clojure(lang)', 'D(lang)', '
Batchfile(lang)', 'OpenSCAD(lang)', 'Coq(lang)', 'PowerShell(lang)
', 'Visual Basic(lang)', 'VimL(lang)', 'NetLogo(lang)', 'OCaml(
lang)', 'VHDL(lang)', 'Ruby(lang)', 'Go(lang)', 'Metal(lang)',
'Jupyter Notebook(lang)', 'KiCad(lang)', 'Smarty(lang)', 'Tcl(
lang)', 'CoffeeScript(lang)', 'QML(lang)', '(lang)', 'Scheme(
lang)', 'HTML(lang)', 'Makefile(lang)', 'CartoCSS(lang)', '
DIGITAL Command Language(lang)', 'AppleScript(lang)', 'Perl6(lang)
', 'Protocol Buffer(lang)', 'Julia(lang)', 'Awk(lang)', 'Elm(lang)
', 'Haskell(lang)', 'TLA(lang)', 'Nimrod(lang)', 'JavaScript(
lang)', 'Max(lang)', 'TypeScript(lang)', 'GAP(lang)', 'Scilab(
lang)', 'Web Ontology Language(lang)', 'XSLT(lang)', 'Objective-C
++(lang)', 'Perl(lang)', 'PLSQL(lang)', 'PureScript(lang)', '
PLpgSQL(lang)', 'Eagle(lang)', 'Objective-C(lang)', 'Racket(lang)
', 'OpenEdge ABL(lang)', 'Groovy(lang)', 'Groff(lang)', 'Frege(
lang)', 'NSIS(lang)', 'Mathematica(lang)', 'CSS(lang)']

```

```

['JavaScript(lang)', 'R(lang)', 'Python(lang)', 'Jupyter Notebook
(lang)', 'VimL(lang)', 'Scala(lang)', 'Clojure(lang)', '
CoffeeScript(lang)', 'Go(lang)', '(lang)', 'CSS(lang)']

```

Most popular languages

```

[( 'Python(lang)', 80), ( '(lang)', 74), ( 'JavaScript(lang)', 74), (
'Jupyter Notebook(lang)', 69), ( 'Java(lang)', 67), ( 'HTML(lang)',
67), ( 'C++(lang)', 66), ( 'Scala(lang)', 65), ( 'Shell(lang)', 62),
( 'C(lang)', 62)]

```

Number of Python programmers: 80

Number of JavaScript programmers: 74

Number of programmers who use JavaScript and Python

74

Number of programmers who use JavaScript but not Python

0

从结果可以看出，GitHub 上受欢迎的语言排行榜中 JS、R、Python 等脚本语言占据主

流。这里 Scala 的关注度也很靠前，主要因为“项目之源”是 findspark；Go 也很火。另一个有趣的现象是，玩 Python 的程序员必会玩 JS，玩 JS 的人不一定会玩 Python。这正符合现在 IT 界的现状，这个差别不明显，换个国内的项目估计就明显了。

以上数据规模并不算大，如果有兴趣可以访问 GitHub Archive。GitHub Archive 提供海量全局层面的数据，我们可以使用一些推荐的“大数据”工具来探索它。

15.1.6 可视化

依靠 NetworkX 的导出能力，可以通过 JSON 的 JavaScript 工具包 D3 实现可视化，但也有许多其他的工具包，比如还可以考虑可视化图。Graphviz 是高度可配置的经典工具，可以设计出非常复杂的图形和位图图像。传统上在终端上运行，但现在也有支持大多数平台的用户界面。另一个流行的开源项目是 Gephi，其在交互上做得很好，在过去的几年中 Gephi 已经迅速流行。这里由于有上万个项目节点，一张图乌漆墨黑什么都看不到，并且浏览时很卡顿。所以下面只截取用户点和语言点的可视化。

```
import os
import json
from IPython.display import IFrame
from IPython.core.display import display
from networkx.readwrite import json_graph
print ("Stats on the full graph" )
print (nx.info(g))
# 只提取用户和语言节点
mtsw_users = [n for n in g if g.node[n]['type'] == 'user'] +[n for n in
g if g.node[n]['type'] == 'lang']
h = g.subgraph(mtsu_users)
print ("Stats on the extracted subgraph" )
print (nx.info(h))
# JSON导出
d = json_graph.node_link_data(h)
json.dump(d, open('force.json', 'w'))
viz_file = 'files/force.html'
# D3可视化
display(IFrame(viz_file, '100%', '900px'))
```

可视化呈现如图 15-5 所示。

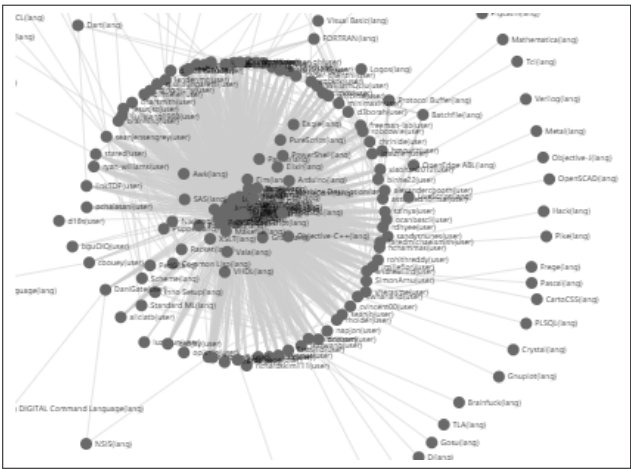


图 15-5 截取部分用户点和语言点的可视化图

如果要分析语言之间的关联，则可以从一个项目中获取多个语言。比如是 a、b、c，然后做无向图，标记 a-b-c 的权重为 1；当在其他项目中还有 a、b 同时出现时，a-b 的权重为 2；以此类推，最后通过图形化展示，如图 15-6 所示。

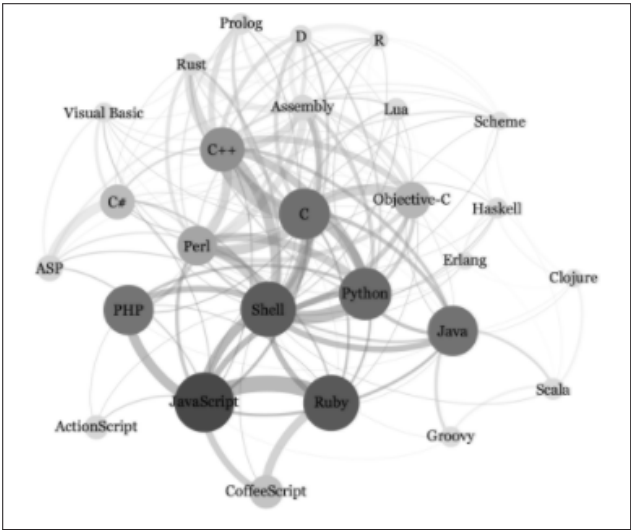


图 15-6 语言之间的关联

15.2 微博话题爬取与存储分析

大数据社会下数据就是黄金，微博提供的开放 API 接口只可以获得少量无关紧要的数据，Twitter 等社交平台会提供一些数据接口供研究人员获取大量研究数据。

本节基于 Python，以新浪微博为数据平台，从数据采集、关键字提取、数据存储三个角度，用最简单的策略来挖掘我们的“黄金”。

有爬虫基础的人可以跳过数据采集部分直接看“上海租房”话题挖掘实战项目，项目地址为 <https://github.com/luzhijun/weiboSA>（目前已更新豆瓣小组爬取）。

15.2.1 数据采集

使用 Python 是因为代码简洁，虽然计算比 Java 和 C 慢很多，但数据采集时间开销大部分是 I/O 部分的，用 Java 或者 C 写效率也提高不了多少。数据采集基本用爬虫机器人，下面介绍常用来做爬虫的几个库。

urllib

怎样“抓”网页呢？其实就是根据 URL 来获取它的网页信息。虽然我们在浏览器中看到的是一幅幅优美的画面，但这其实是由浏览器解释才呈现出来的，实质上它就是一段 HTML 代码，外加 JS、CSS。如果把网页比作一个人，那么 HTML 便是他的骨架，JS 便是他的肌肉，CSS 便是它的衣服。所以最重要的部分是存在于 HTML 中的，下面我们就写个例子来“爬”一个网页下来。

```
import urllib2
response = urllib2.urlopen("http://www.baidu.com")
print response.read()
```

结果就和在 Chrome 等浏览器中右键查看源码是一样的，urllib2 是 Python 内置库，简化了 http 的用法（urllib2.urlopen 相当于 Java 中的 HttpURLConnection）。有 urllib2 那肯定有 urllib，urllib2 可以接受一个 Request 类的实例来设置 URL 请求的 headers，但 urllib 仅可以接受 URL。这意味着不可以伪装 User Agent 字符串等。urllib2 在 Python 3.x 中被改为 urllib.request。接下来用 urllib2 伪装 iPhone 6 浏览，模拟浏览器发送 GET 请求。

```

req = request.Request('http://www.douban.com/')
req.add_header('User-Agent', 'Mozilla/6.0 (iPhone; CPU iPhone OS 8_0
like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/8.0 Mobile
/10A5376e Safari/8536.25')
with request.urlopen(req) as f:
    print('Status:', f.status, f.reason)
    print('Data:', f.read().decode('utf-8'))

```

结果会返回移动版的源码信息：

```
...
```

```
...
```

如果想要以 POST 方式提交，只要在 Request 中附加 data 字段即可，下面附加用户名和密码来登录新浪微博。

```

#我们模拟一个微博登录，先读取登录的邮箱和口令，然后按照weibo.cn的登录页
的格式以username=xxx&password=xxx的编码传入：
from urllib import parse
print('Login to weibo.cn...')
email = input('Email: ')
passwd = input('Password: ')
login_data = parse.urlencode([
    ('username', email),
    ('password', passwd),
    ('entry', 'weibo'),
    ('client_id', ''),
    ('savestate', '1'),
    ('ec', ''),
    ('pagerefer', 'https://passport.weibo.cn/signin/welcome?entry=mweibo
&r=http%3A%2F%2Fm.weibo.cn%2F')
])

```

```

req = request.Request('https://passport.weibo.cn/sso/login')
req.add_header('Origin', 'https://passport.weibo.cn')
req.add_header('User-Agent', 'Mozilla/6.0 (iPhone; CPU iPhone OS 8_0
like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/8.0 Mobile
/10A5376e Safari/8536.25')
req.add_header('Referer', 'https://passport.weibo.cn/signin/login?entry=
mweibo&res=wel&wm=3349&r=http%3A%2F%2Fm.weibo.cn%2F')

with request.urlopen(req, data=login_data.encode('utf-8')) as f:
    print('Status:', f.status, f.reason)
    for k, v in f.getheaders():
        print('s: %s' % (k, v))
    print('Data:', f.read().decode('utf-8'))

```

其中 Origin 和 Referer 字段是反“反盗链”，就是检查你发送请求的 header 里面，Referer 站点是不是它自己。

cookielib

爬虫被封的一个依据就是重复 IP，因此可以为爬虫设置不同的代理 IP。此外有些网站需要 cookie 才能查看。所谓 cookie，是指某些网站为了辨别用户身份、进行 Session 跟踪而储存在用户本地终端上的数据（通常经过加密）。比如说有些网站需要登录后才能访问某个页面，在登录之前，你想抓取某个页面内容是不允许的。那么可以利用 urllib2 库保存我们登录的 cookie，然后抓取其他页面。

cookielib 模块的主要作用是提供可存储 cookie 的对象，以便于与 urllib2 模块配合使用来访问 Internet 资源。cookielib 模块非常强大，我们可以利用本模块的 CookieJar 类的对象来捕获 cookie 并在后续连接请求时重新发送，比如可以实现模拟登录功能。该模块主要的对象有 CookieJar、FileCookieJar、MozillaCookieJar、LWPCookieJar。

它们的关系：CookieJar—派生→FileCookieJar—派生→MozillaCookieJar 和 LWPCookieJar。

```

from urllib import request
from http.cookiejar import CookieJar

cookie=CookieJar()

```

```

cookie_support= request.HTTPCookieProcessor(cookie)#cookie处理器
opener = request.build_opener(cookie_support)
opener.open('http://www.baidu.com')
for item in cookie:
    print(item.name,':',item.value)

```

结果:

```

BAIDUID : E4DECD4AF63915B9AFF5AC28951A3DAA:FG=1
BIDUPSID : E4DECD4AF63915B9AFF5AC28951A3DAA
H_PS_PSSID : 1437_18241_17944_21079_18559_21454_21406_21377_21191_21321
PSTM : 1477631558
BDSVRTM : 0
BD_HOME : 0

```

这里使用默认的 CookieJar 对象, 如果要保存 Cookie, 则可以使用 FileCookieJar 类和其子类中的 save 方法, 加载就用 load 方法。

写脚本从指定网站抓取数据的时候, 免不了会被网站屏蔽 IP, 所以需要一些 IP 代理。可以发现我们需要的信息的页面 URL 有下面的规律: www.test.com/nn/+ 页码。如果直接通过 GET 方法访问, 则会出现 500 错误。在这个规律下的 URL 虽然都是通过 GET 方法获得数据的, 但都有 Cookie 认证, 另外还有反外链等。下面的例子用来获得 Cookie。

```

headers=[('User-Agent','Mozilla/6.0 (iPhone; CPU iPhone OS 8_0 like Mac
OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/8.0 Mobile/10A5376e
Safari/8536.25'),
        ('Host','www.test.com'),
        ('Referer','http://www.test.com/n')]
def getCookie()
    cookie=CookieJar()
    cookie_support= request.HTTPCookieProcessor(cookie)#Cookie处理器
    opener = request.build_opener(cookie_support)
    opener.addheaders=headers
    opener.open('http://www.test.com/')
    return cookie

```

有了 Cookie 就可以爬了，爬的内容怎么处理呢？介绍一个工具——Beautiful Soup。

Beautiful Soup

Beautiful Soup 翻译为鸡汤，现在的版本是 4.5.1，简称 BS4。提供一些简单的、Python 式的函数，提供导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。Beautiful Soup 自动将输入文档转换为 Unicode 编码，输出文档转换为 UTF-8 编码。你不需要考虑编码方式，除非文档没有指定一个编码方式，这时，Beautiful Soup 就不能自动识别编码方式了。然后，你仅仅需要说明一下原始编码方式就可以了。关于 BS 的用法官方文档介绍得很详细，下面爬取 <http://www.pythonscraping.com/pages/page3.html> 中的图片来小试牛刀。

```
import re
from urllib import request
from bs4 import BeautifulSoup

html=request.urlopen("http://www.pythonscraping.com/pages/page3.html")
bs=BeautifulSoup(html,"lxml")
#打印所有图片地址
for pic in bs.find_all('img',{'src':re.compile(".*\.jpg$")}):
    print(pic['src'])
```

结果：

```
../img/gifts/logo.jpg
../img/gifts/img1.jpg
../img/gifts/img2.jpg
../img/gifts/img3.jpg
../img/gifts/img4.jpg
../img/gifts/img6.jpg
```

接上文，我们把上面的高匿代理 IP “爬” 出来放到本地 proxy.txt 中。

```
cookie=getCookie()
# get the proxy
```

```

with open('proxy.txt', 'w') as f:
    for page in range(1,101):
        if page%50==0:#每50页更新cookie
            cookie=getCookie()

        url = 'http://www.test.com/nn/%s' %page
        cookie_support= request.HTTPCookieProcessor(cookie)
        opener = request.build_opener(cookie_support)
        request.install_opener(opener)

        req = request.Request(url,headers=dict(headers))
        content = request.urlopen(req)
        soup = BeautifulSoup(content,"lxml")
        trs = soup.find('table',id="ip_list").findAll('tr')
        for tr in trs[1:]:
            tds = tr.findAll('td')
            ip = tds[1].text.strip()
            port = tds[2].text.strip()
            protocol = tds[5].text.strip().
            f.write('%s://%s:%s\n' % (protocol, ip, port))

```

结果十五秒“爬”了1万条数据（具体数值与PC配置有关），说明1页正好100条，而总页数超过1000页，也就是记录数超过10万条。如果固定用同一个Cookie肯定不安全（谁会有空翻看1000页数据……），因此设置每爬50页更新Cookie。有了代理地址，不一定能保证有效，可能就被封杀了。因此使用思路是把代理地址存入哈希表，验证无效的地址则删除（看状态码），重新在表中取新的记录。代理地址使用方式如下：

```

...
proxy_handler = request.ProxyHandler({'http': '123.165.121.126:81'})
opener = request.build_opener(proxy_handler,cookie_handler ...各种其他
handle)
...

```

另外推荐个“神器”——crawlera，基本能满足各种需求。

假如真要“爬”1000页，需要花150秒。可以通过多进程或者异步处理实现。多进程很好做，手动维护一个HttpConnection的池，然后每次抓取时从连接池里面选连接即可（每秒

几百个连接，大部分服务器一定会封禁)。Python 的异步处理用到了 Twisted 库，却远没有同是异步模式的 Node.js 火，算是 Python 中的巨型框架了。

写爬虫还要考虑其他很多问题，比如授权验证、连接池、数据处理、JS 处理等，这里有个经典爬虫框架：Scrapy。目前支持 Python 3，支持分布式，使用 Twisted 来处理网络通信。架构清晰，并且包含了各种中间件接口，可以灵活地实现各种需求。

Scrapy 与 Pyspider

国内某“大神”开发了一个 WebUI 的 Pyspider，具有以下特性：

- (1) Python 脚本控制，可以用任何你喜欢的 HTML 解析包（内置 pyquery）。
- (2) Web 界面编写调试脚本，启停脚本，监控执行状态，查看活动历史，获取结果产出。
- (3) 支持 MySQL、MongoDB、SQLite。
- (4) 支持抓取 JavaScript 的页面。
- (5) 组件可替换，支持单机/分布式部署，支持 Docker 部署。
- (6) 强大的调度控制。

从内容上讲，两者具有的功能差不多，包括以上 3、5、6。不同的是 Scrapy 原生不支持 JS 渲染，需要单独下载 scrapy-splash，而 PyScrapy 内置支持 scrapyjs；PySpider 内置 pyquery 选择器，Scrapy 有 XPath 和 CSS 选择器，这两个大家可能更熟悉；此外，Scrapy 全部是命令行操作，Pyscrapy 有较好的 WebUI；还有，Scrapy 对千万级 URL 去重的支持很好，采用布隆过滤来做，而 Spider 用的是数据库来去重；最后，PySpider 更加容易调试，Scrapy 默认的 debug 模式信息量太大，warn 模式信息量太少，由于异步框架出错后是不会停掉其他任务的，也就是说程序出错了还会接着运行……从整体上来说，Pyspider 比 Scrapy 简单，并且 Pyspider 可以在线提供爬虫服务，也就是所说的 SaaS。想要做个简单的爬虫推荐使用 Pyspider，但自定义程度相对 Scrapy 较低，社区人数和文档都没有 Scrapy 强，但 Scrapy 要学习的相关知识也较多，故而完成一个爬虫的时间较长。

因为比较喜欢有完整文档的支持，所以后面主要用 Scrapy，简要说一下 Scrapy 的运行流程。

首先，引擎从调度器中取出一个链接（URL）用于接下来的抓取。

引擎把 URL 封装成一个请求（Request）传给下载器，下载器把资源下载下来，并封装成应答包（Response）。

然后，爬虫解析 Response。

- ◎ 若解析出实体 (Item), 则交给实体管道进行进一步的处理。
- ◎ 若解析出的是链接 (URL), 则把 URL 交给 Scheduler 等待抓取。

根据 Scrapy 文档描述, 要防止 Scrapy 被禁用, 主要有以下几个策略。

- (1) 动态设置 user agent。
- (2) 禁用 cookies。
- (3) 设置延迟下载。
- (4) 使用 Google cache。
- (5) 使用 IP 地址池 (Tor project、VPN 和代理 IP)。
- (6) 使用 Crawlera。

Google cache 在国内不可用, 其余的都可以利用。下面主要从动态随机设置 user agent、禁用 cookies、设置延迟下载和使用代理 IP 这几个方式入手。

自定义中间件

Scrapy 下载器是通过中间件控制的, 要实现代理 IP、user agent 切换, 可以自定义一个中间件。在项目下创建好项目后, 在里面找到 setting.py 文件, 先把 agents 和代理 IP 放到 setting.py 中 (代理 IP 较少情况下这样做, 较多的话还是放到数据库中, 方便管理), 设置中间件名字 MyCustomSpiderMiddleware 和优先级。

```

USER_AGENTS = [
    "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; AcooBrowser
    ; .NET CLR 1.1.4322; .NET CLR 2.0.50727)",
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; Acoo Browser;
    SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506)
    ",
    "Mozilla/4.0 (compatible; MSIE 7.0; AOL 9.5; AOLBuild 4337.35;
    Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)",
    "Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)",
    "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
    Trident/5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR
    2.0.50727; Media Center PC 6.0)",
    "Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0;
    WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .
    NET CLR 3.0.30729; .NET CLR 1.0.3705; .NET CLR 1.1.4322)",

```

```

"Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.2; .NET CLR
1.1.4322; .NET CLR 2.0.50727; InfoPath.2; .NET CLR 3.0.04506.30)",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/523.15
(KHTML, like Gecko, Safari/419.3) Arora/0.3 (Change: 287 c9dfb30)",
"Mozilla/5.0 (X11; U; Linux; en-US) AppleWebKit/527+ (KHTML, like
Gecko, Safari/419.3) Arora/0.6",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.2pre)
Gecko/20070215 K-Ninja/2.1.1",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9) Gecko
/20080705 Firefox/3.0 Kapiko/3.0",
"Mozilla/5.0 (X11; Linux i686; U;) Gecko/20070322 Kazehakase/0.4.5",
"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.8) Gecko Fedora
/1.9.0.8-1.fc10 Kazehakase/0.5.6",
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like
Gecko) Chrome/17.0.963.56 Safari/535.11",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/535.20 (
KHTML, like Gecko) Chrome/19.0.1036.7 Safari/535.20",
"Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; fr) Presto/2.9.168
Version/11.52",
]
PROXIES = [
    {'ip_port': '111.11.228.75:80', 'user_pass': ''},
    {'ip_port': '120.198.243.22:80', 'user_pass': ''},
    {'ip_port': '111.8.60.9:8123', 'user_pass': ''},
    {'ip_port': '101.71.27.120:80', 'user_pass': ''},
    {'ip_port': '122.96.59.104:80', 'user_pass': ''},
    {'ip_port': '122.224.249.122:8088', 'user_pass': ''},
]
# 禁用cookie (enabled by default)
COOKIES_ENABLED = False

# 设置下载延迟
DOWNLOAD_DELAY = 1

# 下载中间件
# See http://scrapy.readthedocs.org/en/latest/topics/downloader-
middleware.html

```

```
DOWNLOADER_MIDDLEWARES = {
    'weiboZ.middlewares.MyCustomDownloaderMiddleware': 543,
}
```

middlewares/MyCustomDownloaderMiddleware.py 如下:

```
import random
import base64
from settings import PROXIES
class RandomUserAgent(object):
    """Randomly rotate user agents based on a list of predefined ones"""
    def __init__(self, agents):
        self.agents = agents
    @classmethod
    def from_crawler(cls, crawler):
        return cls(crawler.settings.getlist('USER_AGENTS'))
    def process_request(self, request, spider):
        #随机选一个agent
        request.headers.setdefault('User-Agent', random.choice(self.
            agents))
class ProxyMiddleware(object):
    def process_request(self, request, spider):
        proxy = random.choice(PROXIES)
        if proxy['user_pass'] is not None:
            request.meta['proxy'] = "http://%s" % proxy['ip_port']
            encoded_user_pass = base64.encodestring(proxy['user_pass'])
            request.headers['Proxy-Authorization'] = 'Basic ' +
                encoded_user_pass
        else:
            request.meta['proxy'] = "http://%s" % proxy['ip_port']
```

15.2.2 数据提取

选择需要的数据

并不是所有 JSON 字段的数据都有用，这里只选取有用的字段，总的原则是按需抽取。可以看一下项目中定义的 Items.py。

微博内容 id 对应字段放入数据库中将有唯一约束，防止重复微博。选择 mblogid 作为唯一 id，而千万不能是 itemid。经测试发现 itemid 只代表当天微博的槽位，比如限制浏览 10 条数据，就有 1 ~ 10 个槽位，而 itemid 就代表这 10 个槽位标签，并不代表微博内容 id。另外 mblog 字段下还有个 id 属性，估计和 mblogid 是一样的效果，有兴趣的读者可以试试。发布时间代表信息的实效，JSON 里面由两个字段表示，一个是时间戳 created_timestamp，另一个是显示出来的真实时间数据。这里取真实数据，方便直接提取显示，但后期存储的时候需要统一转换为标准时间格式。评论数、转发数、点赞数和时效结合可以用来综合评估微博信息价值（时间越靠后这三个数字越能评价信息价值）。用户名、粉丝数、说说数可以用来检验用户是否是有价值用户，或者是机器人。后期处理需要提取求/租信息的关键词，包含价格、几号线、行政区划、信息是求租还是出租。

项目中定义的 pipelines.py 文件是 Scrapy 管道处理类，也就是主要的后期数据处理类。其中一个是 JsonPipeline 类，直接将数据打印到 JSON 文件中，这个前期可以用来调试爬虫效果。另一个是 MongoPipeline 类，用来保存后期处理后的数据。在 setting 文件中 ITEM_PIPELINES 属性可以设置具体采用哪个管道处理类。

后期处理的主要任务是提取关键字，如何从微博信息中爬取地理位置、价格等重要信息，这里采用双数组 Trie 树。

DAT

Trie 树是搜索树的一种，来自英文单词“Retrieval”的简写，可以建立有效的数据检索组织结构，是中文匹配分词算法中“词典”的一种常见实现手段。它本质上是一个确定的有限状态自动机（DFA），每个节点代表自动机的一个状态。在词典中，这个状态包括“词前缀”，“已成词”等。

采用 Trie 树搜索最多经过 n 次匹配即可完成一次查找（即最坏是 $O(n)$ ），而与词库中词条的数目无关。缺点是空间空闲率高，它是中文匹配分词算法中词典的一种常见实现。

双数组 Trie（doublearrayTrie，DAT）是 Trie 树的一个简单而有效的实现（日本人发明的），由两个整数数组构成，一个是 base[]，另一个是 check[]。双数组 Trie 树是 Trie 树的一种变形，是在保证 Trie 树检索速度的前提下，提高空间利用率而提出的一种数据结构。其本质是一个确定有限状态自动机（DeterministicFiniteAutomaton，DFA），每个节点代表自动

机的一个状态，根据变量的不同，进行状态转移。当到达结束状态或者无法转移时即完成查询。DAT 采用两个线性数组（base 和 check）对 Trie 树进行保存，base 和 check 数组拥有一致的下标，即 DFA 中的每一个状态（Trie 树中所说的节点）。base 数组用于确定状态的转移，check 数组用于检验转移的正确性，检验该状态是否存在。

在比较用于正向最大匹配分词的速度方面，DAT 分词的平均速度为 936KB/s（2006 年）。项目用到了 GitHub 上一个日本人的 Python 版的 DAT，其查询速度可以达到 2.755MB/s，查询速度和分词速度基本是差不多的，这三倍的差距应该是做了优化的。

词典的收集比较麻烦，因为没有现成的。项目中搜集了上海地铁、街道、行政区、乡镇等信息，其中价格信息范围是从 600 到 9000，可识别二千、二千二、两千一等中文价格。后面在微博上还看到有人用 1.2k 做价格的，暂时没加入，读者自己可以加入词条后重新运行 makeData.py 文件即可收录。

判断信息是租房还是求房也是根据关键字，当信息中出现【“求租”，“想租”，“求到”，“求从”，“要租”，“寻租”，“寻找”，“找新房子”，“找房子”，“找房”，“寻房”，“求房”，“想找”，“希望房”】的信息就标注为求房，否则标注为租房。

此外项目还收集了三千多个楼盘信息，由于有些楼盘信息容易混淆真实语境，比如“黄兴”、“金铭”与人名冲突等。有想根据楼盘查询信息的读者可以把 makeData.py 中第 5、51 行注释取消运行这个文件。

关于时间处理，微博挖到的时间有几种类型：

- ◎ 2016 年 01 月 01 日 00 点 00 分；
- ◎ 1 月 1 日 00: 00；
- ◎ 今天 00: 00；
- ◎ 1 分钟前/11 分钟前；
- ◎ 10 秒前。

使用 DataUtil 类进行统一转化。其中 MongoDB 使用的是 ISO 时间，比北京时间早 8 小时，而 pymongo 中的 datetime.datetime 数据并不会按时区处理，因此手动减少 8 小时后存储。同样从 MongoDB 中取出的时间要转化为当地时间。

```
> d=new Date()
> d
ISODate("2016-10-29T06:59:49.461Z")
> d.toLocaleDateString()
```

10/29/2016

15.2.3 数据存储

其实就这点数据放哪个数据库中都无所谓，假如这个数据量很大，就要好好考虑数据存储了。选择 Oracle、MySQL 还是 NoSQL。

数据库的比较就好比 Java、C#、Python、Go 等的骂战一样，没有最好的，只有最适合场景的。以下仅从 7 个角度进行比较。

- (1) 功能：Oracle>MySQL> NoSQL。
- (2) 写性能：NoSQL>Oracle≈MySQL。
- (3) 简单查询：Oracle>MySQL>NoSQL。
- (4) 复杂查询（含 join）：Oracle>MySQL>NoSQL。
- (5) 架构扩展：NoSQL>MySQL>Oracle。
- (6) 可维护性：Oracle>MySQL>NoSQL。
- (7) 成本：Oracle>MySQL≈NoSQL。

对于现在这个场景：爬虫在前端爬取数据，管道层处理数据后写数据，而这些数据又具有时效性，也就是说只会去读一部分数据，相对来说对写的要求较高。此外，这个场景就一个表，不涉及多表关联、约束等，复杂查询可以说没有，需要的功能较少。另外，网络数据不能保证一致性和可靠性，只要保证高可用性（HA）即可，NoSQL 可以设置副本机制达到高可用性，MySQL 虽然也可以做到但成本稍高。因此这个场景最适合的是 NoSQL。

HBase 还是 MongoDB

HBase 和 MongoDB 性能比较：MongoDB 适合做读写分离场景中的读取场景，并且它是用 JS 开发的，对 JSON 插入支持特别好。

这个业务场景是为了加快查询，需要对价格、行政区、发布时间创建索引。其中价格、行政区由于是数组形式，所以是多键索引，索引属性是稀疏的，即不允许空值。此外给这条微博的 mblog_id 加一个唯一索引。索引在初始运行时创建，之后除非手动删除数据库后运行，否则不会再创建。

为保证每次插入的数据都是最新的，插入前应比较数据的发布时间与数据库中的最新时间，如果是早的说明已经爬过了，则不需要插入。

15.2.4 项目运行与分析

运行项目

将项目 Git 到本地后，先确保以下软件在环境中已经安装。

- ◎ scrapy: <https://scrapy.org/>。
- ◎ datrie: <https://github.com/pytries/datrie>。
- ◎ pymongo: <https://github.com/mongodb/mongo-python-driver>。
- ◎ mongoDB: <https://www.mongodb.com/>。

执行下面的命令：

```
mongod
cd weiboSA
scrapy crawl mblogSpider
```

可选参数：

```
scrapy crawl mblogSpider -a num= -a new_url=
```

- ◎ num 代表爬取页面数，默认为 100 页，目前只支持 100 页。
- ◎ newurl 默认为搜索移动端‘上海租房’返回的 JSON 文件 URL，如果要添加其他上海租房信息，比如浦东租房，则在 Chrome 中找到请求的 JSON 地址，例如，http://m.weibo.cn/page/pageJson?containerid=&containerid=100103type%3D1%26q%3D浦东租房&type=all&queryVal=浦东租房&luicode=10000011&lfid=100103type%3D%26q%3D上海无中介租房&title=浦东租房&v_p=11&ext=&fid=100103type%3D1%26q%3D浦东租房&uicode=10000011&next_cursor=&page=。如果要数据库收录‘浦东租房’历史记录信息，将 pipelines.py 第 87、88 行注释掉。一般如果有‘上海租房’就不要去搜索‘浦东租房’，因为基本上有‘浦东租房’的微博都会有 @ 上海租房，所以下面会出现插入重复记录的错误。

```
weiboZ git:(master) scrapy crawl mblogSpider -a num=10 -a new_url="
http://m.weibo.cn/page/pageJson?containerid=\&containerid=100103\%3D1
\%26q\%3D\%E6\%B5\%A6\%E4\%B8\%9C\%E7\%A7\%9F\%E6\%88\%BF\&type=all\&
queryVal=\%E6\%B5\%A6\%E4\%B8\%9C\%E7\%A7\%9F\%E6\%88\%BF\&luicode
\=10000011\&lfid=100103type\%3D\%26q\%3D\%E4\%B8\%8A\%E6\%B5\%B7\%E6
\%97\%A0\%E4\%B8\%AD\%E4\%BB\%8B\%E7\%A7\%9F\%E6\%88\%BF\&title=\%E6\%
B5\%A6\%E4\%B8\%9C\%E7\%A7\%9F\%E6\%88\%BF\&v_p=11\&ext=\&fid=100103
type\%3D1\%26q\%3D\%E6\%B5\%A6\%E4\%B8\%9C\%E7\%A7\%9F\%E6\%88\%BF\&
uicode\=10000011\&next_cursor=\&page=\"
```

```
2016-10-29 14:41:11 [root] WARNING: 生成MongoPipeline对象
2016-10-29 14:41:11 [root] WARNING: 开始Spider
2016-10-29 14:41:11 [root] WARNING: 允许插入数据的时间大于2016-10-29
14:15:05.875000
2016-10-29 14:41:13 [root] WARNING: do page1.
2016-10-29 14:41:13 [root] WARNING: do other pages.
2016-10-29 14:41:13 [root] ERROR: 编号为:E91f233Ds的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef4ri5bC6的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef3UNqMmV的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef3stkA8a的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef3pzmJ6i的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef10BtvQr的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:Ef03Lj54z的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:EeYLU2GQd的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:EeYlBv7bn的数据插入异常
2016-10-29 14:41:13 [root] ERROR: 编号为:EeXkop2vu的数据插入异常
2016-10-29 14:41:15 [root] WARNING: 结束spider
```

更改日志显示级别要在 setting.py 中修改 LOG_LEVEL，建议采用项目默认的 WARNING，否则信息会很多。

查询示例

查询当前时区的 2016-10-20 至今有在 9 号线附近租房且房租不高于 2000 的信息。

```

db.house.find(
{
    created_at:{$gt:new Date('2016-10-20T00:00:00')},
    $or:
        [
            {price:{$lte:2000}},
            {price:[]}
        ],
    admin:'9号线',
    tag:true
},
{
    _id:0,
    text:1,
    created_at:1,
    scheme:1
}
).hint('created_at_-1').pretty()

{
    "text" : "房子在大上海国际花园，漕宝路1555弄，距9号线合川路地铁站步行5分钟，距徐家汇站只有4站，现在转租大床，有独立卫生间，公共厨房，房租2400，平摊下来1200，有一女室友，室友宜家上班，限女生，没有物业费，包网络，水电自理@上海租房无中介 @上海租房无中介 @上海租房 @上海租房无中介联盟",
    "scheme" : "http://m.weibo.cn/1641537045/EetVm3WBV?",
    "created_at" : ISODate("2016-10-25T09:18:00Z")
}

{
    "text" : "#上海租房##上海出租#9号线松江泗泾地铁站金地自在城，12层，步行、公交或小区班车直达地铁站。精装，品牌家具家电，主卧1800RMB/月；公寓门禁出入，房东直租，电话：xxxxxxxxxx，或QQ：xxxxxxxxx。@上海租房 @互助租房 @房天下上海租房 @上海租房无中介 @应届毕业生上海租房",
    "scheme" : "http://m.weibo.cn/1641537045/Een8cAoy8?",
    "created_at" : ISODate("2016-10-24T16:00:00Z")
}

```

```

{
  "text" : "#上海租房# 个人离开上海：转租地铁9号线朝南主卧带大阳台，离
地铁站两分钟！设备齐全，交通方便，随时入住。具体信息看图片~@上海租
房 @上海租房无中介联盟 @魔都租房 帮转谢谢！",
  "scheme" : "http://m.weibo.cn/1641537045/EdRpfuKuH?",
  "created_at" : ISODate("2016-10-21T07:14:00Z")
}
{
  "text" : "9号线桂林路 离地铁站8分钟 招女生室友哦 @上海租房 @上海租房
无中介联盟 上海·南京西路",
  "scheme" : "http://m.weibo.cn/1641537045/EdJ2U8Kv3?",
  "created_at" : ISODate("2016-10-20T09:57:00Z")
}

```

Python 的第三方 requests 库使用起来比自带的 urllib 更容易，是对 urllib 的进一步封装，读者可以自己尝试，这里不再举例。

附录 A Python 与其他语言调用

C# 调用 Python 脚本并使用 Python 的第三方模块, 详见 <https://chaolongzhang.github.io/2015/dotnet-call-python/>

```
var engine = IronPython.Hosting.Python.CreateEngine();
var scope = engine.CreateScope();
var source = engine.CreateScriptSourceFromFile("hello.py");
source.Execute(scope);
var say_hello = scope.GetVariable<Func<object>>("say_hello");
say_hello();
var get_text = scope.GetVariable<Func<object>>("get_text");
var text = get_text().ToString();
Console.WriteLine(text);
var add = scope.GetVariable<Func<object, object, object>>("add");
var result1 = add(1, 2);
Console.WriteLine(result1);
var result2 = add("hello ", "world");
Console.WriteLine(result2);
```

PHP 调用 Python 脚本并使用 Python 的第三方模块, 详见 <http://www.cnblogs.com/baiboy/p/check.html>:

```
<?php
$name=mb_convert_encoding($_POST['projectname'], "GBK", "UTF-8");
$sumb=mb_convert_encoding($_POST['projectsumb'], "GBK", "UTF-8");
$program="D:/Users/Administrator/Anaconda3/python E:/pythonSource/
CheckArticle/CheckRepeat/checkIndex.py
        ".$name." ".$sumb."; #注意使用绝对路径.$name." ".$sumb
$output = nl2br(shell_exec($program));
// $program1="D:/Users/Administrator/Anaconda3/python ../CheckRepeat
/test.py
        ".$name." ".$sumb."; #注意使用绝对路径.$name." ".$sumb
// $result1 = shell_exec($program1);
echo mb_convert_encoding ($output, "UTF-8", "GBK");
```

```
// if ($output!=null){
//     print_r(nl2br(file_get_contents('../CheckRepeat/database/
DealCorpus/checkout.txt')));
// }
?>
```

Python 脚本调用 Java 并使用其第三方模块, 详见 <http://www.cnblogs.com/baiboy/p/7676236.html>:

```
# -*- coding:utf-8 -*-
# Filename: main.py

from jpye import *

startJVM(getDefaultJVMPath(), "-Djava.class.path=C:\\hanlp\\hanlp-1.3.2.
jar;C:\\hanlp",
        "-Xms1g", "-Xmx1g") # 启动JVM, Linux需替换分号;为冒号:

print("="*30+"HanLP分词"+"="*30)
HanLP = JClass('com.hankcs.hanlp.HanLP')
# 中文分词
print(HanLP.segment('你好, 欢迎在Python中调用HanLP的API'))
print("-"*70)
```

附录 B Git 项目上传简易教程

Linux 在 1991 年创建了开源的 Linux，从此，Linux 系统不断发展，已经成为最大的服务器系统软件了。Linux 的代码是如何管理的呢？Linux 自己用 C 写了一个分布式版本控制系统，这就是 Git！一个月之内，Linux 系统的源码已经由 Git 管理了！Git 迅速成为最流行的分布式版本控制系统，尤其是 2008 年，GitHub 网站上线了，它为开源项目免费提供 Git 存储，无数开源项目开始迁移至 GitHub，包括 jQuery、PHP、Ruby 等。

- (1) Windows 版 Git 下载地址为 <https://git-scm.com/downloads>。安装完成后，在开始菜单里找到 Git→Git Bash，说明 Git 安装成功。
- (2) Git 安装完成后，配置你的名字和 E-mail 地址。

注意：git config 命令的 `--global` 参数，表示本机所有的 Git 仓库都会使用这个配置，也可以对某个仓库指定不同的用户名和 E-mail 地址。

创建版本库

- (1) 创建一个版本库非常简单，首先创建一个空目录：

```
$ cd /d //指定的个人盘符
```

```
$ mkdir learngit //创建版本库根目录
```

```
$ cd learngit //进入版本库目录(tab键盘补全命令)
```

```
$ pwd //查看当前路径
```

```
/d/learngit
```

- (2) 通过 git init 命令把这个目录变成 Git 可以管理的仓库：\$ git init。
- (3) 在 learngit 下创建一个 readme.txt 文件并编写两句话。

```
$ touch readme.txt
```

```
$ vi readme.txt //进入编辑器，按i进入编辑模式，esc退出：wq强制保存
```

```
Git is a version control system.
```

```
Git is free software.
```

```
$cat readme.txt //查看信息
```

- (4) 用命令 `git add` 告诉 Git，把文件添加到仓库：

```
$ git add readme.txt
```

- (5) 用命令 `git commit` 告诉 Git，把文件提交到仓库，`-m` 后面输入的是本次提交的说明：

```
$ git commit -m "wrote a readme file"
```

- (6) 读者先自己注册一个 GitHub 账号。然后即可查看上传文件的详细信息。

GitHub 是版本控制和协作代码托管平台，它可以让你和其他人的项目从任何地方合作。相对于 CVS 和 SVN 的联网限制和传速慢有明显的优势。因此，GitHub 越来越受企业和个人的青睐。在 GitHub 上进行项目管理也是趋势。详细操作教程请访问作者主页：<http://www.cnblogs.com/baiboy/p/github.html#top>。

参考文献

- [1] 刘开瑛, 郭炳炎. 自然语言处理 [M]. 科学出版社, 1991.
- [2] 统计自然语言处理基础, Christopher.Manning 等著, 宛春法等译.
- [3] 冯志伟. 现代语言学流派 [M]. 陕西人民出版社, 1999.
- [4] 冯志伟. 计算语言学基础 [M]. 商务印书馆, 2001.
- [5] 冯志伟. 数理语言学 [M]. 知识出版社, 1985.
- [6] 宗庆成, 吴华, 黄泰翼, 等. 限定领域汉语口语对话语料分析 [J]. 计算语言学论文集, 清华大学出版社, 1999.
- [7] 李航. 统计学习方法 [J]. 清华大学出版社, 北京, 2012.
- [8] 数据挖掘概念与技术 (364-386) 韩家炜.
- [9] 冯志伟. 自然语言处理简明教程 [M]. 上海外语教育出版社, 2012.
- [10] 冯志伟. 机器翻译今昔谈 [M] 北京: 语文出版社, 2007.
- [11] 冯志伟. 当前自然语言处理发展的四个特点 [J]. 暨南大学华文学院学报. 2006 (1) .
- [12] 吴军. 数学之美 [M]. 人民邮电出版社, 2012.
- [13] 白宁超, 唐聃, 王亚强. 基于主动学习的传统中医症状本体构建方法研究综述 [J]. 电子技术与软件工程, 2016 (7): 162-163.
- [14] 舒红平, 游志胜, 蒋建民. 基于信息熵的决策属性分类挖掘算法及应用 [D].
- [15] Russell, Matthew A. Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More. O'Reilly Media, Inc. 2013.
- [16] Csardi, Gabor. "Eigenvalues and eigenvectors of the adjacency matrix of a graph" .
- [17] Barthélemy, M. "Betweenness centrality in large complex networks." Physics 38.2 (2004): 163-168.
- [18] Russell, Matthew A. Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More. O'Reilly Media, Inc. 2013.
- [19] 王思力, 张华平, 王斌. 双数组 Trie 树算法优化及其应用研究 [J]. 中文信息学报, 2006, 20 (5): 24—30.
- [20] 梁锐, 朱清新, 廖淑娇, 等. 基于多特征融合的深度视频自然语言描述方法 [J]. 计算机应用, 2017, 37 (4): 1179-1184.
- [21] Souter C. Natural language identification using corpus-based models [J]. HERMES-Journal of Language and Communication in Business, 2017, 7 (13): 183-203.

- [22] Botha J A, Pitler E, Ma J, et al. Natural language processing with small feed-forward networks[J]. arXiv preprint arXiv: 1708.00214,, 2017.
- [23] Sarfjoo S S, Demiroglu C, King S, et al.Using eigenvoices and nearest-neighbors in HMM-based cross-lingual speaker adaptation with limited data[J]. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2017,25(4): 839-851.
- [24] Corbett-Detig R, Nielsen R. A hidden Markov model approach for simultaneously estimating local ancestry and admixture time using next generation sequence data in samples of arbitrary ploidy[J]. PLoS genetics, 2017,13 (1): e1006529.
- [25] Mohamed S, Mahmoud T, Ibrahim M. Efficient Edge Detection Technique Based on Hidden Markov Model using Canny Operator[J]. threshold, 2017, 6 (01) .
- [26] Jablonowski K. Hidden Markov Models for Protein Domain Homology Identification and Analysis[J]. SH2 Domains: Methods and Protocols, 2017: 47-58.
- [27] Lapuyade-Lahorgue J, Xue J H, Ruan S. Segmenting Multi-Source images using hidden Markov fields with copula-based multivariate statistical distributions[J]. IEEE Transactions on Image Processing, 2017.
- [28] Fiecas M,Franke J, von Sachs R,et al. Shrinkage estimation for multivariate hidden Markov models[J]. Journal of the American Statistical Association,2017, 112 (517): 424-435.
- [29] DeRuiter S L, Langrock R, Skirbutas T, et al. A multivariate mixed hidden Markov model for blue whale behaviour and responses to sound exposure[J]. The Annals of Applied Statistics,2017, 11 (1): 362-392.
- [30] Cowen L L E, Besbeas P, Morgan B J T, et al. Hidden Markov models for extended batch data[J]. Biometrics, 2017.
- [31] Gu W,Lv Z,Hao M. Change detection method for remote sensing images based on an improved Markov random field[J]. Multimedia Tools and Applications,2017, 76 (17): 17719-17734.
- [32] Messing R O, Pomrenze M B, De Guglielmo G, et al. A distributed CRF network in rat extended amygdala regulates anxiety and excessive alcohol drinking[J]. Alcohol, 2017, 60: 212.
- [33] Miczek K A. CRF modulation in VTA-DRN escalates alcohol intake after social stress[J]. Alcohol,2017, 60: 214-215.
- [34] 邬伦, 刘磊, 李浩然, 等. 基于条件随机场的中文地名识别方法 [J]. 武汉大学学报·信息科学版, 2017, 42 (2): 150-156.
- [35] 万业号, 刘利军, 黄青松. 基于层叠条件随机场的中文医疗机构名识别 [J]. 济南大学学报: 自然科学版, 2017 (1): 61-66. [36] 李志义, 王冕, 赵鹏武. 基于条件随机场模型的“评价特征-评价词”对抽取研究 [J]. 情报学报, 2017, 36 (4): 411-421.
- [37] 黄念娥, 黄河, 王儒敬. 本体与条件随机场结合的涉农商品名称抽取与类别标注 [J]. 计算机应用, 2017, 37 (1): 233-238.

- [38] 翟菊叶, 陈春燕, 张钰, 等. 基于 CRF 与规则相结合的中文电子病历命名实体识别研究 [J]. 包头医学院学报, 2017, 11: 052.
- [39] 吴俊峰, 姜志国, 张浩鹏, 等. 半监督条件随机场的高光谱遥感图像分类 [J]. 2017.
- [40] 王茵, 周学广, 陆健. 基于条件随机场的中文情感分析方法比较研究 [J]. 计算机与数字工程, 2017, 45 (9): 1703-1707.
- [41] 丁洁, 赵景惠. 基于 N—gram 模型的中文分词算法的研究 [J]. 福建电脑, 2017, 33 (5): 110-110.
- [42] 邓丽萍, 罗智勇. 基于半监督 CRF 的跨领域中文分词 [J]. 中文信息学报, 2017, 31 (4): 9-19.
- [43] 李雪莲, 段鸿, 许牧. 基于 GRU 神经网络的中文分词法 [J]. 2017.
- [44] 周俊, 郑中华, 张伟. 基于改进最大匹配算法的中文分词粗分方法 [J]. 计算机工程与应用, 2014 (2): 124-128.
- [45] 梁喜涛, 顾磊. 中文分词与词性标注研究 [J]. 计算机技术与发展, 2015 (2015 年 02): 175-180.
- [46] 杜丽萍, 李晓戈, 于根, 等. 基于互信息改进算法的新词发现对中文分词系统改进 [J]. 北京大学学报 (自然科学版), 2016, 52 (1): 35-40.
- [47] Natural Language Toolkit.
- [48] 赵铁军. 将语言计算的理论方法和最新成果呈现给读者-《统计自然语言处理》评述 [J]. 自动化学报, 2014, 40 (5): 1024-1024.
- [49] 李泽魁, 赵妍妍, 秦兵, 等. 中文微博情感倾向性分析特征工程 [J]. 山西大学学报: 自然科学版, 2014, 37 (4): 570-579.
- [50] 秦兵, 刘安安, 刘挺. 无指导的中文开放式实体关系抽取 [J]. 计算机研究与发展, 2015, 52 (5): 1029-1035.
- [51] 刘雄, 张宇, 张伟男, 等. 基于依存句法分析的复合事实型问句分解方法 [J]. 中文信息学报, 2017, 31 (3): 140-146.
- [52] 刘明杨, 秦兵, 刘挺. 基于文采特征的高考作文自动评分 [J]. 智能计算机与应用, 2016, 1: 001.
- [53] 王巍, 赵铁军, 辛国栋, 等. 基于条件随机域模型的比较要素抽取研究 [J]. 自动化学报, 2015, 41 (8): 1385-1393.
- [54] 赵宇婧, 许鑫泽, 朱齐丹, 等. 一种自然语言关键词的人机交互方法 [J]. 2016.
- [55] 王哲, 严璘璘, 孙宇浩, 等. 功能短句的语义成分对两可自然物体分类的影响 [J]. 科学通报, 2017, 5: 007. [56] Chen Y, Zhou G, He S, et al. The CASIA Entity linking System at TAC 2013[J]. 2014.
- [57] 徐浩广, 王宁, 刘佳明, 等. 基于自然语言检索的综合相似度计算算法 [J]. 计算机系统应用, 2017, 26 (6): 170-175.
- [58] Ling Z, Chunxiang X, Chengzhi Z, et al. User Tags and Microblog Posts: Case Study of Sina Weibo[J]. Data Analysis and Knowledge Discovery, 2016, 32(3): 18-24.
- [59] 邓宇昂. 计算机语言发展探析 [J]. 电子世界, 2017 (14): 80-80.

- [60] 郝丹. 结合 NLP 技术的汉语学习系统设计与实现 [D]. 华中师范大学, 2015.
- [61] 曾文, 张均胜, 徐红姣, 等. 多语言科技语料库建设研究 [J]. 数字图书馆论坛, 2015 (8): 43-47.
- [62] 肖忠华. 英汉翻译中的汉语译文语料库研究 [J]. 当代外语研究, 2016, 1: 001.
- [63] 王克非, 秦洪武. 论平行语料库在翻译教学中的应用 [J]. 外语教学与研究, 2015, 47 (5): 763-772.
- [64] Aijmer K, Altenberg B. English corpus linguistics[M]. Routledge, 2014.
- [65] Kennedy G. An introduction to corpus linguistics[M]. Routledge, 2014.
- [66] Chen W, Chen N F, Lim B P, et al. Corpus-based pronunciation variation rule analysis for singapore English[C]//SLaTE. 2015: 35-40.
- [67] Laviosa S. The English Comparable Corpus[J]. Unity in diversity: Current trends in Translation Studies, 2016: 101.
- [68] Talbert R. Inverting the linear algebra classroom[J]. Primus, 2014, 24(5): 361-374.
- [69] Eddelbuettel D, Sanderson C. RcppArmadillo: Accelerating R with high-performance C++ linear algebra[J]. Computational Statistics & Data Analysis, 2014, 71: 1054-1063.
- [70] Williams G. Linear algebra with applications[M]. Jones & Bartlett Learning, 2017.
- [71] Pugachev V S. Probability theory and mathematical statistics for engineers[M]. Elsevier, 2014.
- [72] Feller W. On the Central Limit Theorem of Probability Theory[M]//Selected Papers I. Springer International Publishing, 2015: 207-244.
- [73] Durlauf S N. Lecture Notes 1: Probability Theory[J]. 2015.
- [74] Cressie N. Statistics for spatial data[M]. John Wiley & Sons, 2015.
- [75] Gravetter F J, Wallnau L B. Statistics for the behavioral sciences[M]. Cengage Learning, 2016.
- [76] 谢庆华, 张宁蓉, 宋以胜, 等. 聚类数据挖掘可视化模型方法与技术 [J]. 解放军理工大学学报: 自然科学版, 2015, 16 (1): 7-15.
- [77] 马昱欣, 曹震东, 陈为. 可视化驱动的交互式数据挖掘方法综述 [J]. 计算机辅助设计与图形学学报, 2016, 28 (1): 1-8.
- [78] 杨彦波, 刘滨, 祁明月. 信息可视化研究综述 [J]. 河北科技大学学报, 2014, 35 (1): 91-102.
- [79] 王子毅, 张春海. 基于 ECharts 的数据可视化分析组件设计实现 [J]. 微型机与应用, 2016, 35 (14): 46-48.
- [80] 姚霖, 刘轶, 李鑫鑫, 等. 词边界字向量的中文命名实体识别 [J]. 智能系统学报, 2016 (2016 年 01): 37-42.
- [81] 博客园-伏草惟存
- [82] HanLP 汉语言处理包
- [83] 11 款开放中文分词引擎评测
- [84] 百度 Echart 官网