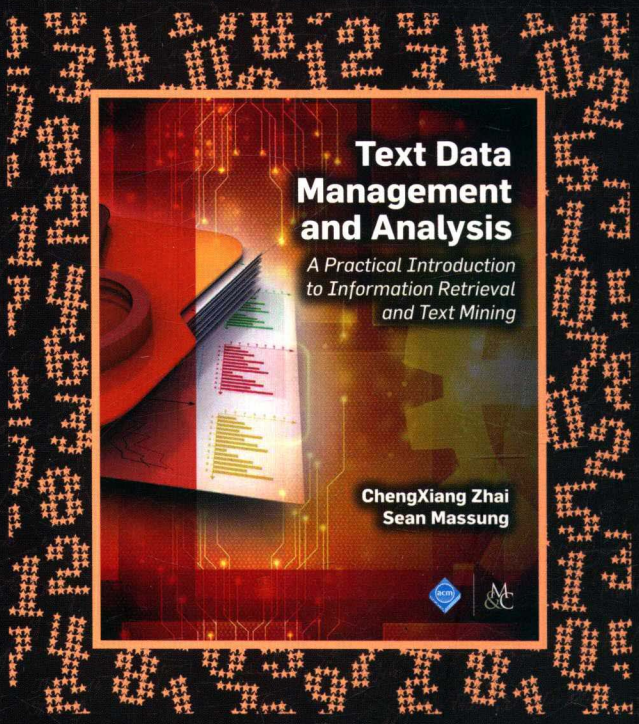
HZ BOOKS
华章IT

数据科学与工程技术丛书

文本数据管理与分析

信息检索与文本挖掘的实用导论

[美] 翟成祥（Chengxiang Zhai）
肖恩·马森（Sean Massung） 著
宋巍 赵鑫 李璐旸 李洋 等译
刘挺 审校



TEXT DATA MANAGEMENT AND ANALYSIS

A PRACTICAL INTRODUCTION TO INFORMATION RETRIEVAL AND TEXT MINING

机械工业出版社
China Machine Press

数据科学与工程技术丛书

TEXT DATA MANAGEMENT AND ANALYSIS

A PRACTICAL INTRODUCTION TO INFORMATION RETRIEVAL AND TEXT MINING

文本数据管理与分析

信息检索与文本挖掘的实用导论

[美] 翟成祥（Chengxiang Zhai）
肖恩·马森（Sean Massung） 著
宋巍 赵鑫 李璐旸 李洋 等译
刘挺 审校

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

文本数据管理与分析：信息检索与文本挖掘的实用导论 /（美）翟成祥（Chengxiang Zhai），（美）肖恩·马森（Sean Massung）著；宋巍等译 . —北京：机械工业出版社，2018.10
（数据科学与工程技术丛书）
书名原文：Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

ISBN 978-7-111-61176-9

I. 文… II. ①翟… ②肖… ③宋… III. 数据管理 IV. TP274

中国版本图书馆 CIP 数据核字（2018）第 240805 号

本书版权登记号：图字 01-2017-3413

Authorized translation from the English language edition, entitled Text Data Management and Analysis: a Practical Introduction to Information Retrieval and Text Mining, 9781970001167 by Chengxiang Zhai, Sean Massung, published by Morgan & Claypool Publishers, Inc., Copyright © 2016 by the Association for Computing Machinery and Morgan & Claypool Publishers.

Chinese language edition published by China Machine Press, Copyright © 2019.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Morgan & Claypool Publishers, Inc. and China Machine Press.

本书中文简体字版由美国摩根 & 克莱普尔出版公司授权机械工业出版社独家出版。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书从理论到实践，深入介绍了文本数据管理与分析中的关键问题，全面涵盖了自然语言理解、信息检索、推荐系统、机器学习、情感分析、文本摘要和主题分析等核心技术。

本书可作为高等院校计算机科学或相关专业本科生、研究生的教材，也可作为专业科研人员与工业界人士的参考书。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：朱秀英　　责任校对：殷 虹
印　刷：中国电影出版社印刷厂　　版　次：2019 年 4 月第 1 版第 1 次印刷
开　本：185mm × 260mm 1/16　　印　张：20.5
书　号：ISBN 978-7-111-61176-9　　定　价：139.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88379833　　投稿热线：（010）88379604
购书热线：（010）68326294　　读者信箱：hzjsj@hzbook.com

版权所有 · 侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

目前，我们正处在一个以大数据与人工智能技术为核心的新的工业革命时代，其主要特征是大量各种可利用的数据可以视为一种特殊的生产资料，经过高效的智能数据分析与挖掘以及机器学习等人工智能技术处理后，这些数据可以产生巨大价值，创造智能。

大数据可以用两种方式创造智能。其一，大量的数据可以作为训练数据，让监督式机器学习方法特别是深度学习，发挥巨大潜力，从大量数据中学得智能，从而使智能机器能够大量代替人力来完成各种任务(此类智能系统可称为自主型智能系统)。例如，大量的可用于训练无人驾驶车的数据可以很自然地从人的驾驶过程中通过传感器获得，使机器可以自动驾驶车辆；又如，大量的客户服务记录数据，可以用来训练客户服务机器人，自动回答客户的问题。其二，大量的数据可以作为对我们生活的世界的感知和观察的结果的描述，用数据挖掘或非监督式机器学习方法对数据加以处理，获得关于被观察系统的各种有用知识，从而拓展人类的感知能力，增强人的智能(此类系统自身往往智能程度不高，可以称为助理型智能系统)。例如，大量电子病历数据可以用来构造一个医生或病人的辅助诊疗的智能助手系统；又如，大量金融数据、社交媒体数据以及新闻数据可以用来构造金融方面的决策支持系统。

比较两类基于大数据的智能系统，自主型智能系统能完成的任务不能太复杂(因机器需独立完成任务)，且对数据的要求较高，需要有标注的数据，而获取极大量的高质量的标注数据在很多问题领域并不现实，所以这类应用目前只能在少量的特定应用领域起作用。而且，由于机器的智能主要来自于人工标注的数据，机器的智能不容易超越人的智能。相反，助理型智能系统由于不需要有标注的数据，任何数据都可以利用，所以在任何领域都可以起作用，有着非常广泛的应用。而且，有趣的是，尽管助理型智能系统本身的智能不高，甚至没有太多智能，但这样的系统一旦与人结合，人与系统相加以后的综合智能往往能大大超越人的智能。这种情形下，助理型智能系统的功能有与显微镜及望远镜的功能相似之处，即它们都可以拓展人对世界的感知能力，从而增强人的智能，特别是有助于在复杂应用领域优化决策。

作为一种特殊的大数据，文本数据泛指各种以自然语言形式存在的数据，包括万维网页、新闻报道、社交媒体、产品评论、科学文献、政府文件等；语音和视频数据，经语音识别后也能产生文本数据。文本数据有着极其广泛的应用。第一，文本数据可被视为人，作为一个富有智能的主观“传感器”所产生的数据，它可以与所有其它非文本数据相结合，共同支持助理型智能系统；又因为任何应用领域都会涉及相关的人群，人们会以各种形式产生可用的文本数据，所以文本数据在任何领域都会有应用价值。第二，由于人的主观性，文本数据富含关于人的观点、偏好以及需求等信息，所以特别有助于挖掘关于人的各种属性，使智能系统可以更好地理解用户，从而可以对每一个特定的用户进行优化服务(即个性化服务)。第三，由于文本数据是人们用自然语言交流和通信的产物，它的语义很丰富，相比非文本数据来说，文本数据更加直接地表达知识。从数据挖掘的角度看，更容易让计算机自动获取知识。然而，由于自然语言是为人类通信而设计的，需要有大量的常

识及推理能力，才能准确理解，所以尽管自然语言理解研究已取得很大进展，计算机目前还不能全面理解不受限的自然语言的结构和语义，所以在所有文本数据的应用中，必须充分利用人的自然语言理解能力，让计算机成为一个智能助理。

《文本数据管理与分析》这本书从文本检索与挖掘的角度，比较全面和系统地介绍了利用文本数据支持助理型智能系统的各种应用技术。文本检索技术可用于构造各种搜索和推荐系统；这类系统可帮助用户快速定位到与当前任务最相关的文本数据，从而避免处理大量不必要处理的不相关数据。而文本分析与挖掘技术则可进一步对相关文本数据进行分析，帮助用户消化相关文本信息，将文本转化为可直接支持任务和决策的知识。这些技术可以有机地结合在一起支持交互式的智能文本检索与分析系统。书中介绍的文本检索与分析技术大多为不依赖具体自然语言的基于统计的通用技术，因而可适用于管理、处理与分析包括英语及汉语的任何自然语言文本数据及构造各种应用领域的应用系统。与书中内容相配的 MeTA 工具集，可帮助读者进一步理解如何实现书中算法和利用已有算法进行科研或开发应用软件。

目前，大数据及人工智能技术的研究与开发在中国正方兴未艾，机械工业出版社华章公司出版《文本数据管理与分析》一书的中文版，可谓时机甚佳。作为原书的一位作者，在此特别感谢华章公司姚蕾编辑对此书翻译的大力支持和协调，以及哈尔滨工业大学刘挺教授和他带领的翻译团队，包括宋巍、赵鑫、李璐旸、李洋、赵森栋及段俊文等的杰出翻译工作。希望此书能有助于许多中文读者学习掌握文本数据检索、分析与挖掘的当前技术，开发基于文本大数据的智能信息应用系统。

翟成祥

2019 年 3 月 2 日

于美国伊利诺伊州香槟市

翟成祥教授是信息检索与数据挖掘领域世界知名学者，具有很高的学术影响力。他关于基于语言模型的信息检索以及概率主题模型相关的一系列研究在信息检索和数据挖掘领域写下了浓重一笔。从读研究生以来，我们一直深受翟老师研究的影响和启发。去年，刘挺教授问我们是否愿意翻译翟成祥老师 2016 年出版的这部著作，我们感到非常荣幸并欣然接受。

本书内容以文本数据处理为核心。文本是最常见也是最具表达能力的数据形式，以自然又自由的方式蕴含着丰富的人类智慧。近年来，文本数据呈现爆炸式增长的趋势，不仅包括传统的静态文本，还包括微博、评论等用户动态生成数据以及科技文献、教育、商务、政务、医疗等专业领域数据。在当前大数据与人工智能的背景下，迫切需要智能的文本数据处理系统来帮助人们及时有效地获取和分析文本中丰富而有价值的知识、信息以及人们的观点和偏好。

本书涵盖了实现以上文本信息系统所涉及的关键问题与核心技术，具有区别于以往类似书籍的独特之处。

首先，本书将信息获取和数据挖掘统一起来，强调了在任何实用的文本信息系统中信息检索与文本分析都是不可或缺且相互依赖的组成部分。信息检索有助于从大规模文本数据中获取与目标任务最相关的小规模数据，数据挖掘则通过对相关文本进行分析得到知识，信息消费者又通过信息检索来验证知识。全书内容也依此逻辑进行组织，全面介绍了信息检索和文本数据挖掘的主要概念、技术和思想。

其次，本书从新的视角看待文本数据，将其视为人类作为主观传感器对现实世界的表达，强调了人既是文本生产者又是信息消费者的角色。书中介绍了情感计算、主题分析、文本摘要等近年来的热点研究课题和相关技术，有利于分析人类主观观点和有效消化信息，也阐述了文本数据与结构化数据联合分析的可能性和典型方法，将人类主观传感器与其他客观传感器产生的数据结合起来，突出在不同上下文下的文本分析。

最后，本书结构清晰，可读性强并注重实践性。尽管涵盖了广泛的主题，但主题之间具有清晰的逻辑关联。每章都提供了具体实例，有助于读者理解，每章最后都列举了参考文献，使得读者能够根据实际控制知识的深度，适用于广泛的读者人群。本书提供了配套软件工具包，并可以与作者以往的在线课程配合，极大地方便了读者的理论学习与工程实践。教材、在线课程、配套软件，三位一体的课程设置对国内教学实践也有着积极的借鉴意义。

本书的翻译由哈尔滨工业大学刘挺教授组织并作为主审。四位主要译者均曾就读于哈尔滨工业大学社会计算与信息检索研究中心，目前在各个高校从事与本书内容密切相关的教学和科研工作。在翻译过程中，宋巍负责翻译了前言、第 1、6、7、9 章以及全书的统稿，赵鑫翻译了第 2、17 章以及附录，李璐旸翻译了第 3、5、8、10 章，李洋翻译了第

11、13、14、15章。哈尔滨工业大学社会计算与信息检索研究中心博士生赵森栋与段俊文同样承担了部分章节的翻译。赵森栋翻译了第4、19、20章，段俊文翻译了第12、16、18章。他们都曾访问伊利诺伊大学香槟分校得到翟成祥教授亲自指导。最后，特别感谢华章出版社姚蕾、朱秀英、唐晓琳等编辑在本书翻译过程当中提供的各种指导和帮助。

译者水平和经验有限，时间紧迫，书中难免存在理解不当和语言运用不妥之处，敬请读者批评指正。

译者

2018年8月

于北京

“大数据”的增长为计算和统计方法带来了前所未有的机遇。利用这些方法可以将未经处理的原始数据(raw data)转变为可应用的知识以支持各类应用任务，尤其是涉及决策(decision making)优化的应用领域，如健康与医疗、防护与安全、学习与教育、科学探索及商业智能等。正如显微镜使我们能够看到微观世界、望远镜使我们能够看到远方一样，“大数据镜”将扩展我们的感知能力去发现埋藏在数据中的隐含信息与知识，这些隐含的信息与知识能够帮助做出预测和优化决策。本书覆盖了用于管理和分析大规模文本数据的通用计算技术，这些技术能够帮助用户在各类应用中对文本数据进行管理和使用。

文本数据包括所有以自然语言文本(如英文文本或中文文本)形式出现的数据：所有的网页、社交媒体数据(如微博)、新闻、科技文献、电子邮件、政府文档以及其他类型的企业数据。文本数据在生活中起到重要作用。由于人们使用自然语言进行交流，每天将产生并消费大量涵盖各类主题的文本数据。文本数据的爆炸式增长使人们不可能至少很难及时地消费所有相关信息。因此，开发智能信息检索系统成为迫切的需求，智能信息检索系统能够帮助人们管理文本数据，并随时快速、准确地获取所需的相关信息。这种需求是近期网络搜索引擎行业迅猛发展的主要原因。文本数据主要是人们为了交流的目的而产生，其中通常包含了丰富的语义内容和有价值的知识、信息、观点以及人们的偏好。因此，作为“大数据”的一种，文本数据提供了巨大的机遇去发掘对多种应用有用的知识，尤其是直接用文本形式表达的用户观点和偏好等。例如，人们普遍通过产品评论、论坛讨论和社交媒体等文本数据获取他人观点帮助决策。由于信息规模巨大，人们需要智能软件工具来帮助发现相关知识以优化决策或更加高效地完成相关任务。尽管支持文本挖掘的技术并没有成熟到如搜索引擎支持文本获取的程度，但相关领域在近年来取得了显著进步，一些专门的文本挖掘工具已经开始在众多应用领域广泛使用。本书的子标题阐释了本书涵盖的两大主题：信息检索与文本挖掘。这两大主题基本对应了前文讨论的两类应用系统(搜索引擎与文本分析系统)所依赖的技术。人为地区分两大主题主要是为了构建本书的高层结构。事实上，复杂的应用系统都会使用两大主题中的多种技术。

文本与结构化数据不同。结构化数据遵循定义良好的结构模式，便于计算机来处理。文本缺少明显的结构，因此以上讨论的智能软件工具的发展需要计算机能够理解文本中包含的内容。当前的自然语言处理技术还没有做到使计算机能够准确地理解自然语言文本(这也是“人”应该被包含到处理流程之中的原因)。但是在过去几十年中，基于统计或启发式的文本数据管理与分析方法得到了广泛的发展。这些方法具有良好的健壮性，可以用于分析与管理任何自然语言以及关于任何主题的文本数据。本书试图对很多此类方法进行系统的介绍，并着重强调构建多种实用文本信息系统所需的最有用的知识与技巧。

本书主要基于作者在伊利诺伊大学香槟分校(University of Illinois at Urbana-Champaign，UIUC)讲授文本数据管理与分析相关课程(即 CS 410“文本信息系统”)，以及第一作者 2015 年在 Coursera 开设的两门在线课程“文本检索与搜索引擎”(Text Retrieval and Search Engines)和“文本挖掘与分析”(Text Mining and Analytics)所使用的材料。书中的大

部分内容及结构与上述两门在线课程一致，因此本书可作为两门在线课程的主要参考书。

信息检索(Information Retrieval，IR)是一个相对成熟的研究领域。关于信息检索的优秀教材也有很多，近期的著作包括 Baeza-Yates 和 Ribeiro-Neto [2011]所著的《Modern Information Retrieval：The Concepts and Technology behind Search》、Büttcher 等[2010]所著的《Information Retrieval：Implementing and Evaluating Search Engines》、Croft 等[2009]所著的《Search Engines：Information Retrieval in Practice》以及 Manning 等[2008]所著的《Introduction to Information Retrieval》。与关于信息检索的已有教材相比，本书涵盖了更广泛的主题，既包括信息检索也包括文本挖掘。我们希望能够描绘出一个可以指导如何逐步构建一个能同时支持信息检索与文本分析的系统的蓝图。例如，书中详细介绍了词关联挖掘、概率主题模型以及文本和非文本数据的联合分析，以往任何信息检索教材都没有包含这些内容。与信息检索相比，文本挖掘(Text Mining，TM)远远没有成熟，还处于幼年时期，甚至如何准确定义文本挖掘都是一个开放问题。就此而言，目前尚没有关于文本挖掘的教材。作为一本关于文本挖掘的教材，本书对文本挖掘中主要的代表性技术进行了基本介绍。通过将文本挖掘与信息检索置于统一的框架内，强调了在任何实用的文本信息系统中信息检索与文本挖掘相结合的重要性。信息检索在任何文本挖掘应用中都会起到两个重要作用：一是能够快速减小数据规模，通过过滤大量不相关文档而获取小规模的、与特定应用问题最为相关的数据；二是支持数据分析人员验证并解释从文本数据中发现的模式，这里需要数据分析人员借助搜索和浏览功能来获取和检查与目标模式最相关的数据。

本书的另外一个特点是带有配套的信息检索与文本挖掘工具集 MeTA(可在 https://meta-toolkit.org/获取)。它实现了书中讨论的很多技术。书中的很多练习也是根据该工具集设计的，可帮助读者获得使用书中习得的技术进行实践的技巧，并能够应用这些技巧去解决现实世界中的实际问题。

本书包括四部分。第一部分纵览全书内容并提供理解后续章节所必需的背景知识。第二部分和第三部分包括本书的主要内容，广泛地涵盖了信息检索相关技术(称为文本数据获取技术)和文本挖掘相关技术(称为文本数据分析技术)。第四部分通过一个文本管理与分析的统一框架总结了全书内容。该框架整合了许多信息检索和文本分析技术，通过人机协同为文本数据获取与分析提供更为先进的支持。

本书尽可能地做到自包含，因此本书内容在理解上所需的背景知识是极少的。但是读者需要具有关于计算机科学的基本知识，尤其是数据结构、编程语言，并了解概率和统计中的基本概念，如条件概率和参数估计。不具备以上背景的读者应该也可以理解本书讨论的大多数算法的基本思想，或者可以通过仔细研读本书第 2 章来获取相关知识。如果必要，可以进一步阅读第 2 章书目说明部分提到的参考文献来深入理解涉及的主要概念。MeTA 工具集可以很方便地被用于实验算法和建立应用，但是若希望扩展该工具集则需要具备基本的 C++编程能力。

本书可作为高年级本科生信息检索与文本挖掘相关课程的教材，或者作为研究生的参考书，书中涵盖了信息检索和文本挖掘中的实践环节，也可以帮助工业界的从业者获取关于管理和分析文本数据的实用技术并快速应用到各种有趣的真实世界的应用中。

致谢

本书的完成是众多人帮助的结果。首先，我们要向 Edward A. Fox 表达最诚挚的谢意。他是 ACM 系列丛书“信息检索和数字图书馆”领域的主编，感谢他邀请我们写这本书。还要感谢 ACM 丛书的主编 Tamer Ozsu，感谢他对本书提案的支持和有益的评价。没有他们的鼓励和支持，这本书是无法完成的。接下来，我们深深地感激 Edward A. Fox、Donna Harman、Bing Liu 和 Jimmy Lin 对本书初稿进行了十分仔细的审阅并提供了有价值的反馈和建设性的意见。尽管我们无法完全实现提出的建议，然而他们所有的评论极有帮助，在很多方面显著提高了本书的品质。本书显然会有不足或错误之处，作者将完全为此负责。

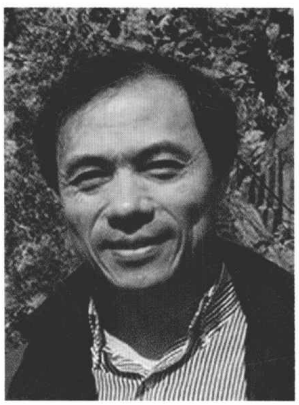
在本书的写作过程中，Morgan & Claypool 出版社的执行编辑 Diane Cerra 给予我们强有力的支持和帮助，她的定期提醒与及时支持是我们能够按时完成本书的关键，我们对此表示衷心的感谢。此外，我们要感谢负责文章编修的 Sara Kreisman，以及 Paul C. Anagnostopoulos 和他在 Windfall Software 的制作团队（Ted Laux、Laurel Muller、MaryEllen Oliver 和 Jacqui Scarlott），感谢他们在索引、插图、美术校对和构图方面的帮助，确保本书能够顺利出版。

本书的内容和对相关主题的理解受益于同研究界和工业界众多同仁的讨论和交流。由于篇幅所限，我们在这里只能提及部分人（对未提及的人表示歉意）：James Allan, Charu Aggarwal, Ricardo Baeza-Yates, Nicholas J. Belkin, Andrei Broder, Jamie Callan, Jaime Carbonell, Kevin C. Chang, Yi Chang, Charlie Clarke, Fabio Crestani, W. Bruce Croft, Maarten de Rijke, Arjen de Vries, Daniel Diermeier, AnHai Doan, Susan Dumais, David A. Evans, Edward A. Fox, Ophir Frieder, Norbert Fuhr, Evgeniy Gabrilovich, C. Lee Giles, David Gross man, Jiawei Han, Donna Harman, Marti Hearst, Jimmy Huang, Rong Jin, Thorsten Joachims, Paul Kantor, David Karger, Diane Kelly, Ravi Kumar, Oren Kurland, John Lafferty, Victor Lavrenko, Lillian Lee, David Lewis, Jimmy Lin, Bing Liu, Wei-Ying Ma, Christopher Manning, Gary Marchionini, Andrew McCallum, Alistair Moffat, Jian-Yun Nie, Douglas Oard, Dragomir R. Radev, Prabhakar Raghavan, Stephen Robertson, Roni Rosenfeld, Dan Roth, Mark Sanderson, Bruce Schatz, Fabrizio Sebastiani, Amit Singhal, Keith van Rijsbergen, Luo Si, Noah Smith, Padhraic Smyth, Andrew Tomkins, Ellen Voorhees, Yiming Yang, Yi Zhang, Justin Zobel。我们要感谢他们对本书间接的贡献。本书中的部分材料，特别是第 19 章，基于第一作者在伊利诺伊大学香槟分校文本信息管理与分析（TIMAN）小组指导的众多博士生的工作。感谢他们所有人，包括：Tao Tao, Hui Fang, Xuehua Shen, Azadeh Shakery, Jing Jiang, Qiaozhu Mei, Xuanhui Wang, Bin Tan, Xu Ling, Younhee Ko, Alexander Kotov, Yue Lu, Maryam Karimzadehgan, Yuanhua Lv, Duo Zhang, V. G. Vinod Vydiswaran, Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, Huizhong Duan, Yanen Li, Hongning Wang, Mingjie Qian 和 Dae Hoon Park。本书作者自身的研究工作得到包括 NSF、NIH、NASA、IARPA、Air Force、ONR、DHS、Alfred P. Sloan 基金会的支持以及许多公司如 Microsoft、Google、IBM、Yahoo!、LinkedIn、Intel、HP 和 TCL 的支持。在此表示感谢。

第一作者 2015 年在伊利诺伊大学香槟分校开设的两门在线课程(即“文本检索与搜索引擎”和“文本挖掘与分析”)是本书的根本基础，书中很多部分的内容来自两门课程的转录笔记。所以，我们要感谢所有帮助建设这两门课程的人，特别是助教 Hussein Hazimeh 和 Alex Morales，学校教学支持人员 Jason Mock、Shannon Bicknell、Katie Woodruff 和 Edward Noel Dignan，以及计算机系主任 Rob Rutenbar，他的鼓励、支持和帮助是两门课程成功建设的关键。第一作者还要感谢伊利诺伊大学香槟分校允许他使用 2015 年秋天的学术休假来完成此书。特别感谢 MeTA 的共同创立者 Chase Geigle。此外，第二作者感谢 Chase Geigle、Jason Cho、Urvashi Khandelwal (以及其他很多人)与其进行深入的讨论和给予的鼓励。

最后，我们要感谢所有的家庭成员，特别是我们的妻子 Mei 和 Kai，感谢她们的爱与支持。第一作者还要感谢他的哥哥 Chengxing 在研究讨论中持续提供的智慧激励以及父母培养他乐于学习和分享知识的热忱。

Chengxiang Zhai
Sean Massung
2016 年 6 月



翟成祥(**Chengxiang Zhai**) 伊利诺伊大学香槟分校计算机科学系以及图书馆与信息科学研究生院、基因生物学研究所和统计系教授、Willett 学者。他于 1990 年获得南京大学计算机科学专业博士学位，2002 年获得卡内基梅隆大学语言与信息技术博士学位。1997～2000 年，他就职于 Clairvoyance 公司，历任研究科学家、高级研究科学家。他的研究兴趣包括信息检索、文本挖掘、自然语言处理、机器学习、生物医学与健康信息学以及智能教育信息系统。他已经在主流会议与期刊发表超过 300 篇研究论文。他现在是《ACM Transactions on Knowledge Discovery from Data》副主编，曾任《Information Processing and Management》副主编、《ACM Transactions on Information Systems》副主编以及《Information Retrieval Journal》编委。他担任多个国际会议的程序委员会主席，包括 ACM CIKM 2004、NAACL HLT 2007、ACM SIGIR 2009、ECIR 2014、ICTIR 2015、WWW 2015，并担任 ACM CIKM 2016、ACM WSDM 2018 大会主席。他是 ACM 会士、ACM 杰出科学家并荣获多项荣誉，包括 ACM SIGIR 2004 最佳论文奖、ACM SIGIR 2014 时间考验奖、Alfred P. Sloan 研究奖金、IBM 教师奖、HP 创新研究项目奖、微软超越搜索研究奖以及美国青年科学家和工程师总统奖。



肖恩·马森(**Sean Massung**) 伊利诺伊大学香槟分校计算机科学专业博士生，此前他在此分别获得学士和硕士学位。他是 MeTA 的联合发明者并在其所有研究中使用 MeTA。他是多门课程的指导者，包括 CS 225“数据结构与编程原则”、CS 410“文本信息系统”以及CS 591txt“文本挖掘研讨”。他位列 2014 年伊利诺伊香槟分校优秀教师名单，获得杰出助教奖以及计算机科学杰出研究项目奖，并在 Jump 实验室以及 UIUC 数据和信息系统研讨会、大数据研讨会和助教研讨会进行多次演讲。他的研究兴趣包括信息检索中的文本挖掘应用、自然语言处理和教育。

目 录

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

中文版序
译者序
前言
作者简介

第一部分 概述和背景

第 1 章 绪论 …… 2
1.1 文本信息系统的功能 …… 4
1.2 文本信息系统的概念框架 …… 5
1.3 本书结构安排 …… 7
1.4 如何使用本书 …… 8
书目说明和延伸阅读 …… 9

第 2 章 背景 …… 11
2.1 概率和统计基础 …… 11
2.1.1 联合概率和条件概率 …… 12
2.1.2 贝叶斯法则 …… 13
2.1.3 抛硬币和二项分布 …… 14
2.1.4 最大似然参数估计 …… 14
2.1.5 贝叶斯参数估计 …… 15
2.1.6 概率模型及其应用 …… 16
2.2 信息论 …… 17
2.3 机器学习 …… 19
书目说明和延伸阅读 …… 20
练习 …… 20

第 3 章 文本数据理解 …… 22
3.1 自然语言处理的历史和研究现状 …… 23
3.2 自然语言处理和文本信息系统 …… 24
3.3 文本表示 …… 26
3.4 统计语言模型 …… 28
书目说明和延伸阅读 …… 31
练习 …… 31

第 4 章 MeTA：一个面向文本数据管理和分析的统一工具箱 … 33
4.1 设计原则 …… 33
4.2 设置 MeTA …… 34
4.3 架构 …… 34
4.4 用 MeTA 分词 …… 35
4.5 相关工具箱 …… 37
练习 …… 38

第二部分 文本数据获取

第 5 章 文本数据获取概述 …… 44
5.1 获取模式：拉取与推送 …… 44
5.2 多模式互动获取 …… 45
5.3 文本检索 …… 47
5.4 文本检索与数据库检索 …… 48
5.5 文档选择与文档排序 …… 49
书目说明和延伸阅读 …… 50
练习 …… 51

第 6 章 检索模型 …… 52
6.1 概述 …… 52
6.2 检索函数的一般形式 …… 53
6.3 向量空间检索模型 …… 54
6.3.1 向量空间模型实例化 …… 55
6.3.2 位向量表示的表现 …… 56
6.3.3 改进的模型实例 …… 57
6.3.4 TF 变换 …… 60
6.3.5 文档长度规范化 …… 62
6.3.6 基本向量空间模型的进一步改进 …… 64

6.3.7 小结 …… 65
6.4 概率检索模型 …… 65
6.4.1 查询似然检索模型 …… 67
6.4.2 文档语言模型的平滑 …… 69
6.4.3 具体的平滑方法 …… 72
书目说明和延伸阅读 …… 76
练习 …… 76

第7章 反馈 …… 78
7.1 向量空间模型中的反馈 …… 79
7.2 语言模型中的反馈 …… 81
书目说明和延伸阅读 …… 84
练习 …… 84

第8章 搜索引擎实现 …… 86
8.1 分词器 …… 86
8.2 索引器 …… 87
8.3 打分器 …… 90
8.3.1 逐个词项排序 …… 90
8.3.2 逐个文档排序 …… 90
8.3.3 过滤文档 …… 91
8.3.4 索引分片 …… 91
8.4 反馈实现 …… 92
8.5 压缩 …… 92
8.5.1 按位压缩 …… 93
8.5.2 块压缩 …… 94
8.6 高速缓存 …… 95
8.6.1 LRU 缓存 …… 95
8.6.2 DBLRU 缓存 …… 96
书目说明和延伸阅读 …… 96
练习 …… 97

第9章 搜索引擎评价 …… 98
9.1 引言 …… 98
9.1.1 要度量什么 …… 98
9.1.2 Cranfield 评价方法 …… 98
9.2 集合检索的评价 …… 100
9.2.1 准确率和召回率 …… 100
9.2.2 F 度量：准确率和召回率的结合 …… 101
9.3 有序列表的评价 …… 102
9.4 基于多级别判断标准的评价 …… 106
9.5 评价中的实际问题 …… 107
书目说明和延伸阅读 …… 110
练习 …… 110

第10章 网络搜索 …… 112
10.1 网络爬虫 …… 113
10.2 网页索引 …… 113
10.3 链接分析 …… 117
10.3.1 PageRank 算法 …… 118
10.3.2 HITS 算法 …… 121
10.4 排序学习 …… 122
10.5 网络搜索的未来 …… 125
书目说明和延伸阅读 …… 127
练习 …… 127

第11章 推荐系统 …… 130
11.1 基于内容的推荐 …… 131
11.2 协同过滤 …… 134
11.3 推荐系统的评价 …… 137
书目说明和延伸阅读 …… 138
练习 …… 138

第三部分 文本数据分析

第12章 文本数据分析概述 …… 142
12.1 动机：文本数据分析的应用 …… 142
12.2 文本与非文本数据：人类作为主观传感器 …… 143
12.3 文本挖掘任务概览 …… 145

第13章 词关联挖掘 …… 148
13.1 词关联挖掘的基本思想 …… 149
13.2 聚合关系的发现 …… 150

13.3 组合关系的发现 …… 153
13.4 词关联挖掘的评价 …… 159
书目说明和延伸阅读 …… 160
练习 …… 160

第 14 章 文本聚类 …… 162

14.1 聚类技术概述 …… 163
14.2 文档聚类 …… 164
14.2.1 凝聚层次聚类法 …… 165
14.2.2 K-均值 …… 165
14.3 词项聚类 …… 167
14.3.1 语义关联的词语 …… 167
14.3.2 点互信息 …… 169
14.3.3 先进方法 …… 169
14.4 文本聚类的评价 …… 172
书目说明和延伸阅读 …… 173
练习 …… 173

第 15 章 文本分类 …… 175

15.1 引言 …… 175
15.2 文本分类方法概述 …… 176
15.3 文本分类问题 …… 177
15.4 文本分类的特征 …… 177
15.5 分类算法 …… 179
15.5.1 k-近邻 …… 180
15.5.2 朴素贝叶斯 …… 181
15.5.3 线性分类器 …… 182
15.6 文本分类的评价 …… 183
书目说明和延伸阅读 …… 184
练习 …… 184

第 16 章 文本摘要 …… 185

16.1 文本摘要技术概述 …… 185
16.2 抽取式文本摘要 …… 186
16.3 抽象式文本摘要 …… 187
16.4 文本摘要的评价 …… 189
16.5 文本摘要的应用 …… 189
书目说明和延伸阅读 …… 190
练习 …… 190

第 17 章 主题分析 …… 192

17.1 用词项表示的主题 …… 193
17.2 用单词分布表示的主题 …… 196
17.3 挖掘文本中的一个主题 …… 198
17.3.1 最简单的主题模型：一元语言模型 …… 199
17.3.2 添加背景语言模型 …… 201
17.3.3 混合模型的参数估计 …… 205
17.3.4 混合模型的行为 …… 206
17.3.5 期望最大化 …… 209
17.4 概率潜在语义分析 …… 214
17.5 PLSA 的扩展及潜在狄利克雷分布 …… 220
17.6 主题分析的评价 …… 223
17.7 主题模型的总结 …… 224
书目说明和延伸阅读 …… 224
练习 …… 225

第 18 章 观点挖掘与情感分析 …… 226

18.1 情感分类 …… 228
18.2 有序回归 …… 230
18.3 潜在方面评分分析 …… 232
18.4 观点挖掘与情感分析的评价 …… 238
书目说明和延伸阅读 …… 238
练习 …… 238

第 19 章 文本与结构化数据的联合分析 …… 240

19.1 引言 …… 240
19.2 上下文文本挖掘 …… 242
19.3 上下文概率潜在语义分析 …… 244
19.4 以社交网络作为上下文的主题分析 …… 249
19.5 以时间序列作为上下文的主题分析 …… 252
19.6 小结 …… 256
书目说明和延伸阅读 …… 256

练习 …… 257

第四部分 统一的文本数据管理和分析系统

第 20 章 面向一个统一的文本管理和分析系统 …… 260

20.1 文本分析操作 …… 262

20.2 系统架构 …… 264

20.3 MeTA 作为一个统一系统 …… 265

附录 A 贝叶斯统计 …… 266

附录 B 期望最大化 …… 271

附录 C KL-散度和狄利克雷先验平滑 …… 275

参考文献 …… 277

索引 …… 287

第一部分

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

概述和背景

第 1 章　绪论
第 2 章　背景
第 3 章　文本数据理解
第 4 章　MeTA：一个面向文本数据管理和分析的统一工具箱

第1章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

绪　论

在过去的20年里，我们经历了在线信息的爆炸性增长。根据加利福尼亚大学伯克利分校2003年的一项研究："……世界每年产生1～2EB(exabyte，艾字节)的不同信息，这对于地球上的男人、女人和孩子来说，每人大约250MB(megabyte，兆字节)。各类印刷文档仅占总量的0.03%。"[Lyman等2003]

大量的在线信息是文本信息(即自然语言文本)。例如，根据上面引用的伯克利的研究："报纸每年发表25TB(terabyte，太字节或称兆兆字节)内容，杂志发表10TB内容……办公文档包含195TB内容。据估计，每年发送的电子邮件总数达到6100亿封，包含11 000TB信息。"当然，还有博客文章、论坛帖子、推文、科技文献以及政府文件等。Roe[2012]将电子邮件数量从2003年的6100亿封更新为2010年的107万亿封。根据IDC最近的一份报告[Gantz和Reinsel 2012]，2005～2020年，数字宇宙将增长300倍，规模达130EB～40000EB。

一般来说，各种类型的在线信息都是有用的，但由于以下原因，文本信息起着特别重要的作用，可以说是最有用的一种信息。

文本(自然语言)是人类知识最自然的编码方式。因此，大多数人类的知识都是以文本数据的形式编码的。例如，科学知识几乎都整理在科学文献中，而技术手册包含如何操作设备的详细说明。

文本是人们遇到的最常见的信息类型。事实上，一个人每天产生和消费的大部分信息都是文本形式的。

文本是最具表达能力的信息形式。它可以用来描述其他媒体，如视频或图像，甚至图像搜索引擎(如Google和Bing支持的图像搜索引擎)也经常依靠匹配图像周围的文本来检索"匹配"用户关键字查询的图像。

网络文本信息的爆炸式增长强烈需要能够提供以下两种相关服务的智能化软件工具，帮助人们管理和利用文本大数据。

文本检索。文本数据的增长使得人们无法及时消费数据。由于文本数据对我们积累的大部分知识进行了编码，因此通常不会被丢弃，从而导致大量文献数据的积累，这些文献数据现在超出了任何个人能够处理的能力范围，即便只是简单浏览。在线文本信息的快速增长也意味着没有人能够消化每天产生的所有新信息。因此，迫切需要开发智能文本检索系统，以帮助人们快速、准确地获取所需的相关信息。这种需求促进了近期网络搜索行业的迅猛发展。事实上，像Google和Bing这样的网络搜索引擎现在已经成为我们日常生活中不可或缺的一部分，每天都有数以百万计的查询。通常，在大量文本数据存在的地方，搜索引擎都是有用的(诸如桌面搜索、企业搜索或特定领域中的文献搜索，例如PubMed)。

文本挖掘。文本数据是人类为了交流而产生的，所以它们通常含有丰富的语义内容，并且通常包含有价值的知识、信息、观点和个人的喜好。它们提供了很多机会来发掘对于许多应用有用的各种知识，特别是关于人类意见和偏好的知识。这些知识通常直接在文本

数据中表达。例如，现在人们习惯于通过产品评论、论坛讨论和社交媒体文本等包含主观见解的文本数据来获取有关他们感兴趣的话题的观点，并优化各种决策任务，如购买产品或选择一项服务。同样，由于信息的巨大规模，人们需要智能的软件工具来帮助发现相关知识以优化决策或帮助他们更有效地完成任务。尽管支持文本挖掘的技术还没有像支持文本获取的搜索引擎那么成熟，但近年来在这方面已经取得了显著的进展，专业的文本挖掘工具已经在许多应用领域得到了广泛使用。

4

结构化数据采用定义良好的模式，使计算机处理起来相对容易，与结构化数据相比，文本没有明显的结构，所以计算机在上述智能软件工具开发过程中需要处理和理解文本编码的内容。目前的自然语言处理技术还没有达到使计算机能够精确理解自然语言文本的水平(这也是人类往往应该参与到处理过程的主要原因)，但是采用许多不同的统计和启发式方法来管理和分析文本数据已经在过去的几十年中得到了发展。它们通常非常健壮，可以用于对任何自然语言、任何主题的文本数据进行分析和管理。本书旨在对其中的一些方法进行系统介绍，重点介绍构建各种实际有用的文本信息系统所需的最有用的知识和技能。

上面讨论的两种服务(即文本检索和文本挖掘)在概念上对应于分析任何“大规模文本数据”过程中的两个自然步骤，如图 1-1 所示。尽管原始文本数据可能很大，但是具体的应用通常只需要少量最相关的文本数据，因此在概念上，任何应用的第一步应该是根据具体的应用或者决策去识别相关的文本数据，避免对大量不相关文本数据做不必要的处理。将原始大规模文本数据转换成规模更小但高度相关的文本数据的第一步通常是在用户帮助下利用文本检索技术来完成(例如，用户可以使用多个查询来收集所有相关文本数据以用于决策问题)。在这第一步中，主要目标是将用户(或应用程序)与最相关的文本数据连接起来。

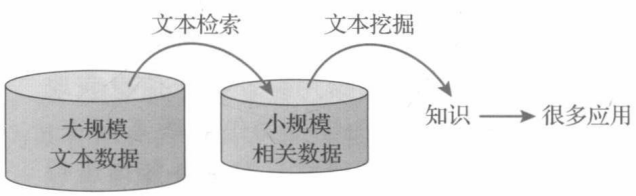


图 1-1　文本检索与数据挖掘是分析大规模文本数据的两项主要技术

5

一旦获得一个小规模的最相关文本数据，我们需要对文本做进一步的分析来帮助用户消化文本数据中的知识和模式。这是文本挖掘的一个步骤，其目标是从文本数据中进一步发现知识和模式，以支持用户的任务。此外，需要对任何发现的知识的可信度进行评估，所以用户通常需要返回到原始的文本数据中去获得用来解释所获得知识的上下文，并通过上下文信息验证知识的可信度。因此，作为主要用于文本获取的搜索引擎系统，也必须在任何基于文本的决策支持系统中提供知识来源。因此，这两个步骤在概念上是交错的，一个完整的智能文本信息系统必须在一个统一的框架中进行整合。

值得指出的是，在“大数据”的背景下，文本数据与其他类型的数据是非常不同的，因为它通常是由人类直接生成的，通常也意味着要被人类消费。相反，其他数据往往是机器生成的(例如通过使用各种物理传感器收集的数据)。由于人类可以比计算机更好地理解文本数据，所以人类参与挖掘和分析文本数据的过程绝对是至关重要的(比其他大数据应用程序更为必要)。如何最佳地将人与机器之间的工作分开从而优化人与机器之间的协作，以最少的人力来最大化其“智能组合”，是所有文本数据管理和分析应用中的一个普遍挑战。以上讨论的两个步骤可以被认为是文本信息系统协助人类的两种不同的方式：信息检索系统帮助用户从大量的文本数据中找到解决具体应用问题所需的最相关文本数据，从而有效地将大规模原始文本数据转换成可以被人类更容易处理的规模较小的相关文本数据；

而文本挖掘应用系统可以帮助用户分析文本数据中的模式，以提取和发现对于完成任务或进行决策直接有用的、可操作的知识，从而为用户提供更直接的任务支持。

从这个角度，我们将本书所涵盖的技术分成两部分来匹配图 1-1 所示的两个步骤，然后再用一章讨论如何将所有的技术整合到统一的文本信息系统中。本书试图从实践的角度全面介绍信息检索和文本数据挖掘的主要概念、技术和思想。其中包括许多亲身实践的练
6 习，使用配套软件工具 MeTA 来帮助读者学习如何将信息检索和文本挖掘技术应用于真实世界的文本数据，并学习如何实验和改进一些有趣的应用任务的算法。本书可作为计算机科学方向的本科生和研究生以及图书馆和信息科学工作者的教材，也可作为从事分析和管理文本数据等相关工作的技术人员的参考书。

1.1　文本信息系统的功能

从用户的角度来看，文本信息系统(TIS)可以提供三种不同但相关的功能，如图 1-2 所示。

信息获取(information access)。这种能力使用户可以在需要时获取有用的信息。有了这个能力，文本信息系统可以在正确的时间连接正确的信息和正确的用户。例如，搜索引擎使得用户能够通过查询来获取文本信息，而推荐系统可以在发现可用的新信息项目时将相关信息推送给用户。由于信息获取
7 的主要目的是将用户与相关信息联系起来，提供这种能力的系统通常只对文本数据进行最小限度的分析，只需满足将相关信息与用户的信息需求匹配，而原始信息项目(例如，网页)通常以其原始形式交付给用户，但是也经常提供项目的摘要。从文本分析的角度来看，用户通常需要阅读信息项目来进一步消化和利用所传递的信息。

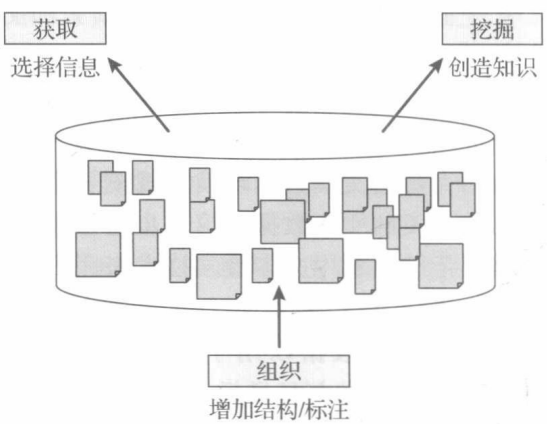


图 1-2　信息获取、知识获取和文本组织是文本信息系统的三个主要功能，文本组织对信息获取和知识获取起到支撑作用，而知识获取也常被称为数据挖掘

知识获取(knowledge acquisition)或文本分析(text analysis)。这种能力使得用户能够获得文本数据中蕴含的有用知识。如果没有对大规模的数据进行合成和分析，用户通常不容易获得这些知识。文本信息系统可以分析大量的文本数据以发现文本中隐藏的有趣模式。具有知识获取能力的文本信息系统可以被称为分析引擎。例如，搜索引擎可以将产品的相关评论返回给用户，分析引擎可以使用户直接获得关于产品的主要的正面或负面意见，并比较人们对多个类似产品的意见。提供知识获取能力的系统通常需要更详细地分析文本数据，综合来自多个文本文档的信息，发现有趣的模式，创造新的信息或知识。

文本组织(text organization)。此能力使系统能够用有意义的(主题)结构来注释一组文本文档，从而可以连接分散的信息，使用户可以根据该结构在信息空间中巡览。虽然这样的结构可以被认为是从文本数据中获得的“知识”，并且直接对用户有用，但是通常它们仅用于促进信息获取或知识获取，或者兼而有之。从这个意义上说，文本组织的能力在文本信息系统中起到了支持作用，使得信息获取和知识获取更加有效。例如，添加的结构可以允许用户使用结构上的约束进行搜索，或者根据结构进行浏览。考虑到结构的约束，结

构也可以用来进行详细的分析。

信息获取可以进一步分为两种模式：拉取和推送。在拉取模式下，用户主动从系统中“拉”出有用的信息，在这种情况下，系统是被动的，等待用户提出请求，然后系统用相关信息回应。当用户具有临时信息需求(即临时需要关于产品的意见)时，这种信息获取模式通常非常有用。例如，像 Google 这样的搜索引擎通常为用户提供拉取模式信息获取。 8
在推送模式下，系统主动向用户“推”(推荐)它认为对用户有用的信息。当用户具有相对稳定的信息需求(例如，一个人的爱好)时，推送模式常常工作良好；在这种情况下，系统可以“预先”知道用户的偏好和兴趣，从而能够向用户推荐信息而不需要用户采取主动。本书涵盖了两种信息获取模式。

拉取模式还包括两种互补的方式让用户获得相关信息：查询和浏览。在查询的情况下，用户通过(关键字)查询指定信息需求，系统将该查询作为输入并返回估计与查询相关的文档。在浏览的情况下，用户简单地沿着将信息项目链接在一起的结构进行巡览并逐渐地获得相关信息。由于查询也可以被看作是一步即导航到一组相关文档，很显然，浏览和查询可以自然地交织。事实上，网络搜索引擎的用户通常交错进行查询和浏览。

从文本数据中获取知识通常是通过文本挖掘过程来实现的。文本挖掘可以被定义为挖掘文本数据以发现有用的知识。数据挖掘社区和自然语言处理(Natural Language Processing，NLP)社区都开发了文本挖掘的方法，但两个社区对这个问题的看法往往略有不同。从数据挖掘的角度来看，我们可能将文本挖掘视为挖掘一种特殊的数据，即文本。遵循数据挖掘的总体目标，文本挖掘的目标自然会被视为发现和提取文本数据中的有趣模式，其中可能包括潜在主题、主题趋势或异常值。从 NLP 的角度来看，文本挖掘可以被看作是理解自然语言文本的一部分，将文本转化为某种形式的知识表示，并基于提取的知识进行有限的推理。因此，一个主要的任务是执行信息抽取(information extraction)，它的目标是识别和提取所涉及的各种实体(例如人员、组织和位置)及其关系(例如谁与谁见面)。实际上，任何文本挖掘应用都可能涉及模式发现(即数据挖掘角度)和信息抽取(即 NLP 角度)。信息抽取丰富了文本的语义表示，使得模式发现算法能够生成语义上更有意义的模 9
式，而不是直接处理文本的字或字符串表示。由于我们的重点是介绍可以用于各种文本数据的具有通用性和健壮性的技术，这些技术不需要太多的人力；而信息抽取技术更具有语言相关性，并且一般需要很多人工参与，所以我们在本书中普遍采用数据挖掘的角度。然而需要强调的是，信息抽取是任何文本信息系统的重要组成部分，它试图支持更深入的知识发现或语义分析。

文本挖掘的应用可以被分类为直接应用和间接应用。直接应用中被发现的知识将被用户直接消费，而间接应用中发现的知识不一定直接对用户有用，但可以通过提供更好的支持间接地帮助用户获取信息。知识获取也可以基于发现了什么知识来进一步分类。然而，由于“知识”涉及的范围广泛，所以不可能使用少量的类别来覆盖所有的形式。尽管如此，我们仍然可以在本书中找出几个常见的类别。例如，可以发现的一种知识类型是一组隐藏在文本数据中的主题或子主题，它们可以作为文本数据中主要内容的简明摘要。另一种可以从用户生成的主观性文本中获得的知识是关于某个主题的观点的总体情感极性。

1.2 文本信息系统的概念框架

从概念上讲，文本信息系统可能由几个模块组成，如图 1-3 所示。

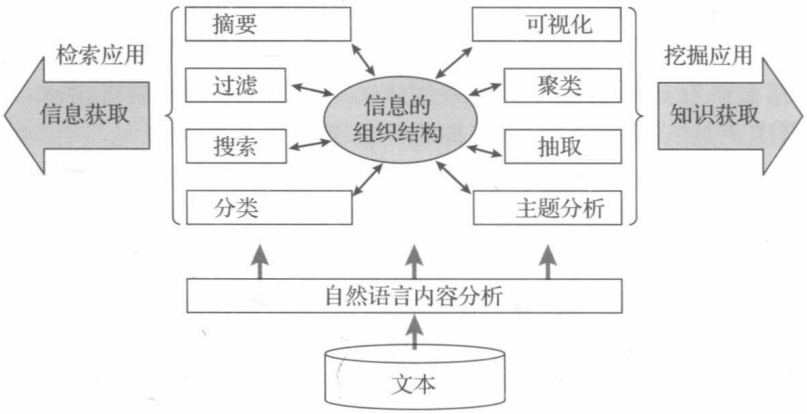


图 1-3　文本信息系统的概念框架

首先，需要基于自然语言处理技术的内容分析模块。该模块允许系统将原始文本数据转换为更有意义的表示，以便在搜索引擎中可以更有效地与用户的查询匹配，在文本分析中可以更有效地进行处理。目前的 NLP 技术主要依赖于统计机器学习，以有限的语言知识作为辅助，进行不同深度的文本数据理解；浅层技术是健壮的，但更深层的语义分析只适用于非常有限的领域。一些文本信息系统能力(如摘要)会比其他能力(如搜索)需要更深的 NLP。大多数文本信息系统使用非常浅的 NLP，其中文本将被简单地表示为“词袋”，词是表示的基本单位，并且文本中词的顺序被忽略(尽管保留了词频)。然而，也可以使用更复杂的表示，可以基于识别出的实体、关系或其他更深层的文本理解技术。

以内容分析为基础，文本信息系统中有多个组件以不同的方式帮助用户。以下是管理和分析文本信息的一些常见功能。

搜索(search)。接收用户查询并返回相关文档。文本信息系统中的搜索组件通常称为搜索引擎。网络搜索引擎是最有用的搜索引擎之一，它使用户能够有效和高效地处理大量的文本数据。

过滤/推荐(filtering/recommendation)。监督传入的数据流，确定哪些项目与用户的兴趣相关(或不相关)，然后向用户推荐相关项目(或者过滤掉不相关的项目)。根据系统是否侧重于识别相关项目或不相关项目，这个组件可以被称为推荐系统(其目标是向用户推荐相关项目)或者过滤系统(其目标是过滤掉非相关项目，允许用户只保留相关项目)。文献推荐器和垃圾邮件过滤器分别是推荐系统和过滤系统的典型例子。

10
~
11

分类(categorization)。将文本对象划分到一个或多个预定义类别，其中类别可根据应用程序而变化。文本信息系统中的分类组件可以用各种有意义的类别对文本对象进行注释，从而丰富了文本数据的表示，进一步提升了文本分析的效率和深度。类别也可用于组织文本数据，便于文本访问。将文章分类为一个或多个主题类别的主题分类器和将句子分类为正面、负面或中性的情感极性的情感标注器都是文本分类系统的具体例子。

摘要(summarization)。对一个或多个文本文件生成一个简要的内容摘要。摘要减少了人们消化文本信息的负担，也可以提高文本挖掘的效率。生成摘要的组件称为摘要器。新闻摘要和意见摘要都是摘要器的实例。

主题分析(topic analysis)。提取并分析给定文档集合的主题。主题直接促进了用户对文本数据的理解，并支持浏览文本数据。当与相关的非文本数据如时间、地点、作者等元数据相结合，主题分析可以产生许多有趣的模式，如主题的时间趋势、主题的时空分布和

作者的主题概况。

信息抽取(information extraction)。从文本中提取实体、实体之间的关系或其他“知识单元”。信息抽取组件可以构建实体关系图。这种知识图有多种用途，包括支持导航(沿着图的边和路径)以及进一步应用图挖掘算法去发现有趣的实体关系模式。

聚类(clustering)。发现相似文本对象(例如术语、句子及文档等)的群组。聚类组件在帮助用户探索信息空间方面起着重要的作用。它使用经验数据来创建有意义的结构，这对于浏览文本对象和快速理解大型文本数据集都非常有用。它对识别无法与其他对象聚集的异常对象也是非常有用的。

可视化(visualization)。以可见的方式显示文本数据中的模式。可视化组件对于吸引人们参与发现有趣模式的过程非常重要。由于人类非常善于识别视觉模式，所以将各种文本挖掘算法产生的结果可视化有很大需求。 12

以上各项也是本书后面将要讨论的主题的纲要。具体来说，第二部分内容讨论文本数据获取，其中将介绍搜索和过滤；第三部分内容讨论文本分析，将介绍分类、聚类、主题分析和摘要。本书没有介绍信息抽取，因为我们希望重点介绍容易应用于任何自然语言文本数据的一般方法，而信息抽取通常需要特定语言的相关技术。由于本书着重于算法，所以可视化也没有涵盖。但是，必须强调的是，信息抽取和可视化是与文本数据的分析和管理密切相关的重要话题。对这些技术感兴趣的读者可以在本章最后的书目说明中找到一些有用的参考资料。

1.3　本书结构安排

本书分为四部分，如图 1-4 所示。

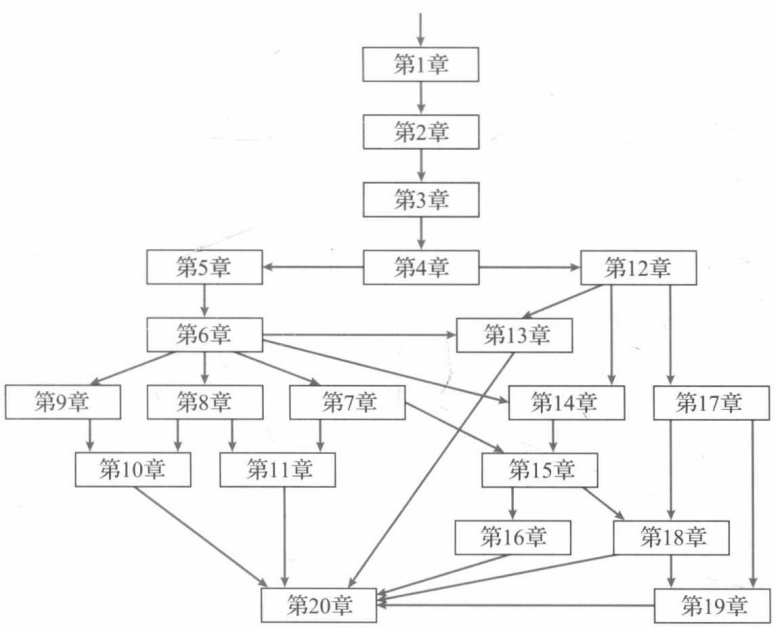


图 1-4　各章之间的依赖关系

第一部分　概述和背景。本部分由前 4 章组成，提供了本书概述和背景知识，包括了读者可能不熟悉的但对理解某些书籍内容而言必备的基本概念，以及这本书中用于练习的 MeTA 工具包的简介。本部分还简要概述了自然语言处理技术。自然语言处理是所有文本

数据分析应用为理解文本数据和获取有信息的文本表示所必需的技术。

第二部分　文本数据获取。本部分由第5～11章组成，涵盖了支持文本数据获取的主要技术。本部分对信息检索基本技术进行了系统讨论，包括将检索任务作为面向查询的文档排序问题(第5章)，构成搜索引擎排名函数设计基础的检索模型(第6章)，反馈技术(第7章)，检索系统的实现(第8章)以及检索系统的评价(第9章)。它涵盖了迄今为止最重要的信息检索应用——网络搜索引擎(第10章)，其中引入了用于分析文本数据中的链接以提高文本对象排名的技术，并简要介绍了利用监督机器学习结合多个特征的排序学习技术。这部分的最后一章(第11章)介绍了推荐系统，它提供了信息获取的“推送”模式，这不同于典型的搜索引擎支持的“拉取”模式的信息获取(即需要用户主动查询)。

第三部分　文本数据分析。本部分由第12～19章组成，涵盖了各种分析文本数据的技术，以方便用户消化理解文本数据，并发现文本数据中有用的主题或其他语义模式。第12章从数据挖掘的角度给出了文本分析的概述，可以将文本数据看作是由作为世界的“主观感受器”的人类生成的数据；这个观点使我们可以在更一般的数据分析和挖掘的背景下看待文本分析问题，并且有助于讨论文本和非文本数据的联合分析。接下来的多个章节，涵盖了许多非常有用的通用技术，可以不依赖或仅依赖最少的人工来分析文本数据。具体而言，第13章讨论了用于发现文本数据中的词汇单元之间的两种基本语义关系的技术，即聚合关系和组合关系，这可以被看作是发现用于生成文本数据的自然语言知识(即语言知识)的示例。第14章和第15章分别介绍了两种紧密相关的技术，即文本聚类和文本分类，用于生成有意义的结构或注释并与其他无组织的文本数据相关联。第16章讨论了文本摘要，能够帮助人们快速消化文本信息。第17章详细讨论了用于发现和分析文本数据中主题模式(即主题模型)的一类重要概率方法。第18章讨论分析文本数据中表达的情感和观点的技术，这些技术是在分析人们所产生的文本数据的基础上发现关于他们的偏好、观点和行为等知识的关键。最后，第19章讨论了文本和非文本数据的联合分析，因为在大数据分析中使用尽可能多的数据来获得知识和智能是有益的，许多应用程序均需要这样的技术。

13～14

第四部分　统一的文本管理和分析系统。最后一部分由第20章组成，我们试图说明本书中讨论的所有技术如何在概念上集成到一个基于操作器的统一框架中，从而有可能在一个统一的文本管理和分析系统中实现，用于支持各种不同的应用。本部分还作为进一步扩展MeTA的路线图，为各种应用提供有效和普遍的高级支持，并讨论如何将MeTA与许多其他相关的现有工具包整合起来，特别是与搜索引擎系统、数据库系统、自然语言处理工具包、机器学习工具包和数据挖掘工具包相结合。

由于我们试图从实践的角度来看待所有的话题，书中的大多数概念和技术的讨论都是非形式化的、直观的。为了满足一些读者可能有兴趣更深入了解某些主题的需求，本书附录对几个重要的主题提供了详细和严格的解释。

1.4　如何使用本书

15

由于我们希望涵盖的主题范围非常广泛，必须在覆盖范围的广度和深度之间进行许多折中。在进行这种折中的时候，我们选择强调文本数据挖掘的基本概念和实用技术的覆盖，代价是不能涵盖许多先进技术的细节，并且在很多章节的最后给出一些参考文献帮助有兴趣的读者更多地了解这些先进的技术。我们的希望是，在阅读本书的基础上，你将能够通过自己或通过其他资源了解更先进的技术。我们还选择了涵盖更多的文本管理和分析

的一般技术，并支持适用于任何自然语言的任何文本的技术。我们讨论的大多数技术都可以在没有任何人力或仅需要最小人力的情况下实施，这与文本数据的一些更详细的分析不同，特别是使用自然语言处理技术。这样的“深度分析”技术显然是非常重要的，对于一些我们想深入理解文本细节的应用来说，这确实是必要的。然而，在这一点上，这些技术往往是不可扩展的，往往需要大量的人力。在实践中，结合这两种技术将是有益的。

我们预想的读者(可能重叠)主要有三类。

学生。本书专门提供使用真实文本挖掘工具和应用程序的实践经验。如果单独使用，我们建议先阅读第1～4章，以便对本书的前提知识有一个很好的理解。第1～3章将帮助你熟悉和理解未来章节所必需的概念和词汇。第4章介绍每个章节的练习中使用的配套工具包MeTA。我们希望练习和章节描述为你自己的文本挖掘项目提供灵感。MeTA提供的代码是一个良好的技术支撑，让你更专注于你的贡献。

如果在课堂上使用，教师可以选择采取几个逻辑流程。作为必备知识，我们假定一些概率和统计方面的基本知识以及用C++或Java等语言编程的能力已经具备。MeTA是用现代C++编写的，但是一些练习只要通过修改配置文件就能完成。

教师。本书涵盖了具有逻辑性、彼此关联的主题，便于与各种课程体系结合。例如，本书的第一部分和第二部分可以作为本科的信息检索导论课程，重点介绍搜索引擎如何工作。练习假定学生具备基本的编程经验以及一些概率以及统计学的数学背景。也可作为通识课程，概览全书作为“文本数据挖掘”的入门，而跳过第二部分中更具体的搜索引擎实 16
现和特定于Web的应用程序的章节。另一种选择是将所有部分作为研究生教科书的补充内容，其中仍然强调实用的编程知识，可以结合每章中的参考文献阅读。研究生的练习可以要求实现MeTA的参考文献中提及的很多方法。

每章末尾的练习除了题目，还给学生提供了一个功能强大但易于理解的文本检索和挖掘工具包。在一个以编程为中心的课堂上，强烈鼓励使用MeTA练习。编程作业可以通过在每章中选择练习的子集来创建。由于工具包的模块化特性，可以通过扩展现有系统或实现目前MeTA默认不包括的其他众所周知的算法来创建新的编程实验。最后，学生可以使用他们通过练习学到的MeTA组件来完成一个更大的最终编程项目。将不同的语料库与工具包结合使用会产生不同的项目挑战，例如评论摘要与情感分析。

从业者。大多数行业读者很可能会用这本书作为参考，我们也希望这本书可以为你的工作带来启发。与学生用户的建议一样，阅读最初的3章就可以得到本书的大部分内容，然后可以选择一个与目前兴趣相关的章节深入研究或更新知识。

由于MeTA中的许多应用程序可以简单地通过配置文件使用，我们预计它是一个无须编程即可快速处理数据集并获得基线结果的工具。

每章的章后练习可以被认为是手头特定任务的默认实现。可以选择将MeTA纳入你的工作，因为它使用宽松的免费软件许可证。事实上，它拥有麻省理工学院和伊利诺斯大学/NCSA双重许可。当然，我们仍然鼓励并邀请读者分享对MeTA的修改、扩展和改进，为了所有读者的利益着想，这些改进不应作为专利。

不管你的目标是什么，我们希望这本书对你有帮助。我们感谢你提出的意见和建议，从而改进这本书。谢谢阅读！ 17

书目说明和延伸阅读

在信息检索(IR)领域已经有多部优秀的教科书。由于信息检索方面的研究历史悠久，

而且在20世纪60年代已经做了大量基础性的工作，即使van Rijsbergen[1979]、Salton和McGill[1983]和Salton[1989]等一些非常古老的书籍在今天也非常有用。另一个有用的早期书籍是Frakes和Baeza-Yates[1992]。更近的包括Grossman和Frieder[2004]、Witten等[1999]和Belew[2008]。最近的是Manning等[2008]、Croft等[2009]、Büttcher等[2010]、Baeza-Yates和Ribeiro-Neto[2011]。与这些书相比，本书对信息检索主题有更广泛的认识，并试图涵盖文本检索和文本挖掘的内容。虽然现有的一些关于IR的书籍也涉及一些话题，如文本分类和文本聚类(我们将其归类为文本挖掘)，但是没有一本书包含了对主题挖掘和分析的深入讨论，对于文本挖掘这是非常有用的一类重要技术。现有的关于IR的书籍似乎缺少推荐系统，我们将其视为另外一种支持用户进行文本获取的方式，从而与搜索引擎相辅相成。更重要的是，本书更为系统化地处理所有这些主题，并通过统一的概念框架来管理和分析大型文本数据。本书还尝试通过提供许多练习的配套工具包来尽量减小算法中的抽象解释和实际应用之间的差异。希望了解更多关于IR研究历史和早期重要里程碑的读者应该关注Sparck Jones和Willett[1997]中的阅读材料。

文本挖掘的主题也被许多书籍所覆盖(例如，Feldman和Sanger[2007])。这本书与那些书的主要区别是我们强调文本挖掘和信息检索的整合，相信任何文本数据应用系统都必须让人参与其中，而搜索引擎是任何文本挖掘系统的基本组成部分，以支持两个关键功能：1) 数据来源——帮助将大的原始文本数据集合转换成更小但更相关的文本数据集，从而可以使用文本挖掘算法有效地分析；2) 知识起源——帮助用户验证文本挖掘算法用于发现知识的原始文本文章。因此，本书更全面地涵盖了开发大型文本数据应用所需的技术。

本书的重点是介绍易于应用到任何自然语言的任何文本数据的通用和健壮的算法，这
18 些算法往往不需要或仅需要最少的人力。因此，一个不可避免的代价是它缺乏对文本挖掘中个别关键技术的覆盖，特别是信息抽取(IE)技术。我们决定不覆盖IE技术是因为它经常是语言相关的，而且需要并不简单的人工工作。另一个原因是，许多IE技术依赖于监督机器学习方法，这在许多现有的机器学习书籍(例如参见[Bishop 2006]和[Mitchell 1997]中已有很好的介绍。有兴趣了解更多关于IE技术的读者可以从查阅书籍[Sarawagi 2008]和评论文章[Jiang 2012]开始。

从应用的角度来看，由于强调对模型和算法的覆盖，本书缺少的另一个重要话题是信息可视化。然而，既然每个应用系统都必须具有用户友好的界面，以便用户与系统进行最佳的交互，所以那些对开发文本应用系统感兴趣的读者肯定会发现学习更多关于用户界面设计的内容是有用的。可以参考Hearst [2009]作为入门，它对信息可视化有较为详细的介绍。

最后，由于强调广度，本书没有深入介绍任何组件算法。为了了解一些主题的更多信息，读者可以进一步阅读自然语言处理的书籍(例如，Jurafsky和Martin[2009]，Manning和Schütze[1999])、关于IR的高级书籍(如Baeza-Yates和RibeiroNeto[2011])以及关于机
19 器学习的书籍(如Bishop[2006])。你可以在每章末尾的书目说明中找到与特定主题相关的
~ 更具体的阅读建议。
20

背　景

本章主要涉及学习本书其他内容需要掌握的背景知识。对于已经熟悉这些基本概念的读者，可以选择性地跳过部分小节，也可以跳过整章内容。首先介绍一些概率论和统计学的基本概念，这些概念在本书的大部分算法和模型中均有涉及。接着概述文本挖掘应用中经常用到的一些信息论概念。最后一节介绍机器学习的基本思想和问题设置，尤其是有监督的机器学习，它对文本分类和基于文本的预测非常有用。总体来说，机器学习被广泛应用到了很多信息检索和数据挖掘任务。

2.1　概率和统计基础

正如本章以及其他章节的内容所述，概率和统计模型在文本挖掘算法中发挥着重要作用。本节将为读者提供足够的背景信息和专业概念，以便理解本书后面章节所涵盖的概率和统计方法。

概率分布是对某个概率空间 Ω 中事件可能性赋值的一种方法。例如，假设概率空间是一个六面骰子，每一面都有不同的颜色，那么 $\Omega=\{$红，橙，黄，绿，青，蓝，紫$\}$，掷一次骰子并观察颜色就是一个事件。

我们通过为所有可能的事件定义一个概率分布来量化掷骰子的不确定性。假设有一个无偏的骰子，掷出任意一种颜色的可能性都是 $\frac{1}{6}$，或者说大概 16%。这样，可以将概率分布表示成一个概率的集合：

$$\theta=\left\{\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}\right\}$$

其中第一项对应 p(红色)$=\frac{1}{6}$，第二项对应 p(橙色)$=\frac{1}{6}$，以此类推。但是如果骰子有偏，情况会是什么样呢？可以用一个不同的概率分布 θ' 来表示相应事件的可能性：

$$\theta'=\left\{\frac{1}{3},\frac{1}{3},\frac{1}{12},\frac{1}{12},\frac{1}{12},\frac{1}{12}\right\}$$

在这种情况下，相比其他颜色，红色和橙色被认为更容易被掷出。注意样本空间 Ω 和用来量化不确定性的概率模型 θ 之间的差别。在文本挖掘任务中，通常会利用与 Ω 有关的一些信息来估计 θ，不同的估计方法将决定一个概率模型的准确性和可用性。

现在引入下面的记号：

$$x\sim\theta$$

它表示随机变量(random variable) x 取自(服从)概率分布 θ。随机变量 x 对应 Ω 中每个元素的概率都由 θ 确定。例如，如果有 $x\sim\theta'$，那么 x 是红色或橙色的概率为 $\frac{2}{3}$。

在文本应用任务中，样本空间 Ω 通常为文本语料库的词汇表 V。例如，词汇表可以是

$$V=\{\text{a, and, apple, …, zap, zirconium, zoo}\}$$

我们用一个概率分布 θ 对文本数据进行建模。因此，给定一个单词 w，可以将其对应的概

率记作 $p(w|\theta)$(读作在 θ 条件下 w 的概率)。如果 w 是单词 data，可能有 $p(w=\text{data}|\theta)=0.003$ 或者 $p_\theta(w=\text{data})=0.003$。

上面的例子中只考虑了离散概率分布(discrete probability distribution)，也就是说，模型只为有限(离散)的结果分配概率。实际上，也有连续概率分布(continuous probability distribution)。在这样的概率分布中，“事件”的数目是无限的、不可数的。例如，正态(高斯)分布就是连续概率分布，它根据参数来分配实值概率。本书将在后面的章节中根据需要讨论连续分布。现在，只需要了解可以使用离散概率分布来模拟从词汇表 V 中观察到
22 单个单词的概率。

柯尔莫哥洛夫公理描述了一般概率分布(包括离散的和连续的)遵循的事实。在设计模型时，柯尔莫哥洛夫公理是一个很好的检查方式，现在对它进行介绍。概率空间 Ω 上一个有效的概率分布 θ 必须满足以下三条公理。

- 每个事件的概率都在 0～1 之间：

$$0\leqslant p_\theta(\omega\in\Omega)\leqslant 1 \tag{2-1}$$

- 不在 Ω 中的事件概率为 0，Ω 中的事件发生的概率为 1：

$$p_\theta(\omega')=0,\ \omega'\notin\Omega \text{ 且 } p_\theta(\Omega)=1 \tag{2-2}$$

- 所有(互斥)事件的概率和为 1：

$$\sum_{\omega\in\Omega} p_\theta(\omega)=1 \tag{2-3}$$

值得注意的是，严格来讲，一个事件被定义为概率空间 Ω 的一个子集，该事件发生当且仅当随机实验的结果(即从 Ω 中随机取一个结果)在该事件对应的子集中。因此，很容易理解空集对应的特殊事件不可能发生，其概率为零；全集对应的特殊事件必然发生，其概率为 1。作为一个特例，我们考虑一个事件空间，这个空间中每一个事件只有一种结果，这正好与之前所述单词概率分布所做的假设相符。每个单词对应的事件是由该单词作为唯一元素的子集所定义的，很明显，这样的事件发生当且仅当结果是相应的单词。

2.1.1　联合概率和条件概率

在本节中，我们修改一下之前掷骰子的例子。首先保持原始的分布 θ_C，也就是出现每种颜色的概率为：

$$\theta_C=\left\{\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}\right\}$$

23 假设每种颜色都由一个特定的形状表示，这样骰子看起来像

{■, ■, ●, ●, ●, ▲}

其中每个形状对应的颜色分别是红色、橙色、黄色、蓝色、绿色和紫色。

现在为形状创建另一个分布 θ_S，θ_S 中的数字分别代表 p(正方形)、p(圆形)、p(三角形)，那么

$$\theta_S=\left\{\frac{1}{3},\frac{1}{2},\frac{1}{6}\right\}$$

然后令 $x_C\sim\theta_C$ 表示颜色随机变量，$x_S\sim\theta_S$ 表示形状随机变量。现在我们有两个变量需要考虑。

联合概率(joint probability)刻画两个事件同时发生的可能性。例如，$x_C=$红色并且 $x_S=$圆形的概率是多少？因为没有红色的圆形，所以概率为 0。$p(x_C=$绿色，$x_S=$圆形$)$又是

多少呢？这种表示用来标记两个随机变量的联合概率。在本例中，该联合概率是$\frac{1}{6}$，因为只有一个绿色圆形。

考虑一个修改后的骰子：



其中蓝色圆形(集合中的第四个元素)的颜色被修改成了绿色。因此，现在有两个绿色圆形而不是一绿一蓝。那么 $p(x_C=\text{绿色}, x_S=\text{圆形})$又会是多少呢？因为 6 个元素中有两个满足条件，所以答案是$\frac{2}{6}=\frac{1}{3}$。再举一个例子，如果有一个 12 面骰子，其中有 5 个绿色圆形以及 7 个其他形状和颜色的组合，那么 $p(x_C=\text{绿色}, x_S=\text{圆形})=\frac{5}{12}$。

条件概率(conditional probability)刻画在一个事件已经发生的情况下另一个事件发生的概率。这里使用原始的有 6 种不同颜色的骰子进行解释。假设已知掷出了正方形，那么颜色为红色的概率是多少？为紫色的概率又是多少？第一个概率可以写成 $p(x_C=\text{红色}\mid x_S=\text{正方形})$，由于已知有两个正方形，并且其中一个是红色，所以 $p(x_C=\text{红色}\mid x_S=\text{正方形})=\frac{1}{2}$。

可以根据联合概率，写出随机变量 X 和 Y 的条件概率：

$$p(X=x\mid Y=y)=\frac{p(X=x, Y=y)}{p(Y=y)} \tag{2-4}$$

24

分子 $p(X=x, Y=y)$是同时观察到 x 和 y 的联合概率，通过被分母 $p(Y=y)$归一化，可得给定 y 后 x 的条件概率。通过上述方式，可以计算出 $p(x_C=\text{绿色}\mid x_S=\text{圆形})$：

$$p(x_C=\text{绿色}\mid x_S=\text{圆形})=\frac{p(x_C=\text{绿色}, x_S=\text{圆形})}{p(x_S=\text{圆形})}=\frac{1/6}{1/2}=\frac{1}{3}$$

另一个需要介绍的重要概念是独立性(independence)。在前面的例子中，两个随机变量是相互依赖的，这意味着一个变量的值将影响另一个变量的值。考虑另一种情况，有 c_1，$c_2\sim\theta_C$，即从颜色分布中选取两种颜色。c_1 是否能够帮忙推测 c_2 的概率呢？答案是不会。因为这两次选取都是互相独立的。在这种情况下，两个随机变量 X 和 Y 是相互独立的，可得 $p(X, Y)=p(X)p(Y)$。为什么会是这样呢？

联合概率和条件概率都可以用于解决一些与文本相关的问题。例如，给定一个文档，在同一句子中同时观察到单词 information 和 retrieval 的概率是多少？如果知道 information 已经存在，观察到 retrieval 的概率是多少？

2.1.2　贝叶斯法则

贝叶斯法则可以根据条件概率的定义推导而来：

$$p(X\mid Y)=\frac{p(X, Y)}{p(Y)}\text{ 且 }p(Y\mid X)=\frac{p(Y, X)}{p(X)}$$

因此，设置这两个联合概率相等，

$$p(X\mid Y)p(Y)=p(X, Y)=p(Y\mid X)p(X)$$

可以将其简化为

$$p(X\mid Y)=\frac{p(Y\mid X)p(X)}{p(Y)} \tag{2-5}$$

上述公式被称为贝叶斯法则，以托马斯·贝叶斯(1701—1761)的名字命名。这个法则有着广泛的应用。本书将会在文本分类章节以及主题分析章节中大量使用这个公式。具体应用细节将在各个章节中分别介绍。从本质上来说，贝叶斯法则是一种根据已知信息对假设进
25 行推理的通用方法。

具体来说，可以将随机变量 X 视为假设，将随机变量 Y 视为观察到的信息。$p(X)$可以理解为假设为真的先验知识，之所以称其为“先验”是因为它是在观察到 Y 之前的估计。相比之下，$p(X|Y)$是假设为真的后验概率，因为它是在观察到 Y 之后得到的。贝叶斯法则是连通先验知识和后验知识的桥梁，它提供了一种基于观察数据 Y 来更新先验概率 $p(X)$并获得后验概率 $p(X|Y)$的方法。显然，如果 X 和 Y 是独立的，更新不会发生，有 $p(X|Y)=p(X)$。

2.1.3　抛硬币和二项分布

在讨论概率的大多数例子中，抛硬币是一个很好的例子。例如，如果想对文本文档中某个单词是否存在进行建模，可以很容易地把它映射到抛硬币问题上。抛硬币有两种可能的结果：正面或者反面。正面的概率记为 θ，反面的概率则为 $1-\theta$。

可以使用伯努利分布来刻画成功(在这个例子中是正面出现的事件)的概率。伯努利分布给出了一次事件——抛一次硬币——成功的概率。如果想要刻画抛 n 次硬币试验、成功 k 次的概率，就需要使用二项分布。因为 k 是整数，所以二项分布是一个离散分布，可以写成

$$p(k\text{ 次正面})=\binom{n}{k}\theta^k(1-\theta)^{n-k} \tag{2-6}$$

也可以写成如下形式：

$$p(k\text{ 次正面})=\frac{n!}{k!\ (n-k)!}\theta^k(1-\theta)^{n-k} \tag{2-7}$$

那么如何解释这个公式呢？分开来看，如果有 n 次试验，并且希望看到 k 次正面，那么必须抛出 k 次正面和 $n-k$ 次反面。观察到 k 次正面的每一次概率都为 θ，同时观察到剩余 $n-k$ 次反面的每一次概率都为 $1-\theta$。因为假设所有抛硬币事件都是独立的，所以可以把所有结果相乘。由于不关心结果的顺序，所以还要乘以从 n 次试验中选出 k 次试验可能的方法数量。

如果需要考虑结果的顺序怎么办？例如，观察到特定结果序列(正，反，正，正，反)
26 的概率是什么？很容易看出，这个序列出现的概率就是每个事件概率的乘积，即 $\theta\times(1-\theta)\times\theta\times\theta\times(1-\theta)=\theta^3(1-\theta)^2$，不再需要为了三次正面和两次反面顺序的不同而调整。

在文本分析中更常用多项分布。多项分布刻画了在特定场景中(例如，在文档中)观察到一个单词的概率。它与伯努利分布非常相似，唯一的区别是多项分布具有两个以上的结果。

2.1.4　最大似然参数估计

现在有一个抛硬币的模型，如何根据已经观察到的数据来估计参数呢？例如，可能已经观察到了前面讨论的数据 D，其中 $n=5$：

$$D=\{\text{正，反，正，正，反}\}$$

接下来，我们想要基于观测数据求出 θ。使用最大似然估计(Maximum Likelihood Estimation，MLE)，可以根据观测数据选择可能性最高的 θ，也就是使观测数据出现概率

最大的 θ。

为了计算 MLE，首先写出似然函数，即 $p(D|\theta)$，前面计算出它的数值为 $\theta^3(1-\theta)^2$。因此问题转化为求解 θ，使得函数 $f(\theta)=\theta^3(1-\theta)^2$ 最大化。因为对数变换可以保持函数的单调性，所以也可以等价地最大化对数似然函数，即 $\log f(\theta)=3\log\theta+2\log(1-\theta)$。根据微积分的知识，函数在 θ 取到最大值的必要条件是在 θ 处导数为零。因此，只需要求解下面的等式：

$$\frac{\mathrm{d}\log f(\theta)}{\mathrm{d}\theta}=\frac{3}{\theta}-\frac{2}{1-\theta}=0$$

很容易发现解是 $\theta=3/5$。

更一般地，令 H 为正面的次数，T 为反面的次数，正面概率的最大似然估计为：

$$\begin{aligned}\theta_{\mathrm{MLE}}&=\arg\max_{\theta}p(D|\theta)\\&=\arg\max_{\theta}\theta^H(1-\theta)^T\\&=\frac{H}{H+T}\end{aligned}$$

27

符号 arg max 表示使似然函数(即 $p(D|\theta)$)达到最大值的参数值(在本例中为 θ)。因此，对 arg max 内的函数进行任何单调变换，arg max 表达式的值都将保持不变。这就是为什么可以在上面的例子中使用对数变换，这样可以更容易地计算导数。

上述最大似然估计的解法是很直观的：使观测数据最有可能出现的 θ 就是硬币正面出现的比例。最大似然估计的一般特性是，这种方法估计的概率是相应事件的发生次数的归一化比率。例如，多项分布的最大似然估计(将在本书后面进一步详细讨论)赋予每种可能结果的概率与该结果出现的次数成比例。值得注意的是，根据最大似然估计，所有未观察到的结果将具有零概率。这通常是不合理的，特别是当数据样本量很小的时候。这一问题也因此引出了下面对贝叶斯参数估计的讨论。

2.1.5 贝叶斯参数估计

最大似然估计的一个潜在问题是，当数据样本很小时，估计结果通常不够准确，因为它总是尽可能地拟合已有的数据。考虑一个极端的例子，只观察两次抛硬币的数据，这两次数据恰好都是正面，最大似然估计会认为正面概率是 1，反面概率是 0。直观上这样的估计是不正确的，即使它最大化了观察到这两次数据的概率。

通过考虑参数的不确定性，使用贝叶斯参数估计代替最大似然估计可以处理和减轻“过拟合”问题。在贝叶斯参数估计中，我们将考虑参数所有可能值的分布，即将参数本身视为一个随机变量。

具体来说，可以使用 $p(\theta)$来表示 θ 所有可能值的分布，它包含了对 θ 真实值的先验认知。数据 D 则为这种先验是否为真提供了证据。我们可以基于观察到的数据来更新先验概率 $p(\theta)$。下面使用贝叶斯法则重写 $p(\theta|D)$，即给定数据条件下参数的概率：

$$p(\theta|D)=\frac{p(D|\theta)p(\theta)}{p(D)} \tag{2-8}$$

其中 $p(D)$可以通过对 θ 的所有情况求和来计算。对于一个连续分布，将是：

28

$$p(D)=\int_{\theta'}p(\theta')p(D|\theta')\mathrm{d}\theta' \tag{2-9}$$

这意味着特定 θ 的概率是

$$p(\theta|D)=\frac{p(D|\theta)p(\theta)}{\int_{\theta'}p(\theta')p(D|\theta')\mathrm{d}\theta'} \tag{2-10}$$

这些数值有专门的名称：

- $p(\theta|D)$：θ 的后验概率。
- $p(\theta)$：θ 的先验概率。
- $p(D|\theta)$：D 的似然。
- $\int_{\theta'}p(\theta')p(D|\theta')\mathrm{d}\theta'$：$D$ 的边缘似然。

最后一个被称为边缘似然，是因为积分将 θ 从等式中“边缘化”（除去）了。既然数据的似然保持不变，所以 $p(\theta|D)$ 必须满足在 θ 的所有取值上和为 1 的约束，通常有

$$p(\theta|D)\propto p(\theta)p(D|\theta)$$

即后验概率与先验概率成正比。

参数 θ 的后验分布充分刻画了参数取值的不确定性，并且可以用于推断依赖于该参数的任何数值，包括计算参数的点估计（即参数的单值估计）。基于后验分布计算点估计有多种方法。一种方法是计算后验分布的平均值，由概率和参数值的加权和给出。对于离散分布，

$$E[X]=\sum_x xp(x) \tag{2-11}$$

对于连续分布，

$$E[X]=\int_x xf(x)\mathrm{d}x \tag{2-12}$$

有时，还可以使用后验分布的众数作为参数的估计，这被称为最大后验（Maximum a Posteriori，MAP）估计：

29

$$\theta_{\mathrm{MAP}}=\arg\max_\theta p(\theta|D)=\arg\max_\theta p(D|\theta)p(\theta) \tag{2-13}$$

这里很容易看出，基于先验概率 $p(\theta)$ 来最大化参数的概率，MAP 估计将偏离 MLE 估计。通过使用合适的先验知识，可以解决 MLE 过拟合的问题，因为先验知识更倾向于无论是正面还是反面都不应该有零概率估计。

如果读者要对这些内容进行更深入的讨论，请参阅附录 A。

2.1.6　概率模型及其应用

通过对前几节统计基础的介绍，现在开始学习如何将概率模型应用于文本分析。

通常来说，在文本处理中，我们关心如何设计面向文本数据的概率模型，可以用来刻画词序列上的概率分布。这种模型通常被称为统计语言模型，或者文本数据的生成模型（即可用于单词序列样本的概率模型）。

正如前面的介绍，通常将语料库中所有单词的集合 V 视为样本空间 Ω，根据数据集中的词项来定义概率分布。如果不考虑单词的顺序，这个分布本质上是多项分布。虽然有许多更复杂的文本数据模型（例如，Jelinek[1997]），但是这种最简单的模型（通常被称为一元语言模型）对于文本数据管理和分析中的许多问题都非常有用，因为词汇表中的词项是人们为了交流而精心设计的有意义的基本单位。

现在，我们来介绍“学习”统计模型的一般框架。学习一个模型意味着估计里面的参数，当概率分布定义在词项上时，V 中每一个元素都有一个参数，工作流程如下所示。

1）定义模型。

2）学习参数。

3）应用模型。

第一步在之前的内容中已经介绍完毕。在本书的例子中，我们希望得到语料库中每一个单词出现的概率。第二步，需要确定如何设置每个单词的概率。一种直观的方法是计算语料库中每个单词出现的概率，即单词 w_i 出现的次数除以语料库中所有单词的总数，就 30
是 $p(w_i|\theta)$的值。可以证明这是该模型最大似然估计的解。本书稍后将讨论更复杂的模型和参数估计方法。最后，一旦确定了 θ，如何使用它进行文本分析呢？一个实例是分析语料库中一个特定的词项子集的概率；另一个实例是对于不可见的文本数据建模并计算在新文本中词项出现的概率。通常可以设计一个模型，使模型的参数刻画了试图从文本数据中发现的知识模式。在这种情况下，模型参数的估计可以直接用作文本挖掘的输出(结果)。

值得注意的是，概率模型是一个通用工具，而不仅仅被用于文本分析——这只是本书的主要应用！

2.2 信息论

信息论处理的是不确定性并以位的形式传输或存储量化信息。它被应用于许多领域，例如电子工程、计算机科学、数学、物理和语言学。信息论中的一些概念在文本数据管理和分析中非常有用，我们在这里简单介绍一下。信息论最重要的概念是熵(entropy)，它是许多其他度量的基石。

问题可以定义为在预测随机变量的值时量化的不确定性。在常见的掷硬币的例子中，有两个值 1 和 0(表示正面和反面)并且用随机变量 X 表示这些结果。换句话说，

$$X=\begin{cases}1 & \text{如果为正面}\\ 0 & \text{如果为反面}\end{cases}$$

随机变量的随机性越大，预测正反面的难度就越大。如何定量地预测一个随机变量，如 X 的随机性？这正是熵的作用。

一般意义来说，随机变量 X 的熵，$H(X)$，是指表示一个事件 $x\sim X$ 的结果所需位数的一种度量。如果结果是已知的(完全确定)，我们不需要表示任何信息，因此 $H(X)=0$。如果结果是未知的，我们希望尽可能有效地利用位数表示结果。这意味着对经常发生的事件比不常发生的事件使用更少的位数。熵利用下面的公式给出了对于任意 $x\sim X$ 的期望的 31
位数：

$$H(X)=-\sum_{x\in X}p(x)\log_2 p(x) \tag{2-14}$$

对于 $\log_2 0$，我们通常定义其为 0，因为$\log_2 0$ 没有定义。对于不同的随机变量 X，我们将会得到不同的 $H(X)$。

这个公式背后的理论和推理超出了本书的范围，但它足以说明 $H(X)=0$ 意味着没有随机性，$H(X)=1$ 意味着所有事件都是完全随机的。因此，随机性的取值是从 0～1 的。对于掷硬币例子来说，样本空间是两个事件(正面或反面)，其熵函数为：

$$H(X)=-p(X=0)\log_2 p(X=0)-p(X=1)\log_2 p(X=1)$$

对于一枚均匀的硬币，我们有 $p(X=1)=p(X=0)=1/2$。$H(X)$可计算如下：

$$H(X)=-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}=1$$

然而对于一枚有完全偏差的硬币，我们有 $p(X=1)=1$，$p(X=0)=0$，于是：

$$H(X)=-0\log_2 0-1\log_2 1=0$$

对于本例，我们只有两种可能的结果(即一个二元随机变量)。从公式中可以看到，熵的概念很容易泛化到两个以上随机变量的情况，在这些情况下，求和部分包括两个以上元素。

对于掷硬币的例子，图 2-1 绘制了 $H(X)$关于正面的概率 $p(X=1)$的变化情况。在 X 轴的两端，$X=1$ 的概率要么非常小要么非常大。在这两种情况下，熵函数的值很小，因为结果的随机性是非常低的。随机性最高的地方是 $p(X=1)=\frac{1}{2}$处。在这种情况下，$H(X)=1$ 达到了最大值。因为这两个概率是对称的，随着 $p(X=1)$变化得到的 $H(X)$是一条对称的倒 U 形曲线。

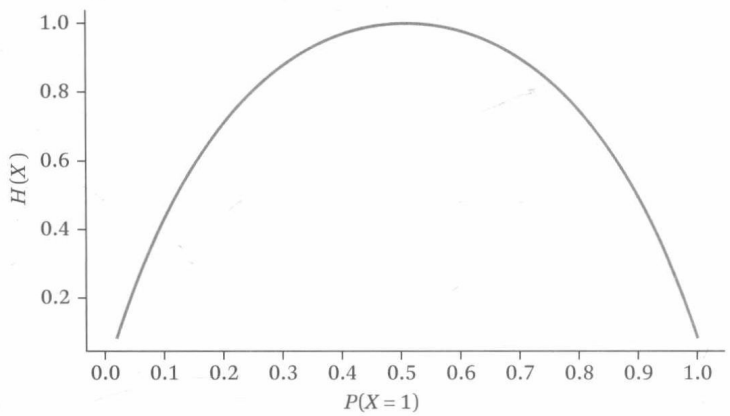


图 2-1　熵作为随机变量随机性的度量

让我们来思考一下一个特定的随机变量(不仅是掷硬币的例子)什么时候获得最大或最小值。我们来考虑一些特殊情况。例如，可能有一个随机变量 Y，它总是取 1。或者有一个随机变量 Z，它可能以相同的可能性取 1、2 或 3。在这些情况下，$H(Y)<H(Z)$，因为 Y 的结果比 Z 更容易预测。这正是熵能捕获的性质。你可以计算 $H(Y)$和 $H(Z)$来证实这个答案。

对于我们的应用来说，考虑某些上下文中的单词 w 的熵是非常有用的。在这里，熵高的词汇将更难预测。用 W 作为一个随机变量来表示一个单词是否在语料库的文档中出现。如果单词出现，$W=1$，否则 $W=0$。那么你认为 $H(W_{\text{the}})$和 $H(W_{\text{computer}})$相比如何？因为单词 the 无处不在，所以其熵接近于 0。在任何文档中看到该单词都不奇怪，因此很容易预测 $W_{\text{the}}=1$。这个例子就像有偏差的硬币总是某一面朝上。另一方面，单词 computer 是一个不太常用的单词，并且很难去预测它是否会出现在文档中，因此其熵值将会更高。

当我们尝试去量化条件概率的不确定性时，也可以定义条件熵 $H(X|Y)$，它表示在观察到 Y 的条件下期望得到 X 的不确定性，该期望是在所有可能的 Y 值的分布中得到的。直观地说，如果 X 完全由 Y 决定，那么 $H(X|Y)=0$，因为一旦知道了 Y，X 便不存在不确定性。反之，而如果 X 和 Y 是独立的，那么 $H(X|Y)$将和 X 的最初的熵是一样的，即 $H(X|Y)=H(X)$，因为知道 Y 对得到 X 的不确定性是没有任何帮助的。

32
~
33

另一个有用的概念是两个随机变量的互信息，$I(X;\ Y)$，它被定义为由于知道另一个随机变量 Y 而导致随机变量 X 的熵的减少，即

$$I(X;\ Y)=H(X)-H(X|Y) \tag{2-15}$$

互信息可以等价地写为

$$I(X;\ Y)=H(Y)-H(Y|X) \tag{2-16}$$

很容易看出，如果 X 和 Y 是相关的，$I(X;\ Y)$较大，而如果 X 和 Y 不相关，$I(X;\ Y)$较小。实际上，在极端情况下，当 X 和 Y 完全独立时，熵不会减小，因此 $H(X)=H(X|Y)$，并且 $I(X;\ Y)=0$。然而，如果 X 完全依赖于 Y，那么 $H(X|Y)=0$，因此 $I(X;\ Y)=H(X)$。直观上来讲，互信息可以衡量两个随机变量之间的相关性。显然，作为 X 和 Y 的相关性度量，互信息是对称的。

这些基本概念的应用，包括熵、条件熵、互信息等，将在本书的后续部分进一步讨论。

2.3 机器学习

机器学习是解决许多问题的重要技术，具有广泛的应用前景。在文本数据管理和分析中，它也有许多广泛用途。对这个主题的任何深入的处理显然超出了本书的范围，但是在这里，我们将介绍一些机器学习的基本概念，以便更好地理解本书后续的内容。

机器学习技术通常可以分为两类：监督学习(supervised learning)和非监督学习(unsupervised learning)。在监督学习中，计算机从一组例子(称为训练数据)中来学习计算一个函数$\hat{y}=f(x)$，每一个例子包括一个输入值 x 和一个预期的输出值 y。通常，对于每一个 x 其对应的 y 值必须由人来提供，所以计算机相当于从人类处获得了某种形式的监督或指导，这也是它被称为监督学习的原因。一旦这个函数被学习出来，计算机就可以对任何未见过的输入 x 计算函数 $f(x)$。

当 y 的取值是一个有限数值集合中的元素时，函数 $f(\cdot)$可以被用作一个分类器。可选数值元素可以被称为标签，而 $f(\cdot)$用来将一个实例 x 映射到一个“正确”标签(在适当的时候可以是多个正确标签)。因此，此类问题被称作分类问题。分类问题中最简单的情形是只有两个标签时(称为二元分类)。当 y 的取值是一个实数时，该问题称为回归问题。这两种形式都可以被称作预测问题，我们的目标主要是根据已知的 x 推断未知的 y。当 y 是指未来事件的某些属性时，“预测”是非常有意义的。 34

在面向文本的应用中，这两种形式问题都可能出现。分类问题更为常见，被称为文本分类(text categorization 或 text classification)。本书专门有一章(第15章)是关于这个主题的。回归问题可能会出现在使用文本数据来预测非文本变量的情形下，如情感评价或股票价格，这两种情况后续也会讨论。

在分类和回归中，(输入)数据实例 x 通常表示为一个特征向量，其中每一个特征都提供了关于 $f(x)$的值最可能是 y 的哪一个取值的潜在线索。计算机从训练数据中学到的是一种优化的方式，该方式将这些特征及其对应的权重结合起来，来体现它们的重要性以及它们对最终函数值 y 的影响。这里的“优化”简单来讲就是最小化训练集上的预测误差，也就是说，使训练数据中所有实例的预测值$\hat{y}$与真实标签 y 尽可能一致。

更形式化地，将对象集合记为 X，$x_i\in X$ 是表示第 i 个对象的特征向量。一个特征是对象的一个属性，以某种方式描述该对象。例如，如果对象是新闻文章，那么一个特征可能就是文章中是否出现了“good”这个词。所有这些不同的特征作为表示该文章的特征向量的一部分。在本例中，特征向量通常与文档中出现的单词有关。

另一方面，Y 是每个对象可能对应的标签集合。因此，y_i可能是新闻分类体系中的体育，而y_j可能是政治。

一个分类器就是一个函数 $f(\cdot)$，该函数以一个特征向量作为输入，输出一个预测的标签$\hat{y}\in Y$。因此，我们可能有 $f(x_i)=$体育，意味着$\hat{y}=$体育。如果真实标签 y 也是体育，

那么分类器的预测就是正确的。

注意，只有在知道数据的真实标签的情况下，才能评价一个分类算法。事实上，为了学习到一个好的函数 $f(\cdot)$，使它能够处理未见过的特征向量并完成分类，我们必须使用真实的标签。由于这个原因，当学习机器学习算法时，我们通常把数据 X 分成两个部分：**训练数据**(training data)和**测试数据**(testing data)。训练数据部分用来构建分类器，测试
35 数据部分用来评价算法的性能(例如确定有多少真实标签被正确预测)。在真实的应用中，训练数据通常是我们可以得到的所有标注了标签的数据，而测试数据则是我们的机器学习程序应用的对象。

函数 $f(\cdot)$实际上是怎样工作的呢？考虑一个非常简单的例子，判断一篇新闻文章的情感是积极(positive)的还是消极(negative)的，也就是说，$Y=\{\text{positive}, \text{negative}\}$：

$$f(x)=\begin{cases}\text{positive} & \text{如果 } x \text{ 中词项 good 的频率大于 1}\\ \text{negative} & \text{否则}\end{cases}$$

当然，这个例子过于简单，但它的确演示了一个分类器的基本思想：它接收一个特征向量作为输入并输出一个类别标签。根据训练数据，分类器可能已经确定了情感积极的文章包括词项 good 多于一次；因此这个知识被编码到函数中。第 15 章将研究一些根据训练数据创建函数 $f(\cdot)$的具体算法。其他一些主题，如信息检索中的反馈(第 7 章)和情感分析(第 18 章)也会使用分类器，或者类似于分类器的方法。因此，了解机器学习是什么以及机器学习所能解决的问题是非常有用的。

与监督学习相比，在非监督学习中，只有数据实例 X，没有标签 Y。在这种情况下，我们显然不知道如何基于一个实例 x 来计算 y。但是，仍然可以学习到 X 的潜在属性或结构。由于学习过程中没有人为参与，这类方法被称为非监督的。例如，计算机可以了解到一些数据实例非常相似，并且整个数据集可以由三个主要的类簇来表示，每个类簇中的数据都是非常类似的。这就是第 14 章将要讨论的聚类技术。另一种非监督学习是设计概率模型对数据建模(称为“生成模型”(generative model))，我们可以在其中嵌入有趣的参数，这些参数表示我们希望从数据中发现的知识。通过用模型拟合数据，可以估计出能够最好地解释数据的参数值，并将获得的参数值作为从数据中发现的知识。第 17 章详细讨论了这种方法在分析文本潜在主题上的应用。

书目说明和延伸阅读

在诸如 Hodges 和 Lehmann[1970]的许多教科书中，可以找到关于概率和统计的基本
36 概念的详细讨论。在 Myung[2003]中，我们可以很好地了解到最大似然估计的介绍。在书籍《Bayesian Data Ana lysis》[Gelman 等 1995]中，全面介绍了贝叶斯统计论。Cover 和 Thomas[1991]为信息论提供了一种易于理解的介绍。在许多有关机器学习的书籍中，对于机器学习的基础概念和许多具体的机器学习方法都有严格的介绍(例如，Bishop 2006，Mitchell 1997)。

练习

2.1　如果我们知道 X 和 Y 是独立随机变量，我们可以说 $p(X|Y)$等于何值？并证明。

2.2　在信息检索课中，有 78 个计算机科学专业、21 个电子和计算机工程专业、10 个图书馆和信息科学专业。两名学生从课程中随机选择。他们来自同一个部门的概率是多少？他们来自不同部门的概率是多少？

2.3 使用贝叶斯法则来解决以下问题。Milo 生活中三分之一的时间是乘公共汽车去上班，剩下时间乘地铁。公共汽车不太可靠，所以他只能在 50%的时间内工作。如果坐地铁，他 90%的时间都是准时的。考虑到他在某个特定的日子里准时，Milo 坐公共汽车的概率是多少？

2.4 在一种基于 52 张牌的游戏中，抽出一张牌。根据牌的类型，要么是赢，要么是输。如果这张牌是 4 张 A 牌之一，赢 10 美元。如果这张牌是 4 张国王中的一张，赢 5 美元。如果这张牌是 11 张不是国王或 A 牌的钻石牌中的一张，赢 2 美元。否则输 1 美元。在一张一张牌之后，预期的奖金或损失是多少？（你会玩吗？）

2.5 想想前一个问题所描述的游戏。假设有两张 A 牌被抽走，留下 50 张牌。下一张的期望值是多少？

2.6 在信息论部分，我们在讨论熵时定义了三个随机变量 X、Y 和 Z。我们比较了 $H(Y)$ 和 $H(Z)$。那么 $H(X)$如何与其他两种熵相比？

2.7 在信息论部分，我们比较了单词 the 和 unicorn 的熵。一般来说，什么类型的单词有较高的熵？什么类型的单词有较低的熵？举例来说，在所有的文档中，单词 the 出现在 10 篇文档中，单词 unicorn 出现在 5 篇文档中，Mercury 出现在 1 篇文档中。那么每一个单词的熵是多少？ 37

2.8 头脑风暴的一些不同的特征可能对本章描述的情绪分类任务很有帮助。这些特征的优点和缺点是什么？

2.9 考虑以下场景。你正在编写面部识别软件，该软件可以识别所给图像中是否有一张脸。你有具有正确答案的 10 万张图片，需要确定新的图片中是否有脸。

回答下列问题：

(a) 是监督学习还是非监督学习？

(b) 我们预测的标签或值是什么？

(c) 是二分类还是多类分类？（或都不是？）

(d) 这是一个回归问题吗？

(e) 可以使用哪些特征？

2.10 考虑以下场景。你正在写一篇作文评分软件，该软件可以输入学生的文章并产生 0～100 之间的分数。为了设计这个系统，你有过去一年的文章，这些文章都是人为评分的。你的任务是用今年的文章作为输入来使用系统。

回答和练习 2.9 一样的问题。

2.11 考虑以下场景。你正在编写一个工具来确定一个给定的 Web 页面是否是一个

- 个人主页
- 到个人主页的链接
- 都不是

为了帮助你完成任务，你得到了 5000000 页已经被标记的页面。

回答和练习 2.9 一样的问题。 38

第3章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本数据理解

本章通过自然语言处理(NLP)来介绍文本数据涉及的基本概念。NLP关注的是开发计算技术使计算机能够理解自然语言文本的含义。NLP是文本信息系统的基础，因为如何有效帮助用户访问和分析文本数据在很大程度上取决于系统能否理解文本数据的内容。因此，在逻辑上，内容分析是文本数据分析和管理的第一步。

一个人可以立即用母语理解一个句子，但对于计算机就是一个很有挑战性的问题。一般来说，这可能涉及以下任务。

词法分析(lexical analysis)。词法分析的目的是弄清楚什么是一种语言中的基本语义单位(例如英文单词)并确定每个词的意思。在英语中，很容易确定文字的边界，因为它们被空格隔开，但是确定其他语言的文字边界并不简单，如汉语就没有明确的分隔符来区分单词。

句法分析(syntactic analysis)。句法分析的目的是确定句子中的单词是如何彼此关联起来的，从而揭示句子的句法结构。

语义分析(semantic analysis)。语义分析的目的是确定一句话的意思。这通常需要根据单词的含义及句法结构，计算整个句子或更大语义单位的含义。

语用分析(pragmatic analysis)。语用分析的目的是确定在上下文语境中的意义，例如推断语言的言语行为。人们用自然语言彼此沟通，比语义分析更深入的自然语言理解是进一步了解沟通的目的。

篇章分析(discourse analysis)。篇章分析需要处理大量包含有多个句子的文本。在这种情况下，必须考虑这些句子之间的关联，而且必须放在涉及其他句子的适当语境中分析一个句子。

图3-1展示了理解一个非常简单的英语句子所涉及的处理模块。句子内容是“A dog is chasing a boy on the playground.”。词法分析包括确定所有单词的句法类别(词性)(例如，dog(狗)是一个名词，chasing(追逐)是一个动词)。句法分析就是要确定a和boy形成一个名词短语，the和playground也形成一个名词短语，而on the playground构成一个介词短语。语义分析是将名词短语映射到实体和将动词映射到关系的过程，以获得一个形式化的句子意思的表示。例如，名词短语a boy可以映射到一个表示男孩的语义实体(即b1)，同样，a dog可被映射到一个表示狗的实体(即d1)。动词短语可以映射到关系谓词chasing(d1，b1，p1)。请注意，有了这个层次的理解，还可以基于任何相关的常识知识推断出额外的信息。例如，如果我们假设有人被追赶，他或她可能会害怕，我们可以推断被追赶的男孩(b1)可能会感到害怕。最后，语用分析可以进一步揭示说这句话的人的意图可能是请求一个行动，如提醒狗主人把狗带回去。

虽然为图3-1中展示的类似简单句子导出清晰的语义表示是可能的，对不受限制的自然语言文本进行此类分析通常是非常具有挑战性的。造成困难的主要原因是因为自然语言被设计的目的是使人们能够高效的交流，这与编程语言形成鲜明对比，设计编程语言是为了方便计算机理解。具体来说，两个原因使自然语言处理非常难。1) 自然语言交流中省

略了许多“常识”知识，因为我们假设听者或读者拥有这样的知识(因此不需要明确地传达它)。2) 我们保留了很多歧义，并假设听者/读者知道如何消解(因此不需要浪费多余的话来澄清)。结果，自然语言文本充满了歧义，而解决歧义一般会涉及大量的基于常识性知识的推理，这是人工智能中一个普遍的难题。从这个意义上说，NLP 是“AI 完全的”，即与人工智能中的任何其他难题一样难。

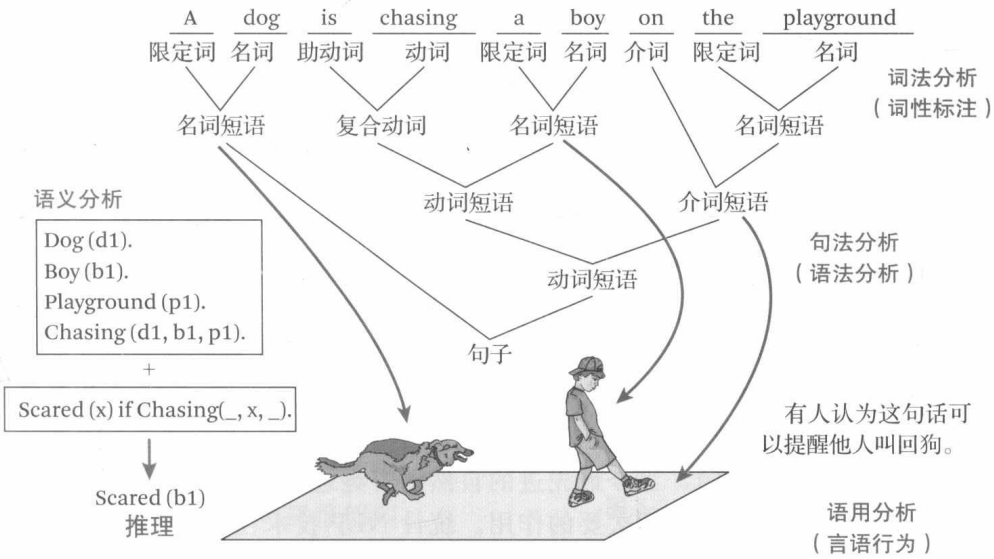


图 3-1　自然语言理解中的层次任务示例

以下是自然语言理解中具体挑战的一些例子。

词级歧义(word-level ambiguity)。一个单词可能有多个句法类别和多个含义。例如，设计(design)可以是名词或动词(词性歧义(ambiguous POS))；根(root)即使作为一个名词也有多重含义(含义歧义(ambiguous sense))。

句法歧义(syntactic ambiguity)。一个短语或句子可能有多个句法结构。例如，自然语言处理可以有两种不同的解释：“自然语言的处理”(processing of natural language)与“语言的自然处理”(natural processing of language)(修饰歧义(ambiguous modification))。又如：A man saw a boy with a telescope 有两个不同的句法结构，这导致了关于谁有望远镜的不同结果(介词短语歧义(ambiguous prepositional phrase(pp) attachment))。

指代消解(anaphora resolution)。代词指代的对象不清楚。比如 John persuaded Bill to buy a TV for himself 一句中，himself 指 John 还是 Bill?

预设(presupposition)。他戒烟了意味着他以前抽烟；以一种通用的方式做出此类推论是困难的。

3.1　自然语言处理的历史和研究现状

NLP 的研究最早可以追溯到 20 世纪 50 年代，当时研究人员对拥有懂得人类语言的计算机持非常乐观的态度，特别是以机器翻译为目的。然而很快，正如 Bar-Hillel 在 1960 年的报告中指出，如果没有知识，全自动的高质量翻译不可能完成。也就是说，一本字典是不够的，我们需要一个百科全书。

研究人员意识到机器翻译可能过于困难。在 20 世纪 60 年代末和 70 年代，一些没有这样困难的 NLP 应用取得了一些成功，但是开发的技术难以扩展，只有有限的应用影响。

例如，以转录语音为目的的语音识别只需要有限地理解自然语言，搞清楚精确的句法结构对于语音识别可能不是非常重要的。因而，此类任务是比较现实的。有两个有趣的项目值得一提：它们清晰地表现出计算机的自然语言理解能力，值得一提。一个是 Eliza 项目，使用浅层规则使计算机能够扮演心理治疗师的角色，能够与人进行一定的自然语言对话。另一个是 block world 项目，展示了当语言限于一个只有积木(block)作为对象的玩具领域，自然语言的深层语义理解是可行的。

在二十世纪七八十年代，研究关注于处理现实世界的自然语言文本数据，特别是故事理解。许多形式化的知识表示和启发式推理规则得到了发展。不过一般结论是，即使是简单的故事也很难被一台计算机理解，确认了对不确定性情况下大规模知识表示和推理的需求。

20 世纪 80 年代后，研究人员开始摆脱传统的符号主义(基于逻辑)自然语言处理方法，这些方法大部分已经被证明在实际应用中不够健壮，转而更注重统计方法。统计方法首先在语音识别领域取得了巨大成功，随后在几乎所有其他的 NLP 任务中遍地开花。与符号
42 方法相比，统计方法更为健壮，因为它们对人工产生的规则的依赖程度较低，通常利用语言的经验性使用中体现的规律和模式，基于人标注的训练数据和应用机器学习技术来实现。

尽管语言知识总是有用的，当今最先进的自然语言处理技术往往依赖于大量使用统计机器学习技术，而语言知识仅起到次要的作用。统计 NLP 技术在一些 NLP 任务中取得了成功。词性标注(part of speech tagging，POS tagging)是一项相对容易的任务，最先进的 POS 标记器可能拥有非常高的精确率(在新闻数据上可超过 97%)。虽然句法分析更难，但部分解析可以达到相当高的精确率(例如，名词短语识别可以超过 90%)[⊖]。

然而，完整的结构解析仍然非常困难，主要是因为歧义。语义分析更加困难，只有某些方面取得一定成功，尤其是信息提取(识别命名实体，如人和组织的名称；识别实体关系，如谁在哪个组织中工作)、词义消歧(辨别一个词的不同含义在不同语境下的使用)以及情感分析(识别产品评论中对产品的正面评价)。推理和言语行为分析通常只在非常有限的领域内才可行。

总之，只有自然语言处理的“浅层”分析能够对任意文本以较为健壮的方式进行处理；“深层”分析往往不能很好地扩展或不能足够健壮地分析不受限制的任意文本。在许多情况下，只有人工标注足够规模的训练数据才能达到合理的精度。

3.2　自然语言处理和文本信息系统

文本信息系统应用对稳健性和效率要求较高，通常健壮的浅层 NLP 技术往往比脆弱的深层分析技术更有用。由于 NLP 的难度导致的不可避免的分析错误可能会损害应用的性能。一些文本信息系统采用各种方式绕过那些试图准确理解自然语言文本含义的困难任务，而直接优化任务本身的表现。在这些系统中，深度 NLP 分析的价值被进一步削弱。
43 所以，改进 NLP 技术一般能够提高文本信息系统的性能，但缺乏 NLP 能力不一定是一些应用任务的主要障碍，特别是文本检索。与更困难的任务(如机器翻译)相比，文本检索是一个相对容易的任务。而机器翻译则明显更加依赖于对自然语言的深层分析。

⊖ 这些数字是在一个特定的数据集上的评价结果，因此它们或许并不能完全推广到其他数据集，即使数据的领域是相同的。

图 3-2 显示了一些文本信息系统的任务，这些任务需要不同层次的 NLP 技术。检索和分类等任务相对容易，在大多数情况下，它们不需要很深的 NLP 技术；确实，通过文中提到的关键字往往足以判断一个文档与查询或某个主题是否相关。但在另一方面，诸如机器翻译和问题回答等任务将需要很多更精确的理解。例如对一个句子的错误解析一般会导致错误的翻译，除非目标语言具有类似的歧义结构；同样，对问题的错误理解也会导致错误的答案。

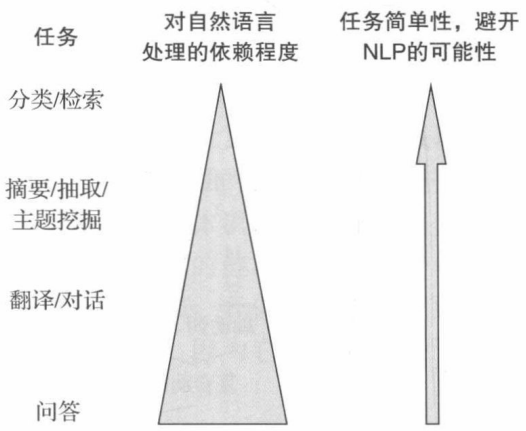


图 3-2　“容易”与“困难”的 NLP 应用

当涉及具体应用任务时，通常可以绕过准确理解自然语言的难题，直接去解决应用问题。一个众所周知的例子是 Eliza 系统[⊖]，它预设为扮演治疗师的角色，并能够与用户进行对话[Weizenbaum1966]。以下是一个对话实例。

人：男人都是一样的。

伊丽莎：以什么方式？

人：他们总是在烦扰我们。

伊丽莎：你能想一个具体的例子吗？

人：那么，我的男朋友让我来到这里。

伊丽莎：你的男朋友让你来这里？

人：他说我大部分时间都很郁闷。

伊丽莎：听说你很郁闷，我很抱歉。

人：这是真的。我不开心。

伊丽莎：你觉得来这里会帮助你高兴吗？

人：我需要一些帮助，这似乎很确定。

伊丽莎：如果你有一些帮助，那对你意味着什么？

人：也许我可以学会与我的母亲相处。

伊丽莎：告诉我更多关于你家庭的信息。

从表面上看，对话似乎是相当自然的，这样的对话确实可能促进抑郁症患者参与交流。然而，系统实际上并不是真正理解了语言，而是完全依赖于类似下面的启发式规则来维持对话：

我记得 X→你经常想到 X？

总是→你能想到一个具体的例子吗？

这样的规则使系统能够直接执行任务，即进行对话，即使它并没有试图去理解单词的真实含义和确定整个句子的含义。

这种基于模版的解决问题的方式已经变得相当强大。事实上，现代机器学习方法对自然语言的理解基本上是以这个为基础的，在很多方面都和 Eliza 系统类似，但有两个重要的区别。首先是机器学习系统中的规则并不精确或严格；相反，它们往往是随机的，而选择哪个规则的概率将根据训练数据集经验地设定，训练数据中包括了待计算函数的预期结果。其次，与用户提供规则不同，“软”规则可以从训练数据中自动学习到并且只需要很少的用户参与，例如在规则中指定元素。

甚至像机器翻译这样的困难任务也可以通过这样的统计方法实现。建立文本信息系统

⊖ http://www.cs.nott.ac.uk/~pszgxk/courses/g5aiai/002history/eliza.htm

最有用的 NLP 技术都是统计方法，它们往往比符号方法更加强大。统计语言模型尤其有用，因为它们可以以一定的理论基础来量化与自然语言使用相关联的不确定性。

3.3　文本表示

NLP 技术使我们能够为文本对象设计许多不同类型的信息特征。来看一下图 3-3 中的例句：A dog is chasing a boy on the playground。我们可以用很多不同的方式表示这个句子。首先，总是可以把这样一个句子表示成一串字符。每种语言都是如此。这也许是表示文本的最一般的方式，我们总是可以使用这种方法来表示任何文本数据。遗憾的是，这种表示的缺点是它不能让我们执行语义分析，而语义分析恰恰是很多文本挖掘应用所需要的。以这种方式，甚至无法识别出单词——任何语言的基本意义单位。（当然，在某些情况下字符也是有用的，但并非一般情况。）

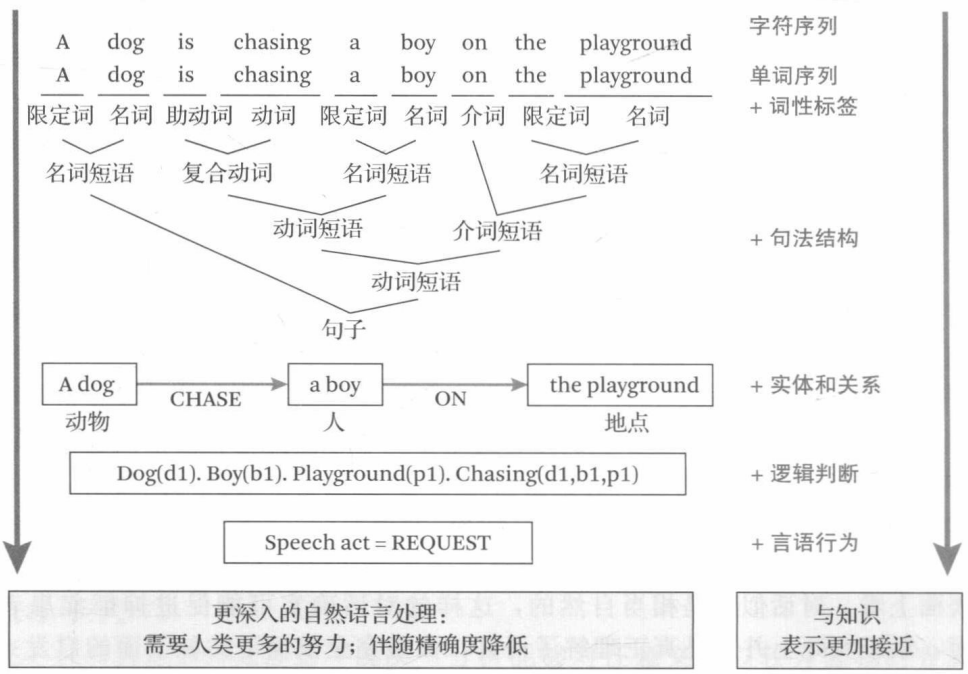


图 3-3　不同层次文本表示的展示

下一个文本表示方案是对文本进行分词以获得一个单词序列。在例句中，我们得到了像 dog 和 chasing 这样的特征。这一层次的表示突然赋予我们更多的自由。例如，通过识别单词，可以轻松地发现这个文件或整个文档集中最常用的单词。这些词可以用来形成主题。因此，将文本数据表示为词的序列为一系列有趣的分析提供了可能性。

但是，与使用字符相比，词语层次的表示的普适性就要弱一些。在一些语言中，如中文，并不容易识别单词的边界，因为中文文本是一系列的字而词之间没有空格。要解决这个问题，我们不得不依靠一些特殊的技术来执行更高级的文本分割，而不是仅基于空格（实际上，基于空格分词也并不总是 100%准确）。所以，词序列表示不如字符串表示那样健壮。对英语而言，获得词层次的表示是非常容易的，因而我们可以一直使用这一层次的表示。

如果进一步进行自然语言处理，我们可以给词语添加词性（POS）标签。这样可以统计，例如最常见的名词；或者，可以确定什么样的名词与什么样的动词相关联。这为进一

步分析开辟了更多有趣的机会。请注意在图 3-3 中，用加号来表示增加了新的特征，因为能够将文本表示为词性标签的序列并不意味着必须替换原来的单词序列。相反，这可以视为表示文本数据的附加方式。

单词和词性标签双重文本表示丰富了文本数据的表示，从而能实现更深入的分析。如果走得更远一些，可以对句子进行句法分析得到一个句法结构。再一次，这进一步提供了进行更有趣的分析的机会，比如分析写作风格或者改正语法错误。

如果继续进行语义分析，那么可能会识别出 dog 是一种动物，也可以识别出 boy 是一个人，playground 是一个地点并分析这些概念之间的关系。一个推测可能是狗在追男孩，男孩在操场上。通过类似这种实体和实体关系识别，能够增加更多的实体和关系。现在，我们可以统计整个新闻报道集合中出现次数最多的人。或者，每当文中提到某个人，另一个人或物体往往也经常在附近。这些类型的重复模式可以成为很好的特征。

46
~
47

这样的高层次表示不如单词序列或词性标签健壮。对所有实体识别出正确的类型并不容易，我们可能会犯错。而实体间的关系更难发现。该层次表示的健壮性弱一些，但是它非常有用。如果我们进一步到逻辑表示，那么有谓词和推理规则。我们运用推理规则可以从文本中推断有趣的事实。我们不能为所有的句子一直这样做，因为它可能会花费惊人的计算时间或大量的训练数据。

最后，言语行为还会为句子的内容增加另一层次的表示。在这个例子中，它可能是一个请求。知道这样的信息后，我们可以分析一下关于观察者或句子的作者的有趣的信息。句子这样表达的用意是什么？什么场景或什么样的行动将会发生？

图 3-3 显示，如果向下移动，通常会涉及更复杂的 NLP 技术。然而这样的技术需要更多的研究者的努力，由于他们试图解决非常难的问题，因此这些技术的健壮性不佳。如果在深层语言分析的层面上分析文本，那么我们必须容忍潜在的错误。这也意味着将这种深度分析与(例如)基于单词序列的浅层分析相结合仍是有必要的。在右边，有一个指向下方的箭头表明当向下发展时，我们对文本的表示更接近于脑海中对知识的表示。这是文本挖掘的目的。

显然，以下二者之间需要一个平衡：深层分析能够从文本获取直接的知识，然而会带来错误；浅层分析无法进行深层次的知识表示，但健壮性高。

文本数据由人类生成，进而被人类使用。因此，在文本数据分析和文本挖掘中，人类扮演着非常重要的角色。他们总是处在这一循环的过程中，这意味着我们应该优化人与计算机之间的合作。从这个意义上说，计算机可能不能产生一个完全准确的文本数据表示。从文本数据中提取出来的模式可以被人类解读，然后人类就可以引导计算机通过标注更多的数据来进行更准确的分析，指导机器学习程序，使它们更有效地工作。

不同的文本表示倾向于启用不同的分析，如图 3-4 所示。我们可以在文本数据表示中逐渐增加更多更深入的分析，这将提供更有趣的表示机会和分析能力。表格总结了我们刚刚看到的内容；第一列列出了文本表示的不同类型，第二列对文本表示的通用性进行了可视化表示。通用性指能够对所有的文本数据进行准确表示(通用性好)或者只能对一些进行准确表示(通用性一般)。第三列显示可用的分析技术，最后一列显示了通过一定程度的表示能够完成的应用。

48

作为字符序列，文本只能通过字符串处理算法来处理。这类算法的健壮性和通用性都较强。在压缩应用中，我们不需要知道单词边界(虽然单词边界是有帮助的)。单词序列(而不是字符)是很重要的表示；有较强的通用性和健壮性，支持词语关系分析、话题分析

和情感分析等多种分析技术。正如你所料，许多应用程序可以通过这些分析变得可行。例如，词汇表发现与发现相关词汇有关系，主题和意见相关的应用也需要基于词表示。人们可能希望了解大量文本集合中蕴含的主题，其中主题被表示为词的分布。

文本表示	通用性	可用分析方法	应用举例
字符串		字符串处理	压缩
单词		词语关系分析；主题分析；情感分析	辞典发现；主题/观点相关的应用
+句法结构		句法图分析	语体分析；结构特征抽取
+实体和实体间关系		知识图谱分析；信息网络分析	关于特定实体的知识发现及观点挖掘
+逻辑谓词		零散知识的综合分析；逻辑推理	对生物学家的知识辅助

图 3-4　文本表示和它们所支持的分析技术

向下移动，我们会看到我们可以逐渐添加其他表示。通过增加句法结构，可以进行句
49 法图分析；我们可以用图挖掘算法来分析这些句法图。例如，文体分析通常需要句法结构表示。我们也可以生成基于结构的特征，有助于通过查看不同的句法结构将文本对象分类到不同的类别。如果你想将文章分类成不同的作者的相应类别，一般需要查看句法结构。当我们添加实体和关系时，可以启用其他技术，如知识图谱或信息网络。使用这些更高级的特征表示允许应用处理实体。

最后，当我们添加逻辑谓词时，我们可以对分散的知识的分析进行整合。例如，我们可以在从文本中提取的信息之上添加一个本体，进而做出推论。这个级别表示所产生的应用的实例有生物学家的知识助手。这个系统可以管理所有和研究问题相关的知识，例如管理与理解基因功能这一研究问题相关的知识。计算机可以做一些生物学家可能感兴趣的推断假设。例如，它可以通过阅读文献查找相关事实来确定基因是否具有一定的功能。它可以使用逻辑系统来追踪研究者的问题“什么基因与什么功能有关”的答案。为了支持这个级别的应用程序，我们需要尽可能地使用逻辑表示。

本书涵盖的技术主要侧重于基于词的表示。这些技术具有较好的通用性和健壮性，并广泛用于各种应用。实际上，在几乎所有的文本挖掘应用中，都需要这种表示。不过，其他层次可以结合起来，以支持更多的语言复杂的应用。

3.4　统计语言模型

统计语言模型(或简称语言模型)是一个在单词序列上的概率分布。因此它给出了任何单词序列一个不同的概率。例如，一个语言模型可能会给出以下三个单词序列不同的概率：

$$p(\text{Today is Wednesday}) = 0.001$$

$$p(\text{Today Wednesday is}) = 0.000\,000\,001$$

$$p(\text{The equation has a solution}) = 0.000\,001$$

显然，语言模型可以是上下文相关的。在如上所述的语言模型中，序列“The quation

ehas a solution”出现的概率小于“Today is Wednesday”。描述一般场景的对话，这个模型可能是合理的，但是在一个数学会议上可能是不精确的，序列“The equation has a solution”将会比“Today is Wednesday”更频繁地出现。 50

给定一个语言模型，我们可以根据分布对单词序列进行抽样，从而获得一个文本样本。从这个意义上说，我们可以用这样的模型来“生成”文本。因此，语言模型通常也被称为文本的生成模型。

为什么语言模型有用？一般的答案是它提供了一个有依据的方法对使用自然语言相关的不确定性进行量化。更具体地说，它使我们能够回答许多与文本有关的文本分析和信息检索的有趣问题。一个语言模型可以帮助回答问题，以下是一些问题的例子。

- 已知我们看到单词“John”和“feels”，之后有多大可能会看到单词“happy”而不是“habit”？因为“happy”和“habit”具有非常相似的声学信号，回答这个问题可以帮助语音识别，但是语言模型可以很容易地指出，“John feels happy”比“John feels habit”出现概率更大。
- 已知我们在新闻文章中看到三次“棒球”一次“比赛”，这篇报道是关于运动的话题的概率有多大？这显然会直接有助于文本分类和信息检索任务的解决。
- 已知用户对体育新闻感兴趣，用户在查询中使用“棒球”一词有多大的可能性？这个问题与信息检索直接相关。

如果我们列举所有可能的单词序列并给出每个序列的概率，模型的评价会较为复杂，原因在于无限数量的单词序列将导致待估计的参数的数量也趋于无穷。也就是说，我们永远没有足够的数据去估计这些数据参数。因此，我们必须做出假设来简化模型。

最简单的语言模型是一元语言模型，假定单词序列是由每个单词相互独立产生的。因此，一个单词序列的概率将等于每个单词的概率的乘积。在形式上，V 表示词汇集合，w_1，…，w_n是一个单词序列，其中 $w_i \in V$ 是一个单词。我们有

$$p(w_1,\cdots,w_n)=\prod_{i=1}^{n}p(w_i) \tag{3-1}$$

给定一个一元语言模型 θ，其中参数数量与词汇表中的词汇数量一样多，满足约束 $\sum_{w\in V}p(w)=1$。这样一个模型本质上指定了所有单词的多项分布。 51

给定一个语言模型 θ，一般来说，生成两个不同文档 D_1 和 D_2 的概率将是不同的，即 $p(D_1|\theta)\neq p(D_2|\theta)$。什么文档会有更高的概率？直觉就是那些包含很多 $p(w|\theta)$中高概率词的文档。从这个意义上讲，θ 中的高概率词可以体现 θ 捕获到的主题。

例如，图 3-5 所示的两个一元语言模型就表明了这一点，其中一个关于“文本挖掘”的话题和另一个关于“健康”的话题。直观地说，如果 D 是文本挖掘论文，我们可以期望 $p(D|\theta_1)>p(D|\theta_2)$，而如果 D'是一个讨论饮食控制的博客文章，我们会期待相反的结果：$p(D'|\theta_1)<p(D'|\theta_2)$。我们也可以期望 $p(D|\theta_1)>p(D'|\theta_1)$和 $p(D|\theta_2)<p(D'|\theta_2)$。

现在假设我们已经观察到一个文档 D(例如一篇关于文本挖掘论文的简短摘要)，假定采用该文档由一元模型 θ 所生成。我们想基于文档 D 推导出模型 θ(即估计每一个单词的概率 $p(w|\theta)$)。这是在统计中的一个典型问题，称为参数估计，该问题可以用很多不同的方法来解决。

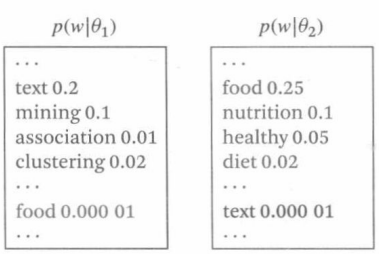


图 3-5　一元语言模型的两个例子，代表两个不同的主题

一个常见方法是最大似然(Maximum Likelihood,

ML)估计，寻求一个能给予观察数据最大的可能性(即对数据具有最佳解释性)的模型$\hat{\theta}$：

$$\hat{\theta}=\arg\max_{\theta} p(D|\theta) \tag{3-2}$$

很容易证明，一个一元语言模型的最大似然估计对每一个单词给出了概率，其概率等于单词在 D 中的频率。

$$p(w|\hat{\theta})=\frac{c(w,\ D)}{|D|} \tag{3-3}$$

52

其中 $c(w,\ D)$是单词 w 在文档 D 中的计数值，$|D|$是 D 的长度或 D 中的单词总数。

在最大化观察到的数据出现概率的意义上说，这样的估计方法是最优的。但是对于特定应用，它是否真的是最优仍然存疑。例如，如果我们的目标是估计研究报告的语言模型，我们仅基于论文的摘要运用最大似然估计来估计模型，那么显然不是最优的。因为模型将把摘要中没有出现的词都分配零概率(除非文章只使用摘要中出现过的单词,)这将使整个文章被分配零概率。请注意，一般来说，最大似然估计将为任何在观察数据中没有出现的词项或事件分配零概率；这是因为给这样一个词项或事件分配非零概率，将削减可观察词的概率(因为所有的概率必须总和为 1)，从而减少了观察到的数据出现的可能性。我们将稍后讨论各种对最大似然估计进行改进的技术，它们被称为平滑技术(smoothing)。

虽然非常简单，但一元语言模型在文本分析任务中很有用。例如，图 3-6 显示了三种不同的一元语言模型对三种不同的文本数据样本的估计，即普通英文文档数据库、关于计算机科学的研究论文数据库和文本挖掘研究论文。一般而言，三种模型中概率最高的词是英语中的功能词，因为这些词在任何文本中均频繁使用。再往下看单词列表，会看到更多包含实质内容和主题的词汇。这些词汇在不同的用于估计的数据中有不同的分布，因此可以用于在不同文本样本中区分主题。

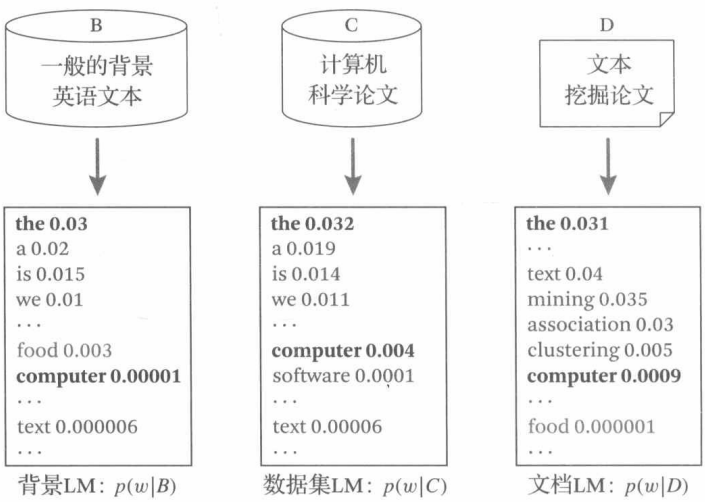


图 3-6　三个不同的语言模型，表示三个不同主题

一元语言模型也可以用来进行单词关系的语义分析。例如，我们可以使用它们来查找哪些单词在语义上与 computer 这个词关联。这样做主要是看看其他什么词往往与 computer 一词共现。具体来说，我们可以先获得一些提到 computer 的文档(或句子)的样本。那么我们可以估计一个基于这些样本的语言模型来获得 $p(w|\text{computer})$。这个模型告诉我们哪些词经常出现在 computer 的语境中。然而，根据这个模型最常见的词语可能是英文中的功能词语或者文档中常见的单词，但与 computer 没有强关联性。为了过滤这些常见的词

语，需要一个模型可以告诉我们哪些词应该被过滤。很容易看到的一般英语语言模型(即背景语言模型)将满足这一要求。所以我们可以使用背景语言模型来规范化模型 $p(w|\text{computer})$，并且获得每个词的概率。假定与 computer 在语义上更相关的词，倾向于在其上下文中频繁出现，但在普通文档中一般不常出现，如图 3-7 所示。

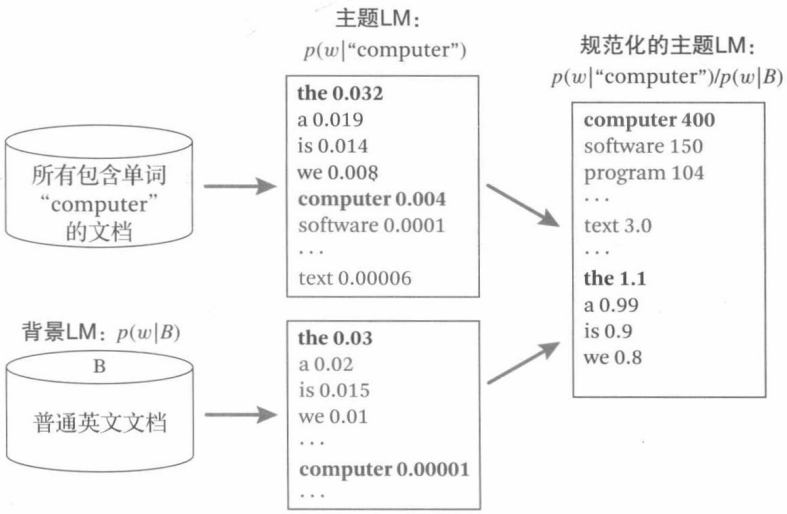


图 3-7　使用主题语言模型和背景语言模型发现语义相关词汇

语言模型在文本信息系统中的应用将在之后讨论，因为在后面的章节中会涉及它们的具体应用。例如，我们可以通过某种语言模型对文档和查询进行表示。鉴于这个背景，读者应该对这个强大的统计工具储备足够的信息，从而支持对未来章节的深入理解。

书目说明和延伸阅读

有许多关于 NLP 的教科书，包括《Speech and Language Processing》[Jurafsky 和 Martin 2009]、《Foundations of Statistical NLP》[Manning 和 Schütze 1999]和《Natural Language Understanding》[Allen 1995]。统计语言模型的深入介绍可以在《Statistical Methods for Speech Recognition》[Jelinek 1997]一书中找到。Rosenfeld[2000]对统计语言模型进行了简要而全面的概述。Zhai[2008]详细讨论了统计语言模型在信息检索中的应用，其中一些内容将在本书后面的章节中介绍。信息抽取作为 NLP 中的一个重要研究主题，在本章没有被过多介绍。在 Sarawagi [2008]中可以找到这个主题的全面介绍，Jiang[2012]对该主题做了非常好的综述。在信息检索的背景下讨论该主题，可以阅读 Moens [2006]的书。

练习

3.1　NLP 与文本挖掘有什么关系？

3.2　性能差的 NLP 是否意味着较差的检索性能？请说明。

3.3　已知一个特定主题的文档集合，如何利用最大似然估计来创建主题一元语言模型？

3.4　文档集合的大小如何影响语言模型的质量？

3.5　为什么最大似然估计不是主题语言模型的最佳参数估计方法？

3.6　假设我们将主题集合的副本附加到自身并重新估计一个最大似然语言模型。θ 会改变吗？

3.7　本章定义的一元语言模型可以将一个单词序列作为输入并输出其概率。解释这个计

算为何具有很强的独立性假设。

3.8　给定一个由本书文本产生的一元语言模型 θ，以及两个文本信息 d_1 = 信息检索、d_2 = 检索信息，那么 $p(d_1|\theta) > p(d_2|\theta)$ 正确吗？试说明原因。

3.9　一个 n 元语言模型记录的是 n 个词的序列。如果用二元语言模型代替一元语言模型，参数的数目将如何变化？如果使用三元语法模型呢？用 V（一元词汇表的词汇数目）给出答案。

3.10　使用你最喜欢的编程语言，用最大似然估计实现一个一元语言模型。通过读取单个文本文件并用空格分隔单词来完成此操作。

3.11　根据上一练习中的概率对单词进行排序。如果你使用了不同的文本文件，那么你的排序列表将有什么不同？怎么才能使二者一致？

56

MeTA：一个面向文本数据管理和分析的统一工具箱

本章介绍附带的软件 MeTA，它是一个免费的开源工具箱，可以用来分析文本数据。我们在本书中给出了很多 MeTA 的上机操作练习，用来练习概念和尝试不同的文本挖掘算法。

本书中讨论的大多数算法和方法都可以在 MeTA 工具箱中以某种形式找到。如果 MeTA 没有包含本书中讨论的某个技术，那么可能是在某一章的练习中需要你自己去实现。尽管 MeTA 是一个功能强大的工具箱，但它的简单性使得功能增加相对简单，通常通过简单扩展类层次结构来实现。

配置文件是 MeTA 前向基础架构的总控部分。因此，一些探索性分析通常不需要用户进行编程，仅需操作配置文件。默认情况下，MeTA 被打包为各种可执行文件，它们可以用来解决某个特定的任务。例如，对于分类实验，用户将在其终端中运行以下命令[⊖]：

```
./classify config.toml
```

这是使用默认的可执行文件的标准程序；它们只需要一个配置文件参数。本章后面会详细解释配置文件格式。本质上，配置文件允许用户选择数据集、明确数据集分词的方式和(针对本例)指定使用某个具体的分类算法。

如果需要更高级的功能，需要使用 C++ 编程来调用 MeTA 的应用程序编程接口(Applications Programming Interface，API)。配置文件和 API 使用文档可在 MeTA 的网站 https://meta-toolkit.org 和本章中找到。另外，我们也有一个围绕 MeTA 工具箱进行探讨的论坛(https://forum.metatoolkit.org)。它包括用户支持主题、社区书面文档和开发人员讨论。

接下来将深入介绍 MeTA 的某些特定方面，以便读者在以后的章节中能方便地使用它。

4.1 设计原则

MeTA 的目标是改进和补充当前的开源机器学习和信息检索软件。这些开源软件的生存环境趋于分散。

很少有一个工具箱包括各种不同的算法；一个很好的例子是支持 SVM 的 LIBLINEAR [Fan 等 2008]软件包。虽然这是线性 SVM 的开源实现中被引用最多的，但它只关注没有使用核方法的情形。如果要处理非线性分类问题，人们就不得不寻找支持核方法的其他软件包(比如同一作者的 LIBSVM 软件包[Chang 和 Lin 2011])。这给研究人员和学生们带来了不必要的负担——他们不但需要对手里的研究问题有详细的了解，而且现在被迫去理解开源软件社区的这种碎片化的本质，在这些鱼龙混杂的实现中找到合适的工具，并且编译和配置合适的工具。

即使完成了这些工作，仍旧有数据格式的问题——这些工具不太可能遵循共同的输入格式标准，因此数据预处理又会消耗很多时间。这些环节都有损于手头的实际任务，显著

⊖ 运行默认分类实验需要的数据集操作。这个 20newsgroups 数据集是默认在 MeTA 的配置文件中的，可以在这下载：https://meta-toolkit.org/data/20newsgroups.tar.gz。把它放在 **meta/data/** 目录下。

影响知识发现和教育学习的速度。

META 解决了这些问题。特别是它为文本索引和分析方法提供了一个统一的框架，允许用户快速进行可控的实验。META 将特征生成、实例表示、数据存储格式和算法的实现
58 模块化；这将允许研究人员和学生以最小的努力在这些维度上进行无缝转换。

META 的模块化支持探索，鼓励贡献，并增加其内部运作的可见性。这些事实使 META 成为本书的完美伴侣工具箱。正如本章开头提到的，读者可以通过那些练习为工具箱增加真正功能。在阅读了本书并学习了文本数据管理和分析之后，读者可继续根据他们新获得的知识来修改 META 以适应他们的文本信息需求。

最后，由于 META 将始终是免费的和开放源码的，所有读者构成一个社区可以一起为增加和提高 META 的功能做贡献，使所有相关人员受益。

4.2　设置 META

本书的所有未来章节都假定读者已经下载并安装了 META。在这里，我们将展示如何设置 META。

META 既有带教程的网站也有 GitHub 上的在线资源库。要真正下载该工具箱，用户将在安装必要的先决条件后，在命令行终端中使用版本控制软件 **git** 查看。

META 网站包含下载和设置特定系统配置的说明。在写这本书的时候，META 支持 Linux 和 Mac OS 操作系统。访问 https://meta-toolkit.org/setup-guide.html 并且按照所需的平台的指导。我们将假定读者已经完成了安装指南中列出的步骤，并获得一个可工作的 META 版本，为本书中的所有练习和演示做好准备。

有两个步骤没有在安装指南中提到。第一个是要确保读者在本书出版时拥有最新版本的 META。为了确保命令和实例与读者下载的软件同步，我们将确保 META 被签出的是版本 2.2.0。在 **meta/** 目录里运行以下命令：

```
git reset --hard v2.2.0
```

第二步是确保下载必需的模型文件。这些都可以在 META 发布页面上找到：https://github.com/metatoolkit/meta/releases/tag/v2.2.0。默认情况下，模型文件应该放在 **meta/**
59 **build/** 目录中，但是只要更改配置文件中的路径，就可以把模型文件放在任何地方。

一旦这些步骤完成，读者应该能够完成任何练习或运行任何演示。如果需要任何额外文件或信息，我们将在相应章节中提供。

4.3　架构

META 中所有的处理数据都被存储在索引中。有两种索引类型：前向索引(**forward_index**)和倒排索引(**inverted_index**)。前者以文档 ID 作为主键，后者以词项 ID 作为主键。

forward_index 被用于像主题建模和大多数分类任务这样的应用。

inverted_index 被用于建立搜索引擎，或者用 k 近邻或其他的类似算法来做分类。

由于每个 META 应用程序都将一个索引作为输入，所有处理的数据在所有组件之间是可互换的。这也给分类带来了很大的好处：META 默认支持 out-of-core 分类！如果数据集足够小(像大多数其他工具的假设)，缓存可以用于，如 **no_evict_cache**，把全部数据保存在内存中而不牺牲任何速度。(索引用法在搜索引擎练习中更详细地解释。)

有 4 种语料输入格式。

line_corpus。每个数据集由以下 1～3 文件组成：

corpusname.dat。每个文档都显示在一行上。

corpusname.dat.labels。可选文件，包括每一行的文档的类别或标签，其顺序与 **corpusname.dat** 一致。这些标签将用于分类任务。

file_corpus。每个文档作为一个单独文件，文件的名称即为文档的名称。还有一个语料库 **corpusname-full-corpus.txt**，其中每一行记录对应文档的类别标签还有文档在磁盘上文件的路径。如果没有类标签，则需要一个占位符标签，例如“[none]”。

gz_corpus。和 **line_corpus** 相似，但它的每一个文件和元数据文件使用 gzip 压缩工具压缩：

corpusname.dat.gz。**corpusname.dat** 的压缩版本。

corpusname.dat.labels.gz。**corpusname.dat.labels** 的压缩版本。

libsvm_corpus。如果仅被用于分类，MeTA 可以拿 LIBSVM 格式的输入来创建一个前向索引 **forward_index**。在 LIBSVM 网站上有许多机器学习数据集都是这种格式的。[1]

有关语料库存储和配置设置的更多信息，我们建议读者参考 https://meta-toolkit.org/overview-tutorial.html。

4.4　用 MeTA 分词

在任何文本数据上创建索引的第一步都是“分词”处理。在高层次上，这意味着将不同的文本文档转化为词频的稀疏向量——这些稀疏向量通常被提供给一个索引器用于为数据集建立倒排索引。

MeTA 把文本分析过程分为几个层次，以便于给用户尽量多的权利来控制文本分析的可能方式。

在大多数情况下一个分析器将采取“过滤器链”。它被用于产生分词过程的最终符号：这个过滤器链总是被定义为一个特殊的分词器类，它后面跟随零个或多个过滤器类。它们中的每一个都读取之前的类别输出。例如，一个简单的过滤器链是把所有的字符都变成小写值保留固定长度的字符：

```
icu_tokenizer → lowercase_filter → length_filter
```

分词(tokenizer)永远是第一位的。它们定义如何将一个文档的字符串分割成词项。下面是一些例子。

icu_tokenizer。遵照 Unicode 标准进行句子和词项分割，将文档的字符串转换为词项序列。

character_tokenizer。把文档字符串转换成单个字符序列。

接下来是过滤器(filter)，可以将多个过滤器链接起来。过滤器定义了文本可以被修改和转换的方式。下面是一些过滤器的例子。

length_filter。这个过滤器接受特定长度以内的词项，拒绝其他的词项。

icu_filter。使用 ICU(Unicode 的国际化组件)[2]对序列中的词项进行转写。例如一个重音字符 $\acute{i}$ 被改写成 i。

list_filter。该过滤器基于一个列表决定接受或者拒绝词项。例如，可以使用一个停用词表来拒绝停用词。

---

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets。

[2] http://site.icu-project.org/；不同的 ICU 版本采用的分词方式可能略有不同。

porter2_stemmer。该过滤器会根据 Porter2 英语词型还原规则[⊖]来转换每一个词项。

分析器(analyzer)在过滤器链的输出上进行操作并且对文档中的词项计数。下面是一些分析器的例子。

ngram_word_analyzer。收集和统计已通过过滤器链过滤的长度为 n 的单词序列。

ngram_pos_analyzer。和 **ngram_word_analyzer** 类似，但该分析器处理词性标签。词性标签由 MeTA 基于 CRF 的词性标注器提供。

tree_analyzer。收集并统计句法分析树特征。

libsvm_analyzer。将 LIBSVM **line_corpus** 格式转换成 MeTA 格式。

MeTA 定义了一个完备的默认过滤器链，鼓励用户在没有特定需求的情况下使用该过滤器链进行一般的文本分析。要使用它，需要在配置文件中指定以下内容：

```
[[analyzers]]
method = "ngram-word"
ngram = 1
filter = "default-chain"
```

该配置文件设定文本分析过程考虑一元词项，通过运行默认的过滤器链可得到这些词项。这个过滤器链应该在大多数语言上都运行良好，因为它所有的操作(包括但不限于分词和句子边界识别)都尽可能地遵照 Unicode 标准。

如果既考虑一元又考虑二元词项，配置文件看起来应该像下面这样：

```
[[analyzers]]
method = "ngram-word"
ngram = 1
filter = "default-chain"

[[analyzers]]
method = "ngram-word"
ngram = 2
filter = "default-chain"
```

62

每一个**[[analyzers]]**块都定义了一个单独的分析器及其对应的过滤器链：像被需求的一样多——每个分析器产生的词项都将被计数并放到一个单独的稀疏计数向量中。这对于将多种不同的特性组合到文档表示中非常有用。例如，下面的配置会融合一元词项、二元词性标签、树结构特征和子树特征。

```
[[analyzers]]
method = "ngram-word"
ngram = 1
filter = "default-chain"

[[analyzers]]
method = "ngram-pos"
ngram = 2
filter = [{type = "icu-tokenizer"}, {type = "ptb-normalizer"}]
crf-prefix = "path/to/crf/model"

[[analyzers]]
method = "tree"
filter = [{type = "icu-tokenizer"}, {type = "ptb-normalizer"}]
features = ["skel", "subtree"]
tagger = "path/to/greedy-tagger/model"
parser = "path/to/sr-parser/model"
```

⊖ http://snowball.tartarus.org/algorithms/english/stemmer.html。

如果一个应用程序需要特定的文本分析操作，可以根据需求修改配置文件来直接指定过滤器链。我们会改变过滤器使其看起来类似于**[[analyzers]]**块，而不是像上面的过滤器那样使用字符参数。每个分析器将有一系列**[[analyzers.filter]]**块，每个块都定义了过滤器链中的一环。所有的过滤器链必须以一个分词器开始。这里有一个针对一元单词模型的过滤器链的例子，它和本节开始的那个例子很像：

```
[[analyzers]]
method = "ngram-word"
ngram = 1
    [[analyzers.filter]]
    type = "icu-tokenizer"
[[analyzers.filter]]
type = "lowercase"

[[analyzers.filter]]
type = "length"
min = 2
max = 35
```

63

MeTA 提供了许多不同的类来支持建立过滤器链。请看 API 文档[⊖]查看更多信息。尤其是，**analyzers::tokenizers** 命名空间和 **analyzers::filters** 命名空间都有很强大的功能。给定类别的静态公共属性 **id** 是配置文件中“类型”所需的字符串。

4.5　相关工具箱

现有的支持文本管理和分析的工具箱一般分为两类。第一类是搜索引擎工具箱，适合构建搜索引擎应用程序，但往往对文本分析/文本挖掘功能的支持有限。典型例子如下。

Lucene。https://lucene.apache.org/。

Terrier。http://terrier.org/。

Indri/Lemur。http://www.lemurproject.org/。

第二类是文本挖掘或通用数据挖掘和机器学习工具箱，它们往往选择性地支持一些文本分析功能，但是通常不支持搜索能力。典型例子如下。

Weka。http://www.cs.waikato.ac.nz/ml/weka/。

LIBSVM。https://www.csie.ntu.edu.tw/~cjlin/libsvm/。

Stanford NLP。http://nlp.stanford.edu/software/corenlp.shtml。

Illinois NLP Curator。http://cogcomp.cs.illinois.edu/page/software_view/Curator。

Scikit Learn。http://scikit-learn.org/stable/。

NLTK。http://www.nltk.org/。

64

然而，现在还缺乏工具能够对搜索引擎功能和各种文本分析功能的无缝集成，这种集成对于建立一个支持文本管理和分析的统一系统是必要的。MeTA 的主要设计理念也是使 MeTA 不同于其他工具箱的部分，是它的重点在于文本分析功能和搜索(确切地说是一般意义上的文本获取能力)的紧密整合，使其对于创建一个强大的文本分析应用能够提供全面的支持。为了促进教育和研究，MeTA 的设计重点强调模块化和可扩展性，并通过面向对象的设计实现。MeTA 可以以多种方式与现有的工具箱搭配使用。例如，对于非常大规模的文本应用程序，使用现有的搜索引擎工具箱支持搜索，而 MeTA 可

⊖ 访问 https://meta-toolkit.org/doxygen/namespaces.html。

以用于进一步支持对搜索结果的分析或从对原始大数据集中获得的任何文本数据子集的分析。NLP 工具箱可以用来对文本数据进行预处理，并生成带标注的文本数据用于 MeTA 模块的输入。MeTA 还可以用于生成文本表示提供给不同的数据挖掘工具箱和机器学习工具箱。

练习

在其最简单的形式中，文本数据可以是一个 **.txt** 格式的单个的文件。这个练习会让你熟悉各种用于文本分析的技术。我们将使用查尔斯·狄更斯(Charles Dickens)的小说《双城记》作为样例文本。这本书叫作 **two-cities.txt** 并被放在 http://sifaka.cs.uiuc.edu/ir/text-databook/two-cities.txt。你也可以使用任何你自己的文件只要它包含多个英语句子。

像所有接下来的练习，我们假定读者按照 MeTA 安装指南并成功地编译执行。在这个练习中，我们只会使用 **profile** 的程序。从目录 build/中运行 **./profile** 将会打印出下面这些使用信息：

```
Usage: ./profile config.toml file.txt [OPTION]
where [OPTION] is one or more of:
        --stem  perform stemming on each word
        --stop  remove stop words
        --pos   annotate words with POS tags
        --pos-replace   replace words with their POS tags
        --parse create grammatical parse trees from file content
        --freq-unigram  sort and count unigram words
        --freq-bigram   sort and count bigram words
        --freq-trigram  sort and count trigram words
        --all   run all options
```

65

如果运行 **./profile** 打印这个信息，那么一切都已经设置正确。我们将会在下面的练习中看看每个选项的意思。

4.1　**停用词去除**。考虑下面的单词：I，the，of，my，it，to，from。如果知道一个文档包含了这些词，你会了解这个文档是在说什么吗？可能不会。这些类型的单词叫作停用词。具体来说，它们是频率很高的但不包含内容信息的词。由于语法的要求，我们需要使用这些停用词，比如来连接句子。

由于这些词不包含任何主题信息，所以它们常常在文本分析中作为预处理步骤被删除。不仅这些无用的词被忽略，而且较少的数据意味着算法可以运行得更加快速！

```
./profile config.toml two-cities.txt --stop
```

现在，使用 **profile** 程序把停用词从 **twocities.txt** 中删除。如果没有这些停用词，你还能了解这本书的内容吗？

4.2　**词干提取**。词干提取就是把一个单词转化成它基础形式的过程。这对搜索引擎特别有用。如果用户想查找有关 running 的书籍，那么包含 run 或 runs 的文档将匹配不到。如果我们对一个词实施词干提取算法，那么在信息检索任务中这个单词的其他形式更可能被匹配到。

最受欢迎的词干提取算法是 Porter2 English Stemmer。它是由 Martin Porter 开发的。这是对 1980 年原来的 Porter Stemmer 来说略为改进的版本。一些示例如下：

{run，runs，running}→run

{argue，argued，argues，arguing}→argu

{lies，lying，lie}→lie

MeTA 默认使用 Porter2 stemmer。你从这里阅读更多关于 Porter2 stemmer 的信息：http://snowball.tartarus.org/algorithms/english/stemmer.html。如果你想玩一玩它，它也有一个在线的演示系统：http://web.engr.illinois.edu/~massung1/p2s_demo.html。

既然你已经知道词干提取是什么，那么可在《双城记》上运行。

```
./profile config.toml two-cities.txt --stem
```

66

像停用词去除一样，词干提取试图保持原始文本背后的基本含义。词干提取之后你还可以了解它们的意思吗？

4.3 词性标注。在学习英语的过程中，学生经常会遇到不同的语法标签，如名词、形容词、动词等。在语言学和计算机科学中，有一个更大的标签体系，称为词性(POS)标签。根据周围的词，我们可以给每一个词赋予一个标签。考虑以下句子：All hotel rooms are pretty much the same, although the room number might change。这里有一个词性标注的版本：

All_{DT} $hotel_{NN}$ $rooms_{NNS}$ are_{VBP} $pretty_{RB}$ $much_{RB}$ the_{DT} $same_{JJ}$,,

$although_{IN}$ the_{DT} $room_{NN}$ $number_{NN}$ $might_{MD}$ $change_{VB}$..

以上 VBP 和 VB 是不同类型的动词，NN 和 NNS 是单数和复数名词，DT 是限定词。这是关于 80 个常用标签的一个子集。不是每一个词都有一个独特的词性标签。例如，flies 和 like 根据上下文可以有不同的标签：

$Time_{NN}$ $flies_{VBZ}$ $like_{IN}$ an_{DT} $arrow_{NN}$..

$Fruit_{NN}$ $flies_{NNS}$ $like_{VBP}$ a_{DT} $banana_{NN}$..

类似情况可以使词性标注非常具有挑战性。然而，人类在词性标注上的一致性是 97%，这是自动标注的上限。

词性标签在文本分析中使用可以作为文本的替代(或附加)表示。这些标签的使用可以捕捉一些文档或语料的语法层面的意义。对于词性标注 **profile** 程序有两个操作。第一个操作像上面的例子中那样来标注每个词，第二个操作是用词性标签替代每个词。

```
./profile config.toml two-cities.txt --pos
./profile config.toml two-cities.txt --pos-replace
```

注意，词性标注这本书可能最多花费你 1 分钟的时间。MeTA 的词性标注看起来是对的吗？你能发现错误吗？当用词性标签来替换单词时，是否可以确定原来的句子是什么？请用本书的文本或者其他任意的文本文件来实验。

4.4 语法分析。语法解析树代表文本句子中的更深层次的句法知识。它们把句子的短语层次表示成一个树结构。请考虑图 4-1 中的示例。

67

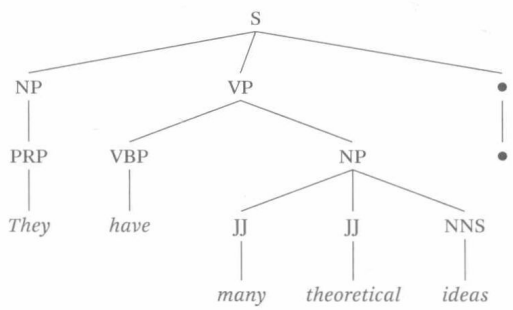


图 4-1　句法解析树的一个例子

这个语法解析树以 S 为根结点，S 代表 Sentence，句子由名词短语(NP)后面跟着一个动词短语(VP)和句号组成。树上的叶子结点是句子中的单词，次终级结点(叶子结点的直接父结点)是词性标签。

语法分析树上的一些共同特征是产生规则(例如 S→NP VP)、树的深度和结构树特征。句法类别(结点标签)也可以使用。

下面的命令将在输入文件的每个句子上运行语法解析器：

```
./profile config.toml two-cities.txt --parse
```

和词性标注类似，句法分析可能需要一到两分钟完成。

4.5 词频分析。也许最常见的文本处理技术就是频率计数。它只是简单的计数每一个单词在文档(或语料库)中出现多少次。查看按词频降序排列的单词列表可以让你了解文档是关于什么的。直观地说，类似的文件应该有一些相同的高频词……当然这其中不包括停用词。

我们还可以以 n 个词的串作为统计的单元(叫 n-grams)，而不是以单个的词作为统计单元。考虑这个句子：I took a vacation to go to a beach.

- 一元词(1-grams 或 unigrams)：

  {I：1，took：1，a：2，vacation：1，to：2，go：1，beach：1}

- 二元词(2-grams 或 bigrams)：

  {I took：1，took a：1，a vacation：1，vacation to：1，to go：1，go to：1，

  68

  to a：1，a beach：1}

- 三元词(3-grams 或 trigrams)：

  {I took a：1，took a vacation：1，a vacation to：1，…}

正如我们将在这个文本中看到的，一元词文档表示是最重要的文本表示。这个计数表示的向量确实有一个缺点：我们丢失单词的顺序信息。这种表示也被称为“词袋”，因为我们只知道每个单词的计数，不能知道上下文或位置。这种一元计数方案可以用词性标签或者源于文档的其他类型标记来代替单词。

使用以下三个命令来在文档上做 n-gram 频率分析，$n \in [1, 3]$。

```
./profile config.toml two-cities.txt --freq-unigram
./profile config.toml two-cities.txt --freq-bigram
./profile config.toml two-cities.txt --freq-trigram
```

这将给输出文件 **two- cities.freq.1.txt** 这个选择 **- - freq- unigram**，等等。

什么使得输出相当清晰？请回顾停用词和词干提取。删除停用词是去掉了那些不提供任何文档内容信息的高频词。词干提取将把所有的变形累加到一个单一计数上。这意味着局部向量{run：4，running：2，runs：3}可以表示为{run：9}。这不仅使人们更容易解释词频分析，而且它还可以改进文本挖掘算法！

4.6 齐普夫定律。在英语中，4 个最高频的词大约占所有单词出现量的 10%～15%。频率最高的前 50 个词占了所有单词出现量的 35%～40%。事实上，在所有的人类的语言上都有类似的趋势。回想那些停用词。这些都是最高频的词占了文本的大部分内容。同时，也有很多单词在一个给定文档中只出现一次。

我们以 x 轴为单词的排序，y 轴为单词的频率计数来画图。这样的图可以给我们一个给定文档或集合中单词分布的直观概念。在图 4-2 中，我们从狄更斯的另一

本书《雾都孤儿》中统计一元单词。左边的图是一个正常的 $x \sim y$ 图，右边的是 $\log x \sim \log y$ 图。

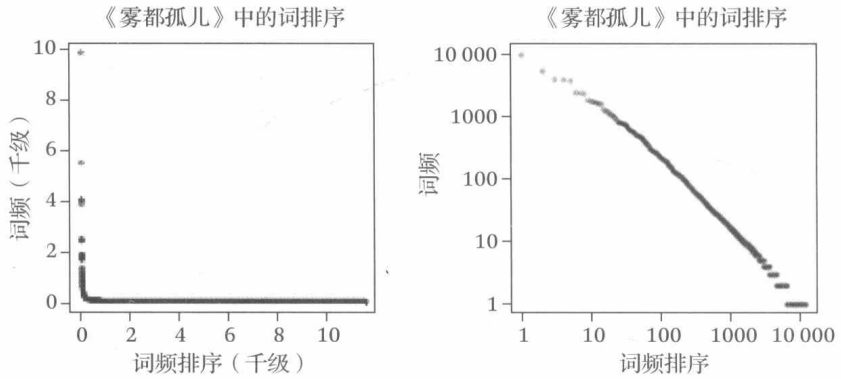


图 4-2　齐普夫定律样例图

齐普夫定律描述这些图的形状。你认为齐普夫定律表达的是什么？这些图形的形状使得我们可以应用某些技术来利用自然语言中的单词分布。

69
~
70

第二部分

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本数据获取

第 5 章　文本数据获取概述
第 6 章　检索模型
第 7 章　反馈
第 8 章　搜索引擎实现
第 9 章　搜索引擎评价
第 10 章　网络搜索
第 11 章　推荐系统

第 5 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本数据获取概述

文本数据获取是文本分析的基础。文本获取技术在文本管理及分析的诸多应用中起到两个重要的作用：其一，文本获取技术能够为特定的分析问题检索到最相关的文本数据，进而避免了处理大量不相关数据的不必要花销；其二，文本获取技术能够在适当的上下文中对分析结果或发现的知识给出解释并提供数据来源(出处)。

文本数据获取的总体目标是使用户在适当的时间关联正确的信息。这种关联可以通过两种方式实现：**拉取**(pull)方式是用户通过系统主动寻找相关信息；**推送**(push)方式是系统为用户提供相关信息。本章将对两种文本数据获取模式进行高度概括性介绍，进而对文本检索问题进行形式化说明。之后的章节将对基于拉取模式以及推送模式的文本获取技术进行详细介绍。

5.1 获取模式：拉取与推送

人类创造了文本数据，因而人类在文本数据的分析及管理的相关应用中也起到了重要作用。特别地，人类能够针对特定的应用问题找到最相关的数据，这能够避免对大量原始文本数据的低效处理工作而专注于分析最相关的信息。从大量文本集合中筛选具有相关性的文本是文本获取的基本任务。这种筛选是基于分析员(用户)对信息需求的说明，其实现方式可以分为拉取及推送。图 5-1 描述了这些模式如何在查询及浏览中相互配合。

在拉取模式中，用户一般通过搜索引擎发起对相关文本数据的访问活动。当用户有临时性信息需求时，这种方式会显示出优势，这类临时性信息需求一旦得到满足就可能会消失。在这种情况中，用户可以通过在搜索引擎中输入查询信息寻找相关信息。例如，用户希望购买一款产品，则他对候选产品的所有相关评价都进行检索；当用户购买过产品后，他将不再需要该类信息。在社交媒体上，分析人员为了理解关于新兴事件的一些观点而进行的社交媒体数据分析，会通过与事件相关联的特定实体(例如一个人)挖掘信息，这一活动也会触发搜索行为。

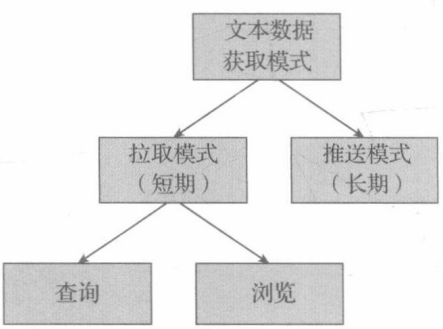


图 5-1 文本信息获取模式的二分图

查询是拉取型文本数据获取的最常见方式，而浏览是另一种与其互补的拉取方式。若用户不知道如何制定一个有效的查询、不方便输入一条查询语句(例如通过智能手机查询)或只是想没有目标地探索一个话题，此时浏览方式是十分有用的。事实上，用户在搜索过程中往往混合运用了查询与浏览方式(譬如遍历超链接)。

一般而言，我们将查询与浏览视作在信息空间中寻找相关信息的两种互补的方式。二者间的关系可以类比于物理世界中对信息的寻找和观赏间的关系。当一位游客知道某个确定的景点时会直接打车去景点，这一过程与以下情况类似，即用户查询目标明确并能够用准确查询关键字直接搜索到相关页面。然而，如果游客不知道景点的确切地址，可能会打

车去一个近似的地点，下车后边走边寻找。在文本信息获取中存在相似的情况，如果一位用户对目标网页不了解，可以通过相近的查询语句获得一些有关联性的网页，再通过浏览方式找到真正相关的信息。所以当查询方式效果不佳时，浏览方式是非常有用的。

在推送方式中，系统主动向用户推荐一些相关信息。对有长期信息需求的用户或分析人员而言，推送方式是更有用的。例如，研究者的研究兴趣在一段时间内是相对稳定的，相比之下，所面对的信息流(如发表科研论文)是动态变化的。在这样一个场景中，尽管用户可以定期运用查询词进行相关文献信息的查询，然而更需要有一个能够监控动态信息流的推荐(也称过滤)系统，该系统可以基于用户兴趣去匹配并推送相关文档(例如以邮件形式推荐)。在一些长期的分析应用中，运用推送方式监控与应用有关主题的相关文本数据(如相关社交媒体)是有必要的。

另一个运用推送方式的场景是信息生成者发起的推荐，这种推荐又称为信息选择性传播(Selective Dissemination of Information，SDI)。在这种场景下，信息的生成者致力于在相关用户中传播信息并将一个信息项推送给用户。在搜索界面上显示产品信息的广告是一个典型的例子。推荐信息可以通过邮件传送或通过搜索引擎结果页面推荐。

有两种类型的信息需求：短期需求和长期需求。短期需求通常与拉取模式相关联，长期需求通常与推送模式相关联。短期信息需求是暂时性的，通常通过搜索或信息页的导航可以满足该类需求。而长期信息需求需要通过过滤或信息推荐更好地得到满足，这类需求需要系统主动为用户进行相关信息推送。临时性的检索非常重要，因为临时性的信息需求比长期的信息需求更频繁地出现。有效的临时检索技术通常也可以被重新用于过滤和推荐。此外，在长期信息需求的情况下，可以收集和利用用户反馈。从这个意义上说，由于 75
没有用户反馈信息(即对特定的查询没有训练数据)，对临时性信息的搜索更为困难。由于训练数据的可用性，通常可以通过使用监督机器学习技术来解决过滤或推荐问题，许多现有的书籍中都涵盖了对这些技术的介绍。所以相较于过滤及推荐技术，我们将在本书中对临时性检索做更详细的介绍。

5.2　多模式互动获取

理想情况下，系统应该为用户提供对文本数据实现多模式互动获取的支持，以便将推送模式和拉取模式集成在同一信息获取环境，查询和浏览无缝集成，为用户提供最大的灵活性，以便用户任意挑选查询或浏览方式。

图 5-2 中展示了原型系统(http://timan.cs.uiuc.edu/proj/sosurf/)的快照。在该系统界面截图中，基于商业搜索引擎中收集的一组查询自动构造主题映射，该主题映射已添加到常规搜索引擎接口，使用户能够灵活地浏览信息空间。有了这个接口，用户随时都可以做以下事情。

查询(远程跳转)。当用户通过搜索框提交新的查询时，搜索引擎的搜索结果将显示在右边的窗格中。同时，主题映射的相关部分也显示在左边的窗格中，以方便用户浏览。

在映射上导航(短程游走)。界面的左窗格是让用户在主题映射上导航。当用户在映射节点上点击时，将刷新此窗格，所点击节点的局部视图将作为当前关注信息显示出来。局部视图显示当前节点(在接口中称为“中心”)的父节点、孩子节点和横向邻居。这样，用户可以放大到子节点，缩小到父节点，或导航到横向的邻居节点。连接到节点的数字是用于对节点进行排序的节点的分数。这样的映射允许用户在信息空间中游走，在不需要重新制定查询的情况下浏览相关文档。



图 5-2　对主题映射的浏览界面样例，界面自然集成了浏览及查询两种方式

查看主题区域。用户可以双击映射上的主题节点，查看主题区域所涵盖的文档。搜索结果窗格将会更新得到与选定主题区域中的文档对应的新结果。从用户的角度来看，结果窗格总是显示用户关注的当前区域中的文档(查询的搜索结果或浏览时对应于映射中的当前节点的文档)。

查看文档。结果窗格作为标准的搜索接口，用户可在其中选择任何文档进行查看。

在图 5-3 中，我们进一步展示了一个跟踪用户浏览的示例，用户开始查询 dining table，放大到 asian dining table，缩回到 dining table，水平导航到 dining chairs，进一步导航到 dining furniture，最后缩小到一般主题 furniture，这个主题下用户有很多选项来探

76
~
77

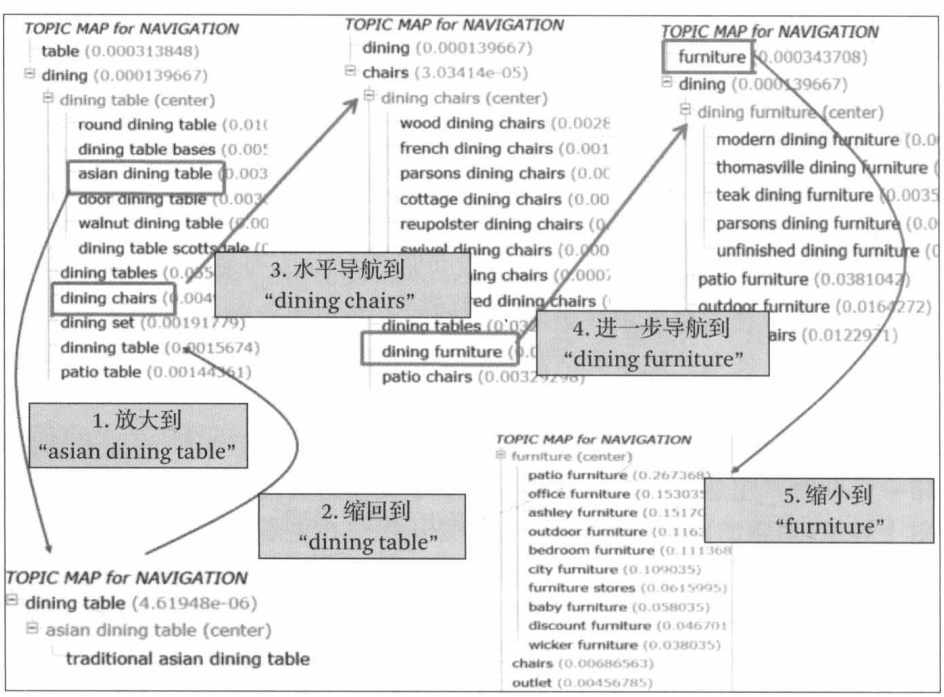


图 5-3　显示用户如何在没有查询的情况下在信息空间中导航的浏览轨迹示例

索不同类型的家具。如果这个用户觉得需要一个远程跳转，那么他可以使用一个新的查询来实现它。由于映射可以被隐藏，只有在用户需要时才显示出来，从用户的角度来说这样的界面是当前搜索界面的一个非常自然的扩展。因此，我们可以看到一个文本获取系统如何结合多种模式的信息获取来满足用户当前的需求。

5.3　文本检索

支持文本数据获取的最重要的工具是搜索引擎，这就是许多人每天都使用 Web 搜索引擎的原因。搜索引擎直接为查询提供支持，并且可以很容易地扩展以提供推荐或浏览。此外，用于实现有效搜索引擎的技术通常也有助于实现推荐系统以及许多文本分析功能。因此，我们在本书中重点讨论搜索引擎技术。

本节讨论文本检索(Text Retrieval，TR)问题，它是通过开发一个搜索引擎系统来解决的。我们详细说明非结构化的 TR 和结构化数据库检索之间的区别。然后，与文档选择相对照地介绍文档排序。这为我们在下一章讨论如何为了查询而对文档进行排序提供了基础。

从用户的角度来看，TR 问题是使用查询在一个文本文档集合中查找相关文档。这是一项频繁被请求的任务，因为用户通常对各种任务有临时性的特定信息需求，并且希望立即找到相关的信息。支持 TR 的系统是一个文本检索系统，或者是一个搜索引擎。 78

虽然文本检索(TR)有时与一般的“信息检索”(IR)互换使用，但后者还包括检索其他类型的信息，如图像或视频。值得注意的是，其他非文本数据的检索技术并不成熟，因此，检索其他类型的信息往往依赖于文本检索技术，用以与非文本数据中附带的文本数据的关键字进行匹配。例如，Web 上当前的图像搜索引擎本质上是一个 TR 系统，其中每个图像由一个文本文档表示，文本文档由任何与图像相关的文本数据组成(例如标题、标题说明或图片的文本上下文(如包括图片的新闻文章的内容))。

在工业领域，TR 问题通常被称为搜索问题，而用于文本检索的技术通常被称为搜索技术或搜索引擎技术。

TR 的任务可以很简单，也可以很困难，具体取决于特定的查询和特定的集合。例如，在网络搜索过程中，查找主页通常很容易，但要找出人们对某个话题的看法(例如：美国的外交政策)将会更加困难。TR 的困难有以下几个原因：

- 查询语句通常很短而且不完整(没有像 SQL 那样的形式语言)；
- 信息需求可能很难准确描述，特别是当用户不熟悉这个主题的时候；
- 对文档内容的精确理解是困难的。一般来说，因为正确的答案是主观的，所以即使由人类专家判断文档的相关性也无法保证一致。

由于缺乏清晰的语义结构和自然语言理解存在的困难，通常很难准确地检索到用户查询的相关信息。实际上，即使当前的 Web 搜索引擎看起来可能是足够的，但是对于一个用户来说，快速查找并获取一个任务的所有相关信息仍然是很困难的。一般来说，目前的搜索引擎对于导航查询和简单的流行的信息查询非常有效，但是在用户拥有复杂信息需求的情况下，比如分析产品购买的意见，或者研究一些症状的医疗信息，它们通常工作得很糟糕。此外，目前的搜索引擎通常很少或不提供支持来帮助用户消化和利用检索到的信息。因此，即使搜索引擎能够检索到最相关的信息，用户仍然需要仔细筛选一长串文档，并详细阅读它们，从而完全消化文本数据中隐藏的知识，以便完成手头的任务。本书后面讨论的技术可以帮助用户快速或直接地分析大量的文本数据，以揭示有用的和可操作的知识，这些知识可以用来优化决策或帮助用户完成任务。 79

5.4　文本检索与数据库检索

将文本检索问题与数据库检索问题进行比较是很有用的。两个检索任务都是为了帮助用户查找相关信息，但是由于这两个任务所管理的数据的差异，存在很多重要的区别。

首先，由搜索引擎和数据库系统管理的数据是不同的。在数据库中，数据是结构化的，根据模式，每个字段都有一个明确的含义。因此，可以将数据视为具有指定列的表。例如，在银行数据库系统中，一个字段可能是客户名称，一个字段可能是地址，而另一个字段可能是每种类型账户的余额。相比之下，由搜索引擎管理的数据是非结构化的文本，这对计算机来说很难理解。[⊖]因此，即使一个句子说一个人生活在一个特定的地址，计算机也很难基于一个查询关键字回答一个人的地址，因为没有一个简单的定义好的结构可以对应到自由文本。因此，结构化数据通常更容易管理和分析，因为它们符合一个清晰定义的模式，其中每个字段的含义都是定义好的。

其次，数据存在差异性的结果是两种检索所支持的查询也不同。数据库查询清楚地指定了数据表字段的约束，因此预期的检索结果(查询的答案)是非常明确而没有歧义的。然而，在搜索引擎中，查询通常是通过关键字查询，这只是模糊地指定应该返回什么文档。通常情况下由于缺乏完整的知识，即使计算机可以完全理解自然语言文本的语义，用户的
80 信息需求也是模糊的(这经常是用户希望先找到一些信息的原因)。例如，在搜索关于研究问题的相关文献时，用户不太可能清晰完整地描述应该返回哪些文档。

最后，两个应用的预期结果也不同。在数据库搜索中，我们可以检索非常具体的数据元素(例如具体列)；在 TR 中，我们通常只能检索一组相关文档。通过在文本文档中标识的段落或字段，搜索引擎也可以检索段落，但通常很难如数据库检索一样检索到特定的实体或属性值。这一区别不像到底查询的“正确”答案是什么的模糊规定那样根本，但这是 TR 中模糊的信息需求带来的直接结果。

由于这些差异，构建一个有用的数据库和一个有用的搜索引擎的挑战也有所不同。在数据库中，由于应该返回哪些项是明确指定的，因此确定哪些数据元素满足用户的查询并应该返回是没有任何挑战的，一个主要的挑战是如何尽快找到答案，特别是当同时发出许多查询的时候。虽然在搜索引擎中也存在着效率方面的挑战，但更重要的挑战是，首先要弄清楚哪些文档应该返回，然后再考虑如何快速返回答案。在数据库应用(至少是传统的数据库应用)中，维护数据的完整性也是非常重要的，也就是说，确保不出现由于电源故障带来的数据不一致问题。在 TR 中，由于用户很难明确地指定信息需求以及自然语言处理自身的困难，建模用户的信息需求和搜索任务是很重要的。

因为对于查询的最佳答案取决于用户，在 TR 中，用户实际上是我们输入的一部分(连同查询和文档集)。因此，没有严格的数学方法来证明一个答案比另一个好，或者证明一个方法比另一个好。相反，我们总是依赖于一些测试集合和用户的经验评价。相比之下，在数据库研究中，由于主要问题是效率，所以通过分析计算复杂度或做一些仿真研究，可以证明一种算法优于另一种算法。但请注意，在进行模拟研究时(为了确定哪个算法更快)，我们也面临着如文本检索中同样的问题，模拟可能无法准确反映实际应用。因此，对于一个特定的应用程序来说，一个显示模拟速度更快的算法可能不会更快。类似

⊖ 尽管与关系数据库的结构化相比较，文本中常见的说法是非结构化数据，但它也会使用狭义的“结构”。例如，从语言学的角度来看，语法提供了良好的结构。为了进一步研究这个问题，我们可以看到 Fox 等[2012]提出的 5S，即 societiy(社会)、scenario(场景)、space(空间)、structure(结构)和 stream(流)。

地，一种被证明更有效的检索算法，对于特定的应用或另一个测试集合，可能会变得不那么有效。如何可靠地评价检索算法本身是一个具有挑战性的研究课题。 81

由于这种差异，这两个领域由不同的群体进行研究，并有着不同的应用基础。数据库在几乎每一个领域都有广泛的应用，拥有一个强大的产业。研究文本检索的 IR 群体一直是一个涉及图书馆和信息科学与计算机科学的跨学科群体。但直到 20 世纪 90 年代初网络诞生之前，它还没有一个强大的产业基础。从网络诞生起，搜索引擎行业已经占据主导地位，随着越来越多的在线信息可用，搜索引擎技术(包括 TR 和其他技术组件，如机器学习和自然语言处理)将继续增加。很快我们就会发现搜索技术像数据库一样广泛使用。此外，由于数据库搜索和 TR 之间的内在相似性(效率和有效性对二者都是重要的，而大多数网上数据由文本字段以及某种结构组成)，两个研究领域现在越来越接近，并面临一些共同的基础性问题，如："什么是正确的查询语言?""怎样才能准确对条目进行排序?""如何快速找到答案?"以及"如何支持交互式搜索?"

从这个比较中获得的最重要的结论是，文本检索的问题是*凭借经验定义的问题*。这意味着哪种方法更有效，不能纯粹用分析推理或数学证明来回答。相反，必须由用户进行实践评价，这使得它成为评价搜索引擎有效性的一个重大挑战。这也是自 20 世纪 60 年代开始研究 TR 以来，文本检索的评价投入了大量研究工作的原因。TR 的评价方法至今仍是一个重要的开放性研究课题，我们将在第 9 章详细讨论它。

5.5　文档选择与文档排序

给定一个文档集合(一组无序的文本文档)，文本检索的任务可以定义为使用用户查询(即用户的信息需求描述)识别出能够满足用户信息需求的文档子集。为了解决 TR 的问题，我们必须首先对其进行形式上的定义。因此，本节将提供 TR 的正式定义，并讨论解决此问题的高级策略。

设 $V=\{w_1, \cdots, w_N\}$是用一种特定的自然语言来表达所有单词的词汇集合，w_i是一个单词。用户的查询 $q=q_1, q_2, \cdots, q_m$是一个单词序列，其中 $q_i \in V$。类似地，文档 $d_i=d_{i1}, \cdots, d_{im}$也是一个单词序列，其中 $d_{ij} \in V$。一般来说，查询比文档要短得多，因 82
为查询通常是由使用搜索引擎系统的用户输入的，用户通常不希望花太多精力输入很多单词。然而情况并非总是如此。例如，在 Twitter 搜索中，每个文档都是一条非常短的推特文，用户也可以从已有的文档中剪切并粘贴一个文本段作为查询，这个查询语句就很长。$C=\{d_1, \cdots, d_M\}$是一组文本文档。一般来说，可以假设集合中存在与用户查询 q 相关的文档子集，即 $R(q) \subset C$；也就是说，它们是相关文档或对输入查询的用户有用的文档。当然，这个相关的设置依赖于查询 q，但是相关的文档通常是未知的；用户的查询只是一个"提示"，文档应该在集合 $R(q)$中。此外，不同的用户可能使用相同的查询来检索一些不同的相关文档集(例如，在极端情况下，一个查询词可能具有歧义)。这意味着期望计算机返回精确的集合 $R(q)$是不现实的，而这在数据库搜索中是可行的。因此，计算机能做的最好的效果就是返回 $R(q)$的近似值，我们将用 $R'(q)$表示。

现在，计算机如何计算 $R'(q)$呢? 在高层次上有两种替代策略：文档选择和文档排序。在文档*选择*中，我们将实现一个二元分类器，根据特定查询将文档分为相关文档或不相关文档。也就是说，我们将设计一个二元分类器函数或者一个指示函数 $f(q, d) \in \{0, 1\}$。如果 $f(q, d)=1$，则假设 d 是相关的文档，而如果 $f(q, d)=0$ 则无关。因此 $R'(q)=\{d \mid f(q, d)=1, d \in C\}$。系统使用这样一种策略估计*绝对相关性*，即文档是否与查询语

句相关。

另一种策略是对文档进行排序，让用户决定何时中断。我们实现一个排序函数 $f(q, d) \in \mathbb{R}$ 并按该函数给出的评分降序排列所有文档。然后用户会浏览排序列表，并在他们认为合适的时候停止。在这种情况下，设置 $R'(q)$ 实际上是由系统和用户共同决定的，因为用户基于在排序的某个位置停下来而隐式地选择一个分数阈值 θ。在这种情况下，$R'(q) = \{d \mid f(q, d) \geq \theta\}$。使用这种策略，系统只需要估计文档的*相对相关性*：哪些文档更有可能相关。

由于相对相关性的估计比绝对相关性更直观，因此我们可以期望它更容易实现排序策略。由于多种原因，实际使用中更倾向于文档排序。

83 首先，由于用户很难确定选择相关文档的准确标准，二元分类器不太可能准确。通常，查询要么约束过强，要么约束不足。在约束过强的查询中，可能没有匹配所有查询词的相关文档，因此强制一个二元决策可能导致没有任何搜索结果的交付。如果查询约束不足，可能会有太多匹配查询的文档，导致过度交付。遗憾的是，在开发文档集合之前，用户通常很难了解“正确”的具体程度，因为用户的大脑中存在知识缺口(这可能是用户想要找到有关主题的信息的原因)。其次，即使分类器是准确的，用户也可以从匹配的相关文档的优先级中获益，因为用户只能在一个时间点检查一个文档，而一些文档可能比其他文档更有用(相关性是一个程度的问题)。由于这些原因，在设计有效的文本检索系统时，对文档进行适当的排序成为一个主要的技术挑战。

概率排序原则(probability ranking principle)[Robertson 1997]指出，在遵循以下两种假设的前提下，根据预测相关性的降序顺序返回有序的文件列表的排序策略在理论上是最理想的。

1) 一篇文档对用户的效用与任何其他文档的效用无关。

2) 用户将按顺序浏览结果。

因而，问题可描述为：我们有一个单词序列构成的查询，以及一个单词序列构成的文档，希望定义可以根据查询和文档计算分数的函数 $f(q, d)$。主要的挑战是设计一个很好的排序函数，它可以将所有相关的文档排列在所有无关的文档之上。很明显，这意味着我们的函数必须能够测量文档 d 与查询 q 相关的可能性，这也意味着我们必须有某种方法来定义相关性。特别是为了实现这个程序，我们必须有一个可计算的相关性的定义，并通过设计一个检索模型来实现这个目标，检索模型将提供相关性的形式化描述。第 6 章将介绍检索模型。

书目说明和延伸阅读

对信息获取更广泛的讨论可通过数字图书馆查看 Goncalves 等[2004]的文献。在 Belkin 和 Croft[1992]文献中讨论了检索(“拉取”)和过滤(“推送”)之间的关系。信息检索与数据库检索的对比在经典信息检索书 van Rijsbergen[1979]中进行了讨论。[Hearst 2009]系统地讨论了搜索系统的用户界面，这一内容与一般信息系统的界面设计有关，特别是，书中讨论了许多可以促进浏览和查询的可视化技术。探索性搜索是一种特定类型的搜索任务，通常需要多种模式的信息获取，包括查询和浏览。这一内容在 Communications of ACM[White 等 2006]以及 White 和 Roth[2009]中有所涵盖。概率排序原则[Robertson 1997]通常被认为是从理论上将检索问题定性为排序问题的奠基性的工作。与此相关的更多截至 1997 年的经典著作可以在*信息检索的阅读资料*中找到[Sparck Jones 和 Willett

84

1997]。在 Sanderson 和 Croft[2012]中可以找到对 IR 历史的概述。

练习

5.1　什么时候浏览比查询更可取？

5.2　给定搜索引擎的用户日志，如何区分浏览行为和查询行为？

5.3　推送和拉取模式通常组合在一个系统中。给出这样一个应用程序的示例。

5.4　假设你有搜索引擎会话日志(session log)，这些日志记录了用户对信息的浏览。如何使用这些日志来增强满足未来用户对临时信息需求的搜索结果？

5.5　我们将在第 11 章中讨论推荐系统。这是推送模式的系统，向用户传递信息。推荐系统的具体应用是什么？你能说出适合这种信息获取模式的一些服务吗？

5.6　推荐系统(推送模式)如何与搜索引擎(拉取模式)结合？这两种服务能互相促进吗？

5.7　设计一个用于搜索音乐艺术家的文本信息系统。例如，可以直接搜索一个艺术家的名字。结果显示为一个图，相似的艺术家通过边相连(通过一些相似度算法度量)。使用 TIS 获取模式词汇表来描述这个系统以及任何可以满足不同信息需求的增强功能。

5.8　以与前面问题相同的方式分类“谷歌知识图谱”(http://www.google.com/insidesearch/features/search/knowledge.html)。 85

5.9　与前两个问题一样，对“引文通知”(citation alert)进行分类。通知是基于学术搜索引擎中先前的搜索历史。基于用户浏览历史，发现新文档可能会令用户感兴趣时，就会创建一个通知。

5.10　概率排序原则的一个假设是，每个文档对用户的有用性独立于索引中其他文档的有用性。这个假设不成立的情形是什么？ 86

第6章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

检索模型

本章介绍两类主要的信息检索模型：向量空间模型和查询似然模型。实践证明它们都属于最有效和有用的检索模型。我们首先从整体上简要概述检索模型，然后具体讨论这两类基本模型。

6.1 概述

过去几十年中，研究者提出了各种各样的检索模型。这些模型可以被大致概括为几种类型(详见 Zhai[2008]的综述)。第一类模型基于相似度的思想。我们假设，如果一篇文档与查询的相似度高于另一篇文档与查询的相似度，则第一篇文档的相关性比第二篇文档的更高。在这种情况下，排序函数将被用于衡量查询与文档之间的相似度。向量空间模型是此类模型的典型代表[Salton 等 1975]，本章稍后将详细介绍。

第二类模型称为概率检索模型(probabilistic retrieval model)[Lafferty 和 Zhai 2003]。概率检索模型遵循不同的策略。我们假设，查询和文档都是随机变量的观察值，二元随机变量 R(取值为 1 或 0)用于指示一篇文档是否与查询相关，那么给定一篇文档和一个查询，可以用随机变量 R 取值为 1 的概率作为度量该文档与查询相关程度的分数。这一基本思路下有不同的具体模型。一种是经典的概率模型，相关工作可以追溯到 20 世纪六七十年代[Maron 和 Kuhns 1960，Robertson 和 Sparck Jones 1976]；一种基于语言模型[Ponte 和 Croft 1998]；另外一种称为随机偏差模型[Amati 和 Van Rijsbergen 2002]。本章将详细介绍一种特定的语言模型方法——查询似然检索模型。经典概率检索框架中衍生出的一个有效检索模型为 BM25[Robertson 和 Zaragoza 2009]。由于 BM25 的检索函数与向量空间检索模型非常相近，我们选择将其作为向量空间模型的一类变体加以介绍。

第三类模型是概率推理模型[Turtle 和 Croft 1990]，其基本思想是将不确定性与推理规则相关联，进一步可以对查询由文档推断而来这一论断的概率进行量化。此类模型在理论上极具吸引力，但是在实际中往往被简化为与向量空间模型或常规的概率检索模型类似的形式。

最后，还有一类模型采用公理式思维[Fang 等 2011]，其基本思想是定义一系列的约束，并要求好的检索函数能够满足这些约束。在这种情况下，问题便转化为找到一个好的排序函数可以满足所有期望的约束。有趣的是，尽管不同模型考虑的出发点不同，最终的检索函数的形式和涉及的变量却是非常相似的。公理式搜索框架已经证明可以有效诊断检索模型的缺陷，从而对模型进行有针对性的改进(如 BM25+[Lv 和 Zhai 2011])。

尽管提出了很多检索模型，但只有少数模型被大规模实验证明了自身的有效性和可靠性。本书所介绍的 4 个具体模型(即 BM25、枢轴长度规范化模型、采用 JM 平滑的查询似然模型以及采用狄利克雷先验平滑的查询似然模型)位于少数的最有效和健壮的模型之列。[⊖]

⊖ PL2 是另一种高效的检索模型，详见[Amati 和 Van Rijsbergen 2002]。

6.2 检索函数的一般形式

在介绍具体模型之前，首先看一下目前最先进的检索模型的一般形式，并审视各个模型共同采用的思想。图 6-1 展示了这些基本思想。首先，这些模型都采用词袋形式的文本表示。关于词袋的相关内容在自然语言处理章节进行了具体解释。词袋仍然是当前所有搜索引擎主要采用的文本表示方式。在这一假设下，计算文档 d 的分数将基于其在每一个查询词上的得分。这里查询是“presidential campaign news”，则文档 d 的分数将根据其在 presidential、campaign 和 news 词上的得分来计算。

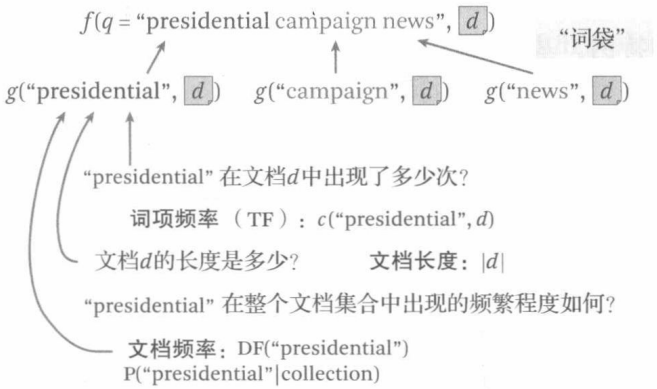


图 6-1 基于词袋表示进行文档评分的基本思想示例

可以看到，计算过程中涉及 3 个不同的组件，每一个都对应于文档匹配每一个查询词的程度。函数背后往往蕴含着一系列的启发式思想。例如，影响函数 g 的重要因素之一是词“presidential”出现在文档中的次数，将其称为词项频率或词频（Term Frequency，TF），也可记为 c(presidential，d)。一般来说，词项在文档中出现的频率越高，该函数的值应该越大。另一个重要因素是文档长度（document length）。一般来说，如果一个词项在一篇长文档中出现很多次，没有在一篇短文档内出现相同的次数更加显著（因为任何词项在长文档中都预期出现更多次）。最后，文档频率（Document Frequency，DF）也是一个因素。文档频率关注的是一个词（如 presidential）在整个文档集合内出现在不同文档中的频繁程度，试图描述词项在整个文档集合中的总体频率。通常来讲，匹配文档集合中的稀有词要比匹配一个高频词对最终评分的贡献更大。TF、DF 以及文档长度在很多最先进的检索模型中承担了重要角色。其他一些模型也会使用概率形式来描述这些信息。

一个很自然的问题是：哪个模型最有效？事实证明，很多模型的性能不相上下。下面列举 4 个主要的、被公认为最有效的检索模型：

- 枢轴长度规范化模型[Singhal 等 1996]；
- Okapi BM25 模型[Robertson 和 Zaragoza 2009]；
- 查询似然模型[Ponte 和 Croft 1998]；
- PL2 模型[Amati 和 Van Rijsbergen 2002]。

Fang 等[2011]的研究指出，经过优化后，这些模型的性能非常接近。其中，BM25 可能是最为常用的模型。事实上，它曾被各个搜索引擎所采用，也是科研论文中最常讨论的方法。本章稍后会更多地介绍该方法。

主要观点可总结如下。首先，设计一个好的排序函数需要一个可计算的相关性的定

义。我们可以通过设计一个合适的检索模型来达到这一目的。其次，很多模型都工作得很好，但是很难选择出哪一个最出色。研究者们依然非常热衷于这个问题，希望能够找到一个真正最优的检索模型。最后，目前最好的排序函数主要依赖以下基本信息：词袋表示、词项频率和文档频率。排序函数利用这些信息并通过对文档长度进行微调来决定一个词项的匹配对最终排序的贡献。这些信息通常会以有趣的方式组合在一起。本书后续部分将介绍如何进行组合来对文档排序。

6.3　向量空间检索模型

向量空间检索模型是一种简单有效的、为信息检索设计排序函数而采用的方法。它是基于相似度的模型的特例。基于相似度的模型假设相关性与文档和查询之间的相似度大致相关。这一假设是否是度量相关性概念的最佳方式依然是悬而未决的开放问题。但是为了解决检索问题，必须将模糊的相关性概念转化为更加精确的定义，从而可以使用编程语言设法实现。在这一过程中，不可避免地需要做出一定假设。这里假设，如果一篇文档与另一篇文档相比和查询更为相似，则认为针对这一查询第一篇文档比第二篇文档更为相关。这是向量空间模型对文档进行排序的基础。这不是形式化定义相关性的唯一方式，后续会看到其他对相关性建模的方式。

向量空间检索模型的基本思想实际上非常容易理解。想象在一个高维空间中，每一个维度对应一个词项，那么文档可以被表示为与词项数量对应的向量，从而可以在这一空间
90 中描绘出来。

图 6-2 展示了一个由三个词 programming、library 和 presidential 构成的三维空间，每一个词定义了其中的一个维度。假设所有的文档和查询都将被置于这个三维向量空间中。例如，向量 d_1 可以表示一篇文档，该文档内容谈论了 library 和 presidential，然而不包括 programming。这一文档表示意味着什么呢？这意味着我们可以仅仅利用这个向量来表示原始文档，而忽略其他所有信息，如词的顺序(在很多情况下词序信息很重要且应该保留)。因此，这并不是最优的表示，但通常足以应对很多检索问题。

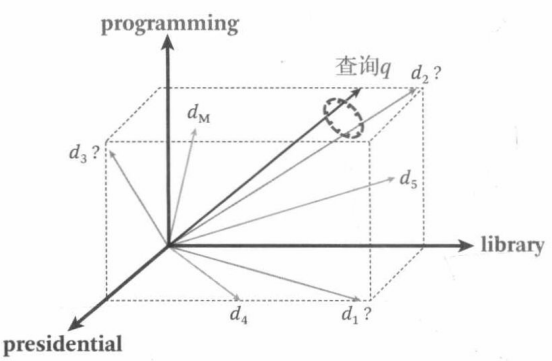


图 6-2　被描绘在向量空间中的文档
(Marti Hearst 授权)

直觉上，在这一表示下，d_1 看起来与主题 presidential 或 library 有关。假设另一篇文档可以表示为 d_2，即包括词 programming 和 library，但是不包括 presidential。可以猜测该文档偏向于与 programming 相关的主题，这里的 library 指的是软件库。显然，d_1 与 d_2 是不同的，利用向量空间表示可以直观地捕捉到文档在主题上的差异。类似地，向量 d_3 所在的方位预示着其主题可能与 presidential 和 programming 有关。将文档集合中的所有文档都在向量空间中表示出来，它们所对应的向量则可能指向该三维空间中的各个方向。

类似地，可以将查询作为另外一个向量置于同一空间中，然后可以度量查询向量与每
91 一个文档向量的相似度。在我们的例子中，显然可见向量 d_2 与查询向量最为接近，因此 d_2 应当排在所有文档的最前面。这就是向量空间模型的基本思想。

确切地说，向量空间模型是一个框架。这个框架下有一些假设：一是文档和查询都被

表示为词项的向量。词项(term)可以是任何基本的概念，如词、短语、n元字串(n-gram)或任何其他的特征表示。我们假设每一个词项定义了空间的一个维度。因此，大小为$|V|$的词表定义了一个$|V|$维空间。查询向量将包括一系列元素对应于不同词项的权重。每一个文档向量也是类似的，包括一系列元素，元素的取值对应于不同词项的权重。在这种情况下，相关性通过两个向量间的相似度进行度量。因此，检索函数也被定义为查询向量与文档向量之间的相似度。

现在，如果要求基于上述方法编程实现一个搜索引擎，你会发现目前对模型的解释并不完备。很多细节还没有提到，因此不可能写出具体的程序来实现这一方法。这也是其被称为向量空间检索框架的原因。这一框架需要进一步细化才能在计算机上实现一个具体的函数。首先，如何定义或选择基本概念(即词项)并没有明确指出。我们假设概念之间是正交的，否则就会产生冗余。例如，两个同义词如果被认为是两个不同的概念，进而定义空间中不同的维度，将导致冗余或者过分强调匹配某一概念(因为尽管匹配了向量中的两个维度，但其实匹配的是同一个概念)。其次，如何将查询或文档置于向量空间的特定位置也没有明确说明。我们已经看到一些查询和文档向量的例子。但是对于任何一个给定的文档，如何确定它对应的向量？这一问题等价于如何定义词项的权重。这是非常重要的问题，因为查询中词项的权重代表了词项的重要性；对词项分配不同的权重意味着我们更希望哪些词项被优先匹配。同样，文档中词项的权重也非常有意义，它指示的是某个词项能在多大程度上准确描述该文档。如果一个搜索引擎返回了很多不相关文档，说明选择的词项以及词项权重不能准确表示这个文档。最后，如何定义相似度度量也是不清楚的。只有解决了这些问题，才能用编程语言实现一个可操作的排序函数。如何解决这些问题是下面内容的主题。

92

6.3.1 向量空间模型实例化

本小节讨论如何实例化空间向量模型，从而得到特定的排序函数。如前文所述，向量空间模型是一个框架，其本身并没有确定任何细节，例如没有说明如何定义向量的维度。将文档向量与查询向量置于空间内的方式也没有明确说明，即如何对查询向量和文档向量中的元素的值进行定义或计算？最后，怎样计算查询向量与文档向量之间的相似度也没有确定。可以想象，为了实现这一模型，我们需要确定如何具体计算与使用这些向量。

图6-3展示了最简单的向量空间模型实例：用词表中的每一个词项定义一个维度，那么空间的维度为$|V|$——基于词袋表示的实例。下面再讨论如何将向量置于该向量空间内。最简单的策略是使用位向量来表示查询和文档，也就是说，每一个元素x_i和y_i的取值将是0或者1。取值为1意味着对应的词在文档或查询中出现；反之，意味着该词没有在文档或查询中出现。如果用户输入若干个词作为查询，那么查询向量中将有少数元素的取值为1，而许许多多元素的取值为0。与查询向量相比，文档向量通常要包括更多的1，但依然有许多的0，因为词表的规模往往很大。词表中的许多词不会出现在一个单独的文档内，也有许多词仅仅偶然地出现在一个给定的文档中。对于一个具体的文档，词表中的大部分词不会在其中出现。

$$q=(x_1,\cdots,x_N) \qquad x_i,y_i\in\{0,1\}$$

1: 词W_i在文中出现

$$d=(y_1,\cdots,y_N)$$

0: 词W_i没有出现

$$\mathrm{Sim}(q,d)=q.d=x_1y_1+\cdots+x_Ny_N=\sum_{i=1}^{N}x_iy_i$$

图6-3 使用位向量表示和点积相似度计算查询向量和文档向量之间的相似度

既然文档和查询已经被置于向量空间内，如何计算它们之间的相似度呢？一个常用的相似度度量是点积。两个向量的点积被简单地定义为两个向量对应元素乘积之和。在图 6-3 中，向量 q 和 d 的点积等于 x_1 与 y_1 之积加上 x_2 和 y_2 之积，以此类推。计算相似度的方
93 式有许多，点积只是其中一种。到目前为止，向量空间的维度以及相似度函数均已定义完毕，最简单的向量空间模型实例化已经完成。该方案基于位向量表示、点积相似度以及词袋表示的实例。因此，终于可以使用编程语言实现一个排序函数。给定一个查询，可以利用该排序函数对文档集合中的文档进行排序。

我们已经了解了使用向量空间模型对检索问题进行建模的过程。随后，对如何将向量置于向量空间内以及如何定义相似度进行了一定假设。最终得到了一个具体的检索函数，如图 6-3 所示。下一步需要思考的是得到的函数是否有意义。该函数会如我们所希望的那样有效地工作吗？我们需要思考函数计算得到的数值，这个数值蕴含的意义是什么？在进入下一节之前，不妨花一点时间考虑一下这些问题。

6.3.2　位向量表示的表现

为了检验提出的最简单的向量空间模型是否奏效，考虑图 6-4 展示的例子。

图中展示了若干文档和一个简单查询。查询是“news about presidential campaign”。我们着重检验语料库中的 5 篇文档，它们覆盖了查询中不同的词项。可以看到，其中一些文档与查询相关，而其他文档很可能与查询无关。如果请你对这些文档进行排序，你会怎么排呢？你(用户)的答案即是理想的排序 $R'(q)$。大多数用户都会同意
94 d_4 和 d_3 比其他文档更好，因为它们更好地覆盖了查询，匹配了 news、presidential 和 campaign，因此它们应该排在最前方。而其他三篇文档 d_1、d_2 和 d_5 则与查询不相关。

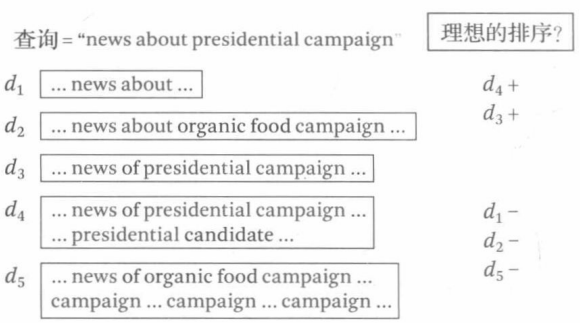


图 6-4　应用位向量向量空间模型的简单举例

提出的向量空间模型能否表现得与理想排序相同或相近呢？首先，考虑模型如何对文档进行打分。图 6-5 中展示了两篇文档 d_1 和 d_3，以及查询。向量空间模型首先要计算文档与查询的向量。查询包含 4 个词，向量中这 4 个词对应的维度将被置为 1，而其余维度将被置为 0。文档 d_1 包含两个词 news 和 about，那么其他词对应的维度均为 0。现在已经有了两个向量，我们可以使用点积计算两个向量之间的相似度，

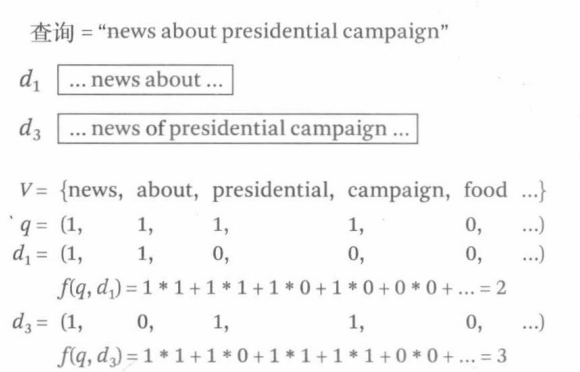


图 6-5　在示例查询与语料库上计算位向量检索模型

即两个向量中相同维度的元素相乘并将所有乘积相加。一对向量对应一个内积，用于表示两者之间的相似度。计算一对向量的点积实际上不必考虑两个向量中值为 0 的维度，因为任何包括 0 的乘积最终结果都是 0。因此，在乘积相加的时候，只需要考虑所有相应的数对中有多少对 1。在本例中，有 1 的对数为 2，因此点积计算的结果为 2，也就是说评分函数赋予文档 d_1 的分数为 2。该评分函数的含义其实就是简单地统计查询中有多少不同的词

被文档匹配。下面再看一下文档 d_3。可以看到计算的结果应该为 3，因为 d_3 匹配了查询中 3 个不同的词：news、presidential 和 campaign。由于 d_1 仅匹配了 2 个词，d_3 的排序应该在 d_1 之前。这看起来不错，但是如果仔细检查这个模型，就会发现一些问题。 95

图 6-6 展示了所有 5 篇文档的分数。基于位向量的评分函数统计文档匹配查询中不同词的个数。如果一篇文档匹配更多查询中的词，那么这篇文档就会被认为更相关。这一假设看起来是有道理的，唯一的问题在于 d_2、d_3 和 d_4 的分数都是 3。仔细观察一下会发现，d_4 应该比 d_3 更为相关，因为 d_3 只匹配了 presidential 一次，而 d_4 提及该词多次。此外，d_2 与 d_3 也有着相同的分数，因为 d_2 也匹配了三个查询词，分别为 news、about 和 campaign。而 d_3 匹配了 news、presidential 和 campaign。直觉上来说，d_3 更为相关，应该被赋予高于 d_2 的分数。因为匹配 presidential 应该比匹配 about 更为重要，尽管这两个词都在查询中。然而，该模型目前并没有考虑这些情况，这也是需要进一步解决的问题。



图 6-6　使用简单的向量空间模型对示例文档进行排序

小结一下，本节介绍了实例化一个向量空间模型需要做的三件事：

1）定义维度（文档包括的概念）；

2）决定如何将文档与查询作为向量置于向量空间中；

3）定义两个向量之间的相似度度量。

基于这一思路，讨论了一个非常简单的向量空间模型实例化策略，衍生出可能是最简单的向量空间模型。使用每一个词来定义一个维度；文档或查询被表示为 0-1 位向量，即只考虑词是否出现而忽略它们的频率；采用点积的方式度量向量之间的相似度，其结果是 96
文档的分数对应于它匹配查询中不同词项的个数。我们举例说明了这个简单的向量空间模型在某些情况下不能很好地工作，依然需要改进。这将是下一小节的主要内容。

6.3.3　改进的模型实例

本小节试图改进前文提到的位向量模型。如前文所述，位向量模型实质上是统计文档匹配查询中不同词项的数目。根据图 6-6 中各文档的内容，d_4 应该排在 d_3 之前，而 d_2 实际上并不相关。存在的问题是该函数没有捕捉到以下现象：

- 首先，文档 d_4 应被给予更多的奖励，因为它比文档 d_3 匹配词项 presidential 的次数更多；
- 其次，匹配 presidential 应该比匹配 about 更为重要，因为 about 是非常常用的功能词，它实质上并没有提供什么内容。

因此，我们需要思考为什么模型会遇到这样的问题。如果重新审视一下在实例化向量空间模型时所做的假设，我们应该意识到正是这些假设导致了以上问题，尤其是关于在向量空间中确定向量位置的假设。为此，我们重新讨论这些假设。一个自然的想法是考虑词项在文档中的多次出现，而不是仅采用二元表示，即应该考察词项的词频（TF）而不是仅仅关注它是否在文档中出现。词频还可以进一步反映一个查询中的词项在文档中出现多次与只出现一次的区别。表达一个词 w 在文档 d 中的词频的最简单方式是

$$\mathrm{TF}(w, d) = \mathrm{count}(w, d) \tag{6-1}$$

使用位向量仅仅体现了一个查询词是否在文档中出现这一信息，而忽略了词出现的次数。因此，将词频信息加入模型中：依然将文档表示为向量，但向量的每一维度对应的是相应词项的权重。换句话说，查询向量或文档向量中各个维度元素的取值不再是非 0 即 1，
97 而是统计每个词在查询或文档中出现的次数，如图 6-7 所示。

下面看一下如果改变表示公式将会是怎样的。排序公式的形式与之前几乎一样，依然采用点积相似度。主要区别体现在求和内部，x_i 和 y_i 与之前不同，它们此时代表的是第 i 个词项在查询或文档中出现的次数。由于查询和文档表示的变化，新的评分函数会有不同的解释。它会解决位向量模型面临的困境吗？

$q=(x_1,\cdots,x_N)$　x_i = 词W_i在查询中出现的次数

$d=(y_1,\cdots,y_N)$　y_i = 词W_i在文档中出现的次数

$$\mathrm{Sim}(q,d)=q.d=x_1y_1+\ldots+x_Ny_N=\sum_{i=1}^{N}x_iy_i$$

图 6-7　词频向量表示和点积相似度

通过图 6-8 中三篇文档来验证。查询向量与之前相同，因为查询中的词都只出现一次。同样，d_2 和 d_3 也没有变化，因为它们都不包含重复出现的词。因此这两个文档的分数也是一样的。d_4 则有所不同，presidential 出现了两次，因此相应维度的权值由 1 变成 2，d_4 对应的分数也相应提高。正如所希望的，使用词频之后，d_4 的得分高于 d_2 和 d_3，从而排序更为
98 靠前。

d_2 [... news about organic food campaign ...]　$f(q,d_2)=3$

$q=$ (1, 1, 1, 1, 0, ...)

$d_2=$ (1, 1, 0, 1, 1, ...)

d_3 [... news of presidential campaign ...]　$f(q,d_3)=3$

$q=$ (1, 1, 1, 1, 0, ...)

$d_3=$ (1, 0, 1, 1, 0, ...)

d_4 [... news of presidential campaign ... ... presidential candidate ...]　$f(q,d_4)=4!$

$q=$ (1, 1, 1, 1, 0, ...)

$d_4=$ (1, 0, 2, 1, 0, ...)

图 6-8　词频向量表示将会奖励查询词项在文档中的多次出现

遗憾的是，d_2 和 d_3 的分数仍然相同。我们期望的是匹配 presidential 获得的奖励比匹配 about 的更多。这个问题一般该如何解决呢？有什么办法能够决定哪些词更重要以及哪些词可以被忽略？像 about 这样没有提供多少内容的词应该被忽略，我们称这样的词为停用词(stop word)。停用词通常出现得非常频繁、随处可见，因此匹配停用词意义不是很大。能否找到有效的统计方法来区分开像 presidential 一样的内容词和像 about 一样的停用词？停用词的一个特点是随处可见。如果统计一个包括 $M(M\gg5)$ 个文档的文档集合中的词频，可以发现 about 的频次将远高于 presidential。这给了我们一个启示，可以利用全局词项统计信息或其他信息尝试降低 d_2 的向量表示中 about 对应的维度的权重。同时，希望能够提高 d_3 的向量表示中 presidential 对应的权重。如果能做到这些，评分函数赋予 d_2 的分数将会低于 3，而赋予 d_3 的分数将高于 3。于是，d_3 便可以排在 d_2 之前。

这个思路被称为逆文档频率(Inverse Document Frequency，IDF)，它是现代检索函数所使用的重要因素。文档频率(document frequency)是指包含某个特定词项的文档的个数。这里使用逆文档频率是因为我们想奖励那些没有在很多文档中都出现的词。如图 6-9 所示，对于一个特定的词，可以通过乘以其 IDF 的方式将逆文档频率引入文档的向量表示。

引入 IDF 可以惩罚常用词——它们的 IDF 值通常较低，并且奖励有信息量的词——它们的 IDF 值通常较高。IDF 可以定义如下：

$$\mathrm{IDF}(w)=\log\left(\frac{M+1}{\mathrm{df}(w)}\right) \tag{6-2}$$

其中，M 为文档集合中文档的总数，$\mathrm{df}(\cdot)$表示文档频率(即包含词 w 的文档数目)。

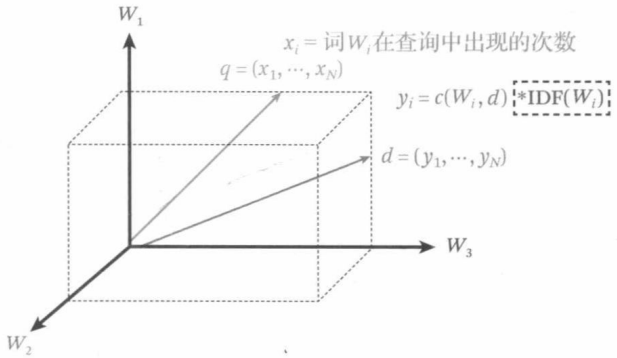


图 6-9　使用 TF-IDF 权重计算方法得到的文档向量以及查询向量

让我们来比较一下词 campaign 和 about。直觉上，about 的 IDF 值应该低于 campaign 的 IDF 值，毕竟它的信息量要小。为清晰起见，假设 $M=10\ 000$，df(about)$=5\ 000$，df(campaign)$=1\ 166$，再对结果取以 2 为底的对数。那么

$$\text{IDF(about)}=\log\left(\frac{10\ 001}{\text{df(about)}}\right)=\log\left(\frac{10\ 001}{5000}\right)\approx 1.0$$

以及

$$\text{IDF(campaign)}=\log\left(\frac{10\ 001}{\text{df(campaign)}}\right)=\log\left(\frac{10\ 001}{1166}\right)\approx 3.1$$

如图 6-10 所示，用 k 表示 df(w)，可以画出一条以 k 为变量，IDF 函数随之变化的曲线。一般来说，一个词的文档频率低则 IDF 较高，意味着该词是一个稀有词。不难看出，IDF 函数的最大值是 $\log(M+1)$。与惩罚常用词的启发式思想相比，这个具体函数的形式并不那么重要，是否还有更好的 IDF 函数的形式依然是一个开放式的研究问题。在第 9 章学习了评价方法后，可以对不同的模型实例进行测试。

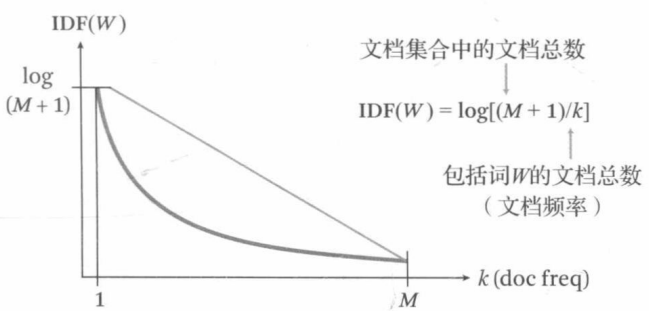


图 6-10　IDF 函数随文档频率变化的示意图

99
~
100

如果 IDF 使用线性函数的形式，曲线即显示为图中对角线。线性函数可能没有之前定义的 IDF 函数形式合理。根据标准的 IDF，大致能够找到一个衰落点，告诉我们文档频率位于衰落点右侧的词项实际上作用不大。如果一个词项出现得如此频繁，它太平常以至于不可能区分两个文档的相关性，因此依据 IDF 的这些推断是合理的。但是如果使用线性函数，很难找到这个衰落点。直觉上，我们更看重 DF 较低的词(而不是那些常用词)的分辨力。当然，哪种形式的表现更好还需要在数据集上进行实验验证。

在图 6-11 中再看一下前面提到的两篇文档。不采用 IDF 方法，我们只能得到位向量。借助于 IDF 权重计算方法，可以通过将基于词频计算而来的权重与 IDF 权重相乘来调整文档向量中每一维度最终的权重。在这一策略下，由于 about 的 IDF 值低于 presidential

的 IDF 值，about 和 presidential 的调整后权重不再相同。因此 IDF 能够根据词的信息量来区分两个词。引入 IDF 权重计算将使 d_3 排在 d_2 之前，因为 d_3 匹配了一个稀有(具有信息量)的词，而 d_2 匹配了一个常用(无信息量)的词。这个例子展示了新的 TF-IDF 权重计算方法如何解决了之前提到的第二个问题。那么，基于 TF-IDF 权重计算的检索模型的实际性能如何呢？我们来看一下之前见过的所有文档。

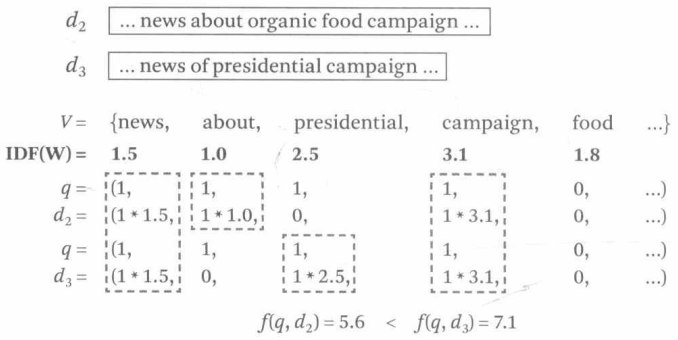


图 6-11　IDF 权重计算方法在文档排序中的作用

101

图 6-12 展示了之前见到的所有 5 篇文档，以及使用 TF-IDF 权重计算方法后它们得到的新的评分。可以看到，前 4 篇文档的得分看起来非常合理。但是，又遇到新的问题。在之前简单的向量空间模型中，d_5 并没有获得特别高的分数，而现在它的分数是最高的。这种情况在设计检索函数时是非常常见的：当你尝试着去解决一个问题时，往往会导致新的问题出现。因此，要设计有效的排序函数不仅需要智慧还需要机警。如何找到“最好”的排序函数依然是开放的研究问题。下一小节将继续讨论更多的想法来进一步提升现有模型并解决当前存在的问题。

6.3.4　TF 变换

在上一小节中，我们使用向量空间模型推导出 TF-IDF 权重估计公式，该模型针对图 6-12 中的例子取得了相当不错的效果。然而，例子中的 d_5 获得了非常高的分数，直觉上该文档是不相关的，因此这一分数并不符合实际。接下来讨论如何利用 TF 变换来解决这一问题。在讨论细节之前，再次审视之前推导的 TF-IDF 权重估计公式和排序函数，如图 6-13 所示。

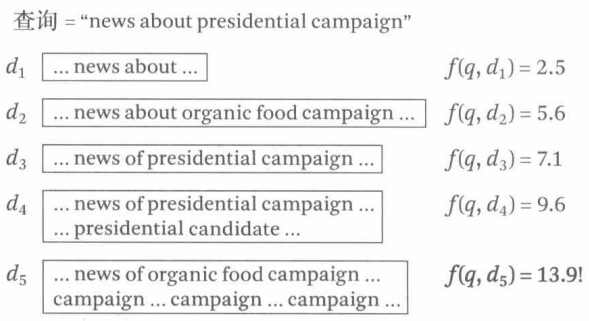


图 6-12　使用 TF-IDF 权重计算方法得到 5 篇文档的评分

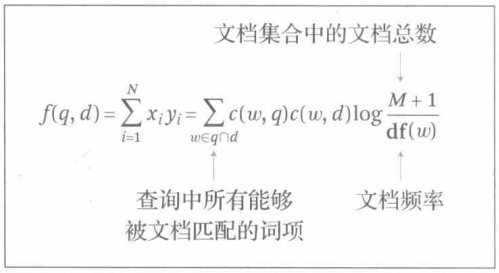


图 6-13　一个采用 TF-IDF 权重计算公式的排序函数

仔细观察会发现，公式包含一个针对所有能被文档 d 匹配的查询词项的求和，其中每一个匹配的查询词项对应一个权重：TF-IDF 权重。公式中 IDF 部分包括两个变量：文档

集合中所有文档的数目 M 以及文档频率 $\mathrm{df}(w)$，即包含词项 w 的文档数目。公式中的其他变量包括词项 w 在查询中的计数 $c(w, q)$ 和 w 在文档中的计数 $c(w, d)$。

不难看出，d_5 获得高分的主要原因是词项 campaign 的数目较大，该词在 d_5 中出现的频率是 4，远高于其他文档，这对 d_5 的高分数做出了贡献。所以，直觉上，为了降低该文档的分数，就需要以某种方法限制文档匹配该词项的贡献。根本上说，不应该如此慷慨地奖励词项的多次出现。文档中词项的首次出现是从无到有的过程，因此从 0 到 1 的增加信息量很大，因为一旦我们在文档中看到一个词，很可能意味着该文档在谈论这个词。词项进一步出现是从一到二的过程，说明该词在文档中出现不是偶然的。但是假定一个词在文档中已经出现了 50 次，再增加一次不会带来关于该词的更多证据，因为我们已经很确认该文档是关于这个词项的。因此，应该限制频率过高的词项的贡献，这便是 TF 变换的出发点，如图 6-14 所示。

变换函数试图将词在文档中的原始计数通过一定手段转换为 TF 权重。x 轴表示原始计数，y 轴表示 TF 权重。实际上，之前的排序函数中已经隐含地采用了一些变换。例如，0-1 位向量表示实质上采用了二元变换，即若词计数为 0 则其权重为 0，否则权重为 1。后来，采用词计数作为 TF 权重，这是一个线性函数。刚刚讨论过，使用线性变换并不合理。如图中两条曲线所示，使用对数函数可以得到次线性(sublinear)变换。这种方式有助于控制词计数的影响，使词计数非常高的词的影响相对变小，同时保持词计数较低的词的影响。如果希望进一步增强这一策略，甚至可以采用两次对数函数，从图形上看曲线将被进一步压弯。研究者们尝试过很多类似策略，实验表明它们确实比简单的线性变换更为有效。到目前为止，最有效的方法是一种特殊的变换，称为 BM25 TF，如图 6-15 所示，这里的 BM 表示最佳匹配(Best Matching)的意思。

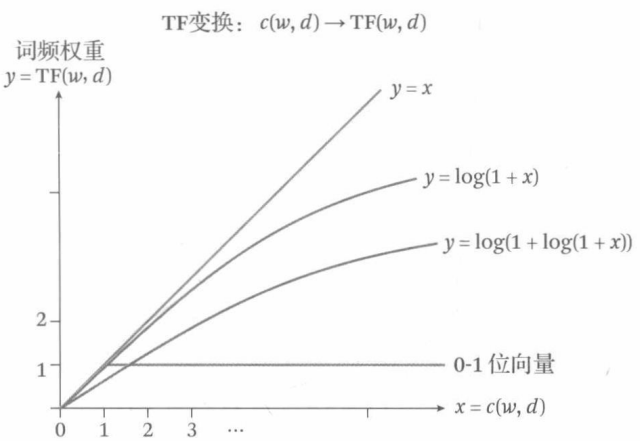


图 6-14　TF 变换的不同方式

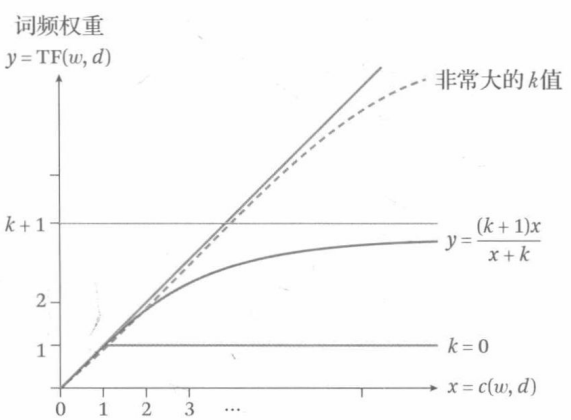


图 6-15　BM25 TF 变换示意图

在该变换中，参数 k 控制着该函数的上界。可以容易看到，该函数存在上界。函数是 $\frac{x}{x+k}$ 与 $(k+1)$ 相乘，由于分子 x 总是小于分母 $x+k$，所以 $\frac{x}{x+k}$ 不可能超过 1，因而它的上界不可能超过 $(k+1)$。这也是 BM25 TF 函数与对数变换的区别，对数变换并没有约束上界。进一步，该函数一个有趣的性质是：随着 k 取值的变化，我们可以模拟不同的变换方程，包括图中展示的两种极限情况。当 $k=0$ 时，可以得到 0－1 位变换。而如果将 k 设为一个非常大的数，函数将非常接近线性变换函数。从这层意义上说，该变换具有很好的弹性，允许我们容易地控制 TF 曲线的

102
⌇
103

形状。简单的常量上界是其另一个良好性质，有利于控制个别词项的影响。例如，我们可
104 以阻止垃圾制造者通过在文档中增加某个词项的词频来影响所有包含该词项的查询的排序结果。换句话说，这个上界确保我们在聚合权重计算分数时所有的词项都会被考虑(而不是仅被少数高频词项决定)。

总结来说，我们需要在 TF 函数中体现一定的次线性。这既代表我们希望减少高计数词项的权重的直觉动机，又避免了文档分数被个别词项“统治”的问题。我们讨论的 BM25 TF 公式具有一个上界，从而使其健壮且有效。如果我们将这个公式代入 TF-IDF 向量空间模型中，将最终得到一个以 BM25 TF 为组件的排序函数。它的形式非常接近于目前最好的排序函数，称为 BM25。我们将很快看到完整的 BM25 公式。

6.3.5　文档长度规范化

本小节讨论**文档长度规范化**(document length normalization)问题。截至目前，向量空间模型中已经考虑了 TF 或词项在文档中的计数以及全局统计量 IDF，但是还没有考虑文档长度。

图 6-16 展示了两篇样例文档。文档 d_4 很短，只有 100 个词，而文档 d_6 包括了 5000 个词。显然，文档 d_6 匹配查询中词项的次数更多，但它可能以分散的方式匹配了查询中的
105 词很多次。而 d_6 的主题与查询主题或许并不是很匹配。文章开始部分讨论的 campaign 与文章结尾部分的 presidential 并没有紧密的关系。一般来说，由于包含词汇多，长文档有更大概率匹配查询词项。事实上，如果我们从所有词的概率分布中通过随机抽样生成一个足够长的文档，则这个文档最终能够匹配任何查询，但这并不意味着它是一个相关文档。从这层意义上说，我们应该对长文档进行惩罚，因为长文档本身就有更大的可能性去匹配任何查询。这是进行文档长度规范化的出发点。尽管如此，对长文档的惩罚也应该适度而不要过分。因为文档长度较大的原因有很多。某些情况下，文档长是因为使用了更多的词来描述同一内容，例如一篇科研论文的正文会比它的摘要使用更多的词。在这种情况下，尽管正文与查询词项在主题及内容上都非常接近，正文对查询词项的匹配却应该被适当惩罚，因为正文显然要比摘要更容易匹配查询词项。当我们比较匹配长文档或匹配短摘要时，长文档一般会有更大的可能性与查询中的词项匹配。因此，我们需要惩罚长文档。

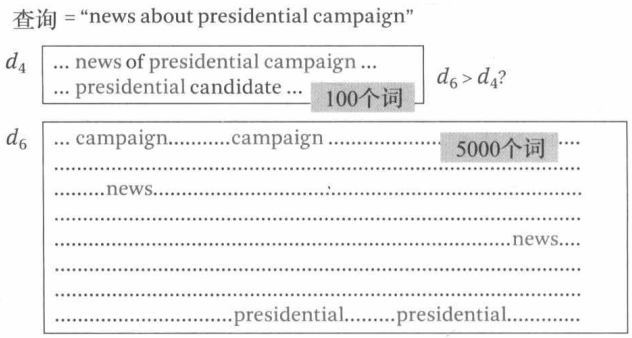


图 6-16　具有不同长度的 2 篇文档

然而，文档长的原因还可能是包含更多的内容。假设我们将多篇论文的摘要拼接起来构造一篇长文档。在这种情况下，不需要对长文档进行惩罚，因为文档中包含了更加丰富的内容。因此，对文档长度的惩罚应该适度。文档长度规范化的优化解决方案需要基于对文档篇章结构的理解。

图 6-17 展示了一个很有效的文档长度规范化方法，称作枢轴长度规范化(pivoted length

normalization)。该方法最早在论文 Singhal 等[1996]中被提出。该方法的基本思想是使用 106
平均文档长度作为枢轴的支点或参考点。假设一篇文档的长度恰好等于平均文档长度，那么该文档中词项的权重被认为是适当的(规范化后应该为 1)。如果一篇文档的长度大于平均文档长度，它包含的词项的权重将会被惩罚；反之，如果长度小于平均文档长度，词项的权重甚至会被奖励。图中 x 轴对应文档长度，y 轴展示的是规范化因子，规范化因子将在排序函数中用于调整词项权重；avdl 表示平均文档长度。规范化因子的计算公式是 1 与规范化文档长度的插值，由参数 b 来控制。用文档长度除以平均文档长度，既可以达到与平均文档长度比较的目的，又不必关心文档长度的量纲是什么。规范化因子具有有趣的特点。首先，如果将参数 b 设为 0，那么规范化因子的值为 1，说明没有对文档长度进行调整。如果将参数 b 设为一个非零的值，对于长度大于平均文档长度的文档，其规范化因子取值就会变得大一些，而短文档的规范化因子取值将相对变得小一些。将长度规范化因子加到向量空间模型的排序函数中，最终会得到目前较为先进的检索模型，图 6-18 展示了其中的几种。从规范化因子在公式中的位置可以看出，长文档受到了惩罚，短文档得到了奖励，可以通过调整 b 的取值(在 0～1 之间)来控制奖惩的程度。

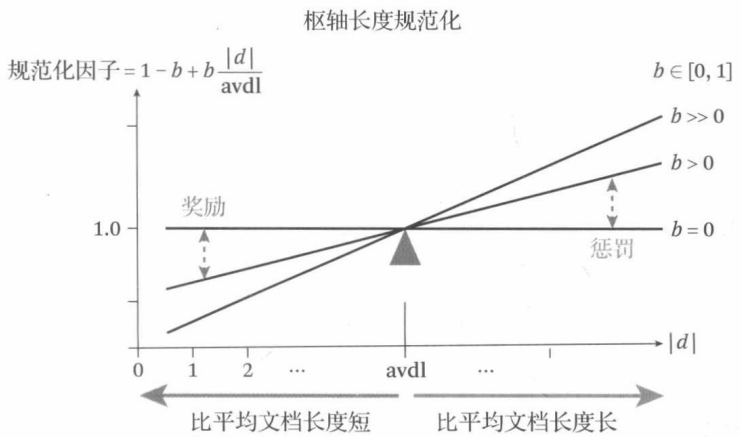


图 6-17　枢轴文档长度规范化

107

枢轴长度规范化的向量空间模型

$$f(q,d)=\sum_{w\in q\cap d} c(w,q)\frac{\ln(1+\ln(1+c(w,d)))}{1-b+b\frac{|d|}{\text{avdl}}}\log\frac{M+1}{\text{df}(w)} \quad b\in[0,1]$$

BM25/Okapi

$$f(q,d)=\sum_{w\in q\cap d} c(w,q)\frac{(k+1)c(w,d)}{c(w,d)+k(1-b+b\frac{|d|}{\text{avdl}})}\log\frac{M+1}{\text{df}(w)} \quad b\in[0,1],\ k\in[0,+\infty)$$

图 6-18　最先进的向量空间模型：枢轴长度规范化和 Okapi BM25

仔细观察图中的每个模型。第一个模型称为枢轴长度规范化(pivoted length normalization)。可以看到，该模型与前面讨论的 TF-IDF 模型基本一致。IDF 部分是公式的最后一项，还包括查询 TF 部分以及中间的规范化 TF。这里使用了双重对数函数来达到一种次线性变换的效果。TF 公式的分母部分加入了文档长度规范化因子，从而达到惩罚长文档的作用，因为分母变大意味着 TF 的值变小了。文档长度规范化由参数 b 来控制。

图中下面的公式称为 Okapi BM25，或简称为 BM25。与第一个模型相同，它也有一个

IDF 部分和查询 TF 部分。中间的规范化部分略有区别，采用了一个有上界的次线性变换，同样也有长度规范化因子在分母部分，因此也能达到类似的效果，即文档长度越长，文档中词项的权重下降幅度越大。

经过系统地思考如何表示文档、逐步调整公式以及最初的假设，我们最终得到了一个目前最有名的检索函数。

6.3.6　基本向量空间模型的进一步改进

到目前为止，我们主要讨论了如何将表示文档的向量置于向量空间之中。这对排序函数的性能起着重要作用。然而，还有其他需要考虑的地方并没有真正地深入探讨。我们一直是假设文档可以表示为词袋的形式。显然，我们可以有很多其他选择。例如，词形还原
108 (将同一个词的不同形式都转化为相同的词根来表示，如英文中由于时态和动名词等形式导致的词形变化)可以认为同一个词的不同形式是相同的，因此它们之间可以相互匹配。还需要去除停用词，去掉那些非常常见但实际上并不携带信息的词汇，如英文中的 the、a 或者 of。可以使用短语，甚至潜在语义分析，使用词的聚类去对文档进行建模。也可以使用更小的单元，如字符的 n-grams，即 n 个连续字符的序列。研究人员在实践中发现包含短语的词袋表示是最有效的文档表示。它的效率也很高，因此是目前最流行的文档表示，也是主流搜索引擎普遍采用的方式。

有时需要采用语言相关或领域相关的表示。这实际上很重要，因为有些情况下词项变体之间难以匹配，尽管在意思上它们是相同的。以汉语为例，首先需要对文本进行词划分获取词的边界，因为汉语文本原本是一串汉字序列。一个词可能对应一个字、两个字甚至三个字。处理英文则简单得多，因为英文词语之间以空格相隔，但是在其他语言中往往需要经过一些自然语言处理来决定词的边界。

此外，相似度函数也可以进一步改进。目前为止，我们只使用了点积一种方法，实际上还有其他度量方式。可以计算两个向量之间夹角的余弦值，也可以采用欧几里得距离度量。点积看起来依然是最好的，其中一个原因是这种方式足够通用。实际上，余弦相似度度量可以视为两个规范化向量的点积。我们先对每个向量进行规范化，而后计算它们之间的点积。从这个角度来说，余弦相似度度量与规范化向量的点积计算是等价的。

前文提及 BM25 是当前最有效的排序公式。有一些方法能够进一步改进 BM25，尽管它们并没有从根本上改变 BM25。一类工作衍生出 BM25-F 模型，这里 F 指的是域(field)。因此这是针对有结构文档的 BM25 模型。例如，科研论文包括题目、摘要和正文，甚至锚文本(在网页上)。需要分配合理的权重将不同域的贡献整合起来得到更准确的文档评分。BM25-F 对每一个域分别应用 BM25 模型进行评分，最后将各个评分合并。但是在应用 BM25 模型时，各个域使用一个共享的全局频率计数。这样做的好处是避免过分奖励词项
109 首次出现带来的评分上的收益。回顾一下在 TF 的次线性变换中，词的首次出现非常重要，对最终评分贡献很大。如果对每一个领域都单独应用 BM25，将会导致某个在每个域中都出现的词获得过大的收益。而如果在统计词频的时候将各个域的词频合并起来统计，就会在一定程度上避免这种情况的发生。该方法对结构化文档评分的效果很好。更多的细节可以参考 Robertson 等[2004]。

BM25 的另一个扩展称为 BM25+，主要处理 BM25 对长文档过分惩罚的问题。处理方式是在 TF 规范化方程中简单地增加一个很小的常数。有趣的是，可以从理论上证明这一简单的修改可以解决原始 BM25 模型过分惩罚长文档的问题。修正后的排序函数即为 BM25+。

BM25+在实际应用与理论分析上都显示出比 BM25 更好的性能[Lv 和 Zhai 2011]。

6.3.7　小结

向量空间检索模型使用相似度来度量相关性，假设文档对于查询的相关性与文档和查询之间的相似度相关。显然，查询和文本需要用相同的方式进行表示。在向量空间模型中，它们都被表示为高维向量空间中的向量。空间的维度根据被用作索引项的词或概念来定义。设计一个排序函数需要使用多个启发式规则，其中重要的规则包括：

- TF(词项频率)权重和次线性变换；
- IDF(逆文档频率)权重；
- 文档长度规范化。

这三项启发式思想确保得到的排序函数对于大多数任务都具有不错的性能。最后，BM25 模型以及枢轴长度规范化策略是最有效的向量空间排序函数。尽管有一些工作扩展了这两种强有力的方法，但它们的本质并没有变化。下一节将讨论与向量空间表示并列的另外一类检索模型。

6.4　概率检索模型

本节将介绍概率检索模型，一种与向量空间模型视角不同的设计排序函数的方式。概率模型根据一个给定文档 d 与查询 q 相关的概率来定义排序函数，即 $p(R=1|d, q)$，其中 $R\in\{0, 1\}$是一个二元随机变量，用于表示相关性。 110

注意，向量空间模型假设所有文档都是具有相同维度的向量，而这里引入一个二元随机变量，并将查询与文档视为由随机变量生成得到的观察数据。因此，主要问题便是如何估计相关性的概率。

此类模型具有很多变种。经典的概率模型可以推导出介绍向量空间模型时讨论过的 BM25 检索函数。另外一类特殊的概率检索函数基于语言建模(language modeling)。本节将重点介绍查询似然检索模型，它是概率检索模型中最为有效的模型之一。还有另外一类函数称为随机偏差模型(divergence-from-randomness model)(如 PL2 函数[Amati 和 Van Rijsbergen 2002])，也是目前最有效、最先进的检索模型之一。

查询似然模型假设相关性的概率可以通过给定文档和相关性的情况下查询 q 的概率$p(q|d, R=1)$来近似。直觉上，该概率体现的是：如果用户喜欢文档 d，有多大可能性他会输入查询 q 来试图检索到文档 d？条件部分包括文档 d 和$R=1$，可以解释为用户喜欢文档 d。为了理解这个想法，首先来了解概率检索模型的基本思想。

图 6-19 展示了一些想象的文档相对于查询的相关性状态(或相关性判断(relevance judgment))。由图可见，q_1 是用户输入的一个查询，d_1 是用户读过的一篇文档，最右侧一列中的“1”表示用户认为 d_1 与 q_1 相

查询 q	文档 d	相关? R
q_1	d_1	1
q_1	d_2	1
q_1	d_3	0
q_1	d_4	0
q_1	d_5	1
⋮		
q_1	d_1	0
q_1	d_2	1
q_1	d_3	0
q_2	d_3	1
q_3	d_1	1
q_4	d_2	1
q_4	d_3	0

$$f(q, d) = p(R=1|d, q) = \frac{\text{count}(q, d, R=1)}{\text{count}(q, d)}$$

$$P(R=1|q_1, d_1) = 1/2$$

$$P(R=1|q_1, d_2) = 2/2$$

$$P(R=1|q_1, d_3) = 0/2$$

图 6-19　基于概率模型的信息检索的基本思想

关。这里的 R 可以利用点击数据(clickthrough data)来近似。点击数据是搜索引擎为了观察用户如何与搜索结果中的文档进行交互而采集的用户行为记录。本例中，可以认为用户输入查询 q_1 并点击了文档 d_1，因而与查询文档对(q_1，d_1)相关联的 R 被置为 1。类似地，用户也点击了 d_2，因而与(q_1，d_2)相关的 R 被置为 1，所以 d_2 也被认为与 q_1 相关。以此类推，d_3 与 q_1 不相关，d_4 与 q_1 不相关，而 d_5 与 q_1 是相关的。表中省略号后面的第二部分可能是来自于另外一位输入同一查询的用户。该用户同样输入了查询 q_1，但与前一用户的判断不同，他认为文档 d_1 与 q_1 并不相关。

假设我们拥有大规模的搜索数据，可以思考“如何估计相关性概率?”这个问题。对于一个特定的文档 d 和查询 q，仅需要简单地去检查表中所有包含 d 和 q 的行就可以计算出表中第三列值为 1 的概率。首先统计表中 q 和 d 成对出现的次数，再统计成对出现时第三列的值为 1 的次数，那么就可以计算一个比率

$$p(R=1|d,\ q)=\frac{\text{count}(R=1,\ d,\ q)}{\text{count}(d,\ q)} \tag{6-3}$$

显然 $p(R=1|d,\ q)+p(R=0|d,\ q)=1$。

看一下具体的例子。假想现在试图计算 d_1、d_2 和 d_3 与 q_1 相关的概率。先计算 d_1 与 q_1 相关的概率，d_1 和 q_1 成对出现两次，其中一次被用户认为文档与查询相关。两种情况中只有一种 R 为 1，因此概率值应为 0.5。类似地，d_2 与 q_1 共现两次，R 都等于 1。d_3 与 q_1 共现两次，R 都等于 0。据此，可以针对查询 q_1，分别计算 d_1、d_2 和 d_3 相关的概率，并根据概率值的大小对三个文档进行排序。这便是概率检索模型的基本思想。在本例中，文档 d_2 的排序最高，因为在所有涉及 d_2 和 q_1 的情况下 R 都等于 1。

111
~
112

搜索引擎可以从规模巨大的点击数据中学习以改进它的排序结果。上面的例子只是为了展示仅仅依靠少数的一些数据，便已经可以进行一定的概率估计。这些概率值可以在一定程度上反映出，对于特定查询哪些文档对用户更有用。但这里存在一个问题，即不可能观察到所有可能的查询、文档以及它们之间的相关性判断的值。因为搜索引擎通常只能从曾经展示给用户的文档中采集用户行为数据，会有很多文档用户从未见过。实际上，从未见过的新查询更多，因为不可能预测出用户在未来会输入什么查询。当前的方法显然无法应用到未见过的查询和文档。尽管如此，我们可以从中了解概率检索模型的基本思想。

那么应该如何处理这种面临大量从未见过的文档和查询的情况呢？解决方案是通过一定手段进行近似估算。其中一种手段称为查询似然检索模型，该模型通过另外一个条件概率 $p(q|d,\ R=1)$来进行近似[Lafferty 和 Zhai 2003]。

前文假设如果观察到用户点击了某篇文档，那么可认为用户喜欢这篇文档。查询似然模型感兴趣的是在所有类似的情形下，即用户喜欢某篇特定文档，他们使用了怎样的查询。注意，这里有一个有趣的假设：用户根据一篇想象的相关文档来构建查询，因为用户在提交查询的时候并不知道会找到怎样的相关文档。如果仅仅将此视为条件概率，该假设看起来并不明显。我们必须在不依赖图 6-19 中的表格的情况下估计条件概率。否则，会面临和之前相同的问题。基于这个假设，可以采用一些方式回避图中表格。

让我们看一下查询似然模型是如何工作的。我们提出这样一个问题：当用户构造查询的时候，哪些文档最可能是用户想象出的相关文档的模样？可通过下面的条件概率来量化这种可能性：如果某篇文档恰好是用户想象出的相关文档，能够观察到用户构造的查询的概率。我们可以比较给定任一文档情况下查询的似然值，并据此对文档进行排序。

小结一下，在概率检索模型框架下对相关性进行建模的基本思想是引入一个二元随机

变量 R，根据条件概率 $p(R=1|d, q)$来定义评分函数。为了近似估计，可以使用查询似然，基于给定文档条件下观察到查询的概率来定义排序函数。这个概率可以解释为如果用户喜欢文档 d，他会构造查询 q 来检索文档 d 的概率。那么问题转化为：如何计算这个条件概率？下一节将进一步讨论其中的细节。 113

6.4.1　查询似然检索模型

查询似然检索模型(query likelihood retrieval model)量化计算用户为了找到某个特定文档而提交一个特定查询的可能性。

图 6-20 展示了查询似然模型如何假设一个用户想象出一个理想文档并根据理想文档产生查询的过程。本例中，理想文档是关于“presidential campaign news”。在查询似然模型中，用户以理想文档为基础尝试构建一个查询，希望能够检索到想要的文档。更具体地，我们假设查询是从文档中抽样得来的。例如，用户可能从理想文档中选择 presidential 作为一个查询词，随后用户继续选择了另外一个词，如 campaign 作为第二个查询词。当然，这只是我们假想的用户构建查询的过程，用户是否真正遵从这个过程是另外一个问题。然而重要的是，该假设可以使我们对给定文档情况下产生查询的条件概率进行形式化的定义，同时不需要借助于前面展示的大表格。基于该基本思想，可以推导出检索函数并使用语言模型实现。



图 6-20　通过从文档中抽样单词来生成查询 114

假设每一个查询词都是彼此独立且都是从想象的、满足用户信息需求的理想文档中产生的。我们需要计算查询似然值，每个查询可视为一个词的序列。既然假设查询词彼此独立，那么该查询的概率可以简单地视为每一个查询词的概率的连乘，而每一个查询词的概率则是词在文档中的相对频率。例如，如果给定文档，presidential 的概率实际上即是文档中 presidential 出现的次数除以文档中所有词出现的总次数(即文档的长度)。于是，我们能够得到一个检索函数对文档进行排序。

仔细看一下图 6-21 中的样例文档。假设现在的查询是 presidential campaign。为了对文档进行打分，仅需统计在文档中见到 presidential 的次数以及见到 campaign 的次数。由图可见，presidential 在文档 d_4 中出现了 2 次，其分数为 $\frac{2}{|d_4|}$，campaign 在文档 d_4 中出现了 1 次，其分数为 $\frac{1}{|d_4|}$。可以以类似的方式计算文档 d_2 和文档 d_3 的概率。如果假设 d_3 和 d_4 具有相同的长度，那么 d_4 将会排序在 d_3 之前，d_3 则排序在 d_2 之前。这种建模

方式看起来很像向量空间模型中的 TF 启发规则。

$$p(q=\text{“presidential campaign”}|d)=\frac{c(\text{“presidential”},d)}{|d|}*\frac{c(\text{“campaign”},d)}{|d|}$$

$$p(q|d_4=\text{... news of presidential campaign ... presidential candidate ...})=\frac{2}{|d_4|}*\frac{1}{|d_4|}$$

$$p(q|d_3=\text{... news of presidential campaign ...})=\frac{1}{|d_3|}*\frac{1}{|d_3|}$$

$$p(q|d_2=\text{... news about organic food campaign ...})=\frac{0}{|d_2|}*\frac{1}{|d_2|}=0$$

图 6-21　使用查询似然公式计算给定文档情况下一个查询的概率

然而，如果尝试一个不同的查询，比如 presidential campaign update，我们可能会发
115 现一定的问题。考虑词 update：图中的文档都不包含这个词。根据查询词是从文档生成的假设，update 这类词的概率将会是零。很显然，这会导致所有文档生成该查询的概率都是零。

对于文档 d_2 来说，概率为零是可接受的，因为它不相关。然而，对 d_3 和 d_4 来说，我们无法通过概率来区分它们的相关性。这显然是不符合预期的。为了清楚这一问题是如何产生的，需要检查推导排序函数时的假设。之前假设每一个查询词必须从用户头脑中的理想文档中产生，为了修正面临的问题，需要假设用户不必仅从文档内选择词汇，从而得到改进的模型。

如图 6-22 所示，设想用户从一个文档语言模型中而不是从文档中挑选词汇。假设文档是从一个一元语言模型中产生的，该语言模型可以分配给 update 一个非零的概率。事实上，我们假设该语言模型为任何一个词都分配一个非零的概率。因此生成过程与之前有了区别：用户在头脑中设想的不是一个理想文档，而是一个语言模型，即词汇表上的一个概率分布。当然，语言模型最终还是需要基于语料库中的文档来进行估计。

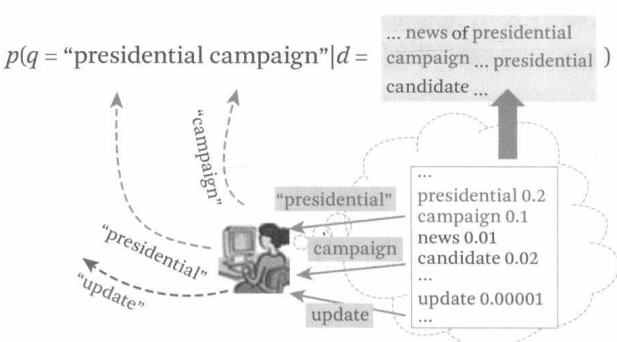


图 6-22　使用文档语言模型计算给定文档情况下一个查询的概率

用户生成查询的过程是类似的。他们可选择一个词，如 presidential，然后选择另一个
116 词，如 campaign。不同的是，现在可以选择像 update 这样的词，尽管它并没有出现在文档内。于是之前遇到的零概率问题可以被解决，而且由于用户是从语言模型而不是某个确定的文本中选择查询词，这种方式也更加通用和合理。

图 6-23 展示了对应于文档 d_1 和文档 d_2 的两个可能的语言模型，查询内容是 “data mining algorithms”。根据独立性假设，$p(q|d)$可以表示为每个查询词在文档语言模型中

的概率连乘。利用计算得到的概率可以对文档进行打分与排序。

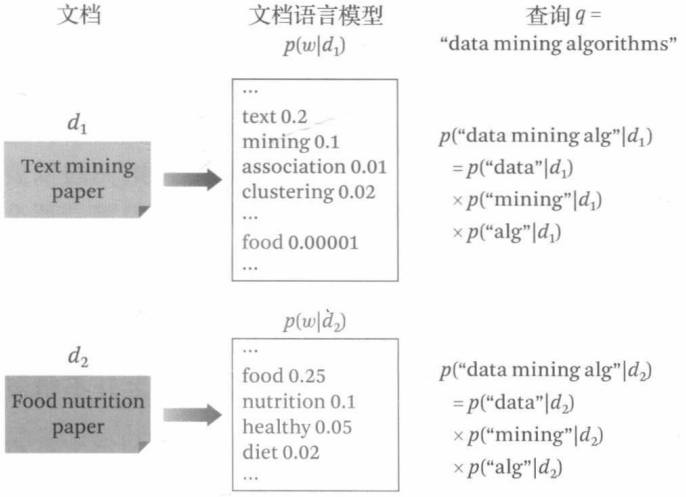


图 6-23　根据两篇文档的语言模型计算查询的分数

形式化地描述计算查询似然的过程。查询 q 包括词

$$q = w_1, w_2, \cdots, w_n$$

因此 $|q| = n$。打分或排序函数即为用户考虑某一个特定文档 d 的前提下，观察到查询 q 的概率。根据独立性假设，该概率是查询中所有的词各自概率的乘积：

$$p(q|d) = p(w_1|d) \times p(w_2|d) \times \cdots \times p(w_n|d) \tag{6-4}$$

117

实践中，针对一个查询对文档评分时，采用的是查询似然值的对数：

$$\text{score}(q,d) = \log p(q|d) = \sum_{i=1}^{n} \log p(w_i|d) = \sum_{w \in V} c(w,q) \log p(w|d) \tag{6-5}$$

这样处理的原因是为了避免过多的小概率连乘导致的下溢和精度损失。使用对数并不会改变文档排序的顺序，却能够避免下溢问题。上面等式中的最后一项对词汇表 V 中所有可能的词进行加和，记录每一个词在查询中出现的次数。实际计算时仅需考虑查询中出现的词，因为查询中不包含的词对最终评分的贡献是零。

现在公式中唯一需要确定的部分是文档语言模型 $p(w|d)$。因此，信息检索问题被转化为文档语言模型估计问题，一旦有了语言模型就可以计算查询被每一个文档生成的概率。在向量空间模型中，将文档转换为向量空间的向量的不同方案可以得到不同的排序函数。同样，不同的估计 $p(w|d)$的方法可以衍生出不同的排序函数。不同的语言模型参数估计方法，能够得到不同的查询似然排序函数。

6.4.2　文档语言模型的平滑

计算查询似然检索分数时，对查询中的每一个词计算给定一篇文档(文档语言模型)产生这个词的概率，并对所有词的概率的对数进行加和。本节详细讨论如何估计文档语言模型。

首先，如何估计语言模型呢？最显然的选择是第 2 章讨论过的最大似然估计(MLE)。在最大似然估计中，每一个词在文档中出现的次数除以文档的长度得到归一化后的频率作为概率。所以，在文档中出现次数相同的词将具有相同的概率，而没有在文档中出现的词的概率将为零。换言之，我们假设用户从文档中进行词汇抽样来构建查询，而没有机会使

118 用文档以外的词。但是，我们已经讨论过，这种导致零概率的方式有很大缺陷。应该如何改进呢？为了给在文档未见过的词分配一个非零的概率，我们需要从观察到的词语的概率值中挪用一部分给未见过的词，否则所有的词的概率之和将不可能为一。

分配给数据中未见过的词以非零的概率从而改进最大似然估计的方法称为平滑(smoothing)。平滑的目的是改进语言模型的参数估计，包括未见词的概率估计。平滑后的语言模型将更加准确地表示真实文档。想象我们读了一篇论文的摘要或者想象一篇文档就是一个摘要。如果我们假定没有在摘要中出现的词的概率为零，那么便无法从摘要外部采样词项。想象一下，如果用户对摘要的主题感兴趣，他很可能使用摘要以外的词做查询，正如文章的正文中必定包括很多摘要中没有出现的词。换一下角度，如果我们要求作者多写一些，作者最终将完成文章全文，里面包含了摘要中没有的词项。因此，平滑的目的正是试图恢复能够表示完整文章的模型。当然，我们实际上并不清楚摘要之外的词的情况，这也是为什么平滑是一个棘手的问题。

这里的关键问题是如何给未见过的词分配概率。有很多不同途径来解决这一问题。一个策略是让未见过的词的概率与其在一个参考语言模型(reference language model)中的概率成比例。也就是说，如果没有在数据中观测到某个词，就假设它的概率由我们构建的另外一个参考语言模型决定，它将告诉我们哪些未出现词项比其他未出现词项具有更高的概率。对于检索来说，一个自然的选择是使用文档集语言模型(collection language model)作为参考语言模型，即如果文档不包括某个词，我们假设该词的概率与其在包括所有文档的整个文档集中的概率成正比。

更形式化地，给定一篇文档，将按照下面方式估计一个词的概率：

$$p(w|d)=\begin{cases} p_{\text{seen}}(w|d) & \text{如果 } w \text{ 在 } d \text{ 中出现} \\ \alpha_d \cdot p(w|C) & \text{否则} \end{cases} \tag{6-6}$$

如果一个词在文档中出现，它的概率将会是一个打了折扣的最大似然估计概率 $p_{\text{seen}}(w|d)$。反之，如果一个词没有在文档中出现，它的概率将正比于其在文档集语言模型中的概率
119 $p(w|C)$，α_d是一个系数，用于控制分配多少概率值给未见过的词。无论词 w 是否出现在文档中，所有的词的概率之和必须为 1，因此 α_d的取值是有约束的。

如图 6-24 所示，将平滑公式代入查询似然排序函数，得到的公式为对文档集合对应

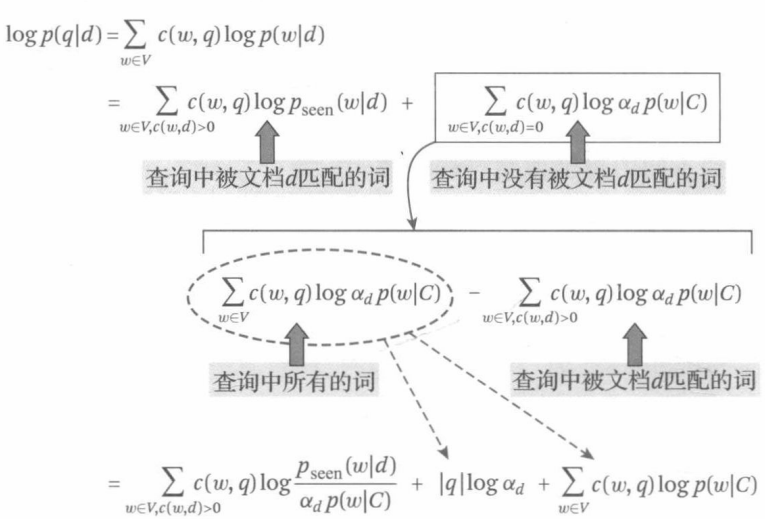


图 6-24　将平滑后的概率替换到查询似然检索公式中

的词汇表中所有词求和的形式，每个词对应的权重为该词在查询中出现的次数。因此，实质上依然是对查询中的词进行求和。写为这种形式是为了便于进行一些变换。进行平滑时，我们假设文档中没有出现的词具有不同的概率形式。因此，可将求和分解为两部分：一部分覆盖查询中所有能够被文档匹配的词，一部分覆盖查询中所有未能被文档匹配的词。这些未被匹配的词根据我们关于平滑的假设而有另外一种形式的概率。

进一步重写第二部分求和(针对查询中未被文档匹配的词)，分解为词汇表中所有词的得分减去查询中被文档匹配的词的分数。其中，对所有 $w \in V$ 求和的部分可以进一步分解为两部分，第一部分为 $\sum_{w \in V} c(w,q)\log\alpha_d = |q|\log\alpha_d$，第二部分为面向查询中所有的词求和。而对查询中被文档匹配的词的分数求和则考察了我们观察到的被匹配的词在查询中的频率。整理之后，同向量空间模型一样，我们现在可以对查询向量与文档向量的交集进行求和。

120

再进一步看看重写之后的排序函数。如图 6-25 所示，重写的排序函数带来两个好处。第一个好处是帮助我们更好地理解该排序函数。特别是，我们将看到基于文档集语言模型的平滑技术与 TF-IDF 和长度规范化等启发式的权重估计之间的联系。第二个好处是使查询似然的计算更加有效，只需考虑查询中被文档匹配的词。

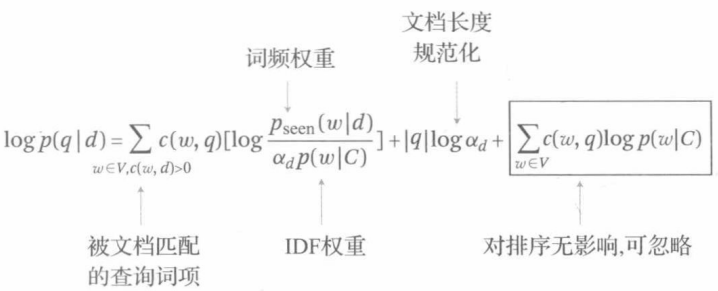


图 6-25　查询似然检索公式体现向量空间模型中三个启发式思想

公式的主要部分对被匹配的查询词进行加和。这比对所有词进行加和要高效得多。使用文档集语言模型对文档进行平滑后，所有的词 $w \in V$ 都有非零的概率。新的公式计算起来更容易。此外，公式的最后一项与具体文档无关，因此在排序的时候可以忽略这一项。这并不会影响文档的排序，因为这一项对每一篇文档的分数的贡献是一样的。

在第一项加和的内部，每一个文档能够匹配的查询词会贡献一个权重。这一权重与向量空间模型中的 TF-IDF 权重设置方法有相似之处。第一，同向量空间模型一样，公式考虑了词在查询中出现的频率。计算点积的时候，词频表现为查询向量中相应维度的权重，文档向量对应位置的值的计算则与 TF-IDF 权重计算方法的作用类似。p_{seen} 在某种程度上与词频相关，如果一个词在文档中频繁出现，那么它的观测概率也会更大。因此，p_{seen} 与 TF 权重计算方法作用相同。在分母部分，$p(w|C)$ 反映了词在整个文档集合中的流行程度。由于 $p(w|C)$ 在分母位置，一个词具有较大的文档集概率会使其最终的权重变小。这意味着频繁出现的词将获得更小的权重，这正是 IDF 起到的作用。可以说我们得到了另一种形式的 TF 和 IDF。尽管 IDF 使用了文档频率的对数，而这里略有不同，但是直觉上，这部分能得到与向量空间模型相似的解释。

121

该公式还与文档长度规范化相关。尤其是 α_d 或许与文档长度相关，它关系到分配给未出现的词多少概率值，或者控制平滑的程度。直觉上，如果文档较长，可以少进行一些平滑，因为可以认为文档能够覆盖作者可以写的所有的词。如果文档较短，未出现的词预期

会比较多，那么需要进行更大程度的平滑。因此，如果文档较长，α_d应该小一些从而起到惩罚长文档的作用。变量α_d实际上出现在两个位置，因此它的整体作用或许未必能惩罚长文档。但稍后考察几个具体平滑方法的时候可以看到，α_d总会以具体的方式惩罚长文档。

以上分析说明，不必考察具体的平滑方法，只需要假设使用文档集语言模型进行平滑，就可以得到与采用 TF-IDF 权重和文档长度规范化相似的公式。此外，得到的排序函数有着非常固定的形式。注意，这里并没有(像 TF-TDF 一样)出于启发式目的使用了对数的必然结果，而是使用了查询似然的对数来给文档打分，并将乘积转换为一系列概率的对数的加和。概率模型的优雅体现在自动地而并非启发式地得到了一个对数函数，起到了对词权重的次线性调整的效果。

总结一下，概率模型一个好的性质是，遵照一定的假设与概率规则能够推导出一个公式。如果试图启发式地设计公式，最终得到的未必是这样具体的形式。此外，我们讨论了对文档语言模型进行平滑的必要性。如果不进行平滑，文档中未出现的词会被分配零概率。在查询中存在文档中没有的词的情况下，这会影响针对该查询的文档评分与排序。还
122 需要更加准确地估计文档语言模型，从而更好地表示文档的主题。检索中平滑策略的基本思想是使用文档集语言模型来指导赋予哪些未出现的词更高的概率。假设未出现的词的概率与它在整个文档集中的概率成比例。借助于这一假设，我们展示了如何为查询似然检索模型推导出一个一般性的排序公式，而且该公式包括了 TF-IDF 权重和文档长度规范化等向量空间模型中的启发式思想。

通过对公式的变换与重写，使用排序函数对文档打分实际上基于对匹配的查询词的权重加和。这个排序函数是基于我们的假设和概率推导得出的，而不像向量空间模型借助于启发式思考得到排序函数的形式。然而，我们依然需要解决一个问题：如何具体地平滑文档语言模型？应该如何使用基于整个文档集合的参考语言模型来调整最大似然估计得到的概率值？这是下一节要讨论的主题。

6.4.3　具体的平滑方法

上一节展示了如何使用文档集语言模型对查询似然检索模型进行平滑。最终得到的检索函数具有如下形式：

$$\sum_{w\in d,q} c(w,q)\log\left(\frac{p_{\text{seen}}(w|d)}{\alpha_d \cdot p(w|C)}\right)+|q|\log\alpha_d \tag{6-7}$$

可以看到，这是对所有能够被文档匹配的查询词求和，在求和内部包括查询词在查询中的频次以及相应的词在文档中的权重。上一节展示了该求和函数如何体现了 TF 和 IDF 的思想，并提到α_d可以用于进行文档长度规范化。如果想使用编程语言实现这一函数，我们还需要计算一些变量，尤其需要知道如何估计每个词的概率以及设置α_d。为了回答这些问题，必须思考如何定义p_{seen}和α_d实现具体的平滑方法。

下面将讨论两种不同的平滑方法。第一种使用固定系数的线性插值，也称为 Jelinek-Mercer 平滑。它的基本思想十分简单。图 6-26 展示了如何使用最大似然估计来估计文档语言模型，用某个词在文档中出现的次数除以文档中所有词出现次数之和。最大似然估计的思想是使观察到的文本的概率最大化。因此，一个未在文档中出现的词如 network，将
123 会得到零概率。平滑依赖于文档集语言模型，该词在文档集语言模型中有非零的概率，这将帮助我们决定分配多大的非零概率给这个词。Jelinek-Mercer 平滑使用线性插值方法将最大似然估计与文档集语言模型结合起来，通过平滑参数$\lambda\in[0,1]$控制。λ越大，平滑

程度越大，分配给背景概率的权重越大。为文档中未出现的词分配非零概率的目的通过混合两个分布得以实现。

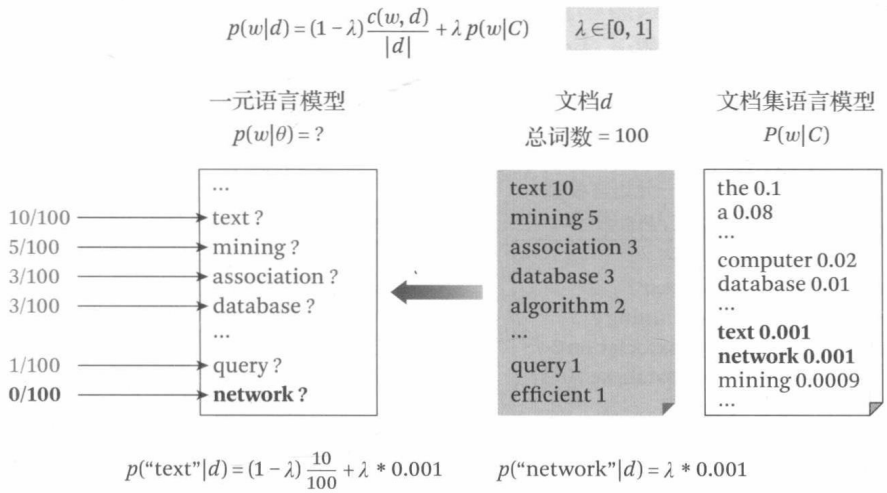


图 6-26　使用线性插值方法平滑查询似然检索函数：Jelinek-Mercer 平滑

具体观察该方法如何工作。例如，计算词 text 平滑后的概率。首先计算最大似然估计概率。既然 text 在文档 d 中出现了 10 次，而 $|d|=100$，最大似然估计即为 $\frac{10}{100}$。假如背景模型中 $p(\text{text}|C)=0.001$，得到平滑后的概率为

$$p_{\text{seen}}(w\mid d)=(1-\lambda)\cdot p_{\text{MLE}}(w|d)+\lambda\cdot p(w|C)$$
$$=(1-\lambda)\cdot\frac{10}{100}+\lambda\cdot 0.001$$

对于图 6-26 中没有出现在 d 中的词 network，其最大似然概率值为 0，平滑后的概率为 $0+\lambda\cdot p(w|C)=\lambda\cdot 0.001$。由此可见，对于该平滑方法，$\alpha_d$ 就是 λ，因为 λ 即是文档集语言模型中词概率之前的系数。 124

第二种平滑方法称为狄利克雷先验平滑(Dirichlet prior smoothing)或贝叶斯平滑(Bayesian smoothing)。同 Jelinek-Mercer 平滑一样，我们使用文档集语言模型来解决像 network 这样的词概率为零的问题。但是，将最大似然估计与文档集语言模型结合起来的方式有所区别。其平滑公式也可以视为对最大似然估计与文档集语言模型进行差值，但是 α_d 不再简单地使用一个固定的值 λ，而是采用动态系数，并使用 $\mu>0$ 作为参数。

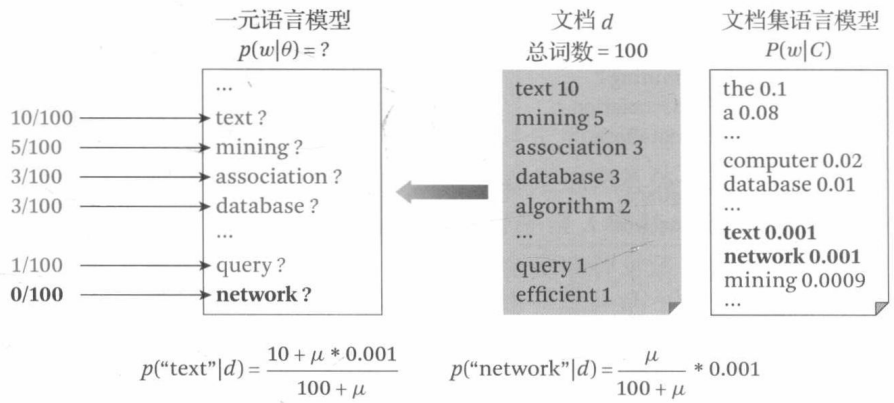
如图 6-27 所示，如果设置 μ 为一个常数，达到的效果是较长的文档将会获得一个较小的系数。所以，长文档得到的平滑要少一些。这与预期一致，看起来比采用固定系数的平滑更好。两个系数 $\frac{|d|}{|d|+\mu}$ 和 $\frac{\mu}{|d|+\mu}$ 加和为 1，说明这是一个有效的概率模型，该平滑方法可以理解为一种动态系数差值。将该平滑方法重写为下面形式，这是对该公式的另一种理解，该形式也有助于记忆：

$$p(w|d)=\frac{c(w,\ d)+\mu\cdot p(w|C)}{|d|+\mu}\tag{6-8}$$

根据现在的形式，可以容易看出我们对最大似然估计做了怎样的修改，即对每一个词增加了 $\mu\cdot p(w|C)$ 个计数，该计数与词 w 在整个文档集中的概率成正比，增加的计数可

以称为伪计数。既然每一个词的计数都增加了，需要重新进行规范化以使所有词的概率和为1，使其满足有效的概率分布要求。因为 $\sum_{w\in V} p(w|C)=1$，所以在分母增加 $\mu=\sum_{w\in V}\mu p(w|C)$，这是所有词在分子上增加的伪计数的总和。

$$p(w|d)=\frac{c(w,d)+\mu p(w|C)}{|d|+\mu}=\frac{|d|}{|d|+\mu}\frac{c(w,d)}{|d|}+\frac{\mu}{|d|+\mu}p(w|C) \qquad \mu\in[0,+\infty)$$



$$p(\text{“text”}|d)=\frac{10+\mu*0.001}{100+\mu} \qquad p(\text{“network”}|d)=\frac{\mu}{100+\mu}*0.001$$

图 6-27　使用线性插值方法平滑查询似然检索函数：狄利克雷先验平滑

再来看具体的例子。我们在文档中观察到词 text 出现了 10 次，再增加与其在整个语料库中概率成比例的伪计数。比如设置 $\mu=3000$，意味着对平滑的语言模型增加额外的3000词计数。我们要分配其中的一定比例给 text。既然 $p(\text{text}|C)=0.001$，将分配 $0.001\cdot 3000$ 的计数给词 text。同样的方式处理词 network，文档 d 中 network 计数为 0，但是增加了 $\mu\cdot p(\text{network}|C)$个伪计数到平滑概率中。

狄利克雷先验平滑的概率包括 $|d|$，因此 α_d的取值依赖于当前评分的文档。在Jelinek-Mercer 线性插值中，$\alpha_d=\lambda$，是一个常数；在狄利克雷先验平滑中，$\alpha_d=\frac{\mu}{|d|+\mu}$，是文档集语言模型的差值系数。关于这些变量的更详细的推导，读者可以参考附录 A。

我们已经在两种平滑方法中分别定义了 p_{seen}和 α_d，下面将这些变量代入最初平滑后的查询似然检索函数。从 Jelinek-Mercer 平滑开始：

$$\frac{p_{\text{seen}}(w|d)}{\alpha_d\cdot p(w|C)}=\frac{(1-\lambda)\cdot p_{\text{MLE}}(w|d)+\lambda\cdot p(w|C)}{\lambda\cdot p(w|C)}=1+\frac{1-\lambda}{\lambda}\cdot\frac{c(w,d)}{|d|\cdot p(w|C)} \tag{6-9}$$

然后，将其代入整个查询似然检索方程，可以得到

$$\text{score}_{\text{JM}}(q,d)=\sum_{w\in q,d}c(w,q)\log\left(1+\frac{1-\lambda}{\lambda}\cdot\frac{c(w,d)}{|d|\cdot p(w|C)}\right) \tag{6-10}$$

这里忽略了 $|q|\log\alpha_d$一项(回顾前一节推导)，因为 $\alpha_d=\lambda$ 并不依赖于当前被评分的文档，因此这一项并不影响文档的顺序。最终得到的函数与向量空间模型惊人的相似，因为都是对所有匹配的查询词进行加和。对数一项是非负的，可以清楚地看到分子部分体现出次线性放缩 TF 权重计算；也可以看到 $p(w|C)$一项位于分母，体现出与 IDF 权重计算类似的作用，一个词在整个文档集中出现的频率越高，分母越大，分子被削减得越多。最后，$|d|$在分母部分实现了类似于文档长度规范化的作用，随着 $|d|$增长，整体的权重将降低。在这种情况下，将会惩罚长文档。第二个分式也可以理解为两个概率的比率：如果比率大于 1，说明 w 在 d 中的概率大于 w 在整个文档集中出现的概率，如果比率小于 1，

125
~
126

说明在文档 d 中观察到 w 的概率低于在整个文档集中观察到 w 的概率。

需要强调的是，概率模型能够在给定假设下自动地推导出权重函数，而在向量空间模型中需要启发式设计获得相应的权重函数。这是使用概率推理的优势，可以显式地从假设中严密推理出来。我们明确知道公式中为什么使用对数，为什么有这些概率。推导出的公式可以有效解释并且实现 TF-IDF 权重计算与文档长度规范化的作用。

下面看一下使用狄利克雷先验平滑的完整公式，已知 $\alpha_d=\frac{\mu}{|d|+\mu}$ 以及

$$p_{\text{seen}}(w|d)=\frac{c(w,\ d)+\mu\cdot p(w|C)}{|d|+\mu}=\frac{|d|}{|d|+\mu}\cdot\frac{c(w,\ d)}{|d|}+\frac{\mu}{|d|+\mu}\cdot p(w|C) \tag{6-11}$$

因此

$$\frac{p_{\text{seen}}(w|d)}{\alpha_d\cdot p(w|C)}=\frac{\frac{c(w,\ d)+\mu\cdot p(w|C)}{|d|+\mu}}{\frac{\mu\cdot p(w|C)}{|d|+\mu}}=1+\frac{c(w,\ d)}{\mu\cdot p(w|C)} \tag{6-12}$$

将此项代入得到完整公式：

$$\text{score}_{\text{DIR}}(q,d)=\sum_{w\in q,d}c(w,q)\log\left(1+\frac{c(w,d)}{\mu\cdot p(w|C)}\right)+|q|\log\frac{\mu}{\mu+|d|} \tag{6-13}$$

函数的形式与 Jelinek-Mercer 评分函数很像。我们计算了一个比率，并用一个非负的对数函数进行次线性放缩。TF 和 IDF 的思想都以类似的计算实现。与 Jelinek-Mercer 平滑方法的不同之处在于对文档长度规范化的处理，这里保留了 $|q|\log\alpha_d$ 一项，因为 α_d 依赖于文档长度 $|d|$。如果 $|d|$ 较大，就少加一点额外的概率给最终的分数；如果 $|d|$ 较小，就多增加一点额外的概率到最终分数中，从而实现对短文档的奖励。

总结一下，本节讨论了两种平滑方法：Jelinek-Mercer 方法，该方法使用固定系数的 127
线性插值技术对最大似然估计进行平滑；狄利克雷先验方法，该方法通过增加与词在背景文档集中概率成正比的伪计数对最大似然估计进行平滑。多数情况下，这些平滑方法得到的检索函数具有清晰的假设和概率推理，而不像向量空间模型那样过于依赖于启发式设计。然而，尽管我们没有显式地定义向量空间模型中的启发式规则，最终得到的模型却实现了与 TF-IDF 权重计算以及文档长度规范化等类似的效果。每种平滑方法都有一个有着直观含义的平滑参数(λ 或 μ)。我们依然需要设置或者采用某种方法估计出参数的值。总的来讲，概率模型与向量空间模型采取了非常不同的策略，然而最终得到的检索函数却是非常相似的。概率模型支配下的检索模型能够更清楚、直接地描述假设，具有更好的解释性。

本节也对查询似然概率检索模型进行小结。回顾一下我们设置了哪些假设推导出最终的检索函数。

1) 相关性通过查询似然来建模，即 $p(R|d,\ q)\approx p(q|d)$。

2) 查询词被独立生成，从而可以将整个查询的概率分解为查询中各个出现的词的概率的乘积。

3) 如果一个词没有出现在文档中，利用背景文档集进行平滑，使它的概率与其在整个文档集中的概率成正比。

4) 最后，采用 Jelinek-Mercer 或狄利克雷先验平滑，它们分别对应不同的假设。

基于以上 4 条假设，可以严格地推导出前文展示的检索函数。函数具有良好的性质，

能够起到 TF-IDF 权重计算和文档长度规范化的作用，并且在实践中也工作得很好。与向量空间模型相比，概率模型推导得到的检索函数更有理论支撑，减少了启发性。

书目说明和延伸阅读

Zhai[2008]第 2 章简要回顾了许多不同类型的检索模型的简要回顾(Zhai[2008])。Singhal 等[1996]提出了具有枢轴长度归一化的向量空间模型并进行了详细讨论。查询似
128 然检索模型最初是由 Ponte 和 Croft[1998]提出。对于 BM25 检索函数有用的参考文献是 Robertson 和 Zaragoza[2009]。对于信息检索的语言模型的全面概述可以在 Zhai[2008]中找到。在 Fang 等[2004]中提出了形式化的检索启发处理方法。Fang 等[2011]提出了一种公理化的评价检索模型缺陷的方法，其中运用了多个改进的基本检索函数。

练习

6.1　下面有两个查询和文档的向量。那么使用点积相似度，给定文档的分数是多少？

$$d=\{1,\ 0,\ 0,\ 0,\ 1,\ 4\}\quad q=\{2,\ 1,\ 0,\ 1,\ 1,\ 1\}$$

6.2　在什么类型的查询中，我们可能不关心查询词频率？

6.3　设 d 是语料库中的文档。假设我们添加另一个 d 的副本。这如何影响语料库中所有单词的 IDF？

6.4　给定一个大小固定的词汇表，文档的长度与用于表示它的向量的长度相同。对或错？为什么？

6.5　考虑欧几里得距离作为文本文档的相似性度量：

$$d(q,d)=\sqrt{\sum_{i=1}^{|V|}(q_i-d_i)^2}$$

欧几里得距离与本章讨论的余弦度量相比主要捕获了什么信息？你比较偏爱其中一个吗？

6.6　如果对 V 中的单词进行词干提取，以创建 V'，则 $|V'|>|V|$。对或错？为什么？

6.7　下列哪种方法最适合缩小大型语料库中的词汇量？

- 删除前 10 个单词。
- 删除发生 10 次或更少的单词。

6.8　为什么使用原始的词频计数与点积相似性不能给出最好的排名？

6.9　如何将向量空间模型应用于文本文档以外的领域？例如，你如何在 IMDB 中找到类似的电影或查找与特定的歌曲类似的音乐？提示：首先定义你的概念空间；什么是
129 你的“词项”向量？

6.10　在 Okapi BM25 中，如何通过设置参数来去掉文档长度规范化？它应该被设为何值？

6.11　检查 MeTA 中的 Okapi BM25 检索函数。你应该看到，它与本章讨论的公式略有不同。有什么区别，你认为它们的作用是什么？

6.12　查询似然和语言模型是如何关联起来的？

6.13　在一个一元文档语言模型中，需要多少个参数？(也就是说，为了描述该语言模型，必须知道多少个概率？)

6.14　在一个二元文档语言模型中，需要多少个参数？

6.15　给定根据本书内容估计的一个一元语言模型 θ 和两个文档 $d_1=$“信息检索”和 $d_2=$

“检索信息”，则 $p(d_1|\theta)>p(d_2|\theta)$。对或错？为什么？

6.16　对于该问题和下一个问题，请参阅称为绝对折扣法的概率检索方法：

$$p_s(w|d)=\frac{\max(c(w,d)-\delta,0)}{|d|}+\frac{\delta|d|_u}{|d|}\cdot p(w|C)$$

和

$$\alpha_d=\frac{\delta|d|_u}{|d|}$$

这里的 $\delta\in[0,1]$，$|d|_u$是特定文档 d 中出现一次的条目的总数。

考虑极限情况 $\delta=0$ 和 $\delta=1$ 时将发生什么？

6.17　绝对折扣法能够捕获文档长度规范化吗？如何实现的呢？

6.18　请给出两个理由，为什么狄利克雷先验平滑要比加一平滑好，它被定义为：

$$p_s(w|d)=\frac{c(w,d)+1}{|d|+|V|}$$

6.19　向量空间模型中的哪些启发式想法能够被平滑的查询似然公式捕获？

6.20　下面的公式可以作为得分函数吗？为什么是或者为什么不是？

这里：

$$\text{score}(q,d)=\sum_{w\in q,d}\frac{k\cdot c(w,C)}{c(w,d)}\cdot\ln\left(\frac{N+1}{\text{df}(w)}\right)\cdot\frac{n}{n_{\text{avg}}}$$

130

其中

- $k>0$ 是一些参数；
- $c(w,C)$和 $c(w,d)$是当前单词分别在集合和文档中的计数；
- N 是文档的总个数；
- $\text{df}(w)$是包含当前单词 w 的文档的数量；
- n 是当前文档的长度；
- n_{avg}是语料库中文档的平均长度。

131
~
132

第 7 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

反　　馈

本章介绍文本检索系统中的反馈机制。反馈利用用户行为或过去的搜索结果提高当前查询的检索结果。如图 7-1 所示，反馈经常通过更新查询以改变返回的文档列表。用户首先输入查询，该查询将被提交给标准的搜索引擎，搜索引擎会返回按照相关性排序的文档列表(如第 6 章所述)展示给用户。用户检查返回的搜索结果，并判断每篇文档是否有用，比如用户可能会说某篇文档很好或者某篇文档用处不大。对文档的每一次判断都被称为一个相关性判断(relevance judgment)。整个过程便是一种相关反馈，用户对搜索结果的评判可以提供一些反馈信息。

相关反馈对检索系统非常有益，因为系统应该可以从中学习到特定用户或用户群真正感兴趣的内容。反馈模块将会利用用户的反馈信息去改善未来的排序。如前所述，典型的反馈模块会更新查询使检索系统返回对用户来说更为准确的排序结果。这便是相关反馈的基本思想。

此类相关性判断很可靠，但是用户往往不情愿花费额外的精力，除非不得不做。另外一种反馈形式称为伪相关反馈(pseudo relevance feedback)或盲目反馈(blind feedback)。伪相关反馈不需要用户参与其中，而是假设排序中前 k 个文档是相关的。例如，前 $k=10$ 个文档是相关的。接下来可以利用这些文档来学习和改进查询表示。但是，如果排序中前 k 个文档是随机的，伪相关反馈还有用吗？实践中，即使排序靠前的文档不相关，它们与相关文档也应该很相似，否则它们不会出现在排序列表中靠前的位置。因此，从排序前 k 个文档中很可能学习到与查询相关的词汇，即使用户并没有显式地判断文档是否相关。

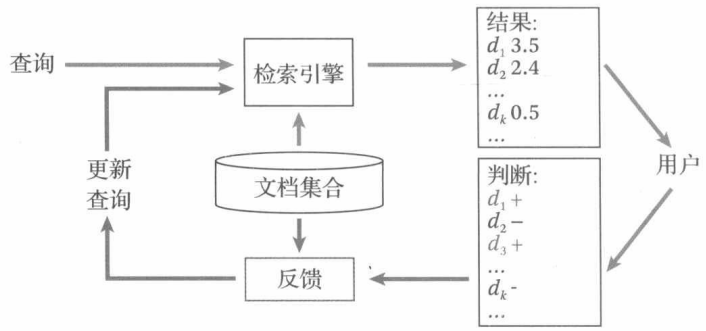


图 7-1　反馈作为信息检索系统的组成部分

回想我们在第 3 章探讨过利用语言模型分析关联词语，能够学习到与词 computer 相关的词汇。首先，使用 computer 检索出所有包含这个词的文档作为搜索结果，即假设查询为 computer。而后，利用匹配查询 computer 最好的 k 个文档中的词计数信息来估计主题语言模型中词项的概率。最后，利用背景语言模型筛选出在检索到的文档集合中频繁出现而在整个文档集合中出现并不多的词项。这种方式可以得到 computer 的相关词汇。这些相关词汇可以被添加到原始查询中作为扩展，从而可以找到那些不仅能匹配 computer 并且可以匹配像 program 和 software 这样并不在原始查询中的词汇。

遗憾的是，伪相关反馈并不完全可靠；我们不得不武断地设置一个截断值并寄希望于排序函数足够好从而能够至少返回一些有用文档。还有一类反馈方法称为隐式反馈(implicit feedback)。隐式反馈也需要用户参与，但是不需要明显地要求用户进行判断，而是通过观察用户点击等行为来分析用户如何与搜索结果进行交互。如果用户点击了某一文档而略过了另一篇文档，这提供了一种指示文档是否有用的线索。我们甚至可以仅利用文档的摘要信息(snippet，即用户在搜索引擎结果页面上看到的文本)，既然用户点击了某个文档，就说明展示的文本信息可能是与用户需求相关或者是用户感兴趣的内容。这便是 134
隐式反馈的基本思想，同样可以利用这些信息来更新查询。隐式反馈是现代搜索引擎使用的重要技术。思考一下谷歌和必应可以怎样收集用户行为来改善它们的搜索结果。

总结一下已讨论的三种反馈类型。相关反馈使用显式的相关性判断，需要用户付出额外的努力，但是这种方法最可靠。伪相关反馈简单地假设前 k 个文档是相关的，从而不需要用户参与其中。因此，伪相关反馈可以在最终的搜索结果展示给用户之前自动完成。最后，隐式反馈利用用户的点击信息。虽然该方法也需要用户参与，但不需要明确要求用户去判断搜索结果。

接下来将讨论如何在向量空间模型和查询似然检索模型中应用反馈技术。后续小节中不会关注反馈文档是怎样得到的。无论它们如何被采集，下面两类反馈方法处理这些文档的方式都是一样的。

7.1 向量空间模型中的反馈

本节讨论向量空间检索模型中的反馈。如前文所述，文本检索系统中的反馈通过从过去的查询中学习去提高未来查询的检索准确性。因此，我们需要获得一些正样本，即与具体查询相关的文档，以及一些负样本，即与具体查询不相关的文档。系统如何获取这些反馈文档依赖于上一节介绍的具体的反馈策略。

向量空间模型中实现反馈的一般方法是修改查询向量，希望能够将查询向量在高维词项空间中置于更好的位置，使其与相关文档更为接近。我们可以调整查询向量中已有词项的权重或者分配权重给新的词项。查询中词项的个数往往会增加，这也常被称为查询扩展(query expansion)。向量空间模型中最有效的反馈方法在几十年前提出，称为 Rocchio 反馈(Rocchio feedback)。

图 7-2 展示了 Rocchio 反馈的基本思想。查询和文档集中的文档都被置于二维空间之内，查询向量在中间，＋(正样本)和－(负样本)表示文档。计算查询向量与文档向量的相似度得到与查询最相似的文档，图中用虚线画了一个圆圈来标记最相似的文档。显然，不是所有排在前面的文档都是相关的正样本，这也是为什么要进行反馈的原因。我们的目标是将查询向量的位置进行调整，使图中的圆圈在平面上随之移动，让圆圈能够覆盖尽可能多的＋文档、覆盖尽可能少的－文档，从而提高检索精度。这是 Rocchio feedback 的基本思想。

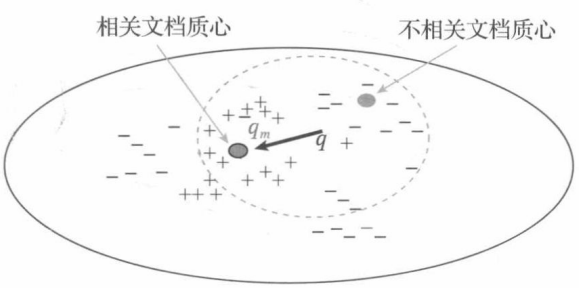


图 7-2 Rocchio 反馈示意图：调整查询向量的权重使其向相关文档聚集的方向移动

从几何的角度，我们谈论的是移动一个向量使其接近一些向量，远离另外一些向量。从代数的角度，可以用下面的公式表示：

$$\boldsymbol{q}_m = \alpha \cdot \boldsymbol{q} + \frac{\beta}{|D_r|} \cdot \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \frac{\gamma}{|D_n|} \cdot \sum_{\boldsymbol{d}_j \in D_n} \boldsymbol{d}_j \tag{7-1}$$

其中 $\boldsymbol{q}$ 是初始的查询向量，$\boldsymbol{q}_m$ 为修改后(即扩展的)的查询向量。D_r 是相关的反馈文档集合，D_n 是不相关的反馈文档集合。另外，α、β 和 γ 是参数，用于控制移动初始向量的程度。可以看到，初始查询中的词项被增强了 α 倍，相关(正)文档中的词项被增强了 β 倍，而不相关(负)文档中的词项被削减了 γ 倍。

对公式中的第二项(围绕所有＋文档求和)以及第三项(围绕所有－文档求和)可以解释为向量的质心。第二项是所有相关反馈文档的质心向量，第三项是不相关反馈文档的质心向量。从这个角度看，我们将初始查询向相关文档的质心靠拢，而远离不相关文档的质心。向量集合的质心是由集合中向量在各个维度的均值构成的。

135
~
136

进行这些操作之后，可得到新的查询向量，并使用新向量对索引中的文档进行重新评分。新的查询向量体现出原始查询向量移向相关文档向量质心，而远离不相关文档向量质心。

下面展示一个具体的例子。假设有一个较小的词汇表

$$V=\{\text{news}, \text{about}, \text{presidential}, \text{campaign}, \text{food}, \text{text}\}$$

以及一个查询

$$\boldsymbol{q}=\{1, 1, 1, 1, 0, 0\}$$

回顾一下第 6 章，词汇表 V 是一个固定长度的词向量。这里不必关心搜索引擎采用怎样的策略计算向量各个维度的权重，因为在 Rocchio 反馈中，仅需要对查询向量中词项的权重进行加法或减法操作。

假如有 5 篇反馈文档，相关文档的词向量用前缀＋进行标记，不相关文档的词向量用前缀-进行标记。

	{	news	about	pres.	campaign	food	text	}
$-d_1$	{	1.5	0.1	0.0	0.0	0.0	0.0	}
$-d_2$	{	1.5	0.1	0.0	2.0	2.0	0.0	}
$+d_3$	{	1.5	0.0	3.0	2.0	0.0	0.0	}
$+d_4$	{	1.5	0.0	4.0	2.0	0.0	0.0	}
$-d_5$	{	1.5	0.0	0.0	6.0	2.0	0.0	}

Rocchio 反馈分别计算相关文档的质心与不相关文档的质心。相关文档的质心向量各个维度的值是所有相关文档向量对应维度的均值，不相关文档的质心的计算也是一样。

	{	news	about	pers.	campaign	food	text	}
$+C_r$	{	$\frac{1.5+1.5}{2}$	0.0	$\frac{3.0+4.0}{2}$	$\frac{2.0+2.0}{2}$	0.0	0.0	}
$-C_n$	{	$\frac{1.5+1.5+1.5}{3}$	$\frac{0.1+0.1+0.0}{3}$	0.0	$\frac{0.0+2.0+6.0}{3}$	$\frac{0.0+2.0+2.0}{3}$	0.0	}

根据两个质心可以修正初始查询得到扩展后的查询 $\boldsymbol{q}_m$：

$$\begin{aligned}\boldsymbol{q}_m &= \alpha \cdot \boldsymbol{q} + \beta \cdot C_r - \gamma \cdot C_n \\ &= \{\alpha + 1.5\beta - 1.5\gamma,\ \alpha - 0.0067\gamma,\ \alpha + 3.5\beta,\ \alpha + 2\beta - 2.67\gamma,\ -1.33\gamma,\ 0\}\end{aligned}$$

137

参数 α 控制初始查询词项的权重，本例中均为 1；参数 β 控制相关文档质心向量 C_r 的影响；参数 γ 则是不相关文档质心向量 C_n 的权重。原始查询向量 $\boldsymbol{q}$ 经过一系列移动得到了修正的查询向量 $\boldsymbol{q}_m$，而后使用新的查询重新进行检索。修正后的查询向量应该能够更好地匹配相关文档，因为它与相关文档更接近，而与不相关文档更远。这正是反馈机制希望达

到的效果。

在实践中采用该反馈策略会导致一个潜在的问题：需要花费很大的计算量去计算质心向量以及修改新查询中所有词项的权重。因此，常用的方法是截断该向量，仅保留权重最高的少数几个词项，从而提高效率。此外，负样本，即不相关文档，与正样本相比用处不大。一个原因是负样本文档的主题是极其分散的，因此平均来看并没有告诉我们应该向哪个方向移动查询向量。另一方面，相关文档会倾向于因主题集中而聚集，因此它们与查询向量的方向通常是比较稳定地保持一致。由于这一原因，有时可以不使用负样本或者设置较小的参数 γ。

避免过拟合也非常重要，这意味着需要将原始查询词项的权重 α 设置得相对较高。我们不希望过分地相信小规模的相关文档、忽略初始查询语义而完全重构查询。初始查询中的词项是用户选择输入的，意味着用户认为这些词是重要的。所以，应该使修改后的查询向量偏向初始查询的方向。尤其是对于伪相关反馈，反馈文档的可信度并不是很高，保持初始查询的重要性尤为重要。尽管存在以上问题，Rocchio 方法通常非常健壮和有效，因此是最常使用的反馈策略。

7.2　语言模型中的反馈

本节讨论查询似然检索模型中的反馈。回顾一下，查询似然排序函数是根据词项独立性等假设逐步推导出来的。作为基本的检索函数，它的表现很好。然而，如何在查询似然模型中融入反馈信息却并不直接。很多情况下，反馈信息是关于查询的额外信息。查询似然模型的基本假设是查询词从一个设想的理想文档语言模型产生，因此我们并不清楚如何简单地增加额外信息。 138

然而，有一种方式可以泛化查询似然函数从而允许更轻易地融入反馈文档信息，称之为 Kullback-Leibler 散度检索模型，简称为 KL-散度检索模型。该模型使查询似然检索函数更加接近于向量空间模型。尽管如此，新形式的语言检索模型依然可认为是查询似然模型的一般形式(不考虑反馈的查询似然模型可视为 KL-散度检索模型的特例)。反馈可以通过查询模型的估计或更新来实现。与 Rocchio 反馈方法更新查询向量殊途同归，这里更新查询语言模型。

图 7-3 展示了初始的查询似然公式与更加一般化的 KL-散度检索模型的区别。图的上部是查询似然检索函数。KL-散度检索模型将查询词项频率泛化为一个概率分布，这是两个模型的唯一区别。该分布能够以更一般的方式对查询进行建模。查询语言模型可以用多种方法估计，包括使用反馈信息。这一方法被称为 KL-散度是因为该模型可以被解释为度量两个分布之间的散度(即区别)：一个分布对应查询模型 $p(w|\hat{\theta}_Q)$，另一个对应于文档语言模型。这里不会深入 KL-散度的细节，更详细的解释可参考附录 C。

查询似然

$$f(q,d)=\sum_{\substack{w\in d\\ w\in q}}\boxed{c(w,q)}\left[\log\frac{p_{\text{seen}}(w|d)}{\alpha_d p(w|C)}\right]+n\log\alpha_d$$

KL-散度（交叉熵）

$$f(q,d)=\sum_{w\in d,p(w|\theta_Q)>0}\boxed{[p(w|\hat{\theta}_Q)}\log\frac{p_{\text{seen}}(w|d)}{\alpha_d p(w|C)}]+\log\alpha_d$$

查询语言模型

$$p(w|\hat{\theta}_Q)=\frac{c(w,Q)}{|Q|}$$

图 7-3　KL-散度检索模型改变了表示查询的方式，使反馈信息能够被更方便地融入查询中 139

两个公式看起来几乎一样，除了泛化公式包含了一个由查询语言模型得到的词概率。我们仅考虑文档中出现且在查询语言模型中概率不为零的词，最终得到一个围绕所有被匹配词项的泛化的加和形式。我们可以由这个泛化的公式还原出初始的查询似然公式：将查询语言模型中词项的概率替换为查询中词项的相对频率$\frac{c(w, q)}{|q|}$，由于查询长度$n=|q|$是个常数，因此可以被消掉。

如图 7-4 所示，首先估计一个文档语言模型，再估计一个查询语言模型，并且计算两个语言模型的 KL 散度，记为$D(\cdot \,||\, \cdot)$。进一步根据包含查询词项的反馈文档估计一个反馈语言模型θ_F。该反馈语言模型的作用类似于 Rocchio 反馈中正样本文档的质心C_r，并可以通过一个线性插值与最初的查询语言模型相结合，从而与 Rocchio 方法类似，得到更新过的模型。

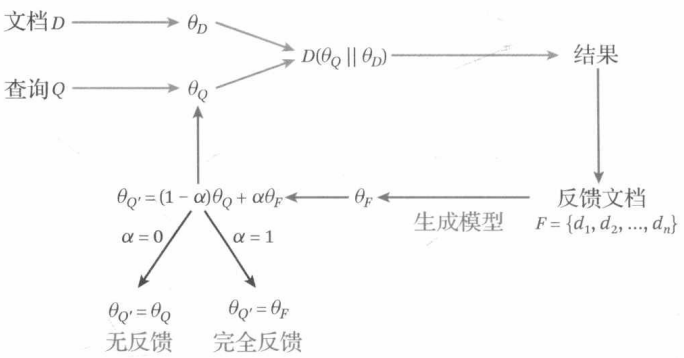
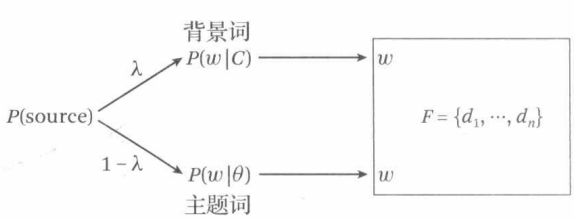


图 7-4　基于模型的反馈

参数$\alpha \in [0, 1]$控制反馈文档的强度。如果$\alpha=0$，相当于没有使用反馈；如果$\alpha=1$，相当于使用全部反馈信息而忽略最初的查询。当然，这些极端情况一般来说并不是我们需要的。现在，主要的问题是如何计算θ_F。

下面讨论一个估计θ_F的方法，该方法基于图 7-5 所示的生成模型。假设我们已经拥有正样本文档，它们可能是经过用户判断的、搜索结果的前k个文档、利用点击日志或者其
140 他方法获得的。正如之前所做的那样，根据这些文档估计语言模型的一个策略是假设这些文档由某个理想的反馈语言模型生成。这样需要统计所有反馈文档中的词项频率并进行规范化。但是，这样得到的分布能够有效地表示反馈吗？θ_F中最重要的词汇是哪些？

图 7-6 右侧列出了采用上面策略得到的语言模型，其中分数较高的词汇实际上都是一些常见词如 the。当与原始查询语言模型结合时，将往查询中增加很多这样的词，这对反馈来说没有好处。显然，需要除去这些停用词。事实上，第 2 章中采用背景语言模型来学习词关联的策略可以用来处理这种情况。这里要介绍一种更有理论基础的方法。我们假设这些不需要的词来自于背景语言模型。如果采用最大似然估计，一个独立模型很可能被迫分配高概率给像 the 这样的词，因为它们出现的频率非常高。为了降低这些词在模型中的概率，引入另外一个模型来解释这些常见词，背景语言模型便可以承担这一角



$$\log p(F|\theta)=\sum_i \sum_w c(w; d_i)\log[(1-\lambda)p(w|\theta)+\lambda p(w|C)]$$

最大似然 $\theta_F=\underset{\theta}{\operatorname{argmax}} \log p(F|\theta)$

λ = 反馈文档中的噪声

图 7-5　反馈混合模型

色，因为该模型会给常见词分配高概率。

查询："airport security"
混合模型方法

$\lambda=0.9$

w	$P(w\|\theta_F)$
security	0.0558
airport	0.0546
beverage	0.0488
alcohol	0.0474
bomb	0.0236
terrorist	0.0217
author	0.0206
license	0.0188
bond	0.0186
counter-terror	0.0173
terror	0.0142
newsnet	0.0129
attack	0.0124
operation	0.0121
headline	0.0121

网页数据集
前10个文档

$\lambda=0.7$

w	$P(w\|\theta_F)$
the	0.0405
security	0.0377
airport	0.0342
beverage	0.0305
alcohol	0.0304
to	0.0268
of	0.0241
and	0.0214
author	0.0156
bomb	0.0150
terrorist	0.0137
in	0.0135
license	0.0127
state	0.0127
by	0.0125

图 7-6 通过伪相关反馈习得的查询模型举例

可以假设这些词按照如下过程生成。首先，掷一枚硬币来决定使用主题词分布或是背景词分布。硬币正面朝上的概率为 $\lambda\in[0,1]$。如果正面朝上，则采用背景语言模型，接下来从背景语言模型中抽样出一个词。相对应的，有 $1-\lambda$ 的概率硬币正面朝下，那么将使用未知的主题模型来生成一个词。因此，这是一个混合模型，两个分布混合在一起，但我们不清楚什么时候使用哪个分布。也可以将反馈混合模型视为一个单独的分布，依然可以以一种随机的方式生成词项。每次生成哪个词既依赖于主题分布也依赖于背景分布，同时也依赖于混合参数 λ。如果 λ 较大，则倾向于按照背景分布生成词项，反之如果 λ 较小，将倾向于仅使用主题词。

按照这个思路，可以同以前一样采用最大似然估计来调整模型并找到合适的参数来解释数据。区别在于不是使用主题模型来解释所有的词，而是用混合模型来解释数据。这样的结果是模型不必给 the 这样的词分配高概率，它将分配高概率给那些在主题分布中频繁出现而在背景分布中不具有高概率的词。具体来说，该主题模型必须分配高概率给反馈文档中频繁出现而在整个文档集中并不很常见的词。

数学上，我们需要计算反馈文档集合 F 的对数似然，参数为 λ，反映反馈文档中的噪声。假设 λ 为一个常数，因此模型的参数仅包括词概率 θ，同最简单的一元语言模型相同。我们使用下面公式来估计反馈语言模型：

141
≀
142

$$\begin{aligned}\theta_F &= \arg\max_{\theta} \log p(F|\theta)\\ &= \arg\max_{\theta} \sum_{d\in F}\sum_{w} c(w,d)\cdot\log[(1-\lambda)\cdot p(w|\theta)+\lambda\cdot p(w|C)]\end{aligned} \tag{7-2}$$

我们要选择一个概率分布 θ_F，该分布可以使反馈文档集合在模型下的对数似然最大。这与最大似然估计的思想一样，需要解决一个优化问题。可以尝试所有可能的 θ 的取值，而后选择使整个表达式概率最大的 θ 取值。一旦得到 θ_F，便可以将其与初始查询模型进行插值从而融入反馈信息。然而实际中不可能尝试所有可能的 θ 的取值，因此可采取 EM 算法来估计模型参数[Zhai 和 Lafferty 2001]。这种模型称为混合模型，涉及多个成分模型的组合。第 17 章将进一步讨论此类模型的细节。

图 7-6 展示了在一个网页数据集上采用伪反馈模型得到的一些例子。排序前 10 的文档被认为是相关文档，分别设置混合模型参数 $\lambda=0.9$ 和 $\lambda=0.7$。查询是 airport security。

将搜索引擎返回的前 10 篇文档输送给混合模型。图中两个表展示了使用提出的模型学习到的词的概率。例如，airport 和 security 仍然获得很高的概率，因为它们在排序靠前的文档中出现的次数很多。但是，也可以看到 beverage、alcohol、bomb 和 terrorist 等词。显然，这些词与主题相关，与初始查询结合起来后可以帮助我们更好地匹配索引中的相关文档。

如果比较两个表，可以看到当 λ 的值被设置得较小时，背景模型不会经常被使用，因此依然可以看到一些常见词。注意，λ 可以影响选择使用背景模型生成文本的概率。如果不特别依赖背景模型，主题模型必须承担生成常用词的任务。设置 λ 的值较高，那么背景模型就会被更频繁地使用，承担更多生成反馈文档中的常用词的任务。这样，主题模型就
143 会变得具有很好的区分性，将会包含更多相关词项和更少的常用词。

总结一下，本节讨论了语言检索模型如何处理反馈机制。我们将初始的查询似然模型转换为更具一般性的 KL-散度模型。这一泛化手段使我们能够用一个语言模型来描述查询，该语言模型可以被用于融入反馈文档。本节还介绍了一种估计反馈模型参数的方法，用于区分主题词(与查询相关)和背景词(无用的停用词)。

本章则讨论了三种主要的反馈场景：相关反馈、伪相关反馈和隐式反馈，介绍了如何使用 Rocchio 方法来处理向量空间模型中的反馈以及查询模型估计的方法处理语言模型中的反馈，简要地描述了混合模型及其参数估计。本书后续还会提及其他参数估计的方法。

书目说明和延伸阅读

Salton 和 Buckley[1990]在早期对各种相关反馈技术进行了实证比较。在 TREC 实验中获得积极的结果之后，伪相关反馈变得流行起来(例如，Buckley 1994，Xu 和 Croft 1996)。语言模型中的反馈方法的比较可以在 LV 和 Zhai[2009]中找到。Lv 和 Zhai[2010]提出的位置相关模型是估计伪反馈查询语言模型最有效的方法之一。由于大量搜索引擎日志数据的可用性，尽管对用户点击的解释必须考虑位置偏差(在 Joachims 等[2007]中详细讨论过)，基于用户交互行为的隐式反馈已经成为非常重要且非常有效的技术，伴随用户的日益增多，网页搜索引擎系统运用该技术能够不断提升准确度。关于隐式反馈的参考书目可以在 Kelly 和 Teevan[2003]中找到。在网络搜索时代，隐式反馈经常在机器学习的排序函数中以反馈特征的形式实现，即排序学习技术；本书第 10 章将简要讨论它们。有关挖掘查询日志的更全面讨论，请参阅教程[Silvestri 2010]。

练习

144

7.1　当使用不同的反馈类型时，你应该如何设置 Rocchio 参数 α、β 和 γ？也就是说，如果使用伪相关反馈与用户提供的相关性判断进行比较，参数设置是否应该不同？那基于点击数据的隐式反馈呢？

7.2　想象一下，你负责一个大型的搜索引擎公司。你可以设计哪些其他策略来获取用户的相关性判断？

7.3　假设你的新策略之一是衡量用户在每个搜索结果文档上花费的时间 t。如何将每个文档的这个 t 合并到特定查询的反馈度量中？

7.4　在 MeTA 中实现 Rocchio 伪相关反馈。

7.5　在 MeTA 中实现语言模型的混合模型反馈。使用最方便的方法来估算 θ_F。比较不同的 θ_F 估计方法。

7.6 在实现 Rocchio 伪反馈之后，对具有相关性判断的数据集进行索引。在不同的 k 值上绘制 MAP(见第 9 章)。你有没有看到一些趋势？

7.7 在实现混合模型反馈之后，对具有相关性判断的数据集进行索引。在混合参数 α 的不同值上绘制 MAP(见第 9 章)。你有没有看到一些趋势？

7.8 设计一种启发式方法，为每一个查询自动确定伪相关反馈中的最佳 K 值。你可以查看查询本身，匹配文档的数量或原始结果中排名分数的分布情况。通过做实验来测试你的启发式想法。

7.9 设计一种启发式方法，为每一个查询自动确定反馈混合模型的最佳 α。你可以查看查询本身，匹配文档的数量或原始结果中排名分数的分布情况。通过做实验来测试你的启发式想法。

7.10 在反馈混合模型中，我们讨论了如何通过语言模型 θ_F 纳入正反馈文档。设计一个也包含一组负反馈文档的公式。确保你的新查询语言模型仍然是有效的概率分布。

7.11 在反馈混合模型中，我们使用一个概率模型来估计反馈 LM，以确保停用词不会影响重新配置的查询。我们是否需要为 Rocchio 反馈做这样的操作？ 145

7.12 在本章讨论的反馈方法中，我们假定只有一组相关和不相关的文档集合。实际上，我们有相关和不相关的文档的两个经过排序的列表。我们如何才能利用这些排序列表来处理反馈文档？换句话说，我们如何根据反馈文档与原始查询的相似程度有区别地对待这些反馈文档？可以考虑使用向量空间模型及查询相似度模型，或将两者结合。

7.13 在真实的搜索系统中，为所有观察到的查询存储修改的查询向量将占用大量的空间。怎么能优化所需的空间量？什么样的解决方案可以在空间和查询时间之间达到平衡？如何看待主要面向大多数用户或大多数查询进行优化的在线系统？ 146

第 8 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

搜索引擎实现

本章重点介绍如何实现信息检索(Information Retrieval，IR)系统或搜索引擎。一般来说，信息检索系统由 4 部分组成。

- 分词器(tokenizer)。该组件将文档视为原始字符串，并确定如何将大量文档字符串划分为单独的词项(或特征)。这些词项将被传递到索引器。这也许是整个系统中最重要的部分，因为糟糕的分词方法会影响索引的其他所有部分，以致影响终端用户的使用。
- 索引器(indexer)。这是处理文档并用适当的数据结构进行索引的模块。索引器可以脱机运行。主要挑战是如何使用有限的内存快速索引大量文档，其他挑战还包括添加和删除文档的技术支持。
- 打分器/排序器(scorer/ranker)。这是接收查询并返回有序文档列表的模块。这里面临的挑战是实现一个有效的检索模型，以便系统可以高效地对文档进行打分(排序)。
- 反馈/学习(feedback/learner)。这是负责相关反馈或伪反馈的模块。当存在许多隐含的反馈信息，例如可用的用户点击(出现在现代网络搜索引擎中)时，这个学习模块将会变得复杂。在前一章中已经详细讨论过，所以在本章只是简要介绍如何将它添加到现有系统中。

147 前三部分中有一些在当前所有的搜索引擎中使用的标准技术。第四部分对于反馈的技术的实现则高度依赖于学习方法和应用。前一章已讨论了一些常见的反馈方法。

本书将介绍两种附加的优化方法。它们并不是为了确保信息检索系统的正确性，而是能够令系统在速度和磁盘使用方面更有效率。

- 压缩(compression)。索引的文件可能会消耗数百 GB 甚至 TB 的空间。系统可以通过对索引中的数据(通常只是整数)进行无损压缩来节省磁盘空间并提高磁盘读取效率。[⊖]
- 缓存(caching)。即使文件检索存储经过了高效的数据结构设计及压缩，其系统仍然受到硬盘速度的束缚。因此在磁盘上，一种常见的做法是在前端 API 和文档索引之间添加一个缓存。缓存将能够保存频繁被检索的条目信息，从而减少慢速磁盘在检索期间搜索的次数。

本章的以下部分依次讨论上述组成部分。

8.1 分词器

文档分词是所有文本挖掘任务的第一步。它决定了如何表示文档。前一章提到，文档通常被表示成为文档向量的形式，其中每个索引对应一个单词。单词在索引中存储的值是特定文档中出现次数的原始计数。

当在文档向量上运用信息检索评分函数时，我们通常更倾向于采用词项计数的替代表

⊖ 正如即将要讨论的，词项字符串本身几乎总是被标示为词项 ID，为了提高效率，大多数对“单词”的处理实际上都是处理词项 ID。

示，例如平滑词项计数或 TF-IDF 权重。在实际搜索引擎系统中，对词项打分通常放到索引评分模块进行。因此，分词部分只是简单地对特征(词)进行原始计数，打分器可基于原始计数进一步计算一些加权的词项表示。类似 TF-IDF 这样的计算比单个文档的简单扫描更复杂(需要计算 IDF)。此外，我们希望打分器能够根据需要运用不同的打分函数，倘若只存储 TF-IDF 权重就意味着我们必须总是使用基于 TF-IDF 的打分函数。 148

因此，分词器的工作是将文档分割成可计数的特征或词项。继而一个文档由它所包含的词项及词项的多少来表示。打分器使用词项的原始计数有策略地选择检索打分函数(在前一章中讨论过)。

最基本的分词器是一个空格分词器(whitespace tokenizer)。这个分词器只是用空格分隔单词。运用空格分词器对句子“Mr. Quill's book is very very long.”进行标记其结果为

{Mr.:1,Quill's:1,book:1,is:1,very:2,long.:1}

更先进的一元词分词器可以首先对该句子进行小写处理，并基于标点符号分词。这里有一个特殊情况，Mr. 之后其实不应被分割(因为它形成了一个独特的词)：

{mr.:1,quill:1,'s:1,book:1,is:1,very:2,long:1,.:1}

当然，我们并不局限于使用一元单词表示形式。回顾第 4 章的练习能够看到，我们可以运用不同的方式表示文本。可以使用二元词汇、词性标签、语法解析树等特征或特征的组合。常见词(停用词)可以被删除，词也可以还原为它们的共同词干(stemming)。第 4 章的练习给出了使用 MeTA 进行转换的例子。实质上，索引器和打分器不应该关心词项 ID 是如何产生的，这是分词器的工作。

分词器的另一个常见任务是分配文档 ID。将文档用唯一编号进行标记比类似 **/home/jeremy/docs/file473.txt** 这样的字符串更高效。整数比较比字符串比较要快得多，占用的空间也更少。字符串词项和词项 ID 之间也存在这种关系。最终，总是要将词项或文档映射到它们的频度上。在 C++ 中，可以使用一些内部数据结构来实现这种映射，如 **std::unordered_map<std::string,uint64_t>** 。众所周知，使用这样的散列表能够以 $O(1)$ 的查找时间找到对应于特定 **std::string** 的 **uint64_t**。 149

然而，如果使用词项 ID，我们可以用 **std::vector<uint64_t>** 取而代之。这个数据结构占用较少的空间，并且允许通过使用词项 ID 这个整数作为索引，以 $O(1)$ 的时间从 **std::vector** 中获取每个词项对应的 **uint64_t**。因此，对于词项 ID 57，我们将查找数组中索引为 57 的位置得到相应信息。

在第二个分词器例子中使用词项 ID，我们可以将 mr. 的词项 id 设置为 0，将 quill 的词项 id 设置为 1，以此类推，那么文档向量类似如下的形式

{1, 1, 1, 1, 1, 2, 1, 1}

当然，真正的文档向量会更大也更稀疏，也就是说，大多数的维度的值都是零。

这个过程也称为特征生成。它定义了构建文档对象的组成元素，并提供了有意义的方法来比较它们。一旦定义了如何对文档进行概念化，就可以对它们进行索引、聚类、分类以及很多其他文本挖掘任务。正如引言中所提到的，分词可能是索引器中最关键的部分，因为所有的下游操作都依赖于它的输出。

8.2　索引器

现代搜索引擎能够索引比系统内存容量更大的数据。例如，维基百科数据仓库有大约 40GB 的未压缩文本。直至编写本书时，这比普通个人计算机的内存容量要大得多，尽管

这个数据量对于计算机科学研究人员来说很常见。TREC 研究数据集甚至可能高达数 TB。更不用说类似谷歌这样的真实世界中的搜索引擎公司能够索引整个网络的信息。

这就要求我们设计的索引系统，一次只能在内存中加载原始语料库的部分数据。此外，在索引文件上运行查询时，我们希望能够快速返回查询结果以确保搜索引擎的可用性。扫描语料库中的每个文档来匹配查询中词项的效率是低下的，即使对于相对较小的语料库也是如此。

倒排索引(inverted index)是搜索引擎中使用的主要数据结构。它允许快速查找包含任何给定词项的文档。相关的数据结构包括词典(即查找表，包含词项特定信息，如文档频率和它在记录文件中的位置)和记录文件(从任意词项的整数 ID 到一个文档 ID 列表的映
150 射，记录词项在这些文档中出现的频率)。

为了支持邻近匹配的启发式规则(奖励匹配的词项相邻的情况)，通常需要存储每个词项出现的位置。这样的位置信息可以用来检查所有的查询词是否在特定的文本窗口内被匹配，例如，它可以用来检查短语是否被匹配。这些信息作为一条记录被存储在记录文件中，每条记录都针对一个特定文档。

图 8-1 展示了词典和记录文件的表示。图像中的箭头实际上是在记录文件中表示位或字节索引的整数偏移量。

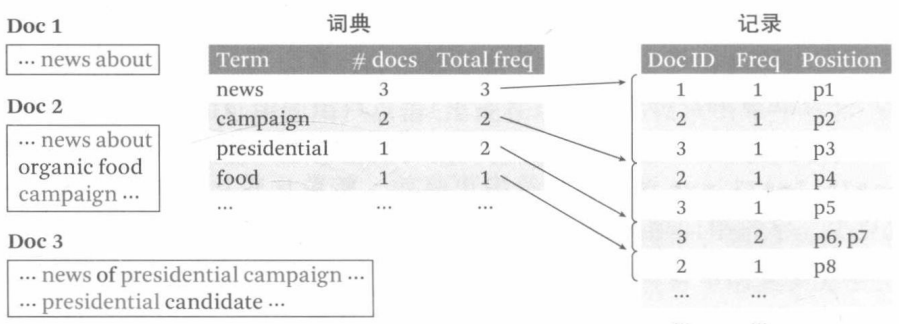


图 8-1　倒排索引记录及词典文件

例如，如果我们想对 ID 为 56 的词项“computer”进行打分，那么在词典中查找 56，得到的信息可能是：

```
Term ID: 56
Document frequency: 78
Total number of occurrences: 443
Offset into postings file: 8923754
```

当然，词典实际上只会存储 56→{78，443，8923754}。由于分词器顺序地分配词项 ID，我们可以将词典表示为由词项 ID 索引的大数组。大规模数组中的每个元素将存储(文档频率，总数量，偏移量)信息元组。如果在大的记录文件中定位 8923754 的位置，我们可以看到类似以下的信息

```
Term ID: 56
Doc ID: 4, count: 1, position 56
Doc ID: 7, count: 9, position 4, position 89, position...
151 Doc ID: 24, count: 19, position 1, position 67, position...
Doc ID: 90, count: 4, position 90, position 93, position...
Doc ID: 141, count: 1, position 100
Doc ID: 144, count: 2, position 34, position 89
:
:
```

这是 ID 为 56 的词项在 78 篇文档中出现的数量和位置信息。请注意文档 ID(以及位置信息)是如何以升序存储的，在记录文件的压缩环节这一特点将得以利用。还要注意词典和记录文件的大小差别。词典中的每个条目只需存储三个信息。在记录文件中，出现过该词项的文档均至少存储三个值(文档 ID，数量，位置)。如果一个词项在所有文档中均出现，我们将从语料库中获得一个以总文档个数为长度的列表。所有词项都以这种形式记录相关信息。因此，我们经常假设词典可以放入内存中，而记录文件驻留在磁盘上，利用词典中的指针进行搜索。

索引是基于一组经过分词的文档来创建这些数据结构的过程。一种流行的索引方法是下面基于排序的方法。

- 依次扫描所有原始文档。首先为每个文档分配一个 ID。对文档分词从而为词项分配 ID，并根据需要创建新词项 ID。
- 在扫描文档时，为每个文档－词项构成的对进行数量累计，并为内存中的文档子集建立一个倒排索引。当达到内存上限时，将不完整的倒排索引写入磁盘。(它与最终的记录文件的格式相同，但更小。)
- 继续这个过程来生成很多不完整的倒排索引(称为“run”)，并全部写到磁盘上。
- 以成对的方式合并所有不完整的倒排索引直到产生一个单一有序(按词项 ID)的记录文件。这个算法本质上是合并排序中的 **merge** 函数。
- 创建记录文件后，扫描记录文件创建词典，并为每个词项 ID 分配偏移值。

图 8-2 说明了文档如何基于文档 ID 顺序产生词项。然后，将多个文档中的词项按照词项 ID 先在内存中排序，然后在磁盘中更新相关信息。 152

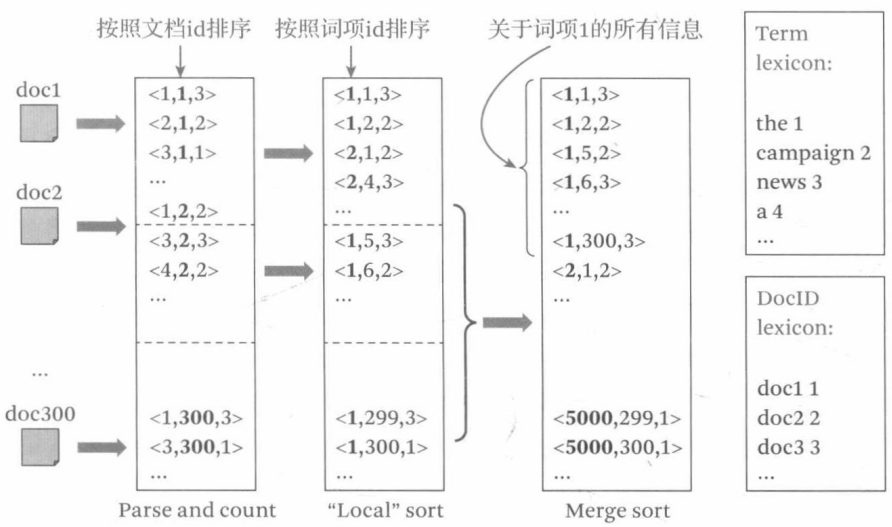


图 8-2　倒排索引块中基于排序的倒排过程

前向索引(forward index)的创建方式与倒排索引类似。前向索引将文档映射到词项列表上，而不是将词项映射到文档。当除了搜索还需进行其他操作时，前向索引非常有用。例如，聚类或分类任务时能够一次访问整个文档的内容。使用倒排索引很难有效完成这样的任务，因为必须扫描整个记录文件来查找特定文档中出现的所有词项。因此，通过前向索引结构能够为每个文档 ID 记录一个词向量。

在下一节中，我们将看到如何使用倒排的词项到文档的映射减少查询打分的时间。还有其他与前向索引相关的提升效率环节，例如压缩和缓存。

8.3　打分器

现在我们有了倒排索引，那么如何运用它高效地对查询进行打分呢？想象一下，如果没有倒排索引，只有前向索引，要对查询向量进行评分，我们需要在前向索引中对每个文
153 档进行一次迭代，并在每对(文档，查询)运行评分函数。

很可能许多文档不包含任何查询词项(特别是删除停用词之后)，这意味着它们的排序分数将为零。为什么还要浪费气力去对这些文件打分呢？这正是倒排索引带来的好处：我们只对至少匹配一个查询词的文档进行评分，也就是说，只对得分不为零的文档进行评分。我们假设(并在实践中得到验证)只对包含查询中的词项的文档进行评分，将会减少很多计算量。这将引出运用倒排索引进行打分的第一个算法。

8.3.1　逐个词项排序

一旦建立了倒排索引，就可以使用算法 8-1 在倒排索引 Idx 上高效地逐词对查询 q 进行打分。

算法 8-1　逐个词项排序

```
scores = {}      // score accumulator maps doc IDs to scores
for w ∈ q do
    for d, count ∈ Idx.fetch_docs(w) do
        scores[d] = scores[d] + score_term(count)
    end for
end for
return top k documents from scores
```

对于每个词项，在倒排索引中获取相应的记录。根据需要创建文档分数累加器(保存每个文档的累计分数的变量)。对当前词项扫描倒排索引的记录，对每个记录(对应于包含该词项的一个文档)基于一些词项加权方法(即 score_term 函数)更新其分数累加器，例如 Okapi BM25。当处理完查询中的所有词项时，分数累加器应该包含所有文档的最终分数，每个文档至少包含一个查询词项。如果文档不匹配任何查询词项，不需要为其创建分数累加器。

实际上，fetch_docs 函数会返回一些对象，这些对象包含文档中的当前词项的信息，如计数、背景概率或其他词项打分函数需要的必要信息。

154 一旦遍历了所有的查询词项，分数累加器便完成了。我们只需要按照累积的分数对这些文档进行排序，并返回前 k 个结果。同样，只对包含查询词项的文档进行排序，而不是对索引中的每个文档进行排序(即使其分数为零)，这样就节省了排序时间。

8.3.2　逐个文档排序

逐个词项排序的一个缺点是分数累加器的规模与匹配至少一个查询词项的文档数量相同。虽然与使用索引中所有文档相比有了巨大改进，但仍然可以使这个数据结构更小。

我们可以一次对整个文档进行评分，而不是对每个匹配的查询词多次遍历每个文档。由于大多数(如果不是全部)搜索是 top-k 搜索，所以在任一时间点只需保存前 k 个文档。这样做的前提是数据结构中有对每个文件的完整评分。否则，就像逐个词项进行排序一样，一个文档开始时可能分数低于另一文档，当更多词项被计分时，两者分数的高低可能会发生变化。

可以使用优先队列保存优先级最高的 k 个文档。使用倒排索引，我们可以获得需要被

打分的文档 ID 和相应数据记录的列表。当对一个完整的文档进行打分时，它被添加到优先队列。我们对低分文档分配高优先级；这是因为在添加第 $k+1$ 个文档后，可以尽快(以 $O(\log k)$的时间复杂度)删除最低分数文档，只保留前 k 个文档。一旦遍历了所有的文档 ID，我们可以很容易地对最终保留的 k 个文档进行排序并返回。请参阅算法 8-2。

算法 8-2　逐个文档排序

```
context = {}      // maps a document to a list of matching terms
for w ∈ q do
    for d, count ∈ Idx.fetch_docs(w) do
        context[d].append(count)
    end for
end for
priority_queue = {}      // low score is treated as high priority
for d, term_counts ∈ context do
    score = 0
    for count ∈ term_counts do
        score = score + score_term(count)
    end for
    priority_queue.push(d, score)
    if priority_queue.size() > k then
        priority_queue.pop()      // removes lowest score so far
    end if
end for
Return sorted documents from priority_queue
```

从逐个词项排序的分数累加器中提取前 k 个文档时，也可以使用类似的优先级队列方法，但是在此之前，仍然需要存储所有文档的分数。

8.3.3　过滤文档

另一个常见的任务是只返回符合特定标准的文档。例如，系统的索引可能将新闻文章的日期存储为元数据。在 top-k 搜索中，系统可以只返回去年编写的文档。这是一个常见的文档过滤(document filtering)问题。

运用逐个词项排序方法，可以通过不在分数累加器中更新分数(因此不会将这些文档 ID 插入结构中)忽略不在正确的日期范围内的文档。运用逐个文档排序方法时，如果在创建特定文档的上下文时没有通过过滤器，则可以简单地跳过该文档。

过滤器不会很复杂，因为它本质上是一个布尔函数，接受一个文档并对该文档是否应该被插入评分文档列表进行判断。过滤函数可以作为评分函数的可选参数，通常需要访问文档元数据存储结构(通常是数据库)和前向索引(以便过滤包含特定词项的文档)。

8.3.4　索引分片

索引分片指特定搜索引擎保留多个倒排索引。在对记录块进行合并的过程中，当块的数量等于所需分片的数量时停止合并即可实现索引分片。所有数据可被视为存储为一个最终的块，只不过被分解成几个部分。

但是，为什么要多个倒排索引块呢？设想当分片数量等于搜索系统中的线程数量(或集群节点)时，你大概可以想象一个算法，每个线程在其各自的分片中搜索匹配词项，然后将最终搜索结果合并在一起。这种类型的算法设计——分发工作进而合并结果——是一种非常

常见的范式，称为 Map Reduce。我们将在以后的章节中讨论它的共性和其他许多应用。

8.4　反馈实现

第 7 章讨论了标准信息检索系统中的反馈，介绍了两种反馈的实现方法：向量空间模型中的 Rocchio 反馈和查询似然混合反馈模型。

两者都可以用前面章节中描述的倒排索引和文档元数据来实现。对于 Rocchio 反馈，可以使用前向索引来获得查询和反馈文档的向量，在该组向量上运行 Rocchio 算法。混合模型反馈方法需要在反馈文档中学习语言模型；同样，这可以通过使用前向索引的词项计数来有效地实现。语料库中唯一需要的其他信息是每个词项在语料库中的背景概率，可以存储在词项词典中。

有了这些信息，现在可以创建一个在线(或“内存中”)伪反馈方法。回想一下伪反馈是利用搜索中返回的前 k 个文档，并假定它们是相关文档。以下过程可用于在线反馈。

- 运行用户的原始查询。
- 使用前 k 个文档和前向索引来修改查询向量(Rocchio 方法)或估计语言模型并与反馈模型(查询似然混合模型)进行插值。
- 用修改后的查询重新进行搜索，并将新结果返回给用户。

这个简单的反馈模型有好处和坏处。首先，它只需要很少的内存和磁盘存储空间，因为一旦返回新的结果，每个被修改的查询就“被遗忘”了。因此，我们不需要为索引创建任何额外的存储结构。

它的缺点是所有的处理都是在查询时完成，计算代价高，尤其是对于有着许多用户的搜索引擎。完全相反的折中办法是将每个修改后的查询都存储在数据库中，搜索时先到数据库中查找查询对应的扩展形式，再针对扩展形式运行一次搜索函数。当然，这不具有可
157 行性，因为查询的数量很快会使数据库爆炸，更不用说向索引添加更多文档后会使存储的查询向量无效(因为新文档也可能与原始查询相匹配)。

在实践中，我们可以在这两个极端之间做一些妥协，例如只存储非常常见的查询的扩展形式，或者使用查询相似性来搜索已经保存的相似查询。后面讨论的缓存技术也适用于反馈方法，具体考虑如何对词项进行缓存和对扩展查询进行缓存。

当然，这只涉及伪反馈方法。还有可以存储在数据库中的点击数据以及可以以相同方式存储的相关性判断。一旦我们知道了所选择的反馈方法中包含哪些文档，所有实现都是相同的，因为它们都是处理反馈文档的列表。

8.5　压缩

检索系统中的另一个技术是整数压缩，该技术用于压缩规模非常大的记录文件。压缩索引不仅更小，而且加载到主内存时也更快。整数压缩(一般压缩也同样)的总体思路是利用值的非均匀分布特性。直观地，我们为频繁出现的值分配一个短代码，为不常见的值分配长代码。最优压缩率与随机变量的熵相关，偏态分布将具有较低的熵，因此更容易压缩。

所有的压缩方法都需要支持随机访问解码这一点很重要。也就是说，我们可能希望在记录文件中寻找特定的位置并开始解压，而不必解压所有以前的数据。

由于倒排索引中的记录是按顺序存储的，所以可以利用这个特性，根据文档 ID(和位置信息)在索引间的间隙进行压缩。否则文档 ID 将被均匀地分配，但是它们的间隙的分布将是不均衡的，因为当一个词项频繁出现时，相应的倒排列表将包含许多文档 ID，导致

很多小的间隙。考虑以下文档 ID 列表的例子：

{23，25，34，35，39，43，49，51，57，59，...}

我们可以存储它们之间的偏移量而不是存储这些确切的数字；这就产生了更小的数字，这些数字更容易压缩，因为它们占用更少的空间并且出现次数更频繁： 158

{23，2，9，1，4，4，6，2，6，2，...}

要获得实际的文档 ID 值，只需将偏移量添加到以前的值。所以第一个 ID 是 23，第二个是 23+2=25，第三个是 25+9=34，等等。

本节将讨论以下类型的压缩，其中部分方法会在间隙编码值上进行操作：

- 一元编码(按位)；
- γ 编码(按位)；
- δ 编码(按位)；
- vByte(按块)；
- 参考框架(块)。

8.5.1 按位压缩

按位压缩编写原始的二进制数字，而不是写出表示数字的字符串(如“1624”)，或是固定字节宽度的块(如 4 字节的整数“00000658”)。当一个数字的表示结束时，下一个数字开始。数字表示没有固定的宽度或长度。使用按位压缩意味着对编码的每一位执行一些有序的位操作，以便将压缩的整数“构建”回到其原始形式。

一元(unary)。一元编码是最简单的方法。为了写出整数 k，我们只需写 $k-1$ 个 0 后跟一个 1。1 作为分隔符，让我们知道什么时候停止读入：

1→1
2→01
3→001
4→0001
5→00001
19→0000000000000000001

请注意，我们不能编码数字零，大多数其他方法也是如此。一元编码序列的一个例子是

0001001000100000001010001000001=4，3，4，8，2，4，5 159

只要词典有一个指向压缩整数开始的指针，可以很容易地支持随机访问解码。该编码的另外一个性质是小数目占用较少的空间，而较大的数占用较多的空间。接下来的两种压缩方法是基于一元编码的概念构建的。

γ 编码(gamma)。要用 γ 编码对一个数字编码，首先将数字表示为二进制。令 k 是二进制字符串中的位数。然后，将 $k-1$ 个 0 加到二进制数之前：

1→1
2→010
3→011
4→00100
5→00101
19→000010011
47→00000101111

解码时读取0直到遇到1，记录已读取的0的数目k。读取1和另外的k位二进制。请注意，所有的γ码的位数都是奇数。

δ编码(delta)。简而言之，δ编码是先用γ编码对一个数字进行编码，然后再对编码的前缀(包括1)的位数进行γ编码：

1→1→1
2→010→0100
3→011→0101
4→00100→01100
5→00101→01101
19→000010011→001010011
47→00000101111→0011001111

要解码，在开始位置解码γ编码得到一个整数k。写下1，然后读取下一个$k+1$位(包括你写的1)二进制数。可见，δ压缩开始时比γ编码有更多的位数，但随着数字变大，最终变得更有效率。哪个压缩方法在压缩比(compression ratio)方面更好可能取决于特定的数据集(整数的分布)。压缩比就是未压缩的大小除以压缩的大小。因此，压缩比3比压缩比2更好(压缩后的文件更小)。

8.5.2　块压缩

160

按位编码由于其细粒度分布模型可以实现非常高的压缩比，然而其缺点是编码和解码处理的数据量较大。为了读取一个整数，每一位都需要被读取。块压缩尝试每次读取字节而不是位，来缓解这个问题。在块压缩方案中，与前面三种方案中的每位至少一个操作相比，通常每字节只需要一个按位操作(例如，可以计算对整数47进行δ编码需要多少位操作)。

数据块压缩旨在减少解码所需的CPU指令的数量，但会增加存储空间。我们将要研究的两个块压缩方法主要处理单位是字节而不是位。

vByte代表可变字节(variable byte)编码。它使用一个字节的低7位来存储一个二进制数，最高位作为一个标志。该标志表示解码器是否应该继续读取二进制数。下面的括号是为了强调而添加的。

1→(0)0000001
2→(0)0000010
19→(0)0010011
47→(0)0101111
127→(0)1111111
128→(1)0000000(0)0000001
194→(1)1000010(0)0000001
678→(1)0100110(0)0000101
20 000→(1)0100000(1)0011100(0)0000001

解码器的工作原理是保存总和信息(从零开始)，并将每个处理的字节添加到总和中。注意这些字节是如何逆向“链接”在一起的。对于每一个“链接”，我们将字节通过左移来添加$7\times k$，其中k是到目前为止读取的字节数。因此，解码整型数20000的总和如下

$$(0100000\ll 0)+(0011100\ll 7)+(0000001\ll 14)$$

等价于

$$
\begin{aligned}
&0100000_b\\
&+00111000000000_b\\
&+00000010000000000000_b\\
&=00000010011100010000_b\\
&=20000_d
\end{aligned}
$$

161

在这种方法中，我们正在编码的“块”是字节。

参照标准(frame of reference)编码采用大小为 k 的块并一次编码 k 个整数，因此该方法中的块实际上是一个数字序列。以下面的大小为 $k=8$ 的块为例：

$$\{45, 47, 51, 59, 63, 64, 70, 72\}$$

我们通过从每个元素中减去列表中的最小值来转换这个块：

$$\{0, 2, 6, 14, 18, 19, 25, 27\}, \min=45$$

到目前为止，这与之前讨论的间隙编码相似。然而，我们不是使用按位压缩(如 γ 编码或 δ 编码)对每个值进行编码，而是简单地使用可能的最小位数的二进制。由于这个块的最大值是 27，我们将每个整数存储在 5 位中：

$$\{00000, 00010, 00110, 01110, 10010, 10011, 11001, 11011\}, \min=45, \text{bytes}=5$$

这可能看起来像

$$45, 5, 00000000100011001110100101001111001110 11$$

其中 45 和 5 可以被方便地存储(例如固定的二进制长度)。这个方法也减少了按位操作的次数，因为每次读取 5 位数据块，并将值加上基值 45 来重新创建序列。另一个不错的附带好处是我们知道存储在这个块中的最小值和最大值的上界。因此，如果我们正在寻找一个特定的整数(比如一个文档 ID)，如果 ID$<$45 或 ID$>$45$+2^5$，则可以跳过这个块，不必解压。

8.6　高速缓存

虽然我们设计的倒排索引结构非常有效，但仍然存在磁盘延迟的问题。出于这个原因，真实世界的搜索引擎也常常在内存中使用某种缓存结构来记录数据对象。本节将概述两种不同的缓存策略。

缓存的基本思想是促使经常访问的对象能够被快速获取。为了达到这个目的，我们试图将经常访问的对象保存在内存中，因而不需要去访问磁盘。在我们的例子中，访问的对象是记录文件。根据齐普夫定律[Zipf 1949]，我们预计相对少数的记录列表(以条目 ID 为关键字)会占据大部分访问时间。但是，如何知道要存储哪些词项？即使我们知道哪些词项频繁被查询，如何为内存消耗设定一个中止条件？

162

我们描述的两种缓存设计解决了这两个问题——最频繁使用的词项能获得最快的访问速度；可以为缓存设置一个容量上限，所以它不会太大。

8.6.1　LRU 缓存

我们首先考虑最近最少使用(Least Recently Used，LRU)缓存，如图 8-3 所示。该 LRU 算法如下，假设我们想要检索词项 ID 为 x 的记录列表。

- 首先，在缓存中搜索词项 ID x。
- 如果它存在于缓存中，则返回 x 所对应的记录列表。
- 如果它不存在于缓存中，则从磁盘检索并将其插入缓存。如果这导致缓存超过其上限，则删除最近最少使用的记录列表。

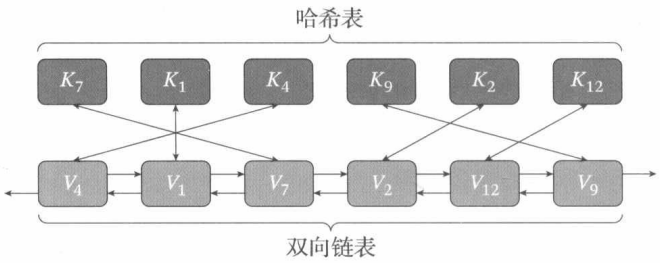


图 8-3　最近最少使用缓存的实现

我们希望能够快速搜索缓存，所以使用一个散列表来存储词项 ID 作为键(key)，记录列表作为值。这使得以 $O(1)$代价分期插入、查找和删除操作成为可能。LRU 缓存中有趣
163 的部分是如何确定对象的访问顺序。为此，我们用双向链表将值链接在一起。插入或访问元素后，将在 $O(1)$时间内将该元素移动到列表的头部(前部)。当需要删除 LRU 项时，我们查看链表尾部的元素，并同样以常数时间删除它。

8.6.2　DBLRU 缓存

双桶 LRU(double barrel LRU)缓存最初在流行的 Lucene 搜索引擎[⊖]中使用。它是 LRU 缓存的简化近似形式。图 8-4 显示了一个大小为 6 的 DBLRU 缓存。

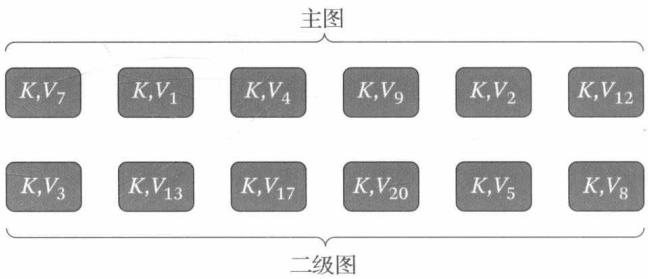


图 8-4　双桶最近最少使用缓存

DBLRU 缓存只是两个哈希表，分别称为 **primary** 和 **secondary**。算法如下，假设我们要检索 ID 为 x 的词项的记录列表。

- 首先，在 **primary** 中搜索词项 IDx；如果存在，则返回。
- 如果不在 **primary** 中，搜索 **secondary**。
- 如果处于 **secondary**，请将其删除。将其插入 **primary** 并将其返回。如果这导致 **primary** 容积达到上限，清除 **secondary** 的全部内容。然后交换两个哈希表。
- 如果它不在 **secondary** 中，则从磁盘中检索并插入 **secondary**。

该缓存具有一个层次化的使用过程：**primary** 包含的元素相较于 **secondary** 会被更频繁地访问。所以当缓存填满时，**secondary** 被清空以释放内存。虽然 DBLRU 缓存的时间准确性不如 LRU 缓存，但是这是一个更简单的设置，能转换为更快的访问时间。这两
164 个缓存在速度和准确性之间有一个折中。

书目说明和延伸阅读

《Managing Gigabytes》[Witten 等 1999]是关于搜索引擎实施的经典参考书。其他有关

⊖ https://lucene.apache.org/

信息检索的书籍，如《Introduction to Information Retrieval》[Manning 等 2008]、《Search Engines：Information Retrieval in Practice》[Croft 等 2009]以及《Information Retrieval：Implementing and Evaluating Search Engines》[Buttcher 等 2010]也是关于搜索引擎的实现的很好的书籍。

练习

8.1　为什么使用倒排索引而非一个更简单的方案来对文档进行评分？

8.2　对于以下符号，请解释我们能否有效地从默认(标准)倒排索引记录文件中获取它们的值。有效的方式是指通过一次查询而非扫描整个索引。

符号	含义
$\|d\|$	已知文档中的所有词项的数量
$\|d\|_u$	已知文档中不同词项的出现数目
df	包含某一特定词项的文档数目
$c(w, C)$	语料库中 w 出现的次数
$c(w, d)$	文档 d 中 w 的出现次数
$p(w, d)$	w 出现在文档 d 中的概率

8.3　对于上一个问题中无法从记录文件中高效提取数值的符号，请说明需要搜索哪些辅助结构，或为什么无法高效地检索它们的值。

8.4　假如每个文档都被标记了情感极性(正面或负面)。如果你只想基于每次查询返回某种指定情绪的文档，请简述方法来说明如何对文档进行评分。

8.5　使用所介绍的几种压缩方法压缩以下整数序列：

$$\{34, 36, 39, 42, 47, 48, 49, 51\}$$

8.6　根据齐普夫定律，下列哪种策略对于减小倒排索引的大小更为有效？1) 删除出现次数等于或小于 10 次的所有单词或 2) 删除出现频度最高的 10 个单词。 165

8.7　META 支持哪种类型的整数压缩？编程实现本章讨论过而在工具包中未被实现的一个压缩算法。

8.8　META 使用哪种方式评分(逐个文档或逐个词项)？提示：请参阅文件 **meta/src/index/ranker/ranker.cpp**。

8.9　实现其他类型的查询评分，并将其运行时间与 META 中的现有方法进行比较。

8.10　META 包含对 DBLRU 缓存的一个实现。尝试在实验中对高速缓存大小进行参数调整，保持其他变量不变、将检索速度与不同的高速缓存大小进行比较。

8.11　META 目前没有实现本章讨论的 LRU 缓存。你能实现一个相应算法，并将它与 DBLRU 缓存在内存使用情况和速度上进行性能比较吗？ 166

第9章

Text Data Management and Analysis：A Practical Introduction to Information Retrieval and Text Mining

搜索引擎评价

本章重点为文本检索系统的评价。之前的章节讨论了许多不同的文本检索方法和排序函数，但是如何知道哪一个最好呢？为了回答这个问题，必须对它们进行比较，这就意味着必须评价这些检索方法。这是本章的重点。我们从评价的方法论开始。然后，比较基于集合的检索与基于排序列表的检索，以及多层次相关性的判断。最后，讨论评价中的实际问题并进行练习。

9.1 引言

首先思考为什么必须做评价。有两个主要原因。第一，必须通过评价来确定哪种检索方法更好。这对于促进知识积累是非常重要的，否则无法知道一个新的思路是否比旧的思路更好。本书第6章讨论了文本检索问题，并将它与数据库检索进行了比较。由于搜索引擎的评价必须依赖于用户，这是非常具有挑战性的问题。因此，需要确定如何让用户参与，并对不同方法进行公正的比较。

第二，必须通过评价来反映整个文本检索系统的实际效用(而不是针对具体的方法)。想象你正在构建自己的应用程序，你一定有兴趣知道你的搜索引擎为用户工作的表现如何。在这种情况下，度量必须反映系统在实际应用中对真实用户的效用，而不仅仅是针对每个检索结果。通常，这是通过用户调查完成的——人们通过系统与语料库进行交互。

在比较不同方法的情况下，使用的度量需要与用户的效用正相关。因而，只要观察哪个系统的度量足够好，便可以确定哪一种方法工作得更好。这通常是通过一个测试集合来完成的，这也是本章讨论的主要思想的基础。整体上讲，评价对于比较不同算法和提高搜索引擎性能都具有重要意义。

9.1.1 要度量什么

可以从很多方面去评价搜索引擎，这里讲三个主要方面。

有效性或精确性(effectiveness or accuracy)。搜索结果有多精确？在这种情况下，需要对系统将相关文档排在不相关文档之上的能力进行度量。

效率(efficiency)。用户能否快速地获取搜索结果？回答一个查询需要多少计算资源？在这种情况下，需要测量系统的时间和空间开销。

可用性(usability)。该系统对实际的用户任务有多大的用处？在这方面，搜索接口和其他很多方面也很重要，通常需要去做用户调查。

本书将主要讨论有效性和精确性的度量，因为效率和可用性维度并不是搜索引擎所特有的(评价其他软件系统同样需要)。其他书中也有资料对其进行了很好地整理，所以我们建议读者参考Harman[2011]来进一步阅读这方面的内容。其他方面的读物包括Sanderson[2010]和Kelly[2009]，涵盖了用户研究和A-B测试的概念(本章后面部分会讨论)。

9.1.2 Cranfield评价方法

Cranfield评价方法是在20世纪60年代开发的一种针对系统组件的实验室测试策略。

事实上这是一种非常有用的方法，不仅对搜索引擎评价有用，而且几乎对所有的经验性任务都是非常有用的。例如，在图像处理或者其他经验性定义问题的领域，通常要使用这样的方法。

这个方法的基本思想是构建合理的测试集并在其上定义度量。一旦建立了这种测试集，就可以反复测试不同的算法或者想法。利用测试集，我们可以定义能够量化系统或算法性能的度量。构建的测试文档集合与搜索应用中的真实文档集合非常类似。我们还需要一组查询或主题的样本集合来模拟用户的信息需求。然后，需要得到相关性的判断，即针 168
对哪些查询应该返回哪些文档的判断。理想情况下，必须由构建查询的用户做出判断，因为他们确切知道文档(搜索结果)将用于做什么。最后，我们必须采取度量来量化系统的结果与基于用户相关性判断构建的理想排序列表的匹配程度。

这个方法对于评价检索算法非常有用，因为测试集可以重复使用很多次。由于其对不同方法的评价过程完全相同，所以它能够公正地比较多种方法。也就是说，采用相同的标准、相同的语料库和相同的相关性判断来比较不同的算法。这使我们能够使用相同的方法将新的算法和许多年前的旧算法进行比较。

图 9-1 描述了 Cranfield 评价方法的工作原理。如上所述，我们需要的一组查询展示在这里，有 Q_1、Q_2，等等。我们还需要文档集合 D_1，D_2，…。图的最右侧是关于每个文档的相关性判断(加号或减号注释)，指明它是相关的(加号)还是不相关(减号)。因为只有两个层次的相关性，所以这些都是关于特定查询，对文档进行的二元判断。例如，D_1 和 D_2 被判断为与 Q_1 相关，D_3 被认为与 Q_1 无关。这些判断是由与每个系统交互的用户创建的。一旦有了这些判断，便可以比较两个或更多的系统。每个查询在每个系统上运行，我们将考察每个系统返回的文档。

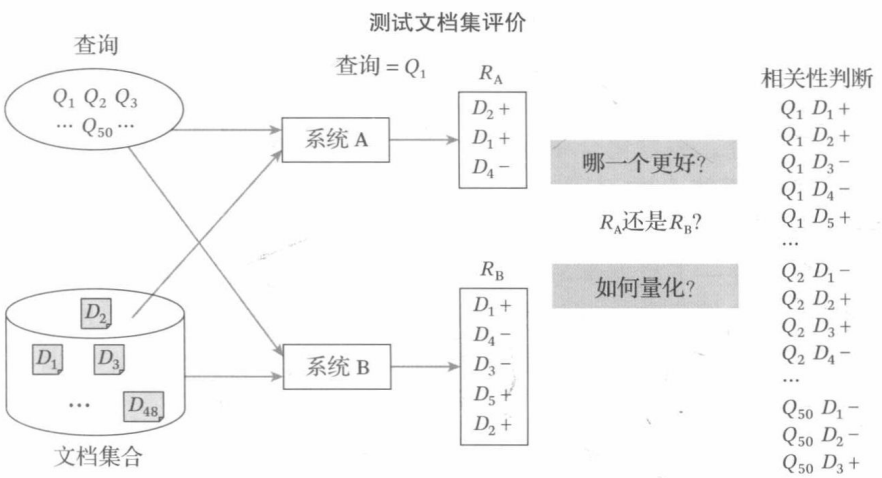


图 9-1　Cranfield 评价方法示意图　169

假设查询是 Q_1。在图中，R_A作为系统 A 中的排序结果，而 R_B作为系统 B 中的排序结果。因此，R_A是系统 A 的近似相关文档，R_B是系统 B 的近似相关文档。让我们看看结果——哪个更好？作为用户，最喜欢哪一个？两个系统的结果中有一些不同的文档，也有一些共有的文档。如果观察这些结果，你可能会觉得系统 A 的结果会更好，因为并没有过多的文档被返回，并且在三个返回文档中有两个是相关的。太好了，系统 A 是精确的。但从另一方面看，也可以说系统 B 比较好，因为它返回了更多的相关文档，它返回了三个而不是两个。那么哪一个更好，我们如何量化呢？这个问题在很大程度上取决于用户的任务，也取决

于具体的用户！对于一些用户来说，如果用户对获取所有相关文档不十分关心，那么系统 A 可能更好，这样用户不必读得太多。但是，也有另一些用户可能需要尽可能多的相关文档，例如在撰写文献调研时。此时，可能会发现系统 B 更好。无论哪种情况，都必须定义量化用户信息需求的度量，且需要定义多个度量，因为用户可能从不同的角度看待搜索结果。

9.2　集合检索的评价

本节将考察面向文本检索系统评价的基本度量，讨论如何设计这些基本的度量以及如何定量地比较两个系统。尽管系统返回一个排序的文档列表，本节讨论的评价指标将返回的搜索结果视为文档集合；也就是说，并没有考虑返回结果的顺序。这些度量和它们背后的直觉可被用来设计其他更复杂的方法，但它们本身也是相当有价值的。

9.2.1　准确率和召回率

回到图 9-1。哪一组结果更好，系统 A 还是系统 B？现在可以讨论如何量化它们的表
170 现。假设语料库中总共有 10 篇文档与当前查询 Q_1 相关。当然，右边显示的相关性判断并不包括全部 10 个。我们在那里只看到了 3 篇相关文档，但是可以想象还有其他文档也为这个查询被进行判断。直觉上，我们认为系统 A 更好，因为它没有太多的噪声。特别是，3 个结果中有 2 个是相关的。另一方面，系统 B 提供了 5 个结果，只有 3 个是相关的。基于此，看起来系统 A 更加精确。这可以通过准确率(precision)来捕获，我们只需计算所有检索结果中相关文档所占的比重。100%的准确率意味着所有检索到的文档都是相关的。因此在这种情况下，系统 A 具有 $2/3=0.66$ 的准确率。系统 B 有 $3/5=0.60$ 的准确率。这表明，根据准确率来说系统 A 更好。

但是我们也提到，其他一些用户希望检索尽可能多的相关文档，这些用户可能会喜欢系统 B。所以，在这种情况下，需要将总的相关文档数量与实际检索到的相关文档数量进行比较。这可以通过另一个度量来描述，称为召回率(recall)。该度量用来衡量检索结果覆盖相关文档的完整性。假设集合中有 10 个相关文档。系统 A 检索到其中两个，所以召回率是 $2/10=0.20$。系统 B 的召回率是 $3/10=0.30$。因此，根据召回率来说系统 B 更好。

这两个度量是评价搜索引擎的基础。它们非常重要，在许多其他评价问题中也被广泛使用。例如，如果关注一下机器学习的应用，你会发现所有类型的任务都倾向报告准确率和召回率。现在，让我们更精确地定义这两个度量，以及如何使用这些度量来评价检索到的文档集合，考察它们对相关文档的趋近程度。根据每个文档的情况，可以将搜索结果区分为 4 种类型，如图 9-2 所示。

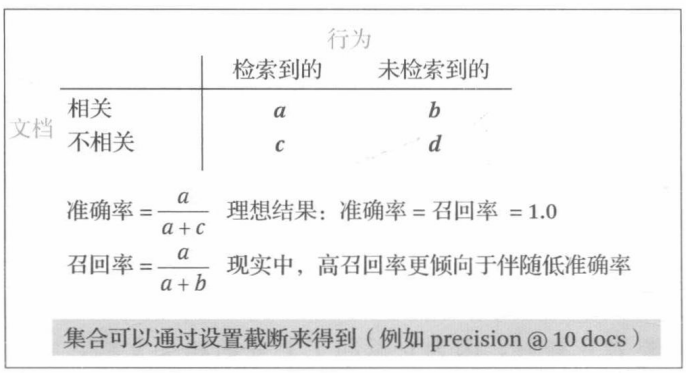


图 9-2　基本度量：准确率和召回率

既然我们关注于检索结果的文档集合，每个文档要么被检索到，要么未被检索到。根据用户是否认为这是一个有用的文档，文档可以是相关的或不相关的。因而，现在有 4 个类型的文档，可以统计每种类型文档的数量。用 a 表示被检索到且相关的文档数，b 为未检索到但相关的文档数，c 为检索到但不相关的文档数，d 为未被检索到且也不相关的文档数。在这个表格中，将检索到且相关的文档数目 a 与检索到的文档总数 $a+c$ 的比率定义为准确率：$\frac{a}{a+c}$。这里，分母就是所有检索到的文档数。召回率是通过 a 除以 a 和 b 之和来定义的，其中 $a+b$ 是相关文档的总数。准确率和召回率都重在考察检索到的相关文档的数量，但两项度量公式的分母有所不同。

那么什么才是一个理想的结果？如果准确率和召回率都是 1.0，那就意味着返回的所有结果都是相关的，而且没有错过任何相关的文档，没有返回任何一个不相关的文档。但实际上，高召回率倾向于伴随低准确率；为了尽可能多地获取相关的文档，往往会包含许多不相关的文档，从而降低了准确率。通常我们对排序在前 10 的文档的准确率更有兴趣，即观察前 10 名的结果，看其中有多少文档是真正相关的。这是一个非常有意义的度量，因为它告诉我们在搜索结果的第一页上(通常有 10 个结果)，用户可以预期看到多少相关文档。

在下一节中，将看到如何将准确率和召回率结合成一个分数来综合描述两个度量。

9.2.2　F 度量：准确率和召回率的结合

准确率和召回率之间往往存在相互妥协，所以将它们结合起来是很自然的选择。通常使用的一个被称为F_β的度量，如图 9-3 所示。在 $\beta=1$ 的情况下，它是准确率和召回率的调和平均值(harmonic mean)。

$$F_\beta = \frac{1}{\frac{\beta^2}{\beta^2+1}\frac{1}{R}+\frac{1}{\beta^2+1}\frac{1}{P}} = \frac{(\beta^2+1)P*R}{\beta^2 P+R}$$

$$F_1 = \frac{2PR}{P+R}$$

其中P = 准确率，R = 召回率，β = 参数（通常设置为1）

图 9-3　F 度量

经过一些简化，F 度量可以写成图中右侧的形式。通常情况下，β 设置为 1，表示对准确率和召回率的偏好相同。$\beta=1$ 的情况对应 F 度量的一个特例，通常称为 F_1。这是一个常用的度量方式，经常用来将准确率和召回率结合起来。如果 β 不等于 1，它就控制准确率和召回率之间的侧重点。很容易看出，如果有很大的准确率或召回率，那么 F_1 的度量值就会很高。但有趣的地方是 F_1 值如何控制准确率和召回率之间的平衡。

171
⌇
172

为了理解这个公式，可以先问一个很自然能想到的问题：为什么不用简单的算术平均值来结合它们呢？这可能是结合它们的最自然的方式。为什么这不如 F_1？即算术平均值有什么问题？

算术平均值趋于以大值为主。这意味着如果你有很高的 P 值或者很高的 R 值，总和就很高，不管另一个值是否低。然而这并不是期望的效果，因为通过返回所有的文档很容易得到完美的召回率！当我们有一个完美的召回率和低准确率的时候，仍然会得到一个相当高的均值。这样的搜索结果对于用户显然不是很有用，然而，用算术平均得到的平均值会相对较高。相比之下，F_1 值将奖励准确率和召回率大致相似的情况。所以，对只有两者之一较高的情况会进行惩罚。因此，F_1 与算术平均值对准确率和召回率进行平衡的时候采取的策略是不同的。这个例子展示了一个非常重要的方法学启示：当试图解决一个问

题的时候，可能自然会想到一个解决方案(例如算术平均值)，但重要的是不要完全局限在这个方案。相反，想想是否有其他的方法来处理它。一旦你有了多个想法，重要的是要分析它们之间的差异，然后在真实场景中思考哪一个更有意义。

总结一下，我们已经讨论了准确率，它回答了“检索结果是否都是相关的”这个问题。我们还谈到召回率，它回答了“所有的相关文档是否都被检索到”这个问题。它们是信息检索评价的两个基本度量，也可用于许多其他任务。我们谈到 F 度量作为一种结合准确率和召回率的方式，也谈到了准确率和召回率之间的平衡，这往往取决于用户的搜索任
173 务和偏好。

9.3　有序列表的评价

在前一节中，我们只考虑了相关文档是否出现在结果中，这是一种二元度量。在本节中，我们将看到在分配评价分数时如何考虑每个文档的位置。

我们已经知道，准确率和召回率是定量衡量搜索结果性能的两个基本方法。但是，正如在第 5 章深入讨论的那样，文本检索问题是一个排序问题，而不是一个分类问题。因此，需要评价有序的文档列表的质量，而不应该忽略相关文档的位置仅考虑它们是否被返回。

如何使用准确率和召回率来评价有序列表？由于相关文档的排序列表是由用户停止浏览的位置决定的，我们将不得不在不同的截止点查看准确率和召回率。假定用户按顺序浏览结果列表，则用户将在某个时刻停下来。这一点将决定集合的大小。因此，当计算准确率-召回率(precision-recall)时，必须考虑这个最重要的截止点。

由于不知道用户到底会停止在哪里，我们必须考虑所有可能停止的位置。如图 9-4 所示，准确率-召回率曲线正是用来描述这一情况。

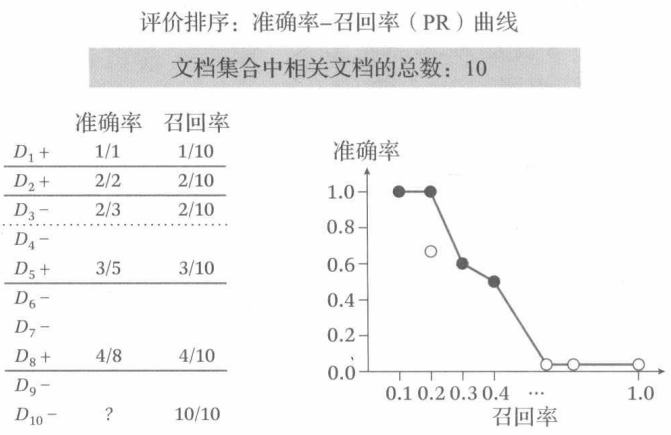


图 9-4　准确率－召回率曲线的计算

174 如果用户在第一个文档停止，在这一点上的准确率－召回率是什么？由于 D_1 是相关的，系统检索到一个文档并且相关，所以准确率是 1∶1。那召回率呢？请注意，我们假定在整个文档集合中有 10 个文档与该查询相关，所以召回率是 1/10。

如果用户在第二个文档位置停住了呢？准确率是相同的，因为 D_1 和 D_2 是相关的：100%，或者 2∶2。召回率是 2∶10，或者 20%。如果用户在第三个位置停止，因为没有获得更多的相关文档，所以召回率不会改变。然而，因为有 2/3 的相关文档，所以准确率会变低。继续下去，直到遇到另一个相关文档，召回率才会改变。本例中，这一点在 D_5。

此时，召回率已经增加到 3/10，精确度是 3/5。以此类推，我们也可以到达 D_8，得到准确率为 4/8，因为有 8 个文档，其中 4 个是相关的，而召回率是 4/10。

什么时候可以得到召回率为 5/10 呢？在这个列表中，没有这种情况。为了方便起见，我们经常假设在这种情况下准确率为零。这是一个悲观的假设，因为实际精确度会更高。我们做这个假设是为了有一个简单的方法来计算稍后将讨论的称为平均准确率(average precision)的度量。

请注意，我们做出了一些显然不准确的假设。但是，这对比较两个文本检索方法的相对优劣还是可以接受的。只要假设产生的偏差不偏向任何特定的检索方法，该度量是可以接受的，因为仍然可以准确地判断哪种方法效果更好。这是要记住的最重要的一点：当比较不同的算法时，关键是要避免任何可能导致偏向于特定方法的因素。只要做到这点，那么在保证顺序的情况下对度量进行一定变换就完全没有问题。

由于可以在不同的位置得到很多准确率－召回率的数值，我们可以绘制一条曲线；这就是图 9-4 右侧显示的内容。x 轴显示召回率的取值，y 轴显示准确率的取值。绘制准确率和召回率的数字以便显示什么样的召回率可以获得一定的准确率。而且，我们可以把这些点连接起来形成一条曲线。正如图中所示，我们假定在高召回率点上对应的准确率的值都是零。虽然真正的曲线不会像这样，但是在比较两种方法时，在这种情况下是否得到精确的准确率值影响不大。

在图 9-5 中，通过在同一个图上绘制 PR 曲线来比较两个系统。哪一个更好？在左边，系统 A 显然更好，因为对于相同的召回率水平，系统 A 的准确率值比系统 B 更好。一般来说，曲线越高越好。问题是我们可能会看到像右图那样的情况——这实际上经常发生在两条曲线交叉的地方。在这种情况下，哪一个更好？这是一个不得不面对的实际问题。假设你建立了一个搜索引擎，并且你有一个旧的算法，这里显示为系统 B。然后，你提出了一个新的想法作为系统 A。 175

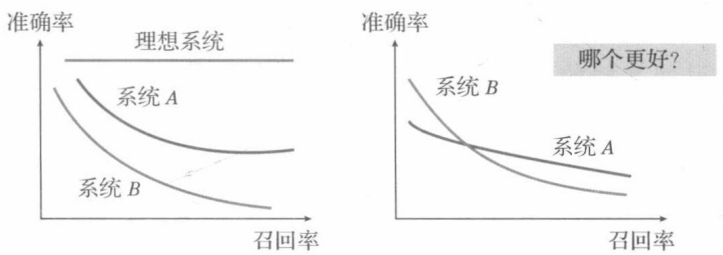


图 9-5　两个准确率－召回率曲线的比较(Marti Hearst 授权)

问题是，新方法比旧方法更好吗？或者，更实际的是，是否需要用另一种新的算法来替换搜索引擎已经使用的算法？如果替换，那么搜索引擎的表现就像系统 A，而如果不替换，就像系统 B。有些用户可能喜欢系统 A，而其他用户可能喜欢系统 B。这里的区别是：在召回率较低的情况下，B 系统效果较好，但在召回率较高的区域，A 系统效果较好。这意味着哪个系统更好取决于用户是关心高召回率，还是低召回率高准确率。想象一下，有人正在查证今天发生的事情，并想找出与此相关的新闻。哪个系统对于这个任务来说更好？在这种情况下，系统 B 更好，因为用户不太可能检查许多结果，即用户不关心高召回率，只有前几个结果是有用的。另一方面，如果用户想要确定一个研究想法是否曾经被提出过，他们会想要强调高召回率，以便看到尽可能多的相关文档，不错过任何能找到提及该想法的文献机会。此时，这些用户会喜欢系统 A。

但是这使我们回到原来的问题：哪一个系统更好？同样，这实际上取决于用户，或者更确切地说是用户的任务。所以，没有一个能够精确描述性能的数字，必须从整个图形上
176 进行整体观察。尽管如此，用一个数字来比较系统的好处是显然的，这样可以很容易对很多不同的系统进行比较；我们需要一个数字来总结准确率－召回率的值的范围。达到这一目的的一种方法是查看曲线下面的区域面积——平均准确率。也就是我们要看看每个不同的召回率的点，并考虑准确率。

将检索到相关文档时的准确率加起来，即检索到第一个相关文档时对应的准确率加上检索到第二个相关文档时对应的准确率，以此类推。图中的例子错过了很多相关的文档，所有这些情况下，都假设它们的准确率为零。最后，把这个平均值除以 10，这是集合中相关文档的总数。注意，我们不会将这个总和除以 4(这是检索到的相关文档的数量)。除以 4 是一个常见的错误；这有利于一个倾向于返回很少文档的系统，因为此时分母非常小。在正确的公式中，分母是所有相关文档的总数。这种方式使我们能够将召回率和准确率结合起来计算准确率－召回率曲线下的面积。

在数学上，我们可以在排序列表 L 上定义平均准确率，其中 $|L|=n$，如公式(9-1)所示。

$$\text{avp}(L)=\frac{1}{|Rel|}\sum_{i=1}^{n}p(i) \tag{9-1}$$

其中 $p(i)$表示 L 中排序为 i 的文档对应的准确率，Rel 是文档集中所有相关文档的集合。如果D_i 不相关，将通过设置 $p(i)=0$ 来忽略这个位置的贡献。如果D_i 是相关的，为了得到 $p(i)$，我们将迄今为止所看到的相关文档的数量除以当前文档在列表的位置 i。如果第一个相关文档排在第二位，则 $p(2)=\frac{1}{2}$。如果第三个相关文档的排序为 7，则 $p(7)=\frac{3}{7}$。让我们用该公式计算图 9-4 展示的返回文档的平均准确率。图 9-6 展示了计算过程。

i	相关性	$p(i)$
1	+	$\frac{1}{1}=1.0$
2	+	$\frac{2}{2}=1.0$
3	–	0.0
4	–	0.0
5	+	$\frac{3}{5}=0.6$
6	–	0.0
7	–	0.0
8	+	$\frac{4}{8}=0.5$
⋮	–	0.0
sum		3.1
avp		$\frac{3.1}{10}=0.31$

图 9-6　计算检索结果有序列表的平均准确率

这一度量对相关文档的位置变动很敏感。如果把第三或第四个相关文档上移，就会增加平均值。相反，如果把任何相关文档往后移，那么平均值就会减少。因此，这是一个很好的度量，因为它对每个相关文档的排名很敏感，可以区分两个有序列表之间的细微差异，这正是我们期望的。

作为对比，如果仅看前 10 个文档的准确率，很容易看出它是 4/10。这个准确率是
177 非常有意义的，因为它告诉我们用户从他们的角度会看到什么。但是，如果仅仅用这个度量比较两个或两个以上的系统就不会那么有效了，因为准确率对于这 4 个相关文档在列表中的位置并不敏感。即使它们在前 10 名的位置上移动，前 10 个文档的准确率仍然保持不变。相比之下，平均准确率是一个更好的衡量标准，因为排序的细微差异能够影响总分。

多查询有序列表的评价

平均准确率是针对一个查询来计算的。一般来说，我们会尝试很多不同的查询，以便捕捉它们之间的差异。例如，一个系统可能针对一个查询执行得非常好，而另一个系统恰好处理得不好；仅使用这个查询就不能准确地评估每个系统的能力。使用更多的查询进行

评价则需要研究人员对所有这些查询取平均准确率。显然，可以简单地计算算术平均值。事实上，这即是所谓的算术平均准确率(Mean Average Precision，MAP)。在这种情况下，对几个查询或主题的所有平均准确率计算算术平均值。令 $\mathcal{L}=L_1$，L_2，…，L_m 是运行 m 个不同查询返回的有序检索结果列表，算术平均准确率如公式(9－2)所示。

$$\mathrm{MAP}(\mathcal{L}) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{avp}(\mathcal{L}_i) \tag{9-2}$$

178

回想一下前文在计算 F_1 时的讨论。在这种情况下，平均准确率的算术平均值可以被接受吗？之前得出结论，准确率和召回率的算术平均值不如调和平均值。这里也存在类似的情况：我们应该考虑整合平均准确率的替代方法。一种替代方案是使用几何平均(geometric mean)来合并平均准确率，称为几何平均准确率(geometric mean average precision)，简称 gMAP。在数学上将其定义为公式(9-3)。

$$\mathrm{gMAP}(\mathcal{L}) = \left(\prod_{i=1}^{m}\mathrm{avp}(\mathcal{L}_i)\right)^{\frac{1}{m}} \tag{9-3}$$

或者在对数空间中将其定义为公式(9-4)。

$$\mathrm{gMAP}(\mathcal{L}) = \exp\left\{\frac{1}{m}\sum_{i=1}^{m}\mathrm{In}\ \mathrm{avp}(\mathcal{L}_i)\right\} \tag{9-4}$$

想象一下，你又在测试一个新的算法，已经测试了多个主题(查询)，并且得到了每个主题的平均准确率。如果希望了解系统的整体表现，你会使用哪种评价策略？你能想到在哪些情境下，采用不同的策略会导致不同的比较结果？即是否存在这样的情形，尽管使用相同的度量，但两种策略得到了不同的排序结果。与关于 F_1 的论点类似，算术平均值中的求和主要由较大值决定。在这里，一个较大的值意味着查询相对容易。另一方面，gMAP 往往会受到较低值的影响——那些性能不佳的查询(平均准确率较低)。如果你想改善搜索引擎的困难查询，那么 gMAP 将是首选。如果你只是想改善所有查询，那么 MAP 可能是首选。所以，答案取决于用户任务和偏好，取决于哪种衡量标准最可能代表用户的需求。

作为算术平均准确率一个特例，我们考虑整个文档集合中只有一个相关文档的情况。这实际上经常发生，例如，在所谓的已知项目搜索(known item search)中，搜索的目标就是某个已知的亚马逊或 Facebook 上的页面；或者在问答系统的应用中，答案只有一个。在这种情况下，系统的目标是使一个特定的答案排在搜索结果的最顶端。在这种情况下，平均准确率将转变为排序倒数(reciprocal rank)，也就是 $1/r$，其中 r 是单个相关文档的位置(排序)。如果这个文档排在最前面，那么排序倒数将是 $1/1=1$；如果排在第二位，那么排序倒数是 $1/2$，以此类推。更进一步，可以取多个主题的所有倒数排序值的平均值来得到整体评价，称之为排序倒数均值(Mean Reciprocal Rank，MRR)。这是一个非常流行的度量，尤其适合于评价只有一个标准答案的问题。

179

这里的 r 是相当有意义的，它表示用户为了找到那个相关文档需要付出多少努力。如果它排在最前面，那么消耗就很低。如果它排在 100 位，那么用户必须(大概)筛选 100 个文档才能找到它。因此，r 也是一个有意义的度量，但是排序倒数使用 r 的倒数，而不是直接使用 r 本身。

那么为什么不是简单地使用 r？在只有一个相关项目的场景下，或许会考虑直接使用 r 作为度量来衡量系统的性能。毕竟，这是衡量用户的努力。但是，如果对大量主题作平均的时候，情况会有所不同。对于单个主题，使用 r 或使用 $1/r$ 不会有任何区别，因为一

个较大的 r 对应一个小的 $1/r$。当有很多主题时，差异就会显现。就像 MAP 一样，总和将被 r 的大值所支配。什么情况下 r 会很大呢？r 很大意味着相关文档在排序列表中的排名很低。因而，平均值将被排在列表较低部分的相关文档所支配。然而，从用户的角度来看，我们更关心排名高的文档，所以通过使用排序倒数来进行这种转换，我们更加关注排序列表顶端的差异。例如，将相关项目排在第一与排在第二时对应的排序倒数的差距是 1-1/2，远高于将相关文档排在第 100 位与 1000 位时对应排序倒数的差距 1/100-1/1000。

小结一下，本节展示了准确率－召回率曲线可以体现有序列表的整体表现，强调了有序列表的实际效用取决于用户愿意检查多少排序靠前的结果，有些用户会比其他用户查看更多。平均准确率是比较两种排序方法的标准度量，它结合了准确率和召回率，同时对每个相关文档的排名都很敏感。最后还介绍了三种用于总结多个平均准确率的方法：MAP、gMAP 和 MRR。

9.4　基于多级别判断标准的评价

本节将解释当相关性判断可以分为多个级别的时候，如何评价文本检索系统。到目前为
180 止，我们已经讨论了二元判断——这意味着文档被判断为相关或不相关。之前提出过相关性是一个程度问题。通常我们能够将高度相关的文档与有用但相关度低一些的文档相区分。

图 9-7 展示了三个相关性级别的示例：级别 3 表示高度相关，级别 2 表示稍微相关，级别 1 表示不相关。如何使用这些判断来评价一个新的系统？我们无法使用平均准确率，因为它只能用于二元相关的情形；如果把 2 级和 3 级看作一个级别，那么就失去了在这两个级别的比较中所带来的额外信息。MAP、gMAP 和 MRR 依赖于平均准确率，因而也无法使用它们。

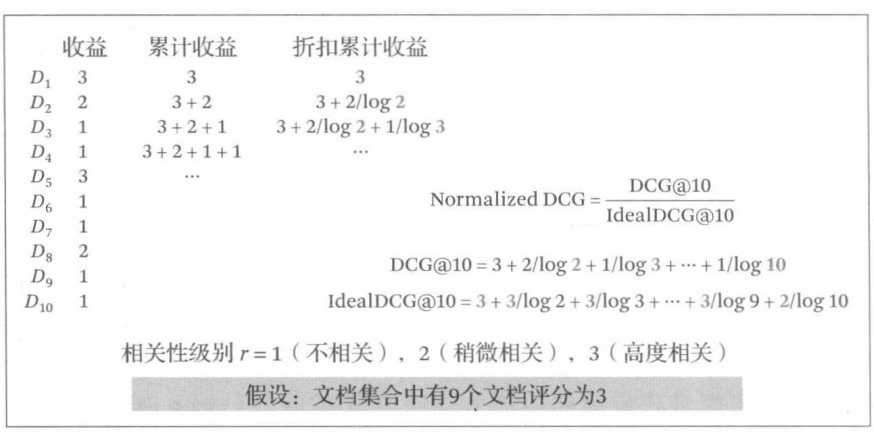


图 9-7　NDCG 的计算

让我们来观察一下使用三级别判断时最相关的结果。假设用户最关心前 10 个结果。我们将这些多级判断称为“收益”，因为它们与用户查看文档时获得的信息的多少相对应。观察第一个文档，用户可以获得三分；而不相关的文档，用户只会得到一分。从用户的角度来看，这些收益通常是与文档的效用相匹配的。如果假设用户在第十个文档处停下来，我们可以遍历返回的文档列表来累加用户获得的信息收益。设 r_i 是第 i 个结果的收益，i 取值范围从 1 到 n，本例将 n 设置为 10。定义累计收益(Cumulative Gain，CG)为

181
$$\mathrm{CG}(L) = \sum_{i=1}^{n} r_i \tag{9-5}$$

如果用户浏览的文档多，那么累计收益也会更多，代价是需要花费更多的时间来检验列表。因此，累计收益反映了一些信息，即如果用户检查所有这些文档，用户将获得多少总收益。

累计收益的一个缺陷是没有考虑每个文档的排序或位置。前 4 个文档中只有一个高度相关的文档、一个稍微相关的文档和两个不相关的文档；它们的累计收益 CG 不能反映出它们在列表中的位置。理想情况下，我们希望获得三个收益的那些文档排在最前面。但是如何表示这个直觉呢？第二个收益为 3 的文档对系统评分的贡献显然不及第一个收益为 3 排在首位的文档。这意味着，来自不同文档的收益必须根据它们的位置进行加权。位置为 1 的文档不需要打折，因为可以假设用户总能够看到这个文档。但是第二个文档会稍微打折一些，因为用户仍会有一定概率没有看到这个文档。因而，把这个收益除以一个基于位置的权重来体现基于位置的惩罚。折扣累计收益(Discounted Cumulative Gain，DCG)正是为了实现这个目的：

$$\mathrm{DCG}(L) = r_1 + \sum_{i=2}^{n} \frac{r_i}{\log_2 i} \tag{9-6}$$

每个文档的收益都除以其在列表中的位置的对数。因此，排名不高的文档将不会获得与排名高的文档相同的收益。例如，如果调换 D_5 和 D_2 的位置，则总体 DCG 得分会增加，因为 D_5 的相关性得分更高，将其置于接近顶部的位置，得分的折扣就会更少。

现在我们有了折扣累计收益来衡量基于多级别判断的有序列表的效用，但仍然需要做一些工作来使这个度量对不同查询的评价具有可比性。这里的想法是标准折扣累计收益(Normalized Discounted Cumulative Gain，NDCG)：

$$\mathrm{NDCG}(L) = \frac{\mathrm{DCG}(L)}{\mathrm{IDCG}} \tag{9-7}$$

对于特定的查询，NDCG 只是使用理想的 DCG(Ideal DCG，IDCG)对 DCG 进行规范化后得到的结果。IDCG 是一个理想有序列表的 DCG，其中最相关的文档排在最前面，按照相关性降序排列。例如，想象一下在整个文档集合中，有 9 个可以得 3 分。那么理想的有序列表将会把这 9 个文档放在最前面。这些所有的得 3 分的文档之后将会是得 2 分的文档，因为它们是 3 分文档之后最好的。然后，可以计算这个理想有序列表的 DCG，这成为 NDCG 的分母，以便使自己的 DCG 规范化后的值在[0，1]之间。因此，NDCG 本质是将 182
实际的 DCG 和当前查询有可能获得的最佳结果相比较。如果只针对一个主题，NDCG 不会影响系统之间的比较，因为 IDCG 对于所有的系统是相同的。不同的是当我们需要处理多个主题时——如果不进行规范化，不同的主题得到的 DCG 的值的范围可能会显著不同。针对本例中的查询，我们有 9 个高度相关的文档，但当然并不总是如此。

因此，NDCG 是一种基于多个相关性级别来量化相关性的度量。从更一般的角度来看，它是可以被应用到具有多级别判断的任何排序任务的基本度量。判断级别的数目取决于手头的应用。NDCG 的主要思想是概括前 k 个文档的总效用；总是选择一个截断点，然后测量截断点之前的总效用。它降低了排名较低的文档的贡献，最后执行规范化以确保跨查询的可比性。

9.5　评价中的实际问题

为了创建测试数据集，我们必须创建一组查询、一组文档和一组相关性判断。事实证明，每个要求都有其自身的挑战。

首先，文档和查询必须具有代表性。它们必须代表真正的查询以及用户与之交互的真

实文档，还必须使用很多查询和很多文档来避免有偏见的结论。为了评价高召回率的检索任务，必须确保每个查询都存在许多相关文档。如果一个查询在文档集合中只有一个相关文档，那么使用这样的查询来比较不同的方法就不是很适用，因为没有太多空间可以看到差异。

在相关性判断方面，面临的挑战是既要确保对所有查询的所有文档都进行完整的判断，又要最大限度地减少人力成本。由于必须用人力来标记这些文档，所以这是一项非常耗费人力的任务。因此，通常不可能针对所有查询标记所有可能的文档，尤其是考虑到像Web这样规模巨大的数据集。

将评价度量与用户的感知效用相关联也是一个挑战。我们必须仔细考虑用户关心的内容，然后设计一些度量来体现他们的偏好。

183 在一定的概率下，可以用数学方法来量化两个系统的评价分数是否确实不同。这便是统计显著性检验。显著性检验能够让我们了解到有多大的可能性评价结果的差异是由于随机因素导致的。这就是为什么要使用很多查询的原因：我们拥有的数据越多，对评价就越有把握。

图 9-8 展示了在两个不同的实验中系统 A 和系统 B 在一些样本上的平均准确率。从图的底部可以看到，在每个实验中都有每个系统的 MAP，而它们碰巧在实验Ⅰ和实验Ⅱ中是一样的。然而，如果仔细查看不同查询的平均准确率，你将会意识到，在某些情况下，你可能会觉得最终的平均结论是可信的；在其他某些情况下，则相反。仅基于 MAP 分数，我们可以很容易地说系统 B 更好，毕竟 0.4 是 0.2 的两倍。但是，如果仔细观察这两个实验，看看具体的结果，你会发现我们会更有把握地说，在实验Ⅰ中系统 B 实际上更好，因为平均准确率总是比系统 A 好。在实验Ⅱ中，我们不确定系统 B 是否更好，因为分数波动很大。

查询	实验Ⅰ 系统A	系统B	查询	实验Ⅱ 系统A	系统B
1	0.20	0.40	1	0.02	0.76
2	0.21	0.41	2	0.39	0.07
3	0.22	0.42	3	0.16	0.37
4	0.19	0.39	5	0.58	0.21
5	0.17	0.37	6	0.04	0.02
6	0.20	0.40	6	0.09	0.91
7	0.21	0.41	7	0.12	0.46
平均	0.20	0.40	平均	0.20	0.40

图 9-8　统计显著性：两组实验具有相同的 MAP 值(Douglas W. Oard 和 Philip Resnik 授权)

如何量化地回答这个问题？我们需要做一个统计显著性检验。检验背后的想法是估计不同查询的平均准确率(或任何其他分数)的方差。如果方差很大，说明结果可能会根据不
184 同的查询而波动，从而导致结果不可靠。

我们再看一下实验Ⅱ中的结果。图 9-9 展示了两种不同的比较方法。一个是符号测试(sign test)。如果系统 B 比系统 A 好，那么用一个加号来表示。如果系统 A 更好，用一个负号来表示。用这种方式，在 4 个查询下系统 B 更好，3 个查询下系统 A 更好。直觉上，这些结果看起来是随机的。如果抛掷 7 枚硬币，用加号表示正面，用减号表示背面，仅仅

通过简单地翻转 7 枚硬币就能得到这些结果。事实上，较大的平均数并不能说明什么！这个直觉可以用 p 值的概念来量化。p 值表示这个结果实际上是随机波动的概率。在本例中，概率是 1，这意味着该结果肯定是一个随机波动。

查询	系统 A	系统 B	符号测试	Willcoxon
1	0.02	0.76	+	+0.74
2	0.39	0.07	−	−0.32
3	0.16	0.37	+	+0.21
4	0.58	0.21	−	−0.37
5	0.04	0.02	−	−0.02
6	0.09	0.91	+	+0.82
7	0.12	0.46	+	+0.34
平均	0.20	0.40	p=1.0	p=0.9375

图 9-9　统计显著性检验(Douglas W. Oard 和 Philip Resnik 授权)

可以使用许多不同的显著性检验来量化观察到的结果差异是由于随机波动导致的可能性。其中特别有趣的一个是 Wilcoxon signed-rank 检验(Wilcoxon signed-rank test)。它是一个非参数检验，不仅要看符号，还要考虑分数差异的大小。另一种是假定正态分布的(参数)t 检验。无论如何，针对我们的例子都可以得出一个类似的结论：结果很可能是随机的。为了进一步研究这个以及其他统计显著性检验，建议读者从 Smucker 等[2007]起步。

为了说明 p 值的概念，考虑图 9-10 中的分布。这是一个正态分布，中心均值为零，即我们先假设两个系统之间没有差别。但是，由于假设随机波动取决于查询，因而我们可能会观察到差异。因此，实际的差异可能会偏左或偏右。这条曲线显示了当从系统 B 的 MAP 中减去系统 A 的 MAP(甚至反之亦然)时，我们将观察到偏离零的值的概率。根据图片，如果在图中的点的位置观察到差异，那么这事实上是随机观察的概率非常高。我们可以定义由于随机波动而观测到的区域，通常使用所有输出的 95%的值。在这个区间内，观察到的值有 95%的机会来自随机波动。如果观察到侧面尾部的值，那么差异不大可能来自随机波动(可能性只有 5%)。这个 95%的值决定了绘制在 x 轴上的线的位置。如果我们仅相信有 1%的机会是由于随机波动造成的，那么垂直线就会重新拉远，绘制线条的确切值 x 取决于所使用的具体显著性检验。

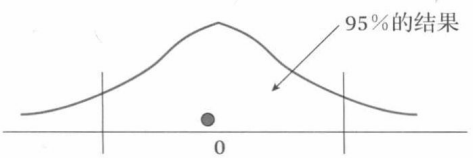


图 9-10　零检验(Douglas W. Oard 和 Philip Resnik 授权)

185

这里传达的信息是，需要使用许多查询来避免武断地认定一个系统比另一个更好而得到错误的结论。有很多不同方法能够进行统计显著性检验，它们本质上决定了将随机机会与系统中实际差异之间的边界放在哪里。

下面讨论进行相关性判断的问题。如前所述，除非是非常小的数据集，否则完全判断所有文档是非常困难的。问题是，如果无法评判集合中的所有文档，我们应该判断哪一个子集？这里的解决方案是池化(pooling)。这是一个在许多情况下都可用来解决这个问题的策略。首先，选择一组不同的排序方法，对应着不同类型的检索系统。我们希望这些方法可以帮助我们提出可能的相关文档。目标是挑选出相关文档，以便用户对其进行判断。于

是，可以让每个系统根据其排序函数返回前 k 个文档。系统之间的 k 值可能会有所不同，但重点是要求它们建议最有可能的相关文档。然后，简单地将所有这些前 k 个文档对应的
186 集合合并，形成一个供评审人员评判的文档池。当然，其中会有很多重复的文档，因为不同系统可能检索到相同的文档。当然也存在一些文档只能由某个系统返回。因而，采用多样化的排序方法的动机即希望确保池中文档的广泛性。我们可以加入尽可能多的随机文档。然后，评审人员将对这个文档池中的文档进行完全的判断。余下的未被判断文档被认为是不相关的，标注人员不需要花费时间和精力来手动判断它们。如果池足够大，这个假设是足够好的。这意味着如果你的系统参与了对文档池的构建，那么它就不太可能会受到惩罚，因为排序高的文档已经全部接受了人工判断。然而，这对于评价一个可能没有对文档池构建做出贡献的新系统来说是有问题的，因为它所返回的文档可能未经判断就被认为是不相关的。

我们尚未涉及一些其他评价策略，如 A-B 测试；评价系统将两种结果随机混合，并将混合的结果展示给用户。当然，用户不会看到哪个结果来自哪个方法，所以用户会在搜索引擎应用中判断这些结果或点击文档。这样，系统可以跟踪被点击的文档，查看是否有一种方法对被点击的文档做出了更多的贡献。如果用户倾向于点击某个方法的结果之一，那么该方法可能会更好。A-B 测试也可以用来比较两个不同的检索界面。

由于文本检索任务是一种经验性的定义，文本检索方法的评价在该任务中非常重要。如果不依赖用户，我们就无法判断一种方法是否更好。如果有不恰当的实验设计，可能会误导我们的研究或应用，得出错误的结论。对于各种经验性评价任务，其主要策略均为 Cranfield 评价方法。MAP 和 NDCG 是必须知道的两个主要方法，在研究论文中经常可以看到这些方法。最后，从用户的角度来看，最多检索 10 个文档(或一些小数字)更容易解释，因为这是他们在真实应用场景中可能看到的文档的数目。

书目说明和延伸阅读

评价一直是信息检索领域以及一般的经验性 AI 问题中一个重要的研究问题。Cranfield 评价方法论是由信息检索的早期先驱者在 20 世纪 60 年代建立起来的；关于这个话题的重要
187 的早期论文可以在 Sparck Jones 和 Willett[1997]中找到。Harman[2011]的《Information Retrieval Evaluation》一书从历史的角度介绍了信息检索评价方法论以及 TREC 等评价活动的发展。Sanderson 关于测试集评价的书[Sanderson 2010]是另一个非常有用的评价方面研究工作的综述。Kelly 的关于交互式信息检索评价的书是介绍通过用户调查进行交互式信息检索评价的另一优秀著作[Kelly 2009]。

练习

9.1　在现实中，高召回率往往和低准确率联系在一起，为什么？

9.2　以下度量的取值范围(最低和最高分数)是多少？

(a) 准确率

(b) 召回率

(c) F_1 值

(d) 平均准确率

(e) 算术平均准确率(MAP)

(f) 几何平均准确率(gMAP)

(g) 平均排序倒数(MRR)

(h) 折扣累计收益(DCG)

(i) 规范化的折扣累计收益(NDCG)

9.3 假设集合中共有 16 个相关文档。考虑到如下结果，加号表示相关，减号表示不相关：

$$\{+, +, -, +, +, -, -, +, -, -\}$$

计算排名列表中如下排名指标。

(a) 准确率

(b) 召回率

(c) F_1 值

(d) 平均准确率

9.4 考虑召回率和准确率如何相互作用。

(a) 对于同一个有序列表，5 个文档(P@5)的准确率是否可以低于 P@10？ 188

(b) 在 5 个文档(R@5)中，对于同一个有序列表，是否会比 R@10 低？

(c) 假设有 100 个相关文档。对于相同的有序列表，R@5 可以高于 P@5 吗？

(d) 假设只有一个相关文档。对于相同的有序列表，R@5 可以高于 P@5 吗？

9.5 在使用 NDCG 评价指标时，研究人员有时会考虑 NDCG@k。这是搜索引擎返回的前 k 个文档的 NDCG 得分。在计算理想的 DCG 时，我们应该只考虑 k 个结果，还是应该考虑所有相关的文档？

9.6 盈亏平衡点准确率是有序列表中准确率和召回率相等时的准确率数值。那么这个值可以用来比较两个或多个检索系统吗？这个测量与其他单点测量如 F_1 和平均准确率相比如何？

9.7 研究人员希望证明他的方法比现有的方法更好。他使用统计显著性检验，结果发现 95%的置信度是随机的(即或许不能说他的方法更好)。如果他把置信度改为 90%，他可以证明他的方法比使用相同显著性检验的基线系统要好。这有什么问题吗？

9.8 词干提取(stemming)如何影响检索性能？设计一个实验来找到答案。你的数据集是什么？你需要什么其他的信息来执行这个测试？你怎样才能量化结果？

9.9 使用 MeTA 执行上述的词干提取实验。在 MeTA 网站上可以找到一个有相关性判断的语料库。

9.10 找到最佳的一元词项分词方法进行检索。例如，考虑小写字母、只保留字母字符、词干提取、删除停用词或以上的组合。通过使用信息检索技术来量化你的结果。

9.11 找到一个公开可用的数据集，并使用 MeTA 格式(网站上有描述)创建自己的查询和相关性判断。 189 ~ 190

9.12 修改 MeTA 的 **index::ir_eval** 类以添加对平均排序倒数(MRR)评价的支持。

第 10 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

网络搜索

本章讨论文本检索最重要的应用之一：网络搜索引擎。尽管在网络诞生之前，很多信息检索算法已经出现，互联网的出现为这些信息检索算法提供了更多的机遇。经典搜索算法必须进一步扩展才能完全解决在网络搜索中遇到的新挑战。

首先网络搜索在可扩展性上面临挑战。我们如何处理网络的大小，确保对所有信息(无论是否为文本信息)的完整覆盖？我们如何通过回答所有查询来快速为多个用户提供服务？在网络诞生之前，搜索的规模相对较小，搜索场景通常集中在图书馆，所以这些问题不严重。

第二个挑战是网络上有大量低质量的信息，它们被称为垃圾信息。*搜索引擎优化*是试图利用网页得分的方式提高一个特定页面的排名，例如添加许多与实际内容不相关的单词，或者添加许多假链接令网页看起来比实际上更受欢迎。许多方法被用于检测和防止这种垃圾信息的欺诈行为[Spirin 和 Han 2012]。

第三个挑战是网络的动态性。新的页面不断被创建并被快速更新。这使得索引始终指向最新的内容变得很困难。这些只是我们建立一个高品质的网络搜索引擎所必须面对的一些挑战。除了这些挑战，还有一些环节可以加以利用来改善搜索结果。例如，可以通过在页面之间加入链接对网页评分进行改善。

一些算法如向量空间模型等对众多搜索应用来说是通用的。另一方面，它们也没有利用特定应用程序(如网页搜索)中页面或文档的特殊特性。基于这些挑战和机遇，出现了一些专门为网络搜索开发的新技术。一种这样的技术是并行索引搜索。它解决了可扩展性的问题，其中尤其著名的是谷歌公司的 MapReduce 框架。

还有一些技术已经被开发来解决垃圾信息问题。我们必须防止这些垃圾页面排在前面。还有一些技术可以根据搜索引擎优化器来实现强大的排名。我们将使用各种各样的特征对网页排名，很难靠一种特殊伎俩来迷惑搜索引擎。

第三个技术是链接分析，这个技术通过利用关于网络链接的额外信息来改善搜索结果。当然，除了链接分析我们还能使用多种特征进行排名，例如网页布局或者描述指向另一个网页链接的锚文本等特征。

网页搜索引擎的第一个组件是*爬虫*。这是一个用于下载我们希望搜索的网页内容的程序。第二个组件是索引器，它将收集这些下载的网页，并创建一个倒排索引。第三个组件是检索，通过浏览器对用户的查询做出回应。浏览器将显示搜索结果并允许用户与网络进行交互。网络搜索引擎与用户的交互信息为反馈(在第 7 章讨论)和评价(在第 9 章讨论)提供了数据。我们将在下一节讨论爬虫。除了信息爬取的细节外，我们已经在第 8 章描述了所有索引步骤。

在讨论过爬取之后，我们转向网络索引的特定挑战。然后，我们讨论如何利用页面之间的链接进行分析。我们讨论的最后一个技术是*排名学习*，这是一种结合多种不同特征进行排名的方法。

10.1　网络爬虫

爬虫还被称为蜘蛛或软件机器人，它从网络上爬取(遍历、解析和下载)网页。建立一个简易爬虫是相对容易的，只需要从一组种子页面开始，从网络获取页面并解析这些页面的新链接。然后，我们将它们添加到一个队列，然后通过广度优先搜索继续探索这些页面链接直到我们满意。

构建一个真正的爬虫是相当棘手的，有一些复杂的问题我们不可避免地要处理。一个问题是健壮性：如果服务器没有回应或返回无法解析的垃圾网页，该如何处理？如果有一 192
个陷阱会持续产生动态网页，吸引爬取工具循环爬取同一个网站，该如何处理？还有一个问题是，我们不想因太多的爬取请求令一个特定的服务器超载。这些可能会导致网站拒绝服务，一些网站也会阻止一些 IP 地址的访问，它们认为来自这些 IP 地址的爬取或请求过多。同样爬虫应该尊重机器人排除协议。位于网站根目录的名为 `robots.txt` 的文件告诉爬虫哪个路径不能爬取。爬虫也需要处理不同类型的文件，如图像、PDF 或任何其他格式的网页，它们都包含搜索应用程序的有用信息。理想情况下，爬虫应该能够识别冗余的网页，这样它就不会重复自己或者陷入循环。最后，发现隐藏的 URL 是有用的，这些网址可能不是从任何页面链接，但仍包含我们想索引的内容。

那么，主要的爬取策略是什么？一般来说，广度优先搜索是最常见的，因为它能够自然地平衡服务器负载。并行爬取也是很常见的选择，因为这类任务很容易并行化。一个有趣的变种称为聚焦爬取(focused crawling)。在这里，我们将爬取一些关于特定的主题的网页，例如关于汽车的所有页面。这通常会以查询开始，从而获得一些结果。进而爬取更多的网页内容。甚至更多聚焦爬取的极端例子是在一个特定的论坛上下载和索引所有论坛帖子。在这种情况下，我们可能有一个 URL，如 `http://www.text-data-book-forum.com/boards?id=3`，这是指论坛上的第三个帖子。通过改变 `id` 参数，我们可以遍历所有的论坛帖子，并很容易地索引它们。在这种情况下，应在请求之间添加一个延迟，这样服务器就不会不堪重负。

爬取的另一个挑战是找到上一次爬取后新创建的网页，如果这些新网页未被已知网页链接，爬取它们非常具有挑战性。如果这些新创建的网页已被已知网页链接，则可以通过已知网页中的链接找到它们。

最后，我们可能会面临增量爬取或重复爬取的情况。假设你希望能够创建一个 Web 搜索引擎。显然，你第一次爬取来自网络的数据。将来只需要爬取更新的页面。这是一个非常有趣的研究问题：我们如何确定页面何时需要被重新爬取(甚至在新页面已被创建时)？主要有两个要考虑的因素。首先是一个特定的页面是否经常更新。如果该页面是一 193
个静态的页面，几个月没有改变，可能没有必要每天重新爬取它，因为它不太可能经常改变。另一方面，如果体育比分页面更新频繁，则可能同一天需要多次爬取。第二个要考虑的因素是搜索引擎系统的用户访问特定页面的频率。如果这是一个使用频率高的页面，那就要确保它是实时更新的。与它相反的另一类网页是一年都未曾被任何用户访问的网页，即使这类网页已经改变了很多，可能没有必要爬取那个页面，至少不必时时更新。

10.2　网页索引

在本节中，我们将讨论如何创建一个网络量级的索引。通过爬取工具获得千兆字节或兆兆字节的数据后，下一步是使用索引器创建倒排索引。一般来说，我们可以使用标准的

信息检索技术创建索引，但是我们必须解决整个网络量级的索引数据规模所带来的挑战。所面临的两大主要挑战是可扩展性和效率问题。

索引的数据规模将会非常大，无法存储于单独一台机器或单个磁盘，我们必须将数据存储在多台机器上。另外，由于数据是如此之大，所以我们可以并行处理数据从而快速生成索引。为了应对这些挑战，谷歌已经提出了一些创新性的解决方案。一个是谷歌文件系统(Google File System，GFS)，它是一个通用的分布式文件系统，可以帮助程序员管理存储在机器集群上的文件。第二个是 MapReduce，这是一个支持并行计算的通用软件框架。Hadoop 是最著名的 MapReduce 的开源实现，现在被许多应用程序所采用。

图 10-1 显示了谷歌文件系统的架构。它使用非常简单的集中管理机制来管理所有具体文件的位置信息。也就是说，它维护一个文件名称空间和查找表，以确切地知道每个文件实际存储在哪里。应用程序客户端与 GFS 主节点进行通信，获得要处理的文件的特定位置。这个文件系统存储在机器上固定大小的块中，每个数据文件分成许多 64 MB 大小的块。这些块被复制保存以确保可靠性。程序员不必担心所有这些细节，这一切通过这个文件系统自动完成。从应用程序的角度来看，程序员会看到一个普通的文件。该程序不必知道它存储在哪里，只需调用高级操作权限来处理文件。另一个特点是数据直接在应用程
194 序和块服务器之间传输，所以在使用上会感觉数据传输方面效率很高。

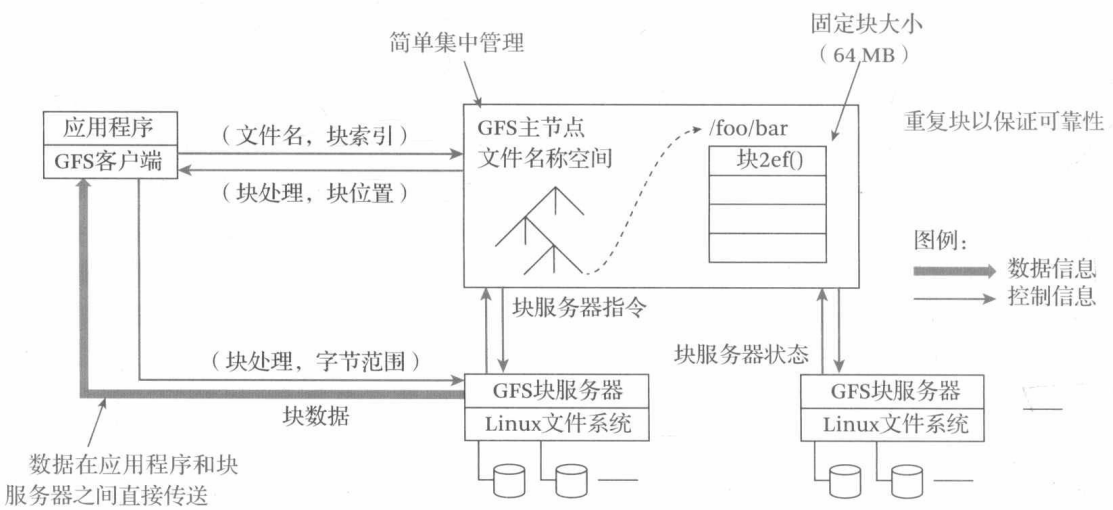


图 10-1　谷歌文件系统的流程演示(Ghemawat 等[2003])

在 GFS 之上，谷歌将 MapReduce 视作并行编程的通用框架。它支持倒排索引的创建等任务。和 GFS 一样，这个框架包揽了程序员的低层工作。从而程序员可以仅用最小的气力创建一个可以并行运行在大集群上的应用程序。隐藏在框架中的一些低层细节是通信、负载平衡、任务执行以及容错功能。其中容错功能是避免发生因一台服务器出现故障导致有些任务可能无法完成的情况。MapReduce 机制会检测出任务还没有完成，自动调度该任务到其他可以完成这项工作的服务器上。关于这一点程序员也不必担心。

在 MapReduce 中，输入数据被分成多个(键，值)对。其中值取决于数据。编程人员编写的映射函数将用于处理(键，值)对信息。然后 **map** 函数将处理这些(键，值)对并生成一些其他的(键，值)对。当然，新键通常不同于 **map** 中的旧键。一切关于 **map** 的调用的输出均根据键进行收集和排序，与同一个键相关的值将被分组在一起。

对于每个唯一的键，我们有一组值与这个键对应。这是发送到归约(**reduce**)函数的

数据。每个 **reduce** 实例将处理不同的键。这个函数将一个键与一组值作为输入进行处理，产生一个(键，值)对的集合作为输出。这是 MapReduce 的一般框架。现在程序员只需要 195
编写 **map** 函数和 **reduce** 函数，其他的工作都由 MapReduce 框架来处理。有了这样的框架，输入数据就可以被分割成多个部分，首先由 **map** 进行并行处理，并在 **reduce** 阶段重新运用并行机制进行处理。

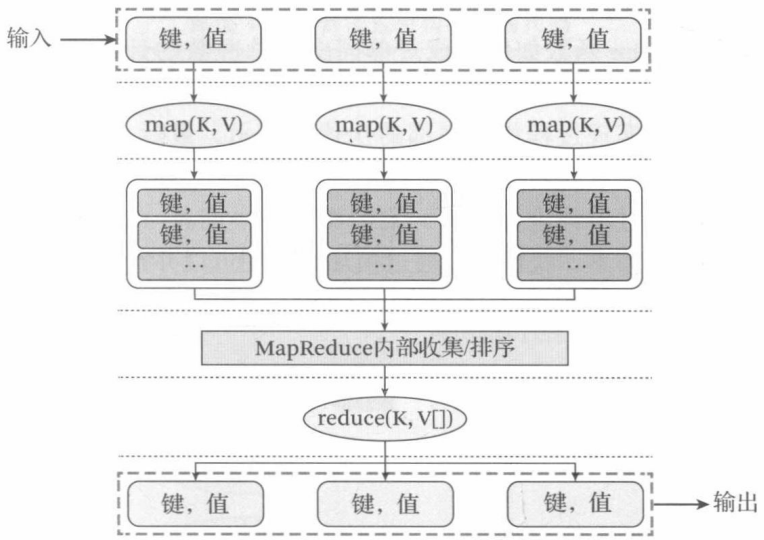


图 10-2　MapReduce 的计算流程图

图 10-3 显示了一个字数统计的例子。输入是包含分词结果的文档，输出是每个词出现的次数。这种计数将有助于评估数据集合中的流行度或为搜索任务进行 IDF 加权计算。所以我们如何解决这个问题呢？一个自然的想法是，这个任务可以通过以下方式来完成：简单地对文件的不同部分进行并行计数，再将计数所得数字组合起来。这正是我们可以运用 MapReduce 所做的事情：我们可以对输入文件中的行进行并行化处理。更具体地说，我们可以假设映射函数的每一个输入是一个(键，值)对，表示行号和这一行包含的字符串。 196

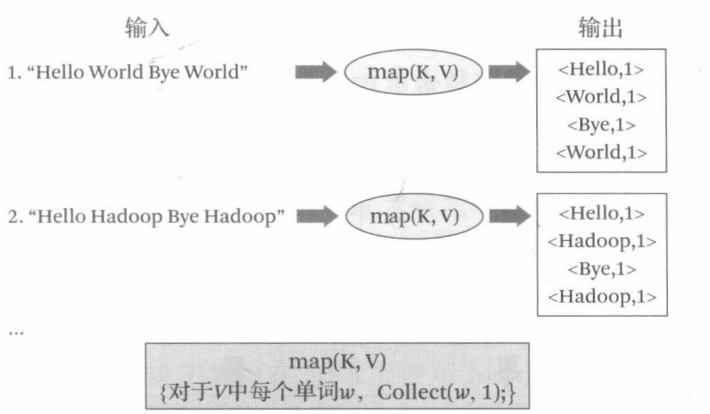


图 10-3　map 函数进行单词计数的过程

第一行是(1，Hello World Bye World)对。这个键值对将被发送到 **map** 函数来计算这一行中单词出现的次数。在这种情况下只有 4 个单词，每个单词对应一个数字 1。图底部显示了 **map** 的伪代码。它只需要对这一行中的所有单词进行一遍迭代，然后调用收集(**Collect**)函数，将单词与计数发送到收集器。然后收集器将尝试对来自不同的映射函数

的键值对进行排序。程序员指定这个函数处理每个部分的数据的方式。当然，第二行将被一个映射函数的不同实例来处理，这将产生一个类似的输出。

如上所述，收集器将执行内部分组或排序。在这个阶段，可以看到我们已经收集了多个键值对。每一对由一个单词及它在行内出现的次数构成。一旦我们获得所有键值对，就能够根据键(即单词)进行排序。现在每个单词都对应多个计数值。这些计数值表示这个词在不同行中出现的次数。这些新的(键，值)对将被输送到 **reduce** 函数。

图 10-4 显示了 **reduce** 函数如何完成对单词出现总次数的计数。该函数已经获得每个部分的计数，所以它需要做的是简单地把它们加起来。我们有一个计数器，然后遍历在这个数组中看到的所有单词，实现过程如图底部的伪代码所示。最后，我们输出键和计数总和，这正是我们想要在整个程序最后获得的输出。正如我们所见，这个过程非常类似于构建一个倒排索引，这里的输出通过单词进行索引。我们拥有一个词典，所缺少的是文档 ID 和每篇
197 独立的文档中单词出现的频率。我们可以通过稍加修改便可以并行地构建一个倒排索引。

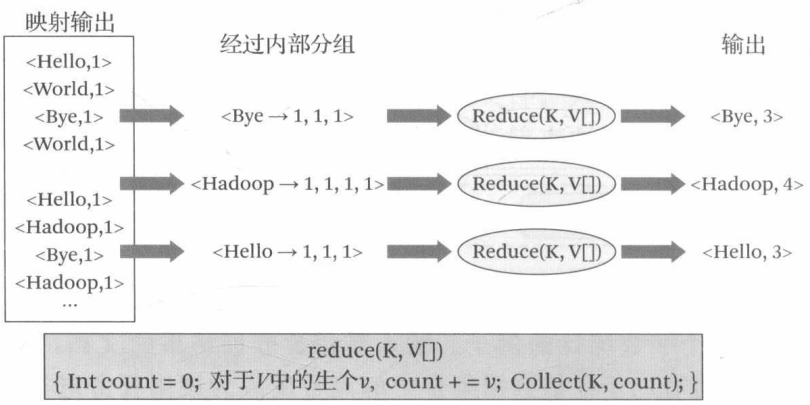


图 10-4　**reduce** 函数进行单词计数的过程

我们修改一下单词计数的例子来创建一个倒排索引，图 10-5 演示了这个例子。现在，我们假设 **map** 函数的输入是一个(键，值)对，其中键是文档 ID，值表示该文档中的所有单词构成的字符串。**map** 函数将做类似于在前面的例子中看到的工作：将单词在文档中的计数进行聚合，组合成新的键值对。在新的键值对中，每个键是一个单词，值是单词在一篇文档中的总计数值。在之后的倒排索引中，我们想保留这个文档 ID 信息，以便 **map** 函数能继续跟踪它。

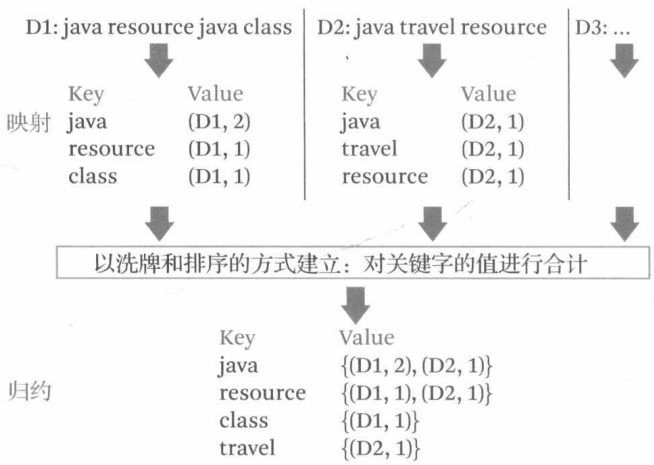


图 10-5　使用 MapReduce 创建倒排索引的过程

经过 **map** 函数，系统运用排序机制将相同的单词分到一组，并将这些数据输入 **reduce** 函数中。**reduce** 函数的输入看起来像一个倒排索引条目。这个条目由一个单词和一系列数值组成，这些数值是该单词在包含它的文档中出现的频率。我们所要做的只是将它们连接成一个连续的数据块，然后将其存储在文件系统上。**reduce** 函数做的工作很少。算法 10-1 可以用于实现倒排索引。

算法 10-1　倒排索引构建的伪代码

```
function MAP(docid n, doc d)
    H ← new ASSOCIATIVEARRAY
    for term t ∈ d do
        H[t] ← H[t] + 1
    end for
    for t ∈ H do
        EMIT(t, [n, H[t]])
    end for
end function

function REDUCE(term t, postings [(a1, f1), (a2, f2), ...])
    P ← new LIST
    for all (a, f) do
        APPEND(P, (a, f))
    end for
    SORT(P)
    EMIT(t, P)
end function
```

改编自 Lin 和 Dyer[2010]的算法 10-1 描述了 **map** 和 **reduce** 函数。程序员会指定这两个函数运行在 MapReduce 集群的上层。如前所述，**map** 函数通过一个关联数组(例如字典)统计单词的出现次数，并将所有计数连同文档 ID 一起输出。**reduce** 函数简单地连接所有的输入，每一个文档 ID 键作为一个输入条目存在。MapReduce 函数尽管简单却允许我们构造一个规模巨大的倒排索引。数据可以被不同的机器处理，程序员不会关心实现的细节。这就是我们为网络搜索并行构建索引的过程。

198 ~ 199

总而言之，网络规模索引需要一些超越传统索引技术的新技术。首先是由于我们必须在多台机器上存储索引信息，这通常通过类似 GFS 的分布式文件系统来完成。其次，由于索引的规模巨大，需要并行创建。这个是通过使用 MapReduce 框架来完成的。要注意的是 GFS 和 MapReduce 框架均为通用框架，除索引之外可以支持很多其他的应用。

10.3　链接分析

在本节中，我们将继续讨论网络搜索，特别是侧重于如何利用页面之间的链接来改善搜索。在之前的章节，我们讨论了如何在 GFS 上使用 MapReduce 创建大型索引。现在建立好了索引，我们想看看如何提高网页在网络上的排名。当然，这里可以使用标准的 IR 模型。事实上，它们是支持网页搜索的重要组成部分，然而它们还不足够，其原因如下。

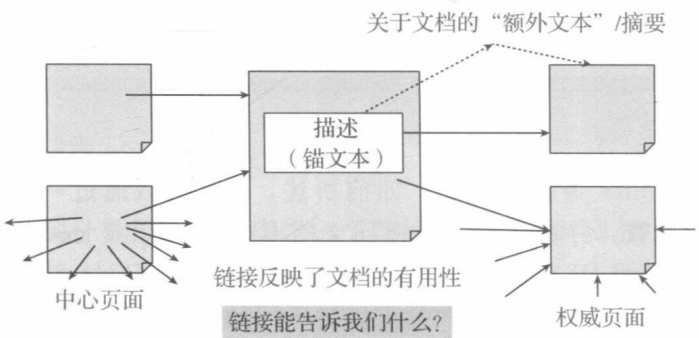
首先，在网络上人们的信息需求各不相同。例如，人们可能会搜索一个网页或入口页面——这不同于传统的图书馆搜索，在那里人们主要为了收集文献信息。这些类型的查询通常被称为*导航查询*，其目的是导航到特定的目标页面。对于这样的查询，我们可能会受益于使用链接信息。例如，导航查询可能是 Facebook 或雅虎金融。用户只是试图避免通

过在浏览器的地址栏中输入精确的 URL 来获取网页。

其次，网页文档比纯文本有更多的信息；有分级组织和注释，如页面布局、标题或超链接。这些功能能够通过使用额外的文档的上下文信息来提高评分。最后，信息质量发生很大变化。这一切意味着我们必须考虑多种因素来提高标准排名算法，从而使网页排序更
200 加健壮，并能够使垃圾信息发布者更难以通过操纵单一信号来改善单个网页排名。

基于以上考虑，研究人员对标准排序算法做出了一系列重大扩展。一个是利用链接来改善网页得分，这是本节的主要内容。另外还有一些算法利用以点击形式存在的大规模的隐式反馈信息。当然，这属于反馈技术的范畴，机器学习技术常应用于该任务。一般来说，网络搜索排名算法是基于结合各种特征的机器学习算法，例如在第 6 章提到的许多都是基于我们讨论的 BM25。在现代网络搜索系统中，链接信息是评分函数中的重要特征。

图 10-6 显示了一部分网页的快照。我们可以看到有很多连接不同页面的链接，而在中间则有一个链接的描述指向右侧的文档。这个描述文本被称为*锚文本*。它对搜索引擎是非常重要的，因为它提供了指向页面的一些额外的描述。例如，如果有人想要标记 **Amazon. com** 的首页，这个人可能会建立一个名为“大型网上书店”的链接，该链接指向亚马逊。这个描述与用户在查询框中寻找这样一个页面时输入的内容非常相似。假设有人在查询中输入*网上书店*或*大型网上书店*等内容，该查询将匹配页面中的此锚文本。这实际上提供了用于匹配被指向的页面的证据，即 Amazon 入口页面。因此，如果你匹配了一个描述链接到页面的锚文本，它为指向页面的相关性提供了很好的佐证。



201 图 10-6　链接提供了关于网页的有用信息

在图 10-6 的底部，有一些链接的模式，它们能够显示出文档的有用性。例如，在右侧可以看到一个页面有很多指向它的链接，这意味着许多其他页面指向该页面。这个表明这个页面非常有用。在左边你可以看到一个页面指向许多其他的页面。这是一个中心页面，可以让你看到很多其他的页面。我们称第一种页面为*权威页面*(authority page)，第二种为*中心页面*(hub page)。这个意味着链接信息有两个作用；一个是为匹配提供额外的文本(在存在锚文本的情况下)，另一个是提供一些额外的分数用于评估一个网页是一个中心页面还是一个权威页面。

10.3.1　PageRank 算法

Google 的 PageRank 是最初用于链接分析的最主要的技术，这个算法是一个利用页面链接信息的很好的例子。PageRank 捕获网页的流行度，也是权威的另一种说法。直觉上，链接类似于文献中的引用。考虑一个页面指向另一个页面；这与一篇论文引用另一篇论文非常相似。因此，如果一个网页经常被引用，我们可以认为这个页面更有用。PageRank

利用了这一直觉并以原理性的方法实现。从最简单的意义上讲，PageRank 基本上是在做引文计数或链接计数。

它通过两种方式改进了这个简单想法。一个是考虑间接引用。这意味着你不仅要考虑链入网页的数量，还要递归地考虑链入的网页中的链入信息。如果网页的链入网页本身有很多链入网页，则该网页将获得信誉度。总之，如果重要的网页指向一个网页，则这个网页也必然重要。另一方面，如果一个网页的链入网页未被很多页面所指向，则该网页不会得到很多信誉度。这是间接引用或级联引用的概念。

再次，我们可以通过论文来理解这个想法。如果你的论文是由 10 篇不太有影响力的论文引用，显然不及被 10 篇高引用率文章所引用。这是间接链接的一个例子，PageRank 的原理也是如此。另一个想法是通过平滑引用以调解尚未观察到的潜在引用。假设每个页面都有一个非零的伪引文计数。本质上，你可以想象许多虚拟链接将把所有的页面链接在一起，所以你实际上得到来自每一个节点的伪引用。

另一种理解 PageRank 的方法是随机访问每个网页。我们来仔细看一下这个例子，如图 10-7 所示。在左边有一个小图，其中每个文档 d_1、d_2、d_3 和 d_4 是一个网页，文档之间 202
的边是网页之间的超链接。让我们假设随机游走者可以访问这些网页中的任何一个页面。当随机游走者决定移动到不同的页面时，他们可以随机跟随当前页面的链接或随机选择一个其他的文档来跳转出当前整个文档集合。所以如果随机游走者是在 d_1，有一些随机游走者将遵循链接以一定的概率访问到 d_3 或 d_4。随机访问模式也考虑了如下情况，即游走者有时会觉得无聊而决定忽略实际的链接，随机跳转到网页上的任何页面。如果是游走者采取这种选择，他们将能够达到任何其他网页，即使没有直接到该页面的链接。基于这个模型，我们可以问这个问题，“平均来说，游走者到达某一页面的可能性有多大?” PageRank 正是用于计算这个概率的算法。

文档 d_i 的 PageRank 分数是游走者访问 d_i 的平均概率。直观地说，这应该与链入数成正比。如果一个页面有一个高链入值，那么它将有更高的机会被访问，因为该网页将有很大的概率被游走者沿着链接访问到。随机游走模式基于此原理对链入进行计数。但该算法也考虑间接链接，如果指向 d_i 的页面自己有很多链入，这将意味着随机游走者很可能会访问它们其中之一。这增加了访问 d_i 的机会。这是一个捕获间接链接和直接链接的好方法。

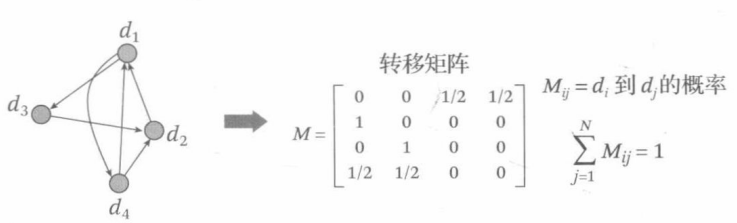


图 10-7　一个网络图及相应的转移矩阵的实例

在数学上，我们可以将这个文档网络表示为矩阵 $\boldsymbol{M}$，如图 10-7 的中部所示。每一行代表一个起始页面。例如，第一行表示从 d_1 去四个网页中的任何一个的概率。我们看到只有两个非零的条目。由于 d_1 只指向了两个网页，每个获得 1/2 的概率；因此如果我们选择从 d_1 任意访问其中任何一个网页，它们每个都有 1/2 的访问概率。因为 d_1 没有链接 203
到自身，它不链接到 d_2，所以 d_1 的前两列为零。M_{ij} 是从 d_i 到 d_j 的概率。由于随机游走者一定会去其中一个网页，每行的值总和应该为 1。现在，我们如何计算一个游走者访问

一个特定页面的概率？

我们可以通过如下公式计算出到达页面的概率：

$$p_{t+1}(d_j) = \underbrace{(1-\alpha)\sum_{i=1}^{N} M_{ij}\, p_t(d_i)}_{\text{通过链接到达}d_j} + \alpha \underbrace{\sum_{i=1}^{N} \frac{1}{N} p_t(d_i)}_{\text{通过随机跳转到达}d_j} \tag{10-1}$$

公式左半部分是下一次在时间 $t+1$ 访问页面 d_j 的概率。在右侧，我们可以看到公式包含在当前时间节点 t 处访问页面 d_i 的概率。该公式汇总了两种在时间 $t+1$ 达到页面 d_j 的可能性：通过随机游走或链接。该公式的第一部分捕捉随机游走者通过链接访问页面的概率。随机游走选择这个策略的概率为 $1-\alpha$；因此，这部分之前有一个 $1-\alpha$ 因子。这部分将游走者在时间 t 所能访问的所有可能的 N 个页面的概率进行加和。在累加和里是两个概率的乘积。一个是游走者在时间 t 处在 d_i 的概率，即 $p_t(d_i)$。另一个是从 d_i 到 d_j 的转移概率，用 M_{ij} 表示。所以为了到达 d_j 页面，游走者必须首先在时间 t 处于 d_i 网页，并将遵循链接从 d_i 到 d_j。第二部分是道理相似的一个累加和。唯一的区别就是这部分的转移概率相同，数值为 $1/N$。这部分计算通过随机跳转到达这个页面的概率，其中 α 是随机跳转的概率。

由此我们也能够获知 PageRank 为何对转移矩阵进行平滑。可以这样想，$1/N$ 来自另一个转移矩阵，其所有元素均为 $1/N$。很显然，我们可以合并这两部分。因为它们是同样的形式，我们可以想象有一个不同的矩阵是这个 $\boldsymbol{M}$ 和均匀矩阵 $\boldsymbol{I}$ 的组合。在这个意义上，PageRank 运用这个想法进行平滑以确保在转移矩阵中不存在值为 0 的条目。

现在，我们可以想象，如果我们要计算平均概率不必考虑时间标记。所以我们去掉时间下标，并假设它们将被平等对待；由此我们会获得 N 个方程，因为每个页面都有自己的等式。同样也有 N 个变量。这意味着我们现在有一个包含 N 个变量的 N 元线性方程组。该问题归结为解这个方程组，我们可以写成如下形式：

204

$$p(d_j) = \sum_{i=1}^{N}\left[\frac{1}{N}\alpha + (1-\alpha)M_{ij}\right]\cdot p(d_i) \rightarrow \boldsymbol{p} = (\alpha \mathrm{I} + (1-\alpha)\boldsymbol{M})^{\mathrm{T}}\boldsymbol{p} \tag{10-2}$$

其中

$$I_{ij} = \frac{1}{N} \quad \forall\, i,\ j$$

向量 $\boldsymbol{p}$ 等于矩阵的转置乘以 $\boldsymbol{p}$。转置矩阵实际上是以矩阵形式写成的从 1 到 N 的和。回想线性代数，这正是特征向量的方程。因此，这个等式可以通过使用迭代算法来解决。这个迭代算法叫作幂迭代(powen iteration)，我们只是随机以一个 $\boldsymbol{p}$ 开始。通过将转置的矩阵表达式乘以 $\boldsymbol{p}$ 再一次更新 $\boldsymbol{p}$。

我们来看一个具体的例子：将 α 设置为 0.2。这意味着有 20%的概率随机跳到网络上的一个页面中，以 80%的概率随机地跟随当前页面的链接进行跳转。通过对图中已有链接进行编码获得原始的转移矩阵 $\boldsymbol{M}$。接着我们有这个表示随机跳转的均匀平滑转移矩阵 $\boldsymbol{I}$。我们将它们与通过 α 的插值组合形成另一个矩阵，称为 $\boldsymbol{A}$：

$$\boldsymbol{A} = (1-0.2)\boldsymbol{M} + 0.2\boldsymbol{I} = 0.8\begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix} + 0.2\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \tag{10-3}$$

PageRank 算法首先会随机初始化 $\boldsymbol{p}$，然后通过使用矩阵乘法来迭代更新它。就 $\boldsymbol{A}$ 而言如果我们重写这个矩阵乘法，会得到以下结果：

$$\begin{bmatrix} p_{t+1}(d_1) \\ p_{t+1}(d_2) \\ p_{t+1}(d_3) \\ p_{t+1}(d_4) \end{bmatrix} = \boldsymbol{A}^{\mathrm{T}} \begin{bmatrix} p_t(d_1) \\ p_t(d_2) \\ p_t(d_3) \\ p_t(d_4) \end{bmatrix} = \begin{bmatrix} 0.05 & 0.85 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.45 \\ 0.45 & 0.05 & 0.05 & 0.05 \\ 0.45 & 0.05 & 0.05 & 0.05 \end{bmatrix} \begin{bmatrix} p_t(d_1) \\ p_t(d_2) \\ p_t(d_3) \\ p_t(d_4) \end{bmatrix} \tag{10-4}$$

如果要计算 d_1 的更新值，则将 $\boldsymbol{A}$ 中首行乘以上一次迭代的 PageRank 分数的列向量。这个是我们如何更新向量；我们从一些初始值开始并迭代地将矩阵相乘，产生一组新的 205
值。重复这个乘法直到值在 $\boldsymbol{p}$ 收敛。从线性代数我们知道，由于矩阵中没有零值，所以这样的迭代能保证收敛。我们能够获得所有页面的 PageRank 值。

有趣的是，这个更新公式可以被解释为在图中传播分数。我们可以想象在每个页面上都有初始化的值，如果你看看这个等式，我们将所有链向该页面的网页的值结合到一起。也就是说，我们将查看所有指向页面的网页，将它们的分数与传播的分数结合起来，以获得该文档的下一个分数。我们对所有文档重复这一过程，从而将沿着网络传播概率。

在实践中，PageRank 分数的计算实际上非常高效，因为矩阵是稀疏的——这意味着如果当前页面没有链入信息，我们不必在计算中处理该页面。还可以对公式进行规范化，所获得的结果在形式上略有不同，但页面的相对排名不会改变。规范化是为了解决没有链出所带来的潜在问题。在这种情况下，从当前页面到下一页的概率和将不会为 1，因为当我们假设会以一定的概率尝试访问链接时，已经失去了一些概率(虽然在这种情况下，没有链接要访问)。

PageRank 有很多扩展。一个扩展是做面向特定查询的 PageRank，也称为个性化 PageRank(Personalized PageRank)。例如，在特定主题的 PageRank 下，我们可以简单地假设当游走者感到无聊时，他们不会随机跳转到网络上的任何页面。相反，他们只跳转到那些与查询相关的页面上。例如，如果查询的内容关于体育，那么可以假设当我们做随机跳转的时候，随机跳到一个体育网页。通过这样做，我们的 PageRank 分数始终与运动相关。所以，如果你知道目前的查询是关于运动，我们可以使用这个专门的 PageRank 分数来对结果排名。显然，这比应用于整个网络的普通的 PageRank 更好。

PageRank 是一个通用算法，可以在许多其他应用程序中使用，如网络分析，尤其是在社交网络中。我们可以想象，如果你在社交网络上计算一个人的 PageRank 分数(一个链接表示一个关注关系)，你会获得和这个人相关的一些有意义的分数。

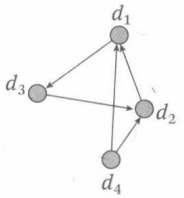
10.3.2　HITS 算法

我们已经谈到了 PageRank 可以作为获取权威页面的一种方式。在本节开始的时候，我们也提到中心页面是有用的。我们将讨论的另一种算法称为 HITS，旨在为每个页面计算这些分数。 206

权威页面捕捉被广泛引用的页面的直觉知识。中心页面是指向质量好又权威的页面，或者以链接的形式收集一些知识的页面。HITS 算法的主要思想是一个帮助改善中心和权威度得分的增强机制。算法假定权威度高的页面被更好的中心页面引用。这意味着如果你被许多中心分数(hub score)高的网页引用，那么你的权威分数也会增加。同样，好的中心指向权威度高的网页。所以如果你指向很多的权威度高的页面，那么你的中心分数将会增加。类似于 PageRank，HITS 被普遍使用并应用于许多图和网络分析应用。我们会简要描

述它工作的过程和机理。

图 10-8 显示了以下内容。首先构造邻接矩阵 $\boldsymbol{A}$，如果 d_i 链接到 d_j，则在位置 A_{ij} 处为 1，否则为 0。页面的中心分数 $h(d_i)$定义为它指向的所有页面的权威分数的总和。在第二个等式中，页面的权威分数 $a(d_i)$定义为指向它的所有页面的中心分数的总和。这形成了一个迭代的强化机制。



邻接矩阵

$$A=\begin{bmatrix}0&0&1&1\\1&0&0&0\\0&1&0&0\\1&1&0&0\end{bmatrix}$$

初始值：$a(d_i)=h(d_i)=1$

$$h(d_i)=\sum_{d_j\in \mathrm{OUT}(d_i)} a(d_i)$$

$$a(d_i)=\sum_{d_j\in \mathrm{IN}(d_i)} h(d_i)$$

迭代

$$\bar{h}=A\bar{a};\ \bar{a}=A^T\bar{h}$$

$$\bar{h}=AA^T\bar{h};\ \bar{a}=A^TA\bar{a}$$

归一化：

$$\sum_i a(d_i)^2=\sum_i h(d_i)^2=1$$

图 10-8　HITS 算法在小型图上的运算过程

这两个方程也可以用矩阵形式写出来。中心向量是邻接矩阵和权威向量的乘积。第二个等式可以写成权威向量等于 $\boldsymbol{A}^{\mathrm{T}}$ 与中心向量的乘积。这些只是表达方程的不同方式。有趣的是，如果你看矩阵的形式，你可以在第一个等式中插入权威度的等式。也就是说，你
207 实际上可以完全消除权威向量，你得到只有中心分数的方程。我们可以对中心分数公式使用同样的技巧。因此，虽然我们把这个问题定义为计算中心分数和权威分数，我们实际上可以消除其中的一个获得关于另一个的方程式。

这和 PageRank 的区别在于现在矩阵实际上是一个邻接矩阵与其转置的乘法。经过数学运算后，我们会计算一个非常近似的问题。在 HITS 中，我们将初始化这些值并运用矩阵方程 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 和 $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$。我们仍然需要对每次迭代之后的邻接矩阵进行规范化。这将使我们能够控制值的增长，否则值会变得越来越大。

总而言之，这一节已经表明链接信息是非常有用的。尤其是锚文本在页面的文本表示中是一个重要特征。我们还谈到了两个在网络搜索中主要的链接分析算法：PageRank 和 HITS 算法。除了标准的 IR 排名功能之外，两者都可以为网页生成分数。PageRank 和 HITS 以及一些变体算法是非常通用的并且有用的，所以除了互联网，这些算法还能分析其他的图或网络。

10.4　排序学习

在本节中，我们讨论使用机器学习来在排序函数中结合多种不同的特征，以优化搜索结果。我们已经讨论了许多文档排序的方法。我们谈到了一些检索模型，如 BM25 或查询相似模型；这些可以生成基于内容的分数，用于将文本与查询进行匹配。我们也谈到了基于链接的方法，如 PageRank 算法，该方法可以提供一个额外的分值帮助提升排序。现在的问题是如何将所有这些特征(可能还有很多其他的特征)结合起来做排序？这对网页排序非常有用，不只是为了提高准确性，还要提高排序的健壮性，从而使垃圾信息发布者不能轻易通过改变一个或几个特征推广一个网页。

排序学习的总体思路是利用机器学习来结合这些特征，优化不同特征的权重以产生最佳的排序函数。我们给定一个查询－文档对(q，d)，可以定义一些特征。这些特征不一定

是基于内容的特征。他们可能是关于查询文档的一个分数，分数的获得可能是基于BM25、查询相似、枢轴长度规范化或 PL2 等检索函数。也可以是一个以链接为基础的分数，如 PageRank、HITS，或对页面的锚文本(指向文档 d 的链接的描述)进行检索的模型 208
应用。这些都可以作为这个文档是否与查询相关的线索。我们甚至可以包含这样的特征，如 URL 是否包含波浪符号，如果包含可能表示这个网址是一个主页的地址。

问题是我们如何把这些特征组合成一个单一的得分？在这个方法中，我们简单地假设这个文档与这个查询相关的概率是包含所有这些特征的一个函数。相关性的概率与包含参数的特征计算公式有关联。这些公式中的参数控制着不同特征对相关性的不同影响。这当然只是一个假设。是否这个假设真的有意义还是一个悬而未决的问题。

当然，接下来的问题是如何估计这些参数。我们如何知道哪些特征应该具有高权重以及哪些特征应该具有低权重？这是一个训练或学习的任务。

在这种方法中，我们使用训练数据。这是已经被用户判断过的数据，所以我们已经知道了相关性的判断结果。我们知道哪些文档与哪些查询高度相关(排序高)，这些判断结果来自于用户的真实判断或者使用点击信息进行近似判断(我们在第 7 章讨论)。我们将尝试在训练集上通过调整参数优化我们的搜索引擎的检索精确性(例如使用 MAP 或 NDCG)。训练数据看起来像一个元组表。每个元组都有三个元素：查询、文档和判断。一个基于逻辑回归的方法如下：

$$\log \frac{P(R=1|q,d)}{1-P(R=1|q,d)} = \beta_0 + \sum_{i=1}^{n} \beta_i X_i \tag{10-5}$$

这是许多不同方法中的一种，实际上是其中较简单的方法。在这种方法中，我们简单地假设文档与查询的相关性是涉及的特征的线性组合。这里我们用 X_i 表示第 i 个特征值，并且我们可以尽可能多地定义特征。我们假设这些特征可以以线性方式组合。参数 β_i 影响特征 X_i 的权值。β_i 的值越大将意味着这个特征的权重更大，说明该特征对网页分数计算影响越大。

相关性概率的计算公式如下：

$$P(R=1|d,q) = \frac{1}{1+\exp\left\{-\beta_0 - \sum_{i=1}^{n} \beta_i X_i\right\}} \tag{10-6}$$ 209

我们知道相关性概率的取值在[0，1]范围内，假设评分函数是特征的线性组合的一种转换形式。我们可以直接基于 β 和 X 的线性组合得到一个评分函数，但是这个线性组合的值很容易大于 1。因此，我们使用逻辑回归而不是线性回归，从而将这个组合的值映射到值域[0，1]中。这使我们能够将相关性概率(值在 0 和 1 之间)与采用任意系数的线性组合建立联系。如果我们将这个权重组合重写成一个概率函数，我们会得到预测的分数。

如果这个特征和权重的组合等于一个较大的值，那么这个文档具有较高的相关性。这不一定是最好的假设，但是它是将特征与相关性概率联系起来的一种简单方法。

接下来的任务是如何估计参数，使函数可以真正被应用；也就是说，我们需要估计 β 值。我们来看看图 10-9 中的实例。

在这个例子中，我们有三个特征。一个是文档关于查询的 BM25 得分。一个是文档的 PageRank 分数，它可能依赖或者不依赖于查询。我们也可能有一个依赖于查询的话题敏感的 Pag-

	$X_1(q, d)$	$X_2(q, d)$	$X_3(q, d)$
$d_1(R=1)$	0.7	0.11	0.65
$d_2(R=0)$	0.3	0.05	0.4

图 10-9　在排序中对多种特征的一个组合实例

eRank 分数。最后，我们有一个文档的锚文本的 BM25 分数。这些是关于特定的(文档，查询)对的三个特征值。在这种情况下文档是 d_1，并且预测结果是相关。文档 d_2 是另一个不同的训练实例，有不同的特征值，但在这个案例中它是不相关的。当然，这是一个过度简化的例子(只有两个实例)，但能够说明上述问题。

我们使用最大似然估计方法来估计参数。我们将根据特征预测文档的相关性。运用模
210 型对这两个文档的相关性状态进行如下相似性计算

$$p(\{q,\ d_1,\ R=1\},\ \{q,\ d_2,\ R=0\})=\frac{1}{1+\exp\{-\beta_0-0.7\beta_1-0.11\beta_2-0.65\beta_3\}}\times\left(1-\frac{1}{1+\exp\{-\beta_0-0.3\beta_1-0.05\beta_2-0.4\beta_3\}}\right)$$

我们假设相关性概率与特征以上述公式的方式相关。我们将看看当 β 的值是什么时我们可以有效地预测相关性。d_1 的表达式应该比 d_2 的表达式给出更高的值；事实上，我们希望 d_1 的价值接近于 1，因为这是一个相关的文档。

我们来看看这个问题如何用数学方式表达。这与表达一个文档的概率相似，只不过我们不仅讨论词汇的概率，还会考虑相关性概率。我们需要插入 X 值。β 值还不知道，但是如果我们假定这个模型，这个表达式为我们提供了这个文档相关的概率。因为 d_1 是一个相关的文档，我们希望最大化这个概率。对于第二个文档，我们想预测文档不相关的概率。这意味着我们必须计算 1 减去相关性概率。这就是表达式背后的机理，这是我们预测这两个相关性值的概率。整个方程计算的是概率，其中 d_1 和 d_2 分别的观察值为 $R=1$ 和 $R=0$。模型的目标是调整 β 值，使整个表达式达到最大值。换句话说，我们在寻找一个函数，选择 β 值使这个函数尽可能大。

当回归参数学习完后，一旦有了特征，我们可以用这个表达式来表示任何新的查询和新文档。继而应用这个公式为特定查询生成排名分数。

还有更多比基于回归的方法更先进的算法。他们通常试图在理论上优化检索评价方法，例如 MAP 或 NDCG。请注意，我们刚刚讨论的优化目标是与检索评价方法没有直接关系的。通过最大化预测，我们不一定优化这些文档的排序。想象一下，虽然我们的预测可能不会太糟，但排序可能是错的。对于 d_2，我们可能有比 d_1 更大的相关概率。所以，尽管基于概率计算的函数也不差，从检索的角度来看那不会是好的。更高级的方法将尝试
211 纠正这个问题。当然，优化问题将更难解决。我们还可能会遇到另外一个案例，对不相关的文档预测相关性概率达到 0.9 左右。虽然预测的值非常高，但只要真正相关的文档的相关性概率大于 0.9，最后的排序结果仍然是用户可以接受的。

实际上这些排序学习方法的使用相当普遍。除了检索问题，它们还可以应用在许多其他的排序问题上。例如推荐系统、计算广告、摘要等其他相关的应用程序都可以使用这种方法来解决。

总而言之，我们谈到了通过机器学习来结合特征对排序结果进行预测。其实几十年前在信息检索中已经开始运用机器学习方法。Rocchio 反馈(在第 7 章中讨论过)是机器学习方法，被用于学习最佳反馈。许多算法是由大量的训练数据驱动，这些数据是用户的点击信息。这些数据提供了很多关于相关性的有用信息，机器学习方法可以对其加以利用。机器学习方法还能够结合多种不同的特征，从而预测出一个准确的排序。由于网络上有众多特征能够被用于搜索问题，网络搜索尤其需要引入机器学习方法。使用许多不同的特征也增加了评分函数的健壮性，这对于打击垃圾信息很有用。现代搜索引擎均采用某种机器学习技术并结合了多种功能来优化排序，这是 Google 和 Bing 等当前搜索引擎的主要特点。

10.5　网络搜索的未来

由于本章对搜索引擎的方方面面都有所介绍，所以我们就网络搜索和智能信息检索系统的未来可能的趋势简要地谈一谈。为了进一步提高搜索引擎的精度，考虑信息需求的特殊性很重要。一个特别的趋势是有更多更专业化和个性化的搜索引擎，这可以称为*垂直搜索引擎*。这些垂直搜索引擎能够预期比目前的一般搜索引擎更有效，因为他们可以假设属于特定群组的用户可能具有共同的信息需求。

由于这种定制特点，搜索引擎也可以做个性化服务。因为我们对用户有了更好的了解，搜索可以个性化。对搜索引擎的领域进行限制也可以在处理文档中有一些优势，因为我们会更好地了解这些文档。例如，特定的单词在某个领域中可能不会存在歧义，所以我们可以绕过歧义的问题。

212

我们期望看到的另一个趋势是搜索引擎能够随时间推移以终身学习的形式不断学习。这很有吸引力，因为这意味着搜索引擎将能够自我改善。随着越来越多的人使用它，搜索引擎将会越来越好。这已经发生，因为搜索引擎可以从相关反馈中学习。使用搜索引擎的用户越多，对频度高的查询的返回结果质量就越好。

第三个趋势可能是整合信息获取。搜索、导航和推荐可以结合起来，形成一个完整的信息管理系统。在本书的开头，我们讨论了推送访问与拉取访问；这些模式可以结合使用。例如，如果搜索引擎检测到用户对搜索结果不满意，那么可以进行“记录”。在将来，如果一个新的文档被爬取，在记录中匹配上用户需要记录的信息，这个新文档可以推送给用户。目前，大部分信息推荐的情况是广告，但在未来，你可以想象推荐能够以多模式信息获取无缝集成到系统中。

另一个趋势是系统试图超越搜索功能进而支持用户任务。毕竟，人们搜索的原因是要解决一个问题或者做出决定来执行任务。例如，消费者可能会搜索关于产品的意见，以便购买一个产品，所以搜索将有益于支持整个购物流程。例如，你可以在搜索结果中直接看评论信息；如果用户决定购买产品，他们可以直接点击一个按钮进入购物网站采购。尽管对购物有很好的支持，但目前的搜索引擎却没有为许多其他任务提供良好的任务支持。研究人员可能想找到相关工作或建议的参考文献。目前，对写一篇论文之类的任务的支持并不多。

我们可以考虑任何智能系统——特别是智能信息系统——由三个节点指定。如果我们将这些节点连接成一个三角形，那么我们可以具体化一个信息系统。我们可以把这个三角形称为*数据-用户-服务三角形*。你可能会问的三个问题如下。

- 你在服务谁？
- 你在管理什么样的数据？
- 你提供什么样的服务？

213

这些问题指明了一个信息系统；有很多不同的方法来连接它们。根据它们如何连接，我们可以指定所有不同的系统类型。我们来看一些例子。

在图 10-10 的顶部，有不同类型的用户；在左侧，有不同类型的数据或信息；在底部有不同的服

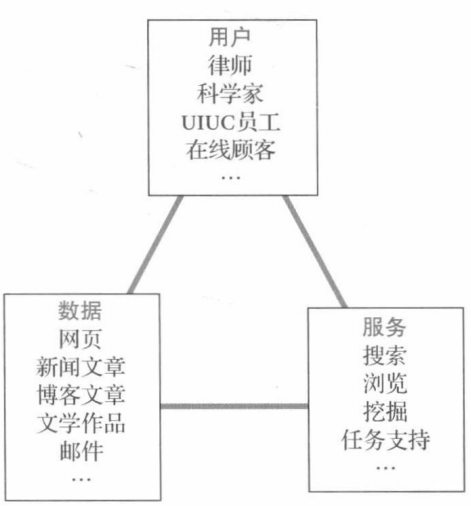


图 10-10　数据-用户-服务三角形

务功能。现在想象你可以通过不同方式连接所有信息。例如，如果你通过网页连接每个人，并支持搜索和浏览，你会得到网络搜索。如果我们通过组织文档或企业文档连接大学的员工，并支持搜索和浏览，我们将获得一个企业搜索。

我们可以把科学家与文献资料联系起来，服务包括搜索、浏览、提醒新的相关文档，挖掘或分析研究趋势，提供任务和决策支持。例如，我们能够针对一篇研究论文自动生成一段相关工作；这更接近任务支持。那么，我们可以认为这种智能信息系统将是一种文献助理类型。

如果我们将在线购物者与博客文章或产品评论联系起来，那么我们就可以帮助这些人改善他们的购物体验。我们可以提供数据挖掘分析评论的功能，比较产品和产品评论的情
214 感，并对要购买的产品提供任务或决策支持。我们也可以通过电子邮件将客户与客户服务人员建立联系。由此可以考虑建立一个系统，通过对这些电子邮件的分析发现客户的主要诉求。我们可以想象一个系统能够自动生成对顾客电子邮件的响应消息，在适当的时候智能地附加促销广告。如果系统检测是一条积极的信息(不是投诉)，则可以借此机会附上一些推广促销信息。如果是投诉，那么系统可能会首先自动生成一些通用的回复对客户进行响应，并告知他或她随后会收到一个详细的答复。所有的这些旨在帮助人们提高生产力。

图 10-10 显示了技术的趋势和从三个角度刻画了智能信息系统。在图的中部有一个三角形，它连接关键字查询以词袋形式来搜索。这意味着目前的搜索引擎基本上为用户提供搜索支持，大部分基于关键字查询对用户建模，通过词袋表示来查看数据。目前的搜索引擎并不真正“理解”索引中的信息文档。研究者可以考虑一些趋势，将每个节点远离中心、向更高级的功能发展。想象一下，如果我们能够超越关键字查询，查看用户搜索的历史记录，然后进一步对用户建模来理解用户的任务环境、上下文或其他信息。显然，这能够推动个性化以及创建更完整的用户模型，这是建立智能化的信息系统的一个主要方向。在文档方面，我们也超越了词袋表示，通过实体关系表示来实现文档内容的表示。这意味着我们能够识别人们的姓名、他们之间的关系、地点以及任何其他潜在的有用信息。通过运用当今的自然语言处理技术已能够实现这些任务。

谷歌通过知识图谱启动了一些工作。如果我们可以不借助过多的人力达到上述功能，搜索引擎就能够提供更好的服务。在未来，我们希望设计一个优越的知识表示方法，能够在其中加入推理规则，使搜索引擎更加智能。这需要进行大规模的语义分析，也许最初更适合于垂直搜索引擎。也就是说，在一个特定的领域取得进展比较容易。

在服务方面，我们看到需要超越搜索来支持一般信息获取；搜索只是获取信息的一种方式。超越信息获取，我们也需要帮助人们对发现的信息进行消化理解，这需要信息分析或数据挖掘的技术支持。我们必须找到模板，或者将文本信息转换成可以在应用程
215 序中使用的真实知识或可用于决策的可操作知识。此外，这些知识将用于帮助用户提高完成任务的生产力。在这个层面上，我们预计未来的智能信息系统将会对交互式任务提供支持。

我们在这里强调交互(interactive)，因为对用户和系统相结合的智能进行优化是很重要的。我们可以以一种自然的方式从用户那里得到一些帮助，而不是假设系统必须完成一切工作。那就是用户与机器能够以一种智能和高效的方式进行协作。这样结合起来的智能将会很高，总的来说在解决用户目前的问题时，我们可以最大限度地减少他们的整体工作量。

这是未来的智能信息系统的蓝图，希望由此为如何基于目前的技术做出更多的创新性

工作提供一些启示，同时也促使我们关注本书后面的章节中所介绍的其他技术。

书目说明和延伸阅读

PageRank 的经典参考是 Page 等[1999]，关于 HITS 的参考是 Kleinberg[1999]。Lin 和 Dyer[2010]详细介绍了在文本处理应用程序中运用 MapReduce 技术，特别包括一个详细说明，介绍如何使用 MapReduce 来构建倒排索引。Liu[2009]中对排序学习的研究工作进行了很好的总结。

练习

10.1　检查几个常见网站的 **robots.txt** 文件。你能弄清楚这个文件的格式吗？这些网站不希望你爬取什么类型的数据？什么是用户代理？

10.2　简单的网络爬虫。在 Linux 或 Mac 上，尝试使用 **wget** 下载网页 **wget http://www.[insert-domain-here].com/**。该文件可能以扩展名 **.html** 保存。在你最常用的文本编辑器中打开它。它只是一堆 HTML 代码，如果右击浏览器中的页面并选择“查看源代码”，你将看到同样的结果。

10.3　解析 Web 内容。网页爬虫的一个重要步骤就是将 HTML 解析为纯文本。一些函数库能够完成这个工作。这些函数库提供了所有的链出链接(如标签 **href=...**)，爬虫会根据链接逐一爬取网页并避免陷入死循环。系统可以通过维护一个已访问列表避免上述情况。 216

使用 **wget** 不是构建简单网页爬虫程序的唯一方法。可以使用 Ruby 或 Python 创建自己的网页爬虫程序。这些语言有许多有用的函数库。也可以使用现有的网络爬虫。对于 Python 语言，Scrapy 和 Beautiful Soup 是两种流行的爬虫工具。

最后一个重要的注意事项是要在页面请求之间留出一个等待时间。不仅是因为出于礼节考虑在请求之间等待几秒钟以避免攻击服务器，如果尝试爬取太快可能会被封锁！

使用上述方法创建一个简单的网页爬虫。将初始爬取的网页数量控制在 100 页。

10.4　JavaScript 爬虫。对于一个简单的页面，简单地调用 **wget** 能够完成爬取工作。但是现在大多数网页都是动态生成网页内容，而不是可下载源的一部分。尝试爬取生成动态内容的页面(提示：可以找到一个使用 **wget** 生成动态内容的网页，并搜索你知道的页面上的文本。如果在下载的 HTML 文件中找不到它，则该内容是动态生成的内容)。

大多数现代网站都是由动态生成的内容构成的。简单地下载基本的 HTML 源将不会为检索提供必要的内容。我们需要实际加载页面并运行 JavaScript 语言，这被称为填充内容。

打开动态网页的 URL 地址，在浏览器中进行爬取，并打开 JavaScript 语言控制台。(在 Chrome 浏览器中，它是 **CTRL-SHIFT-I**。)可以通过 JavaScript 语言与页面进行交互。尝试在控制台中输入：

```
alert('I'm a popup!')
```

然后尝试输入

```
for(var i = 0; i < 5; ++i) { console.log('Hello ' + i); }
```

除了做恼人的弹出窗口和无用的计数器，我们可以访问页面标题：

```
document.title
```

或者我们可以访问文本内容：

```
document.body.innerText
```

这是我们想要的索引！使用 PhantomJS 等技术实现一个 JavaScript 语言爬虫。
217 我们提供了一个简单的脚本来下载一个页面，代码位于：http://sifaka.cs.uiuc.edu/ir/textdatabook/files/text-scraper.js。尝试使用该脚本实现一个真正的爬虫，而不是只能下载一个页面。

10.5 爬取一个域名。现在你已经知道如何下载一个 HTML 文件了，你可以开始抓取一个域名。这就意味着从一些基本的 URL 地址逐层深入地获取和爬取链接。**wget** 提供了一些有用的功能使这个工作变得简单。

```
wget --recursive --level=3 --wait=2 --accept html [url]
```

这个命令可以让 **wget** 逐层递归地爬取深度为 3 的域名。它在请求之间等待时长为 2(这种行为是合乎情理的，并且会帮助你避免被封锁)。这段代码最终只下载 HTML 页面，因为这些是我们想要索引的文本。

输出的结构与在网络上的结构保持一致，将其下载到本地的工作目录中。

如果不手动停止(使用 CTRL-C)，这个爬虫将继续爬取网页，直到下载了指定深度的所有页面。从一个保守的深度开始，因为你不能确定某个域下有多少网页。

10.6 清洗 HTML 文件。在向搜索引擎的索引添加页面之前，我们可能需要“清洗”HTML 页面。这意味着将 HTML 文件转换为纯文本，使得这些关键字能够更好地匹配爬取到的文档中的单词。

有一些工具可用于将 HTML 转换成文本，其中一些是提供在线服务的。在使用中，我们希望有一个基于命令行的工具，可以自动化地完成爬取工作。我们建议使用 Python 或 Ruby 语言。下面两个简单的程序实现了同样的工作。

- Python：

```
from bs4 import BeautifulSoup

html = open('filename.html').read()
soup = BeautifulSoup(html)
print soup.get_text() # or save to another file
```

对于这种方法，需要安装 BeautifulSoup 库。

- Ruby：

218

```
require 'nokogiri'
html = File.open('filename.html').read
noko = Nokogiri::HTML(html)
puts noko.text # or save to another file
```

对于这种方法，需要安装 Nokogiri 库。

尝试使用这些数据清洗方法中的一个或两个来解析已爬取的文件。或者用不同的技术实现自己的数据清洗方法。

10.7 通过将爬虫、数据清洗和 META 相结合，创建能够动态更新的网络搜索引擎。可以将爬虫程序设定为在指定的时间间隔内运行。一旦下载并清洗新文件，将重新创建索引。

10.8　对如何改进网页搜索的排序函数提出建议，考虑额外的页面信息，例如标题栏或页面布局。

10.9　编写 MapReduce 伪代码用以创建倒排索引，倒排索引中包含使用狄利克雷先验平滑或 Jelinek-Mercer 平滑对文档进行排序的所有必要信息。

10.10　如果我们在网上添加新的页面，其他现有的 PageRank 分数会发生什么变化？尝试做出说明。

10.11　比较 PageRank 与个性化的 PageRank。它们是否可以预先计算以节省查询处理时间？请阐述一下可以或者不可以的原因。

10.12　给出一个查询，得分高的权威页面恰好是期望的文档；再给出一个查询，该查询中高得分的中心页面是期望的文档。

10.13　阅读完第 15 章之后，你可能会有一些关于如何设计一个排序学习算法的想法。例如，你能否概述一下如何为一组训练查询优化 MAP？

10.14　回想第 9 章，排序学习算法的优化目标是什么？MAP 是最好的选择吗？请阐述一下是或者不是的原因。

10.15　提出一种将用户反馈与排序学习算法相结合的方法。

219
~
220

第 11 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

推荐系统

前面章节关于搜索引擎系统的讨论主要针对短期(临时)信息需求问题。短期信息需求指的是用户主动从静态信息源获取相关信息的临时需求，比如图书馆或网络搜索。反之，大多数用户也有长期的信息需求，如从某些动态信息源获取的经过过滤或推荐的文档(或任何其他项目类型)。这种情况下，用户往往被动地接受系统推送的信息，系统可能是新闻过滤器、电子邮件过滤器、电影/书籍推荐系统或文献推荐系统等。尽管推荐系统(强调为用户推荐有用的项目)和过滤系统(强调为用户过滤掉无用的项目)存在一些差别，但是用到的技术十分相似。方便起见，后文介绍中会交替使用“推荐”和“过滤”这两个概念。

临时搜索通常无法获得足够的用户反馈，而在过滤系统中，可以预期从用户那里收集到丰富的反馈信息。因此，从用户反馈中学习以提升过滤效果是十分重要的。

在过滤系统中，文档是从动态信息源传递过来的。当文档到达时，系统必须立即判断出它是否和用户相关。这相对于搜索引擎提供给用户一个排序列表，依靠用户自己进行选择来说，要更加困难。另一方面，由于能够收集到反馈信息，即可以了解更多用户喜好，从而也更加容易区分用户相关和不相关的文档。

过滤的最本质问题是判断用户 u 对项目 x 的喜好。通常采用以下两种方法：

- 内容过滤：考察用户 u 的喜好并刻画 x。
- 协同过滤：考察喜欢 x 的用户并刻画 u。

内容过滤(content-based filtering)是根据学习到的用户对内容的喜好给用户推送符合其兴趣内容原型的项目。协同过滤(collaborative filtering)是指根据其他相似用户的喜好推断用户的喜好，假如与目标用户相似的用户喜欢某个项目，则该目标用户可能也喜欢这个项目。因此，只要我们能够得到用户对项目的评分，协同过滤可以用于任何项目的推荐。而基于内容的过滤只适合于我们知道如何度量项目之间的相似度的情况。通常在特定的某个应用中，这两种方法结合起来会取得更好的过滤效果。

基于内容的过滤可通过在检索系统的基础上加入阈值判定模块来实现。但是阈值的设定存在两个挑战。首先，在最开始时，必须在不需要用户提供太多信息的情况下，设置初始阈值。其次，随着时间的推移，需要根据反馈来优化这个阈值。尽管已经出现了许多阈值学习的方法，但在实际应用中，例如后面要讨论的 beta-gamma 这类简单的阈值设定方法也已经能够取得较好的结果。

协同过滤的主要思想是根据相似用户对某个项目 x 的加权平均值来预测当前活跃用户 u 对于项目 x 的评分。因此，这种方法应该涉及两个步骤：第一步，通过计算两个用户向量的相似度检索当前用户 u 的相似用户。每个用户可以表示为一个评分向量(即该用户给出的全部评分)。两个向量的相似度通常由余弦相似度或皮尔森相关系数来度量，这两种方式经实践检验效果较好。第二步，计算所有被检索到的相似用户对 x 的评分的加权平均值，其中权重是目标用户与相应用户之间的相关性。

在这一章我们将看到，许多推荐系统是前面讨论过的信息检索系统和技术的扩展。因此，如果读者不熟悉基本概念或术语，在本章之前请先阅读第 6 章。接下来，本章将继续

介绍基于内容的推荐，然后是基于用户的推荐(协同过滤)。

11.1　基于内容的推荐

图 11-1 展示了一个接收内容流的通用信息过滤系统。系统基于项目自身内容或是其他喜爱该项目的用户，判断是否将该项目传递给用户。在本节中，过滤系统将检查该项目的内容，并将其与用户的偏好和反馈进行比较，而不考虑来自其他用户的信息。 222

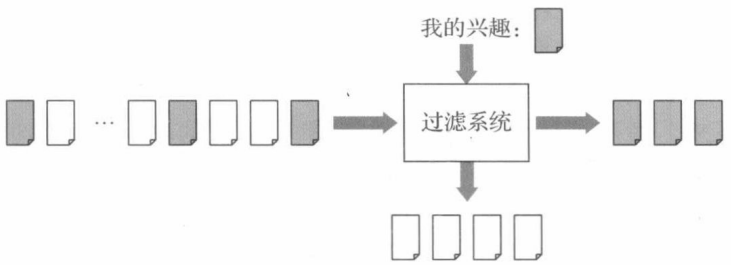


图 11-1　信息过滤系统框架图，根据用户兴趣将相关文档推送给用户

如图 11-2 所示，在信息过滤系统中，通常存在一个理解用户兴趣，即用户兴趣画像(user interest profile)，的二元分类器[⊖]。文本推荐初期用户兴趣画像可由文本摘要或关键词来充当，将用户指定的关键词或选择的类别信息作为初始化模块的输入。

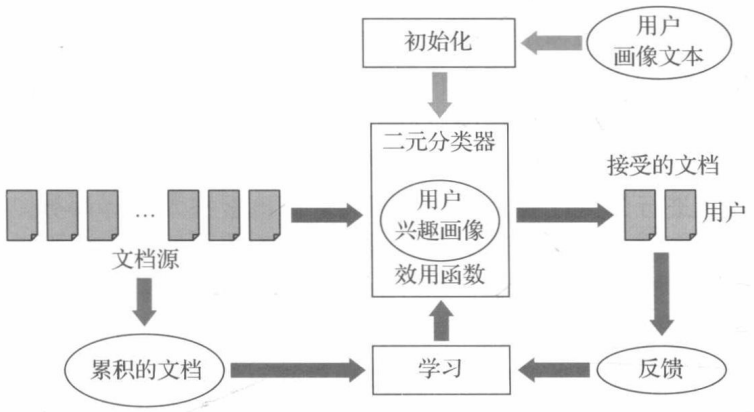


图 11-2　基于内容的信息过滤系统，包含各模块详细内容 223

接下来有一个效用函数，该函数帮助系统设置接受阈值 θ，确定当前项目是否应向用户展示。而学习模块则随着时间的推移根据用户的反馈调整参数。通常在信息过滤的应用中，用户的信息需求是稳定的。正是由于这一点，系统有多种途径可以观测到用户是否查看了推荐给他的项目，即该项目对用户来说是否相关。因此，这种长期反馈能够使系统收集到许多用户兴趣相关的信息，这些信息将用于提高分类器的效果。

那么，我们如何知道这个过滤系统是否表现良好呢？在这种情况下，不能使用诸如 MAP 或 NDCG 之类的排序评价方式，因为系统无法等到积累了足够数量的文档并对文档进行排序后再决定给用户推荐哪些信息，必须实时做出决定。一般来说，这个决定取决于该项目是否高于接受阈值 θ。也就是说，根据绝对的相关性来决定是否接收该项目。最常

⊖ 二元分类器是一种能够接收某个项目作为输入并判断最终属于两个类别中的某一类的算法。在此例中，类别是指相关和不相关。想了解更多与分类相关的知识，请见本书第 15 章。

见的策略之一是使用效用函数，下面是线性效用函数的一个例子：

$$\mathcal{U}=3\cdot|R|-2\cdot|R'| \tag{11-1}$$

其中，R 是推送给用户的相关文档的集合，R' 是推送给用户的不相关文档(用户拒绝的文档)集合。某种意义上讲，这类似一种赌博游戏。如果你提供给用户一个好的项目，则你可以赚到 3 块钱，如果提供给用户一个不好的项目，赔 2 块钱。这个效用函数用于衡量通过这个游戏能够积累或输掉多少钱。显然，如果想最大化这个效用函数，所选的策略必须是尽可能多地给用户提供好的项目，而同时尽量减少坏的项目。

一个有趣的问题是如何设置这些系数。刚刚设置 3 和－2 作为效用函数可能的系数，但是可以思考一下“它们是否合理？是否还有其他选择呢？”例如，可以设置成 10 和－1，或 1 和－10。接下来，考察这些效用函数将如何影响系统的输出。如果使用 10 和－1作为效用函数的系数，发现当系统提供一个好的文档，则给予很大的奖励，而系统提供一个坏的文档，给予很小的惩罚。如此，系统将被鼓励提供更多的文档，因为推送更多的文档可
224 以获得更高的奖励机会。如果使用 1 和－10 作为系数，则是相反的情况：推送了好的文档，却不会得到这么大的奖励，而一旦推送了一个坏的文档，将会造成很大的损失。在这种情况下，系统将不再愿意推送文档，必须绝对确保相关的文档才会推送。简而言之，效用函数必须基于具体的应用偏好设计，对于不同的用户而言可能是不同的。

基于内容过滤的三个基本组件如下。

初始化模块：仅基于用户非常有限的文本描述或非常少的实例使得系统启动。

决策模块：给定文本文档和用户兴趣画像描述，决定是否将文档推送给用户。

学习模块：从有限的用户对推送文档的相关性判断中学习模型参数。(如果不向用户推送文档，永远不会知道用户是否喜欢。)

所有上述模块都需要优化以使效用函数 $\mathcal{U}$ 取得最大值。为了解决这个问题，接下来介绍如何扩展检索系统进行信息过滤。首先，可以重用检索技术进行评分。我们知道如何针对查询对文档进行排序以及如何计算用户文本描述与文档之间的相似度，则可设置一个分数的阈值 θ 进行过滤决策。当分数 $\text{score}(d)>\theta$ 时，则认为文档 d 对于用户而言是相关的，因此系统将该文档推送给用户。当然，还需要从历史记录中学习。为此，可使用像 Rocchio 这样的传统反馈学习技术来帮助提升评分。然而，在不知道如何设定 θ 值的情况下，只能先对它进行初始化，随着系统给用户推送更多的文档，得到更多信息来帮助更新 θ。

图 11-3 展示了基于向量空间模型的推荐系统。文档向量输入基于向量空间模型的搜索引擎评分模块中，其中用户兴趣画像将被视为查询。画像向量可以与文档向量匹配以生

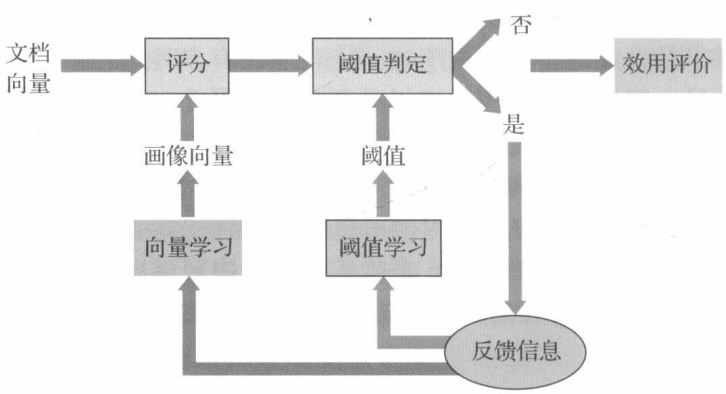


图 11-3　基于内容推荐的三个模块以及架构

成分数。该分数将被输入阈值判定模块，该模块将根据当前的 θ 值来决定是否推送给用户。评价则基于效用函数。如果输出结果为“是”，文档将被发送给用户，然后可以从用户处得到反馈。反馈信息将用于调整阈值并更新向量表示。在这个意义上，向量学习与搜索中的查询修改或反馈基本相同。阈值学习是一个新的组件，接下来将详细介绍。

阈值学习面临一些很有意思的挑战。图 11-4 展示了可以在过滤系统中收集的数据类型，包括得分和相关状态。第一个文档的得分为 36.5 分，是相关文档。第二个是不相关文档。首先，有很多文档我们不知道其相关状态，由于它们的分数小于 θ，没有展示给用户作判断。因此，用户判断过的文档不是随机抽样，是存在偏差的，增加了学习最佳参数 θ 的难度。其次，由于标签数据和相关数据很少，这对于需要大量训练数据的机器学习方法来说是非常有挑战性的。在极端的情况下，根本就没有任何有标签的数据，但是系统仍然要做出决定。

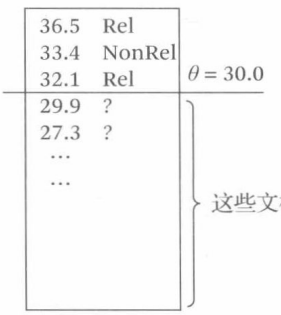


图 11-4　基于内容的推荐系统的可用信息

这个问题被称为探索利用(exploration-exploitation)平衡问题。也就是说，要探索文档空间来查看用户是否对尚未标记的文档感兴趣，但是又不希望给用户推送太多的不相关文档，导致他们对系统不满意。那么应该如何做呢？可以稍微降低阈值，并向用户提供一些接近阈值的文档，考察用户对额外文档的响应。这是一种权衡，一方面希望探索，但另一方面，又不希望探索过多，因为可能导致推送过多不相关的信息。意味着利用学习到的用户信息，如果知道用户对某个特定的主题感兴趣，则围绕用户兴趣进行推荐。当然，如果完全不偏离用户兴趣，那么根本不会探索新的，可能会错过了解用户其他兴趣的机会。显然，这是一个很难解决的问题。

为什么不能使用经验效用优化策略来优化 U 呢？原因在于该策略只适用于根据历史数据进行的阈值优化。也就是说，可以基于训练数据对每个候选分数阈值计算效用函数，记录给定 θ 时的最高效用。这没有考虑到前面提到的探索问题，而且还存在训练样本偏置的问题。一般来说，只能获得真正最优阈值的上限，因为阈值可能低于这个值。可能某些丢弃的项目实际上是用户感兴趣的。如何解决这个问题呢？可以通过适当降低阈值来进行探索。这里介绍一种特殊的阈值学习方法，称为 beta-gamma 阈值学习法[Zhai 等 1998]。

beta-gamma 阈值学习算法的基本思想如下。给定训练集合中经过相关性以及特定效用函数 U 排序的文档列表，根据其在 x 轴上的分数，绘制出每个不同截止位置 θ 处的效用值。每个截止位置对应一个分数阈值。图 11-5 展示了 beta-gamma 阈值学习算法，可见 α 的选择决定最优和零效用点之间的截止点，β 和 γ 根据训练数据中判断实例的数量动态地调整 α。最优点 θ_{opt} 是使得效用最大化的阈值，阈值 θ_{zero} 是零效用阈值。这两个 θ 值之间是允许探索潜在截止值的安全空间。从公式可以看出，阈值是通过插值参数 α 对零效用阈值和最优阈值进行插值得到的。

现在的问题是应该如何设置 α 并偏离最佳效用点。这可以取决于多个因素，一种解决问题的方法是鼓励这种阈值机制只能探索到 θ_{zero} 点(这仍然是一个安全的点)，但不一定探索到达该点的全部值。相反，在给定一些额外信息时使用其他参数来进一步定义 α 值。

参数 β 控制阈值相对于 θ_{opt} 的偏离，可基于之前的文档(即训练数据)观测出。更有趣的是，参数 γ 控制训练数据集合中实例的个数 N 的影响，当 N 变大时，将鼓励更少的探

索。换句话说，当 N 值很小时，算法尝试探索更多，因为如果只观测小部分实例，无法确认是否已经充分考察兴趣空间。但是，如果观察到许多用户的数据点，将不再需要继续探索。这为我们提供了一个动态的探索策略：看到的实例越多，要做的探索更少，所以阈值将更接近于 θ_{opt}。

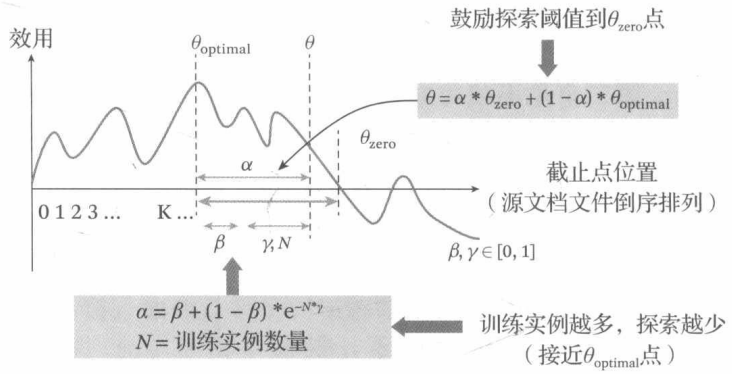


图 11-5　beta-gamma 阈值学习算法设置 θ 的最优值

这种方法在一些实证研究中表现良好，特别是在 TREC 过滤任务上。同时它也很方便，任何具有适当下限的效用函数都可以。该方法处理了探索利用的折中，并采用了 θ_{zero} 作为保障措施。也就是说，永远不会进行比零效用点更进一步的探索。可以将其类比于赌博，不想冒险亏钱，这将是一种“安全”的策略。然而，这种方法是纯启发式的，零效用值作为下界在实践中过于保守。许多先进的机器学习方法被提出来解决这些问题，目前来说，这是一个非常活跃的研究领域。

228

11.2　协同过滤

在协同过滤中，系统根据其他用户的判断来帮助某个用户做决策(因此是“协同的”)。其基本思想是根据相似用户来推断个人兴趣或偏好。给定用户，协同过滤找到一组相似的用户，基于相似用户的集合预测当前用户的偏好。

该方法做了以下假设：

- 具有共同兴趣的用户将具有类似的偏好。
- 具有相似偏好的用户具有相同的兴趣。

例如，如果用户对信息检索感兴趣，他们可能会喜欢 SIGIR 上发表的论文。而如果用户喜欢 SIGIR 论文，那么他们可能对 IR 感兴趣。此时，项目的文本内容变得不再重要。这与上一节形成鲜明对比，上一节我们通过基于内容的过滤来查看项目相似度。

协同过滤中，根据其他相似用户来推断某个用户的兴趣，其大体流程如图 11-6 所示：给定一个用户 u，根据与 u 的相似度对其他用户进行排序，u_1，…，u_m。然后基于这 m 个用户的偏好预测该用户的偏好。这里偏好是建立在公共项目集 o_1，…，o_n 上的。我们把用户和项目排列成矩阵 $\boldsymbol{X}$，则用户 u_i 和项目 o_j 视为矩阵中的点(u_i，o_j)，而该用户对该对象的评分则放置在 X_{ij} 位置上。

再强调一下，此时每个项目的内容已经无关紧要。我们只考虑用户和项目之间的关系，这使得方法的通用性非常强，因为它可以应用于任何项目——而不仅仅是文本文档。项目可以是电影或产品，用户可以给出评分(例如 1 到 5)。有些用户已经看过某电影并对其进行了评分，但是给定用户的大多数电影都是未进行评分的(因为用户不太可能检查了

229

对象空间中的所有项目)。因此，许多项目的用户评分是未知的，而协同过滤的工作是基于其他已知的值来推断该矩阵中未知元素的值。

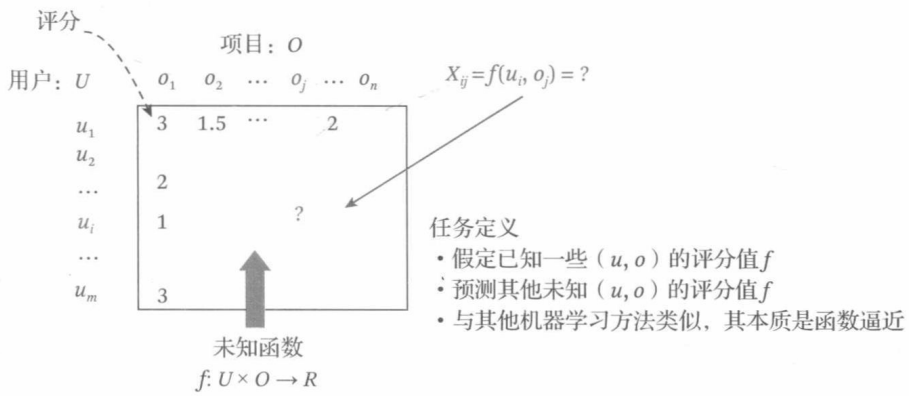


图 11-6　将协同过滤视为具有部分数值的 $m\times n$ 矩阵，其中数值表示用户评分

我们必须做的另一个假设是有足够多的用户偏好数据。例如，我们需要一定数量的电影评分来表明用户对电影的喜好。如果没有足够的数据，就会出现数据稀疏问题，通常被称为冷启动问题(cold start problem)。

假定存在一个未知函数 $f(\cdot, \cdot)$，将用户和对象映射到一个评分。在矩阵 $\boldsymbol{X}$ 中，我们已经观察到这个函数的一些输出值，要推断其他没有值的用户－对象对的函数值。这与其他机器学习问题非常相似，即已知训练数据上的函数值，希望预测那些未知的测试数据的函数值。解决这个问题有许多方法，实际上有一个专门针对这个问题的重要会议。

这里将介绍基于记忆的方法(memory-based approach)。当我们考虑特定用户时，首先尝试检索到当前用户的相关(即相似的)用户。然后，使用这些用户来预测当前用户的偏好。令 n_i 为用户 u_i 对所有对象的平均评分。我们可以通过从用户对所有对象的评分中减去平均评分来进行规范化。这样做的目的是使来自不同用户的评分具有可比性，一 230
些用户可能比较慷慨，通常会给比较高的评分，而其他用户可能更苛刻，平均评分较低。所以，他们的评分不能直接相互比较或聚合在一起，这就是为什么我们首先要进行规范化。

假设 u_a 为我们的目标用户(当前用户)，我们想给用户 u_a 推荐的项目是 o_j，首先要考虑与 u_a 相似的用户是否喜欢 o_j。在数学上，该用户对该对象的预测评分是这些相似用户的规范化评分的组合，但并不是所有用户的贡献都一样；每个用户的权重控制着该用户对最终预测评分的影响。权重与 u_a 和特定用户 u_i 之间的相似度有关。他们之间相似度越大，u_i 在预测 u_a 的评分时起到的作用越大。

我们用如下公式来描述方法。首先，得到规范化的评分：

$$V_{ij}=X_{ij}-n_i \tag{11-2}$$

则待预测的规范化评分可以写成：

$$\hat{V}_{aj}=k\cdot\sum_{i=1}^{m}w(u_a,u_i)\cdot V_{ij} \tag{11-3}$$

其中 $w(\cdot, \cdot)$是相似度函数，k 是归一化因子：

$$k=\frac{1}{\sum_{i=1}^{m}w(u_a,u_i)} \tag{11-4}$$

k 保证了 $\hat{V}_{aj} \in [0, 1]$。有了待预测的规范化评分，我们可利用下述公式将其转换成 u_a 的评分范围：

$$\hat{X}_{aj} = \hat{V}_{aj} + n_a \tag{11-5}$$

此时，如果想编写一个程序来实现这种协同过滤，仍然面临如何确定权重函数的问题。一旦我们知道这一点，那么参照上述公式很容易实现。加权函数定义了对协同过滤评分估计的不同解释。有很多种可行的相似度计算函数，其中一种比较流行的是皮尔森相关系数：

$$w_p(u_a, u_i) = \frac{\sum_j (X_{aj} - n_a)(X_{ij} - n_i)}{\sqrt{\sum_j (X_{aj} - n_a)^2 \sum_j (X_{ij} - n_i)^2}}$$

231

这是一个对所有共同评价过的项目的求和，它衡量了两个用户是否对相似的项目的评分的趋势相同，都给予高分或都给予低分。另一个方法是余弦相似度，将评分向量作为向量空间中的向量，并测量两个向量之间的夹角的余弦：

$$w_c(u_a, u_i) = \frac{\sum_j x_{aj} x_{ij}}{\sqrt{\sum_j x_{aj}^2 \sum_j x_{ij}^2}}$$

正如本书之前讨论过的，这种方法已被用于向量空间检索模型。我们还将在第 14 章看到它如何被用于聚类。

在所有上述情况下，用户的相似度是基于他们对项目的偏好，实际上并没有使用这些项目的任何内容信息。这些项目是什么都没有关系，它们可以是电影、书籍、产品或文本文档。这使得这种方法能够被广泛应用到很多问题上。

有一些针对这种方法的改进措施，大多数都关注用户相似度的度量。其中有一些实际的问题要处理，例如缺失值问题。可以将其设置为默认值或其他用户的平均评分。这是一个最简单的解决方案。但是一些更加先进的方法尝试预测那些缺失值，使用预测的值改进相似性度量。基于记忆的协同过滤方法就可以预测缺失值，使用迭代的方法，首先做一些初步预测，然后使用预测值进一步提高相似度函数。

另一个与文本研究中提到的 IDF 思想非常相似的方法被称为*逆用户频率*(Inverse User Frequency，IUF)。其核心思想是计算用户相似度时，着重考察他们评分相似的项目。如果是被许多用户观看过的非常流行的项目，其提供的信息性是有限的。相反，如果是罕见的没有被许多用户观看的项目，能够更加凸显用户的相似性。因此，这里强调了两个用户在没有被许多用户观看的项目上的评分相似性。

下面对推荐系统做一个总结。推荐就任务而言，有时很容易，有时则很困难。容易是因为用户的期望低，也就是说，任何建议都比没有建议要好。系统主动将信息推送给用户，用户无需耗费任何精力。除非系统推荐的都是噪声或无用的项目，否则任何信息都是

232

有益的。因此在这个意义上，推荐是相对容易的。

推荐同时也是非常艰难的任务，因为必须及时做出推荐与否的决定，而不能等待通过与其他项目的对比得出哪个更好。一旦新闻过滤系统检测到新闻文章，此时系统必须决定用户对这个消息是否感兴趣。如果过了几天，即使是准确地推荐了最相关的消息给用户，用户也不再感兴趣。另一个原因是数据稀疏。例如，如果把推荐单纯地视为协同过滤中的学习问题，那么只能基于过去的评分进行学习。假如没有足够的历史评分数据，将很难进

行。前面提到针对这些问题有许多解决策略，比如，利用更多的用户信息来评估其相似度，而不是仅仅使用项目偏好。

前面介绍过推荐任务的两个策略。一个是基于内容的推荐，主要考察项目内容相似度。另一个是考察用户的相似度，也就是协同过滤。同时，前文还提到过访问文本数据的两种策略“推送”和“拉取”。推荐系统以推送的模式帮助用户，而搜索引擎则以拉取的模式服务用户。

11.3　推荐系统的评价

在协同过滤的评价设置中，存在一组预测分数$\hat{r}$和实际评分r构成的配对的集合P，其中每一个配对都对应于一个用户-项目配对。最常见的评价方式为均方根误差(RMSE)，公式如下：

$$\mathrm{RMSE}(P)=\sqrt{\frac{1}{|P|}\sum_{(\hat{r},r)\in P}(\hat{r},r)^2} \tag{11-6}$$

与之相似的是平均绝对误差(MAE)，定义为

$$\mathrm{MAE}(P)=\sqrt{\frac{1}{|P|}\sum_{(\hat{r},r)\in P}|\hat{r},r|} \tag{11-7}$$

由于 RMSE 中存在平方，RMSE 更容易受较大误差的影响。RMSE 和 MAE 之间的差别与 gMAP 和 MAP 之间十分相似。RMSE 和 MAE 都能量化$\hat{r}$与真实评分r之间的差值。这两种方法适用于评分是等级数据时的情况。

一个比较重要的问题是 RMSE 或 MAE 都只能捕捉预测分数的精确率。在实际推荐系统中，通常向用户推荐前k个项目。尽管 RMSE 或 MAE 很低，但排在前面的文档可能与用户并不相关。在$k=1$的极端情况下，推荐的可能是误判为高分的项目。为了解决这个问题，可以对推荐给特定用户的所有项目按照评分进行排名，将它们的真实得分作为相关性判断，然后使用诸如 NDCG 之类的信息检索指标对整个列表进行评价。 233

信息过滤任务中，如果系统认为用户对该项目感兴趣，则把项目推送给用户。在这种情况下，系统对每个项目没有明确的评分，而是直接做出相关或不相关的判断。一旦用户查看该项目，便可决定该推荐是否合理。对于单个用户，可以使用第 9 章所述的信息检索中的一些评价指标进行评价。

由于项目在到达之后立即被推送给用户，因此不能使用任何排名敏感的评价指标。但是，可以检查在一段时间内推送给用户的所有文档的集合，并计算准确率、召回率、F_1分数和其他类似指标。

在真实的信息过滤系统中，有大量用户接收被推送的项目。一种简单的评分汇总方法是对每个用户的指标取平均值，例如所有用户的平均 F_1 值。但是有些用户比其他用户有更多的推荐，因此这种平均的方式就不是最好的方案。因此为每个用户设置单独的θ，不同用户将聚合不同数量的已查看文档。此外，在真实系统中，用户不是同时加入，有些用户拥有更完整的用户兴趣画像或者加入系统的时间早于其他用户，则可能拥有比其他用户更多的训练数据。基于上述原因，比较好的方法是将用户的指标加权平均起来。所有用户的权重之和为 1，同时每个用户的权重由其总的评价数来决定。

更有趣的是随着时间计算准确率或召回率，其中时间可通过过滤系统看到的文档数量来度量，或者是基于用户判断过的项目数量(仅考虑向用户显示的元素)。理想情况下，随着文档数量的增加，整体准确率(或最近推荐的k个文档的准确率)会随之增加。一旦准确

率达到了稳定值，则意味着系统在当前的设置下很可能达到了 θ_{opt}。当然，这种随时间学习的评价方式也可以像前面讨论的一样，推广到多个用户。

关于两种推荐系统类型的最后一个建议，最有价值的是找到那些最能影响评价指标的
234 用户。也就是说，是否有任何异常导致评价指标显著低于预期？如果是，则应该重点考虑那些用户，调整推荐系统来满足这些用户。当然，这些异常值在很大程度上取决于所使用的评价指标，因此使用多个评价指标可以最全面地了解用户满意度。

书目说明和延伸阅读

作为进一步阅读，建议读者参考 Herlocker 等[2004]、Shani 和 Gunawardana[2011]的文献。Ricci 等[2010]是一个较为综合的资源，用于更多地了解推荐系统。关于 Beta-gamma 阈值设置算法的更多信息可以在 Zhai 等[2000]中看到。Breese 等[1998]提供了基于记忆的协同过滤算法的描述，也对不同的协同过滤算法进行了比较。而最近的关于更多协同过滤算法的比较可参看 Cacheda 等人[2011]的文献。

练习

11.1　向用户推送内容时，一个重要原则是不推送重复信息。请描述一种策略，使其满足如果某文档与以前推送过的文档是重复的，则不会向用户传递该文档；请描述一种策略，如果某文档与以前推送过的文档相似，则不会推送该文档。注意确保方法节省空间，不要把文档的所有内容都存储起来。

11.2　用户可能对不同方面的信息需求感兴趣，比如用户既可以喜欢浪漫电影也可以喜欢动作电影。这会给推荐系统带来问题吗？如果是，请给出建议如何解决这个问题。如果不是，请解释推荐系统的现有结构将如何处理此问题。

11.3　在这一章的引言中提到将基于内容过滤和基于协同过滤的方法结合能够取得最佳的推荐结果。请就这两种方法如何结合提出你的看法并解释原因。

11.4　beta-gamma 阈值法将搜索引擎得分与截止参数 θ 进行比较。“如果使用单一固定的 θ 则查询得分不具有可比性，原因是分数没有归一化”，对于此说法，请表明立场，支持或反对并说明原因。

235 11.5　利用 MeTA 搜索引擎实现 beta-gamma 阈值法。利用具有相关性判断的数据集作为用户偏好，把少量相关查询作为用户初始兴趣画像。把文档 ID 打乱，模拟数据流对其顺序地检验。

11.6　本章描述了线性效用函数 $\mathcal{U}$，用于对过滤系统输出的有用性进行评分。请举一个非线性效用函数的例子。与本章介绍的版本相比有什么区别？

11.7　在对 beta-gamma 阈值法的描述中，如果 $\text{score}(d)>\theta$，则向用户显示一个文档。你能想到一个替代的得分策略吗？当文档或用户特征可用时，策略可以包含有关文档或用户的其他特征。如果可能，在 MeTA 中评价这个备用的评分函数。

11.8　给定 n 个用户和 m 个对象，确定使用协同过滤向每个用户推荐一个项目的运行时间。首先假定使用余弦相似度，记住其运行时间，那么如果改用皮尔森相关系数会改变运行时间吗？

11.9　作为提高协同过滤运行时间的一种方法，只考虑最活跃的 k 个用户(即给予大量评分的用户)，其中 $k \ll n$。这种方法有什么潜在的问题？

11.10　试想如果在电影数据库中存在一个协同过滤系统。某个电影制片人在协同过滤系

统上创建了许多账户，并且只对自己公司制作的电影评级较高，而给所有其他的电影评级较低。本章描述的协同过滤系统是否容易受到这种垃圾评分的影响？如果是这样，请集中讨论一些反作弊的措施。

11.11　协同过滤系统单独地处理每一个项目，即使它们几乎相同。例如，三部曲中的每个电影都被单独对待。举例说明这样做的优点和缺点。

11.12　当新元素添加到协同过滤数据集(例如，新书的发布)时，需要采取哪些处理步骤？仍存在冷启动问题吗？

11.13　RMSE、MAE 同 L_1、L_2 误差函数之间有关系吗？注意 L_1 误差函数的公式为 $\sum_{i=1}^{n}|y_i-f(x_i)|$，$L_2$ 误差函数的公式为 $\sum_{i=1}^{n}(y_i-f(x_i))^2$，其中 y_i 代表着预测的真实值，$f(x_i)$表示预测值，n 是预测总数。

236
~
238

第三部分

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本数据分析

第 12 章　文本数据分析概述
第 13 章　词关联挖掘
第 14 章　文本聚类
第 15 章　文本分类
第 16 章　文本摘要
第 17 章　主题分析
第 18 章　观点挖掘与情感分析
第 19 章　文本与结构化数据的联合分析

第 12 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本数据分析概述

第二部分介绍了文本数据获取技术，逻辑上这是处理文本数据的初始步骤，目的在于显著减少需要(由人类或机器)进一步处理的数据集的大小，并且去除文本数据中明显的噪声，以便将重点放在与特定应用问题真正相关的数据上。本书的第三部分将介绍进一步处理相关文本数据的技术，以便提取和发现可直接用于决策或支持用户任务的有用的、可操作的知识。第二部分和第三部分的一个区别是强调用户信息需求的程度，或者等价地说是特定应用的信息需求。具体来说，由于文本数据获取的目的通常是将用户在正确的时间与正确的信息连接在一起，以便他们可以进一步整理和利用相关的文本数据(在有或没有附加的文本分析技术的帮助下)，信息需求的概念以及与之紧密联系的相关性的概念在第二部分所涵盖的所有技术中都起着重要的作用。例如，查询在任何搜索引擎中都起着至关重要的作用，用户兴趣和信息需求的准确建模在任何推荐系统中都起着同样重要的作用。然而，在第三部分中，通常假设要考虑的文本数据都是相关的，所以我们将不再过多地强调信息需求，而是强调理解文本中更多的内容细节，并在文本数据中找到各种有趣的模式，例如主题趋势或情感极性，以便最终提取和发现直接用于完成用户任务的可操作的知识。因此，我们尝试将文本数据分析视为数据挖掘一般过程的特殊情况，用户可以使用各种数据挖掘操作及它们的组合以交互的方式来探查和分析数据。具体来说，我们将人类视为世界的“主观传感器”，那么文本数据可视为由这些主观传感器生成的数据。于是，文本数据看起来与客观机器传感器产生的其他类型的数据更加接近，并使我们能够自然地讨论如何联合地分析文本和非文本数据。

然而，必须指出，将文本数据获取阶段和文本数据分析阶段分离开(从而形成了本书中的第二部分和第三部分)是人为的。因为在复杂的应用中，这两个阶段经常是交错的，并且经常采用包括两个阶段的迭代过程。例如，在将范围缩小到一组相关文档并且执行分析任务(例如将文档聚类成主题簇)之后，用户还可以选择在特定簇内做进一步搜索从而将范围缩小到文档集合的特定子集，之后其他分析操作(如情感分析)可以应用于这个新获得的较小子集。此外，第二部分和第三部分的技术通常可以结合起来，为用户提供更多有用的功能(例如，摘要技术可以自然地与搜索引擎或推荐系统结合)，并且它们可以彼此促进(例如，第二部分中讨论的词项赋权方法对于许多任务如第三部分中的聚类、分类和摘要也非常有用，第三部分中的聚类可用于改进第二部分所述的检索算法)。尽管如此，我们选择将它们分开，以便让读者了解我们所涵盖的所有技术的总体情况以及它们之间高层次的联系。

12.1 动机：文本数据分析的应用

从我们每天都处理大量文本数据的事实可以很容易地看出文本数据对生活的重要性。然而，在大多数情况下，计算机只在使用文本数据的整个过程中发挥很小的作用。例如，我们经常使用搜索引擎，但一旦找到相关文档，对找到的文档的进一步处理往往仍然利用人工。当要处理的文本数据量小，应用任务不要求快速响应，以及我们有时间消化理解文

本数据时，这样的人工过程是可以接受的。然而，随着文本数据量的增加，人工处理文本数据将不可行或不可接受，特别是面向对反馈时间要求很高的应用程序时。因此，开发高级文本分析工具来帮助我们高效地消化和使用文本数据变得越来越重要。

一般来说，可以将文本分析应用分为两类。一类可以取代目前的人工来消化文本内容，它们有助于提高生产力，但不能超越人类做人类无法做到的事情。例如，电子邮件的 242
自动排序将使我们节省大量的时间。另一类是那些可以发现人们即使有“足够”的时间阅读所有的文本数据也无法发现的知识。例如，智能生物医学文献分析系统可以通过综合分散在许多不同研究文献中的基因-基因关系和基因-疾病关系来揭示基因和疾病的关联链，从而为研发针对特定基因的药物来治疗疾病提供潜在的机会。

由于文本数据的知识覆盖面广，同时人们又依赖文本数据进行通信，所以可以想象文本分析几乎可以在任何领域运用。下面是一些具体的例子，它们可以为理解后面章节所涉及的文本分析技术提供一些应用背景。

文本分析的一个重要应用领域是商业智能。例如，产品经理可能非常愿意听取客户关于他们的产品的反馈意见，了解他们的产品相对于竞争对手产品的优势。这是利用 Web 上商品评论形式文本数据的好机会。如果能够开发和掌握文本挖掘技术，利用这些信息源来获取人们对这些产品的知识和意见，我们就可以帮助产品经理获得商业情报，或者从他们的客户那里获得反馈。

另一个重要的应用领域是科学研究，及时整理文学文献中蕴含的知识至关重要。科学家们也有兴趣了解研究课题的发展趋势或了解与自己相关的领域的最新发现。这个问题在生物学研究中尤其重要——不同的研究团体倾向于使用不同的术语，但他们描述的是非常类似的问题。我们如何才能整合不同研究团体所涵盖的知识(使用不同的词汇)来帮助研究特定的问题？回答这个问题能够加快科学发现。还有更多这样的例子，可以利用文本数据来发现可用的知识以优化决策过程。

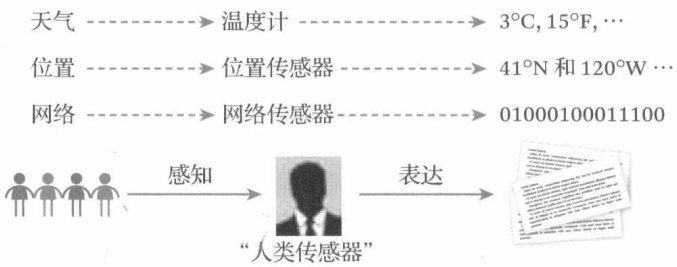
另一大类应用是利用社交媒体来优化决策。总的来说，我们可以设想建立一个智能传感器系统来“监听”实时生成的所有文本数据，特别是诸如微博(tweet)之类的实时报告真实事件的社交媒体数据，并监视与应用相关的有趣模式。例如，针对人们对政策的看法进行情感分析可以帮助更好地了解社会对政策的反应，从而有可能在需要时改进政策。灾难响应和管理将有助于及早发现自然灾害的任何警告信号，这可以是通过实时分析微博数 243
据来实现的。

一般来说，“大数据”可以增强我们的感知能力。就像显微镜一样，我们可以看到“微观世界”中的东西，而望远镜可以让我们看到远处的东西，在大数据时代，我们可以设想一个“数据观测器”，它将允许我们“看到”埋藏在大量数据中的有用的隐藏知识。作为一种特殊的数据，文本数据提供了独特的机会来帮助我们“看到”文本中蕴含的各种知识，特别是有关人们的观点和想法的知识，这在其他类型的数据中可能并不容易观察到。

12.2　文本与非文本数据：人类作为主观传感器

为了进行数据挖掘，将文本数据视为人类作为主观传感器产生的数据是有用的。我们可以将人类作为主观传感器与物理传感器(如网络传感器或温度计)进行比较。任何传感器以某种方式监测现实世界。它感测来自现实世界的一些信号，然后将信号报告为各种形式的数据。例如，温度计将监测真实世界的温度，然后将温度报告为华氏或摄氏度的数据。类似地，地理传感器将监测到地理位置，然后将其报告为 GPS 坐标。有趣的是，我们也

可以将人类视为从自己的角度观察现实世界的主观传感器。人类以文本数据的形式表达了他们观察到的内容。从这个意义上说，人类实际上是监测世界上正在发生的事情的主观传感器，然后用数据文本的形式表达所观察到的东西。这个想法如图 12-1 所示。



244

图 12-1　人类作为主观传感器

从这个角度看待文本数据的好处是可以将所有类型的数据集成在一起，这几乎适用于所有的数据挖掘问题。在数据挖掘场景中，我们将处理与特定问题相关的现实世界的数据。大多数问题既处理非文本数据也处理文本数据。当然，非文本数据通常由物理传感器产生，并且可以以许多不同的格式存在，例如数值、类别或关系，甚至可以是视频或语音等多媒体数据。文本数据也非常重要，因为它们包含有关用户的知识，特别是偏好和意见。

通过将文本数据视为从人类传感器观察到的数据，我们可以在同一框架中考察所有这些数据。然后可以将数据挖掘问题定义为将所有这些数据转化为可操作的知识，我们可以利用这些知识来改变世界。如图 12-2 所示。

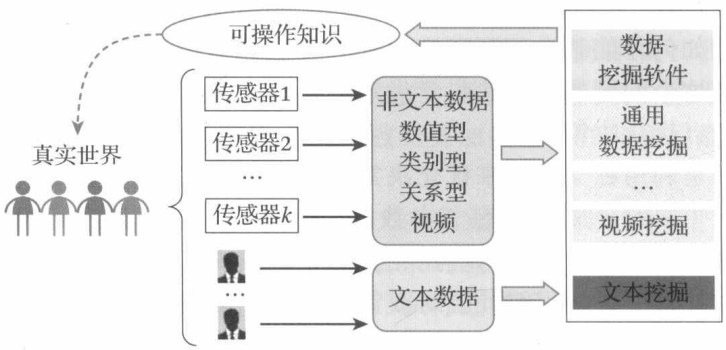


图 12-2　数据挖掘的一般问题

可以看到，在数据挖掘模块的内部有许多不同类型的挖掘算法。当然，对于不同类型的数据，通常需要不同的算法，每种算法都适用于挖掘特定类型的数据。例如，视频数据需要计算机视觉来理解视频内容，这将有助于更有效的挖掘。我们还有许多适用于各种数据的通用算法。这些算法当然是非常有用的，但是对于特定类型的数据，为了获得最佳的挖掘结果，通常仍然需要开发专门的算法。本书的这一部分将涵盖对挖掘和分析文本数据
245 特别有用的专门算法。

在图 12-3 中进一步看文本挖掘问题，该问题与一般数据挖掘相似，但将更多地关注文本数据。我们需要文本挖掘算法来帮助我们将文本数据转化为可在现实世界中使用的可操作知识，特别是用于决策或完成任何需要文本数据支持的任务。许多真实的数据挖掘问题也可能包括非文本的其他类型的数据。因此，在更宏观的视图中应该包括非文本数据。那时，我们可能会关注文本和非文本数据的联合挖掘。通过以上问题定义，我们现在可以概括地看一下文本挖掘和分析涉及的主题。

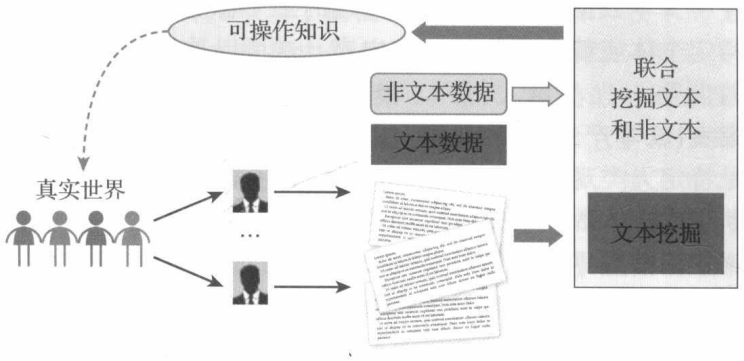


图 12-3　文本挖掘作为数据挖掘的特例

12.3　文本挖掘任务概览

本节将提供各种文本挖掘任务概况的高层次描述，这也是第三部分后续章节涵盖的主题的路线图。

图 12-4 更详细地解释了生成文本数据的过程。具体来说，人类传感器或人类观察者将从一些角度来看世界。不同的人视角也不同，他们会注意不同的事情。同一个人在不同时期也会关注观察到的世界的不同方面。然后，每个人——一个传感器——将形成自己的世界观。它或许与真实世界有所不同，因为人们看问题的角度往往存有偏差。观察到的 246
世界可以表示为如实体-关系图或使用某种知识表示语言。这基本上是一个人对世界的看法。

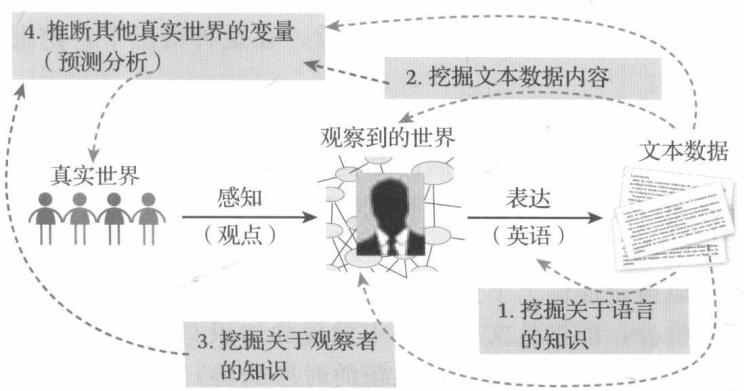


图 12-4　从文本数据中挖掘不同类型的知识

作为人类生成数据的用户，我们永远不会确切地知道作者在观察时真实的世界是什么样子。人类使用自然语言(如英语)来表达观察结果：其结果是文本数据。在某些情况下，我们可能会拥有不同语言或不同语言混杂的文本数据。

文本挖掘的主要目标是逆向文本数据生成过程，并发现人类传感器观察到的关于真实世界的各种知识。如图 12-4 所示，我们可以区分 4 种类型的文本挖掘任务。

挖掘关于自然语言的知识　由于所观察到的文本是用特定的语言编写的，因此通过挖掘文本数据，我们可以挖掘自然语言本身的使用知识。例如，如果文字用英文写成，我们可能会发现有关英语的知识，例如用法、搭配、同义词和口语用词。

挖掘关于观察到的世界的知识　这与挖掘文本数据的内容有关，重点是提取文本数据

中的主要陈述，并将文本数据转换为我们感兴趣的某个特定方面的高质量信息。例如，我
247 们可以发掘关于特定个体或特定实体的所有内容。这可以被认为是通过挖掘内容来描述作者头脑中观察到的世界。

挖掘关于观察者(文本产生者)的知识　由于人类是主观传感器，人类所表达的文本数据通常包含可能是特定人类观察者(文本产生者)所独有的主观言论和意见。因此，可以挖掘文本数据来推断生成文本数据的作者的一些属性，例如人们对问题的情绪或情感。请注意，我们将挖掘观察到的世界的知识与挖掘关于文本产生者的知识区分开来，因为文本数据通常混合了关于所观察世界的客观陈述以及反映文本产生者的观点和信念的主观陈述或评论，因此对两者单独提取是可能且有用的。

推测关于现实世界的属性的知识　图的左侧展示了文本挖掘还可以通过利用这些变量的值和文本数据中的内容的相关性来推断感兴趣的现实世界变量的值。例如，股票市场股票价格变动与新闻数据中报告的事件之间可能存在一定的相关性(例如，公司的盈利报告可能与公司股价涨幅相关)。这种相关性可以用来进行基于文本的预测，我们使用文本数据作为预测其他变量的基础，而这些变量可能与文本数据(例如，股票价格的预测)相关性很小。关于影响决策的未知因素的推断可以有很多应用，特别是对未来事件做出预测(即基于文本的预测分析)。

请注意，当我们推断其他的真实世界变量时，利用各种文本挖掘算法的结果来生成用于预测模型的特征，这些特征可能比从原始文本数据中直接生成的基本特征更有效。例如，如果可以挖掘文本数据来发现主题，我们将能够使用主题(即一组语义相关的单词)而不是单个单词作为特征。讨论主题可以解决词义歧义和词汇使用习惯的问题，所以这种高
248 级语义特征可能在预测中比词级特征更有效。另一个例子是基于用户在文本数据(例如评论)中所说的内容来预测什么产品可能被用户喜欢，在这种情况下，挖掘关于观察者的知识显然对预测非常有用。

此外，非文本数据在预测分析中可能非常重要。例如，如果你想预测股票价格或股票价格变动，历史股价数据可能是用于预测的最佳数据，即使在线讨论、新闻文章或社交媒体也可能通过提供基于文本数据的额外的有效特征(可与非文本特征相结合)来进一步提升股票价格的预测精度。

非文本数据也可以通过提供上下文来分析文本，从而开辟了从文本数据中挖掘对上下文敏感的知识的有趣机会，即将从文本数据中发现的知识与非文本数据相关联(例如，将从文本中发现的主题与时间关联能够产生主题的时序趋势)。当我们孤立地看待文本数据时，主要关注文本中所表达的内容或观点。然而，文本数据通常也具有与其相关联的上下文。例如，文本数据产生的时间和位置都是文本文档的有用“元数据”值。这个上下文可以为分析文本数据提供有趣的角度；由于时间的可用性，我们可能会将文本数据分割成不同的时间段，分析每个时间段的文本数据并进行比较。类似地，我们可以根据位置或与之相关联的任何其他元数据对文本数据进行分区，从而在这些区域中形成有趣的比较。在这个意义上，非文本数据可以为文本数据分析提供有趣的角度或观点。它可以帮助我们对内容、语言使用或者文本数据的观察者或作者的观点进行上下文相关的分析。我们将在第19章详细讨论文本和非文本数据的联合分析。

以上对文本挖掘和分析中的主题提供了一个相当概括的介绍。本书将选择性地介绍不同类型的文本挖掘任务的一些主题。第2章和第3章已经涵盖了自然语言处理和机器学习的基础知识，使我们能够理解、表示和分类文本数据，这是任何文本挖掘任务中的重要步

骤。本书第三部分的其余章节将列举基于前面讨论的 NLP 和 IR 技术的文本挖掘任务。

首先，我们将讨论如何从文本数据中挖掘词关联(第 13 章)，揭示关于语言的词汇知识。在关联挖掘之后，将看到文本对象聚类(第 14 章)，将类似的对象聚在一起，允许在许多其他应用中进行探索性分析。第 15 章涵盖了文本分类，它扩展了第 2 章对机器学习的基本介绍。我们还将探讨不同的文本摘要的方法(第 16 章)。接下来，继续讨论主题挖掘与分析(第 17 章)。这只是分析文本内容的一种方法，但它非常有用并用在各种应用中。第 18 章将介绍观点挖掘和情感分析，这可以被视为挖掘关于观察者的知识的一个例子。最后，简要讨论基于文本的预测问题，尝试基于文本数据预测一些现实世界的变量，以及展示如何对文本和非文本数据进行联合分析的一些前沿研究结果(第 19 章)。

249
~
250

第 13 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

词关联挖掘

本章将探讨如何从文本中挖掘词的关联，词关联挖掘是从文本中挖掘自然语言知识的一个实例。本章首先介绍什么是词关联，然后解释为什么要挖掘这种关系，最后介绍并讨论一些词关联挖掘的常见方法。

通常来说，词语之间的关系有两种，聚合关系和组合关系。如果词 w_a 和词 w_b 可以互相替换，则它们具有聚合关系。也就是说，两个词如果含有聚合关系，则在相同的语义或句法类别中可以利用其中一个来替换另一个而不影响对整个句子的理解。在第 14 章中我们将会给出关于如何将相似词项组合在一起的方法。

例如，cat 和 dog 这两个单词有聚合关系，因为它们在同一词类中：动物。也就是说，在一句话中若用 cat 来代替 dog，句子仍然是(大部分)可以理解的。同样，Monday 和 Tuesday 这两个词也是典型的聚合关系。

第二种关系称为组合关系。在这种情况下，有这种关系的两个词可以互相结合。如果 w_a 和 w_b 在一个符合语法的句子里可以相互结合，则这两个词具有组合关系，即这两个词语义相关。例如，词语 cat 和 sit 是相关的，因为猫可以坐在某个地方(通常在它们喜欢的任何地方)。同样，car 和 drive 在语义上是相关的，因为它们可以相互结合来传达某些含义。但是在句子中不能用词语 cat 来替代 sit 或是 car 来替代 drive，而保持语义不变。因此，前面的词语对是组合关系而不是聚合关系。

实际上，这两个关系是非常基本的，它们可以被推广到捕捉任意序列中单位之间的基本关系。它们可以被概括为描述语言中任意项的关系；也就是说，w_a 和 w_b 不必须是词，也可能是短语或实体。如果考虑序列挖掘这种一般性问题，那么任何单位都可以是单词。聚合关系可以看作是在句子中具有相似位置(通常是一系列数据元素)的单元之间的关系。组合关系捕捉在同一序列中经常共同出现的元素。因此，这两种关系的度量都可以从文本数据中自动挖掘出来。

挖掘这样的词语关系有很多应用。首先，由于这些关系捕获了一些关于语言的知识，因此可以直接用于提高许多 NLP 任务的准确率。例如，如果已知两个词是同义词，那么这对许多不同的任务都有帮助。语法学习也可以通过使用这种技术来完成；如果能够学习到词语的聚合关系，就可以构建词语的分类。如果学习到词语的组合关系，则能够通过学习句子结构，掌握基于成分表达式构成较大表达式的规律。词汇关系对于文本检索和挖掘中的许多应用也是非常有用的。

在搜索和文本检索中，可以使用词关联来修改查询以获得反馈，从而使搜索更有效，这就是第 7 章中提到的查询扩展。还可以利用相关联的词给用户提供相关的查询，以探索信息空间。另一个应用是使用单词关联自动构造层次结构以供浏览。可以将单词作为节点，关联作为边，允许用户从一个词导航到另一个词查找信息。最后，这种词关联也可以用来比较或总结观点。如果想了解关于某个新款智能手机的正面和负面评价，可以观察在正面或负面评价中有哪些词与某个功能词最密切相关，例如电池。这样的组合关系将有助于展示对某产品的具体评价。

13.1　词关联挖掘的基本思想

那么，怎样才能自动发现这种关联呢？首先来看看聚合关系。在这里，最基本的可以利用类似的上下文。图 13-1 用 dog 和 cat 这两个词做了一个简单的例子。 252



图 13-1　聚合关系发现示例

一般来说，当看到这两个词发生在相似的上下文中，那么它们是聚合关系。在图的右侧，从这个小样本的文本数据中提取 cat 和 dog 的上下文。可以从不同的视角来看待上下文。例如，首先可以查看上下文左边的单词。也就是说，考察 cat 和 dog 之前，出现了什么词？显然，这两个词的上下文相似。同理，如果考察 cat 和 dog(右边的上下文)之后出现的单词，我们会发现它们在这个例子中也非常相似。总的来说，有许多其他单词可以与猫和狗一同出现。甚至考察整个上下文，包括这个句子或这个词周围的句子中的所有单词。即使是全部上下文，这两个词之间也有相似之处。因此，考察上下文是发现词聚合关系的常用方法。

考虑以下问题，cat 和 dog 的上下文有多相似？相比之下，cat 和 computer 的上下文又有多相似呢？直观上，cat 和 dog 的上下文将比 cat 和 computer 的上下文更相似。这意味着，在第一种情况下，相似度值是高的，在第二种情况下，相似度很低。这就是捕获聚合关系的基本思想。

对于组合关系，根据组合关系的定义，将探索相关出现的事件。图 13-2 显示了与前例相同的文本示例。在这里，最感兴趣的是知道哪些单词与动词 eats 相关。在图中的右边，已经拿掉了这两个词。思考 eats 的左边会出现什么词？eats 的右边又会出现什么词？ 253
这个问题转换为是否存在与 eats 共现的词。例如，已知 eats 是否出现在句子中通常会帮助预测 meat 是否也出现。换句话说，如果看到 eats 在句子中出现，这将增加 meat 也会出现的概率。

相反，看图中底部提出的问题，eats 的出现对预测 text 的发生有多大帮助？因为 eats 和 text 没有真正的关系，已知 eats 是否出现在句子中并不能真正帮助预测 text 是否也出现在句子中。本质上，我们需要捕捉两个词出现的相关性。

总之，聚合关系通过上下文来考虑每个词，即计算上下文的相似度。假设词之间如果具有较高的语境相似度，则拥有较高的聚合关系。对于组合关系，则计算两个词一起出现在上下文中的次数，这里的上下文可以是一个句子，一个段落，甚至一个文档。比较它们共同出现的次数与它们各自出现的次数。假设两个词之间具有很高的共现但相对较低的单独出现次数，则这两个词将存在组合关系，因为它们往往同时出现，而不会单独出现。



图 13-2　组合关系发现示例

注意，聚合关系和组合关系是十分相关的，因为具有聚合关系的词通常对同一个词具有组合关系。这一事实表明，可以对这两种关系进行联合发现。

254

13.2　聚合关系的发现

根据定义，如果两个词聚合相关，则它们有相似的上下文信息。由此，可以考察每个词的上下文，然后计算它们之间的相似度来发现这种关系。

在图 13-3 中，我们把 cat 这个词从上下文中除去。而包含词 cat 的句子中的其他词往往是与它一起出现的词。同样，可以对词 dog 做一样的处理。一般来说，先获得这样的上下文信息，然后尝试评估 cat 以及 dog 的上下文的相似度。

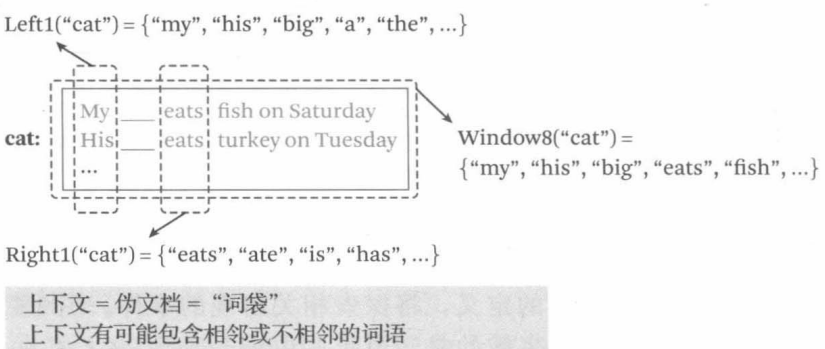


图 13-3　词语上下文传达语义

那么，问题是如何规范地表示上下文并定义上下文之间的相似度函数。首先，上下文中通常包含许多词。这些词可以被看作是一个伪文档，当然也有其他不同看待上下文的方法。例如，考察在单词 cat 之前出现的词，称这种上下文为左上下文 L。此例中，可以看到 my、his、big、a、the 等词。同样，也可以收集词 cat 之后出现的词，这就是所谓的右上下文 R。在这里，有词 eats、ate、is 和 has。更一般地说，可以在目标词周围的文本窗口中查看所有单词。例如，可以将窗口大小设置为目标词周围的 8 个词。

这些单词上下文从左起或从右起组成了基于词袋的表示。这种基于词袋的表示实际上给我们提供了一种定义上下文相似度的有效方法。例如，只比较 L 上下文、R 上下文，或者两者都比较。上下文可能包含诸如 eats 和 my 的相邻词或 Saturday 和 Tuesday 这样的非相邻词。这种灵活性使得能够以不同的方式来匹配相似度。如果基于一般内容捕获相似度，将产生松散相关的聚合关系。如果只使用左右两个单词，将捕获与它们的句法类型密

255

切相关的词。因此，发现聚合关系的一般思想是计算两个词的上下文相似度。例如，可以根据 cat 和 dog 的上下文相似度来度量这两个词的相似度，如图 13-4 所示。

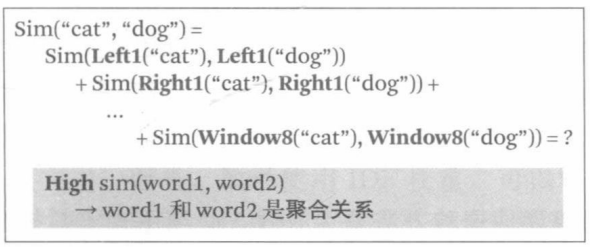


图 13-4　词的多角度上下文可用于计算相似度

相似度函数可以是不同上下文相似度的组合，例如将权重分配给这些不同的相似度，从而能够更专注于特定类型的上下文。当然，这是应用相关的，但发现聚合关系的词的主要思想是计算它们的上下文相似度。

接下来介绍如何精确计算这些相似度函数。毫无疑问，可以利用基于上下文词袋的向量空间模型对一个词的上下文进行建模，进而用于聚合关系发现。一般来说，可以将 cat 的伪文档或上下文表示为一个频率向量 d_1，而另一个单词 dog 的上下文表示成另一个向量 d_2。然后对这两个向量进行相似度计算。通过观察向量空间模型中的上下文，将聚合关系发现问题转化为向量及其相似度的计算问题。

这里要解决的两个问题是如何计算每个向量以及如何计算它们的相似度。有许多方法可以用来解决这个问题，例如信息检索中的一些方法能够很好地匹配查询向量和文档向量。可以对这些方法进行调整和改进来计算上下文文档的相似度，以解决上述问题。 256

图 13-5 显示了根据词重叠期望来匹配上下文的相似度的方法，简称 EOW。将词的上下文表示成一个词的向量，其中每个词的权重是从这个文档向量中随机抽取单词为当前单词的概率。等价地，给定一个文档向量 x，x_i 被定义为经过归一化的词 w_i 出现在该文档 d_1 的次数，可以解释成从 d_1 中以随机的方式选择这个词的概率。所有的 x_i 相加起来等于 1，因为它们是归一化的频率，也就是说，文档向量是词的概率分布。同理，向量 d_2 也可以用相同方法计算出来，从而得到两个基于概率分布表示的上下文，解决了如何计算向量的问题。

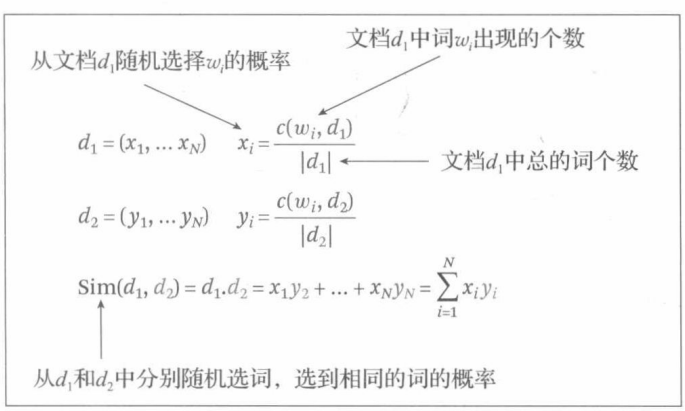


图 13-5　词语上下文的相似度函数

对于相似度，可以简单地使用两个向量的点积，表示从两个上下文中随机抽取得到两个相同词的概率。也就是判断，从一个上下文中选择一个单词，从另一个上下文中选择一

个单词，这两个词是否相同呢？如果这两个上下文非常相似，那么这两个词应该有很高的概率是相同的。相反，如果两个上下文很不一样，那么从两个上下文中抽取同一个词的概率就很小。这是测量上下文相似度的非常直观的一种方法。

257 接下来，通过公式来解释为什么相似度可以表示为随机选择的两个词相同的概率。在求和公式中的每一项，我们可以看到一个特定的单词 w_i 重叠的概率，其中 x_i 表示从 d_1 中选择该词的概率，y_i 表示从 d_2 中选择该词的概率。这就是词汇重叠期望在上下文相似度中工作的原理。

像以往一样，如果想评估这种方法是否有效，必须用真实数据来测试这个方法，看看它是否能返回真正的语义相关的词。初步分析这个公式，此方法是可行的，当两个上下文之间重叠越多，这个公式给出的分数越高。然而，如果进一步分析，可能存在一些潜在的问题。

第一个问题是，这个方法对于匹配一个频繁出现的重叠项非常有效，而匹配多个不同重叠项的效果不好。这是因为在点积中，如果一个元素具有很高的值，并且这个元素由两个上下文共享，那么它对总的加和值有很大的贡献。这样计算得到的分数高于两个向量有重叠的情况。在我们的例子中，直观上应该选择上下文中匹配更多的情况，这样才更有力地说明这两个词确实出现在相似的上下文中。如果只依赖一个高得分的项，那么这种方法是不具有健壮性的。第二个问题是这个方法对每个词都一视同仁。如果匹配 the 这样的单词，与匹配类似 eats 的单词效果是一样的，这显然是不合理的，因为 the 这样的词几乎无处不在。这是这种方法的另一个问题。

可以在文本检索中引入一些启发式方法来解决这些问题，因为在进行查询与文档匹配时也会出现这样的问题。为了解决第一个问题，可以使用词频的次线性变换。也就是说，不必使用词的原始频率计数来表示上下文，转换成一种不太强调原始频率的形式。为了解决第二个问题，可以在匹配到一个罕见词的时候给予奖励。因此，这里需要词频的次线性变换和逆文档频率(IDF)的权重；在本书第 6 章讨论了权重的不同类型方案。

基于 BM25 检索函数的 BM25 权重是实现这样的方法之一。通过对 w_i 在 d_1 中出现的次数进行次线性变换以及在相似度计算中加入 IDF 的启发式权重解决以上两个问题。

在 BM25 方法中，将文档向量定义为含有规范化 BM25 TF 值的元素，如图 13-6 所
258 示。规范化函数对所有单词进行求和，从而使得每个词的权重由所有单词的权重之和进行归一化。这样做能保证向量中的所有 x_i 加和为 1，该向量则可视为一个词的分布(因为 x_i 的总和为 1)。在相似度计算中，把词 w_i 的 IDF 值与 $x_i y_i$ 的乘积作为第 i 维的相似度。因此，第一个问题(次线性变换)是在向量表示中解决的，而第二个问题(缺乏 IDF)是在相似度函数中解决的。

$$d_1=(x_1,\cdots,x_N) \qquad \mathrm{BM25}(w_i,d_1)=\frac{(k+1)c(w_i,d_1)}{c(w_i,d_1)+k(1-b+b*|d_1|/\mathrm{avdl})}$$

$$x_i=\frac{\mathrm{BM25}(w_i,d_1)}{\sum_{j=1}^{N}\mathrm{BM25}(w_j,d_1)} \qquad b\in[0,1],\ k\in[0,+\infty)$$

$$d_2=(y_1,\cdots,y_N) \qquad y_i\text{也类似地定义}$$

$$\mathrm{Sim}(d_1,d_2)=\sum_{i=1}^{N}\mathrm{IDF}(w_i)x_i y_i$$

图 13-6　基于 BM25 的相似度计算公式

同样可以使用此方法来发现组合关系。当利用词的向量去表示基于词向量的上下文时，发现有些词有较高的权重，而有些词的权重较低。这取决于对这些词的权重分配，权重可以用于发现与上下文中候选词密切相关的词语。核心思想是利用上下文的转换去发现哪些词语的得分高。如果某个词语有很高的权重，那么该词语与候选词极其相关。

把每一个 x_i 定义为归一化 BM25 权重。该权重反映了 w_i 出现在上下文的频率。但是不能简单地说一个频繁出现在上下文的词语与候选词是相关的，这是由于有许多常用词，例如 the 经常出现在上下文中。但是，如果使用 IDF 权重，可以将这些词重新附加权重，即像 the 这样的高频词将会得到惩罚。权重较大的词不会是那些高频词，因为它们的 IDF 值比较低。相反，权重较大的词将会变成在上下文中出现的次数较多，且在整个集合中出现较少的词。显然，这样的词比较容易出现在候选词的上下文中。也正是如此，在加权向 259
量中权重较大的词可被认为是组合关系的候选。

当然，这只是发现聚合关系方法中的副产物。在下一节中，将讨论更多关于如何发现特定的组合关系的话题。这一讨论清楚地显示了发现这两种关系之间的联系。的确，这两种词间关系可以利用某种关联以一种联合方式进行挖掘。两种关系的发现也有有趣的联系。具体地说，具有聚合关系的两个词倾向于与相同的词具有组合关系。

总的来说，计算聚合关系的过程是找到候选词的上下文构建成一个伪文档，通常由词袋来表示。然后计算两个候选词相应的上下文文档的相似度。词对越相似，具有越紧密的聚合关系，即词语拥有相似的上下文。实现这个方法有许多不同的方式，在这里只介绍了几种方法。特别地，使用了文本检索模型设计有效的相似度函数来计算聚合关系。具体来讲，使用 BM25 的 TF 值和 IDF 值去发现聚合关系。最后，组合关系同时可以作为一个发现聚合关系过程中的副产物。

13.3　组合关系的发现

具有共现关系的词语之间有很强的组合关系。也就是在某个上下文中一个词出现了，那么另外一个词也会出现。

思考图 13-7 中给出的例子，当 eats 出现时，哪些词也同时出现了？左侧的句子中，可以发现诸如 cat、dog 以及 fish 等许多词都与 eats 共现。如果移除这些词并且将 eats 周围用两个空格来代替，那么是否能够预测出左右两个词呢？



图 13-7　在一个词的上下文中预测其他词

如果这些词与 eats 相关，说明它们与 eats 经常出现在相同上下文中。更具体地说，预测问题是给定任意文本片段(句子、段落或文档)判断哪些词更可能共现在同一特定上下文之中。

考虑一个特定的词 w，例如图 13-8 中，词 w 是否出现在文本片段中？当然，预测不同词语的难易程度是不同的，有些词语比其他词语更容易预测，那么，图中的三个词语(meat，the 和 unicorn)哪一个更容易被预测呢？显然，the 更容易预测，因为它经常出现。相对来说，词 unicorn 也比较容易预测，因为 unicorn 稀有。然而，词 meat 在频率上介于两者之间，因此更难预测(因为它可能出现在片段中)。

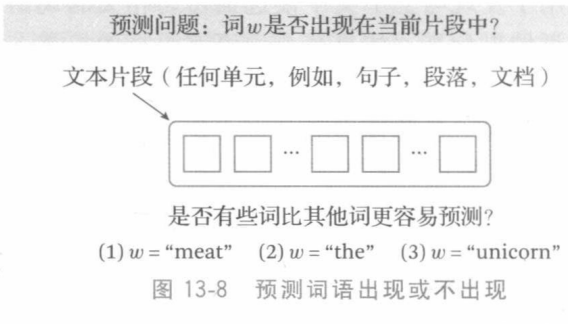


图 13-8　预测词语出现或不出现

在第 2 章熵理论中，介绍了如何利用熵来表示预测某个单词是否出现的难易程度。假定用一个随机变量 X_w 来描述词 w 是否出现，这与前面的问题是相关的。在这里，将进一步讨论条件熵，条件熵可用于发现组合关系。

260 ~ 261

现在假设一种不同的场景，假设已知随机变量的一些信息，例如已知 eats 出现在文本片段中。那么这将给词 meat 是否出现的预测带来怎样的帮助呢？如果用熵来解释这个问题，我们想知道在已知 eats 出现的情况下，能减少 meat 出现的不确定性吗？或者说，能减少 meat 是否出现所对应的随机变量的熵吗？此外，假设已知 eats 未出现在文本片段中呢？能够帮助预测词 meat 是否出现吗？这些问题都可以用条件熵来解决。

为了解释这个概念，在对一个文本片段一无所知的情况下，可以用一个概率表示某词是否出现在这个片段中。于是，产生了图 13-9 中的熵函数。

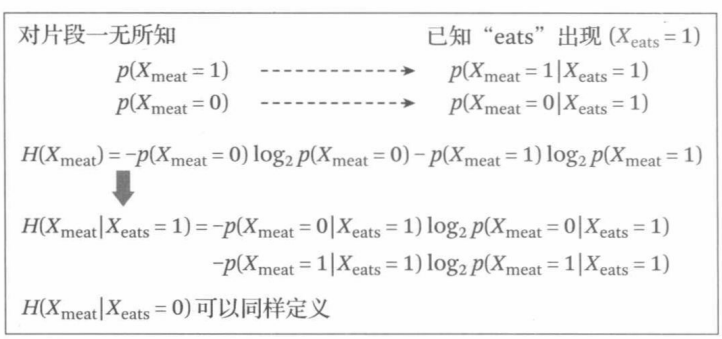


图 13-9　条件熵说明

假设已知 eats 出现，则 X_{eats} 已知，因此所有概率转化为条件概率。在已知 eats 出现在上下文的情况下，判断 meat 是否出现 $p(X_{meat} \mid X_{eats}=1)$。如果在熵函数中把这些概率替换为条件概率，则得到条件熵(在 eats 出现的条件下)。这与前面介绍的熵函数基本一致，不同之处是所有的概率都有条件。于是，我们知道了在已知 eats 出现后 meat 的熵。同时，也可以定义在 eats 不出现时的条件熵。然后将这两种情况放在一起，则产生了条件熵的完整定义：

262

$$
\begin{aligned}
H(X_{meal} \mid X_{eats}) &= \sum_{u \in \{0,1\}} p(X_{eats}=u) H(X_{meat} \mid X_{eats}=u) \\
&= \sum_{u \in \{0,1\}} p(X_{eats}=u) \\
&\quad \cdot \sum_{u \in \{0,1\}} \left(-p(X_{meat}=v \mid X_{eats}=u) \log_2 p(X_{meat}=v \mid X_{eats}=u)\right)
\end{aligned}
$$

这个公式考虑了在 eats 取不同值时，即等于 1 或 0(出现或不出现)的不同情况下的条件熵。我们将 meat 对于 eats 的条件熵定义为在 eats 取不同值时的期望熵：

$$H(X_{\text{meat}} \mid X_{\text{eats}}) = \sum_{u \in \{0,1\}} p(X_{\text{eats}} = u) H(X_{\text{meat}} \mid X_{\text{eats}} = u) \tag{13-1}$$

一般来说，对于任意离散的随机变量 X 和 Y，所得到的条件熵不大于变量 X 的熵，于是有：

$$H(X) \geqslant H(X \mid Y) \tag{13-2}$$

这是条件熵的上界。这个不等式表明只有通过增加更多信息来减少不确定性，这是合理的。正如知道更多的信息，总是会帮助我们作出预测，而不会损害预测。

条件熵给出了一种衡量两个词关联的方式，因为它能够表示当已知一个词出现或不出现的情况下，预测另一个词出现的可能性。

在利用条件熵来发现组合关系之前，考虑一种非常特殊的情况，已知词本身的条件熵：$H(X_{\text{meat}} \mid X_{\text{meat}})$。这表示在已知 meat 出现在句子中，预测 meat 是否出现。这个值等于 0，因为词 meat 的出现是已知的。这也是条件熵取最小值的条件。

接下来看其他例子，例如已知 the 预测 meat，或是已知 eats 预测 meat。那么问题是哪个更小呢，$H(X_{\text{meat}} \mid X_{\text{the}})$还是 $H(X_{\text{meat}} \mid X_{\text{eats}})$？熵值越小，越容易预测。

在第一个例子中，the 并不能给出太多关于 meat 的信息，已知 the 的出现并不能降低 meat 的熵值，所以这使得条件熵与 meat 的原始熵结果接近。而在 eats 的例子中，已知 eats 的出现与否能够帮助预测 meat 的出现，能够降低 meat 的熵值。也正是因此，第二项 $H(X_{\text{meat}} \mid X_{\text{eats}})$具有更小的熵值，也就是说这两个词之间具有更强的关联关系。 263

使用条件熵挖掘组合关系的算法如下：

1) 对于每一个词 w_1，枚举语料库中的所有其他词 w_2。

2) 计算 $H(X_{w1} \mid X_{w2})$并将所有候选的条件熵按照升序排序。

3) 排在前面的候选词作为与词 w_1具有组合关系的词语。

注意在抽取重要候选词时需要一个阈值，可以将阈值设定为选取候选个数，或是直接用某个中止值作为条件熵的阈值。

上述方法能够得到与 w_1最相关的词，但是却不能挖掘出整个集合中前 k 个较强组合关系。为此，需要保证这些不同词语的条件熵具有可比性。如果要挖掘目标词 w_1的组合关系，则只需要对比给定不同词的条件熵。

词 w_1在给定 w_2时的条件熵与其在给定 w_3时的条件熵是可比的，因为两者都是用于衡量预测 w_1出现的难度。但是，如果预测其他词而不是 w_1，那么将会得到计算熵的不同上界。因此，不能将不同词语的条件熵进行对比。下一节将介绍如何使用互信息来解决这个问题。

利用互信息挖掘组合关系

条件熵最大的问题在于不同词语之间的条件熵值不具有可比性，使得在一个完整的语料库中很难找到相关的词语。为了解决这个问题，这里使用了互信息。

X 和 Y 的互信息可以定义为 $I(X; Y)$，表示在已知 Y 之后 X 的熵的减少。这里关注的是得到 Y 的信息后，X 的熵减少了多少。互信息的数学表示如下：

$$I(X; Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) \tag{13-3}$$ 264

互信息通常是非负的，这是很容易理解的，因为原始熵永远不会低于(可能)减少的条件熵。换句话说，条件熵永远不会超过原始熵，在已知一些信息的情况下，可能会有助于

对 X 的预测。互信息的另一个特征是对称性：$I(X;Y)=I(Y;X)$。第三个特征是当且仅当两个随机变量完全独立，互信息能够达到最小值 0。即已知其中一个变量，对于另一个的预测毫无影响。

当 X 固定，对不同 Y 按照条件熵来排序时，可以得到与基于互信息相同的排序顺序。因此，按照互信息排序与按照已知 Y 时的条件熵排序效果是一样的，但是互信息允许对不同的 X、Y 对进行对比。这也是互信息更加通用的原因。

图 13-10 中展示了利用互信息挖掘组合关系的方法。

每当 eats 出现时，哪些其他词语也同时出现？这个问题可以用互信息来解释。也就是说哪个词与 eats 含有较高的互信息。因此，需要计算 eats 与其他词之间的互信息值。

互信息：$I(X;Y)=H(X)-H(X|Y)=H(Y)-H(Y|X)$

当“eats”出现时，其他哪些词容易出现？

哪些词与“eats”具有较高的互信息？

$$I(X_{\text{eats}};X_{\text{meats}})=I(X_{\text{meats}};X_{\text{eats}})>I(X_{\text{eats}};X_{\text{the}})=I(X_{\text{the}};X_{\text{eats}})$$

$$I(X_{\text{eats}};X_{\text{eats}})=H(X_{\text{eats}})\geqslant I(X_{\text{eats}};X_{w})$$

图 13-10　利用互信息发现组合关系

例如，已知 eats 和 meat 之间的互信息与 meat 和 eats 之间的互信息是相同的，因为互信息具有对称性。显然，这个互信息的值要比 eats 和 the 之间的互信息值要高，因为 the 对于其他词的预测几乎没有帮助。很容易发现，一个词与它自己的互信息值是最大的，等于这个词本身的熵。此时，熵的减少是最大的，因为已知一个就可以把另一个完全预测出来。换句话说，条件熵
265 等于零时，互信息达到最大值。

为了计算互信息，经常使用另外一种不同形式的互信息，数学表示为：

$$I(X_{w1};X_{w2})=\sum_{u\in\{0,1\}}\sum_{u\in\{0,1\}}p(X_{w_1}=u,X_{w_2}=v)\log_2\frac{p(X_{w_1}=u,X_{w_2}=v)}{p(X_{w_1}=u)p(X_{w_2}=v)} \tag{13-4}$$

上述公式就是 KL 散度(见附录 C)。公式中分数的分子是所观察到的联合分布，分母是当变量相互独立时，预期的联合分布。KL 散度量化了这两个分布之间的差异。也就是说，在独立假设下，衡量实际的联合分布与期望分布的散度。散度越大，互信息越高。

继续查看这种形式的互信息公式，可发现它也是对由两个随机变量的不同值组合加和而成的。在这个加和中，同时比较了两个联合分布。分子是实际观察到的两个随机变量的联合分布，而分母可以解释为两个随机变量的联合分布的期望。如果这两个随机变量是独立的，它们的联合分布等于这两个概率的乘积，因此这个比较表明这两个变量是否确实独立。

如果它们确实是独立的，那么分子和分母是相同的。如果分子与分母是不同的，这就意味着两个变量不是独立的，分子与分母的区别可以衡量它们的关联强度。加和计算简单地考虑到这两个随机变量的所有值的组合。在前面的例子中，每个随机变量可以选择两个值中的一个，0 或 1，所以有 4 种组合。如果考察这种互信息的形式，互信息衡量了独立假设下的实际联合分布与期望分布的散度。散度越大，互信息越高。

下面进一步查看图 13-11 中显示的互信息公式所涉及的概率。

首先，需要计算每个词语对应是否出现的概率。对于词 w_1，有相应的两个概率，这
266 两个概率相加为 1，因为一个词可以出现或不出现在文本片段中。同理，对于第二个词，也计算出两个概率表示出现或不出现，相加为 1。最后得到许多个表示这两个词共同出现情况的联合概率。它们的加和同样为 1，因为两个词的组合只有如图所示的 4 种情况。一

旦知道这些概率如何计算，则将很容易计算互信息。

注意，这些概率中有一些约束条件。第一是这些词的边缘概率相加和为 1。第二是两个词之间有 4 种共现的方式。这些额外的约束条件列在图 13-12 下方。

$$I(X_{w_1};X_{w_2})=\sum_{u\in\{0,1\}}\sum_{v\in\{0,1\}}p(X_{w_1}=u,X_{w_2}=v)\log_2\frac{p(X_{w_1}=u,X_{w_2}=v)}{p(X_{w_1}=u)p(X_{w_2}=v)}$$

w_1出现或不出现：$p(X_{w_1}=1)+p(X_{w_1}=0)=1$

w_2出现或不出现：$p(X_{w_2}=1)+p(X_{w_2}=0)=1$

w_1和w_2共同出现：

$$\underbrace{p(X_{w_1}=1,X_{w_2}=1)}_{w_1\text{和}w_2\text{共同出现}}+\underbrace{p(X_{w_1}=1,X_{w_2}=0)}_{\text{只有}w_1\text{出现}}+\underbrace{p(X_{w_1}=0,X_{w_2}=1)}_{\text{只有}w_2\text{出现}}+\underbrace{p(X_{w_1}=0,X_{w_2}=0)}_{\text{上述都不出现}}=1$$

图 13-11　互信息定义中涉及的概率

w_1出现或不出现：$p(X_{w_1}=1)+p(X_{w_1}=0)=1$

w_2出现或不出现：$p(X_{w_2}=1)+p(X_{w_2}=0)=1$

w_1和w_2共同出现：

$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=1,X_{w_2}=0)+p(X_{w_1}=0,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=0)=1$$

约束：

$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=1,X_{w_2}=0)=p(X_{w_1}=1)$$
$$p(X_{w_1}=0,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=0)=p(X_{w_1}=0)$$
$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=1)=p(X_{w_2}=1)$$
$$p(X_{w_1}=1,X_{w_2}=0)+p(X_{w_1}=0,X_{w_2}=0)=p(X_{w_2}=0)$$

图 13-12　互信息函数中的概率约束

267

第一个新的约束意味着如果把两个词同时发生的概率与当第一个词出现且第二个词不出现的概率相加，就能够得到第一个词被观察的概率。其他三个新的约束也可以类似的解释。这些方程使得能够根据其他概率计算某些概率，进而简化计算。更具体地说，如果知道一个词出现的概率，就可以计算出其不出现概率。用这些方程来计算每个单词是否出现的概率是很容易的。

w_1：出现或不出现　$p(X_{w_1}=1)+p(X_{w_1}=0)=1$

w_2：出现或不出现　$p(X_{w_2}=1)+p(X_{w_2}=0)=1$

w_1 和 w_2共同出现：

$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=1,X_{w_2}=0)+p(X_{w_1}=0,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=0)=1$$

$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=1,X_{w_2}=0)=p(X_{w_1}=1)$$
$$p(X_{w_1}=0,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=0)=p(X_{w_1}=0)$$
$$p(X_{w_1}=1,X_{w_2}=1)+p(X_{w_1}=0,X_{w_2}=1)=p(X_{w_2}=1)$$
$$p(X_{w_1}=1,X_{w_2}=0)+p(X_{w_1}=0,X_{w_2}=0)=p(X_{w_2}=0)$$

我们只需知道$p(X_{w_1}=1),p(X_{w_2}=1)$, 以及 $p(X_{w_1}=1,X_{w_2}=1)$.

图 13-13　互信息计算

接下来看一下联合概率，假定已知它们同时出现的概率，显然可以根据它们来计算所有其他概率，如图 13-13 所示。用四个公式中的第一个公式，可以计算出第一个词出现而第二个词不出现的概率，因为此时左右两个矩形框中的概率是已知的。类似的，用第三个公式也可以计算出只有第二个词出现的概率。根据图中所示，只需要计算三个矩形框中的概率，即每个词出现的概率以及两个词共现的概率。基于这些概率可以把其他的都计算出来。

268 一般来说，我们可以使用所观察到的数据中的事件的经验计数来估计概率，如图 13-14 所示。一种常用的技术是最大似然估计(MLE)，只需对观察到的计数进行规范化。使用 MLE，可以计算这些概率如下。为了估计某个词在一个片段中出现的概率，只需对片段中包含这个单词的个数进行规范化。在图 13-14 的右侧，可见一些片段的列表。在某些片段中，可见两个词是共现的，表现为两列数字均为 1。在只有一个词出现的情况下，只有那一列为 1 其他为 0。

$$p(X_{w_1}=1)=\frac{\text{count}(w_1)}{N}$$

$$p(X_{w_2}=1)=\frac{\text{count}(w_2)}{N}$$

$$p(X_{w_1}=1,X_{w_2}=1)=\frac{\text{count}(w_1,w_2)}{N}$$

	w_1	w_2	
Segment_1	1	0	只有w_1出现
Segment_2	1	1	都出现
Segment_3	1	1	都出现
Segment_4	0	0	都不出现
…			
Segment_N	0	1	只有w_2出现

count(w_1)=包含 w_1的所有片段数
count(w_2)=包含 w_2的所有片段数
count(w_1, w_2)=包含w_1和w_2的所有片段数

图 13-14　互信息定义中涉及的概率估计

为了估计这些概率，只需要知道以下三个数值：w_1出现的次数(含有 w_1的片段总数)，w_2出现的次数，以及两个词共现的次数(两列都是 1)。一旦有了这些数，直接用总的片段个数 N 对其进行归一化，由此得到用于计算互信息的概率。

还有一个小问题是有时有很多 0 值。在这种情况下，如果不想要这种 0 的概率，则可以进行平滑处理，正如前面书中讨论的。

平滑时，可直接在这些计数中加一个较小的常数，这样在任何情况下都不会得到零概率。这个例子的平滑显示在图 13-15 中。假设一些伪片段的存在将增加这些单词的计数，从而不会有任何事件具有零概率。特别地，在这个例子中，引入了 4 个伪片段，每一个权
269 重为 1/4。它们表示了两个词的共现条件的 4 种组合。

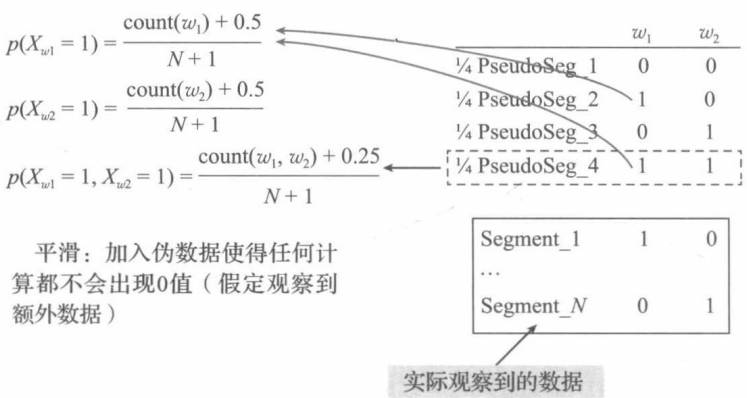


图 13-15　互信息计算中概率估计的平滑

每种组合都至少有一个来自伪片段的非零的计数，因此，在待观察的实际片段中，如果没有出现所有的组合也是可以的。更具体地说，可见 0.5 是来自于两个伪片段中的两个 1，因为每一个都有四分之一的权重。将其加起来等于 0.5。与此类似，0.25 来自于表示两个词同时出现的单个伪片段。在分母中要加上伪片段的总数，在这个例子中是 4。

总结一下，组合关系可以通过衡量两个词的相关性的方式来发现。这里使用了来自信息理论的三个概念：熵，衡量一个随机变量 X 的不确定性；条件熵，在给定 Y 的条件下 X 的熵；X 与 Y 的互信息，在给定 Y 的情况下，X 的熵的减少，或是已知 X 的情况下，Y 的熵的减少。这三个概念对于其他应用也非常有用。互信息允许不同词对之间的值进行对比，可以对这些词对进行排序从而发现文档集合中的较强的组合关系。

组合关系的挖掘与聚合关系的挖掘存在一定的联系。前面已经介绍了使用 BM25 方法去给上下文中的词赋权重从而找出与目标词具有组合关系的候选。这里，一旦使用互信息挖掘组合关系，也可以把上下文表示成带有互信息权重，由此提供了另一种表示词语上下文的方法。如果对于所有词都这样做，则可以把这些词进行聚类然后计算上下文相似度。 270
这也为聚合关系挖掘中的词权重提供了另外一种思路。

在这一章的词关联关系挖掘中，介绍了两种关联关系，即聚合关系和组合关系。这两种关系是非常通用的，因为它们可以应用于各种语言中的各种项。也就是说，待挖掘关系的单元不必须是词语(也可能是短语或是实体)。在这一章介绍了许多统计的方法来挖掘这两种关系，并证明了使用纯统计学方法是能够挖掘这两种关系的，也可以结合起来进行联合分析。这些方法可以应用于任何文本，并且不需要任何人为干预。它们虽然基于简单的单词计数，却可以发现有趣的词关联关系。当然，还可以使用不同的方法来定义上下文和片段，从而带来一些其他应用变种。例如，上下文可能非常狭小，比如只有围绕一个单词的几个词、句子，甚至可能是段落。使用不同的上下文能够挖掘到不同粒度的聚合关系。当然，本章挖掘的这些关联关系可以支持信息检索和文本数据挖掘中的许多其他应用。

词关联挖掘与词聚类有很紧密的关系，在第 14 章将会详细介绍聚类，同时一些更加先进的技术也可以用于关联关系挖掘。

13.4　词关联挖掘的评价

词关联挖掘是一项非常基本的技术，它经常被用作许多其他任务的第一步。在本章中，给出了使用词关联挖掘进行查询扩展的例子。

说服应用程序开发人员使用词关联挖掘的最好方法是展示它如何改进他们的应用。如果应用是搜索，那么问题就变成了：通过词关联挖掘增加查询扩展是否能显著地提高检索结果的 MAP 值？从第 9 章已经知道如何进行这种评价。在两个对比实验之间控制的变量是是否执行查询扩展。更加深入的对比，可以比较基于关联词挖掘的查询扩展与基于 Rocchio 反馈的查询扩展。

为了独立评价词关联挖掘，需要一套标准数据。如果没有这样的数据，则需要使用人工来判断发现的关联是否可接受。让我们首先考虑一下有标准数据集的情况。 271

不失一般性，假设想要评价组合关联的挖掘。则这个任务变成了给定一个词，对词表内的其他词与该目标词的相似度进行排序。因此，可以计算每个单词的平均准确率，然后计算所有单词的平均准确率 MAP。如果排序列表包含数值相关的分数，可以用 NDCG 和平均 NDCG。

入侵检测是一种人工的评价方式。事实上，它是主题模型[Chang 等 2009]中描述的一

种度量方法，在第 17 章会进一步讨论。如果词关联关系学习得好，就应该很容易找到被添加到 k 个类似的单词中的“入侵者”。例如，考虑 Chang 等[2009]提出的入侵检测的两个例子，有两个列表包含了 $k+1=6$ 个词。其中 $k=5$ 个词是从词表中选出来的，另外一个词是随机从词表中添加进来的。

$$L_1=\{\text{dog, cat, horse, apple, pig, cow}\}$$

$$L_2=\{\text{car, teacher, platypus, agile, blue, Zaire}\}$$

这个做法的核心思想是如果很容易发现入侵者，则前面 k 个单词构成一个语义连贯的组，是具有非常强的词关联组。在 L_1 中，很明显 apple 是入侵者，因为它与列表中其他词不一致。剩下的词构成了具有非常强的词关联列表。在 L_2 中，不能真正分辨出哪个单词是入侵者，这意味着用于在 L_2 中创建 k 个候选词的关联挖掘算法不如生成 L_1 的单词算法好。通过对词表中的许多不同的词进行这种实验，是一种很好(但也很昂贵)的方法来严格地评价词关联关系。之所以说这种方法很昂贵，因为它需要很多人工的判断。

最后，词关联挖掘中使用这种方法的时间-精度的权衡也是很重要的。想象这样一个场景，在某数据集上基线系统能够得到 MAP 的值为 0.89，如果使用基于词关联挖掘的查询扩展，可以使得 MAP 值显著地提到 0.90。然而，这没有考虑词关联挖掘的处理时间。在这个例子中，词关联挖掘是提前线下完成的，所以查询时间不会受到影响。应用程序的管理者需要衡量 MAP 值 0.01 的提升是否值得实现、运行和维护查询扩展。新技术替代或
272 扩展已有技术，这是一个非常现实同时也是很常见的问题。作为一个数据科学家，说服别人认可这样的修改，并认识到对整个系统是有用的和有价值的，这常常也是工作的一部分。

书目说明和延伸阅读

Manning 和 Schütze[1999]中有两个章节与词关联挖掘相关，第 5 章和第 8 章。较早使用互信息挖掘词关联关系的工作是 Church 和 Hanks[1990]。聚合关系和组合关系也可以在词语的邻接图上进行随机游走来挖掘，Jiang 和 Zhai[2014]提出了一个同时挖掘这两种词关联关系的模型。例如 hot dog 这类的非成分短语(也称为词汇原子)，也可以用类似的启发式方法发现，这个在本章讨论过(见 Zhai 1997，Lin 1999)。另外一个挖掘词关联关系的算法是 n 元语言模型[Brown 等 1992]。最近，词嵌入技术(如 **word2vec**；Mikolov 等 2013)在学习词语的向量表示方面表示出了非常好的效果，进而可以计算两个词语的相似度，支持聚合关系的发现。n 元语言模型和 **word2vec** 都将会在第 14 章聚类中介绍。

练习

13.1　条件熵 $H(X|Y)$的最小和最大值是什么？它们发生在什么情况下？

13.2　在互信息部分，采用了一种简单的平滑技术。根据在第 6 章掌握的知识，定义一个更健壮的平滑方法计算组合关系。

13.3　特征选择是减少特征空间的维数以提高性能和减少运行时间的过程(因为特征变少)，概述一种基于词关联的一元词特征表示的特征选择方法。

13.4　组合和聚合词关联挖掘方法能够有效地用于其他类型特征吗？举例说明它可能起作用的其他特性，以及其他可能不适用的特性。

273 13.5　使用 MeTA 实现一个或两个词关联挖掘方法。使用默认的一元分词来读取语料并为每个词项 ID 创建特征向量。然后，给定一个查询词项，返回最相似的词项。

13.6　概述一种把具有相似意思的单词(多于 2 个)分组在一起的方法，例如 plane、car 和 train 都是相关的。

13.7　概述基于搜索引擎日志确定同义词的方法。也就是说，给定多个查询，而每个查询都有一个点击的(假定相关)文档的列表。

13.8　概括一种基于搜索引擎日志的同形异义词的方法(两个单词拼写相同)辨别方法。例如，我们如何根据这些日志区分金融机构 bank 和河岸 bank?

13.9　在什么情况下(如果有)词关联规则挖掘是 n 元语言模型的推广形式?

13.10　根据上下文的大小，能够从词关联中获得许多不同类型的语义。请给出当上下文很大或很小时两种关系类型的极端情况。

13.11　调整词关联挖掘算法，使其挖掘反义词而不是同义词，可行吗? 如果可行，解释如何去做；如果不可行，解释为什么不可能。也就是说，尝试给(cold，hot)这样的词对打高分，因为它们是对立的；而给(freezing，cold)打低分，因为它们是同义词。

274

第 14 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本聚类

聚类是探索性文本分析中的一个自然问题。其基本意义是，通过收集相似的对象把对象聚类(即分组)可以发现语料库中的一些固有结构。这些对象可以是文档、句子或者词。除此之外，还可以对搜索引擎的查询、搜索引擎的结果，甚至是用户进行聚类。

聚类是一种常用的分析大规模数据集的数据挖掘技术。当面对一个大规模的文本数据的集合，聚类以多个数据对象的簇(组)的形式揭示数据的语义结构。聚类的结果有时可以直接作为知识应用。例如，对客户电子邮件聚类可以发现客户对产品的主要投诉类型。通常，聚类结果能够在深入数据的某个子集中进行集中分析之前，在总体上提供对数据的一个概览。聚类结果还可以支持导航到相关的数据子集，因为这些结构可以促进簇内对象的连接和相关簇的连接。一般来说，聚类方法对于文本挖掘和具有广泛应用前景的探索性文本分析非常有用，特别是由于聚类算法大多无监督，不需要任何人工努力，因此可以应用于任何文本数据集。

聚类的对象类型能够适应不同的任务，这种变化产生许多有趣的应用程序。例如，检索结果的聚类可以用作结果摘要或作为删除冗余文档的方法。在整个语料库中对文档进行聚类可以发现共同的底层主题，并能更好地理解它包含的数据类型。词语聚类是查找概念或创建同义词库的有效方法。

那么，如何正式定义聚类问题呢？特别是，对于一个对象在某个特定的簇中，它实际上意味着什么呢？直观地说，同一簇中的对象在某种程度上比那些出现在两个不同的簇的对象更相似。然而，这种对聚类的定义严格来说不是很好定义，因为我们并不清楚如何精确地度量相似度。事实上，对相似度的准确定义对于聚类是非常关键的，因为不同的定义显然会导致不同的聚类结果。

考虑图 14-1 中的例子，怎样将图左侧的对象进行聚类呢？一个理想的聚类结果是怎样的？显然，这些问题不能回答，除非能够清楚地定义相似度，即注入一个特定的“聚类偏置”(clustering bias)。如果按照对象的形状来定义相似度，那么将会得到图中中间的聚类结果。然而，如果按照对象的大小来定义相似度，那么将会得到图中右侧的聚类结果。因此，当定义聚类任务时，最重要的是要定义好期望的相似度度量，称之为聚类偏置。这种偏置也是评价聚类结果时的基础。相似度视角的模糊性不仅存在于这样一个人造的例子中，而且无处不在。比如说词“car”和“horse”相似吗？car 和 horse 在物理上显然不相似，然而，如果从功能来看，可以说是相似的，因为它们都能用作交通工具。“正确”的聚类偏置显然必须由特定的应用来决定。

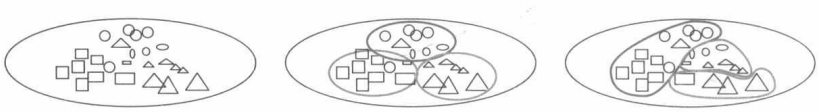


图 14-1 聚类偏置示例。图中左侧集合中的对象可以根据相似度定义(聚类偏置)以不同的方式聚类。中间表示用形状作为相似度定义时的聚类结果。右侧表示用大小作为相似度定义时的聚类结果

在不同的算法中，聚类偏置以不同的方式注入。有一些聚类算法，由用户定义或选择

一种相似度算法来聚类。它将根据所选择的相似算法将所有相似的文档放在一个簇内。其 276
他聚类算法是基于模型的(通常基于生成概率模型)，其中数据的模型的目标函数(例如，生成概率模型中的似然函数)是定义间接相似度产生的偏置。在基于模型的方法中，每个对象通常会被赋予一个在所有簇上的概率分布，也就是并非像基于相似度聚类中的硬性的簇分配。在本书中，主要介绍基于相似度和基于模型的聚类方法。本章主要介绍基于相似度的聚类，而第 17 章主题分析是基于模型聚类的方法之一。

在本章中，将研究单词和文档的聚类技术。句子聚类可以看作是小型的文档聚类的一个例子。首先从聚类技术的概述开始，对不同的方法进行分类。接下来，通过两种常用方法(层次和分裂方法)讨论基于相似度的聚类。在提到两个更高级的主题之前，通过语义相关性和点互信息来介绍词的聚类。最后介绍聚类的评价方式。

14.1　聚类技术概述

如前所述，文档聚类将文档分组成簇。这里，将文档聚类方法分为两类。

基于相似度聚类(similarity-based clustering)：这些聚类算法需要一个相似度计算函数。任何两个对象都有相似的可能性，这取决于从哪个角度来评判。因此，用户必须以某种方式定义相似度。

凝聚聚类(agglomerative clustering)：是一种“自下而上”的方法，也被称为层次聚类，在此方法中，我们逐步合并类似的对象来生成聚类簇。

分裂聚类(divisive clustering)：是一种“自上而下”的方法，逐渐将整个对象分割成更小的簇。

对于以上两种方法，每个文档只能属于一个簇。这是一个“硬分配”，不同于从基于模型的方法得到的簇。

基于模型的技术(model-based technique)设计了一种概率模型来捕捉数据的潜在结构(即文档中的特征)，并调整模型使其与数据相匹配以获得聚类。这是一个典型的软聚类(soft clustering)的例子，因为一个对象可以(以一定的概率)在多个簇中。在主题分析的章节中会有更多的讨论。 277

词项聚类(term clustering)可以应用于查询扩展中，使得在查询中添加类似的检索词，增加与索引匹配的文档的可能数量。同时，在文本具有数百或数千个特征或很难被概念化时，也能够帮助更容易地理解文本。稍后，将讨论第 2 章中提到的通过主题语言模型的语义关联进行词语聚类。然后，介绍一种简单的概率技术，即点互信息(pointwise mutual information)。简单地介绍一种层次聚类方法布朗聚类(Brown clustering)。该技术与文档合并聚类以及基于模型方法的概率性质方面具有相似性。最后以词向量(word vector)的解释来完成词项聚类的介绍。词向量是一种基于上下文的词表示，我们可以回顾一下介绍信息检索问题时的词表示并与之比较。

在第 17 章，基于模型的主题分析额外输出相似词语的分组(实际上是主题)。因此，主题分析能够提供给用户词和文档的聚类簇。

尽管在无监督的聚类算法中，通常不需要提供簇内可能需要包含哪些对象的先验期望，但是却可以提供一些有监督信息，从而表明两个对象在或不在同一个簇内。在对于聚类问题有一些了解的情况下，这样的监督是有用的，希望把它融入聚类算法中，并允许用户以灵活的方式“引导”聚类。用户还可以通过事先设置来控制所需的簇的数目，或者用户把它留给算法来确定对象的自然分解是什么，在这种情况下，聚类簇的数量通常是基于

统计方法来进行优化得到的，例如多少个簇能把数据解释得更好。

大多数聚类方法的输出不能给找到的簇提供标签。可以由用户来检查词或文档的聚类簇，并指定一个标签，如“生物学”或“体系结构”。当然，也有方法能够自动给文本簇分配标签，这里的标签通常是一个短语或多个短语[Mei 等 2007b]。这种标记任务可以看作是文本摘要的一种形式，将在第 16 章中进一步讨论。

最后，简要介绍了聚类算法的实现。在本书剩余的章节中，可以看到在第二部分讨论的信息检索技术对于文本分析(包括聚类)等其他算法都是十分有效的。例如，在文档聚类
278 中，可以假设根据某些特征表示，我们已经有了标记文档的前向索引。在支持文本数据获取和文本分析的统一软件系统中，利用已有的可用于支持搜索的数据结构是特别理想的。我们讨论的聚类技术是通用的，可以潜在地用于聚类许多不同类型的对象，包含诸如单字词、两字组词、三字组词的词性标注或句法树特征等。所有聚类算法都需要一个词汇表将词项表示为对应的 ID。聚类算法只关心词项的出现和概率，而不关心它们实际代表什么。因此，使用相同的聚类算法，我们可以将文档按单词用法或类似的语法模式表示为语法解析树段。对于词语聚类，我们可能不使用索引，但是我们假设每个句子或文档是可分的，并且分配词项的 ID。

14.2　文档聚类

在这一节中，我们通过两种方法研究基于相似度的文档聚类：凝聚聚类和分裂聚类。由于这些都是基于相似度的聚类方法，所以需要相似性度量。如果需要回顾相似性度量，建议读者参考第 6 章内容。

特别地，用于聚类的相似度算法需要是对称的，也就是说，$\mathrm{sim}(d_1, d_2)$必须等于 $\mathrm{sim}(d_2, d_1)$。此外，相似度算法必须在一定范围内进行规范化。通常，这个范围是[0，1]。这些约束确保可以比较不同对象对的相似度分数。大多数检索公式都是不对称的，如 BM25、枢轴长度规范化以及查询似然模型，因为它们把查询视作与当前待评分文档不同。Whissell 和 Clarke[2013]对几种流行的检索公式的对称版本展开研究，并证明了其有效性。

尽管默认的查询文档相似度方法不适用于聚类，可以使用(例如)Okapi BM25 计算文档向量中的词权重，然后用类似余弦相似度的简单对称相似算法计算得分。余弦相似度定义如下：

$$\mathrm{sim}_{\mathrm{cosine}}(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i (x_i)^2}\sqrt{\sum_i (y_i)^2}} \tag{14-1}$$

由于文档向量表示中的所有项权重都是正的，因此余弦相似度分数范围是[0，1]。如前所
279 述，词的权重可以是原始计数、TF-IDF 或者是用户自定义的。余弦相似度捕捉在高维空间下两个文档向量的夹角余弦值。夹角越大，相似度越低。

另一个常用的相似性度量是杰卡德相似度(Jaccard similarity)。这个度量用于衡量集合的相似性，也就是说，它只捕获词的存在与否，而不在乎其权重大小。它的定义如下：

$$\mathrm{sim}_{\mathrm{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{14-2}$$

其中 X 和 Y 分别代表文档向量 x 和 y 中的元素集。简单地说，它捕获两个集合中共有对象和总对象的比率。

如需更深入地研究相似性度量及其有效性，建议读者参考 Huang[2008]。对于本章的其

余部分，假设文档-文档相似性度量是余弦和杰卡德相似度已经足够了。在任何情况下，特定的相似度算法的目标是找到最佳的数据分区，同时最大化簇内相似度并最小化簇间相似度。

14.2.1　凝聚层次聚类法

现在，讨论第一个聚类策略。此方法逐步构造聚类簇以生成合并簇的层次结构。这种自下而上的(凝聚)方法逐渐将相似的对象(单个文档或一组文档)组合到大的簇内直到只剩下一个簇。然后可以根据需要对树进行分割，或者，当找到所需的簇数时，可以停止合并。这一系列的合并形成了一个**树状图**(dendrogram)，如图 14-2 所示。

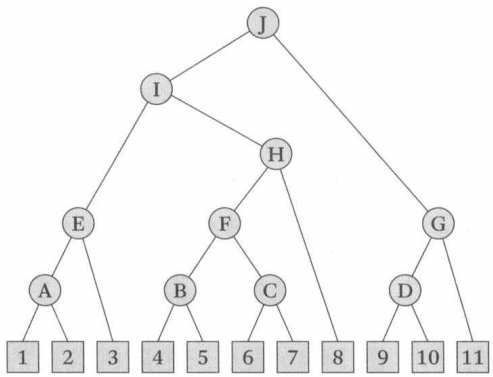


图 14-2　层次聚类表示为树状图

在图中，原始文档被编成 1～11 的编号并置于图的最底部。圆圈代表不止一个文档的簇，而线条代表哪些文档或簇被合并在一起形成下一个更大的簇。

聚类算法很直接：在存在多个簇的情况下，算法找到两个最相似的簇并合并它们。当需要比较簇与簇或簇与单个文档的相似性时，就存在一个问题。到目前为止，只定义了两个文档作为输入的相似性度量。为了简化这个问题，可把单个文档看作簇，因此只需要比较簇的相似性即可。簇的相似性度量使用先前提出的文档-文档相似性度量。

接下来将介绍三种簇的相似性度量，并在图 14-3 中表示出来。

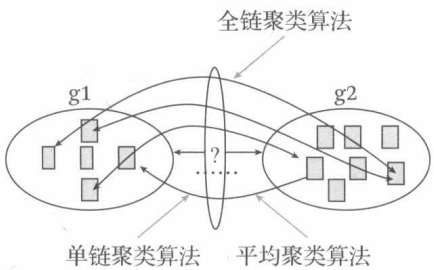


图 14-3　三种不同的聚类簇间相似性度量

单链聚类(single-link)合并具有最小最近(smallest minimum)元素距离的两个簇。这就导致了“松散”的簇，因为只需要在两个簇中找到靠近的独立元素便可执行合并。

全链聚类(complete-link)合并具有最小最远(smallest maximum)元素距离的两个簇。这样导致比较紧密和紧凑的簇，因为簇直径保持很小(即所有元素之间的距离很小)。

平均链接聚类(average-link)是前两个方法之间的折中。顾名思义，它将合并具有最小平均(smallest average)元素距离的两个簇。

单链聚类和全链聚类对异常值比较敏感，因为它们只依赖于一对文档的相似度。平均聚类本质上是一组决策，使得它对离群值不那么敏感。当然，正如本书中讨论的大多数方法一样，根据特定的应用来决定所选的方法。事实上，可以尝试不同的文档-文档相似性度量并且结合不同簇间的相似性度量以查看数据集被如何划分。

14.2.2　K-均值

与前面介绍的层次聚类算法互补的一种聚类方法是自上而下的、分裂的方法。在这种方法中，反复应用平坦型聚类算法将数据分割成更小的簇。在平坦型聚类中，将从初始试探聚类开始，并反复改进，直到达到某种停止准则。这里，将簇表示成**质心**(centroid)；质心是一个特殊的文档，它代表簇中的所有其他文档，通常是所有成员值的平均值。

K-均值算法[⊖]设置 *K* 个质心，迭代地重新分配文档到每一个簇直到簇分配变化很小或不存在。下面的算法描述了这种技术。令 sim(·)来表示被选择的文档-文档相似性度量。

算法 14-1　*K*-均值聚类算法

```
Initialize K randomly selected centroids
while not converged do
    Assign each document to the cluster whose centroid is closest to
        it using sim(·) (Ex.)
    Recompute centroids of the new clusters found from previous step (Max.)
end while
```

K-均值算法中的两个步骤为期望步骤和最大化步骤。这个算法是广泛应用的期望最大化算法的一个实例，即 EM 算法(Expectation-Maximization algorithm)。在第 17 章将通过主题分析中的 PLSA 算法来介绍这个算法的更多细节。在本章中，充分认识到 *K*-均值是通过 EM 算法实现硬聚类的一个特殊表现形式即可。

282

图 14-4 展示了 *K*-均值算法的运算过程。图 14-4a 是待聚类数据点的初始设置。在这里，用不同的形状可视化数据点，表明有三个不同的簇，对应于三个形状(交叉、圆和三角形)。图 14-4b 表示三个随机选择的质心($K=3$)。在图 14-4c 中，黑线表示文档在各自质心中的分区。这些线可以先通过画线连接每对质心，然后通过找到连接两个质心的线段的垂直平分线来发现。这一步骤在伪代码中标记为(Ex.)。一旦确定了簇的划分，图 14-4d 显示了如何重新计算以改进质心位置，标记为(Max.)的伪代码。因此图 14-4c 和图 14-4d 代表算法为改进质心位置的一轮迭代。图 14-4e 进一步显示了如何使此算法继续得到改进的边界，进而进一步提高质心。

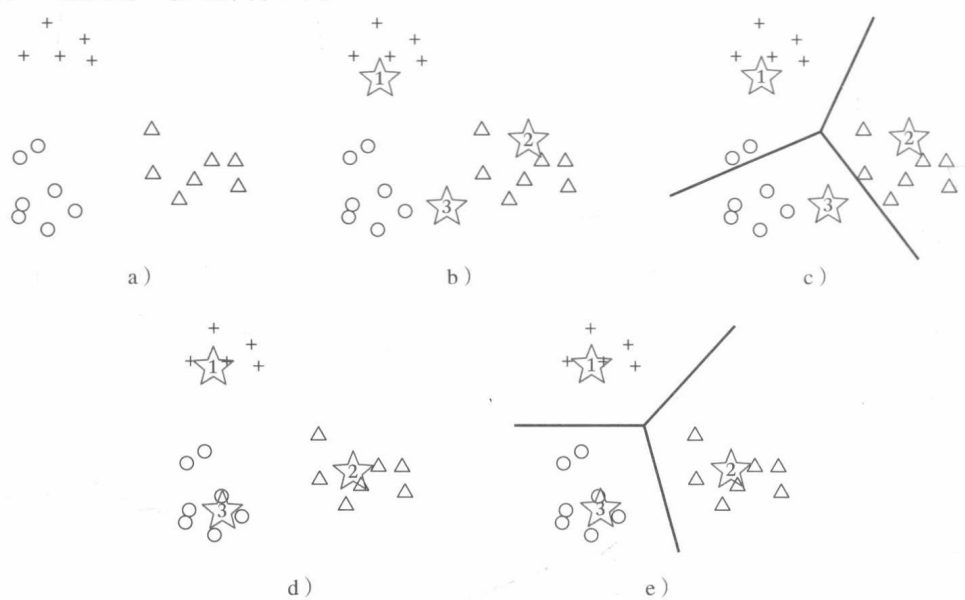


图 14-4　一些小的数据点进行 *K*-均值聚类算法的步骤(如 a 所示)。首先，随机选择三个初始(开始)的质心(如 b 所示)。然后，根据与每个质心的距离将所有数据点分配到三个集群中的一个；决策边界显示为 c 中的线。由此分配得到三个簇，每个簇可以用来计算出新的质心来更好地代表这个簇(如 d 中三个星的新位置)。该算法以新的质心作为起始质心，重新分配所有的数据点反复迭代。e 中展示了新的边界，这时候很容易看到已经非常接近从这个数据集产生的三个簇的最优的质心

283

⊖ *K*-均值算法与分类算法 *K*-NN(见第 15 章)无关。

当一个文档被表示为一个词项向量(如第 6 章所讨论的)，使用欧几里得距离函数时，K-均值算法可以最小化一个目标函数，该函数计算簇中所有数据点到簇质心的平均距离。该算法可以收敛到局部极小值，但不能保证收敛到全局最小值。因此，为了获得较好的局部极小值，通常需要进行多次试验。

K-均值算法可以反复应用，逐步将数据集划分为更小的簇，因此，可以创建出类似于凝聚层次聚类算法的具有层次结构的簇。由此，凝聚层次聚类和 K-均值均可用于层次聚类；在某个意义上来说，它们相互补充，即 K-均值逐步将整个数据集分解构建层次(自上而下的策略)，而凝聚层次聚类通过逐步合并数据点构建层次(自底向上的策略)。值得注意的是，凝聚层次聚类的基本形式是生成一棵二叉树，它可以很容易地在一轮迭代中通过融合两组以上的簇形成两个以上分支。类似地，如果只需要一棵二叉树，则不必在 K-均值算法中设置 K 来创建一个层次结构。

14.3　词项聚类

词项聚类的目标与文档聚类非常相似，希望找到相关的词项。“相关”通常指的是具有相似语义的词。例如，soccer 和 basketball 在某种意义上说，它们都是体育运动。同样，evaluation 和 assessment 是相关的，因为它们是同义词。

在这一节中，将把“词项”和“词或单词”替换使用，但记住，不一定只有词能够聚类。使用词为例，是因为它理解起来比较容易。在本节中描述的词项通常适用于任何一种特征，无论是单词还是分析树片段。重要的是要记住，我们讨论的算法在大多数情况下是适用于词的。注意在某些形式的“聚类”中，得到结果仅仅是两个词 w_1 和 w_2 之间成对的评分。如果这些分数是规范化的，则仍然可以使用成对的分数对整个词项集合进行聚类。 284

在所有聚类问题中，相似度的定义是非常重要的。在词语聚类的情况下，问题就变成应该如何定义两个词之间的相似度。不难看出，词之间的聚合关系或组合关系是定义相似度最自然的候选方式。聚合关系相似度将导致聚类簇中的词通常是出现在相似的上下文且具有相同的相对位置，而组合关系会导致聚类簇中的词语义相关，也往往出现在相似的上下文中，但是在不同的位置。

在剩下的部分中，将首先回顾本书前面介绍的关于挖掘语义相关词项的方法。然后，介绍点互信息的概念，以及如何用于关联词项的发现。最后，介绍一些更高级的词语聚类方法。

14.3.1　语义关联的词语

回顾 3.4 节想要挖掘哪些词与 computer 一词有语义关联。图 14-5 是从 3.4 节截取而来的。

例子中，使用最大似然估计对一元语言模型进行估计并得到 $p(w|\hat{\theta})$，这里$\hat{\theta}$是主题语言模型中含有词 *computer* 的文档，即

$$p(w|\hat{\theta})=\frac{c(w,\ D)}{|D|} \tag{14-3}$$

在估计了主题和背景语言模型之后，使用下面的公式给词汇表中的每个词打分：

$$\text{score}(w)=\frac{p(w|\text{computer})}{p(w|C)} \tag{14-4}$$

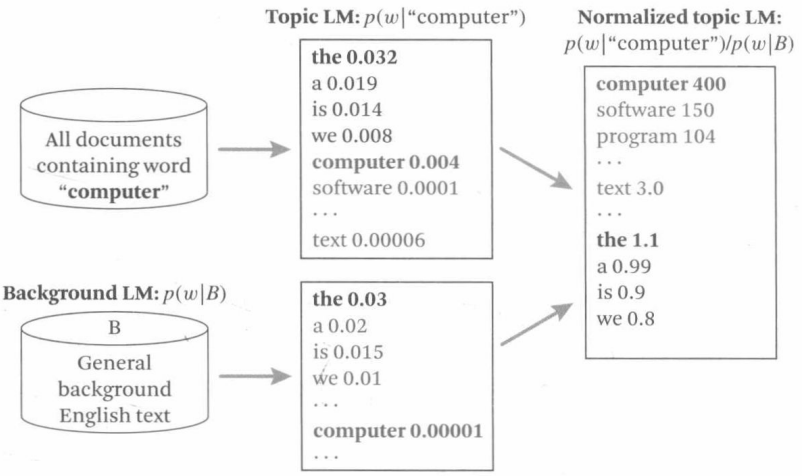


图 14-5　使用主题语言模型和背景语言模型找到语义关联的词语

这个分数表示某个单词与主题语言模型中的词 computer 的相关性。使用最大似然估计，得到：

$$\text{score}(w)=\frac{p(w|\text{computer})}{p(w|C)}=\frac{\frac{c(w,\ D)}{|D|}}{\frac{c(w)}{|C|}}=\frac{c(w,\ D)\cdot|C|}{c(w)\cdot|D|} \tag{14-5}$$

其中 D 表示含有词 computer 的文档集合，C 表示整个文档集合。

我们发现那些更可能在 computer 上下文出现的词将有一个比分母更大的分子，从而增加分数。不管上下文如何，出现机会相等的词(例如 "the")都会有一个接近于 1 的分数。通常不在 computer 上下文出现的单词的分母小于分子，导致分数小于 1。

正如 3.4 节所提到的，这个规范化公式有一个小问题。例如，假设词语 artichoke 在语料库中只出现一次，它恰好出现在 computer 也同时出现的文档中。使用上面的公式将会得到 artichoke 与 computer 非常相关，然而实际并非如此。

解决这个问题的一个方法是通过假设观察到额外的伪词语个数，包括未出现词，来平滑最大似然估计。因此，平滑背景语言模型的计算公式如下：

$$P(w|C)=\frac{C(w,\ C)+1}{|C|+|V|} \tag{14-6}$$

285
~
286

这里 $|C|$ 是指集合 C 中所有词的个数，$|V|$ 是指词表中全部词个数。注意分母中 $|V|$ 是指对于词表中所有词加入的用于平滑的伪词语个数。

在这个公式下，未出现词就不会被赋予 0 概率值，然后估计出的概率也更加准确。可以用平滑后的公式替换前面公式中的 $p(w|C)$。在这个例子中，得到的分数比原本低很多，因为我们假设它在背景词中也出现了。而真正语义相关(即在 computer 上下文中共现得更频繁)的词将不会被这种平滑所影响，相反，在分数排序列表中，由于不相关词被排在后面，这些词的排名反而能够上升。

在第 6 章中已经讨论过，这种加一平滑可能不是最好的平滑方式，因为它给未出现词施加了太多的概率。我们可以使用其他平滑方法(诸如狄利克雷先验以及 Jelinek-Mercer 插值)来进一步改进评分步骤。

这种语义相关性是在词项聚类应用中我们希望捕捉到的。然而为词表中的每一个词做这样的计算是不可能的。因此，在下一节将会介绍一种更有效的词项聚类方法，其核心思

想是一致的。

14.3.2　点互信息

点互信息(PMI)将词共现视为随机变量，并将其同现在窗口上下文中的概率进行量化。例如，在大小为 n 的上下文窗口中与词 w_i 共现的词为

$$w_{i-n},\ \cdots,\ w_{i-1},\ w_i,\ w_{i+1},\ \cdots,\ w_{i+n}$$

由此，能够计算 w_i 与 w_j 共现的概率，用联合概率 $p(w_i,\ w_j)$ 来表示，再给定独立概率 $p(w_i)$ 和 $p(w_j)$，可得到 PMI 公式如下：

$$\mathrm{pmi}(x_i,\ x_j)=\log\left(\frac{p(w_i,\ w_j)}{p(w_i)p(w_j)}\right) \tag{14-7}$$

注意如果 w_i 和 w_j 是独立的，那么 $p(w_i)(w_j)=p(w_i,\ w_j)$。对 1 取 log 之后，得到 PMI 的值为 0。两个独立的词是无法传递任何信息的。如果两个词共现的概率 $p(w_i,\ w_j)$ 高于两个词相互独立时共现的期望概率 $p(w_i)p(w_j)$，则 PMI 比期望的更高。 287

根据应用需要，可以将上下文定义为前面提到的大小为 n 的窗口、句子、文档等。更改上下文可以带来对 PMI 的不同理解，例如，如果只考虑窗口大小为 $n=1$ 的上下文，那么将比语料库中的整个文档作为上下文得到显著不同的结果。为了获得可比较的 PMI 值，还需要确保 PMI 度量是对称的，这又取决于上下文的定义。如果定义上下文为“w_j 在 w_i 之后出现”，那么 $\mathrm{pmi}(w_i,\ w_j)\neq\mathrm{pmi}(w_j,\ w_i)$，这对于词项聚类是必要的。

PMI 值可以归一到[0，1]之间：

$$\mathrm{npmi}(w_i,\ w_j)=\frac{\mathrm{pmi}(w_i,\ w_j)}{-\log p(w_i,\ w_j)} \tag{14-8}$$

使得可以对不同的词对进行比较。然而，这种归一化并不能解决 PMI 公式本身中的一个主要问题。想象一下，如果有一个罕见的词，总是出现在另一个(可能是非常普遍的)词的上下文中。看起来这个词对是高度相关的，但实际上数据太稀疏，无法恰当地建模这种关联。这个问题可以通过使用第 13 章中引入的互信息来缓解，它不仅考虑了观察到稀有词的情况，而且还考虑了未观察到的情况。事实上，由于互信息受到两个变量之一的熵所限定，而稀有词具有很低的熵，所以它通常不会与其他词有很高的互信息。

尽管 PMI 和 nPMI 有各自的缺点，但是它们的应用十分广泛，也经常被用作构建更高级方法的基础模块，同时能够帮助理解词共现背后的主要思想。因此，在本书中讨论了这个问题。

接下来主要介绍词项聚类的两种更先进的方法。这里采用的窗口技术在下面两种方法中都是非常重要的。

14.3.3　先进方法

在这一节将介绍词语聚类更先进的方法。

14.3.3.1　n 元类语言模型

布朗聚类[Brown 等 1992]是一种基于模型的词项聚类算法，它最大化 n 元类语言模型的似然。然而，由于优化问题很难计算，实际构建词聚类的过程类似于层次化聚类，词语逐个进行合并，但布朗聚类中的合并准则是基于似然函数的相似度函数。特别地，最大化似然函数与最大化相邻词类的互信息是等价的，因此在合并两个词时，算法倾向于合并两 288
个分布相似的词，因为这样的词被它们各自的类替换时，将会最小化相邻词类之间互信息

的减少。

按照数学定义，假设将词表中的所有词分成 C 个类别，则 n 元类语言模型定义了在已知 w_n 前 $n-1$ 个词，即 w_{n-1}，…w_1，的情况下，w_n 出现的概率

$$p(w_n|w_{n-1}, \cdots, w_1)=p(w_n|c_n)p(c_n|c_{n-1}, \cdots, c_1)$$

其中 c_i 是词 w_i 的类别。它本质上假设 w_n 出现的概率依赖于前面词语的类别，而不是某个指定的词，除非 C 与词表大小相等(即每个词自己的类别)，n 元类语言模型通常比常规的 n 元语言模型参数要少。

在生成模型中，要生成某个词，首先要查看它前面词的类别，即 c_{n-1}，…，c_1，然后利用 $p(c_n|c_{n-1}, \cdots, c_1)$给位置 n 的词采样一个类别 c_n，最后根据概率 $p(w|c_n)$采样位置 n 的词。概率 $p(w|c_n)$给出了当词的隐含类别 c_n给定时 w 出现的概率。

如果已知词语被分到 C 个类别中，那么最大似然估计变得并不难，可以按照常规的 n 元语言模型一样的方式，简单地将词语替换为它们对应的类别去估计 $p(c_n|c_{n-1}, \cdots, c_1)$。给定一个类别，概率 $p(w|c)$可以很简单地通过统计数据中所有属于类别 C 的词的词频并进行归一化得到，即根据词 w 在数据集中的个数对 $p(w|c)$进行估计。

然而，找到最佳的词分类是非常困难的。幸运的是，可以使用贪心算法去构建词的类别，这与凝聚层次聚类十分相似，即根据似然函数最大化的目标逐个将词语合并去构建类别。在二元语言模型中，一个比较理想的理论结果是，最大化似然函数与最大化两个相邻
289 类别之间互信息是等价的。因此，最容易合并的词是具有相似上下文的(如 Tuesday 和 Wednesday)，因为把这些词合并在同一类，对于类别的预测能力与原始词一致，能够最小化互信息的损失。为了计算方便，直接采用基于似然函数的层次聚类，这个似然函数可以表示两词合并时互信息的损失。由于模型的复杂性，最初只在二元语言模型上进行了探索[Brown 等 1992]。

经验表明，二元类语言模型是非常有效的，并且如果将同一类别词语视为聚合关系，则能够产生出非常高质量的词语聚合关系。图 14-6 展示了从文献[Brown 等 1992]抽样出来的词语聚类簇，可见这些簇可以体现聚合关系。

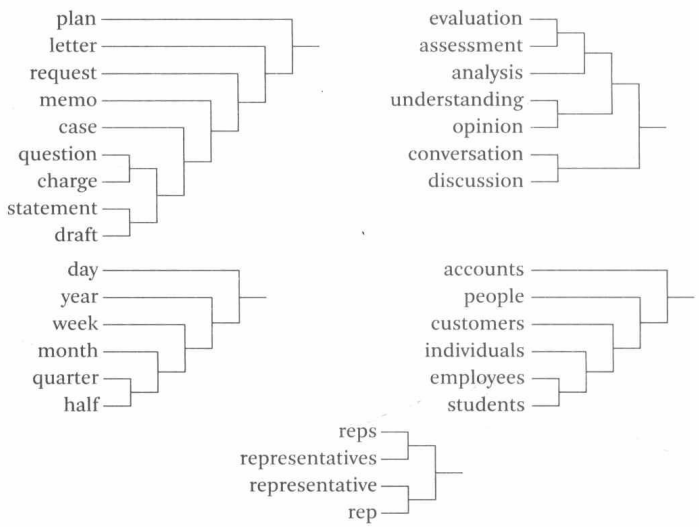


图 14-6　使用 n 元类语言模型层次化构建词语类别采样(来自 Brown 等[1992])

通过计算出现在不同位置词语之间的点互信息，这个模型也可以用于生成组合关
290 系。当共现的窗口限制为 2 个词(即邻接共现)，这个模型可以用于发现固定搭配(见

图 14-7 中的结果样例），这些短语的意思并非由两个词的意思直接组合而成，是非组合短语。当然，这种非组合短语也可以通过其他统计方法来发现（如 Zhai 1997，Lin 1999）。

14.3.3.2　神经语言模型（词嵌入）

第 13 章讨论了如何基于上下文将一个词表示成词项向量，并且根据其向量表示计算相似度。这种上下文视角的词项表示不仅可以用于发现聚合关系，同时也能够帮助词项聚类，因为任何文档聚类方法都可以把一个词看作一个由向量表示的文档。同时也能够帮助词义消歧，由于歧义词的周围通常有不同的词，具有不同的上下文表示。

词对	互信息
Humpty Dumpty	22.5
Klux Klan	22.2
Ku Klux	22.2
Chah NuIth	22.2
Lao Bao	22.2
Nuu Chah	22.1
Tse Tung	22.1
avant garde	22.1
Carena Bancorp	22.0
gizzard shad	22.0
Bobby Orr	22.0
Warnok Hersey	22.0
mutatis murtandis	21.9
Taj Mahal	21.8

图 14-7　使用 n 元类语言模型发现非组合短语采样

此技术不仅局限于一元词语，同时也可以引入词性标签甚至是情感。加入这些额外的信息意味着把词向量从 $|V|$ 扩展到需要的维度。此外，除了找到语义相关的词，利用更丰富的词表示能够帮助下游的其他任务，例如语法分析或统计机器翻译。

然而，在第 13 章讨论的启发式获取向量表示的方法的缺点是需要额外的权重设置。另一个不足是向量需要覆盖整个词表，增加了对向量处理的复杂度。

另一种选择是利用神经语言模型[Mikolov 等 2010]通过优化目标函数系统地为每个词学习一个向量表示。这样的方法叫作词嵌入（word embedding），把词语映射到低维空间的一个向量中。这种方法的基本思想是假设每个词对应于一个未知（潜在）低维空间中的向量，并仅基于所涉及的单词的向量表示定义一个语言模型，使得这种语言模型的参数是单词的向量表示。因此，通过将模型拟合到一个特定的数据集，可以学习所有单词的向量表示。这种语言模型叫作神经语言模型，因为可以用神经网络来表示。例如，建模一个 n 元语言模型 $p(w_n|w_{n-1}, \cdots, w_1)$，这个神经网络以 $w_{n-1}, \cdots, w_1$ 作为输入，w_n 作为输出。在一些神经语言模型中，连接到单词的神经网络中的隐层可以被解释为单词的向量表示，其元素是连接到该单词的边上的权重。

例如，在 skip-gram 神经语言模型[Mikolov 等 2013]中，目标函数是利用每个词预测其上下文窗口中其他词，已知 w_2 预测 w_1 的概率是

$$p(w_1 \mid w_2) = \frac{\exp(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)}{\sum_{w_i \in V} \exp(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)}$$

其中，v_i 是 w_i 对应的向量表示。也就是说，在这个模型中，概率 $p(w_1 \mid w_2)$ 与两个词 w_1 和 w_2 的向量点积成正比。模型能够为每一个词学习到一个向量表示，使得利用当前词预测小窗口周围其他词的概率最大化。实际上，模型希望学习到的向量有这样的特点，如果两个词语义相关，即它们经常共同出现在同一个窗口中，两个向量进行点积也能够得到更大的值。

Google 基于 skip-gram 模型实现的 **word2vec**[Mikolov 等 2013]工具在这个领域是最著名的。它们证明了在向量空间对词向量进行相加的操作能够得到一些有趣的结果。例如，将 Germany 的词向量与 capital 的词向量相加，能够得到一个与 Berlin 非常相近的向

量。图 14-8 展示了由这个工具产生的结果实例。虽然类似的结果也可以通过启发式的聚合关系发现(例如，使用在 13 章描述的方法)以及 n 元类语言模型得到，但是词嵌入带来了一种新的解决思路，它可以灵活地定义优化的目标函数，并且通过优化明确定义的目标函数系统地学习向量表示，从而开辟了许多有趣的文本挖掘新应用。在当前版本中，词嵌入的一个缺点是，词的向量表示中的元素是没有意义的，很难进行解释。因此，利用这些词向量的应用还仅局限于词的相似性计算，而这一点通过许多其他方法也能实现。

词	与“france”的余弦相似度
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

图 14-8　基于 word2vec 找到与查询词“france”最相似的词语。(来自 Mikolov 等[2013])

293

总结一下，这一章介绍了计算词项相似度以及词项聚类的几种方法。首先介绍了一元语言模型，然后是点互信息的方法。简要介绍了两种基于模型的方法，一种是基于 n 元语言模型，一种是基于神经语言模型的词嵌入方法。这些词项聚类方法可以通过允许非精确词项匹配(例如，允许同一个聚类簇中的词或具有较高相似度的词相互“匹配”)来改进文档或其他文本对象之间相似度的计算。

14.4　文本聚类的评价

所有聚类方法都试图最大化以下度量指标。

凝聚度(coherence)：同一个簇内的对象有多相似？

分离度(separation)：不同簇的对象距离有多远？

效用性(utility)：发现的聚类簇对于一个应用多有用？

对于大多数文本挖掘(或其他)任务，可以通过以下两个策略中的任何一种：人工评价(使用人力)和自动评价(预先定义指标)。在上面提到的三点中，凝聚度和分离度可以通过计算向量相似度、纯净度或者互信息来实现。在词项聚类中有一个小的挑战是词之间的相似度算法并不像文档之间的相似度一样明显。因此可以把词表示成向量，然后利用文档相似度指标。或者基于其他已有的本体诸如 WordNet[⊖] 中语义相似度的概念。如果最终的系统输出能够量化地衡量，效用性也可以计算。例如，如果聚类结果被用于搜索中，可以查看不同聚类算法对于搜索结果 F_1、MAP 或 NDCG 的提升(见第 9 章)。

所有聚类方法都需要一些相似度(或偏置)的概念。毕竟，聚类的目标是希望找到在某种程度上一组彼此相似的对象。虽然在这本书中已经介绍了许多不同类型的特征，这里主要讨论了一元词的表示。实际上，特征工程是实现聚类算法中很重要的一个模块，同时也是大多数文本挖掘算法中重要的一步。为文本选择正确的表示能够量化两个对象被分配到相同或不同簇的差异。即使某聚类算法在簇内的相似度方面表现出色，但是如果不使用适当的特征表示，簇的形成可能无法被人们接受；如果特征表示不能够捕捉到关键的概念，特征将会被重新选择。第 4 章概述了许多由 MeTA 提供的不同的文本特征。在下一章的文本分类(第 15 章)中，还将继续讨论如何选择对于分类精度较为重要的特征。

294

⊖ https://wordnet.princeton.edu/

正如在本章中所看到的，基于相似度的算法在实现中明确地编码了相似度函数。理想情况下，对象之间的相似度的优化目标是最大化簇内的凝聚度和最小化簇内的分离度。在基于模型的方法中(第 17 章将会介绍)，相似度函数并不是模型的一部分。相反，对象相似度的概念经常以较高概率共同出现在相似对象中的词来表示。

通过利用分类的数据集可以自动完成凝聚度和分离度的测量。数据集需要提前定义好每个文档所属的类别。例如，一个文本分类的语料库可以是在线零售商的产品描述，每个产品都属于一个产品类别，如 kitchenware、books、grocery 等。好的聚类方法应该是能够基于产品的文本描述把产品划分到其预先定义好的类别中。一种简单的评价方式是考察每一个输出的簇的大多数是否受某个预先定义好的类别支配。换言之，给定每一个簇 C_i，计算簇内所占预定义类别的比例。对于一个簇 C_i 来说，如果聚类算法是有效的，则该簇由一个预先定义好的类别占主导，其对象很少被分配到其他簇内。实际上，聚类算法相当于在没有任何监督信息的情况下在原始数据集上重新进行类别分配。当然，这里需要注意的是需要设定最终的簇的个数要与类别数相匹配。

实际上，决定最优的簇的个数对于所有方法来说都是很难的问题。例如，在 K-均值中，最终的聚类簇取决于最初随机初始化的位置。通常采用多次运行算法或者人工检查结果。算法 G-均值[Hamerly 和 Elkan 2003]是在 K-均值基础上更加严谨的一种方法，如果分配给每个簇的数据不是正态分布，则分裂该簇。基于模型的方法在确定最优簇个数方面 295
可能有一些优势，但是模型本身通常不够准确。在实践中，可以根据需要或领域知识将簇的数量设置为固定值。

哪个方法好取决于偏置(相似度的定义)是否正确地反映了对聚类的期望以及方法的假设是否适用于问题和应用。通常，基于模型的方法通过将更多的约束加入概率模型中，可以做更多的“复杂聚类”。

书目说明和延伸阅读

聚类技术在数据挖掘中是一种非常通用的技术，在任何一本关于数据挖掘的书中都有详细介绍[Han 2005]，[Aggarwal 2015]。在 Aggarwal 和 Zhai[2012]的文本聚类一章，对许多聚类方法进行了综述。Steinbach 等[2000]中对一些文本聚类技术进行了实验性的比较。词语聚类与第 13 章讨论的词语关联关系挖掘十分相关。Kleinberg[2002]是一个非常有趣的理论工作，表明没有任何聚类能够满足一些期望的性质(即聚类算法的不可能理论)。

练习

14.1　将检索结果进行聚类从而有助于浏览是这一章中聚类的一个重要应用。如果考虑到简洁、有效以及运行时间等问题，在你的搜索引擎中，你会选择哪一种聚类方法？

14.2　假设用户从一组小的簇开始，并不断地改进(或回溯)他们的路径以搜索有趣的信息。你会使用哪种类型的聚类算法来支持用户浏览语料库？

14.3　聚类的簇数量对聚类算法的输出起着重要的作用。哪一种聚类算法，簇的数量在整个运行时间中扮演着重要的角色？

14.4　聚类标记是一个非常活跃的研究领域。想一想在文档聚类或词项聚类中如何给簇分配标签。一个好的聚类标记算法可能包含一些基于词语或簇统计的公式。 296

14.5　考虑本章介绍的三种簇与簇间相似度计算方法，它们中哪一个最有可能形成文档的“链条”，而不是更紧密的组？为什么？

14.6　利用 MeTA 实现 K-均值或凝聚层次聚类。通过对已经分词的文档进行聚类，使得算法具备较好的通用性，适用于任何词袋表示方法。

14.7　在给定一个树状的层次聚类数据的情况下，设计一个启发式设置聚类中簇的数目(和内容)的算法。尽量使算法只在树上进行一次遍历。

14.8　计算语义相关性时，在主题语言模型中利用狄利克雷先验和 Jelinek-Mercer 平滑重新定义评分函数，即对下面函数中的 $p(w|C)$进行平滑：

$$\text{score}(w)\frac{p(w|\text{computer})}{p(w|C)}$$

14.9　(未归一化的)PMI 的最大值和最小值分别是什么？

14.10　本章讨论了 PMI 和 nPMI 的一个潜在缺点。能否用一些预处理来帮助解决这个问题？例如，是否可以设置窗口中单词类型、每个词的最小或最大频率以及窗口本身的阈值？

14.11　可以使用什么类型的索引结构来有效地存储单词向量？假设每个向量中的值分布是稀疏的。

14.12　设计一种基于多个特征类型的文档聚类方法。第一种情况下，考虑利用一元词和一元词性标签进行聚类。更进一步的情况下，考虑通过单词、句子长度和结构句法分析树特征进行聚类。提示：在 MeTA 中如何实现？

14.13　第 11 章介绍了协同过滤。一个问题是，当用户或项目数量很大时，它不能很好地扩展。提出一个使用聚类的解决方案，使得能够在许多用户同时需要推荐时提供更快的运行速度。

297
~
298

文本分类

15.1 引言

前面章节讨论了如何进行文本聚类——基于相似的特征将文档组织起来。聚类技术是无监督的，并且具有不需要任何人工操作、可以适用于任何文本数据的优点。但是，我们常常希望按照一组预定义类别将文本对象按特定的方式进行分组。例如，新闻机构可能希望将新闻文章分类成一个或多个主题类别，如技术、体育、政治或娱乐等。如果使用聚类技术来解决这个问题，我们可以获得许多主题相关的簇，但是这些簇却并不一定与新闻机构预先设定(符合其应用需求)的类别相对应。可以使用广泛应用的文本分类技术来解决这个问题。

通常，文本分类问题是这样定义的。给定一组预先设定的类别，可能形成一个层次，同时还存在一组带标签的文本对象(即已知类别标签的文本对象)，文本分类的任务是给未出现的文本对象赋予一个或多个类别标签，如图 15-1 所示。

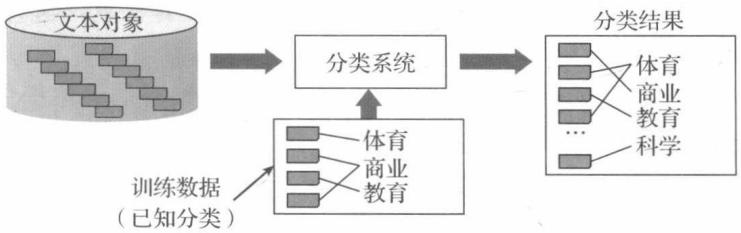


图 15-1 文本分类任务(带有训练实例)

从较高层次而言，文本分类通常有助于实现以下两个应用目标。

1) 丰富文本表示(即更加深入理解文本)：通过文本分类，能够在多个级别(关键字+类别)上表示文本。在这样的一个应用中，我们也称文本分类为文本注释(text annotation)。例如，分配给文本的语义类别对于应用来说是直接有用的，如垃圾邮件检测。分配给文本数据的语义类别还能够以更有意义的方式促进文本内容的聚合。例如，情感分类能使对产品的所有正面/负面意见聚合起来，从而对意见进行更全面的评估。

2) 推断与文本数据关联的实体的属性(即知识发现)：只要一个实体以某种方式与文本数据关联，就可以使用文本数据来帮助对关联的实体进行分类。例如，可以使用一个人写的英语文本数据来预测此人是否是非英语母语者。另一个例子是利用政治演说预测政治立场。当然，在这种情况下，文本分类的任务要困难得多，因为类别和文本内容之间的“鸿沟”很大。事实上，在这种应用中，文本分类应该被称为基于文本的预测(text-based prediction)。

这两个稍有区别的目标也可以根据分类类别的不同来区分。为了丰富文本表示，分类类别往往是描述文本对象的“内部”类别(例如，主题类别、情感类别)。为了利用文本数据推断关联实体的属性，分类类别往往是“外部”类别，它刻画与文本对象相关联的实体(例如，可能通过间接链接产生的作者属性或与文本数据相关的任何其他有意义的类别)。

然而，这些不同类型分类在计算上都是相似的，因为输入是文本对象，输出是一个或多个类别。因此，没有进一步区分这些不同的形式。

文本分类对象可以是多样的，包括文档、句子、段落或文本集合，因此，文本分类的应用前景更加丰富。

15.2　文本分类方法概述

300 当没有可用的训练数据时(即带有明确标记的已知类别的文本数据)，经常需要手动创建启发式规则来解决分类问题。例如有这样一个规则：如果“governor”出现→则分类标签为政治。显然，设计有效的规则需要对分类的特定问题有大量的知识。

这种以规则为基础的方法在以下情况下效果较好：1)类别的定义非常明确(即意味着分类比较简单)；2)根据文本中的表面特征很容易区分类别(例如，特定的词仅出现在特定类别的文档中)；3)有足够的领域知识可以提供许多有效的规则。

然而，这种人工的方法有一些明显的缺点。首先，消耗人力，不能很好地扩展类别的数量(因为新的类别需要新的规则)和适应数据的增长(新数据也可能需要新的规则)。其次，现实中不可能提出完全可靠的规则，难以处理规则中的不确定性。最后，规则可能并不完全一致。因此，分类结果可能取决于不同规则的应用顺序。

以规则为基础的人工方法存在的问题大多可以通过使用机器学习来解决，人们通过标注一些带有正确类别标签的实例(即创建训练实例)帮助机器，同时，机器会一定程度地从这些例子中自动构建分类规则，只是这些规则是软规则且带有权重，这些规则如何结合也是基于训练数据集学习的。注意，虽然在这种有监督的机器学习方法中分类似乎是“自动”的，但是在构造训练数据时是需要人工的，除非训练数据是天然提供给我们的(有时会发生)。人工创造的规则，如果有的话，也可以作为机器学习方法的特征，它们将以加权的方式相结合，通过最小化训练数据上的分类误差自动学习权重。在决策树中，机器还可以基于人类提供的原始特征自动构建规则[Quinlan 1986]，可以理解为基于规则的分类器，然而从根到叶子的路径(即规则)是通过机器学习自动引入的。一旦一个分类器被训练好，它可以用于对任何之前未见过的文本数据进行分类。

一般来说，所有这些基于学习的分类方法都依赖文本对象的特征来区分类别，它们将以加权方式组合多个特征，特征的权重是自动习得的(即调整并最小化训练数据上的分类
301 错误)。在训练数据中，不同的方法计算误差有所不同，即它们优化不同的目标函数(也叫损失函数)以及对于特征有不同的组合方式(例如，线性或非线性)。

在本章的其余部分中，我们将更详细地讨论基于学习的方法。这些自动分类方法一般分为三类。懒人学习(lazy learner)或基于实例的分类器(instance-based classifier)不显式地对类标签进行建模，而是将新实例与前面看到的实例进行比较，通常使用相似性度量。这些模型由于缺乏明确的泛化或训练步骤而被称为“懒惰”；大多数计算是在测试的时候进行的。生成式分类器(generative classifier)对每个分类构建数据分布(如，每个类别对应一个一元语言模型)。它们根据在每个分布下可观测到的对象的似然来对对象进行分类。判别式分类器(discriminative classifier)计算可以提供对象应该属于哪个类别的线索的文本对象特征，并将它们与参数结合起来控制它们的权重。通过最小化训练数据的分类错误来优化参数。

与聚类一样，在前面几章中讨论的许多技术同样能够用来创建分类器，这些算法基于已出现的标记数据为未出现的数据分配标签。本章首先解释了分类问题的定义。接下来，

讨论哪些类型的特征(文本表示)经常用于分类。然后，我们研究了几种常见的学习算法，这些算法可以用前向索引和倒排索引实现。之后，介绍如何进行分类的评价，这个问题与搜索引擎的评价有本质的区别。

15.3　文本分类问题

下面按照直观上对文档分类的理解，将第 2 章中的例子改写成更为数学化的形式。令文档集合为 X；它们可以存储在一个前向索引中(见第 8 章)。$x_i \in X$ 是用于表示文档 i 的词项特征向量。和检索的设置一样，每一个 x_i 的维度 $|x_i| = |V|$，每个维度是一个特征，通过分词器分配。第 8 章中的向量正是这样的一个词汇表大小只有 $|V| = 8$ 的向量。

{mr.：1，quill：1，'s：1，book：1，is：1，very：2，long：1，.：1}

如果文档 x_j 含有文本 long long book，那么

{mr.：0，quill：0，'s：0，book：1，is：0，very：0，long：2，.：0}　302

在前向索引中，存储如下：

$$x_i = \{1, 1, 1, 1, 1, 2, 1, 1\} \quad x_j = \{0, 0, 0, 1, 0, 0, 2, 0\}$$

所以 x_{ik} 是第 i 个文档中的第 k 项。

同时还存在 Y，它是每个文档的标签向量。因此，y_i 可能是新闻文章分类中的 sports，而 y_j 可能是 politics。

分类器是指一个函数 $f(\cdot)$，把文档向量作为输入，输出预测的标签 $\hat{y} \in Y$。因此得到 $f(x_i) = \text{sports}$。在这个例子中，$\hat{y} = \text{sports}$ 而真实值也是 sports，分类器的这个预测是正确的。

请注意，只有在知道数据的真实标签的情况下能够评价分类算法。事实上，必须使用真正的标签来学习一个好的函数 $f(\cdot)$来接受未出现的文档向量并对它们进行分类。由于这个原因，经常将语料库 X 分成两部分：训练数据(training data)和测试数据(testing data)。训练部分用于构建分类器，测试部分用于评价性能(例如，有多少标签被正确预测)。

但是函数 $f(\cdot)$实际上做了什么呢？考虑一个简单的例子，判断一篇新闻是正面还是负面情感，即 $\boldsymbol{Y} = \{\text{positive}, \text{negative}\}$：

$$f(x) = \begin{cases} \text{positive} & \text{如果单词 good 在 } x \text{ 中出现次数大于 1} \\ \text{negative} & \text{否则} \end{cases}$$

当然这个例子很简单，但是它证明了分类器的基本思想：将文档向量作为输入，同时输出类别标签。基于训练数据，分类器可以判断正面情感的文章通常包含词语 good 一次以上。因此，这种知识在函数中体现出来。在本章后面，将研究一些基于训练数据创建函数 $f(\cdot)$的特定算法。

还需要注意的是，这些学习算法有几种不同的方式。在二元分类(binary classification)中只有两类，根据分类器的类型，它可能只支持区分两个不同的类。多元分类(multiclass classification)可以支持任意数量的标签。可以结合多个二元分类器来创建多元分类器。回归(regression)是一个与分类非常相关的问题；它给出的是某范围内的实数值，而不是离散标签。例如，一个回归问题可以是给定前几年的降雨量，预测特定一天的降雨量。输出 $\hat{y}$ 是一个非负的值，例如可以用英尺来表示降雨量。另一方面，分类变量可以预测是否会有降雨，即 $\boldsymbol{Y} = \{\text{yes}, \text{no}\}$。　303

15.4　文本分类的特征

第 6 章强调了文档表示对于检索性能的重要作用。在第 2 章，我们总体上强调了特征

表示的重要性。在文本分类中甚至更加重要。假设想确定一份文档是否有正面或负面的情绪。显然，糟糕的文本表示方法可能是平均的句子长度。也就是说，文档词向量是每个文档的句子长度的直方图。直观地说，句子长度并不能很好地反映情绪。即使最好的学习算法也不能根据句子长度区分文档的正面和负面情绪。㊀

另一方面，假设任务是基本的作文评分，其中 $Y=\{\text{fail}, \text{pass}\}$。在这种情况下，句子长度确实可以作为作文质量的一个指标。虽然不完美，但是可以想象，这样的分类器要比基于句子长度预测情感分类的效果好。㊁

接下来，举一个更实际的例子，还是回到情感分析问题。使用一元语法词汇表示，而不是使用句子的长度。也就是说，每个特征都可以用来区分正面或负面情绪。通常，大多数特征都是无用的，大部分的决策基于更小的特征子集。确定这个较小的子集正是特征选择的定义，但在这一点上我们不作深入讨论。

虽然可能有效，但是一元语法词汇表示也不是最好的表示。考虑上一节中的分类器例子中提到的词 good 和 bad。在这种场景中，上下文是非常重要的：

I thought the movie was good.

I thought the movie was not bad.

或者

I thought the movie was not good.

304 I thought the movie was bad.

显然，二元词项表示可能是最好的特征表示方法，因为捕捉 was good 和 was bad 的同时，也可以捕捉 not good 和 not bad 的特征。

作为一个反例，只使用二元词项会导致错过一些不常见但是对任务有益的词，如 overhyped。而二元词项可以捕捉的词语特征诸如 overhyped(period)和 very overhyped。如果同样罕见的有益词出现在不同的上下文中，例如 was overhyped，则视为词汇表外的新词，并且不能用于判断句子的极性。出于这个考虑，比较常见的做法是把多个特征集组合在一起。在这种情况下，可以将文档基于一元词项和二元词项进行切分。

Stamatatos[2009]的研究策略表明，低层次的词汇特征结合高层次的句法特征能够给出分类器最好的性能。这两种特征更为正交，从而捕捉文本的不同视角，丰富特征空间。有许多不同类型的特征允许分类器在很大的空间上创建不同类标签之间的决策边界。

Massung 等[2013]使用了更高级的特征，请看图 15-2 中再现的第 4 章讨论的语法解析树。

在这里，我们看到三个版本的“高级”句法特征。左下角方框显示重写的规则；这些是句子中包含句法结点类别的语法分析结果。例如，S 表示句子，它由一个名词短语(NP)后面接一个动词短语(VP)以及一个句号来组成。图 15-2 中间的方框中隐去了除了子树根结点之外的所有结点。这就形成了更加抽象的注重结构的生成规则。最后，右边方框则是一个完全抽象的特征集，没有任何语法类别标签。作者发现这些结构化的特征结合低层次的词汇特征(即单字词)相对于单一特征类型能够帮助提升分类的精确度。

在 Massung 和 Zhai[2015]中提出了另外一种特征生成方法 SYNTACTICDIFF。它的核心思想是定义三个基本(通用)的编辑操作：插入一个词、移除一个词以及用一个词代替

㊀ 假设作为一个练习，尝试为 MeTA 创建一个文档分析器，使用句子长度为特征。能够得到准确率较高的情感分类吗？

㊁ 同样，使用 MeTA 来尝试这个实验，采用同样的句长特征。

另一个词。这些编辑操作用于转换给定的句子。给定一个源句子 S 以及参考文本集合 R，使用编辑操作使得 S 看似出现于 R 中。在非母语文本分析的场景中[Massung 和 Zhai 2016]，对来自于非母语作者的文本进行处理。因此，将按照 R 对 S 进行转换可以看作是一种语法错误纠正。

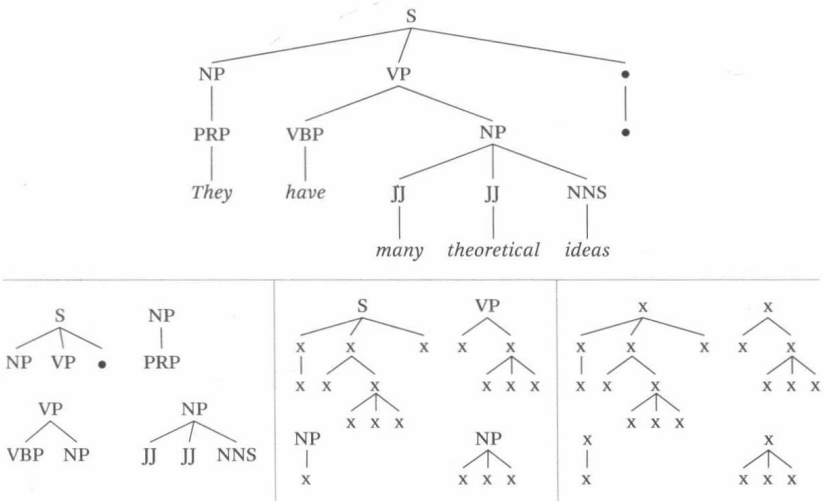


图 15-2　语法解析树及其派生的不同特征表示。对于每个特征类型，特征向量中的每个维度将对应于特定解析树结构的权重

虽然这本身是一个特定的任务，但对每个句子执行的一系列编辑也可以用来表示句子本身。例如，

{insert(the)：3，substitute(a→an)：1，…，remove(of)：2}

可以表示为一个句子的特征向量。这只是除了词袋特征之外的一个特征表示的例子。这些“编辑特征”的有效性决定了分类器在学习模型中如何有效地将不同的类分开。在这个例子中，这些特征可以用来区分文章作者的不同母语。同样，值得强调的是，几乎所有的机器学习算法都不受所引入的特征类型的影响(在操作方面，可能会影响精度)。因为在机器学习算法内部会把每个特征对应成一个 ID，算法不会知道它是在处理语法解析树、词或是二元词性标签还是编辑特征。

在第 3 章讨论的 NLP 流水线以及第 4 章分词中给出了有效特征表示的一些例子。通常，单个词是默认的方法，在必要时加入更先进的方法来提高精度。引入更先进的技术也带来了运算时间的需求。在使用语法解析树特征时，解析器必须在整个数据集上运行，这往往至少比只需要空格来切分的单个词特征在时间上慢一个数量级。运行语法分析需要句子进行词性标注，而指代消解需要语法解析树。句法或语义特征的复杂程度通常取决于对处理时间和内存使用的容忍度。

305
〜
306

15.5　分类算法

在本节中，将深入研究函数 $f(\cdot)$如何区分不同的类标签。这里将介绍三种不同的算法，所有这些算法都在 MeTA 中实现了。继续以情感分析为例，将新文本分类为 positive 标记或 negative 标记。假设将数据集分割成训练集和测试集。训练集将用于构建 $f(\cdot)$，而在每个测试文档上可以评价它的性能。

同时，对于所有文档的元数据信息 Y 都是已知的，所以所有测试和训练数据的真实标

签是已知的。在实际应用时，真正的标签是未知的(除非人工指定一个)，但是，基于算法在测试数据上的性能，可以对算法的预测有一定的信心。测试数据用于模仿真实世界中未出现的数据。测试数据越接近随机的真实数据，对分类器精度的信心就越大。

15.5.1　k-近邻

k-近邻是一个直接利用倒排索引和搜索引擎的学习算法。与下面将讨论的其他两种算法不同，k-近邻没有明确的训练步骤；所要做的就是对训练文档进行索引。这使得 k-近邻是一个懒惰学习或基于实例的近邻分类器。

如训练和测试算法所示，k-近邻的基本思想是找出与查询文档最相似的文档，并使用相似文档中最常见的类标签。它的假设是相似的文档具有相同的标签。图 15-3 给出了在文档向量空间中使用 k-近邻的实例。这里有三个不同的类，表示在向量空间中绘制的不同颜色。如果 $k=1$，将红色标签分配给查询；
307 如果 $k=4$，将分配蓝色标签，因为在前四个文档中有三个是蓝色的。对于有平局的情况，将根据平局的类别中排名最高的文档进行选择。

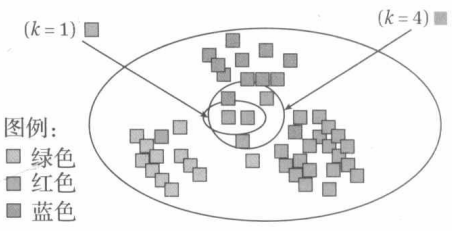


图 15-3　三类 k-近邻举例，$k=1$，4。白色正方形表示查询

算法 15-1　k-近邻训练

Create an inverted index over the training documents

算法 15-2　k-近邻测试

Let R be the results from searching the index with the unseen document as the query
Select the top k results from R
return the label that is most common in the k documents via majority voting

k-近邻可以应用到任何距离度量和任何文档表示。只要稍加修改，就可以利用现有的倒排索引直接使用分类方法。不需要前向索引。尽管 k-近邻有许多优点，但它还存在一些缺点。首先，查找最近的邻居需要对每个测试实例执行搜索引擎查询。即使已经进行优化操作，但在测试时它仍然比其他机器学习算法慢得多。正如将看到的，其他两个算法在查询向量上执行简单的向量操作，而不是查询倒排索引。然而，这些其他算法的训练时间比 k-近邻要长得多，这就是折中。再者，k-近邻的分类标签获取只取决于其 k 个邻居。相反，其他两个算法则考虑全部训练数据。因此，k-近邻对 k 个文档构成的特
308 征空间的局部结构很敏感。如果恰巧有与查询相近的不同类别的离群值，将会被错误地分类。

有几种基于 k-近邻框架的变体。例如，可以根据当前查询与邻居的距离进行权重的投票，从而得到权重 k 近邻(weighted k-nearest neighbor)。意味着与当前查询接近的邻居会有更大的影响或者更高的权重。一个比较简单的权重方法是给每个近邻乘以一个 $\frac{1}{d}$，其中 d 表示与查询的距离。因此，距离越远的邻居对预测的标签影响越小。

另一种变体是最近质心分类器(nearest-centroid classifier)。在该算法中，不使用单个文档作为邻居，而是考虑每一类标签的质心(关于质心和聚类的更多信息参见第 14 章)。这里，如果有 n 个类，只需看当前查询到 n 中的哪一个最接近。每个类别标签的质心可以

看作是一个原型，或者是来自于那个类的一个理想的文档。同时在性能上也有优势，因为只需要做 n 次相似度比较，而不是针对所有训练文档执行一个完整的搜索引擎查询。

15.5.2　朴素贝叶斯

朴素贝叶斯是生成式分类器的一个例子。除了类标签本身的分布之外，它还会在每个类标签上创建特征的概率分布。这与语言模型主题概率计算非常相似。使用语言模型，为每个主题创建一个分布。当看到一个新的文本对象时，使用现有的主题去找到最有可能产生的主题语言模型$\hat{\theta}$。回顾第 2 章，

$$\hat{\theta} = \arg\max_{\theta} p(w_1, \cdots w_n | \theta) = \arg\max_{\theta} \prod_{i=1}^{n} p(w_i | \theta) \tag{15-1}$$

算法 15-3　朴素贝叶斯训练

Calculate $p(y)$ for each class label in the training data
Calculate $p(x_i \mid y)$ for each feature for each class label in the training data

算法 15-4　朴素贝叶斯测试

return the $y \in \mathbf{Y}$ that maximizes $p(y) \cdot \prod_{i=1}^{n} p(x_i \mid y)$

309

朴素贝叶斯分类器看起来与之非常相似。本质上，存在一个特征分布 $p(x_i | y)$，其中 x_i为特征，y 为类别标签。给定一个未出现的文档，可计算生成它的最有可能的类分布。也就是说，要为每个标签 $y \in Y$ 计算 $p(y|x)$，利用第 2 章的贝叶斯法则，给定文档 x，把它改写成以下形式：

$$\begin{aligned}
\hat{y} &= \arg\max_{y \in \mathbf{Y}} p(y | x_1 \cdots, x_n) \\
&= \arg\max_{y \in \mathbf{Y}} \frac{p(y) p(x_1 \cdots, x_n | y)}{p(x_1 \cdots, x_n)} \\
&= \arg\max_{y \in \mathbf{Y}} p(y) p(x_1 \cdots, x_n | y) \\
&= \arg\max_{y \in \mathbf{Y}} p(y) \prod_{i=1}^{n} p(x_i | y)
\end{aligned} \tag{15-2}$$

注意，这里消除了贝叶斯法则产生的分母，因为它不会改变取最大值的变量的结果。最后的简化是独立假设，即特征之间没有互相依赖，因此得到联合概率时，可以简单地把所有的概率相乘。这正是朴素贝叶斯被称为朴素(naive)的原因。

我们需要估计以下分布：所有类的 $p(y)$以及每个类别特征的 $p(x_i | y)$。这种估计方法与一元语言模型的估计方法是一样的。也就是说，最简单的推理方法是最大似然估计，计算某个类中出现的特征除以其总发生的次数。正如第 2 章所讨论的，这可能会导致一些未出现的词或稀疏的数据。在这种情况下，可以使用我们第 6 章中讨论的任何平滑方法来平滑估计的概率，例如狄利克雷先验平滑和 Jelinek-Mercer 插值。最后，我们需要计算$p(y)$，这正是每个类别标签的概率。当类别标签不平衡时，这个参数是必不可少的，因为我们不希望在模型的预测结果中，那些仅在训练集中出现几次的标签与那些流行的标签占据相同比例。

k-近邻大部分计算时间消耗在测试时，而朴素贝叶斯则把时间用在训练模型的参数估计。在测试中，进行 $|Y|$ 次计算，找到最有可能的标签。在学习参数时，使用前向索引，因此知道哪些属性对应哪些类别标签；也就是说，查找每个文档中的计数，并更新该文档真正的类别标签的参数。这种用法不需要倒排索引。除了前向索引外，内存需要存储参

310

数，它可以表示为$O(|Y|+|V|\cdot Y|)$个浮点数。

由于其简单性和强独立性假设，朴素贝叶斯常常不敌更复杂的分类方法，其中一些方法基于下一节讨论的线性分类器。

15.5.3　线性分类器

线性分类器本质上是二元分类器。考虑下面的线性分类器：

$$f(x)=\begin{cases}+1 & \text{如果 } w\cdot x>0\\ -1 & \text{否则}\end{cases} \tag{15-3}$$

将未出现的文档向量和权重向量w之间进行点积(其中$|w|=|x|=|V|$)。此时，训练线性分类器的权重向量w，将其与文档向量进行点积，产生一个大于0的值则判为正例，而小于0的则判为负例。有很多这样的算法，包括最先进的支持向量机分类器[Campbell 和 Ying 2011]。

这组学习算法又叫线性分类器，因为该算法的决策基于特征权重的线性组合(w和x)。图15-4显示了点积如何在一
311 个简单的二维实例中，创建两个标签组之间的决策边界。图中展示了两个可能的决策边界。垂直的那条线刚好区分两个类，而另一个在两个类之间有很大的边界。上一段提到的支持向量机算法试图最大化决策边界和两个类之间的界限，这样就给新的实例留下了更多的“空间”来正确分类，因为它们将会落在同一类实例附近的决策边界的一边。

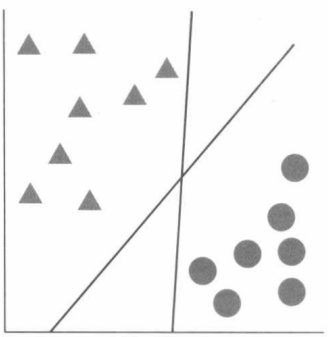


图15-4　二元线性分类器在二维空间中产生的两个决策边界

算法15-5　感知机训练

```
w ← {0, . . . , 0}
for iteration t ∈ T do
    for each training element i do
        ŷ = w · x_i
        w_j = w_j + α(y_i − ŷ) · x_ij, ∀j ∈ [0, |V|]
    end for
    break if change in w is small
end for
```

当然，并不是所有的数据点都是线性可分(linearly separable)，所以创建决策边界的目的在于尽可能精确地区分两类。朴素贝叶斯也可以视作线性分类器。与之形成对比的是k-近邻，因为它只考虑了查询绘制的局部子空间，忽略了语料库的其余部分。还有一些更先进的方法，例如核方法将线性分类器改为非线性分类器，更详细的内容建议参考[Bishop 2006]。

在本书中，选择研究相对简单的感知机分类器(perceptron classifier)，它是其他线性分类器的基础。需要指定在训练算法中使用的几个参数。令T代表运行训练的最大迭代次数。令$\alpha>0$为学习率。学习率控制在每一步调整权重的多少。如果w的变化很小，可能提前终止训练；这通常是通过比较当前迭代的权重与前一个迭代的权重的范数来衡量的。

关于学习率的选择、收敛准则等的研究有很多，但在本书中并没有讨论这些问题。相反，希望使读者熟悉算法的一般思路，介绍算法实现和理论性质的更多细节请参见[Bishop 2006]。
312

算法 15-6　感知机测试

$\hat{y} \leftarrow w \cdot x$
return $+1$ **if** $\hat{y} > 0$, **else return** -1

正如算法描述的那样，在训练感知机分类器时，基于已知实例持续更新权重向量。如果 y_i 和 $\hat{y}$ 有同样的符号(正确分类)，权重不变。如果 $y_i < \hat{y}$，目标对象将被分为 -1，则从 w 中的每个活跃特征中减去权重。在相反的情况下($y_i > \hat{y}$)，将权重加到 w 中的每个活跃特征中。活跃特征是指在当前例子 x 中出现的特征，只有 $x_{ij} > 0$ 的特征能够对 w 的更新有贡献。

最后，w 在迭代几次之后的变化很小，这意味着该算法已经找到了最佳的精度。保存最终的 w 向量作为模型，它可以用来分类未出现的文档。

如果需要支持多类别分类呢？并不是所有的分类问题都能很好地分成两类。幸运的是，使用多个二元分类器创建一个 k 类的多元分类方法有两种常用方法。

One-vs-All(OVA)方法为每个类别训练一个分类器(共 k 个分类器)。每一个分类器针对它所对应的类别预测为 $+1$，对于其他类别预测为 -1。在这种情况下，如果有多个分类器预测为 $+1$，则会产生歧义。因此，线性分类器能够给出一个置信度分数作为预测。如 $+0.588$ 或 $+1.045$ 的置信度分数都表示“$+1$”标签，但后者比前者更“可信”，因此将选择算法预测 $+1.045$ 的类。

All-vs-All(AVA)方法训练 $k(k-1)/2$ 个分类器去区分 k 个类别。选择预测最多 $+1$ 的类作为最终答案。同样，基于信任的评分可以用来为每个类标签添加权重总数。

15.6　文本分类的评价

文本分类的评价与信息检索的评价一样，可以通过计算真阳性、假阳性、真阴性和假阴性来获得准确率、召回率和 F_1 值，通常更关注精确率(正确预测的数量除以总预测数)。 313

前面章节中提到了训练集和测试集的分割，但有时也使用另一种语料库的划分：即用于参数调整的开发集。通常，一个语料库被分成大约 80% 的训练集、10% 的开发集和 10% 的测试集。例如，如何确定 k-近邻中一个好的 k 值问题。在训练文档上创建一个索引，(例如)$k=5$。精确率是通过开发文档来确定的。然后设置为 $k=10$，15，20，25，以此重复。表现最佳的 k 值将会在测试集上跑出最后的精确率。开发集的目的是防止过拟合(overfitting)，或者对某一特定语料子集进行过多的学习算法剪裁，使其失去通用性。不容易过度拟合的训练模型具有健壮性(robust)。

另一种评价方法是 n 折交叉验证(cross validation)。将语料库分割成 n 个部分。在 n 轮中，选择一个分区作为测试集，剩余的 $n-1$ 个用于训练。最后的精确率、F_1 值或任何其他评价指标均在 n 轮平均后得到。n 轮之间的分数的变化可以看出算法是否容易过拟合。如果方差高，则表示各轮之间的分数不太相似。如果其中一轮具有非常高的精确率，表明学习算法可能在训练阶段过拟合；当使用训练好的模型在另一个数据上预测时，精确率很有可能会很低，因为该模型针对某一特定划分中的噪声或无用信息进行了建模(即过拟合)。

另一个重要概念是基线精确率(baseline accuracy)。表示使用分类器时需要“击败”的最小得分。例如有 3000 个文档，由三个类组成，每个集合有 1000 个文档。在这种情况下，随机猜测会得到大约 33% 的精确率，因为大约会有 1/3 正确的可能。这时，分类器必须做得比 33% 精确率才有用。再举个例子，有 3000 个文档和三个类，但是类分布不均匀：一个类有 2000 个文档，另外两个类有 500 个。在这种情况下，基线精确率为 66%，因为选择大多数类标签将产生 2/3 的正确预测可能性。因此，在评价分类器时，考虑类的不平衡是很重要的。

混淆矩阵(confusion matrix)是一种在每个标签级别检查分类器性能的方法。考虑图 15-5，在一个三类分类问题上运行 MeTA 的输出，以确定英文文本作者的母语。

	chinese	english	japanese
chinese	0.802	0.011	0.187
english	0.0069	0.807	0.186
japanese	0.0052	0.0039	0.991

图 15-5　MeTA 中三类分类问题的混淆矩阵输出

每个(行，列)表中的指数表示行被分类为列的百分比。因此，每行加起来总和为 1。对角线表示真阳率(true positive rate)，大部分的概率分布表明这是一个好的分类器。基于混淆矩阵，可发现预测母语为汉语的精
314 确率为 80.2%，母语为英语和日语分别为 80.7%和 99.1%。这表明，英语和汉语有着相同的难度，但日语很容易被分类器区分开来。还可发现，如果分类器在汉语到英语标签上的预测是错误的，它几乎总是选择日语作为分类答案。基于混淆矩阵还可以发现，分类器似乎默认为“日语”这个标签。这个表并没有表明为什么会这样，但是可以根据数据集来做一些假设。基于这一观察，可能需要调整分类器的参数，或者做更深入的特征分析。

书目说明和延伸阅读

Manning 等[2008]对文本分类做了广泛的研究，而更早期的研究在 Sebastiani[2002]文献中可见。最近 Aggarwal 和 Zhai[2012]的书中有一个章节专门介绍这个主题。Yang[1999]对多种常用文本分类方法进行系统的实证评价和文本分类评价的讨论。此外，由于文本分类通常使用有监督的机器学习来完成，所以任何有关机器学习的书都与之相关(如 Mitchell 1997)。

练习

15.1　在 15.4 节中，有两个关于句子长度特征生成的相关脚注。按照建议，在 MeTA 中实现这个分析器，看某个特定数据集类型是否受益于此方法。

15.2　使用 MeTA 进行文本分类的实验。哪一种 k-近邻的变体效果最好？k-近邻的精度对排序函数的依赖程度如何？

15.3　在 MeTA 中，SVM 被称为使用 hinge loss 的 SGD(默认分类器)。SVM 总是比朴素
315 贝叶斯和 k-近邻效果好吗？在 MeTA 中的运行时间与其他三个方法相比如何？

15.4　在文本分类中，通常有数千个甚至数百万个特征。这使得可能创建一个超平面，在这个高维空间中分离类对象。基于这些信息，请推荐一个默认的分类器用于文本分类，并说明原因。

15.5　在某应用设置中，必须在朴素贝叶斯或 k-近邻之间进行选择，对文本文档进行分类。如果应用程序需要很高的性能，需要很快地对文档进行分类。解释你在此设置中的分类器的选择。

15.6　请指出 k-近邻和朴素贝叶斯算法的一个相同和不同之处。

15.7　朴素贝叶斯中为什么叫“朴素”？为什么叫“贝叶斯”？

15.8　比如现有一个数据集和一个分类器。对分类器使用 5 折交叉验证和 10 折交叉验证进行评价。你认为哪个会取得更高的精确率？为什么？

15.9　文本分类的评价方式与信息检索的评价方式有何区别？

15.10　如何确定是否有足够的分类器训练数据？使用图来说明，其中 x 轴表示训练数据的大小，y 轴表示在未出现的测试数据上的分类精度。

15.11　解释如何解读混淆矩阵，以确定经常被分类器互相误分的两个类。

316 15.12　从混淆矩阵中能看出类别不均衡吗？请解释原因。

文本摘要

文本摘要是将相对较大量的文本数据或长文本文章压缩成更简洁的形式，以便于理解的任务。这对于文本数据获取显然是非常重要的，它可以帮助用户查看文本数据中的主要内容或要点，而无需阅读完所有文本。对搜索引擎的返回结果进行摘要是此应用的很好的一个例子。然而，摘要也可以用于文本数据分析，因为它可以帮助减少要处理的文本的数量，从而提高各种分析算法的效率。

但是，摘要是一个较为复杂的任务。给定一个大的文档，怎么能只用几句话来表达重要的观点呢？这里所说的“文档”和“重要”又是什么意思呢？尽管人类很容易识别好的摘要，但定义这个过程并不那么简单。简言之，对于任何文本摘要应用，我们希望对文本进行语义压缩；也就是说，我们希望在更少的空间中基本上传达相同量的信息。应用的输出应该是流利的，可读的语言。一般来说，我们需要有摘要的目标，尽管目标通常很难定义。一旦确定了目标，我们就可以开始设计完成任务的方法，问题也就容易评价了。

举一个具体的例子，我们以新闻摘要为例。如果输入是一天的新闻文章的集合，潜在的有效的输出是就是新闻标题列表。当然，这不会是所有的新闻标题，而只是那些用户感兴趣的新闻标题。从另一个角度，考虑新闻摘要任务，其输入是新闻的文本，输出应该是新闻中以可读格式解释文章的内容的某个段落。每个任务都需要不同的解决方案。

对检索结果进行摘要也是特别有意义的。在搜索引擎结果页面上，我们如何帮助用户点击相关链接？常见的策略是在短片段中突出显示与查询匹配的字词。另一种方法是用几句话来总结每个结果，并在结果页面上显示这个精简的摘要。以这种方式使用摘要可以让用户在决定阅读该文档之前更好地了解文档中包含了哪些信息。

观点摘要对企业和购物者都非常有用。对产品的所有评论进行摘要，让企业知道买家是否满意(以及为什么)。评论摘要也让购物者在网上搜索时能够对不同产品进行比较。评论可以进一步细分为正面评论摘要和负面评论摘要。Wang 等[2010]和 Wang 等[2011]描述了更细粒度的处理方法，本书将在第 18 章中使用主题模型对涉及产品不同层面的产品评论进行总结时进一步讨论。对于酒店评价，可能对应于服务、位置、价格和价值。虽然这两个任务的输出不是人类可读的摘要，我们可以想象一个系统，它能够为用户总结所有的英语的(或任何其他语言的)酒店评论。

在本章中，我们概述了两种摘要技术的主要范例，并对其不同的应用进行了研究。

16.1 文本摘要技术概述

文本摘要有两类主要方法。第一类是基于选择(selection-based)或抽取式摘要(extractive summarization)。使用这种方法，摘要由原始文档中选择的一系列句子组成。因为没有新的句子产生，因此这种摘要是抽取出来的。第二类方法是基于生成(generation-based)或抽象式摘要(abstractive summarization)。在这里，摘要可能包含不在任何原始文档中的新句子。在本章中探索的一种方法是使用语言模型。在本书前面的章节中，我们使用语言模型来计算一些文本的可能性；本章将演示如何使用语言模型反过来生成句子。在讨论抽

象摘要的方法时，我们还简要介绍了自然语言生成(natural language generation)。

按照前几章的模式，接下来将介绍文本摘要的评价。这两种方法都有侧重于各自实现的评价指标，但它也可以在一个抽取式算法生成的摘要上使用(例如)抽象摘要的评价指标。最后，本章将研究一些文本摘要的应用，并了解它们在现实系统中是如何实现的。

文本摘要是一个广阔的领域，本章只涉及其中的核心概念。如果想进一步阅读，我们建议读者从 Das 和 Martins[2007]开始，它提供了对该领域的系统概述，包含了本章很多
318 内容的扩展。

16.2　抽取式文本摘要

基于信息检索的技术使用句子向量和相似度函数创建摘要文本。句子向量在结构上等同于文档向量，尽管仅仅基于较少数量的单词。下面将概述一个基于信息检索的基本摘要系统。

1) 将文档分解为章节或段落。

2) 对于每个段落，将其句子“压缩”成较小数量的相关(但不是冗余)句子。

该策略保留了连贯性，因为摘要中的句子大多与原始文档中的顺序相同。

第一步如图 16-1 所示。按顺序遍历文档中的句子，对相邻的句子对应用归一化的、对称的相似性度量方法(见第 14 章)。该图右侧的部分显示了句子之间相似度的变化。可以检查这些变化以便在相似度较低时将文档分割成段落，即主题发生了偏移。这种分割的另一种方法是只使用段落，如果正在处理的文档包含该信息，尽管大部分情况下并非如此。这种基本的分割策略是篇章分析(discourse analysis)(NLP 的子领域)中的任务。篇章分析涉及的是句子序列，而不是只有一句话。

现在有了这些段落，如何在第二步中消除冗余并同时增加摘要结果的多样性？最大边际相关性(Maximal Margin Relevance，MMR)重排序技术可以应用于我们的问题中。本质上，该算法贪婪地重新排列了当前段落中的每个句子，仅输出了前几个作为摘要。图 16-2 显示了当我们只从每个段落中选择一个句子时算法的输出。

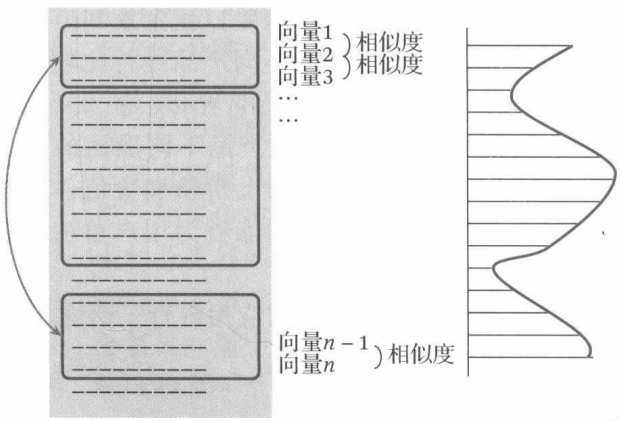


图 16-1　通过基于相似度的篇章分析方法将文档分割成段落

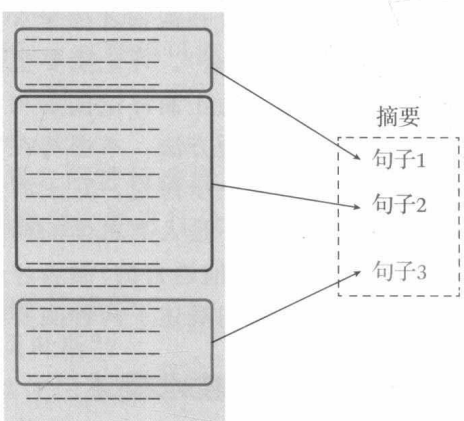


图 16-2　利用最大边际相关性从每个段落中选择一个句子作为摘要的文本摘要方法

MMR 算法如下。假设给定原始列表 R 和配置 p 来构造一组选定句子 S(其中 $|S| \ll |R|$)。R 是需要进行摘要的文档中的一个分区块。配置 p 确定“相关性”的确切含义。最初，MMR 公式被应用于从信息检索系统返回的文档(因此称为重新排序)。除了对于已

经选择的文档是非冗余的之外，文档是基于与查询的边际相关性(我们的变量 p)选择的。由于我们的任务处理的是句子检索，p 可以是用户描述文档(关于用户的文本)，整个文档本身，或者甚至可以是由用户制定的查询。

319 ~ 320

根据边际相关性，要添加到所选列表 S 中的下一个句子s_i被定义为

$$s_i = \arg\max_{S \in R/S} \{(1-\lambda) \cdot \mathrm{sim}_1(s, p) - \lambda \cdot \arg\max_{s_j \in S} \mathrm{sim}_2(s, s_j)\} \tag{16-1}$$

可以将 $R \setminus S$ 符号读为“R 集合减 S”，即在 R 中不在 S 中的所有元素。MMR 公式使用 $\lambda \in [0, 1]$来控制相关性与冗余度；正相关得分是由与已选择句子的冗余度(相似度)来折算的。同样，这两个相似性度量方法可以是任何归一化的、对称的度量方法。相似性度量的最简单方法是余弦相似度，这实际上是在 Carbonell 和 Goldstein[1998]中使用的度量方法。

一旦在 S 中有适当数量的单词或句子，或者如果分数 $sim_1(s, p)$低于某个阈值，则该算法终止。此外，相似度函数也可以被调整。你能想到在相似度函数中包含句子位置的方法吗？也就是说，如果一个句子距离候选句子很远(不相似)，我们可以从相似度得分中减去。更好的是，我们可以将这两个值插值到一个新的相似度得分中，如

$$\mathrm{sim}(s, s') = \alpha \cdot \mathrm{sim}_{\mathrm{cosine}}(s, s') + (1-\alpha) \cdot \left(1 - \frac{d(s, s')}{\max d(s, \cdot)}\right) \tag{16-2}$$

其中 $\alpha \in [0, 1]$控制正则余弦相似度和距离度量之间的权重，$d(\cdot, \cdot)$是两个参数之间的句子数。注意距离计算前面的“1 减”，因为较小的距离意味着更大的相似度。

当然，MMR 公式中的 λ 也可以设定。事实上，对于多文档摘要，Das 和 Martins [2007]建议以 $\lambda = 0.3$ 开始，然后缓慢增加到 $\lambda = 0.7$。其背后的原因是首先强调新颖性，然后默认为相关性。这应该使你想起在第 11 章中讨论的探索利用平衡。

16.3　抽象式文本摘要

抽象文本摘要创建原始文档或多个文档中不存在的句子。本章中将使用语言模型代替原始文本，而不是文档向量。与文档向量不同，语言模型提供了生成文本的原则性方法。想象一下，将文档切分为一元的词。在语言模型中，将有一个参数表示每个单词出现的概率。要创建我们自己的文本，我们将从这个概率分布中得到单词。

321

假如有一个从需要进行摘要的文档中估计得到的一元语言模型 θ。我们希望从 θ 中得到组成摘要的单词w_1，w_2，w_3，…。我们希望在摘要中出现w_i的概率与在原始文档中出现的概率大致相同——这就是为什么生成的文本近似于较长的文档。图 16-3 描述了完成这个任务的步骤。首先，创建一个所有参数的列表，并增量地对其概率加和；这使得我们能根据[0，1]区间上的随机数选择一个单词w_i。简单来说，我们得到一个从 0 到 1 均匀分布的随机浮点数。然后，迭代词表中的单词，将其概率相加，直到得到该随机数。输出该词并重复这个过程。

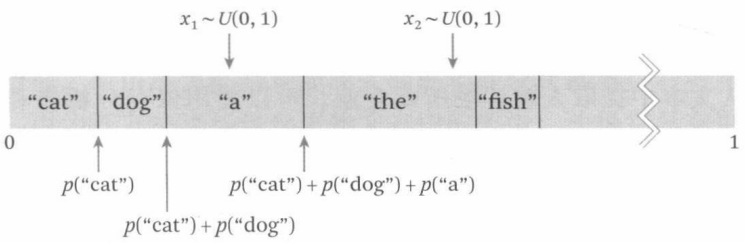


图 16-3　从一元语言模型中抽取词

在这个例子中，我们假设有以下值：

$p(\text{cat})$	0.010
$p(\text{cat})+p(\text{dog})$	0.018
$p(\text{cat})+p(\text{dog})+p(\text{a})$	0.038
⋮	⋮
$p(\text{cat})+p(\text{dog})+p(\text{a})+\cdots+p(\text{zap})$	1.0

假设使用[0，1]上的均匀分布生成一个随机数x_1。这被表示为$x_1 \sim \mathcal{U}(0, 1)$。现在想象$x_1=0.032$。在分布中找到累积概率0.032所处位置并输出“a”。我们可以重复这个过程，直到摘要达到指定长度，或者直到生成句子结束标记`< /s>`。

至此，你可能会认为我们生成的文本不会有任何意义——这当然是对的，因为如果仅使用一元语言模型，每个单词都是独立生成的，而不考虑其上下文。如果需要更流利的语言，我们可以使用n元语言模型，其中$n>1$。与每个单词独立生成不同的是，新单词将取
322 决于以前的$n-1$个单词。生成方式与一元语言模型的情况相同：比如有w_i这个词，希望用二元语言模型生成w_{i+1}。二元语言模型给出了已知w_i后单词的分布，我们通过与图16-3所示相同的方式得到下一个单词。

来自二元语言模型的句子生成如下：从(例如)The开始。然后，使用累积求和的方法从分布$p(w|\text{The})$中选择。下一个选择的单词可能是cat。然后，我们使用分布$p(w|\text{cat})$来找到下一个w，以此类推。一元模型只有一个“求和表”(图16-3)，但是二元语言模型需要V个表，$p(w|w')$中的每个w'都有一个表。

通常，根据原始数据的多少，n值取3～5。当n太小时发生了什么：我们得到了一堆杂乱无章的单词。但是如果n太大，还有另一个问题。考虑$n=20$的极端情况。给出19个单词，我们希望使用20元语言模型生成下一个单词。这19个词构成的序列不太可能在原始文档中出现多次。这意味着第20个词只有一种选择。因此，我们只是复制原始文档，这不是一个很好的摘要。在实践中，我们想选择一个足够大的n值以产生连贯的文本，但同时足够小到不是简单地重现语料库。

这种抽象摘要方法有一个主要的缺点。由于它的性质，给定的单词只取决于n个周围的单词。也就是说，生成的文本中不会有远距离依赖关系。例如，考虑从三元语言模型生成的以下句子：

They imposed a gradual smoking ban on virtually all corn seeds planted are hybrids.

所有三个词组都有意义，但总的来说这句话是难以理解的；似乎作者在句子中间把话题由禁烟改为了混合作物。在特殊情况下，当将高度冗余的文本的摘要的长度限制为几个字，这样的策略似乎是有效的，如Ganesan等[2012]所述的微观点(micropinion)摘要方法所示。

高级抽象方法

一些高级的抽象方法更多地依赖于自然语言处理来构建文档的模型来进行摘要。命名实体识别可用于从文本中提取人物、场所或企业。可以使用依存分析器和其他句法技术来
323 查找实体之间的关系及其发生的动作。一旦这些动作和角色被发现，它们将被存储在一些内部表示中。为了生成实际文本，从解析的集合中选择一些表示，并根据它们创建英文句子；这就是所谓的实现(realization)。

与上述基本抽象语言模型生成器相比，这种实现系统对生成的文本进行更细粒度的控

制。可能存在模板文档结构(如引言→第 1 段→第 2 段→结论)，选择文档结构并填充每个点。对文本摘要和布局的这种控制使得摘要更易于阅读，因为其具有自然的话题变化。在这种情况中，可以根据上下文将带有诸如“和”或者“但是”等连接词的相似的句子合并。为了使摘要更加自然，如果实体名已经被提到，可以使用代词代替实体名。以下是这两个操作的例子：

今天黄金价格下跌了。今天银价下跌。→今天黄金和白银价格下跌。

公司 A 今天亏损 9.43%。公司 A 的波动最大。→公司 A 今天亏损 9.43%。它的波动最大。

更好的是，

公司 A 今天波动最大，亏损 9.43%。

这些操作是可能的，因为实体以结构化格式存储。关于先进的自然语言生成，我们推荐阅读 Reiter 和 Dale[2000]，其重点是实用性和实现性。

16.4　文本摘要的评价

在抽取式摘要中，从文本段落中选出代表性的句子作为摘要。该解决方案被建模为信息检索问题，我们可以用类似的方法对其进行评估。冗余是一个关键问题，我们讨论的 MMR 技术试图减少冗余。在进行评价时，我们应该考虑冗余的句子是不相关的，因为用户不想两次读到相同的信息。有关 IR 评价指标的更详细说明，请参阅第 9 章。

对于全输出评分，我们应该优先采用不考虑结果位置的 IR 评价指标。虽然我们的摘要是通过对每个段落的句子进行排名生成的，但是由于原始文档由多个段落组成，所以整个输出并不是排名列表。因此，我们可以使用准确率、召回率和 F_1 得分。 324

可以使用位置相关度量(如平均准确率或 NDCG)对段落评分检索函数进行排名，但对最终输出则不可行。因此，我们需要决定到底是评价段落评分还是整个输出(或两者)。整个输出的评分可能对实际用户更有用，而段落评分则可能帮助研究人员对方法进行微调。

在抽象式摘要中，我们不能使用 IR 评价指标，因为没有固定的候选句子集合。如果不知道相关句子的总数，如何计算召回率？由于没有中间排序阶段，也不能使用平均准确率或 NDCG(再次，我们甚至不知道正确的句子的完整集合)。

一个费力但是准确的评价方法是让人类标注者创建一个黄金标准的摘要。这个“完美的”摘要将与生成的摘要进行比较，并且一些度量(例如，ROUGE)将用于量化它们的差异。对于比较指标，我们有许多可能性——任何能够比较两组文本的指标都可能适用。例如，我们可以使用黄金标准和生成摘要之间的余弦相似度。当然，这样做的缺点是流畅性被忽略(使用单字)。另一种方法是在黄金标准摘要上学习一个 n 元语言模型，然后计算生成的摘要的对数似然。这可以确保在 n 元层次的语言流畅度的基本水平，同时也产生一个可解释的结果。其他比较两种概率分布的方式也同样适用，如 KL 散度。

摘要的总体有效性可以通过让用户阅读摘要，然后回答关于原始文本的问题进行测试。摘要能够捕获评价人员需要的重要信息吗？如果原始文本是教科书一整章，用户是否可以通过阅读三段式的摘要并获得足够的信息来回答提供的练习？这是唯一可同时用于抽取和抽象方法的指标。使用语言模型对抽取式摘要和抽象式摘要进行打分，结果可能会倾向于抽取式摘要，因为该方法直接包含原始文本中的短语，使得它具有更高的可能性。

16.5　文本摘要的应用

本章开头已经介绍了几个摘要应用；我们提到了新闻文章、检索结果和观点总结。摘

要可以节省用户手动阅读整个语料库的时间，同时通过突出“标注”增加预先存在的
325 数据。

前面提到的方面意见分析将用户评论的各部分划分为描述为一个特定主题。我们可以使用这种主题分析来将段落文本归类到关于某个方面的一大堆评论中。我们可以在每个主题上运行一个摘要器，生成可读的文本作为输出，而不是用排序的一元单词描述这个方面。这两种方法相互补充，因为第一步找到用户感兴趣的方面，而第二步则传达信息。

本书中的一个主题是结构化和非结构化数据的联合，将在第 19 章中更详细地提及。摘要是这个应用的一个很好的例子。例如，考虑一个财务摘要器，其中包括证券交易委员会(SEC)的文本报告以及原始股票市场数据。将这两个数据源汇总于同一位置对于(例如)共同基金经理或其他金融工作者来说将是非常有价值的。能够(以文本的方式)总结大量的结构化交易数据可以揭示人类难以意识到的模式——这是知识发现(knowledge discovery)的一个例子。

电子发现(electronic discovery)是在诉讼(法律诉讼和法庭案件)中查找相关信息的过程。律师依靠电子发现筛选大量的文字信息来构建案例。安然电子邮件数据集[⊖]是这个领域中众所周知的语料库。总结两个人或一个部门之间的电子邮件通信，调查人员可以快速决定是否需要在特定区域深入挖掘或尝试其他方法。以这种方式，摘要和搜索是耦合的；搜索允许选择与查询相关的数据的子集，而摘要可以采用搜索结果并快速向用户解释它们。最后，将电子邮件通信联系在一起(从发件人到接收者)是电子邮件本身的非结构化文本内容的结构化补充。

对于阅读本书的人来说，更感兴趣的是能够对一个给定领域的研究进行摘要。给定会议的论文集，是否能用摘要器解释一下当前的主要趋势和常见方法？与以前的会议相比，最新颖的是什么？撰写自己的论文时，你能只完成除了引言和相关的工作外的内容吗？引
326 言是你的论文的概述。相关工作大部分是与你的论文相似的论文的总结。

书目说明和延伸阅读

如本章所述，Das 和 Martins[2007]是关于摘要技术的综述。此外，Nenkova 和 McKeown[2012]也非常值得阅读。对于应用，潜在方面评分分析[Wang 等 2010]，[Wang 等 2011]是一种应用于产品评论的摘要形式。我们在第 18 章中更详细地提到这个具体的应用。Radev 等[2004]提出了一个典型的抽取式摘要器 Ganesan 等[2010]提出了一个典型的抽象式摘要器，Steinberger 和 Jezek[2009]提出关于评价的建议。MMR 算法最初在 Carbonell 和 Goldstein[1998]中描述。对于高级 NLG(自然语言生成)技术，Reiter 和 Dale[2000]是一个好的起点。

练习

16.1　你认为一个摘要方法(抽取或抽象)在小数据集上表现会更好吗？大数据集怎么样？证明你的推理。

16.2　说明如何通过仅查看相邻句子来改善段落检测。你如何实现它？

16.3　在 MeTA 中编写一个基本的段落分割器。作为输入，取一个文档，并用一个内置的切分器将这些句子提取到一个向量中。使用相似度算法将向量分割成段。

⊖ https://www.cs.cmu.edu/~./enron/

16.4　现在你已将文档分割为段落，请使用 MeTA 构建一个搜索引擎，将每个段落视为单个文档。确保每个段落有足够的句子。你可能需要调整你以前的答案来实现这一点。

16.5　使用你的段落搜索引擎，从每个段落中找到有代表性的句子，以创建原始文档的摘要。

16.6　使用 MeTA 的语言模型来获得你想要总结的文档中的单词分布。

16.7　向语言模型添加 **generate** 函数。它应该读取一个上下文($n-1$ 个词项)并产生第 n 个词项。使用本章描述的计算方法生成下一个单词。

16.8　使用生成器对输入文档进行摘要。试验不同的停止条件。哪个工作得最好？

16.9　为自然语言生成实现创建一些简单的后处理规则。我们在文本中给出的例子是句子连接和代词插入。你还能想到别的吗？

16.10　解释我们如何结合文本摘要和主题建模来创建一个强大的探索性文本挖掘应用。

16.11　如果将一个抽象式摘要器的语言模型分布与另外一个概率分布(可能是从已有摘要中估计得到)进行插值，会得到怎样的效果？

327
⌇
328

第 17 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

主题分析

本章内容涉及主题挖掘和分析，涵盖了一种无监督的文本挖掘技术，即概率主题模型。概率主题模型主要用于发掘文本数据中的隐含主题。

主题很容易直观理解，但却不容易被形式化定义。粗略地讲，主题可以是文本数据中叙述的主要内容，也可以是讨论或交流的话题。主题有不同的粒度，例如，可以谈论一个句子的主题、一篇文章的主题、一个段落的主题，或者是图书馆中所有学术论文的主题。不同粒度的主题有不同的应用。

许多应用都需要发掘和分析文本中的主题。例如，可能想要了解推特(Twitter)用户今天在谈论什么：是 NBA 运动、国际事件还是其他话题？也可能想要了解研究课题：人们可能对当前数据挖掘中的研究课题很感兴趣，并且想要了解它们与 5 年前的研究课题之间的区别。为了回答这些问题，我们需要从过往以及现在的数据挖掘文献中提取主题，以便能够进行比较。

我们还可能想要了解人们喜欢什么样的产品，比如智能手机，这需要从正面评论和负面评论中发现主题。亦或想要了解总统大选辩论的主要议题是什么。所有这些都与发现和分析文本中的主题有关。如何挖掘文本中的主题并且进行相应的分析是本章的主要内容。

如图 17-1 所示，主题可以被视为描述关于世界的知识。我们想要从文本数据中发现一些可以用来描述世界的主题，也就是说，主题告诉我们关于世界的一些内容(例如，关于一个产品或者一个人)。

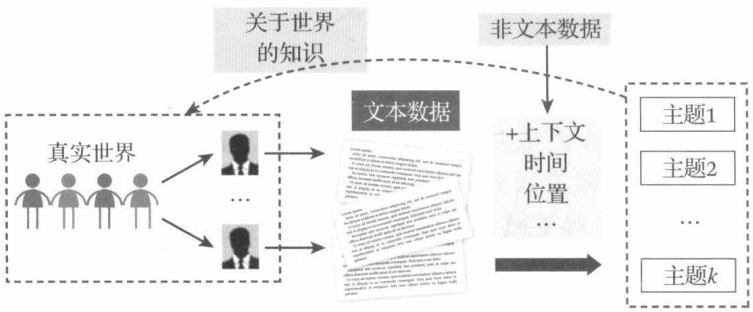


图 17-1 挖掘主题来反映世界知识

除了文本数据，通常可以获取一些非文本数据作为辅助内容来进行主题分析。例如，可以获取与文本数据相关联的时间信息、文本数据生成时的位置信息、文本作者的信息或文本来源的信息。所有这些元数据(或上下文变量)都可能与我们发现的主题相关联，因此可以利用这些上下文变量来分析主题模式。

例如，随着时间的变化观察话题，能够发现是否存在处于上升趋势的话题或者处于消逝趋势的话题。同样，观察不同地点相关的话题，可能有助于了解不同地点人们的观点。

首先来看一个主题挖掘和分析的例子。如图 17-2 所示，主题分析首先要发现一些主题。在这个例子中，共有 k 个主题。还希望知道哪些文档覆盖了哪些主题，以及其覆盖程度。例如，在 Doc 1 中，可视化结果显示主题 1 被很好地覆盖，而主题 2 和主题 k 只被小

部分覆盖。另一方面，Doc 2 很好地覆盖了主题 2，在一定程度上覆盖了主题 k，但完全没有覆盖主题 1。总体来说，通常有两个不同的任务：第一个是从文本集合中发现 k 个主题；第二个是找出哪些文档在多大程度上涵盖了哪些主题。

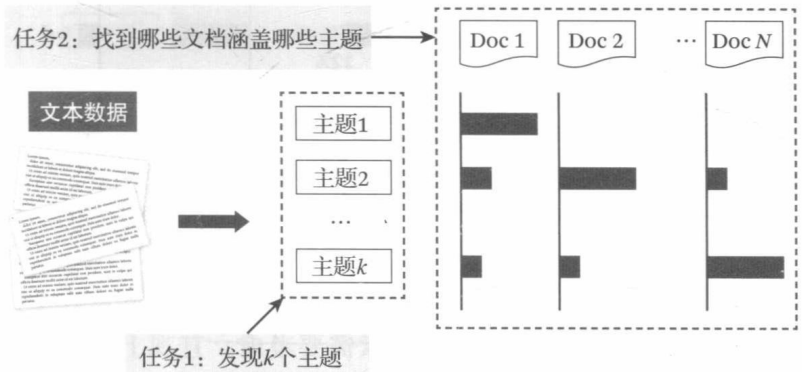


图 17-2　主题挖掘的任务

更形式化地，可以定义一个如图 17-3 所示的问题。首先，输入一个包含 N 个文本文档的集合，这里可以将文档集合表示为 C，并使用 d_i 来表示一个文档。此外，还需要输入主题数量 k，尽管这个数字可以根据数据特征自动设置，但是在本章讨论的技术中，需要指定主题的数目。

- Input
  - A collection of N text documents $C = \{d_1, \ldots, d_N\}$
  - Number of topics: k
- Output
  - k topics: $\{\theta_1, \ldots, \theta_k\}$
  - Coverage of topics in each d_i: $\{\pi_{i1}, \ldots, \pi_{ik}\}$ $\sum_{i=1}^{M} \pi_{ij} = 1$
  - π_{ij} = prob of d_i covering topic θ_j

How to define θ_j?

图 17-3　主题挖掘任务的形式化定义

输出包括被发现的 k 个主题和每个文档 d_i 中主题的覆盖率，其中主题由 θ_1，…，θ_k 表示，覆盖率由 π_{ij} 表示。π_{ij} 是文档 d_i 涵盖主题 θ_j 的概率。对于每个文档，都有一组这样的 π 值来表明文档在多大程度上覆盖了每个主题。进一步，假设这些概率和为 1，这也意味着一个文档不会涵盖已经发现主题以外的其他主题。

接下来的问题是，如何定义一个主题 θ_i？定义 θ 是定义我们任务的关键。下一节将首先讨论定义一个主题(词项作为主题)最简单的方法。

330
~
331

17.1　用词项表示的主题

一种简单自然的方法就是将一个主题表示为一个词项。词项可以是单词或短语。例如，我们可以使用 sports、travel 和 science 这样的词项来表示文本数据中蕴含的三个独立的主题，如图 17-4 所示。

如果以这种方式来定义主题，就可以通过文档中主题词项的出现情况来分析文档对于主题的覆盖程度。图 17-4 展示了一种可能的情况：Doc 1 中 30%的内容关于 sports，12%关于 travel，等等。也可以发现 Doc 2 完全不涵盖 sports，所以覆盖率 π_{21} 为零。

本章关心的两个任务，一个是发现主题，另一个是分析主题覆盖率。为了解决第一个问题，需要从一个集合中挖掘出 k 个主题词项。可以有很多不同的方法来进行主题词项的抽取。一个自然的方法是首先解析集合中的文本数据获得候选词项。这里候选词项可以是单词或短语。最简单的处理方法就是把每一个单词作为一个词项，这些单词就成为了候选主题。

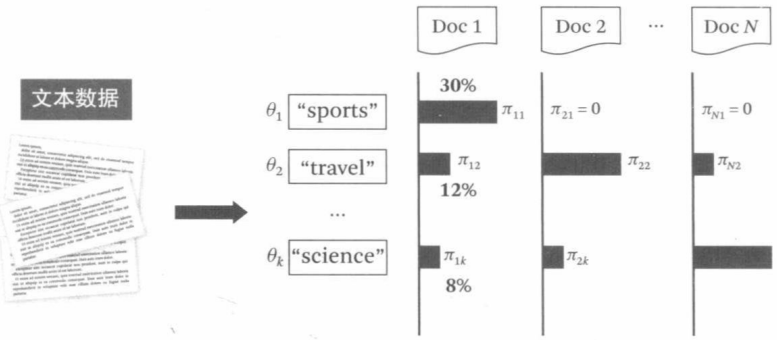


图 17-4　用词项表示的主题

接下来，需要设计一个评分函数来量化每个词项作为主题的好坏程度。在设计这样一个以词项统计为主要依据的函数时，有很多因素需要考虑。直观上这个函数应该赋予有代表性的词项更高的分数。所谓有代表性的词项，就是那些可以代表集合中很多内容的词项。然而，这样的选择将会倾向选择高频出现的词项。如果直接使用频率来设计评分函
332 数，那么得分最高的词项将会是通用词或者像 the 和 a 这样的功能词。这些词项在英语中频繁出现，所以需要避免将它们排在前面。也就是说，得分高的应该是那些使用相对频繁但又不太频繁的词项。

实现这个目标的一种方法是使用 TF-IDF 加权，这在本书前几章关于检索模型和词项关联发现的内容中讨论过。使用这种统计方法来定义评分函数的优点在于，评分函数将是非常普适的，可以应用于任何自然语言和任何文本。当然，将这种方法应用于一个特定问题时，应该尝试使用一些特定领域的启发式方法。例如，在新闻中，由于作者倾向使用标题来描述一篇文章的主题，因此用标题词可能会更好。如果处理推文，使用专门表示主题的标签可能会更好。

在设计了评分函数之后，可以通过简单选取分数最高的 k 项作为 k 个主题词。可能存在这样一种情况，得分最高的词项彼此之间非常相似，换句话说，它们在语义上是相似或者相关的，甚至就是同义词。这不是想要达到的结果，因为我们希望很好地覆盖集合中的所有内容，这意味着要删除冗余。一种途径是使用被称作最大边缘相关性(Maximal Marginal Relevance，MMR)重排序的贪心算法。

这种方法是根据评分函数从上到下选择 k 个主题词。第一个词项显然会被挑选出来。当选择下一个词项时，查看已经被选择的词项并尽量避免选择与它们语义过于相似的词项。所以在考虑列表中一个词项的排名的同时，也会考虑候选词项与已选择词项之间的冗余度。通过适当的阈值，可以在去除冗余与选择高分词项间找到一个平衡。第 16 章详细介绍了 MMR 技术。

在获得表示主题的 k 个主题词项之后，下一个问题是如何计算每个文档中每个主题的覆盖率 π_{ij}。如图 17-5 所示，一个解决方法是直接计算每个主题词出现的次数。例如，sports 在文档 d_i 中可能出现了 4 次，travel 出现了 2 次。接着我们可以将计数归一化，作为对每个主题覆盖概率的估计。归一化是为了保证文档中所有主题覆盖率之和为 1，从而形成每个文档的主题分布，进而来刻画覆盖率。

一般来说，在考虑一个问题的解决方法时，需要考虑以下几个问题：这个方法效果如
333 何？它是解决这个问题最好的办法吗？通常情况下，需要使用实际数据集对它进行评价。然而，分析一些具体例子往往也是具有指导意义的，所以现在来通过图 17-6 所示的简单示例来检验一下刚刚讨论的方法。

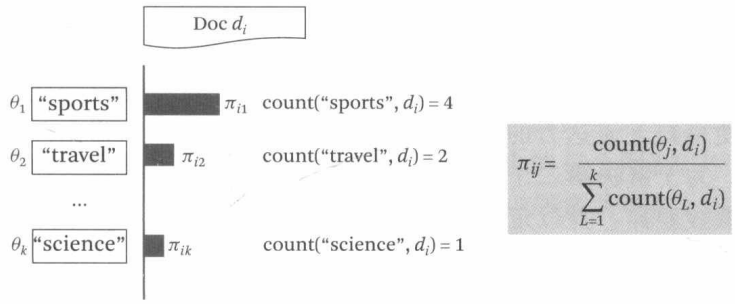


图 17-5　使用词项作为主题来计算主题覆盖率

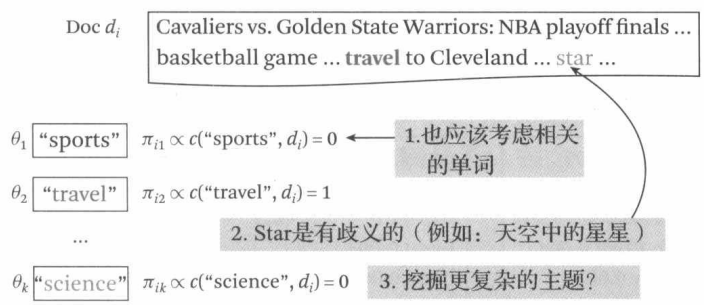


图 17-6　用词项代表主题的问题

这里有一篇关于 NBA 篮球比赛的文章。在内容方面，它是关于运动的。但是如果使用直接计数的方法来寻找代表主题的单词，我们会发现文章中并没有出现 sports 这个词。由于 sports 计数为零，所以对“运动”覆盖率的估计也会为零。可以注意到，science 这个词也没有在文档中出现，因此它的覆盖率估计也为零。显而易见，这是我们想要的，因为该文档不是关于科学的。但是 sports 覆盖率为零显然不合理，因为我们知道文档内容是关于运动的。更糟糕的是，文档中出现了 travel 一词，所以在估计主题 travel 的覆盖率 334
时，将会得到一个非零的数值，它高于 sports 的覆盖率估计，这显然也不是理想的结果。

通过对于这个简单例子的分析，不难看到这种方法的一些问题。首先，当考虑什么单词属于一个主题时，还需要考虑相关词。不能简单地计数被抽取代表主题的词项(例如，sports)，因为它可能在相关主题的文档中完全不出现。另一方面，有很多与主题相关的单词，比如 baskball 和 game，这些在估计主题的覆盖率时也应该被计算进去。

第二个问题是单词的歧义性，例如 star。在这篇文档中它表示篮球明星，而在其他文档中它可能表示天上的星星。所以需要考虑多义单词的语义不确定性。

最后，这种方法的主要限制是只使用一个词项来描述主题，因此不能描述复杂的主题。例如，一个专业的运动主题很难仅仅通过一个单词或短语来描述，而是需要使用更多的单词。

上述分析的核心在于说明，简单地将主题刻画为单一词项存在三个基本问题：第一，缺乏表达力。它只能表示简单普遍的主题，而不能表示需要更多单词描述的复杂主题。第二，词汇覆盖不全面。这意味着主题本身只能由一个词项来表示，它不能揭示与该主题相关的其他词项，使得我们无法估计相关单词对主题覆盖率的贡献。第三，单词的歧义性。主题词项以及相关的单词可能存在多种词义。下一节将会讨论一种改进的表示主题的方法(用单词分布表示主题)，它可以解决上述问题。

17.2　用单词分布表示的主题

为了解决使用单一词项表示主题所带来的问题，一个自然的想法就是使用更多的单词来表示主题，这使得缺乏表达力的问题迎刃而解。当使用更多单词来描述主题时，就可以表示复杂的主题。为了解决第二个问题(如何引入相关单词)，需要引入单词权重。这样能够区分主题间的微妙差异，并以定量的方式引入语义相关的单词。最后，为了解决单词歧义问题，需要“分解”有歧义的单词，使它们可以被用来(潜在地)描述多个主题。

335

事实证明，使用单词概率分布(即一元语言模型)表示主题可以完美实现上述想法。如图 17-7 所示，可以看到，每一个主题都对应着一个单词分布，这个分布包含词汇表中的所有单词。

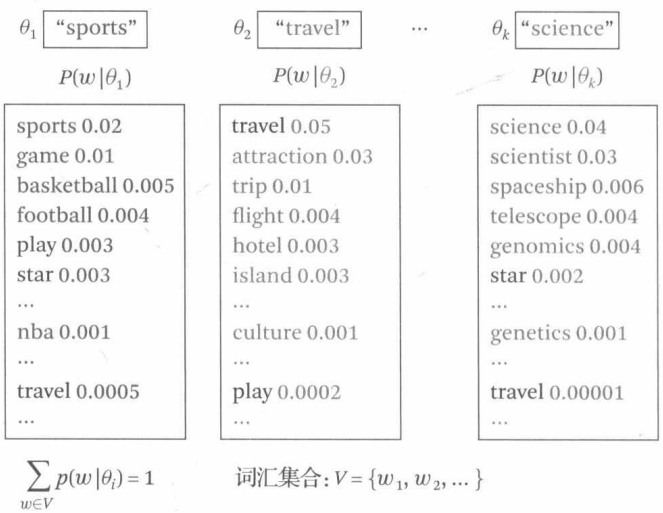


图 17-7　使用单词分布来表示主题

例如，“运动”主题的高概率单词是 sports、game、basketball、footballplay 和 star。这些都是直观上与运动相关的词项，它们的出现有助于判定一篇文章涵盖“运动”主题的可能性。值得注意的是，一般来说，分布可能会赋予所有单词一个非零概率，因为即使一个单词与一个主题不太相关，也总是会有很小的机会出现在这个主题的文章中。还要注意，对于每个主题，所有单词概率和为 1，从而形成所有单词上的一个概率分布。

使用这样的单词分布能够代表一个主题，是因为当从这个分布中采样单词时，可能看到与该主题相关的单词。有趣的是，存在这样一种特殊情况，如果概率完全集中在一个单词上，例如 sports，那么这个主题的单词分布将会退化成为之前讨论过的情况，也就是使用单个词项来代表主题。在这种意义上，单词分布表示是对单一词项表示的自然推广与扩展。

336

然而，通过单词分布的方法可以使用很多单词来描述一个主题，并且捕捉到主题之间的细微差异。通过调整不同单词的概率，可以构建出广义“运动”主题的变种，具体到一个特殊的运动种类，如篮球(basketball 有很高的概率)或足球(football 的概率比 basketball 高很多)。

同样，在“旅行”这个主题的单词分布中，会看到 attvaction、trip、flight 等词汇。在“科学”这个主题的单词分布中，会看到 scientist、spaceship 和 genomics，它们都与“科学”这个主题直接相关。需要注意的是，这并不意味着运动相关的词项在代表“科学”主题的分布中一定具有零概率，只是通常它们的概率通常会很小。

注意到有一些单词被这些主题所共享，这就意味着它们在这些主题中都有很高的概

率。例如，travel 一词同时出现在这三个主题的高频单词列表中，但具有不同的概率。它在“旅行”主题中具有最高的概率 0.05，但在“运动”和“科学”中的概率要小得多，这是合理的。同样，可以看到 star 这个词以相当高的概率出现在“运动”和“科学”中，这是由于它的语义具有歧义性，所以与这两个主题都相关。因此可以看到，通过单词分布来表示主题可以有效地解决之前用单个词项表示主题所产生的三个问题。

- 使用多个单词来描述主题，使得描述比较复杂的主题成为可能。
- 赋予词项权重，既可以捕捉到相关主题语义的微妙差异，也可以汇集相关单词来构建主题和估计主题覆盖率。
- 因为相同的单词在不同主题中具有不同的概率，所以可以表示一个单词的多种含义，也就解决了单词歧义的问题。

接下来，我们对这种表示主题的方法进行检验。由于表示方法是一个概率分布，因此很自然地使用概率模型来刻画这种单词分布，这种方法被称为概率主题模型。

当使用单词分布来表示主题时，基于图 17-3 中的正式定义，我们可以通过将每个主题视为一个单词分布来进一步完善主题分析。也就是说，每个 θ_i 现在是一个单词分布，有 337

$$\sum_{w \in V} p(w|\theta_i) = 1 \tag{17-1}$$

当然，对主题覆盖率仍然有相同的限制，即

$$\sum_{j=1}^{k} \pi_{ij} = 1, \forall i \tag{17-2}$$

作为一个计算问题，主题模型的输入是文本数据，即文档集合 C。假设主题的数量为 k，或者说假设文本数据包含 k 个主题。作为输入的一部分，还需要给定词汇表 V，它决定了哪些单元被视为分析的基本单元(即单词)。在大多数情况下，单词被直接看作分析的基本单元，因为它们本身就是最自然的单元。但是这种方法也很容易进行推广，就像单词一样，可以使用短语或者文本中其他任何能被识别的单元作为基本单元。主题模型的输出由两个概率分布族组成。第一个是由一组 θ_i 表示的主题集合，每一个主题都是一个单词分布。第二个是每个文档 d_i 的主题覆盖分布$\{\pi_{i1}, \cdots, \pi_{ik}\}$。

现在的问题是如何从输入中产生这样的输出。可能有许多种不同的方法实现这一点，但是这里介绍一种解决这个问题的一般方法，也就是生成模型。事实上，这是使用统计模型解决文本挖掘问题的一个非常通用且重要的方法。

这种方法的基本思想是首先为数据设计一个生成模型，即一个刻画数据如何生成的概率模型，也是一个可以计算观察数据可能性的概率模型。实际数据未必(通常不太可能)按照这种方式生成，但是假设数据根据某一模型以特定方式生成，就能够使用形式化的方式来刻画数据，这有助于主题发现。

一般来说，模型会有一些参数(可以用 Λ 表示)。这些参数通过控制数据出现概率的高低来控制模型的行为。如果将这些参数设置为不同的值，模型的行为也会变得不同，也就是说，模型会倾向于赋予不同的数据点高(或低)概率。

我们通过这样的方式设计模型，使其参数能够反应想要发现的信息。进一步，尝试根据数据来估计这些参数，以参数值的形式生成想要的输出。模型对数据的拟合程度或者参 338
数对数据的契合程度往往是统计学中要考虑的一个标准问题，有许多不同的方法可以达到这一目的，正如第 2 章中所讨论的。

使用生成模型从文本数据中发现主题和主题覆盖率，可以发现生成模型需要包含所有代表主题的 k 个单词分布和所有文档的主题覆盖率分布，这是问题需要计算的所有输出。因

此，模型中将包含很多参数。首先，每一个单词分布中都有$|V|$个参数，所以一共有$|V|k$个单词概率参数。其次，对于每个文档，π都有k个值，所以一共有Nk个主题覆盖率参数。因此，总共有$|V|k+Nk$个参数。进一步，由于对θ和π都有约束，自由参数的数量比$(|V|-1)k+N(k-1)$要小。对于每个单词分布，只需要指定$|V|-1$个概率；对于每个文档，只需要指定$k-1$个概率。

一旦建立起模型，就可以将参数用来拟合数据。这意味着可以根据数据来估计或推断参数。换句话说，需要调整这些参数值直到数据集获得最大概率。就像刚刚提到的那样，根据参数值，有些数据点的概率会比其他数据点高。这里主要关心是什么参数值会使数据达到最高概率。

在图 17-8 中，我们把参数 Λ 作为一元变量。虽然看起来过于简化，但是足够用来说明这个想法。y 轴表示数据的概率，这个概率显然取决于 Λ 的设置，这就是为什么在寻找 Λ^* 的过程中，它随着 Λ 值的变化而变化，这里的 Λ^* 是使观察到的数据概率达到最大的参数值。这样的搜索产生了对模型参数的估计。这些参数正是我们希望从文本数据中发现的，所以将它们视为数据挖掘或主题分析算法的输出。

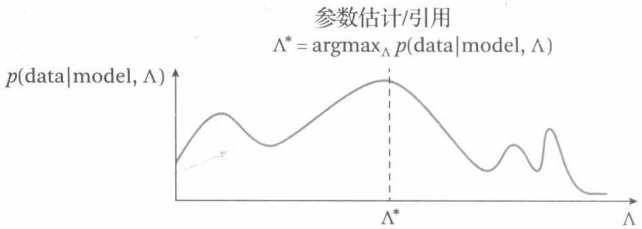


图 17-8　生成模型的最大似然估计

这是使用生成模型进行文本挖掘的一般思路。首先设计一个含有参数的模型来尽可能地描述数据。拟合数据之后，通过学习得到参数值。学习到的参数就是从文本数据中发现的知识。

17.3　挖掘文本中的一个主题

本节将介绍文本数据建模中一个最简单的生成模型示例。在这个示例中，假设文本中仅包含一个主题并且目标是发现这个主题。

如图 17-9 所示，更确切地说，我们将关注如何分析每一篇文档，并发现文档中包含的单一主题。这是主题模型中的最简单的示例。模型的输入不再是 k 个主题，因为已经指定有且仅有一个主题。又因为每篇文档的挖掘过程都是相互独立的，不失一般性，可以进一步假设集合中有且仅有一篇文档。在输出中，假设文档 100%覆盖了想要发现的主题。因此，输出就是表示这个主题的词分布，或者可以说成是词汇表中所有单词的概率分布，如图 17-9 所示。

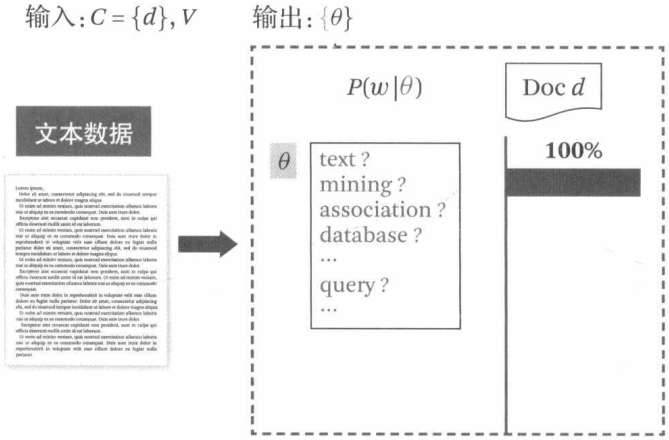


图 17-9　仅有一个主题的简单主题模型

17.3.1　最简单的主题模型：一元语言模型

当使用生成模型解决问题时，首先需要考虑的是需要刻画什么类型的数据，以及从什么方面刻画。如果从不同的方面考虑问题，数据就会“看起来”不同。例如，当把一篇文档仅仅看作是单词的集合，而不考虑词频信息时，我们将得到一个比特向量表示，正如在讨论向量空间检索模型时那样。当把文档看作一个词袋，并考虑单词频率时，这样的表示就需要一个不同的生成模型。在主题分析中，词频能够帮助区分细微的语义变化，所以通常需要保留词频信息。

一旦确定从哪一方面看待数据，就需要设计一个具体的模型按照预期来生成数据，即基于预期的数据表示来对数据进行建模。具体模型的选择一方面取决于对数据的了解，另一方面取决于想要发现的知识。这个目标将决定模型中涉及哪些参数，因为参数刻画和表示了想要挖掘的知识(得到参数值之后)。

这里我们希望通过单词分布表示主题，因此很自然地可以选择一元语言模型，这与 3.4 节的内容相似。在确定模型之后，形式化地写出似然函数，即在给定假设模型后数据的概率。似然函数通常是一个包含参数的函数，它的值会随着参数值的变化而改变。因此，我们尝试找到一个参数值，使函数值达到最大。这种估计参数值的方法称为最大似然估计(Maximum Likelihood Estimate，MLE)。有时候，还可以通过贝叶斯估计的方法引入一些参数的额外知识(先验)，该方法对满足最大似然和保持与先验一致的需求进行折中。

在一般情况下，一旦有了生成模型，就能够使用该模型拟合已有的数据，得到能最好解释数据的参数值。这些参数值就是数据挖掘的输出。

我们将根据这些步骤设计一个简单的主题模型，并用它从文档中发现单一主题。在之后的内容中将会讨论更多复杂的情况。图 17-10 展示了这个模型，其中文档被看作词序列，每个单词用 x_i 表示。这个模型是一元语言模型，也就是使用单词分布表示希望发现的隐含主题。模型参数的个数是词汇表中单词的个数，即为 M。为了方便，使用 θ_i 表示单词 w_i 的概率。根据模型要求，所有单词的概率和应为 1，即 $\sum_{i=1}^{M}\theta_i=1$。

341

- **数据**：文档 $d=x_1x_2\ldots x_{|d|}, x_i\in V=\{w_1,\ldots,w_M\}$ 是一个单词
- **模型**：一元语言模型 $LM\theta(=\text{topic}):\{\theta_i=p(w_i\mid\theta)\}, i=1,\ldots,M; \theta_1+\cdots+\theta_M=1$
- **似然函数**：

$$\begin{aligned}p(d\mid\theta)&=p(x_1\mid\theta)\times\cdots\times p(x_{|d|}\mid\theta)\\&=p(w_1\mid\theta)^{c(w_1,d)}\times\cdots\times p(w_M\mid\theta)^{c(w_M,d)}\\&=\prod_{i=1}^{M}p(w_i\mid\theta)^{c(w_i,d)}=\prod_{i=1}^{M}\theta_i^{c(w_i,d)}\end{aligned}$$

- **最大似然估计**：$(\hat{\theta}_1,\ldots,\hat{\theta}_M)\ \arg\max_{\theta_1,\ldots,\theta_M}p(d\mid\theta)=\arg\max_{\theta_1,\ldots,\theta_M}\prod_{i=1}^{M}\theta_i^{c(w_i,d)}$

图 17-10　应用一元语言模型挖掘单一主题

接下来，我们推导似然函数，即生成整个文档的概率。在一元语言模型中，假设生成每个单词的过程是相互独立的，那么文档的概率等于文档中每个单词概率的乘积(似然函数公式的第一行)。接下来对公式进行变形，把相同的单词划为一组。因此，公式变成了词汇表

中不同单词的乘积(而不是文档中每个位置上单词的乘积)，如似然函数公式的第二行所示。

有些单词可能会重复出现，所以对不同单词进行求积时，需要统计文档 d 中单词 w_i 的个数，用 $c(w_i,d)$ 表示。尽管这个乘积遍历了词汇表中的所有单词，但是如果一个单词没有出现在文档中，那么很显然 $c(w_i,d)$ 值为 0，并且它在公式中的对应项也会消失，因此这个乘积本质上还是文档中实际出现的单词的乘积。有时我们更倾向于使用乘积遍历整
342 个词汇表的似然函数，因为它能够方便地推导出公式中的参数。

现在，我们已经确立了形式化的似然函数，接下来就是求解使得似然函数获得极大值的参数值(即单词概率)。图 17-11 给出了最大似然估计问题较为细致的说明。第一行是最大似然估计的原始优化问题。下一行是使用对数似然对原始函数进行等价变换。因为对数函数是对原始似然函数的单调变换，所以这个等价变换不会影响优化问题的解。这样的变换仅仅是为了计算方便，因为在进行对数变换后，公式中的乘法将变成加法，求和使得求导更容易进行，而导数常常用于寻找函数的优化解。

最大化 $p(d\mid\theta)$：$(\hat{\theta}_1,\ldots,\hat{\theta}_M)=\arg\max_{\theta_1,\ldots,\theta_M} p(d\mid\theta)=\arg\max_{\theta_1,\ldots,\theta_M}\prod_{i=1}^{M}\theta_i^{c(w_i,d)}$

最大化对数似然函数：$(\hat{\theta}_1,\ldots,\hat{\theta}_M)=\arg\max_{\theta_1,\ldots,\theta_M}\log[p(d\mid\theta)]=\arg\max_{\theta_1,\ldots,\theta_M}\sum_{i=1}^{M}c(w_i,d)\log\theta_i$

约束条件：$\sum_{i=1}^{M}\theta_i=1$　　使用拉格朗日乘数法

拉格朗日函数：$f(\theta\mid d)=\sum_{i=1}^{M}c(w_i,d)\log\theta_i+\lambda\left(\sum_{i=1}^{M}\theta_i-1\right)$

$$\frac{\partial f(\theta\mid d)}{\partial\theta_i}=\frac{c(w_i,d)}{\theta_i}+\lambda=0\rightarrow\theta_i=-\frac{c(w_i,d)}{\lambda}$$

$$\sum_{i=1}^{M}-\frac{c(w_i,d)}{\lambda}=1\rightarrow\lambda=-\sum_{i=1}^{M}c(w_i,d)\rightarrow\hat{\theta}_t=p(w_t\mid\hat{\theta})=\frac{x(w_i,d)}{\sum_{i=1}^{M}c(w_i,d)}=\frac{c(w_t,d)}{|d|}$$

图 17-11　计算一元语言模型的最大似然估计

虽然这个对数似然函数很简单，但它反映了一些更复杂生成模型对数似然函数的一般特征。

- 求和遍历了所有的数据点(词汇表中的单词)。
- 在求和中，需要对每个数据点计数，即对观察到的数据中的每个单词计数，然后再
343 与对应的数据点概率的对数相乘。

至此，问题已经转换为一个数学优化问题，目标是找到约束最大化问题的最优解。目标函数是对数似然函数，约束条件是所有单词的概率和为 1。如何求解这样的优化问题已经超出了本书的范围。然而，这种情况可以利用拉格朗日乘数法得到一个简单的解析解。图 17-11 展示了拉格朗日乘数法这一常用方法的详细细节。

首先建立一个拉格朗日函数，它把目标函数与约束条件通过拉格朗日乘子 λ 相结合，同时引入一个新的参数。可以看出，原始约束优化问题的求解与新的(非约束)拉格朗日函数的求解是等价的。

因为问题中不再有约束条件，所以能够直接对所有参数求偏导并令其为零，得到关于每个参数的方程，进而求解这个优化问题。[⊖]因此，总共有 $M+1$ 个线性方程，对应于 M

⊖ 导数为零是函数达到最优解的必要条件，但不是充分条件。然而，在只有一个局部最优解的情况下，这个条件也是充分的。

个单词的概率参数以及 λ。注意拉格朗日乘子 λ 对应的方程正是原来的约束。我们能够容易地求解这个线性方程组从而获得一元语言模型的最大似然估计

$$p(w_i|\hat{\theta}) = \frac{c(w_i,d)}{\sum_{j=1}^{M} c(w_j,d)} = \frac{c(w_i,d)}{|d|} \tag{17-3}$$

估计结果有一个非常有意义的解释：一个单词的概率估计就是这个单词出现的次数除以文档长度，文档长度也就是文档中所有单词的计数和。为了最大化似然函数，这个估计和我们的直觉相符：经常看到的单词“值得”更高的概率，并且只有看到的单词“允许”有非零概率(未出现的单词的概率应该为零)。一般来说，最大似然估计往往转换为概率估计，概率估计是由相应事件的频数归一化计算而来，因此，经常看到的事件有较高的概率，看不到的事件概率为零。

在这个简单的例子中，虽然得到了最大似然估计的解析解，但是这样的解析解并不总是能够获得的，实际上常常不能够获得它。最大似然估计的优化问题通常非常复杂，所以 344
我们一般使用数值优化算法解决这类问题。

从文档中挖掘的主题是什么样子呢？想象文档是一篇文本挖掘领域的相关论文。在这种情况下，一元语言模型(单词分布)的估计如图 17-12 所示。在顶部，出现的高概率词通常是那些常见词，即英语中的功能词。紧随其后的内容词才是真正描述这个主题的单词，例如 text 和 mining。在最后，会看到一些小概率单词，这些单词并不是真的与主题有关，而是可能碰巧在文档中被提及。

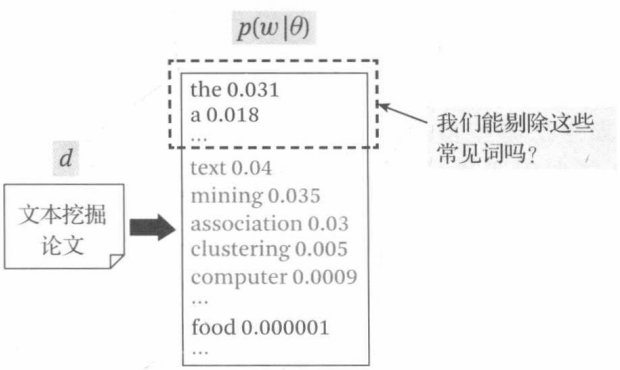


图 17-12　常见词在估计的一元语言模型中占主导地位

作为一种主题表示，这样的分布是不理想的，因为高概率单词是功能词，不能描述这个主题的特征。生成模型使用一个概率分布生成文本中的所有单词，赋予常见词高概率是生成模型的一个直接结果。怎样才能改进生成模型，以降低主题中常见词在词分布中的概率呢？答案是，可以把一个背景词分布引入生成模型中，这样常见词就可以由背景模型产生，因此主题词分布只需要产生表达实际内容的主题词。因为多个组件模型“混合”在一起生成数据，所以这样的模型称为混合模型，我们将在下一节中详细讨论。

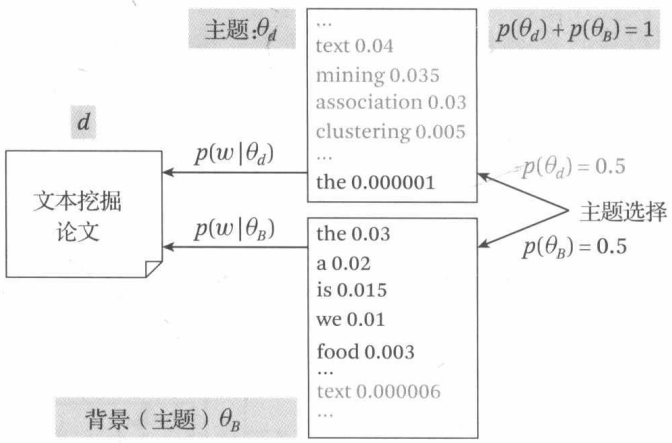
17.3.2　添加背景语言模型

为了解决在一元语言模型中文档里常见词概率较高的问题，思考为什么会产生这个问 345
题对于解决问题是有帮助的。不难发现这个问题的出现有两点原因。首先，这些常见词在数据中频繁出现，因此任何最大似然估计都会赋予这些词较高的概率。其次，生成模型假设所有单词都从简单一元语言模型中产生，最大似然估计因此别无选择，只能分配这些常见单词较高的概率，以达到最大化似然的目标。

为了剔除常见词，需要设计一个不同的生成模型。在这个生成模型中，表示主题的一元语言模型不用生成文本数据中的所有单词。具体来说，理想的主题一元语言模型不应该生成常见词。因此，需要引入另一种分布来生成这些常见词，这样就有一个完整的生成模

型来产生文档中所有的单词。因为第二个分布是用来解释常见词，所以这个分布的一个自然的选择就是背景一元语言模型。因此，形成了由两个一元语言模型为组件的混合模型，一个是未知的主题模型，用于发掘隐含的主题信息；另一个是已知的背景语言模型，用于给常见词分配高概率。

在图 17-13 中，可以看到两个分布混合在一起生成文本数据，使用背景模型生成常见词，使用主题语言模型生成文档中具有实际意义的单词。因此，我们希望发现(学习)一个一元主题语言模型，这个模型分配高概率给具有实际意义的单词，而分配低概率给英语中的常见功能词。



346

图 17-13　带有背景模型的两组件混合模型

这样的混合模型生成一个单词的过程与最简单的一元语言模型略有不同。因为包含两个分布，所以需要决定使用哪个分布生成单词，但是每个单词仍然是两个分布其中一个的样本。文本数据还是以逐一生成单词的相同方式被生成。具体来说，当生成一个单词时，首先要确定使用哪一个分布。因此引入一个新的概率分布来选择使用哪个组件(在本例中有两个选择)，具体包括概率 θ_d(使用未知的主题模型)和概率 θ_B(使用已知的背景模型)。因此，$p(\theta_d)+p(\theta_B)=1$。

在图中，$p(\theta_d)$和 $p(\theta_B)$都设为 0.5。可以类比于用抛硬币的方式来决定使用哪个分布，虽然一般来说这些概率不是必须相等的，因为一个主题可能比另一个更有可能产生。

从混合模型中生成单词概率的过程如下：首先，抛一枚不均匀的硬币来决定使用哪个单词分布，其中正面朝上的概率为 $p(\theta_d)$(因此反面的概率 $p(\theta_B)=1-p(\theta_d)$。如果硬币是正面，使用 θ_d，否则使用 θ_B。然后使用选中的单词分布生成单词。如果使用 θ_d，采样单词的概率为 $p(w|\theta_d)$，否则概率为 $p(w|\theta_B)$，具体说明如图 17-13 所示。

在单词生成的过程中，这个生成模型对于使用哪个单词分布有一些不确定性。如果将这个生成模型看作一个黑盒，这个模型的表现和之前所述的简单主题模型非常相似，虽然之前的模型只使用一个单词分布来刻画一个单词序列的概率。因此，能够通过混合模型刻画任何特定单词的概率，并计算观察到词序列的概率。

如图 17-13 所示，假设有一个混合模型，然后考虑两个特定的单词，the 和 text。在混合模型中，观察到单词 the 的概率是多少？既然有两种方式产生 the，那么直观上概率

347 是两种方式的和。观察到单词 the 从背景模型中产生的概率是多少？为了使单词 the 从背景模型中生成，首先需要选择使用背景模型，然后从背景模型 $p(w|\theta_B)$中生成单词 the。因此，单词 the 从背景模型中产生的概率为 $p(\theta_B)p(\text{the}|\theta_B)$。如图 17-14 所示，无论使用

哪个分布，单词 the 从混合模型中产生的概率为 $p(\theta_B)p(\text{the}\,|\,\theta_B)+p(\theta_d)p(\text{the}\,|\,\theta_d)$。图 17-14 还展示了如何计算单词 text 的概率。

$$P(\text{“the”}) = p(\theta_d)p(\text{“the”}\,|\,\theta_d) + p(\theta_B)p(\text{“the”}\,|\,\theta_B)$$
$$= 0.5 \times 0.000001 + 0.5 \times 0.03$$

$$P(\text{“text”}) = p(\theta_d)p(\text{“text”}\,|\,\theta_d) + p(\theta_B)p(\text{“text”}\,|\,\theta_B)$$
$$= 0.5 \times 0.04 + 0.5 \times 0.000006$$

图 17-14　单词 the 和 text 的概率

不难得出结论，从混合模型中产生任一单词 w 的概率公式如下

$$p(w)=p(\theta_B)p(w\,|\,\theta_B)+p(w\,|\,\theta_d)p(\theta_d) \tag{17-4}$$

这是对于两种方式生成这个单词的概率进行加和，分别对应之前所述的两种分布。公式中的每一项分别表示的是单词从其中一个分布中生成的概率。例如，$p(\theta_B)p(w\,|\,\theta_B)$表示的是单词 w 从背景语言模型中生成的概率。为了计算单词 w 的概率，需要 1)已经确定使用背景模型分布(即已知概率 $p(\theta_B)$)，2)已经从分布 θ_B 得到单词 w(即已知概率 $p(w\,|\,\theta_B)$)。为了观察单词 w 从背景分布中生成的概率，这两个事件必须发生，因此将这两个概率相乘来获得从背景分布中观察到 w 的概率。相似地，$p(\theta_d)p(w\,|\,\theta_d)$表示从主题单词分布中观察到单词 w 的概率。将两者相加得到单词 w 从两个分布中生成的概率。

这种形式的似然函数实际上反映了混合模型似然函数的一些共有特征。首先，从混合模型中观察到数据点的概率是不同生成方法的概率和，每个方法对应混合模型中的不同组件。其次，和中的每一项是两个概率的乘积：一个是选择相应组件的概率，另一个是数据点从选定组件模型中生成的概率。乘积表示的是使用相应的组件模型时，观察到数据点的概率，这就是无论选择哪个组件模型来生成数据点，总和给出的是观察到数据点的总概率的原因。后面我们将看到，更复杂的主题模型会使用两个以上的组件，但是这些模型生成单词的概率形式与这里看到的一样，唯一的区别就是和中有两个以上的乘积(更准确地说，乘积数量与组件模型一样多)。

348

一旦写出了一个单词的似然函数，就很容易看出，混合模型整体可以视作一种单一的单词分布，只不过它的形式相对复杂。换句话说，上面定义的形式给出了单词上的概率分布。因此，概念上来讲，混合模型也是一种生成模型，通过独立生成每个单词来生成单词序列。一般情况下，简单一元语言模型通过显式地指定每个词的概率来定义单词的分布。

如图 17-15 所示，混合模型的主要思想是把多个分布合成为一个模型。图中框内“封装”的两个分布形成了一个简单的生成模型。当把整个框当作一个模型时，可以很容易地看出它就像其他生成模型一样，能够给出每个单词的概率。然而，这个概率在混合模型中是如何被决定的与只有一个一元语言模型的情况完全不同。

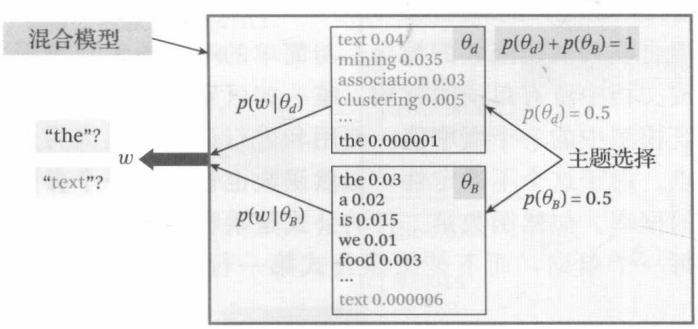


图 17-15　混合语言模型的主要思想

349

研究模型的一些特例可以直观地帮助我们解释模型，并且揭示简单模型和复杂模型之间的关系。在这种情况下，可以检查如果设置选择背景组件的概率为零时将会发生什么。

很容易看到，在这种情况下，对应背景模型的公式项将在和中消失，混合模型将退化为仅用一个分布来描述主题的特殊情况。从这个意义上说，混合模型比之前的模型更一般化，它包含仅有一个分布的特殊情况。自然地，使用混合模型时应该将选择背景语言模型的概率设置为非零常数，这样它才能帮助解释数据中的常用词并让主题词分布更加集中于有意义的词。

一旦写出似然函数，下一个问题就是如何估计参数。如果是简单的一元语言模型，我们可以使用任何一种方法(例如，最大似然估计)来估计参数，然后把参数视作从文本中发现的知识。

在两组件混合模型中需要估计哪些参数呢？图 17-16 总结了两组件混合模型，列出了所有的参数，阐明了参数估计问题。首先，输入数据仅是一篇文档 d，模型是包含两个组件的混合模型。其次，参数包括两个一元语言模型以及两个语言模型的分布(混合权重)。在公式中，θ_d 表示文档 d 的主题分布，θ_B 表示背景词分布。背景词分布通常可以被设置为一个固定单词分布，这个分布赋予常用词高概率。将所有的参数用 Λ 表示。(现在你知道一共有多少参数吗?)

- **数据**：文档 d
- **混合模型**：参数 $\Lambda = (\{p(w \mid \theta_d)\}, \{p(w \mid \theta_B)\}, p(\theta_B), p(\theta_d))$
  - 两个一元语言模型：θ_d（d的主题）；θ_B（背景主题）
  - 混合权重（主题选择）：$p(\theta_d) + p(\theta_B) = 1$
- **似然函数**：

$$p(d \mid \Lambda) = \prod_{i=1}^{|d|} p(x_i \mid \Lambda) = \prod_{i=1}^{|d|} [p(\theta_d) p(x_i \mid \theta_d) + p(\theta_B) p(x_i \mid \theta_B)]$$

$$= \prod_{i=1}^{M} [p(\theta_d) p(w_i \mid \theta_d) + p(\theta_B) p(w_i \mid \theta_B)]^{c(w,d)}$$

- **最大似然估计**：$\Lambda^* = \arg\max_{\Lambda} p(d \mid \Lambda)$

$$\text{满足} \sum_{i=1}^{M} p(w_i \mid \theta_d) = \sum_{i=1}^{M} p(w_i \mid \theta_B) = 1 \quad p(\theta_d) + p(\theta_B) = 1$$

图 17-16　两组件混合模型的总结

图中还展示了混合模型似然函数的推导。与简单的一元语言模型形式上一样，这个似然函数仍然可以看成文档中所有单词的乘积。唯一的区别是内部乘积，现在每一项是概率和，而不是一元语言模型中的一个概率值。使用和进行计算，是因为使用时不确定选择哪个模型来生成数据点。由于这个不确定性，似然函数也包含了一个参数，该参数表示选择每个具体组件分布的概率。似然函数第二行的公式是乘积的另一种书写形式，现在这个乘积遍历了词汇表中每一个单词，而不是按照公式第一行的方式遍历文档中每个位置上的单词。

现在有两种约束：一种是所有单词的分布和需要为 1，另一种是选择每个主题模型的概率和需要为 1。因此，最大似然估计问题可以看作约束优化问题，需要寻找满足所有约束的参数值来最大化似然函数。

17.3.3　混合模型的参数估计

350 ~ 351

本节将讨论如何估计混合模型中的参数。首先从最简单的场景开始，一个组件(背景模型)已经完全已知，主题选择分布选择背景分布和主题单词分布的概率相等。我们的目标是估计未知的主题单词分布，并且避免常用词在其中有较高的概率。如图 17-17 所示，一个主要的假设是，常用词应由背景模型生成，而内容词则由主题单词分布生成。这里面的情景也是提出使用混合模型的主要原因。

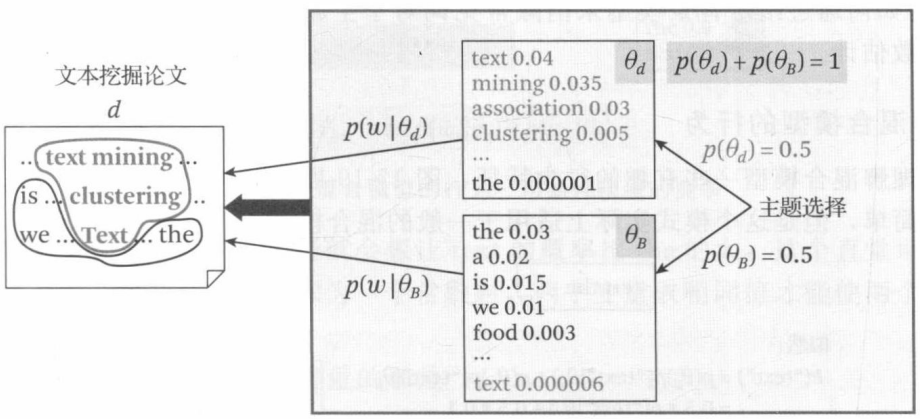


图 17-17　分离背景单词的混合语言模型

图 17-18 阐述了这样一个场景。在这个场景中，只有主题单词分布 $p(w|\theta_d)$这组参数未知。因此，该场景需要估计的参数数量与一元语言模型完全相同。值得注意的是，这是一个为了说明内部原理而定制的概率模型的例子，已经根据合理的假设对于其他部分进行了简化，以便能够嵌入一个未知变量进行计算。对大量英文文本使用一元语言模型进行最大似然估计，然后使用估计后的语言模型来设置背景模型，这种方案既是可行的，也是符合预期的，因为我们的目标是为了减少常用词对于计算主题单词分布的影响。在混合模型中，引入一个估计好的背景单词分布是一项为模型注入先验知识的有效技术，表明了哪些词应该作为噪声(停用词)。同样，参数 $p(\theta_B)$可以根据我们期望的常用词比例进行设置，$p(\theta_B)$设置越大，更多的常用词就会从主题单词分布中移除。很容易看出，如果 $p(\theta_B)$设置为 0，将无法删除任何常用词，模型退化为仅使用一个分布的简单例子(解释所有单词)。

352

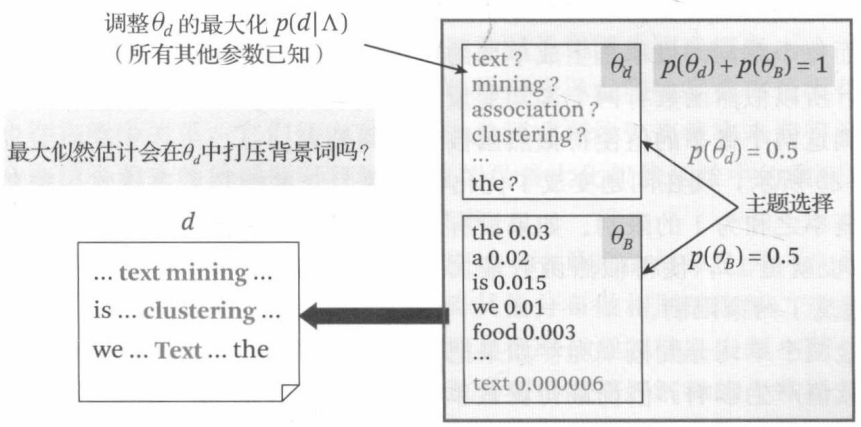


图 17-18　包含一个主题的语言模型的估计

值得注意的是，我们也可以假设 θ_B 和 θ_d 是未知的，然后使用最大似然估计的方法来学习参数。但是在这种情况下，不再能够保证得到的分布 θ_B 分配给常用词高概率。对于当前任务的场景(即分离常用词)，更适合预设背景词分布，使模型偏向于将常用词分配到背景单词分布中，从而使得主题单词分布更关注内容词。下面做进一步解释。

如果将图 17-18 中的混合模型看作一个黑盒，那么模型参数的个数与单一元语言模型完全相同(实际上，相同的参数)。但是，混合模型有一个不同的似然函数，它需要 θ_d 与固定的背景模型 θ_B 一起“合作”来解释观察到的文档。下一节将解释这种带约束的“合作”是如何通过给定背景模型来消除常见词对于主题分布的影响，这需要理解混合模型中的参数估计。

17.3.4　混合模型的行为

为了理解混合模型一些有趣的行为特征，图 17-19 展示了一个简单的例子。虽然这个
353 例子非常简单，但是这个模式实际上适用于一般的混合模型。

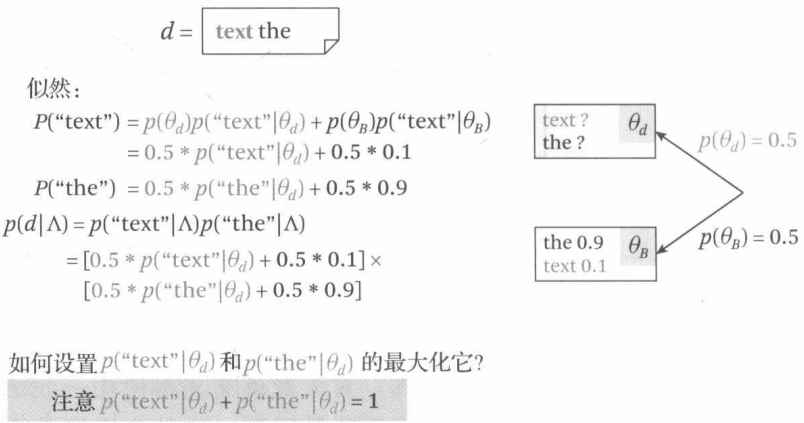


图 17-19　混合模型行为说明

假设选择两个模型的概率是相同的，也就是可以通过投掷一枚无偏的硬币来决定使用哪个模型。进一步，假设词汇表中恰好只有两个单词：the 和 text。显然，这是对实际文本的简化。但是这样一个特例对于研究模型的工作机理是有用的。此外，假设背景模型设置 the 的概率是 0.9，设置 text 的概率是 0.1。

这种情况的似然函数如图 17-19 所示。这个包含两个单词的文档的概率是每个单词概率的乘积，而每个单词的概率是生成单词的两个分布的概率和。因为除了 θ_d，已经知道了所有的参数，所以似然函数有两个未知变量 $p(\text{the}|\theta_d)$ 和 $p(\text{text}|\theta_d)$。计算最大似然估计的目标是找到这两个概率的值使得似然函数值达到最大。

如图 17-20 所示，现在问题变成了优化包含两个变量的简单表达式。需要注意，仍应满足这两个概率之和为 1 的限制。如果没有这个约束条件，就可以将两个概率都设置为自身的最大值(也就是 1)，使得似然函数最大。但是，如果设置两个概率都为 1，那么概率和就为 2，违反了约束限制。

如何给这两个单词分配概率呢？如果把概率值从一个单词转移到另一个单词，很明显会对似然函数值产生影响。假设最初设置单词 the 和 text 的概率都为 0.5，然后把 the 概率值逐渐转移到 text 上，或者把 text 的概率值逐渐转移到 the 上。这种变化会如何影响似
354 然函数值呢？

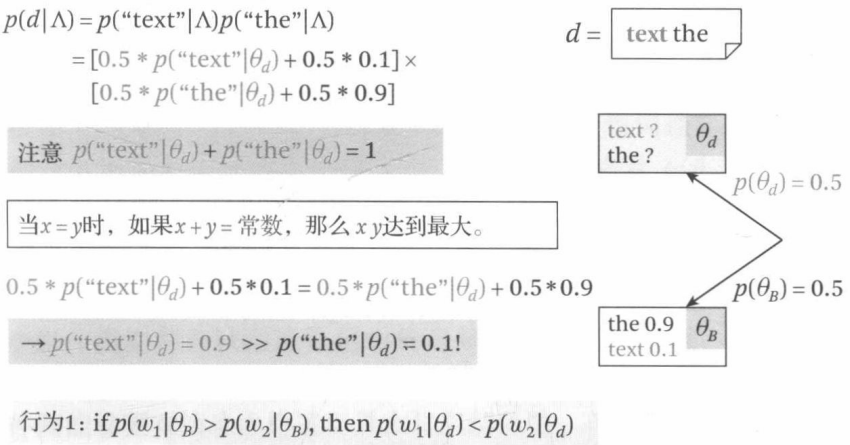


图 17-20　混合模型的行为：两组件模型的竞争

如果仔细检查这个公式，可能就会想让 text 的概率比 the 的大，这个直觉可以用一个数学事实来说明：当两个变量的和为一个常数时，两个变量取相同值才能使两个变量的乘积达到最大。

在这个模型中，乘积中两个变量的和是

$$0.5\cdot p(\text{text}|\theta_d)+0.5\cdot 0.1+0.5\cdot p(\text{the}|\theta_d)+0.5\cdot 0.9=1.0$$

所以在下面这种情况乘积最大：

$$0.5\cdot p(\text{text}|\theta_d)+0.5\cdot 0.1=0.5\cdot p(\text{the}|\theta_d)+0.5\cdot 0.9$$

代入约束条件 $p(\text{text}|\theta_d)+p(\text{the}|\theta_d)=1$，可以得到 $p(\text{text}|\theta_d)=0.9$，$p(\text{the}|\theta_d)=0.1$。

因此，text 的概率的确比 the 大很多，能够有效地分解出常用词。值得注意的是，当只有一个分布时，这个分布中 the 的概率要远远大于 text 的概率。减少 the 概率的效果显然是由背景模型导致的，背景模型赋予了 the 高概率、text 低概率。

仔细研究模型的求解过程，可以看出 text 比 the 概率高的原因是背景模型中 text 的概率 $p(\text{text}|\theta_B)$比 the 小；假如背景模型赋予 the 的概率比 text 小，那么结果将是主题分布中 the 的概率比 text 大，以确保两个模型一起工作得到的总体概率对 the 和 text 是一样的。因此，如果一个单词在背景模型中概率低，那么极大似然估计倾向赋予这个单词高概率。更一般地说，如果在一个分布中 w_1 的概率高于 w_2，那么在另一个分布中 w_2 的概率高于 w_1，这样才能使得 w_1 在两个分布中结合起来的概率与 w_2 相同。换句话说，两个分布倾向赋予不同单词高概率，好像它们在试图避免赋予相同单词高概率。 355

在这样的两组件混合模型中，两个分布互相协作使得观测数据概率最大化，但是两个分布对单词也存在竞争关系，它们倾向对不同的单词“押注”高概率来获得竞争优势。为了使得两个分布结合起来的单词概率和相等(也就是最大化似然函数中的乘积)，θ_d 分配给一个单词的概率必须大于 θ_B 分配给该单词的概率。

混合模型的一般行为是，如果一个分布分配给一个单词的概率比另一个单词更高，其他分布会进行相反的概率分配，也就是它会阻碍其他分布做相同的事情。这也说明，通过使用一个背景模型固定地给常用词(停用词)分配高概率，可以鼓励未知主题分布给这些单词分配较小的概率，而把更多的概率分配到背景模型不能很好解释的内容词上。

接下来通过图 17-21 来研究混合模型的另一个行为，即估计所得的概率对于数据频率的响应。在图 17-21 中，我们为文档加入了更多的单词，具体来说，加入了更多的 the 到

文档中。如果添加越来越多的 the 到文档中，估计参数 $p(w|\theta)$时会发生什么事情呢？

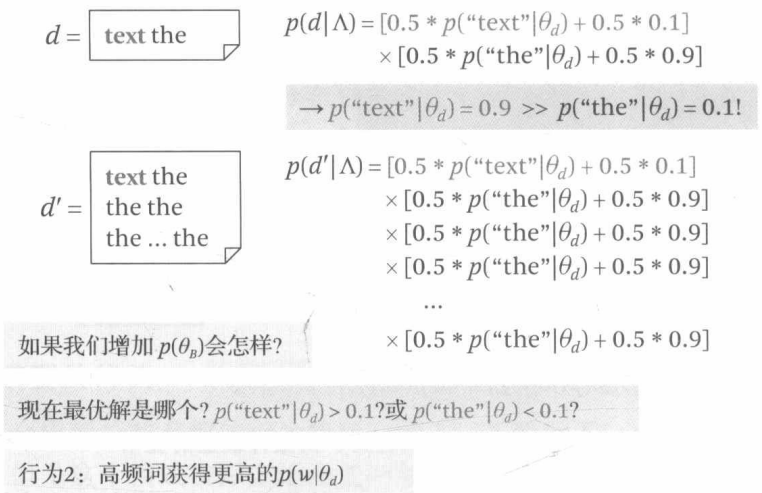


图 17-21　混合模型的行为：最大化数据似然

当加入更多的单词到文档中时，需要在似然函数中添加新增项来刻画新增事件。在这个例子里面，因为新增的单词都是 the，仅需要乘上表示 the 的概率项。这显然改变了似然函数，因此也改变了最大似然估计求解的结果。新增的关于 the 多次出现的概率项是如何改变最大似然估计的呢？之前推导的结果 $p(\text{the}|\theta_d)=0.9$ 已经不是最优解了。如何修正这个答案使得新的似然函数有最优解呢？

解决这个问题的一种方法是减少某个词的概率，从而增加其他词的概率，以确保它们总和为 1。当然，问题是减少哪些词的概率，增加哪些词的概率？是否应该使 the 的概率大于 text 的概率呢？

仔细观察目前的公式，可能发现新的似然函数(也就是用于优化的目标函数)更多地会受到 the 的影响，所以与减小 text 的概率相比，the 的概率的减小会引起似然函数的更大幅度的减小。事实上，减小 text 的概率只影响一项，而增加 the 的概率会给似然函数中很多项增益(因为 the 出现了很多次)，因此产生了增加似然函数值的整体效果。换句话说，因为 the 在似然函数中出现了很多次，如果增加 the 的概率，会对似然函数产生很大的积极影响。而降低 text 的概率只会有相对较小的负面影响，因为它只出现过一次。

356
~
357

以上分析阐述了混合模型最大似然估计的另一种行为：文档中的高频词倾向在所有分布中都具有高概率。这种行为并不难理解，因为要最大化数据的似然函数，一个词如果多次出现时，似然函数的总体概率应该越高。事实上，这是所有极大似然估计的一个非常普遍的现象。在这个特例里面，如果一个单词在文档数据中多次出现，将鼓励未知分布 θ_d 分配给这个单词稍高的概率。

可以使用这个例子来研究选择背景模型的概率 $p(\theta_B)$的影响。在之前的讨论中，都假设每个模型是等概率被选择的，也就是 $p(\theta_B)=0.5$。但是，再一次查看图 17-21 中的这个似然函数公式，试想一下如果增加选择背景模型的概率，似然函数将会发生什么变化。

不难发现，如果 $p(\theta_B)$设置成大于 0.5 的较大数值，那么背景分布被选择的概率较高，公式中包含 the 的所有概率项都会增大，而且 0.9 前面的系数(之前为 0.5)将变得更大。

结果是即使向文档中加入更多的 the，the 的概率增加对 θ_d 的影响也不大。这是因为

the 的概率已经非常大了(原因是 $p(\theta_B)$和 $p(\text{the}|\theta_B)$值很大)，并且调节系数 $p(\text{the}|\theta_d)$带来的影响被系数 $p(\theta_d)$所限制，在 $p(\theta_B)$很大的情况下，$p(\theta_d)$就会很小。因此，确保 $p(\text{text}|\theta_d)$较大对 θ_d 是更有帮助的，因为单词 text 不能从背景模型中获益，必须依赖 θ_d 来分配一个高概率值。在估计概率值 $p(w|\theta_d)$的过程中，高频词趋向于有较高的概率，由于文档中单词的次数增加而增加的概率值大小受到 $p(\theta_d)$(或者 $p(\theta_B)$)限制。$p(\theta_d)$越小，数据中一个单词的增加对 θ_d 的影响越小。一般来说，在混合模型中一个组件被选中的可能性越大，该组件在模型中越重要，也就会为那些高频词分配更高的概率值。

在这一节中，我们讨论了混合模型、混合模型的参数估计问题和最大似然估计的一般行为。首先，每个组件模型都会试图给数据中的高频词分配高概率值，目的是最大化似然函数。其次，为了避免“竞争”或浪费概率值，不同的组件模型倾向赋予不同的单词高概率值，这将使组件间能够更有效地协作进而最大化似然函数。再次，选择组件的概率值能够调节组件模型之间的协作和竞争关系，这将使得一些组件模型对数据中词频的变化具有更好的反映能力。此外，我们还讨论了固定背景模型这一单词分布，可以基于大量英文文档使用最 358
简单的单一一元语言模型来建模数据。这种混合模型的最大似然估计行为确保了使用背景模型能够有效地从其他单词分布中分离出像 the 这样的常用词，使发现的主题更加具有区分性。

我们可以将这种特殊的混合模型视为一个对参数施加了很强先验的模型，从而使用贝叶斯参数估计。在两组件混合模型中，先验施加在其中一个一元语言模型，并要求它与预定义的背景语言模型完全一致。总体来说，贝叶斯估计寻找的是先验和数据似然之间的平衡。但在这种情况下，可以理解为假设先验无限强大，从而保证一个组件模型不变(与提供的背景模型一样)。这种混合模型曾用于本书前面章节关于信息检索的反馈技术中。

17.3.5　期望最大化

对混合模型最大似然估计的讨论说明了为什么可以使用混合模型从文档中挖掘单一主题，并通过使用背景模型将常见词分离出来。在这一节中，我们将进一步讨论如何计算最大似然估计。不像简单一元语言模型的最大似然估计具有解析解，即使固定背景模型和选择组件模型的概率(即 $p(\theta_d)$)，并与单一一元语言模型保持一样多的参数，两组件混合模型的最大似然估计仍然没有解析解。因此我们必须使用数值优化算法来计算最大似然估计。

本节将介绍一个特定的算法来计算两组件混合模型的最大似然估计，叫作期望最大化(EM)算法。EM 算法是解决混合模型最大似然估计常用的算法。

假设 $p(w|\theta_B)$和 $p(\theta_B)$已知，那么模型中唯一的未知参数是 $p(w|\theta_d)$，并且该分布在所有单词上和为 1，如图 17-22 所示。直观上，当使用最大似然估计时，需要探索单词分布 θ_d 所有可能的取值，直到找到一组值，使得观察到的文档概率最大。

在混合模型中，可以想象文档数据中的单词被分为两组。一组是由背景模型产生，另 359
一组是由未知的主题模型(主题词分布)产生。最大似然估计中的挑战是不知道使用混合模型中哪个分布来产生单词。

如果知道了每个词从哪个分布中产生，最大似然估计的计算将如图 17-22 所示的一样简单。其中 d'表示一篇由 θ_d 生成的伪文档，是由文档 d 中所有单词组成，并且 θ_d 的最大似然估计可以看作是在伪文档 d'中词频归一后的概率。也就是可以简单地将所有从 θ_d 中生成的单词聚集到一起，计算每个单词的数量，然后使用伪文档中的全部词数进行归一化。在这种情况下，混合模型由两个独立的一元语言模型构成，因此可以根据从各自分布中生成的单词独立进行参数估计。

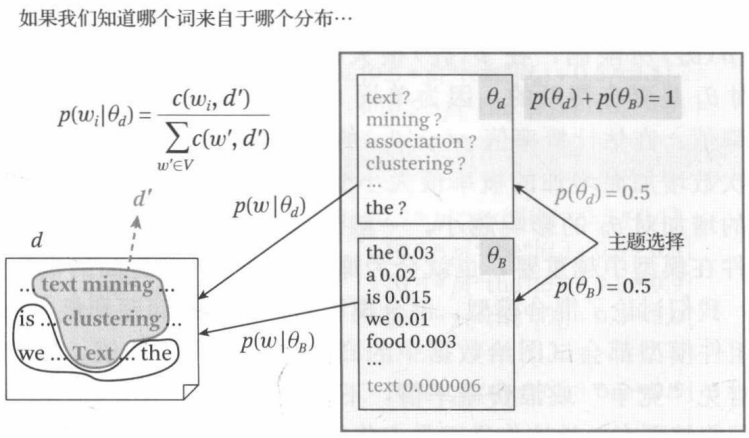


图 17-22　当已知每个单词属于哪个特定分布时主题的估计

但是，在真实情况中，我们并不知道每个单词来源于哪个分布。EM 算法的主要思想就是基于临时估计的参数来猜测(推断)单词的来源，然后使用推断的来源来改进参数的估计。当参数估计有了改进后，又会相应地提高推断来源的准确性。这样就得到了一个迭代的爬山优化算法，不断改进参数的估计，直到似然函数收敛到局部最大值。在每次迭代 360 中，E 步在前，M 步在后，下面将进行详细的解释。

现在，首先假设已经获得了全部参数的一个初步估计。那么，如何推断一个单词是从哪个分布中产生呢？考虑一个具体的单词，如 text，它是由 θ_d 产生，还是由 θ_B 产生呢？为了回答这个问题，计算另一个条件概率 $p(\theta_d \mid \text{text})$。$p(\theta_d \mid \text{text})$的值依赖于两个因素。

- 一般情况下，使用 θ_d(与之相对的是 θ_B)生成单词的概率是多少？这个概率用 $p(\theta_d)$ 表示。如果 $p(\theta_d)$的值大，那么期望 $p(\theta_d \mid \text{text})$的值也大。
- 如果 θ_d 被选中用于生成单词，那么观察到 text 的可能性有多大？这个概率由 $p(w|\theta_d)$表示。如果 $p(\text{text}|\theta_d)$的值大，那么期望 $p(\theta_d \mid \text{text})$的值也大。

可以使用贝叶斯法则来刻画上述直观因素，从而完成 $p(\theta_d \mid \text{text})$的推断。在这种方法中，需要比较 $p(\theta_d)p(\text{text}|\theta_d)$和 $p(\theta_B)p(\text{text}|\theta_B)$两者的大小，来看 text 最有可能从哪个分布($\theta_d$ 还是 θ_B)中生成，如图 17-23 所示。

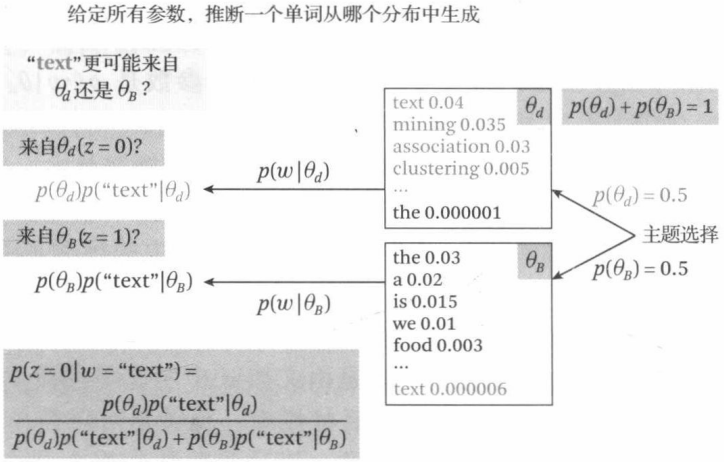


图 17-23　推断一个词从哪个分布中生成

上述处理方法是贝叶斯推理技术的一个典型应用示例。在推理过程中可以获取两个概率分布用来生成单词的先验知识(也就是 $p(\theta_d)$和 $p(\theta_B)$)。之所以称为先验，是因为这些信息都是在观察到单词 text 之前可以获得的。如果先验 $p(\theta_d)$更大，它将鼓励从 θ_d 中生成 361
单词。最终的概率是使用先验和数据似然(即 $p(\text{text}|\theta_d)$和 $p(\text{text}|\theta_B)$)一起计算得出，以便我们选择赋予 text 高概率的那个分布。

图 17-23 的例子使用了两个模型是等概率的先验，因此，这是无信息的先验知识(即没有偏见)。因此，推断使用哪个分布生成单词仅取决于概率 $p(w|\theta_d)$和 $p(w|\theta_B)$。由于 $p(\text{text}|\theta_d)$的值比 $p(\text{text}|\theta_B)$大，所以 θ_d 更有可能用于生成 text。一般来说，先验可能偏向某个特定的分布，一个严重有偏的先验甚至可以压制数据似然起到最终的决定作用。例如，假设先验规定 $p(\theta_B)=0.99\,999\,999$，那么推断的结果将是 text 更可能从 θ_B 中生成而不是 θ_d，即使 $p(\text{text}|\theta_d)$的值比 $p(\text{text}|\theta_B)$大。贝叶斯法则为我们提供了一种结合先验和数据似然的重要方法。

在图 17-23 中，我们引入了一个二元隐变量 z 来表示单词来自背景模型还是来自主题模型。当 z 为 0 时，表示单词来自主题模型；z 为 1 时，表示单词来自背景模型。后验概率 $p(z=0|w=\text{text})$刻画了使用哪个分布生成 text 的猜测，它和先验 $p(\theta_d)$与似然 $p(\text{text}|\theta_d)$的乘积是成比例关系的。这种比例关系直观上非常容易理解，因为为了从 θ_d 中生成单词 *text*，需要首先选择 θ_d(与之相对的是 θ_B)，这由 $p(\theta_d)$决定。然后从选择的 θ_d 中获得单词 text，这由 $p(w|\theta_d)$决定。

基于试探性的参数值，理解如何使用贝叶斯推理来判断使用哪个分布来生成单词，对于理解 EM 算法是非常关键的。EM 算法中 E 步的本质是用贝叶斯法则在混合模型的所有组件间分配数据点的来源概率。

注意到，E 步的本质是判断哪个单词从 θ_d 中生成(同样，哪个单词从 θ_B 中生成)，但是它不是分配一个完整的单词到 θ_d(或者 θ_B)，而是将一个单词在两个分布之间进行分配。$p(z=0|\text{text})$表示单词 text 应该分配到 θ_d 中的比例，因此有助于 θ_d 的估计。通过这种方式，能够收集分配到 θ_d 中的所有单词，并重新对它们的频数进行归一化，以获得对 $p(w|\theta_d)$潜在改进的估计。基于 E 步结果重新估计参数的步骤叫 M 步。 362

有了 E 步和 M 步作为基础，EM 算法可以描述如下。首先，随机初始化所有未知的参数。这样，就能够配置好一个混合模型的全部参数，从而能够进一步运用贝叶斯法则来推断哪个分布更可能生成每个单词。E 步的预测将单词按照分配比例划分到两个分布中。最后，收集属于主题单词分布的所有可能单词的“概率化”个数，并且归一化为概率分布作为改进估计的参数。可以重复这个过程来逐渐改善参数估计，直到似然函数达到局部最大值。EM 算法可以保证函数达到局部最大值，但不能保证在有多个局部最大值的情况下达到全局最大值。正是由于这一点，实践中通常使用不同的初始值多次重复算法，使用在最高似然时得到的参数估计值。

图 17-24 简要说明了 EM 算法的原理。图中引入二元隐变量 z，用来表示一个单词从背景模型中生成($z=1$)或者从主题模型中生成($z=0$)。例如，图中说明了 the 是从背景模型中生成的，因此 z 的值为 1，text 是从主题模型中生成的，因此 z 的值为 0。为了方便理解，这里假设每个单词都关联着一个未知的二元潜变量，但是在实际数据中无法观测到这些 z 变量的存在。这就是这样的变量被称为隐变量的原因。

EM 算法的主要思想是利用隐变量来简化最大似然估计的计算，因为确定隐变量的值以后，可以很容易进行最大似然估计。可以通过聚集 $z=0$ 的所有单词，并且对于计数归

一化。已知 z 值可以帮助简化最大似然估计的计算任务，EM 算法通过在每次迭代中交替进行 E 步和 M 步计算，以爬山的方式改进参数估计。

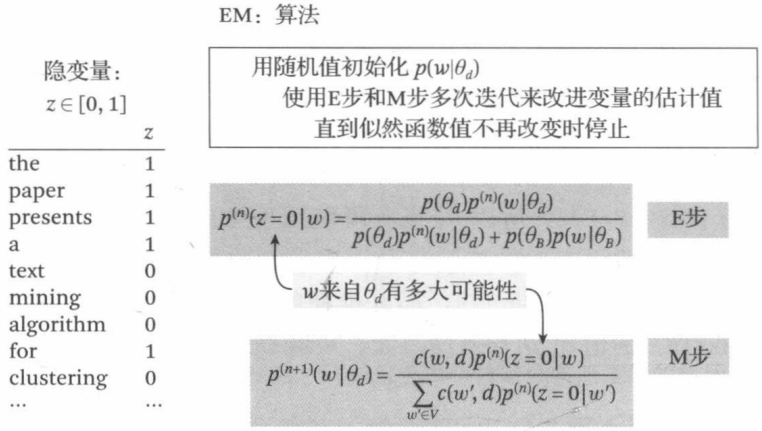


图 17-24　EM 算法

363

具体来说，E 步是推断所有单词的 z 值，M 步是使用推断的 z 值来划分单词到两个分布中，并使用分配到 θ_d 的数量来改进参数估计，从而得到一组改进后的参数值，这组参数值可以用来执行新的 E 步和 M 步的迭代，来进一步改进参数估计。

在 M 步中，根据 $p(z=0|w)$（也就是单词 w 从 θ_d 生成的概率）来调整 $c(w, d)$ 的值，这样可以得到“折扣过”的计数，即 $c(w, d)p(z=0|w)$。这个值可以被解释为单词 w 从 θ_d 中生成的期望值。同样，θ_B 也有自己对应的统计量，即 $c(w, d)p(z=1|w)=c(w, d)[1-p(z=0|w)]$，可得如下公式

$$c(w, d)p(z=0|w)+c(w, d)p(z=1|w)=c(w, d) \tag{17-5}$$

表明单词 w 的所有出现计数已经被分配到两个分布中。

因此，M 步只是对于所有“折扣后”的单词计数进行归一化来获得单词的概率分布，进而被当作 $p(w|\theta_d)$ 的改进估计。值得注意的是，在 M 步中，如果所有单词满足 $p(z=0|w)=1$，实际上就是在基于观测到的单词来计算单一一元语言模型（这是有道理的，因为 E 步表明背景模型不会生成单词）。

图 17-25 进一步说明了 EM 算法每次迭代中的具体情况。首先，使用 E 步和 M 步公式中的上标来表示参数的轮数。因此，M 步使用的是第 n 次迭代后的参数和新推断的 z 值，来获得新的 $n+1$ 次迭代的参数值（即 $p^{n+1}(w|\theta_d)$）。其次，假设两个组件（θ_d 和 θ_B）概率相等，并且背景模型单词分布是已知的（如表格第三列所示）。

364

EM 的计算是从准备相关的单词计数开始的。这个例子假设只有 4 个单词，文本中 4 个单词的计数结果如表格第二列所示。然后 EM 算法初始化所有待估计的参数。在这里，设置所有的概率为 0.25，如表格第四列所示。

在 EM 算法的第 1 次迭代中，我们将应用 E 步来推断使用哪个分布来生成单词，也就是计算 $p(z=0|w)$ 和 $p(z=1|w)$ 的值。图中仅列出了 $p(z=0|w)$ 的值，它将会用于 M 步中的计算（$p(z=1|w)=1-p(z=0|w)$）。$p(z=0|w)$ 对于不同的单词有不同的值，这是因为在背景模型和初始的 θ_d 中这些单词具有不同的概率值。因此，尽管两个分布被选择的概率相同并且 $p(w|\theta_d)$ 初始值是一个均匀分布，如果 $p(w|\theta_B)$ 有较高的概率值，那么 $p(z=0|w)$ 将试图赋予这些单词小概率值。例如 $p(z=0|\text{text})>p(z=0|\text{the})$。

$$\text{E步：}\quad p^{(n)}(z=0\mid w)=\frac{p(\theta_d)p^{(n)}(w\mid\theta_d)}{p(\theta_d)p^{(n)}(w\mid\theta_d)+p(\theta_B)p(w\mid\theta_B)}$$

$$\text{M步：}\quad p^{(n+1)}(w\mid\theta_d)=\frac{c(w,d)p^{(n)}(z=0\mid w)}{\sum_{w'\in V}c(w',d)p^{(n)}(z=0\mid w')}$$

假设 $p(\theta_d)=p(\theta_B)=0.5$ 和 $p(w\mid\theta_B)$ 已知

			第1次迭代		第2次迭代		第3次迭代	
单词	No.	$p(w\mid\theta_B)$	$P(w\mid\theta)$	$p(z=0\mid w)$	$P(w\mid\theta)$	$P(z=0\mid w)$	$P(w\mid\theta)$	$P(z=0\mid w)$
The	4	0.5	0.25	0.33	0.20	0.29	0.18	0.26
Paper	2	0.3	0.25	0.45	0.14	0.32	0.10	0.25
Text	4	0.1	0.25	0.71	0.44	0.81	0.50	0.93
Mining	2	0.1	0.25	0.71	0.22	0.69	0.22	0.69
对数似然				−16.96		−16.13		−16.02
似然增大→								

图 17-25　EM 算法的一个例子

一旦获得了这些 z 值的概率，就可以执行 M 步了，这些概率将用于调整相应单词的计数。例如，the 的计数是 4，但由于 $p(z=0\mid\text{the})=0.33$，当在 M 步中估计 $p(\text{the}\mid\theta_d)$ 时，将获得一个“折扣”值 4×0.33。同样，调整 text 的值为 4×0.71。在 M 步之后，如表格所示，$p(\text{text}\mid\theta_d)$的值将比 $p(\text{the}\mid\theta_d)$大(如第 2 次迭代的第一列所示)。根据 E 步，这些从主题单词分布 θ_d 中生成的单词也会有较高的概率。 365

新一轮的参数值可以进一步用于调整潜变量或者隐变量的推断值，这将导致 z 值概率的更新，它可以在下一个 M 步中用来改进 θ_d 的估计。

表格的最后一行列出了每次迭代之后的对数似然函数值。因为每次迭代将产生新一轮的参数估计值，所以对数似然函数的值也会不同。因为概率在 0～1 之间，在经过对数运算之后就变为负数了，所以对数似然值是负数。可以看到，在每次迭代之后，对数似然值是在增加的，这表明 EM 算法在爬山过程中迭代地改进了估计的参数值。对于为什么它最后收敛于一个局部最大值，下面将给出一个直观的解释。

现在值得指出的是，EM 算法的主要目标是获得更有识别力的单词分布来表示发现的主题，也就是 $p(w\mid\theta_d)$。收敛后，推断得到的 $p(z=0\mid w)$是有意义的，有时它也发挥一些额外的作用。具体来说，这些数值刻画了单词来自主题分布的概率，可以使用这些概率进行加和，来估计文档中背景模型和主题模型的覆盖程度，或者文档内容在多大程度上偏离了“典型”的背景文档。这将给每篇文档一个数值得分，这些分数可以用来比较不同文档或文档子集(例如，来自不同的作者或者不同的来源)。因此，简单的两组件混合模型不仅可以从文档中发现单一主题，还可以在一些应用中提供一个度量文档典型程度的有效方法。

接下来，在图 17-26 中，我们对为什么 EM 算法将收敛于一个局部最大值提供一些直观的解释。这里用 X 轴表示参数 θ_d 的值，用 Y 轴表示似然函数值。这个图是简化过的，因为实际情况中 θ_d 是 M 维向量。使用一维表示可以更容易理解 EM 算法。在一般情况下，原来的似然函数(θ_d)可能有多个局部极大值。计算最大似然估计的目标是找到全局最大值，也就是使似然函数达到全局最大值的 θ_d。

EM 算法是一种爬山算法。首先，它随机初始化参数值，然后逐渐改进它们的值。图中展示了从第 n 次迭代到第 $n+1$ 次迭代的场景。在第 n 次迭代中，参数当前估计值是 366

$p^{(n)}(w|\theta_d)$，在图中可以看出不是最优的取值。在 E 步中，EM 算法（概念上）计算一个辅助函数，也就是原始似然函数的下界。下界意味着对于任意 θ_d 的值，辅助函数的值都不会大于原始似然函数的值。在 M 步中，EM 算法找到一个参数值使辅助函数值达到最大，将这个参数作为 $p^{(n+1)}(w|\theta_d)$的改进估计。因为辅助函数是原始似然函数的下界，最大化辅助函数确保新的参数使得原始似然函数有更大的值，除非它已经达到了局部最大。在这种情况下，最大化的辅助函数的值也是原始似然函数的局部最大值。这就是 EM 算法可以保证达到局部最大值的原因。好奇的读者可能已经想到为什么不直接在原始的似然函数上找一个改进的参数值呢。虽然这有时是可行的，但是在 EM 算法中，优化辅助函数通常比优化原始似然函数简单。所以在这个意义上，它将原始问题规约为一个更简单的问题。尽管辅助函数通常更容易优化，但它并不总是有解析解，这意味着辅助函数的最大化可能需要另一个迭代过程，这个过程将嵌入在 EM 算法的整个迭代过程中。

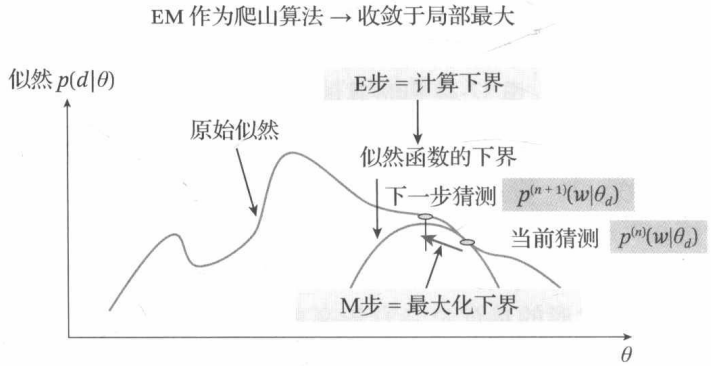


图 17-26　EM 作为爬山算法来优化似然

367

在简单混合模型的例子中，E 步没有显式计算这个辅助函数。因为辅助函数非常简单，M 步能够有解析解，从而可以绕过这个辅助函数的显式计算，并且直接找到一个参数新的估计值。因此在 E 步中，我们只需要计算辅助函数中的一个关键组件，即一个单词分别从两个分布中生成的概率。此外，M 步直接给出了优化辅助函数问题的解析解，这个解析解直接使用了从 E 步得到的值。

EM 算法有很多应用。例如，混合模型的参数估计通常都可以通过使用 EM 算法来求解。引入混合模型中的隐变量通常用于表示哪个组件模型用来生成数据点。因此，一旦知道这些隐变量的值，就能够对数据进行分配，确定数据点可能从哪个特定的分布中生成，从而方便组件模型的参数估计。一般来说，应用 EM 算法时，我们会利用隐变量来增广原始数据从而简化估计问题。EM 算法的步骤如下。第一，随机初始化所有要估计的参数。第二，在 E 步中，基于当前迭代的参数值推断隐变量的值，并获得隐变量取值的概率分布。直观来说，这是希望对隐变量获得一个较好的推测。第三，在 M 步中，使用推断的隐变量值来计算改进参数的估计。重复这个过程直到收敛到似然函数的局部最大值。尽管似然函数保证收敛到局部最大值，但无法保证参数估计一直稳定收敛到一个特定的值。也就是说，即使似然函数值还在增加，但是参数值可能还在振荡。只要参数满足某些条件，EM 算法就会保证收敛（见 Wu 1983）。

17.4　概率潜在语义分析

在这一节中，我们通过一些应用来引入概率潜在语义分析（PLSA）这一最基础的主题

模型。简而言之，PLSA 是简单的两组件混合模型的泛化形式，可以从文本数据中挖掘多个主题。在本章前面我们介绍了两组件混合模型。因此，如果读者已经理解了两组件混合模型，就会很容易理解 PLSA 的工作原理。

368

如之前所述，主题分析任务是指从文本文档中挖掘多个主题，并计算文档中每个主题的覆盖率。PLSA 正是用来完成这个任务的模型。所有的主题模型都会有两个关键假设。第一，假设可以用一个单词分布表示一个主题(或更一般的词项分布)。第二，假设一个文本文档是来自一个概率模型的单词样本。图 17-27 说明了这两个假设。图中展示了一篇关于卡特里娜飓风的博客文章和一些假想设置的主题。每个主题可以由一个单词分布表示，这些主题包括关于政府应急反应的主题(θ_1)、关于新奥尔良城市洪水的主题(θ_2)、关于捐赠的主题(θ_3)以及背景主题 θ_B。文章被认为包含来自所有分布的单词。具体来说，摘录文章的开头是对城市洪水的讨论，紧随其后的是对政府应急反应的批评，然后有一个关于捐赠的语句，同时在整篇文档中混有背景单词。

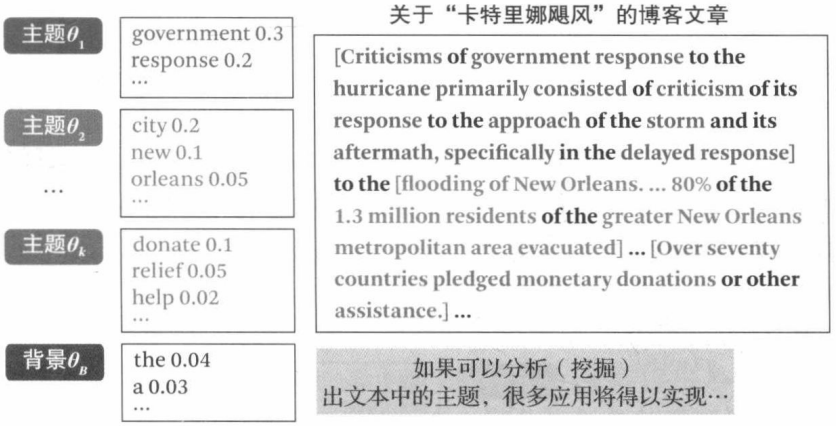


图 17-27　由混合主题的单词生成的文档例子

主题分析的主要目的是尝试挖掘隐藏在文本(段落)中的隐含主题，并且判断哪些词来自哪个分布，这样可以获得文本数据中的所有主题特征以及它们在每篇文档中的覆盖率。一旦完成这些挖掘与分析，得到的模型输出就可以直接在许多应用中使用，如摘要、语义切分和聚类。

369

图 17-28 展示了从文本中挖掘多个主题的形式化定义。输入的是文本数据集合、主题个数以及词汇表集合。输出有两种类型。一种是主题集合，每个主题由 θ_i 表示，它为单词分布。另一种是每篇文档的主题覆盖分布，其中 π_{ij} 是指文档 d_i 中主题 θ_j 的覆盖率。

挖掘文本主题这一任务可以使用 PLSA 来完成。PLSA 是简单两组件混合模型的拓展与泛化，可以用来刻画多个主题。如图 17-29 所示，在这样的生成模型中我们保留了背景模型(之前用于挖掘单一主题)。不同于之前讨论的简单混合模型，该模型包含 k 个组件模型，每个都代表一个不同的主题，可以用来生成文本数据中的单词。再加上背景模型 θ_B，PLSA 是由 $k+1$ 个组件组成的一元语言模型。[⊖]

和简单混合模型一样，PLSA 模型生成一个单词也包括两个步骤。第一步是选择使用哪个组件模型：这是由参数 λ_B(表示选择背景模型的概率)和参数 $\pi_{d,i}$(表示在不使用背景

⊖ 原始的 PLSA[Hofmann 1999]是不包含背景模型的，因此如果在预处理阶段没有删除常用词，它将在学习到的主题中赋予常用词很高的概率。

模型的情况下，选择主题 θ_i 的概率)控制的。如果不使用背景模型，就必须从 k 个主题中选择一个，所以有约束 $\sum_{i=1}^{k}\pi_{d,i}=1$。因此，选择背景模型的概率是 λ_B，而选择主题 θ_i 的概率是 $(1-\lambda_B)\pi_{d,i}$。

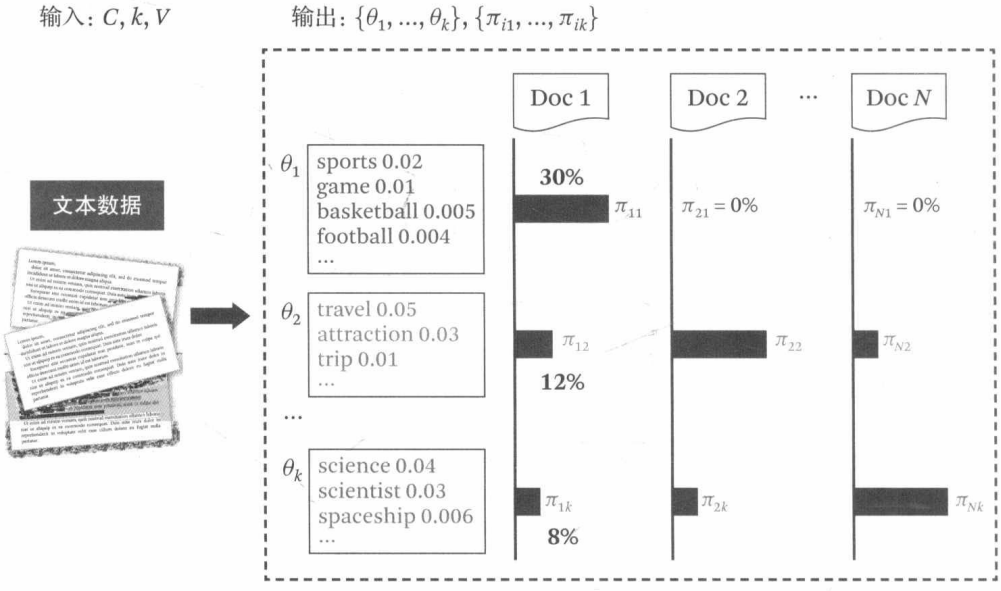


图 17-28　挖掘文本中的多个主题

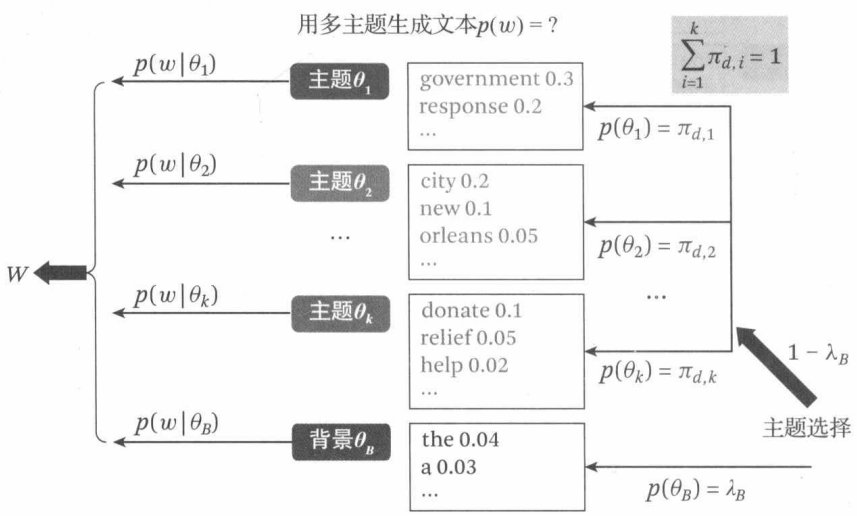


图 17-29　从多主题混合模型中生成单词

一旦决定使用哪一个单词分布，生成过程的第二步就是从选定的分布中生成一个单词，这和简单混合模型是相同的。

370 ~ 371

通常情况下，完成生成模型的设计后，下一步就是写出似然函数。首先思考这样一个问题，从混合模型中观察到一个单词的概率是多少？和简单的混合模型一样，其生成概率是生成该单词所有方式的和。这里一共有 $k+1$ 个不同的组件模型，所以有 $k+1$ 项的和，其中每一项是从对应组件的单词分布中观察到单词的概率，这可以进一步写成选择特定组件的概率与从特定单词分布中生成特定单词的概率的乘积。

图 17-30 对似然函数进行了说明。具体来说，从背景模型中观察到一个单词的概率是

$\lambda_B p(w|\theta_B)$，从主题 θ_j 中观察到一个单词的概率是$(1-\lambda_B)\pi_{d,j}p(w|\theta_j)$。从混合模型中观察到单词的概率 $p_d(w)$是所有情况的和。

概率潜在语义分析（PLSA）

背景词的比例（已知）　背景语言模型（已知）　主题θ_j在文档d中的覆盖率　词w在主题θ_j中的概率

$$p_d(w) = \lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k}\pi_{d,j}p(w|\theta_j)$$

$$\log p(d) = \sum_{w\in V} c(w,d)\log[\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k}\pi_{d,j}p(w|\theta_j)]$$

$$\log p(C|\Lambda) = \sum_{d\in C}\sum_{w\in V} c(w,d)\log[\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k}\pi_{d,j}p(w|\theta_j)]$$

未知参数：$\Lambda = (\{\pi_{d,j}\},\{\theta_j\}), j=1,\ldots,k$

图 17-30　PLSA 的似然函数

假设文档中单词是独立生成的，图 17-30 中的第二和第三个公式分别展示了文档 d 的似然函数和整个集合 C 的似然函数。

PLSA 模型中的参数有哪些呢？首先是 λ_B，它表示在文本数据出现的背景单词的比例（我们想把背景单词分离出来）。可以根据经验设置 λ_B 来控制主题模型的区分度。第二组是背景语言模型 $p(w|\theta_B)$。$p(w|\theta_B)$通常假设为已知，可以使用任何大规模文本集，或者集合 C 中所有可用文本来估计 $p(w|\theta_B)$（例如，假设所有文本数据都从 θ_B 中生成，可以使用最大似然估计来计算 $p(w|\theta_B)$，即单词 w 计数归一后得到的概率）。第三组是 $\pi_{d,j}$，它表示文档 d 中主题 θ_j 的覆盖情况，这是主题分析任务的重要输出之一。最后是 k 个单词分布，每个分布代表一个主题 $p(w|\theta_j)$，这也是主题分析任务的重要输出之一。你能计算出 PLSA 模型中有多少未知参数吗？这个练习有助于理解 PLSA 模型用于文本数据分析的输出结果。 372

在得到似然函数之后，接下来的问题是如何进行参数估计。通常可以使用如图 17-31 所示的最大似然估计。与简单混合模型相似，这个问题的本质是一个约束优化问题，除了以下的不同：

带约束的优化：　$\Lambda^* = \arg\max_{\Lambda} p(C|\Lambda)$

$$\forall j\in[1,k], \sum_{i=1}^{M} p(w_i|\theta_j) = 1 \qquad \forall d\in C, \sum_{j=1}^{k}\pi_{d,j} = 1$$

图 17-31　PLSA 的最大似然估计

- 现在有一个文档集合而不仅仅是一个文本文档。
- 需要估计更多的参数。
- 有更多的约束方程（这是多个参数的结果）。

尽管有更多的约束方程，各种约束的本质都与之前相同。一共有两种约束：一是每篇文档所有主题的覆盖率和为 1，二是每个主题中所有单词的概率和为 1。

与简单混合模型类似，可以使用 EM 算法来计算 PLSA 的最大似然估计。在 E 步中，由于有更多的主题，所以需要引入更多的隐变量。隐变量 z，是单词的主题指示器，z 现在有 $k+1$ 个候选值{1，2，…，k，B}，对应 k 个主题和一个额外的背景主题。如图 17-32 所示，在 E 步中使用贝叶斯法则来推断 z 的每种取值的概率。对比简单两组件混合模型 E 步中的那些公式，可以发现 PLSA 模型与其在本质上是相同的，只是现在有更多的主题。如果假设仅有一个主题，即 $k=1$，就能得到简单混合模型 E 步的公式，唯一的区别是

373 $p(z_{d,w}=j)$不表示单词从主题 θ_j 中生成的概率，而是表示在不选择背景模型条件下单词从主题 θ_j 中生成的概率。换句话说，使用 θ_j 生成单词的概率是$(1-p(z_{d,w}=B))p(z_{d,w}=j)$。在只包含一个主题而不包含背景模型的例子中，对于 θ_j 有 $p(z_{d,w}=j)=1$。

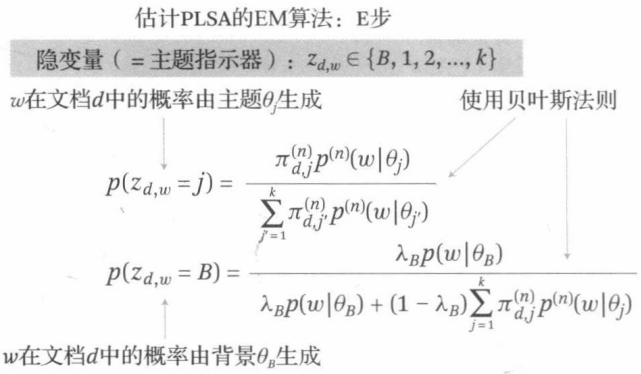


图 17-32　PLSA 参数估计：EM 算法之 E 步

注意，在这里我们使用文档 d 来索引单词 w。在模型中，w 是否从指定的主题中生成取决于文档。参数 $\pi_{d,j}$ 是与每篇文档有关的，因此每篇文档在主题覆盖分布上是不同的。这一假设也是符合直觉的：不同的文档聚焦的主题往往不同。也就是说，在 E 步中，同一单词在不同文档中的主题推断概率是不同的，因为不同的文档有不同的 $\pi_{d,j}$ 值。

M 步与简单混合模型的 M 步也是相似的，公式如图 17-33 所示。在 π 和 $p(w|\theta)$的重新估计中，两个公式都包含一个关键项 $c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)$，它表示为主题 θ_j 分配单词 w 的计数。直观上，使用 E 步中推断的 z 值的分布来划分 w 的计数到所有分布。θ_j 能够得到的 w 的计数是由推断出的 w 从主题 θ_j 中生成的可能性所决定的。

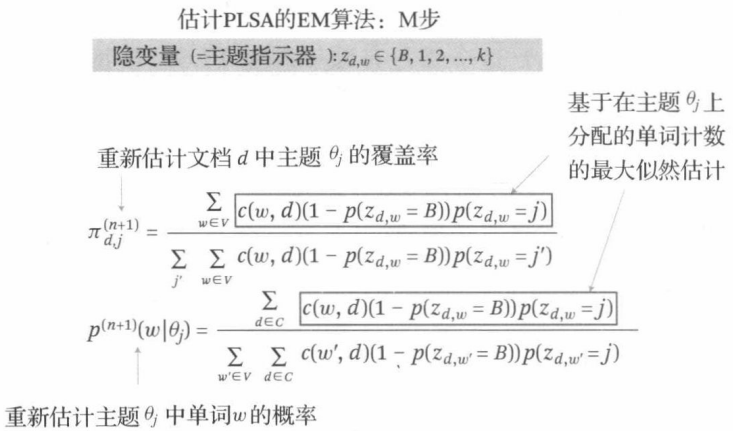


图 17-33　PLSA 参数估计：EM 算法之 M 步

如图 17-33 所示，一旦知道每个单词分配到每个分布的数量，就可以统计这些划分后的计数来重新估计 π 和 $p(w|\theta)$。为了重新估计 $\pi_{d,j}$，即文档 d 中主题 θ_j 的覆盖率，需要统计文档 d 中属于主题 θ_j 的所有划分后的单词计数，然后在 k 个主题上进行归一化获得概率分布。相似地，为了重新估计 $p(w|\theta_j)$，需要统计集合中所有文档属于 θ_j 的划分后的单词计数，然后在所有单词上进行归一化。注意在这两种情况下，归一化的目标并不相同，这与参数约束条件有直接关系。在重新估计 π 的情况中，约束条件是每个文档 π 值的和应为 1，因此归一化必须确保每篇文档重新估计的 π 值和为 1；而在重新估计 $p(w|\theta)$的过程

中，归一化需要保证每个主题是一个单词分布。

上述过程可以拓展到通用 EM 算法的使用。也就是说，在 E 步中计算的隐变量的分布可以被用于计算一个事件的期望计数，之后可以被用于聚合和归一化以获得参数新的估计值。在 EM 算法的实现中，可以保留不同事件的计数，然后对不同参数进行相应的归一化运算来获得新的估计值。

图 17-34 展示了 PLSA 模型使用 EM 算法求解的更多计算细节。首先随机初始化所有未知参数，包括每篇文档 d 的主题覆盖分布 $\pi_{d,j}$ 以及每个主题的单词分布 $p(w|\theta_j)$。初始化之后，EM 算法将迭代到似然函数收敛为止。如何判断似然函数已经收敛了呢？可以记录每次迭代时的似然函数值，然后和前一次迭代的似然值或者之前几次迭代的似然平均值进行比较。如果当前的似然值与之前的相差不多(由一个阈值决定)，就可以认为似然函数已经收敛，并且中止算法。

374 ~ 375

- 初始化所有未知随机参数
- 重复直至似然函数收敛

-E步　$p(z_{d,w}=j) \propto \pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)$　　$\sum_{j=1}^{k} p(z_{d,w}=j)=1$

$p(z_{d,w}=B) \propto \lambda_B p(w|\theta_B)$　← 这一项的归一化因子是什么?

-M步　$\pi_{d,j}^{(n+1)} \propto \sum_{w \in V} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)$　　$\forall d \in C, \sum_{j=1}^{k} \pi_{d,j}=1$

$p^{(n+1)}(w|\theta_j) \propto \sum_{d \in C} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)$　　$\forall j \in [1,k], \sum_{w \in V} p(w|\theta_j)=1$

图 17-34　估计 PLSA 模型的 EM 算法的计算过程

在每次迭代中，EM 算法首先调用 E 步，再调用 M 步。在 E 步中，使用隐变量的预测值来进行数据增广。在这种情况下，隐变量 $z_{d,w}$ 表示 d 中的单词 w 是来自“真的”主题还是背景模型。如果来自真的主题，$z_{d,w}$ 表明单词来自于 k 个主题中的哪一个主题。

在图 17-34 中，E 步需要计算每篇文档中每个不同单词的 z 的取值概率。因此，可以遍历所有文档，同时遍历每篇文档中所有单词来完成对应 $p(z_{d,w})$ 的计算。这个过程涉及计算选择主题的概率与选择指定分布中单词 w 的概率的乘积。然后可以基于约束条件 $\sum_{j=1}^{k} p(z_{d,w}=j)=1$，对得到的乘积进行相应的归一化。这里的归一化计算是对所有主题进行的。

在 M 步中，我们也将统计相关计数然后进行相应的归一化操作，以获得不同参数的重新估计。在这一步，将会使用 E 步估计的概率分布 $p(z_{d,w})$ 将文档 d 中单词 w 的计数划分到所有主题上。值得注意的是，同一单词在不同的文档中将会按照不同的方式划分。一旦将单词按照这种方式划分后，就可以进行重新计数，并进行归一化处理。例如，为了重新估计 $\pi_{d,j}$(文档 d 中主题 θ_j 的覆盖率)，相关计数是文档 d 中与主题 θ_j 有关的单词计数，归一化因子是所有主题的计数和，这样归一化后就可以获得所有主题的概率分布。同样，为了重新估计 $p(w|\theta_j)$，相关计数是所有文档中单词 w 的划分后的计数和，然后将基于词汇表中所有单词的计数和进行归一化处理。归一化之后，能够再得到一个分布，这次获得的分布是对所有单词的分布而不是所有主题的分布。

376

如果在 M 步开始之前完成了 E 步的所有计算，那么需要分配更多的内存来记录 E 步的所有结果。然而，可以合理安排 E 步和 M 步交错进行。在 E 步计算中，可以收集和统计 M 步需要的相关计数进行使用，这将防止保存许多不必要的临时变量。

17.5　PLSA 的扩展及潜在狄利克雷分布

作为一种完全无监督的学习方法，PLSA 能够分析文本数据中的主题信息，不需要任何人工指导和监督信息。尽管最大限度地减少人工干预是 PLSA 的一个优势，但是主题的发现完全由数据特征决定，没有考虑数据集关于主题的任何额外知识。很多情况下，能够获得这样额外的知识或者应用任务有特殊的主题偏好，所以加入一些关于参数估计的先验知识是有用的甚至是必要的。这样，估计得到的参数不仅能解释文本数据，还符合先验知识。先验知识或偏好可用于所有的参数。

首先，用户可能对文本数据中要分析的主题信息有一些期望，这样的知识可以用来定义关于主题单词分布的先验。例如，分析人员可能希望将“检索模型”作为一个关于信息检索研究文章的主题，因此可以让模型为“检索模型”分配一个主题。类似地，用户可能对分析笔记本电脑的特定方面感兴趣，例如电池寿命、屏幕大小等，因此希望该模型为电池寿命和屏幕大小分别分配两个主题。

其次，用户可能知道文档中涉及(或不涉及)哪些主题。例如，如果有用户给文档打的
377 (主题)标签，可以把这些标签作为文档中所包含主题的信息。因此，可以为主题覆盖定义一个先验，给定一个文档，以确保只使用与分配给它的标签相对应的主题来生成文档内容。这实际上是一个限制，它规定了在文档中使用什么主题来生成单词。这对于在以主题为上下文学习共现单词是很有用的，因为在数据稀疏的情况下，只基于共同出现的统计数据不足以形成一个有意义的主题。

通过使用最大后验(MAP)估计而非最大似然估计，所有这些先验知识都可以引入 PLSA 模型中。将所有的参数都表示为 Λ，并对 Λ 的所有可能值引入一个先验分布 $p(\Lambda)$来反映我们的偏好。这样的先验分布包括所有可能的词分布(用于主题表征)，以及所有可能的文档主题覆盖分布(用于主题覆盖)，并且可以灵活刻画希望融入模型里的各种知识和偏好。有了这样的先验，就可以用 MAP 来估计参数了：

$$\Lambda^* = \arg\max_{\Lambda} p(\Lambda)p(\text{Data}|\Lambda) \tag{17-6}$$

其中，$p(\text{Data}|\Lambda)$是似然函数，它是最大似然估计情况下需要最大化的唯一部分。加入先验 $p(\Lambda)$将促使模型在最大似然估计(最大化 $p(\text{Data}|\Lambda)$)和先验(最大化 $p(\Lambda)$)之间寻求一个折中结果。

可以有很多种不同的方法来定义 $p(\Lambda)$。然而，使用共轭先验分布尤其方便。在此分布中，作为参数 Λ 的函数，先验概率函数 $p(\Lambda)$与似然函数 $p(\text{Data}|\Lambda)$具有相同的形式。由于这两个函数的形式相同，通常可以合并它们，从而得到一个融合的函数(具有同样的形式)。换句话说，后验分布可以看作参数 Λ 的函数，所以后验概率的最大化与似然函数的最大化相似。由于后验分布与原始数据的似然函数有相同的形式，后验分布可以看作一个“伪数据”集合的似然函数。这个“伪数据”集合是由原始的数据与附加的“伪数据”合并形成的，这样先验的影响完全是通过“伪数据”来反应的。

当使用共轭先验时，可以稍微修改前面介绍过的 PLSA 的 EM 算法来完成 MAP 的计算，
378 在这个算法中通过加入适当数量的伪数据来引入先验。作为一个具体的例子，如果在表示主题 $p(w|\theta_j)$的单词分布上定义一个共轭先验，那么计算 MAP 的 EM 算法就如图 17-35 所示。主要的不同是在 M 步中为单词 w 添加一个额外的伪计数，它与这个词的先验概率 $p(w|\theta_j')$成正比。具体来说，单词 w 的伪计数是 $\mu p(w|\theta_j')$。同时，分母需要进行相应调整(增加 μ，它是所有单词的所有伪计数的总和)，以确保一个主题中单词的概率之和为 1。

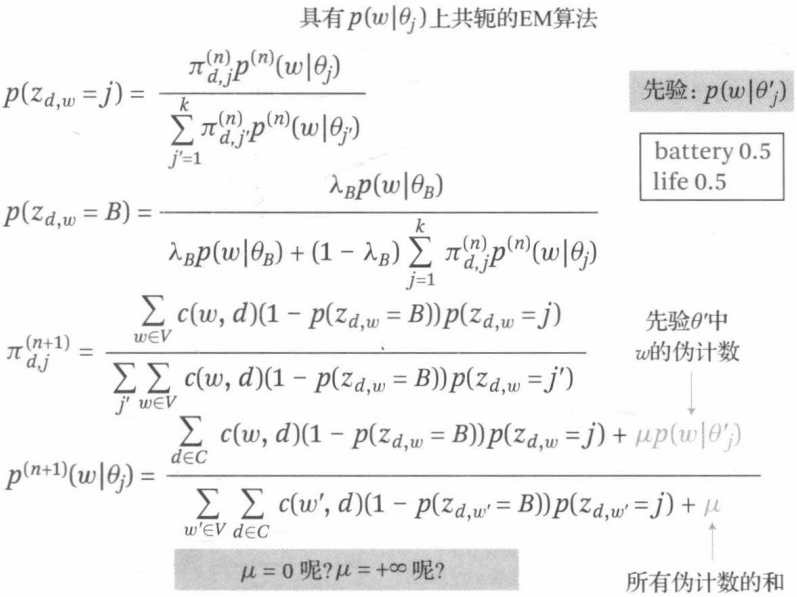


图 17-35　融入先验的 PLSA 最大化后验估计

这里，$\mu\in[0,+\infty)$是一个表征先验强度的参数。如果 $\mu=0$，即没有先验，会得到原始 PLSA 模型的 EM 算法。一个更有趣的例子是当 $\mu=+\infty$时，M 步实际上会直接设置单词的估计概率 $p(w|\theta_j)$为先验概率 $p(w|\theta'_j)$，即单词分布固定为先验概率分布。这就是为什么可以将一个背景词分布包含在 PLSA 中，并且看作一个主题的原因——这个主题上施加了一个无限强的先验。直觉上，在贝叶斯推断中，如果先验无限强，那么无论看到多少数据都无法改变先验。但是在一般情况下，随着数据量增加，我们可以让数据逐渐占据 379
主导地位，并且最终完全覆盖先验。可以采用相似的方法来引入覆盖分布 $\pi_{d,j}$的先验，在参数更新中强制增加一些主题的概率而减少其他主题的概率。在极端情况下，为了将某一主题的概率置为零，也可以使用无限强的先验赋予该主题零概率。

PLSA 是一种生成模型，用于对给定文档中的单词进行建模，但它不是一个真正意义上的文档生成模型，因为它不能为一个新的未出现的文档赋予概率，也就是它不能给出所有可能的文档上的分布。然而，有时我们需要全部文档的生成模型。例如，如果每个主题类别都有这样一个模型，就可以将它用于文本分类中——比较每个类别对应的生成模型预测这个文档的概率，把这个文档分配到概率最高的那一类。在 PLSA 模型中，主题覆盖概率与观测到的文档紧密相关。对于新的未出现的文档来说，模型中没有可用的主题覆盖率，然而这个参数又是生成新文档时所必需的。尽管可以使用启发式方法来估计一个新的未出现的文档的主题覆盖分布，但一个更好的方法是在 PLSA 模型的参数上添加先验，从而获得贝叶斯版本的模型，这也导致了潜在狄利克雷划分(Latent Dirichlet Allocation，LDA)模型的诞生。

特别地，在 LDA 模型中，每个文档的主题覆盖分布(一个多项式分布)都被设定为来自于狄利克雷的先验分布，它定义了多项式分布在整个参数空间上的分布，即一个关于主题的概率向量。类似地，在一个文本集合中，表示隐含主题的所有单词分布也被设定为来自另一个狄利克雷分布。在 PLSA 中，主题覆盖分布和单词分布都被设定为模型中的(未知的)参数。作为对比，在 LDA 中，它们不再是模型的参数，而是来自相应的狄利克雷(先验)分布中的随机变量。

因此，LDA 中的参数只用来刻画这两种类型的狄利克雷分布。一旦确定了这些参数，

就可以确定这两个狄利克雷分布的行为，从而确定整个生成模型的行为。一旦对整个集合中所有主题对应的单词概率分布(共享这些主题)以及文档的主题覆盖分布进行了采样，那
380 么在文档中生成单词的其余部分与 PLSA 模型完全相同。图 17-36 中给出了从 PLSA 到 LDA 的泛化，控制主题覆盖的狄利克雷分布有 k 个参数 α_1，…，α_k，控制主题词分布的狄利克雷分布有 M 个参数 β_1，…，β_M。根据之前的讨论，每个 α_i 可以被解释为对应主题 θ_i 的伪计数，而每个 β_i 可以被解释为对应单词 w_i 的伪计数。由于没有额外的知识，可以将它们设置为相等的数值，即假设我们对每个单词分布中的任何单词都是无偏的，并且对每个文档中的任何主题也是无偏的。

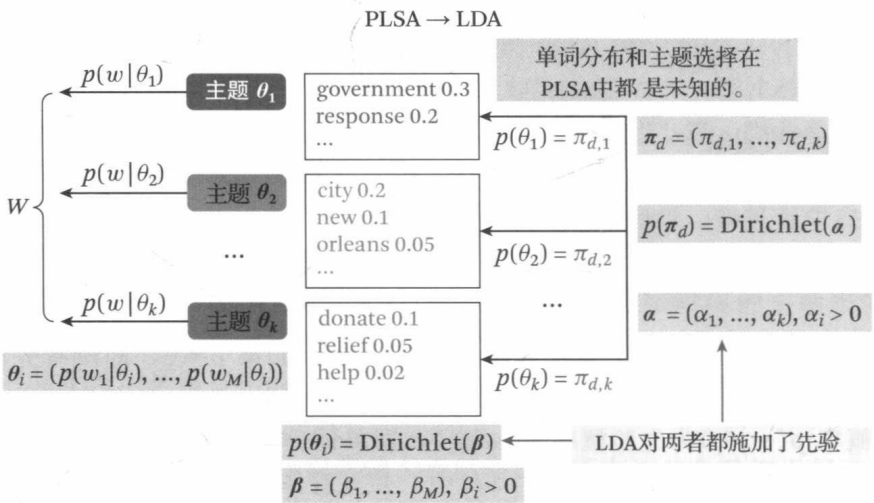


图 17-36　LDA 看作融入狄利克雷先验 PLSA 的说明

LDA 的似然函数在图 17-37 中给出。比较一下 PLSA 和 LDA 的似然函数，可以看到 PLSA 和 LDA 都有相同的生成模型组件。这个生成模型组件可以定义为从包含 k 个单词分布 θ_1，…，θ_k 的混合模型，观察到文档 d 中单词 w 的概率，其中混合系数为文档 d 的主题覆盖分布 $\pi_{d,j}$。实际上，这种混合的一元语言模型是多数主题模型中的常用组件，并且是刻画包含多主题文档的关键。然而，一个文档和整个集合 C 的似然函数与 LDA 明显
381 不同，LDA 的似然函数增加了主题覆盖分布的不确定性，以及所有单词分布的不确定性。

尽管 LDA 的似然函数比 PLSA 复杂，仍然可以使用最大似然估计来获得它的参数 $\boldsymbol{\alpha}$ 和 $\boldsymbol{\beta}$：

$$(\hat{\alpha}, \hat{\beta}) = \underset{\alpha,\beta}{\operatorname{argmax}} \log\ p(C|\alpha, \beta) \tag{17-7}$$

自然地，解决这种优化问题所需的计算比 PLSA 要复杂得多。

LDA 只有 $k+M$ 个参数，比 PLSA 少得多。然而，这是有代价的。在完成参数估计之后，不能够直接得到主题分析中有用的输出——描述一个集合中所有主题的 k 个单词分布 $\{\theta_i\}$ 以及每个文档的主题覆盖分布 $\{\pi_{d,j}\}$。实际上，作为贝叶斯推理的一种常见处理方法，为了获得 LDA 中隐变量的值，需要使用后验推断。也就是说，使用贝叶斯法则来计算 $p(\{\theta_i\}, \{\pi_{d,j}\}|C, \alpha, \beta)$：

$$p(\{\theta_i\}, \{\pi_{d,j}\}|C, \alpha, \beta) = \frac{p(C|\{\theta_i\}, \{\pi_{d,j}\}p\{\theta_i\}\{\pi_{d,j}\}|\alpha, \beta)}{p(C|\alpha, \beta)} \tag{17-8}$$

这个概率给出了这些变量所有可能取值的后验分布，从中可以进一步得到一个点估计，也可以计算其他依赖于这个分布的有趣性质。由于涉及某些概率的积分，计算过程再次变得

复杂。研究人员提出了许多不同的推理算法，其中一个非常流行和有效的方法是吉布斯采样，它的工作原理与 PLSA 的 EM 算法非常相似。

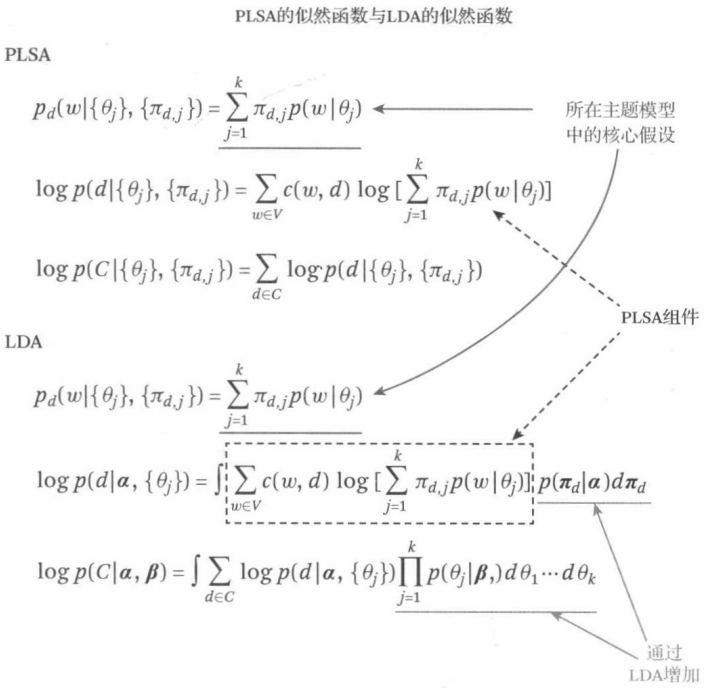


图 17-37　PLSA 和 LDA 的似然函数

从经验上来说，在学习文档的低维度语义表示时，LDA 和 PLSA 的工作原理是相似的(通过 $\pi_{d,j}$ 把文档映射到 k 维空间中)。学习到的单词分布看上去也非常相似。

17.6　主题分析的评价

主题分析的评价会遇到和信息检索评价一样的困难。在这两种任务中，通常都没有一个真正的答案，评价指标严重依赖于人工判断。一个主题的定义是什么呢？在定义模型的时候，处理这个问题是最合适的，但是这个看似简单的问题却使最终的评价任务变得复杂。

对数似然和模型困惑度是评价语言模型最常见的两种方法，它们同样可以用于主题分析。两者都是预测性度量，这意味着将数据提交给模型，并将模型应用于新的数据，计算其可能性。如果模型能够很好地扩展到这些新数据(通过赋予它很高的可能性或较低的困惑度)，那么就认为模型是有效的。

在第 13 章中，我们提到了 Chang 等[2009]的工作。人工裁判对词汇扰乱检测任务做出回应，以度量主题-单词分布的一致性。第二个测试是文档-主题分布评价，可以通过先前使用的扰乱测试来量化文档中发现的主题的一致性。

设置如下：给定集合中的一个文档 d，选择与其关联紧密的前三个主题，并将这三个主题称为 θ_1、θ_2 和 θ_3。另外一个低概率主题 θ_u 也被选中，并与前三个主题一起展示给人工裁判。展示的信息包括文档 d 的标题和一个内容片段，以及每个主题的几个高概率单词。人工裁判必须决定哪一个 θ 是引入的噪声主题 θ_u。与单词“扰乱测试”一样，如果最相关的三个主题与标题以及文档片段相匹配，那么人工裁判应该很容易判断出噪声主题。如果很难辨别 θ_u，那么选择的三个主题就没有很好地表示 d。当然，这个过程需要对集合

382
~
383

中的不同文档反复进行。

摘自 Chang 等[2009]：

…我们发现传统的度量标准不能评判主题是否一致。实际上，传统的度量标准与主题质量呈负相关。

“传统度量”指的是生成模型对于留存数据的对数似然。这种结果的不一致无疑是一个严重的问题，尽管大部分最新的研究仍然依赖于传统的评价标准。

作为另一种方式，下游应用任务的改进可能是最有效的(也是最透明的)评价指标。如果发现一个不同的主题分析模型显著地提高了某些任务的精确度，那么我们就更倾向于选择新的模型。例如，如果主题分析旨在为文本分类提供新的表示特征，那么分类准确性就是待改进的度量标准。在这种情况下，虽然模型的解释能力会因为主题不能很好地被人区分而被削弱，但是如果能提高分类的准确率，那么留存数据的对数似然甚至主题的一致性都不需要再被特别关注了。

17.7　主题模型的总结

本章主要介绍了主题分析技术。首先从一个简单的模型开始介绍，也就是使用一个词项来表示一个主题，并讨论了这种方法的不足。然后介绍了使用单词分布来表示主题的方法，这种单词分布也被称为一元语言模型。进一步介绍了 PLSA 模型，它是一个混合模型，使用 k 个一元语言模型来表示 k 个主题。此外，还添加了一个预先指定的背景语言模型来帮助发现有区分度的主题，因为这个背景语言模型可以帮助找出常见词。我们使用最大似然估计(利用 EM 算法计算)来估计 PLSA 的参数。估计的参数值能够帮助发现两种知识，一是 k 个单词分布，每一个表示一个主题，二是每个文档中每个主题的比例。

384 主题单词分布和每个文档的主题覆盖率有助于进一步的分析和应用。例如，可以聚合一个特定的时间段内的文档，用来分析该时间段内的主题分布情况，从而能够生成一个主题的时态变化趋势。同时，还可以收集特定作者发表的文档，分析其中涉及的主题，以揭示该作者的专业领域。此外，主题模型还可以用于词项聚类和文档聚类。事实上，每个主题词分布都可以看作是一个聚类(例如，可以通过选择具有最高概率的 N 个单词来获得聚类)。因此，很容易根据 PLSA 的输出得到词项的聚类。文档也可以以同样的方式进行聚类——可以将文档分配给其覆盖最多的主题聚类。回忆一下，$\pi_{d,j}$ 表示每个主题 θ_j 在文档 d 的覆盖率，可以将文档分配给具有最高 $\pi_{d,j}$ 的主题聚类。对 PLSA 结果的另一种使用方法是，将文档的主题覆盖分布作为文档在低维语义空间中的表示，每个维度对应一个主题。这样一种表示可以作为词袋表示的补充，以增强同一主题中词项的非精确匹配。这样一种表示非常有用，可用于信息检索、文本聚类和文本分类等任务。

最后，通过向文档主题分布和主题单词分布添加先验信息，LDA 模型拓展了 PLSA 模型。LDA 中引入的先验可以促使少量主题在一篇文档中占据主导地位。这种主题分布上的稀疏性是有意义的，因为通常文档主要讨论一个或两个主题，而不是所有 k 个主题的混合。其次，添加这些先验信息可以在每个主题中提供稀疏的单词分布，这类似于之前讨论过的单词的 Zipfian 分布。最后，LDA 是一个生成模型，它可以用来模拟(生成)模型中参数的值，并将模型应用到一个新的、未出现的文档[Blei 等 2003]。

书目说明和延伸阅读

本章提及了 PLSA 的原始论文[Hofmann 1999]和其变种 LDA 的原始论文[Blei 等

2003]。Asuncion 等[2009]对主题模型的各种推断方法进行了比较，并得出它们都非常相似的结论。为了评价主题模型，在这一章中引用了 Chang 等[2009]的工作，它提供了一些方便的数学度量方法，如与人类度量不相关的对数似然。关于书籍，Koller 和 Friedman [2009]对概率图模型做了大量详细的介绍。Bishop[2006]涵盖了大量机器学习算法，包括图模型、混合模型、EM 和推理等。Steyvers 和 Griffiths[2007]提供了一个简短的主题模 385
型调研总结。在练习中，我们提及了有监督的 LDA 模型[McAuliffe 和 Blei 2008]。此外还有许多其他的 LDA 变种，比如 MedLDA[Zhu 等 2009](另一种有监督的模型，试图最大化类间的距离)和 LabeledLDA[Ramage 等 2009](结合了元数据标签)。

练习

17.1　写出两主题混合模型的输入和输出。

17.2　写出 PLSA 的输入和输出。

17.3　在一个产品评论数据集中，共有 k 种不同的产品类型。k 的取值范围是[2，5]。你认为有多少种产品类型？如何使用主题分析来帮助你得到上述结果？(产品类型类似于“CPU”和“路由器”)。

17.4　写出一种使用主题模型来增强搜索结果的方法。该方法支持什么样的访问模式？

17.5　写出一种在向量空间中使用主题模型表示文档的方法。如何度量这种表示的相似性？

17.6　给定一个数据集，数据集中的文档带有时间标记，写出如何使用 PLSA 对主题随时间变化趋势进行建模。

17.7　第 18 章讨论了情感分析和观点挖掘。为了发现正面和负面的情感主题，我们设置 $k=2$ 并且使用主题分析的方法。这个想法存在什么问题？

17.8　本章中提及过 PLSA 是一种判别模型，而 LDA 是一种生成模型。讨论这些差异在如下方面的影响：

(a) 定义模型。

(b) 加入先验知识。

(c) 学习模型参数(推断)。

(d) 将模型应用到新数据中。

17.9　另一种主题分析评价方案是在一些文档中留存一定数量的词汇进行评价。解释该方案是如何评价主题模型的。这种方法评价了主题-文档分布，还是评价了主题-单词分布，亦或是两种都可以进行评价？ 386

17.10　有监督的 LDA(sLDA)模型是一个面向有标注文档的概率模型，其中每个文档都包含一些实值响应变量。例如，如果数据集是电影评论，那么响应变量可以是平均打分。与 LDA 或 PLSA 相比，sLDA 在预测新文档的响应变量时还能发现哪些额外的知识？ 387～388

第 18 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

观点挖掘与情感分析

本章中将讨论挖掘另一种类型的知识，即关于生成文本数据的观察者或者人类的知识。本章将特别讨论观点挖掘和情感分析。如前所述，文本数据可以被认为是以人类作为主观传感器生成的数据。相比之下，我们有其他的设备，比如录像机，可以记录真实世界中发生的事情来生成数据。文本数据和其他数据(如视频数据)之间的主要区别在于它具有丰富的观点，内容往往是主观的，因为它是由人类产生的，如图 18-1 所示。与其他数据相比，这实际上是文本数据的独特优势，因为这提供了一个了解观察者的绝佳机会——我们可以通过挖掘文本数据以了解其观点。

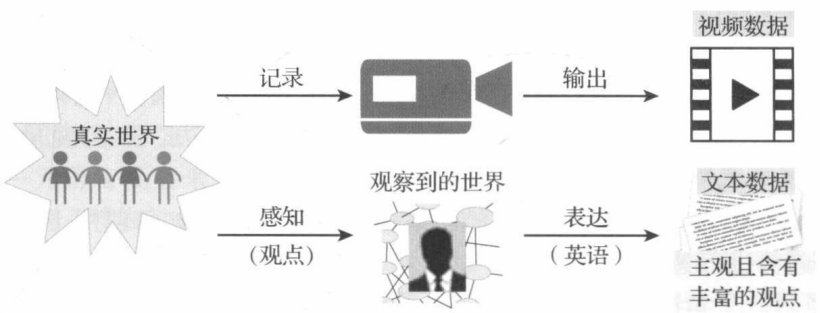


图 18-1 客观和主观传感器

我们从观点(opinion)的概念开始。正式定义观点并不容易，但在大多数情况下，我们将观点定义为描述某人对某事的信念或想法的主观陈述，如图 18-2 所示。

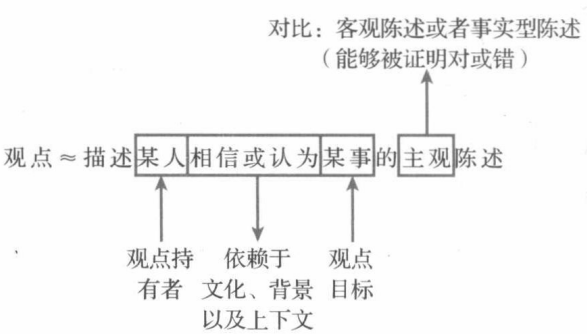


图 18-2 “观点”的定义

图中的关键词主观(subjective)，与客观(objective)陈述或事实型陈述形成对照。这是一个关键的区别因素，观点往往不容易被证明是错误的或正确的，因为它反映的是人们对某事物的看法。相比之下，客观的陈述则通常可以被证明是错误的或正确的。例如，你可能会说一台计算机有一个屏幕和一块电池。显然，你可以通过检查确认它是否有一块电池。与此相反，考虑这样一句话，如“这台笔记本电脑有最长的电池寿命”或“这台笔记本电脑有一个漂亮的屏幕”。这些说法更主观，很难证明是错误的还是正确的。

单词某人(person)表示观点持有者。当我们谈论观点时，这是一个人所持有的观点。当然，观点通常将依赖于文化、背景和上下文。这个思考过程表明，为了表征观点，我们需要包含多个要素。

下一个逻辑问题是“基本的观点表示是什么样的?”它至少应该包含三个要素。第一，它必须说明观点的持有者。第二，它还必须指定目标，或者观点是关于什么的。第三，当

然，还需要观点内容。如果你能识别出这些，就能够对观点有一个基本的了解。如果想进一步了解，则需要更丰富的观点表示。这意味着，我们也希望了解，比如说这个观点的背景以及在什么情况下表达的观点。我们也想了解观点的情感，即它是否是一种积极或消极的感受。

389
~
390

来看一个简单的产品评论的例子。在这个例子中，已知观点的持有者和针对的目标。当评论发布时，通常可以提取此类信息。通过分析用户生成的文本来得到额外的理解，进一步增加了挖掘观点的价值。

图 18-3 显示了从新闻文章中提取的一个句子。这个例子中有隐含的观点持有者和隐含的目标，因为我们不会自动得到这些信息。这使得任务变得更加困难。作为人类，我们可以将观点持有者认定为康涅狄格州的州长。我们也可以确定目标，飓风桑迪，但还有另外一个提到的目标是 1938 年的飓风。那么观点是什么？有表现负面情感的词，如坏(bad)和最差(worst)。我们也可以识别上下文，这是新英格兰。所有这些元素必须通过使用自然语言处理(NLP)技术来提取。分析新闻的情绪还是相当困难的，比分析产品评论中的观点要困难得多。

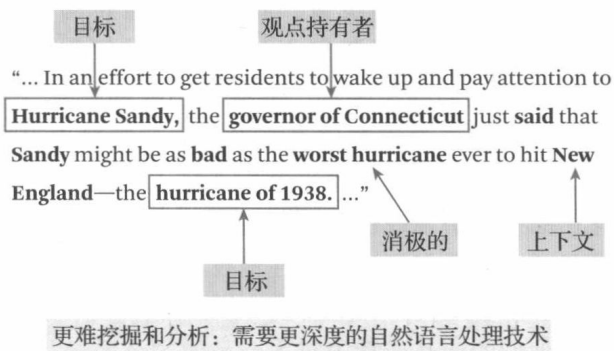


图 18-3　新闻中一个带有情感的句子(© 2012 Henry Blodget / Business Insider)

也有一些其他有趣的变种。首先，考虑观点持有者。持有者可以是一个人，也可以是一群人。有时候，这个观点来自于一个委员会或一整个国家的人。观点目标也将有很大差异，它们可以是关于一个实体、一个特定的人、某种特定的产品或者特定的政策，等等。观点也可能只是关于特定实体的某个属性。例如，它可能只是关于智能手机的电池。甚至可能是别人的观点，一个人可能会对另一个人的观点进行评论。显然，这里有很多变化会导致问题具有不同的形式。

391

观点内容(content)也可能在表面上有所不同：可以识别一句话或一个短语的观点，也可以有更长的文字来表达观点，如整篇新闻文章。此外，还可以识别观点持有者的情绪或情感的变化以及区分正面与负面或中性的情感。

最后，观点上下文也可能有所不同。可以有简单的上下文，如不同的时间或不同的位置。还可能会有复杂的上下文，例如，正在讨论的主题的一些背景。当一个观点在一个特定的语境中表达时，相对于在另一个语境中，它必须用不同的方式来解释。

从计算的角度来看，我们主要关心可以从文本数据中提取什么样的观点。一个计算目的可能是确定观点的目标。例如，“我根本不喜欢这个电话”，显然是这个说话者关于电话的观点。相比之下，文本也可能报告其他人的观点。我们可以观察另一个人的观点并报告这个观点。例如，“我相信他爱这幅画”。这个观点实际上是由另一个人表达出来的，这并不意味着当前说话者喜欢那幅画。显然，这两种观点需要以不同的方式进行分析。有时，评论者可能会提及他(她)朋友的观点。

另一个复杂因素是，可能需要通过对文本中表达的内容进行推论而获得的间接观点或推论观点。例如，一个声明可能是，“这个手机一个小时内就耗尽了电池。”现在，这是一种事实陈述，因为它要么是真要么是假。但是，我们也可以推断出一些关于手机电池质量的负面观点，或者关于电池寿命的观点。这些是当我们提取观点时需要注意的有趣的

变化。

观点挖掘任务可以定义为使用上下文语境的输入来生成一组观点表示，如图 18-4 所示。每个表示应该确定观点持有者、目标、内容和上下文。理想情况下，我们也可以从评论和上下文推断出观点的情感，以更好地理解观点。通常，表示中的一些元素是已知的。我们刚刚看到产品评论的例子，观点持有者和观点目标通常被显式地识别了。

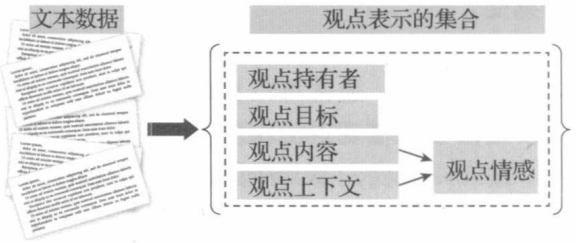


图 18-4　观点挖掘任务

观点挖掘是重要和有用的，有三个主要原因。第一，它可以辅助决策支持，帮助我们优化我们的决策。我们经常通过阅读他人的评论来了解别人的观点，以便作出决定，例如购买哪种产品或使用哪种服务。当我们决定投谁的票时，我们也会对别人的观点感兴趣。政策制定者在设计新政策时也可能想了解他们选民的观点。第二个应用是了解人。例如，它可以帮助我们了解人的偏好。如果我们知道人们对什么感兴趣，我们可以优化产品搜索引擎或优化推荐系统。它也可以对投放广告有益，如果我们知道哪类人往往喜欢哪类产品，我们可以针对性地投放广告。第三种应用是一次聚集许多人的观点，以评估更一般的观点。这对于商业智能来说是非常有用的，制造商想知道他们的产品在哪里有优势或劣势。他们的产品或竞争对手产品的“杀手锏”是什么？市场研究需要理解消费者的观点。数据驱动的社会科学研究可以从中受益，因为他们可以通过文本挖掘来理解群体观点。如果聚合社交媒体的观点，我们可以研究人们在社交网络中的行为。一般来说，我们可以在任何预测任务中获得巨大的优势，因为我们可以利用文本数据作为任何问题的额外数据。

18.1　情感分类

如果假设观点表示中的大部分元素都是已知的，那么唯一的任务就是情感分类。也就是说，知道观点持有者和观点的目标，也知道观点的内容和上下文，唯一剩下的部分是确定评论的观点情感。情感分类可以更具体地定义如下：输入是一个有观点的文本对象，输出通常是可以以两种方式定义的情绪标记(sentiment label)(或情绪标签(sentiment tag))。一个是带有类别的极性分析(polarity analysis)，如正面(积极)、负面(消极)或中性。另一种是情绪分析(emotional analysis)，可以超越极性，以表征观点持有者的确切感受。在极性分析的情况下，有时也会具有数值评分，就如你在网络上的一些评论中经常看到的那样。例如，五星级可能表示最正面的，也可能是最负面的。在情绪分析中，也有不同的设计类别的方法。一些典型的类别是快乐、悲伤、恐惧、愤怒、惊讶和厌恶。因此，任务本质上就是之前看过的分类任务或分组任务。

如果简单地应用默认分类技术，准确性可能不会太好，因为情感分类需要对常规文本分类技术进行一些改进。特别是需要两类改进。一个是使用可能更适合情感标记的更复杂的特征。另一个是考虑这些类别的顺序，特别是在极性分析中，因为选择之间具有明确的顺序。例如，可以使用有序回归(ordinal regression)来预测某个范围内的值。本章的下一节将讨论这个想法。

现在，讨论一些通常而言对于文本分类和文本挖掘都非常有用的一些特征，并且这些特征在情感分析中尤其需要。最简单的特征是字符 n 元，即作为一个单元处理的 n 个相邻字符的序列。这是表示文本数据的非常通用和健壮的方法，因为我们可以将这种方法用于

任何语言中。这种方式也可以应对拼写错误或识别错误；如果你拼写错了一个单词的一个字母，它仍然能够让你匹配到该单词，和它在文本中正确地出现一样。当然，这样的表示不会如单词那样好区分。

接下来是词 n 元，一个单词(word)序列而不是字符(character)序列。我们可以混合使用不同的 n 值。一元词通常对文本处理任务非常有效；这主要是因为单词是人类用于沟通的基本信息单位。然而，一元文法单词可能不足以满足情感分析的任务。例如，可能有一句话是："它不好"或者"它不如别的东西"。在这种情况下，如果只是提取特征"good"，这将表明这是一个正面情感的样本。显然，这并不准确。如果采用二元词($n=2$)的形式，那么二元组"not good"将出现，它使我们的表示更准确。因此，更长的 n 元通常更具区分性。然而，长的 n 元可能导致过拟合，因为它们创建了非常独特的特征，机器学习程序 394
把它与特定的类标签高度关联，而实际上并非如此。例如，如果一个 7 元文法的短语只出现在一个正面的训练文档中，那么这个 7 元短语将总是与正面情感相关联。然而实际上，这个 7 元短语只是出现在正例中，在其他地方没有出现过，因为它是如此罕见。

也可以考虑 n 元的词性标签。一个二元特征可以是一个形容词，后面跟着一个名词。也可以混合 n 元的单词和 n 元的词性标签。例如，单词 great 可能跟着一个名词，这也可以成为一个特征，一个混合特征，这可能对情感分析有用。

其次是词类别。这些类可以是句法，如像词性标签，或者通过在词典或本体中表示概念(如 Word-Net[Princeton Univensity 2010])来代表语义。或者，它们可以是被识别的命名实体(如人物或地点)。这些类别可作为附加特征用于丰富表示。我们也可以学习词聚类，因为已经在第 13 章中讨论了挖掘词之间的关联。一些聚合相关或者组合相关的聚类能够作为特征对基础词表示进行补充。

此外，还可以用频繁模式语法表示频繁词集。这些词不一定出现在彼此的旁边，但往往出现在同一上下文中。我们还可以获得词语可能会更频繁出现的位置的信息，这种模式提供了比词更具判别性的特征。它们可能比普通的 n 元语法模型泛化能力更好，因为它们是频繁的，这意味着它们可能在测试数据中出现，尽管它们可能仍然面临过拟合的问题，因为特征变得更加复杂。这是一般的问题，以分析树为基础的特征也面临这一问题，例如，频繁子树作为特征更具有区分性，但是它们也更可能导致过拟合。

一般来说，模式发现算法对于特征构造非常有用，因为它们允许搜索许多比单词更复杂的可能特征，而自然语言处理对于帮助我们得到可以丰富文本表示的复杂特征非常重要。

正如在第 15 章中提到的，特征设计极大地影响分类精度，并且可以说是任何机器学习应用中最重要的部分。如果你可以在设计特征时结合机器学习，错误分析和特定领域知识，将是最有效的。首先，要使用领域知识，就是对这个问题的专门理解。有了这个，我们可以设计一个具有许多可能特征的基本特征空间，使机器学习程序能够正常工作。机器 395
学习方法可以应用于选择最有效的特征或甚至构造新特征。这些特征可以进一步通过在这本书中讨论的评价技术人工进行错误分析。我们可以查看分类错误，并进一步分析哪些特征可以帮助我们从这些错误中恢复，或者哪些特征会导致过拟合。这样通过特征验证对特征集合进行进一步的修订。然后重复这些步骤，直到达到期望的精度。

总之，设计特征的一个主要挑战是在充分性和特异性之间取得最好的权衡。这种权衡是非常困难的。充分性意味着我们希望特征在许多文档上具有高覆盖率。在这个意义上，我们希望特征是频繁的。特异性要求这些特征具有可区分性，因此，特征往往不那么频

繁。显然，这会导致在频繁与不频繁的特征之间进行折中。特别是在情感分析的情况下，特征工程是一项关键任务。

18.2　有序回归

本节中将讨论将有序逻辑回归用于情感分析。典型的情感分类问题与评分预测有关，因为我们经常尝试在某个范围预测情感值，例如，正负之间还含有其他的值。将一个带有观点的文本文档 d 作为输入，我们希望生成 1～k 范围内的输出。由于它是一个离散的评分，这可以被视为一个分类问题(发现 k 类中哪个是正确的)。遗憾的是，这样的解决方案不会考虑类别的顺序及它们之间的依赖。直观上，可以将评分 2 与 1 区分开的特征可能与区分 k 与 k-1 的特征类似。例如，正面的词通常表示更高的评价。当将这些类别视为独立的来训练分类问题时，我们不会捕捉到这些信息。解决这个问题的一种方法是有序逻辑回归(ordinal logistic regression)。

先想想我们如何用逻辑回归处理二元情感(这是一个二分类问题)。假设我们只想区分正数与负数。预测因子(特征)表示为 X，我们可以根据对数概率比输出分数：

$$\log \frac{p(Y=1 \mid X)}{p(Y=0 \mid X)} = \log \frac{p(Y=1 \mid X)}{1-p(Y=1 \mid X)} = \beta_0 + \sum_{i=1}^{M} x_i \beta_i \tag{18-1}$$

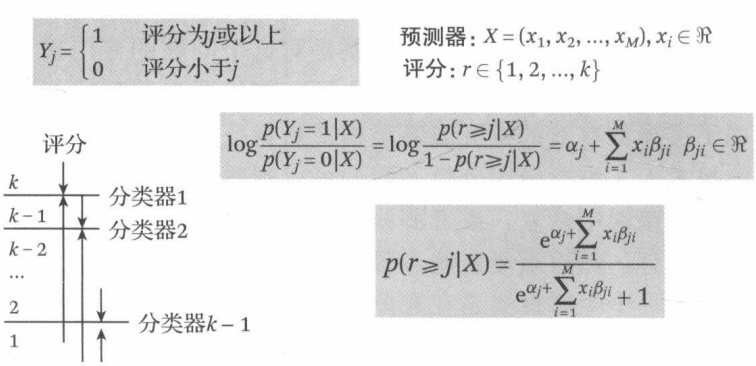
396

或条件概率

$$p(Y=1 \mid X) = \frac{\exp\left\{\beta_0 + \sum_{i=1}^{M} x_i \beta_i\right\}}{1 + \exp\left\{\beta_0 + \sum_{i=1}^{M} x_i \beta_i\right\}} \tag{18-2}$$

共有 M 个特征，每个特征值 x_i 都是一个实数。像往常一样，这些特征可以是文本文档的表示。X 是二元响应变量 0 或 1，其中 1 表示 X 为正，0 表示 X 为负。当然，这是一个标准的二分类问题，我们可以应用逻辑回归。你可以回顾第 10 章，在逻辑回归中，假设 $Y=1$ 是特征的线性函数的对数概率。这将使我们也可以将 $p(Y=1 \mid X)$ 写为特征的线性函数的变换形式。其中 β_i 是参数。这是对二分类逻辑回归的直接应用。

如果我们有多个类别或多个级别，我们将适当调整二元逻辑回归问题来解决这个多层次的评分预测，如图 18-5 所示。这个想法是我们可以引入多个二元分类器；在每种情况下，我们要求分类器预测评分是否等于 j 或大于 j。那么当 $Y_j=1$ 时，这意味着评分是 j 或高于 j。当它为 0 时，这意味着评分低于 j。如果想预测 1～k 的范围内的评分，首先有一个分类器来区分 k 与其他值。而后，我们需要另一个分类器来区分 $k-1$ 和其他值。最后，我们需要一个分类器来区分 2 和 1，总共给出了 $k-1$ 个分类器。



397

图 18-5　逻辑回归用于多层情感分析

通过这种修改，每个分类器需要一组不同的参数，从而产生了更多的参数。我们将通过索引 j 对逻辑回归分类器进行索引，该索引对应于评分级别。这是为了使符号与我们在有序逻辑回归中使用的更一致。所以我们现在有 $k-1$ 个常规逻辑回归分类器，每个都有自己的一组参数。通过这种方法，现在可以预测评分，如图 18-6 所示。

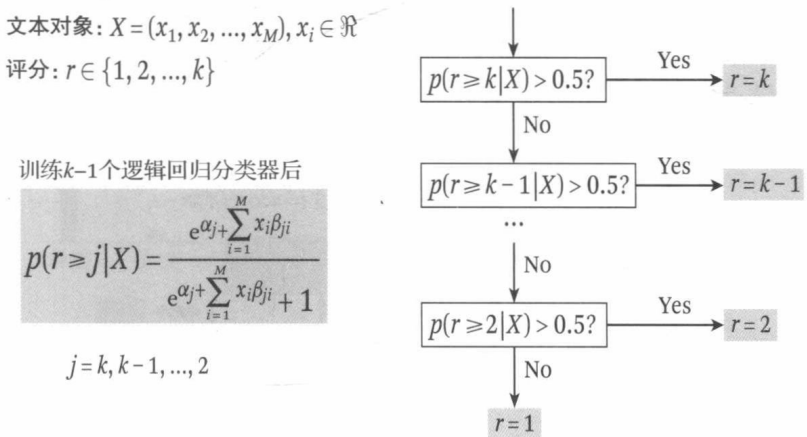


图 18-6　多层逻辑回归用于情感分析：预测评分

在分别训练了这 $k-1$ 个逻辑回归分类器之后，我们可以提交一个新的实例，然后依次调用分类器做出决定。首先，我们看看对应于评分 k 的分类器。该分类器将告诉我们这个对象是否应该具有评分 k。如果根据这个逻辑回归分类器的概率大于 0.5，则认为评分是 k。如果小于 0.5，我们需要调用下一个分类器，告诉我们它是否至少是 $k-1$。我们继续调用分类器，直到我们决定是否为 2 或 1 为止。

遗憾的是，这样的策略不是解决这个问题的最佳方式。具体来说，这种方法有两个问题。第一个问题是参数太多了。对于每个分类器，我们将 $M+1$ 个参数与 $k-1$ 个分类器组合在一起，因此参数的总数为 $(k-1)\cdot(M+1)$。当分类器有很多参数时，我们通常需要更多的训练数据来帮助我们确定这样一个复杂模型的最优参数。

第二个问题是这 $k-1$ 个分类器并不是真正独立的。我们知道，一般来说，正面情感的词会使这些分类器的评分更高，所以我们应该能够利用这一事实。这正是有序逻辑回归 398
的思想，是对 $k-1$ 个独立逻辑回归分类器的改进，如图 18-7 所示。

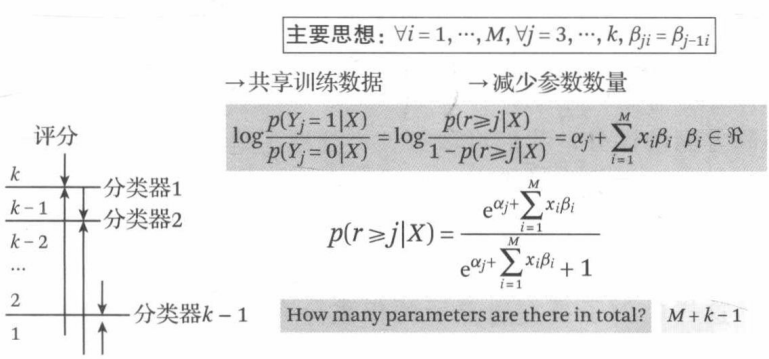


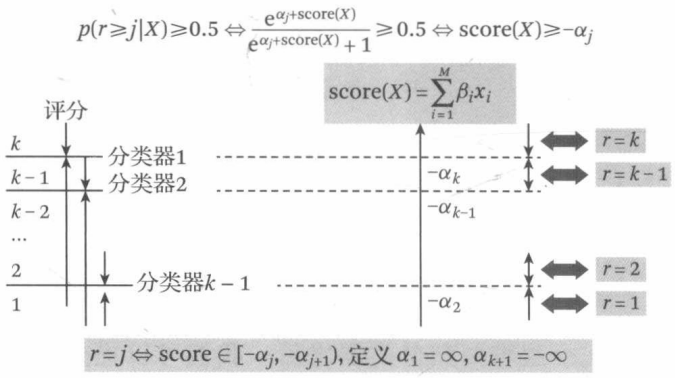
图 18-7　有序逻辑回归的原理

改进之处在于将 β 参数结合在一起，这意味着我们将假设所有 $k-1$ 个分类器的 β 值是相同的。这代表了我们的直觉，正面情感的词（一般来说）更有可能有更高的评价。事实上，这将使我们得到两个好处。一个是显著减少参数的数量。另一个是允许我们在所有分

类器之间共享训练数据，因为参数是相同的。实际上，我们有更多的数据来帮助我们选择好的 β 值。

所得到的公式看起来与以前看过的非常相似，现在的 β 参数只有一个对应于一个特征的索引；它不再具有对应于评分的其他索引。然而，每个分类器仍然具有不同的预测评分值。当然，这个值用来预测不同的评分水平。所以 α_j 是不同的，因为它取决于 j，而其余的参数(β_i)是相同的。我们现在有 $M+k-1$ 个参数。

通过这个想法来绑定所有的参数，最终用一个类似的方式做出决策，如图 18-8 所示。



399

图 18-8　有序逻辑回归的决策过程

更具体地说，预测因子概率是否在 0.5 以上的标准等同于对象的得分是否大于或等于 α_k。评分函数只是将所有特征与 β 值进行线性组合。这意味着现在我们可以简单地做一个评分决定，看看评分函数的值以及它属于哪个区间。在这种方法中，我们将通过使用特征和经过训练的参数值给对象评分。然后将该分数与一组经过训练的 α 值进行比较，以查看分数在哪个范围。然后，根据该范围，我们可以决定对象应该接受哪个评分。

18.3　潜在方面评分分析

本节将继续讨论观点挖掘和情感分析。特别地，我们将介绍潜在方面评分分析(Latent Aspect Rating Analysis，LARA)，允许对评论的整体评分进行详细的分析。

图 18-9 显示了两条酒店的评论。两个评论人都给了五颗星。如果你只是看整体评分，酒店位置或服务是否不错并不是很清楚。也不清楚为什么评论者喜欢这家酒店。我们想要做的是将这个总体评分分解为不同评分，如价值、房间、位置和服务。如果可以将整体评分分解为这些不同方面的评分，我们可以更加详细地了解评论者对酒店的观点。这也将使我们能够按价值或房间质量等不同维度对酒店进行排名。

使用这些知识可以更好地了解评论者是如何从自己的角度来看待这家酒店的。我们不仅要推断这些方面的评分，还要推测各个方面的权重。也就是说，一些评论者可能更关心价值而不是服务。这样的例子是左边显示的权重分布，你可以看到最重要的是价值。显

400 然，不同的用户将优先考虑在不同方面的评分。例如，想象一个有五星价值的酒店。尽管如此，它可能仍然非常昂贵。如果评论者真的关心酒店的价值，那么五星级评论很可能意味着有竞争力的价格。因此，为了准确地解读不同方面的评分，我们需要知道各个方面的权重。当这些不同的方面与每个用户的特定权重组合在一起时，我们可以对整体观点有更为详细的了解。



图 18-9　LARA 的动机

因此，该任务是将这些评论和它们的总体评分作为输入，产生方面评分和方面权重作为输出。这称为潜在方面评分分析(LARA)。

更具体地说，我们有一个关于某个主题的评论集合和它们的总体评分，我们希望生成三样东西。一是评论提及的主要方面。二是对各方面的评分，如价值和房间服务。三是每个评论者对不同方面给出的相对权重。这个任务有很多应用。例如，它可以用于基于观点的实体排名，或者生成一个方面的观点总结，还可以分析评论者的偏好，比较他们，或者比较他们对不同酒店的偏好。所有这一切都使个性化产品推荐变得可能。

与其他高级主题的情况一样，我们将不详细介绍该技术。相反，我们将介绍为此问题所提出的方法，如图 18-10 所示。首先，我们将讨论如何分两个阶段解决问题。之后，我们将提及可以在一个统一的模型中做到这一点。 401

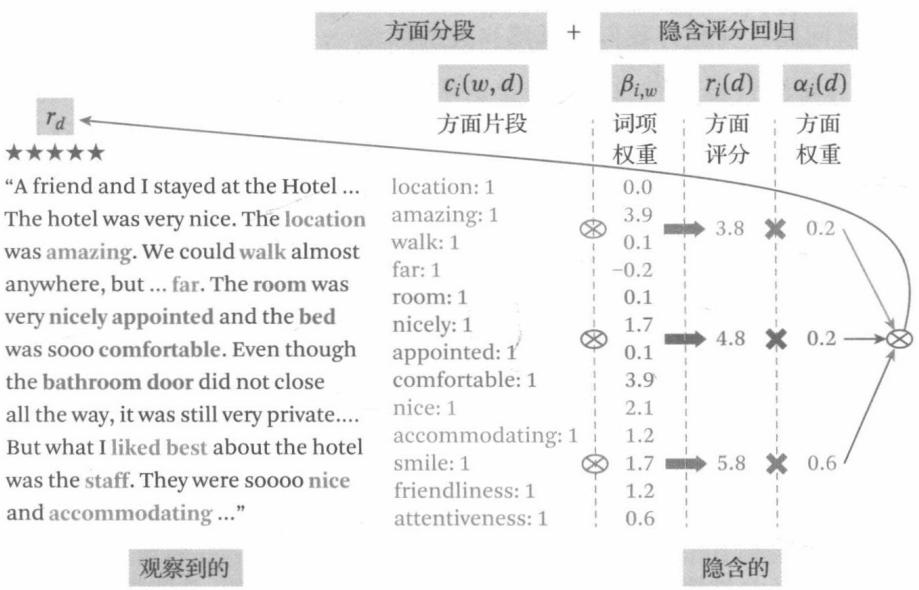


图 18-10　两步解决 LARA 问题的方法(Hongning Wang 授权)

作为输入，我们有评论及其总体得分。首先，我们切分出各个方面对应的片段。我们要挑选出哪些词说的是地点，哪些词说的房间条件，等等。特别地，我们将获得每个片段中所有单词的计数，由$c_i(w, d)$表示，其中 i 是特定的分段索引。片段切分可以通过使用

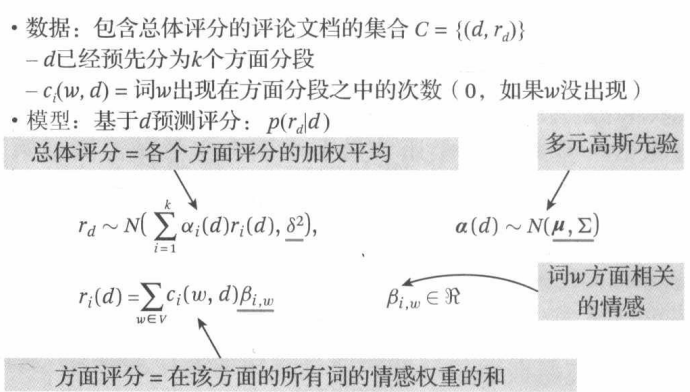
如位置、房间或价格之类的种子单词来检索获得具有典型方面标签的段落。从这些段落中可以用这些种子词进一步挖掘各个方面的相关词，使我们能够将文本分成讨论不同方面的部分。稍后将看到，我们也可以使用无监督模型进行分段。

在第二阶段的潜在评分回归中将使用这些单词及其在不同方面的频率来预测整体评分。这个预测发生在两个阶段。第一阶段将在每个方面使用这些词的权重来预测方面的评分。例如，如果在讨论位置时，你看到像 amazing 这样的单词被提及很多次，它将具有很高的权重(在图中，它的权重为 3.9)。这种高权重增加了位置的方面评分。在另一个词的
402 例子中，如 far，多次提到，其权重会降低。方面评分假设分数将是这些单词频率的加权组合，其中权重是单词的情感权重。当然，这些情感权重在不同方面可能会有所不同。对于每个方面 i，我们有一组词 w 的词项情感权重，表示为$\beta_{i,w}$。

在第二阶段，我们假设整体评分仅仅是这些方面评分的加权组合。我们假设我们有方面权重$\alpha_i(d)$，这些将被用来取得方面评分$r_i(d)$的加权平均值。这种方法假设总体评分仅仅是这些方面评分的加权平均值，这使我们能够基于可观察的单词频率来预测整体评分。

图 18-10 的左侧是所有观察到的信息，r_d(总体评分)和$c_i(w, d)$。在右侧是我们希望发现的所有潜在(隐含)信息。这是嵌入有趣的潜变量的生成模型的典型案例。然后，为给定观察词提供了整体评分的生成概率。我们可以调整这些参数值，以最大化给定文档的观察评分的条件概率。之前已经在其他模型(如 PLSA)中看到过这样的例子，在那些例子中预测文本数据中的主题。在这里，我们预测方面评分和其他参数。

更正式地，我们在这里建模的数据是一组具有整体评分的评论文档集合，如图 18-11 所示。每个评论文档表示为 d，总体评分由r_d表示。我们使用$c_i(w, d)$来表示方面片段 i 中的单词 w 的计数。该模型将基于 d 预测评分，因此我们的兴趣点在 $p(r_d|d)$的评分回归问题上。该模型设置如下。假设r_d遵循正态分布，均值是方面评分$r_i(d)$的加权平均值，方差为 δ^2。当然，这只是我们的假设。与往常一样，当我们做出这一假设时，我们有一个正式的方法来建模问题，并允许我们计算感兴趣的数量——在这种情况下，为方面评分和方面权重。



403 图 18-11　潜在评分回归

每个方面评分$r_i(d)$被假定为方面 i 中词语的情感权重之和。总体评分中各个方面的权重向量 α 是从另一个多元高斯分布 $\alpha(d) \sim \mathcal{N}(\mu, \Sigma)$得出的。这意味着当我们产生我们的总体评分时，将首先从这个多元高斯先验分布中抽样出一组 α 值。一旦我们得到这些 α 值，我们将计算方面评分的加权平均值作为正态分布的均值以产生整体评分r_d。请注意，β 由 i 和 w 两者进行索引。这为我们提供了一种建模同一个词的不同方面的方法，因为同一个词

对于一个方面可能有正面的情感，对另一个方面可能是负面的情感。

如何估计所有这些参数？我们将它们统称为 $\Lambda=(\beta_{i,w}, \mu, \Sigma, \delta^2)$。像往常一样，我们可以使用极大似然估计，得到参数在已知对应评论的条件下最大化观察到的评分，如图 18-12所示。

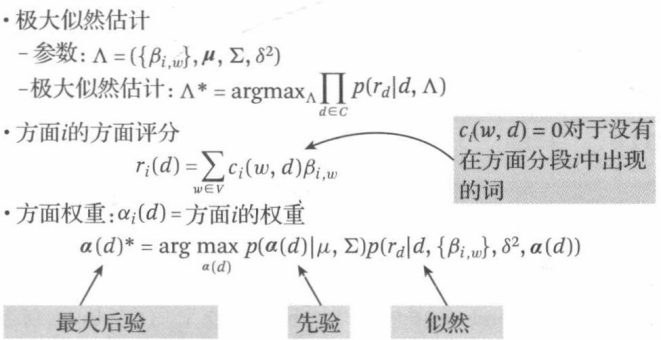


图 18-12　隐评分回归估计

一旦估计得到参数，我们可以通过对分段 i 中出现的所有单词进行计数，并将它们乘以$\beta_{i,w}$，对所有单词进行求和，从而轻松计算某个特定方面的方面评分。对于不出现的单词总和为零，所以我们可以简单地对词汇表中的所有单词求和。

404

为了计算$\alpha_i(d)$的值，我们必须使用最大后验。这意味着最大化 α 的先验(根据我们假设的多元高斯分布)和评分r_d的似然值的乘积：

$$\alpha(d)^*=\arg\max_{\alpha(d)} p(\alpha(d)|\mu, \Sigma)p(r_d|d, \beta_{i,w}, \delta^2, \alpha(d)) \tag{18-3}$$

似然评分是给定该 α 值和一些其他参数的情况下产生该观察到的总体评分的概率。有关这个模型的更多细节，我们推荐读者阅读 Wang 等[2010]。

早些时候，我们讨论了如何通过两个阶段解决 LARA 问题。首先，对不同方面进行分段，然后使用潜在回归模型来学习方面评分和权重。也可以构建一个统一的生成模型来解决这个问题。也就是说，不仅可以基于文本对整体评分的产生进行建模，还可以对文本本身的生成进行建模。一个自然的解决方案是使用主题模型。

给定一个实体，我们可以假定有一些由词分布(即主题)描述的方面。然后，使用主题模型来模拟评论文本的生成。假设评论文本中的单词是从这些分布中抽样得到，这与在 PLSA 中所假设的相同。然后，我们可以插入潜在回归模型，使用文本进一步预测整体评分。为了根据生成的文本预测总体评分，首先预测各个方面的评分，然后将它们与方面权重相结合，以预测整体评分。这给我们一个统一的生成模型，同时建模文本的生成和基于文本的整体评分。这里没有空间详细讨论这个模型，推荐读者进一步阅读 Wang 等[2011]的工作。

来看看通过使用这些生成模型在一些应用上的应用。首先，考虑一些具有相同总体评分的酒店的分解的评分。如果你只是看整体评分，这些酒店之间的差异并不大，但是通过将这些评分分解为方面评分，可以看到一些酒店的某些维度(如价值)的评分较高，而其他酒店则可能会在其他维度上评分更好(如位置)。这个细分可以在方面层面给我们详细的解释。

另一个应用是可以比较同一家酒店的不同评论。在更高层次上，整体评分可能看起来一样，但分解评分后，你可能会看到他们在不同维度上具有高分。这是因为该模型可以辨别不同评论者的观点分歧。这样详细的理解可以帮助我们了解评论人，并更好地纳入他们的反馈意见。

405

图 18-13 中显示了一些在 4 个方面(价值、房间、位置和卫生)中各个维度上具有较高的正向权重和负向权重的词。这说明我们也可以直接从数据中学习情感信息。这种词典是非常有用的，因为一般来说，一个词如“长”在不同的语境中可能会有不同的情感极性。如果我们看到“这台笔记本电脑的电池寿命很长”，那就是正面的。但是如果看到“笔记本电脑的重新启动时间很长”，那显然不是很好。即使对于相同产品(即笔记本电脑)的评论，词“长”的情感也很模糊。然而，使用这种词典，我们可以了解一个单词对于特定方面是正面的还是负面的。这样的词典可以直接用于标注其他关于酒店的评论或标注社交媒体中关于酒店的评论。

价值	房间	位置	卫生
resort 22.80	view 28.05	restaurant 24.47	clean 55.35
value 19.64	comfortable 23.15	walk 18.89	smell 14.38
excellent 19.54	modern 15.82	bus 14.32	linen 14.25
worth 19.20	quiet 15.37	beach 14.11	maintain 13.51
bad −24.09	carpet −9.88	wall −11.70	smelly −0.53
money −11.02	smell −8.83	bad −5.40	urine −0.43
terrible −10.01	dirty −7.85	road −2.90	filthy −0.42
overprice −9.06	stain −5.85	website −1.67	dingy −0.38

图 18-13　LARA 学习到的情感词典(基于 Wang 等[2010]的结果)

由于除了整体评分外，这几乎完全无监督，这使得我们可以从互联网上潜在的更大量的数据中学习，以创建一个主题特定的情感词典。回想一下，该模型可以推断评论者是否关心服务或价格。我们如何知道推测的权重是否正确？这对评价是一个非常困难的挑战。

图 18-14 显示了不同城市酒店的价格。这些是不同的评论者群体所喜爱的酒店的价格。这里展示了价值与其他方面的重要性的比率。例如，我们有价值与位置。在图中，“前 10”是指通过特定方法计算得到的具有最高比率的评论者。这意味着与其他方面相比，
406 前 10 个评论者倾向于重视价值。“后 10”是指在其他方面比价值权重更高的评论者；这些是关心另一个维度的人，不关心价值，至少与“前 10”组相比。

前10：最高价值/X比的评论者（强调价值）
后10：最低价值/X比的评论者（强调非价值方面）

城市	平均价格	分组	价值/位置	价值/房间	价值/服务
阿姆斯特丹	241.6	top-10	190.7	214.9	221.1
		bot-10	**270.8**	**333.9**	**236.2**
旧金山	261.3	top-10	214.5	249.0	225.3
		bot-10	**321.1**	**311.1**	**311.4**
佛罗伦萨	272.1	top-10	269.4	248.9	220.3
		bot-10	**298.9**	**293.4**	**292.6**

更高

图 18-14　抽样的结果显示推断的权重是有意义的(基于 Wang 等[2010]的结果)

这些比率是根据模型推断的权重计算的。我们可以看到前 10 评论者喜欢的酒店的平均价格确实比那些受到“后 10”青睐的酒店便宜得多。这提供了一些验证推断权重的间接方式。看这三个城市的平均价格，实际上可以看到“前 10”组的平均价格低于平均水平，而后半部分(关心服务或房间情况等)往往会有比平均价格水平高的酒店。

通过这些结果，我们可以构建许多有趣的应用。例如，直接应用是为每个方面生成一

个总体摘要，包括关于每个方面的正面的和负面的句子。这比只有整体评分和评论文本的原始评论有更多的信息。

图 18-15 显示了分析用户评分行为的一些有趣的结果。你所看到的是不同方面的不同维度的平均权重。在左边，你看到的是喜欢昂贵酒店的评论者的权重。他们给了昂贵酒店 5 星，在服务上给了很高的权重。这表明人们喜欢昂贵酒店，因为服务好，这并不奇怪。这是通过推断的权重验证模型的另一种方法。

	昂贵酒店		廉价酒店	
	5星	3星	5星	3星
Value	0.134	0.148	**0.171**	0.093
Room	0.098	0.162	0.126	0.121
Location	0.171	0.074	0.161	0.082
Cleanliness	0.081	**0.163**	0.116	**0.294**
Service	**0.251**	0.101	0.101	0.049

人们喜欢昂贵酒店因为服务更好

人们喜欢廉价酒店因为价值更好

图 18-15　评论者偏好分析(基于 Wang 等[2010]的结果)

右侧的 5 星级评分对应于喜欢廉价酒店的评论者。正如预期的那样，他们更加重视价值。你会发现当他们不喜欢廉价酒店时，他们倾向于给房间的卫生更高的权重。这表明，通过使用这个模型，我们可以推断一些即使你读了所有的评论也很难获得的信息。这是一种文本挖掘算法可以超越人类能力的事情，即发现数据中有趣的模式。我们可以通过比较不同消费群体在不同地点的意见来比较不同的酒店。当然，该模型是非常通用的，所以它可以应用于任何有整体评分的评论上。

最后，将该模型应用于实体的个性化排名或推荐，结果如图 18-16 所示。因为我们可以在不同的维度上推断评论者的权重，所以我们可以允许用户指出他们实际关心的内容。例如，我们在这里有一个查询，显示 90%的权重应该是价值，10%为其他。也就是说，这个用户只是关心获得便宜的酒店——强调价值维度。使用这个模型，我们可以找到权重与查询用户相似的评论者。然后，我们可以使用这些评论者推荐酒店；这就是我们所说的个性化或针对特定查询的推荐。非个性化建议显示在顶部，你可以看到顶部的结果通常比底部组具有更高的价格。这是因为在底部的评论者更关心价值。这表明通过做文本挖掘，我们可以更好地理解和服务用户。

查询：
0.9价值
0.1其他

非个性化

个性化
（查询特定）

	酒店	总体评分	价格	位置
方法2	Majestic Colonial	5.0	339	Punta Cana
	Agua Resort	5.0	753	Punta Cana
	Majestic Elegance	5.0	537	Punta Cana
	Grand Palladium	5.0	277	Punta Cana
	Iberostar	5.0	157	Punta Cana
方法1	Elan Hotel Modern	5.0	216	Los Angeles
	Marriott San Juan Resort	4.0	354	San Juan
	Punta Cana Club	5.0	409	Punta Cana
	Comfort Inn	5.0	155	Boston
	Hotel Commonwealth	4.5	313	Boston

图 18-16　个性化实体推荐(基于 Wang 等[2010]的结果)

总结我们的讨论，情感分析是一个有很多应用的重要主题。文本情感分析可以通过使用文本分类轻松完成，但是标准技术往往不够充分，因此我们需要具有丰富的特征表示。如果有两个以上类别，我们还需要考虑情感类别的顺序。这就是我们在介绍有序回归时讨论的。我们还表明，生成模型对于挖掘潜在用户偏好，特别是挖掘潜在评分回归的生成模

型是非常有效的。在这里，我们通过在模型中对词加权来嵌入一些有趣的偏好信息和情感信息。对于产品评论，观点持有者和观点目标明确，易于分析。当然，我们有很多实际的应用。来自新闻和社交媒体的观点挖掘也很重要，但是比分析评论数据更困难，主要是因为观点持有者和观点目标不明确。这个挑战需要更先进的自然语言处理技术。

进一步阅读本章涉及的主题，我们建议阅读一篇关于观点挖掘和情感分析的综述[Pang 和 Lee 2008]。

18.4　观点挖掘与情感分析的评价

在本章中，我们从分类(或回归)和主题分析的角度研究了观点挖掘和情感分析。因此，从这些角度的评价将类似于第 15 章和第 17 章中讨论的评价方法。

从分类的角度来看，我们可以使用标注为正或负的文档，或者按照本章所描述的具有数值等级的文档作为数据集。这就成了标准的机器学习测试设置，我们可以使用诸如交叉验证的技术来确定我们的方法的有效性。此外，特征选择可以显示哪些特征对于确定句子(例如)是正或负最有效果。基于有用的特征，我们可以调整算法或尝试微调特征集合。

408
~
409

从主题分析的角度来看，我们希望确保主题是一致的，并且对数据集中的文档有可信的覆盖和分布。我们提到语料库对数似然是一种测试模型拟合数据的程度的方式。虽然这个评价指标并不总是符合人类的判断[Chang 等 2009]，它确实是一个符合常理的对真实有用性的检验或替代。

此外，我们可以通过第 13 章讨论的方法来测试对现有系统添加观点挖掘和情感分析的有效性。也就是说，我们可以在情感分析前比较系统的性能，然后比较其加入情感分析后的性能。如果在与手头任务相关的某些评价指标下，系统的效果显著性地好，则说明情感分析的效果确有提升。

书目说明和延伸阅读

观点挖掘与情感分析得到广泛的研究。关于这个一般主题的两本专题书是《Opinion Mining and Sentiment Analysis》(Pang 和 Lee 2008)和《Sentiment Analysis and Opinion Mining》[Liu 2012]。主题模型的多个扩展已经被用于分析观点主题(例如，主题混合模型[Mei 等 2007a]、多粒度主题模型[Titov 和 McDonald 2008]以及情感方面统一模型[Jo 和 Oh 2011])。潜在方面评分分析主要包含在两篇 KDD 论文中[Wang 等 2010]，[Wang 等 2011]。

练习

18.1　在本章中，我们主要讨论了如何确定文本对象的总体情感。想象一下，我们已经将情感信息作为对象的一部分，而我们更有兴趣确定情感所针对的目标。使用本书中提到的 NLP 技术来头脑风暴一些想法。

18.2　MeTA 工具包有 LDA 主题模型算法的实现，该算法包含单词和主题分布的对称先验。修改 MeTA 以在主题词分布中包含非均匀分布先验，以便将先前的一些附加知识进行编码。例如，在情感分析任务中，创建两个对特定的“好”和“坏”主题分布具有高权重的先验分布。也就是说，以前的“好”的主题的先验应该对“优秀”的这个词赋权更高，“坏”主题应该给“破坏”这个词更高权重。

410

18.3　情感分析的简单版本称为词价态评分(word valence scoring)。有限的单词列表具有

正或负的价态分数。在确定一个句子的意思时，对该句子中的每个单词的价态分数进行累加；如果结果是正的，则情感是正面的。如果结果是负的，则情感是负面的。例如，一些可能的价态数，坏的＝－4.5，极好的＝6.3，可接受＝1.1。这种方法的潜在不足是什么？

18.4　根据标记为正或负的句子列表，我们如何自动创建情感词的价态分数？

18.5　除了情感分析外，本章讨论的技术可以应用于其他问题。给出一些也可以受益于本章讨论的一般方法的应用的名字。说明如何实现它们。

18.6　交叉验证是评价分类器的有用方法，如第 15 章所述。如何使用交叉验证来检测潜在的过拟合？

18.7　我们提到特征工程的主要挑战是优化冗余与特异性之间的权衡。你可以设计一个实验训练设置将这些变量考虑在内吗？

18.8　在 LARA 中，为什么认为权重的向量 α 是从多变量高斯分布中得出的？也就是说，为什么不使用一些其他的分布呢？

18.9　在 LARA 中，每个单词被分配一个方面，但这意味着一个句子可能被分配许多不同的方面。你可以描述一种调整方法，以确保每个句子只涵盖一个主题或少量主题？同样，我们能确保词序列都属于同一个方面吗？

18.10　不同于分类任务，通过文档对象的正面程度来进行排序可能是有益的(因此排名较低文档是非常负面的)。概述实现这一目标的几种方法。与分类相比有什么优点和缺点？

18.11　想象一下，你有一个未标注的产品评论数据集。如何设计基于此数据集的情感分类器，而不需要手动标记所有文档？

411
～
412

第 19 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

文本与结构化数据的联合分析

本章将讨论对文本和结构化数据进行联合分析的技术。这些技术不仅丰富了文本分析，而且对于涉及结构化数据和非结构化文本数据的许多预测问题来说都是必要的。鉴于方法的复杂性和内容长度的限制，我们只简单介绍一下主要思想，并展示一些样例来告诉大家这些技术具体可以支持哪些应用。本章中提到的一些方法的技术细节请参见本章结尾处提供的参考资料。

19.1 引言

在实际的大数据应用中，我们可以使用结构化数据和非结构化文本数据来帮助预测和决策支持。所以关键是要联合分析这两种数据，尤其是当目标是要去预测一些在数据中没有直接观测到的隐变量时。为了说明这些问题的设定和数据挖掘的全貌，我们在图 19-1 中展示这个数据挖掘循环(data mining loop)。

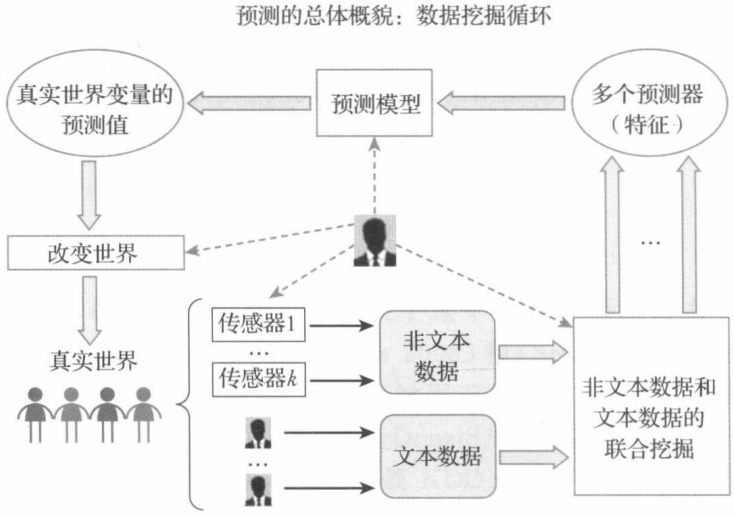


图 19-1 数据挖掘循环

在这个图中我们看到有多个传感器(包括人类传感器)以数据的形式报告我们在现实世界中看到的东西。这些数据包括非文本数据和文本数据。我们的目标是去看我们是否可以预测真实世界中对我们至关重要的一些变量的值。例如，我们可能对预测桥梁状况、天气、股票价格或总统选举结果感兴趣。对预测这些变量感兴趣是因为我们可能想要根据这些推断值行事，或者根据推断出的结果做出决策。

怎样才能从这些数据中获得这些预测值呢？首先要对数据进行挖掘和分析。一般来说，应该尽可能利用可以收集到的所有数据，而这其中对非文本和文本数据进行联合挖掘是至关重要的。通过对所有数据进行分析，我们可以对感兴趣的变量构建多个预测因子。我们把这些预测因子称为特征。它们可以被进一步结合在一起并放到同一个预测模型中来预测任何我们感兴趣的变量的值。预测结果可以让我们采取行动来改变世界。这是根据数

据进行预测的一般过程。

值得强调的是，尤其是因为文本数据的介入，人类在这一过程中实际上扮演着非常重要的角色。第一，人类用户将参与数据的挖掘。用户可以控制这些特征的生成，甚至可以手动创建特征。第二，人类可以帮助理解文本数据。因为文本数据是人类创造的，所以人类可以比机器更有效地使用和解释文本数据。然而当我们面临海量的文本数据的时候会面临挑战，因为人类不可能阅读并消化吸收所有文本信息。所以我们需要机器的帮助，这就是我们需要进行文本数据挖掘的原因。机器有时甚至可以看到人类无法看到的数据模式，即使人类有时间去阅读所有的数据。

413
~
414

接下来人类还必须参与建立、调整和测试预测模型。具体地，人类有关于预测问题的领域知识。我们可以把这些领域知识加入预测模型中。一旦有了这些变量的预测值，我们将会采取行动来改变世界或根据这些特定的值做出决策。

最后，有趣的是人类可以参与控制传感器来收集对预测*最有帮助的数据*。这就形成了一个数据挖掘循环。因为当我们调整传感器时，它们会收集到额外的、新的、更有用的数据，来提高预测结果。在这个循环中，人类将认识到需要收集哪些额外的数据。机器可以帮助人类识别下一步该收集什么数据。总的来说，我们想收集对学习最有用的数据。这种关于如何识别最有助于机器学习的数据点的研究通常被称为主动学习。主动学习是机器学习中一个重要的分支。

这就是一个循环，它从数据采集到数据分析，从数据挖掘到价值预测，从行动到改变世界；最后，我们观察发生了什么。然后我们可以决定收集哪些额外的数据并相应地调整传感器。对预测误差的分析有助于揭示需要获得哪些额外的数据以提高预测的准确性。这幅大图实际上很简略，可以作为在大数据上的许多重要应用的模型。

由于本书的重点是文本数据，因此考虑图 19-2 所示的循环的特定例子会很有帮助。这个特定例子的目的是使用文本数据推断真实世界中的其他变量的值，而这些变量可能与文本无关。这样的分析任务不同于主题挖掘这样的任务。主题挖掘任务的目的是直接描述文本的内容。在基于文本的预测中，我们的目标可以是推断关于世界的任何信息。当然，只有在目标变量的文本数据中存在线索的情况下基于文本的预测才是可能的。在许多情况下(例如股价预测)，非文本数据(历史股票价格)比文本数据能更有效地帮助预测，尽管文本数据通常能为预测任务提供额外的指示。

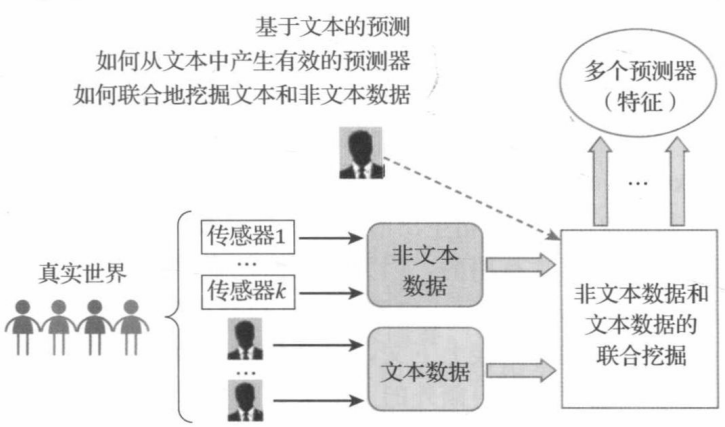


图 19-2　基于文本的预测示例图

有时文本数据包含比非文本数据更有用的指示，文本数据本身可能就足以用来做预

测。通常情况下，这些预测是关于人类行为或人类偏好或观点的。一般来说，人们常常把文本数据和非文本数据放在一起使用。

在所有基于文本的预测中，有两个重要的问题。第一个问题，什么样的特征(指示器)是对预测任务最有帮助的？第二个问题，我们如何从文本数据中产生有效的指示器？为了方便起见，我们将交替使用“特征”和“指示器”这两个词。第一个问题与具体的预测问题紧密相关，因此不可避免地是任务特定的。然而，也有一些通用的特征让我们可以以它们为着手点(如 n 元模型或主题)就像我们在前面的章节中讨论的那样。可以用有监督学习方法来学习那些最有效的特征。第二个问题在本书的前几章已经得到了一定程度的解决，因为我们已经介绍的许多技术可以用来从文本数据中获取特征。例如，主题挖掘基于主题的指示器或预测因子都非常有用，这些指示器或者预测因子被进一步放到预测模型中。这种基于主题的特征可以与基于词的特征混合来丰富特征表示。某一组词项或文档聚类的大小也可能是潜在的特征。一组具有聚合关系的术语与任何单个术语相比是更好的指示器，而且我们基于文本数据生成的情感标记也是一些预测问题的另一种有用的特征。

尚未讨论的是我们如何联合挖掘文本和非文本数据以发现有趣的知识，特别是一些无法单独从文本或者非文本数据中获得的知识。这种文本和非文本数据的互动是本章的主题。

415
~
416

联合分析文本和非文本数据有两方面的好处。首先，非文本数据可以丰富文本分析。具体而言，非文本数据通常可以提供文本数据挖掘的上下文，从而使我们能够基于不同的非文本数据(例如，基于时间或位置的文本划分)以不同的方式划分数据。这就开启了上下文文本挖掘的可能性或者在非文本数据定义的上下文中挖掘文本数据来发现上下文特定的知识(例如与特定非文本变量(如时间)相关联的主题)，或在不同上下文(如时间趋势)中的模式。其次，文本数据有助于解释从非文本数据中发现的模式。例如，如果从非文本数据中发现了一个高频模式，我们可以将与带模式的数据实例的关联文本从那些不带模式的数据实例关联文本中分离出来。然后，我们可以分析这两组文本数据之间的差异。这些差异可能与模式的含义相关，因此可以提供给我们关于如何解释这个模式的见解。如果只通过观察非文本数据，这个模式非常难解释。这种技术被称为模式注释，在[Mei 等 2006]有详细的讨论。

19.2　上下文文本挖掘

在这一节中，我们将讨论如何使用非文本数据作为上下文来丰富文本数据的主题分析。这种分析技术可以被认为是主题分析的延伸，以进一步揭示主题和任何相关上下文(例如时间或地点)的相关性。当主题代表观点时，我们也可以揭示与上下文相关的观点。上下文文本挖掘(contextual text mining)对基于文本的预测是非常有帮助的，因为它允许我们将非文本数据与文本数据结合起来使用，从而衍生出潜在的非常有效的复杂预测因子。

上下文文本挖掘通常是非常有用的，因为文本往往有丰富的上下文信息。首先，文本数据几乎总是有可用的元数据(如时间、位置、作者和数据源)，这可以被视为直接上下文信息。其次，文本数据也可能有间接上下文，这是指与元数据相关的附加数据。例如，通过文章的作者，我们可以进一步获得额外的上下文，如作者的社会网络、作者的年龄或作者的地理位置。这样的信息一般不直接与文本相关，但是通过这样的传播过程，我们可以把它们都连接起来。常常还会有和当前文章同源的其他文章，我们可以将这些同源的文章连接起来使它们相互关联。一般来说，任何相关数据都可视为上下文。

文本数据的上下文可以用来做什么？上下文可以用非常多有趣的方式来划分文本数据。它几乎可以让我们以任何我们想要的方式划分文本数据，因此可以让文本的对比分析在任何我们感兴趣的上下文维度上都得以实施。

417

作为一个具体的例子，我们将展示一些上下文变量是如何用图 19-3 中的许多不同的方式来划分科研论文的。图的横轴表示发表论文的不同会议，纵轴表示出版物的时间。我们可以把每一篇论文当作一个独立的个体来对待；在这种情况下，论文的编号可以作为上下文而且每一篇论文都有自己的上下文。我们可以把所有在 1998 年出版的论文作为一个组群来对待，并且把时间作为非文本变量来划分。这样的划分将使我们能够比较不同年份的主题。同样，我们还可以根据不同的会议或者期刊来对数据进行划分。我们可以把所有发表在 SIGIR 上的论文看成一个组群并把它们与那些发表在 KDD 或 ACL 上的论文进行比较。这种比较的可行性是由不同的会议上的非文本变量提供的。此外，我们还可以通过使用作者的附加上下文信息划分数据来获得美国的作者所写的论文。这样的数据上下文视角将使我们能够对比美国作者所写的论文和其他国家作者所写的论文。

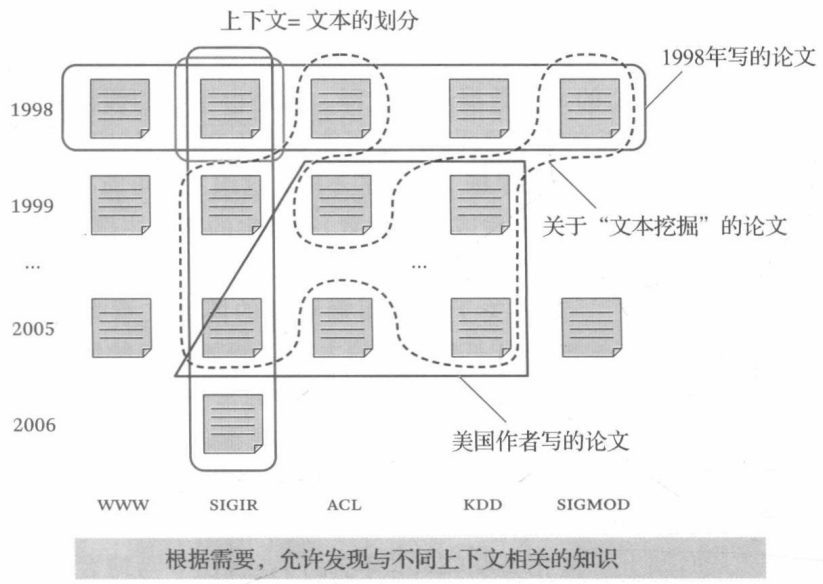


图 19-3　上下文产生文本数据的划分

418

有时我们还可以使用主题来划分数据，而不涉及非文本数据。例如，我们可以获得一系列关于“文本挖掘”的论文，并将它们与另一主题的论文进行比较。注意，这些划分可以相互交叉以生成更复杂的划分。因此，我们可以在非文本变量上形成约束，为划分后的文本数据构造有趣的上下文，以便于发现与不同上下文相关的知识。

非文本上下文变量的引入使文本数据的主题与许多潜在的不同上下文相关联，这样就产生了很多既有趣又有用的模式。例如，随着时间的变化来比较主题的变化，我们可以看到主题的发展趋势。比较不同上下文的主题也能揭示这两种上下文的差异。

有许多有趣的问题需要上下文文本挖掘来回答。例如，要回答“数据挖掘研究中什么主题在最近得到越来越多的关注?”我们需要利用时间作为上下文来分析文本。不同地区的人们对某一事件的反应有什么不同吗？要回答这样一个问题，位置信息就可以是上下文。两个研究人员共同的研究兴趣是什么？在这种情况下，作者就可以是上下文。美国的作者和其他国家的作者的论文中研究主题有什么不同吗？这里就必须包括作者及其从属关

系和地理位置这样的上下文信息。这就是一个我们不仅需要关注作者本身还需要关注与作者关联的额外信息的例子。

一个社交网络中的所有主题的意见与另一个社交网络相比有什么不同吗？在这种情况下，作者和主题的社交网络可以作为上下文。新闻数据中的主题是否与股价的突然变化有关呢？这样的问题可以通过使用诸如股票价格这样的时间序列作为上下文来解决。在2012年的总统竞选中哪些问题是重要的问题呢？这里的时间再次充当上下文。显然，上下文文本挖掘可以支持很多应用。

19.3　上下文概率潜在语义分析

在本节中，我们会简要介绍一种针对上下文文本挖掘的特定技术，一般称为**上下文概率潜在语义分析**(Contextual Probabilistic Latent Semantic Analysis，CPLSA)。CPLSA是PLSA的扩展，它在PLSA的基础上把上下文变量融入生成模型中，因此无论是主题的选
419 择和主题的词分布都依赖于和文本相关联的上下文。

记得在PLSA中文本数据是这样产生的：首先选择一个主题；然后从这个主题中产生一个词。主题在集合中的所有文档之间共享，但是选择概率(即主题的覆盖率)是特定于文档的。在CPLSA中，也采用类似的文本数据产生过程。但是由于我们假设有关于一个文档的上下文信息(时间、位置)，所以文档中的词的产生可能取决于文档的特定上下文。我们不是假设集合中只有一组公共主题，而是假设根据上下文的不同，这组主题可能有不同的变体。例如，我们可能对特定上下文(例如特定的时间段或特定位置)所产生的所有主题有特定的视图，因此我们会有多组可比较的主题，它们表示与不同上下文相关联的这些主题的不同视图。

在图19-4中，我们使用了一系列关于卡特丽娜飓风的博客文章来说明这个想法。在这样一个集合中，我们完全能够想象出一些潜在的主题，例如政府的回应、捐赠和新奥尔良的洪水泛滥。这些都被显示为不同的“主题”，它们中的每一个都由一个词分布来表示。除了这些主题本身，我们还展示了它们三个可能的不同视图：视图1与位置上下文(德克萨斯)相关而且它包含了图中的三个主题的很有德克萨斯风格的词分布，这可能反映出德克萨斯作者如何谈论这些主题，这大概会与伊利诺伊作者不同，这可以表示为一个不同的视图。同样，视图2与时间上下文相关联(2005年7月)，视图3与博客作者的职业相关联(社会学家)。

当在文档中产生单词时，主题的选择也可能会受到文档上下文的影响。例如，德克萨斯的作者可能会对某一个特定的主题的讨论热情多于其他的主题，而在其他地区的作者可能又会不同。同样，像时间和职业等这些上下文也可能暗示对某些主题的偏爱。在标准的PLSA中，我们假设每一个文档都有自己的偏好主题；我们可以把它视为CPLSA的一个特殊例子，其中每一个文档ID被视为这个文档的一个上下文。图19-4底部显示了不同上下文的不同主题选择偏好。

为了在文档d中生成一个单词，我们需要首先根据文档的上下文选择所有主题的特定视图。例如，所选择的视图可能是位置(德克萨斯)；在这种情况下，我们将使用德克萨斯风格的主题来生成单词。下一步是决定主题覆盖，这取决于上下文。我们可以选择与时间
420 有关的报道(2005年7月)。请注意，当以2005年7月作为时间上下文生成所有文档时，将使用相同的覆盖。这和PLSA中每个文档都有自己的覆盖偏好非常不同。在PLSA中每个文档的覆盖偏好是相互独立的。这里引入的依赖(上下文上的)使所有与特定上下文相关联的文本有助于主题覆盖的学习。

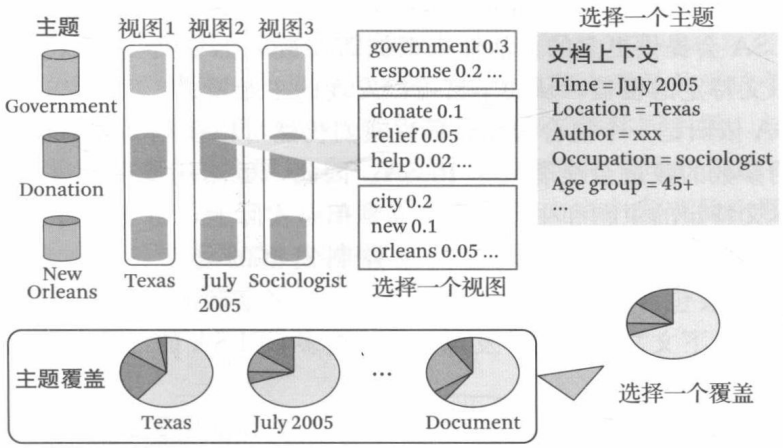


图 19-4　CPLSA 是一个基于上下文的生成模型(Qiaozhu Mei 授权)

一旦主题的视图和主题的覆盖分布被选定，其余的生成过程几乎和 PLSA 一模一样。也就是说，我们将使用主题覆盖和选择分布来取样一个主题，然后使用主题对应的词汇分布(即选定的主题视图)生成一个词。在这样一个生成过程中，每个词都可以使用不同的视图和不同的主题覆盖分布来生成，这取决于选择用来指导生成过程的上下文。注意，决定主题视图选择的上下文可以与被选择用来决定主题覆盖的上下文不同。图 19-5 展示了这个过程，其中我们发现文档中所有的词现在都是 CPLSA 生成的，它本质上是一个和不同的上下文相关联的许多组件模型混合的混合模型。

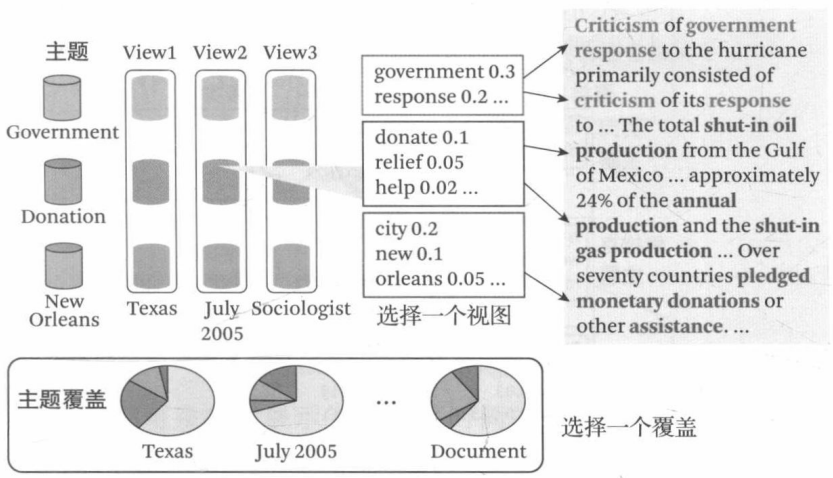


图 19-5　用 CPLSA 产生文档中的所有词(Qiaozhu Mei 授权)

在 CPLSA 中，主题的视图和主题的覆盖都依赖于上下文，使得它能够发现在不同上下文中的同一主题的不同变体(由于主题的视图对上下文的依赖)和在不同上下文中的不同的主题覆盖(由于主题覆盖对上下文的依赖)。正如前面提到的那样，这种上下文相关的主题模式对于回答各种问题非常有用。

标准的 PLSA 模型可以很容易地被视为 CPLSA 的特殊情况。我们通过利用全集作为上下文来使用所有主题的单一视图，并使用每个文档 ID 作为上下文来确定主题覆盖。结果我们可以发现使用 PLSA 只是主题的一个单一集合。这个单一集合描述了无法揭示覆盖在不同的上下文中的主题之间差异的文本数据中内容的特性。标准的 PLSA 只能显示每个

文档的主题范围，但不能发现与特定的上下文相关的主题覆盖。

相反，CPLSA 会提供更多的灵活性来根据需要嵌入上下文变量以发现主题的多个视图和主题的上下文特定的覆盖，从而丰富可以发现的主题模式。

因为 CPLSA 依旧是一个混合模型，所以我们仍然可以使用 EM 算法求解参数估计问题，但要估计的参数的数量会显著大于 PLSA。因此，虽然从理论上讲 CPLSA 可以使我们发现任何上下文特定的主题和主题覆盖分布。但是实际上，由于不可避免的数据稀疏性问题，我们必须限制上下文变量的空间来控制模型的复杂性。一旦进行参数估计，CPLSA 的参数自然会包含上下文变量，尤其包括许多主题的条件概率(给定的上下文)，这正是我们期望在上下文文本挖掘中发现。关于更多 CPLSA 的细节，请读者参考[Mei 和 Zhai 2006]的工作。

421
~
422

我们现在展示 CPLSA 的一些示例结果。首先在图 19-6 中我们分别展示通过比较两组关于伊拉克战争和阿富汗战争的新闻而得到的结果，包括共同主题和集合特定主题中的高概率词。在这种情况下，上下文是主题，而且它还会把新闻文章依据两场不同的战争分成两组，CPLSA 退化成一个简单的跨集合混合模型[Zhai 等 2004]。

我们有 30 篇关于伊拉克战争的文章和 26 篇关于阿富汗战争的文章。我们的目标是比较两组文章，以发现在这两个集合中有哪些共同主题，并了解在每一组中这些主题的变化。图 19-6 展示 CPLSA 发现的有意义的结果。第一列(聚类 1)展示了一个关于联合国在第一行上的一个非常有意义的共同主题。给定这两个集合的主题，这个结果在直观上是合理的。这样一个主题对那些了解这些文章背景的人来说并没有什么可惊奇的，但是这个结果表明 CPLSA 可以自动发现这样的主题。



图 19-6　用来比较两组新闻文章的样例结果(该结果通过使用 Zhai 等[2004]中提出的方法产生，它是 CPLSA 的一个特例)

423

然而更有趣的是在第一列的第二和第三行中的单词分布的两个单元格，就在那个关于联合国的单元格的下面。这两个单元格分别展示了联合国主题在伊拉克战争上的视图和它在阿富汗战争上的视图。结果表明，在伊拉克战争中，联合国更多地参与武器核查，而在阿富汗战争中，它更多地参与了对北方联盟的援助。这两个联合国主题的上下文特定视图说明了这两场战争在主题上的不同变化，并以上下文特定的方式揭示了对这些主题的更详细的理解。该表不仅可以立即用于了解这两组新闻文章中的主要主题和这些主题的变化，

而且为方便浏览文本集合上非常具体的主题提供切入点。例如，我们可以让每一个单元格都能被点击来让用户查看在新闻中相关讨论的细节。

第二列展示了关于伤亡的另一个共同的主题，鉴于这些文章是关于战争的，这一点就不会令人惊讶了。这也再次证实了 CPLSA 能够提取有意义的共同主题。与第一列的情况一样，第三列中的特定于集合的主题还进一步显示了在这两种不同上下文中主题的变化。

在图 19-7 中，我们展示了关于卡特丽娜飓风的博客文章中发现的主题随时间变化的趋势。x 轴是时间，y 轴是主题覆盖。这些图直接由 CPLSA 的参数产生，在这个 CPLSA 中我们使用时间和位置作为上下文。

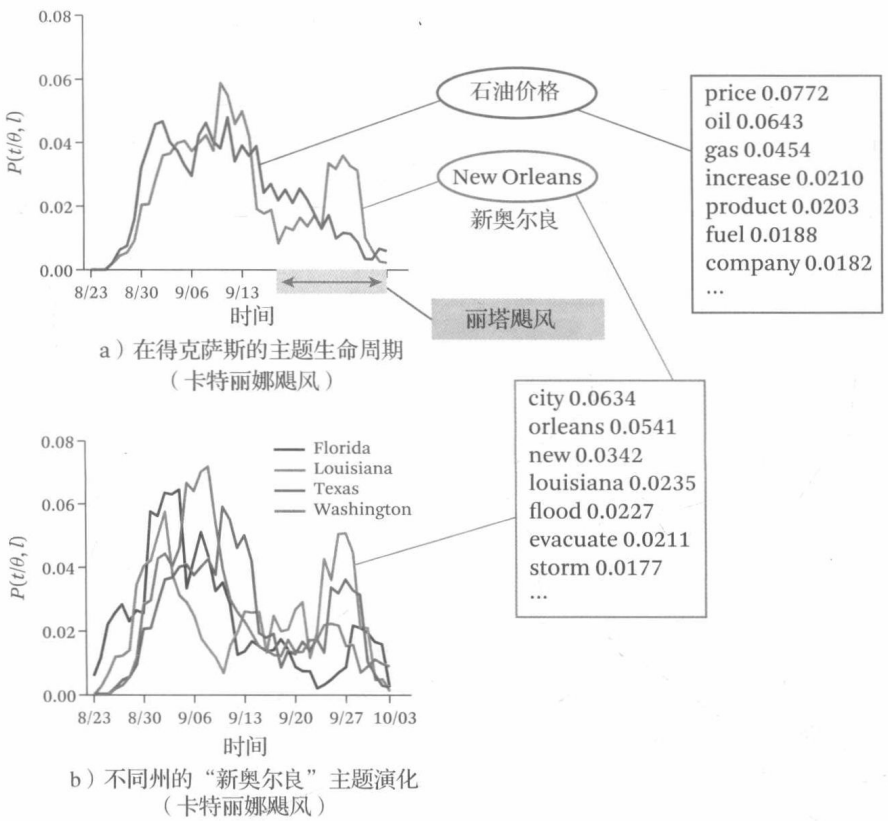


图 19-7　从卡特丽娜飓风的博客文章中发现的主题随时间变化的趋势(基于 Mei 等[2006a]中的结果)

在图 19-7 中，我们可视化地展示了主题随时间的变化趋势。上面的图基于一个主题在给定时间显示了两个主题随时间变化的趋势。一个是石油价格，另一个是新奥尔良市的大洪水。这个图是基于特定的时间段内的一个主题的条件概率，这是 CPLSA 的参数之一，用来捕捉时间依赖的主题覆盖。在这里我们看到在初期这两条曲线彼此紧随。有趣的是，随后新奥尔良的主题还会被提起但是石油价格的主题就不太被提及了。这正是另一场飓风(丽塔飓风)袭击该地区的时候，这显然引发了更多关于城市洪水的讨论，而不是石油价格。

下面的图显示了不同的地方(美国的不同州)的博客文章作者对新奥尔良大洪水的主题覆盖。我们看到，这个主题最初是在受害地区(如路易斯安那)的作者中大量覆盖，但是这个主题后来被得克萨斯的作者讨论起来，原因可能是从路易斯安那到得克萨斯的人口流动。那么这些主题趋势不仅有助于揭示主题及其随时间变化的动态，而且还可以对不同上下文中的主题进行比较分析，以帮助发现有趣的模式和与这些模式相关联的可能的有趣

事件。

在图 19-8 中，我们在同一组关于卡特丽娜飓风的博客文章中展示了关于政府响应主题的覆盖的时空模式。这些视图显示了该事件在美国不同州的不同时间范围内的分布情况。我们看到最初的报道主要集中在南部的受害州，但随着时间的推移，这个主题逐渐蔓延到其他地区。在第四周(右下角显示)，覆盖分布模式与第一周非常相似(左上角显示)。这是因为丽塔飓风差不多在那个时间袭击了该地区。这些结果表明，CPLSA 可以利用位置和时间作为上下文去揭示文本数据中的有趣主题模式。请注意 CPLSA 模型是通用模型，所以它可以很容易地被应用于其他类型的文本数据中去揭示相似的时空模式或主题模式。

424
~
425



a）第一周：08/23-08/29



b）第二周：08/30-09/05



c）第三周：09/06-09/12



d）第四周：09/13-09/19

主题1 政府响应
bush 0.0716374 president 0.0610942 federal 0.0514114 govern 0.0476977 fema 0.0474692 administrate 0.0233903 response 0.0208351 brown 0.0199573 blame 0.0170033 governor 0.0142153



e）第五周：09/20-09/26

图 19-8　政府响应的主题覆盖时空趋势(基于 Mei 等[2006a]中的结果)

在图 19-9 中，我们展示了 CPLSA 的另一个应用，用它来分析一个事件的影响。这个应用的基本思想是比较事件前后文本中所涵盖的主题的视图，以便揭示任何差异。这种差

426

异可以被认为是与事件的影响有关。这里显示的结果都是从信息检索领域的研究论文中发现的主题，特别是 SIGIR 上的论文。我们所关注的主题是关于检索模型(在左侧显示)。我们的目标是分析两个事件的影响。一个事件是 1992 年左右推出的文本检索会议(TREC)，它是由美国政府资助的一个主要的年度评价工作，它被认为对信息检索研究的主题产生了巨大的影响。另一事件是由 Ponte 和 Croft 共同发表的一篇开创性论文[Ponte 和 Croft 1998]，其中介绍了信息检索中的语言建模方法。该论文也被认为对信息检索研究产生了重大影响。要了解这两个事件的影响，我们可以把事件发生前和发生后的时间段作为上下文来使用 CPLSA 做分析。

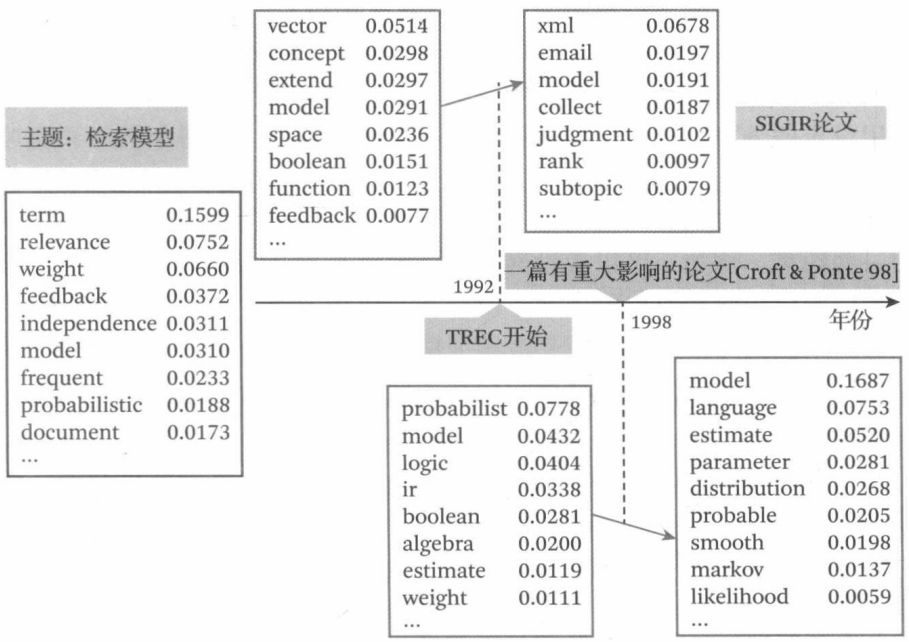


图 19-9　利用 CPLSA 进行事件影响分析(Qiaozhu Mei 授权)

在顶部的结果显示在 TREC 之前，检索模型的研究主要是集中在向量空间模型和布尔模型上，但在 TREC 之后，检索模型的研究明显在探索多种应用任务(例如，XML 检索、邮件检索和副主题检索，这些都是多年来在 TREC 上介绍的任务)。底部的结果表明，在语言建模论文发表于 1998 年之前，对检索模型的研究主要集中在概率、逻辑和布尔模型上，但 1998 年之后，人们对语言建模方法和参数估计有了明确的关注，这是 IR 语言模型研究中不可分割的一部分。因此，这些结果有助于揭示事件在文本数据中反映的影响。

427

19.4　以社交网络作为上下文的主题分析

在这一部分中，我们将讨论如何使用社交网络作为上下文来挖掘文本数据。一篇文本文章的上下文有时可以形成一个网络；研究文章的作者可能形成协作网络。同样，社交媒体内容的作者可能通过 Twitter 或 Facebook 这样的社交网络平台形成社交网络。例如，在 Twitter 中，人们可能互相关注，而在 Facebook 中，人们可能是朋友。这样的网络上下文可以间接地连接一个网络中的作者们写的内容。

同样，与文本相关联的位置也可以连接成地理网络，这又在不直接连接的文本数据之间添加了间接连接。

一般来说，如果我们能够识别元数据之间的一些关系，那么我们就可以想象文本数据的元数据很容易形成一个网络。当我们有可用的网络上下文时，就有机会来联合分析文本及其相关的网络上下文。这种联合分析的好处如下。首先，我们可以使用网络对文本主题施加一些限制。例如，有理由假设在协作网络中相互连接的作者倾向于写类似的主题。这种启发式方法可以用来指导我们分析主题。其次，文本也可以帮助描述每个子网的相关内容。例如，在两个子网表达的意见之间的差异可以通过做这种类型的联合分析来揭示。

我们现在介绍一种用网络作为上下文的特定文本分析技术，被称为网络监督主题模型。这样一个模型的总体思路如图 19-10 所示。首先，我们把任何生成模型(例如，PLSA)看成是定义一个优化问题，其中变量就是参数(在图中由 Λ 表示)，目标函数是似然函数。从文本数据挖掘的角度，估计的参数值 Λ^* 可以作为基于模型的挖掘算法的输出。有了这个视图，我们就可以使用文本数据的任何上下文信息在参数上加上约束或偏好来融合领域知识或者由用户或应用设定的偏好。这将导致对原始优化问题的修改，使得似然不再是唯一的优化
428 目标。

按照这种思路，对文本和相关网络上下文进行联合分析的主要思想是使用网络来对模型参数施加一些约束。例如，可以假设网络相邻的节点有相似的主题覆盖。事实上，在许多情况下，他们倾向于涵盖类似的主题；两个相互协作的作者倾向于在类似的主题上发表论文。这样的约束或偏好本质上试图平滑图或网络上的主题分布，从而使相邻节点具有非常相似的主题分布。这意味着我们期望它们在主题上共享一个公共分布，或者只在主题覆盖分布上有细微的变化。

如图 19-10 所示，我们在似然目标函数上添加了一个基于网络的正则项。我们不是去优化的概率 $p(\text{TextData}|\Lambda)$，而是去优化的另一个函数 f，它融合了似然函数 $p(\text{TextData}|\Lambda)$与一个正则项 $r(\Lambda, \text{Network})$，该正则项定义在任何源自网络上下文的偏好。当优化新的目标函数 f 时，我们将在最大似然值和满足正则化约束或偏好的参数值之间寻求折中。因此，如果我们将新的优化问题视为参数值的贝叶斯推断，即使我们没有任何明确定义的参数先验分布，我们也可以把网络上下文的影响看成是对模型参数强加的
429 一些先验。

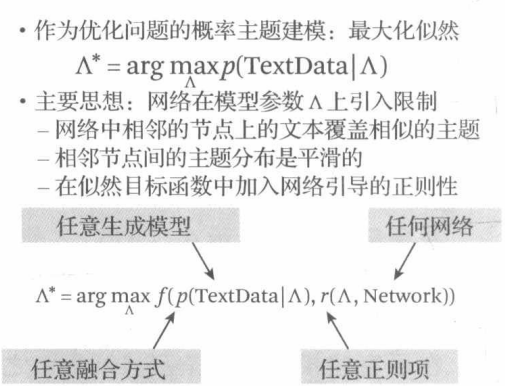


图 19-10　正则化似然函数的一般思路

值得注意的是，正则化似然函数这种思想是相当普遍的；事实上，概率模型可以是任何面向文本的生成模型(如语言模型)，网络也可以是连接我们希望去分析的文本对象的任何网络或图。正则项也可以是任何可用于捕捉不同的启发式的想法来适合特定的应用的形式；它甚至可以是多个正则项的组合。最后，函数 f 也可以变化，允许各种不同的方式来融合似然函数和正则项。另一种变体是指定基于网络上下文必须满足的单独约束，从而形成一个约束优化问题。

虽然这个想法是相当普遍的，在实践中，面临的挑战在于如何用具体的正则项来实现这样一个总体思路，使优化问题依然易于求解。下面我们介绍一个被称为 NetPLSA 的具体实例(如图 19-11 所示)，它是 PLSA 的扩展，在 PLSA 的基础上融入网络上下文。网络上下文的融入是通过实现启发式想法来实现的，即网络上的邻居节点必须有相似的主题分布。

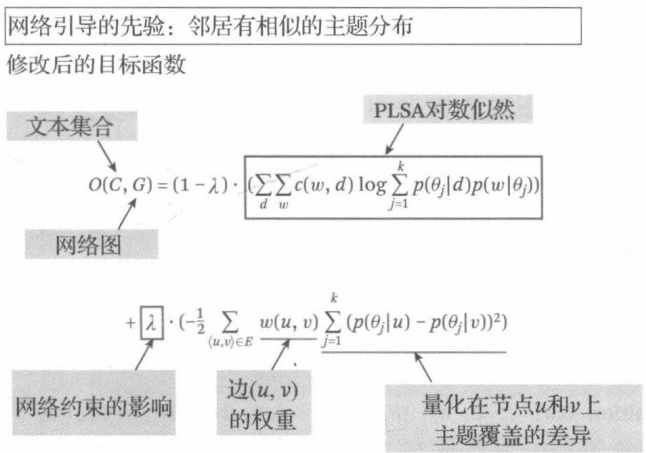


图 19-11　NetPLSA 模型

如图 19-11 所示，修改后的新的目标函数是一个标准的 PLSA 的似然函数和正则项的加权和，$\lambda\in[0,1]$来控制正则项的权重。显然，如果 $\lambda=0$，模型就退化为标准的 PLSA 模型。在正则项中，我们看到的主要约束是定义在两相邻节点 u 和 v 上主题选择概率差异的平方损失：$\sum_{j=1}^{k}(p(\theta_j|u)-p(\theta_j|v))^2$，它强烈地倾向于给 $p(\theta_j|u)$和 $p(\theta_j|v)$相似的值。在这个正则项之前，我们看到一个权重 $w(u,v)$，它基于我们的网络上下文，其中网络中的边上都被定义了权重。这个权重说明两个节点连接越多，去确保两个节点间有相似的主题就越重要。如果在边上不加权重，如果 u 和 v 之间存在边，我们可以设定 $w(u,v)=1$，如果无边就设定 $w(u,v)=0$，基本上只保留图上边的正则。注意，在正则项前有一个负号是因为我们想最大化似然部分同时最小化正则项定义损失。

430

像这种修改过的优化问题仍然可以使用 EM 算法的一个变种来解决，称为通用 EM。在 M 步，该算法并不试图去寻找辅助函数的最大值，反而只是去寻找能够增加辅助函数值的新的参数值。因此也保证了似然函数的增加，因为辅助函数是原始函数的下界(参见 17.3.5 节关于 EM 算法的更多解释)。整个算法仍然是一个爬山算法，能够保证收敛到局部最大值。

在图 19-12 中，我们展示了从文献数据库 DBLP 上使用标准的 PLSA 发现的 4 个主要的主题。DBLP 是由信息检索、数据挖掘、机器学习、万维网这 4 个研究社群的学术论文的标题构成。该数据集是由这些研究社群的论文汇聚而成，我们的目标是看 NetPLSA 是否比标准模型能更有效地学习和社群一致的主题。图 19-12 中展示的结果表明 PLSA 无法按我们的直觉产生 4 个社群。原因是它们都是混合在一起的而且有很多单词是这些社群之间共享的，数据中的共现统计不足以将它们区分开。

相比之下，图 19-13 所展示的 NetPLSA 的结果更加合理，其中的 4 个主题和我们要从数据集中发现的 4 个社群很好地对应。事实上，很容易用表中所示的 4 个社群来标记它们。NetPLSA 之所以能很好地区分这些社群发现更多有意义的主题，是由于网络上下文的影响。由于我们的网络是作者的协作网络，当我们强行让网络中连接的两个节点具有相似主题时，该模型将进一步调整所发现的主题，以便更好地反映参与同一协作网络中的作者所讨论的主题。因此，主题会更连贯，也更符合社群(以协作子网为代表)。这些结果也可用于表征每个协作子网的相关内容。

通过对网络上下文文本挖掘采用更一般的视角，我们可以认为文本数据处在一个丰富的信息网络环境中。也就是说，我们可以将所有相关数据连接成一个大网络，并将文本数

据与网络中的各种结构相关联。例如，文本数据可以跟网络中的节点关联；这种情况下可以用 NetPLSA 进行分析。然而，文本数据也可以与网络中的边相关联，甚至子网还有助于发现主题或实施比较分析。

主题1		主题2		主题3		主题4	
term	0.02	peer	0.02	visual	0.02	interface	0.02
question	0.02	patterns	0.01	analog	0.02	towards	0.02
protein	0.01	mining	0.01	neurons	0.02	browsubg	0.02
training	0.01	clusters	0.01	vlsi	0.01	xml	0.01
weighting	0.01	stream	0.01	motion	0.01	generation	0.01
multiple	0.01	frequent	0.01	chip	0.01	design	0.01
recognition	0.01	e	0.01	natural	0.01	engine	0.01
relations	0.01	page	0.01	cortex	0.01	service	0.01
library	0.01	gene	0.01	spike	0.01	social	0.01

图 19-12　PLSA 的采样结果(基于 Mei 等[2008]的结果)

信息检索		数据挖掘		机器学习		万维网	
retrieval	0.13	mining	0.11	neural	0.06	web	0.05
information	0.05	data	0.06	learning	0.02	services	0.03
document	0.03	discovery	0.03	networks	0.02	semantic	0.03
query	0.03	databases	0.02	recognition	0.02	services	0,03
text	0.03	rules	0.02	analog	0.01	peer	0.02
search	0.03	association	0.02	vlsi	0.01	ontologies	0.02
evaluation	0.02	patterns	0.02	neurons	0.01	rdf	0.02
user	0.02	frequent	0.01	gaussian	0.01	management	0.01
relevance	0.02	streams	0.01	network	0.01	onotology	0.01

431
~
432

图 19-13　NetPLSA 的采样结果(对比于 PLSA)(基于 Mei 等[2008]的结果)

19.5　以时间序列作为上下文的主题分析

在许多应用中，我们可能对挖掘文本数据以理解真实世界中发生的事件感兴趣。作为一个特殊的例子，我们可能对使用文本挖掘手段来理解一个时间序列感兴趣。例如，我们可能观察到某一特定时期股票市场价格突然下跌，并想看看是否有类似的新闻等文本数据可能有助于解释发生了什么。如果股票的崩溃时间与新闻流中突然出现某个新闻主题的时间相关，那么主题与股市崩盘之间可能存在着潜在的联系。同样，人们可能也有兴趣了解新闻流中报道的什么主题与总统选举有关，因此有兴趣在新闻流中找到与总统预测市场波动有关的主题(衡量人们对每一位总统候选人的看法)。

所有这些案例都是对文本和时间序列进行联合分析来发现因果主题这个一般性问题的特殊情况。在这里，我们用非严格的因果概念来指代任何和时间序列相关的主题，当然也有可能与时间序列存在因果。图 19-14 展示了这个分析任务。

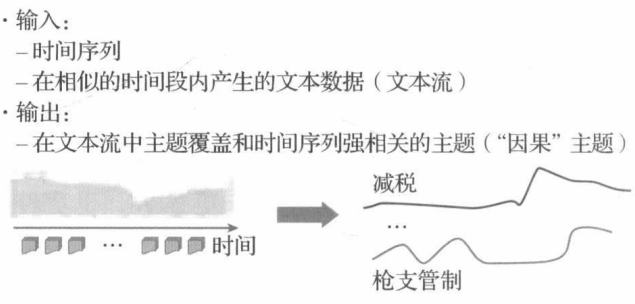


图 19-14　因果主题挖掘任务

输入包括一个时间序列加上在同一时间段内生成的文本数据，也称为伴随文本流。时 433
间序列可以看作是分析文本数据的上下文。我们要生成的输出是文本流中的覆盖与时间序列有强相关性的主题。也就是说，每当在文本流中频繁提到该主题时，时间序列变量往往具有更高(或更低)的值。

我们称这些主题为因果主题，因为它们或许可以解释时间序列波动的原因，并为人类进一步分析主题来更好地理解时间序列提供了见解。它们也可以用于预测时间序列。

直观地说，它的输出类似于我们使用主题模型生成的结果，但有着重要的区别。在常规主题建模中，我们的目标是发现最能解释文本数据内容的主题，但在我们发现因果主题的过程中，要发现的主题不仅语义上合理连贯(像在常规的主题建模中那样)，而且还应与外部的时间序列相关联。

为了解决这个问题，一个自然的想法是在文本流上使用像 CPLSA 这样的模型以便于发现大量随时间变化的主题及其覆盖。这将使我们能够为每一个主题获取一个时间序列，这代表文本中的主题覆盖，例如图 19-7 所示的时间趋势。然后，我们可以从这个与外部时间序列强相关的集合中选择主题。

然而，这种方法不是最优的，因为主题的内容完全是基于文本数据(例如，最大化似然函数)而没有考虑时间序列。事实上，所发现的主题往往是能很好解释文本数据的主要主题(它们应该如此)，但它们并不一定与时间序列相关。即使我们从中选择最好的，最相关的主题可能仍然有很低的相关性，因此从发现因果主题的角度来看不是很有用。

一种改进方法是运用时间序列上下文，不仅选择与时间序列相关度最高的主题，而且还影响主题的内容。一种方法称为迭代因果主题建模，如图 19-15 所示。

这种方法的思想是利用时间序列对主题模型发现的主题进行迭代调整来产生先验。具体地说，如图 19-15 所示，我们首先将文本流作为输入，并应用普通的主题建模方法来生成许多主题(这里展示了 4 个)。接着我们使用外部的时间序列来评估哪个主题是和 434
时间序列更因果相关的。评估的时候我们使用像格兰杰因果检验这样的因果度量方法。例如，在这个图中，主题 1 和主题 4 比主题 2 和主题 3 更相关。我们前面讨论过的简单方法就停在了这里并将主题 1 和主题 4 选为潜在的因果主题。然而，我们在这里进一步改进，通过缩放到词级别，来进一步确定与时间序列最密切相关的词。具体地说，我们可以对每个主题(有最高概率的词)查看排名靠前的所有词，并计算每个词与时间序列的相关性。

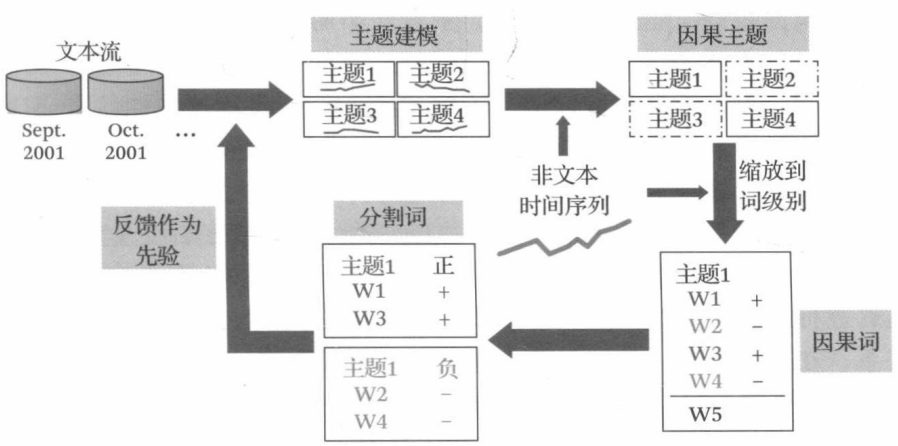


图 19-15　因果主题挖掘算法(调整自 Kim 等[2013])

这将使我们进一步把这些词分成三组：强正相关词；强负相关词；弱相关词。前两组的词可以被用来作为种子去定义两个新的子主题，被期望分别与时间序列正相关或负相关。图中显示一个对主题 1 的可能的分割，把它分割成两个更相关的子主题：一个用 w_1 和 w_3(正)，一个用 w_2 和 w_4(负)。然而，这两个子主题在语义上可能是不连贯的。为了提高连贯性，算法不会把它们直接作为主题，而是把它们作为先验加到主题模型中，使得主题模型能够更好地发现匹配那两个子主题的主题。因此，我们可以预期主题模型在下一次迭代中发现的主题比前次迭代中发现的原始主题与时间序列更相关。一旦我们发现了新一轮的主题，我们可以重复这一过程来分析相关主题中的单词并生成另一组种子主题，然
435 后再作为先验加到主题模型中。

整个过程被看作是优化因果关系和连贯性的一种启发式方法，这正是我们发现因果主题的目标。应用主题模型时，我们确保在发现的主题上的语义连贯性，但当我们把主题切分成正向和负向的子主题时，我们要提高它们和时间序列的相关性。如图 19-16 所示，本质上是交替地改进凝聚性和相关性(因果关系)。

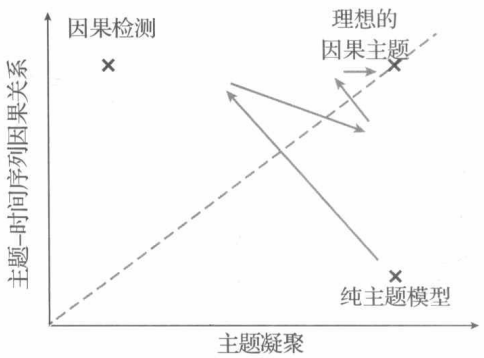


图 19-16　交替的优化连贯性和因果性/相关性(Hyun Duk Kim 授权)

在这里，我们看到纯主题模型将非常擅长最大限度地提高主题连贯性，因此在 x 轴上得分很高，这意味着所发现的主题都是合理的。如果我们只使用因果关系检验或相关性度量，那么我们将生成一组与时间序列紧密相关的词，因此在 y 轴(因果关系)上得分很高，但它们在语义上并不一定连贯。我们的目标是获得一个在连贯性和相关性上得分都很高的因果主题。上面讨论的方法可以看作是最大化 x 轴和 y 轴上得分的另一种方式。当我们使用主题模型时，我们最大化了连贯性，然而当我们把主题模型词分解成与时间序列紧密相关的词集时，我们将选择与时间序列最密切相关的词。因此，我们实际上是将模型推回到因果维度，使其在因果关系上得分更高。当我们把选定的词作为先验来指导主题模型的主题发现时，我们又反过来优化的连贯性。这种迭代过程最终预计会在语义连贯性和时间序列强相关性之间
436 折中选择。

这种通用方法依赖于两个特定的技术组件：主题模型和因果关系度量。前者在本书前面已经介绍过，因此我们简要地讨论后者。测量两个时间序列的因果关系有很多种方法。最简单的度量是皮尔森相关性。皮尔森相关性是测量两个变量相关关系的常用方法之一。它会给我们一个在[−1，+1]范围内的值，输出值的符号表示相关性的方向(我们将利用它来量化因果关系的影响)。我们也可以测量相关性值的显著性。如果直接使用，最基本的皮尔森相关性将具有零延迟，因为它比较的是同一时间戳上的值。然而，我们可以通过改变输入时间序列中的一个变量来计算滞后的相关性，并在改变后度量皮尔森相关性。

在时序数据上做因果关系检验的一种较为常用的方法是格兰杰检验。格兰杰检验用不同的时间延迟进行统计显著性检验，用自回归法来检验一个时间序列是否与另一个时间序列有因果关系。在形式上让 y_t 和 x_t 是两个待检测的时间序列，其中我们希望查看 x_t 是否与 y_t 具有最大 p 时间延迟的格兰杰因果关系。格兰杰检验的基本公式是：

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_p y_{t-p} + b_1 x_{t-1} + \cdots + b_p x_{t-p} \tag{19-1}$$

然后，我们进行 F 检验来评价保留或删除滞后 x 项是否会在拟合数据方面产生统计学上的

显著性差异。因为格兰杰检验本质上是一个 F 检验，它自然给我们一个因果关系的值。我们可以根据所有 x_i 项的系数来估计 x 对 y 的影响；例如，我们可以在所有 x_i 项的系数上求平均 $\frac{\sum_{i=1}^{p} b_i}{p}$，并把它作为一个“影响值”。影响值可以被用来给选择的种子词分配权重，这样高度相关的词在我们使用主题建模之前就有了更高的概率。

我们现在展示一些通过这种方法生成的示例结果来说明这个方法可以支持的一些应用。我们首先在图 19-17 中展示了一个从新闻数据集中以两个不同的股票时间序列作为上下文来发现因果主题的例子。



图 19-17　采样结果：使用两个不同的时间序列挖掘到的主题（调整自 Kim 等[2013]）

这里所用的文本数据集是从 2000 年 6 月至 2011 年 12 月期间《纽约时报》上的新闻文章。使用的时间序列是两家公司的股票价格，这两家公司是美国航空公司（AAMRQ）和苹果公司（Apple）。如果我们使用主题模型挖掘新闻数据集来发现主题，我们将获得对美国航空和苹果公司都中性的主题。

我们想看看当我们使用美国航空和苹果公司的股票时间序列作为上下文时，是否能发现偏向美国航空公司或苹果公司的主题。这个结果表明，迭代因果主题建模方法确实产生了更偏向每个股票时间序列的主题。具体来说，我们在左栏上看到与美国航空高度相关的主题，包括机场和航空公司的主题，以及有关恐怖主义的主题（这个主题是相关的，因为 2001 年 9 月 11 日的恐怖袭击对美国航空公司造成了负面影响），但其他主题与美国航空公司股票的相关性并不明显。相反，我们在右边看到的主题非常明显地与 AAPL 更相关，包括计算机技术主题，还有另一个互联网主题。虽然并不是所有的主题都可以很容易地解释，但是很明显的是把时间序列作为上下文已经对发现的主题产生了影响，并使得模型能够找到与时间序列直觉上相关的特定主题。这些主题可以作为分析的切入点，以便进一步研究主题和时间序列之间可能存在的因果关系的细节。这些结果也清楚地表明人类必须在因果主题发现的任何实际应用中发挥重要作用，但这些主题可以立即用作预测模型中的特征（用于预测股票价格）。我们可以合理地假定这些主题和像 n 元词这样的简单特征相比或者和那些没有考虑时间序列上下文而得到的普通主题相比都是更好的特征。

437
~
438

在图 19-18 中，我们从总统选举时间序列的分析中看到了一些额外的结果。这里使用的时间序列数据来自爱荷华州的电子市场，文本数据是 2000 年 5～10 月的《纽约时报》数据，至少匹配上一位候选人的名字（即不是布什就是戈尔）。这里的目标是看是否可以用因

果主题挖掘来帮助理解 2000 年总统竞选和选举中哪些问题是重要问题。这里展示的结果是来自《纽约时报》中最显著的主题(因果主题)中排名最高的前三个词。从直觉上来说，这些词确实与竞选有关。发现的主题的高相关性至少部分是由于把总统候选人的名字作为一个额外的上下文使用(即过滤器)，这有助于消除很多非相关的文本数据。然而，有趣的是，这份名单中确实包含了一些在总统选举中非常重要的主题，包括减税、石油能源、堕胎、枪支管制(更多的结果讨论细节请参见 Kim 等[2013])。

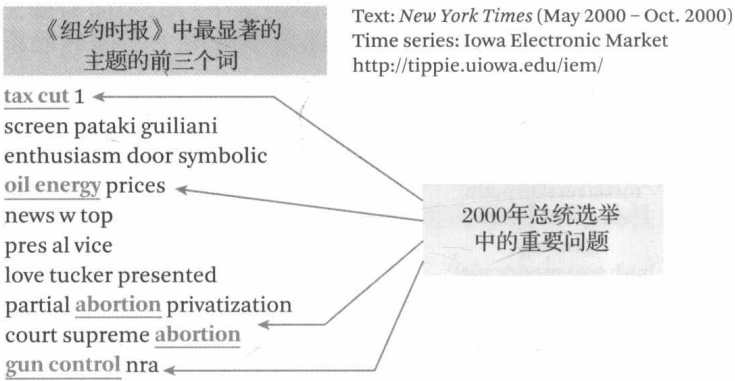


图 19-18　利用总统选举市场发现的因果主题采样(调整自 Kim 等[2013])

19.6　小结

在本章中，我们讨论以联合方式分析结构数据和非结构文本数据的一般性问题，这个对于大数据上的预测建模来说是需要的。我们特别关注了基于文本的预测，这对于涉及文本数据的大数据应用通常很有用。因为基于文本的预测可以帮助我们推断出关于世界的新知识，其中一些甚至可以超越文本内容中讨论的内容，基于文本的预测常常对优化我们的决策非常有用，特别是当与其他可用的非文本数据相结合时，它也有广泛的应用。

439

非文本数据可以提供挖掘文本数据的上下文，而文本数据也有助于解释从非文本数据(如模式注释)中发现的模式。文本与非文本数据的联合分析是一个比较新的前沿领域。在这一章中，我们介绍了一些结合主题分析与非文本数据的一般性技术，包括上下文概率潜在语义分析(CPLSA)，该模型把像时间和位置这样的上下文变量直接嵌入一个主题模型中，网络监督主题建模，它使用文本数据的伴随网络或者图结构来正则化主题发现，还有以时间序列作为上下文从文本数据中发现可能的因果主题。由于篇幅有限，我们仅简要介绍了这些方法的顶层思想并没有详细阐述，但我们已经将所有这些方法的示例结果包括在内，以帮助读者理解它们可能的应用场景。这些方法都是一般性的通用方法，因此它们可以在不同的领域中使用。

书目说明和延伸阅读

Mei[2009]的博士论文对上下文文本挖掘进行了很好的讨论，进行了许多上下文文本分析的具体的探索，特别是主题建模。具体来说，本文详细讨论了上下文概率潜在语义分析模型[Mei 和 Zhai 2006]和以网络为上下文的主题建模[Mei 等 2008]。在 Kim 等[2013]中，作者描述了以时间序列为监督的迭代主题建模方法。文本驱动的预测及其应用的一般讨论请参考 Smith[2010]。文本和伴随结构数据通常可以被建模为一个信息网络。分析信息网络算法的系统讨论请参考 Sun 和 Han[2012]。如 Zhang 等[2009]中描述的那样，文本

和结构数据的交互联合分析也可以通过结合带主题建模的传统联机分析处理(OLAP)技术来支持，使用户在“文本维度”中使用分层主题结构向下钻取和卷起。

练习

19.1　我们给出了几个例子，说明如何将文本和结构化数据集成到一个应用中。头脑风暴：本章未讨论的其他一些想法。什么样的技术将能够支持这些应用？例如，聚类、分类或主题分析。 440

19.2　在关于上下文文本挖掘的讨论中，我们展示了几种划分数据的方法，例如，研究论文的年份或地点。假设我们想添加另一个上下文变量，比如作者从属关系。我们可以简单地基于目标单位来过滤结果，还是需要重新运行整个应用以引入这些额外的知识？

19.3　在上下文文本挖掘中，我们给出了几个查询示例。

- 在数据挖掘研究中，最近哪些主题受到越来越多的关注？
- 不同地区的人们对某一事件的反应有什么不同吗？
- 两个研究人员共同的研究兴趣是什么？
- 美国的作者和其他国家的作者发表的研究课题有什么不同吗？

这样一个系统返回的实际答案是什么？例如，系统将返回什么实际对象？人类用户需要做什么来解释它们呢？

19.4　CPLSA 和 LARA 或其他情感主题的挖掘算法以哪些方式相关？在某些特定任务中，是否有某一个优于其他方法？

19.5　对于具有网络上下文的主题分析，我们使用网络来强制对主题模型进行约束。我们能用同样的模型来预测节点之间何时形成联系吗？如果没有，有可能调整模型来支持这一点吗？

19.6　MeTA 包含一个图处理库。你能把这个与 MeTA 的主题模型结合起来创建一个网络约束的主题分析吗？例如，Yelp 的学术资料集(https://www.yelp.com/academic_dataset)，除了商业评论数据外，它还包含文本和网络信息。

19.7　时间序列数据的分辨率会对文本数据的联合应用有一定的影响。例如，考虑一天之内股票数据的分辨率为 1 秒(也就是说，每秒钟有一个数据点)。集成报纸文章的文本挖掘应用能利用这些数据吗？如何更好地利用这些数据？

19.8　在时间序列应用中，数据通常是流式的(在指定的时间间隔接收更新)。讨论的模型支持流时间序列(和文本)数据吗？如果它们没有，有什么办法来调整吗？ 441 ~ 442

第四部分

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

统一的文本数据管理和分析系统

第 20 章　面向一个统一的文本管理和分析系统

第 20 章

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

面向一个统一的文本管理和分析系统

在前几章中，我们介绍了许多用来管理和分析文本数据的特定算法和技术。这些算法目前是用单独的工具包实现的。但从应用程序的角度来说，如果能开发一个统一的系统，那么就可以以更加通用的方式来支持所有这些算法，让它在许多不同的应用程序中都可以使用。在本章中，我们将讨论如何在统一的分析框架下建模这些不同的算法，并开发一个通用的系统来实现这样的框架。

从用户的角度出发，我们应该区分开两个相关的应用任务：文本数据获取和文本分析。文本数据获取的目的是使用户能够识别、获取与应用程序相关的最有用的文本数据。这通常是要依靠搜索引擎来实现的。目前的搜索引擎主要支持查询功能，然而也只有这一种功能可以帮助用户查找相关文档。浏览是信息获取的另一种有效形式，用户可以主动地通过一个导航链接到相关信息中。人们常常通过使用聚类、分类或主题分析向文本数据中添加结构来实现对浏览功能的支持。除了支持查询和浏览，搜索引擎还可以支持信息推荐，从而通过拉取模式(查询和浏览)和推送模式(推荐)提供多种模式的信息获取。

文本数据是由人类创建的，这也就意味着文本数据通常是由人来使用的。由于目前的 445
NLP 技术还不够成熟，计算机还无法准确地理解文本数据，所以通常才需要将人也纳入文本数据应用中。从这个意义上说，文本数据管理和分析系统应该作为需要信息的用户的智能助手，或者希望利用文本数据进行智能决策的分析人员的智能助手。

因此，优化人类与机器的协作是非常重要的。这意味着我们应该利用计算机处理大量文本数据的能力，同时利用人类在细节的语言理解和获取知识进行决策的专长。支持交互式过程，使用户能与智能文本信息系统“对话”是有益的，因为它能使用户和系统有更多的沟通和互助，以通过对文本数据的分析实现解决用户问题的共同目标。

以这种方式看待问题，我们可以很容易地看到以不同的方式划分人和机器之间的工作的可能性。一种极端情况是，我们主要依靠用户完成获取和分析任务，并且让计算机只做计算机能做的事情。这是目前人们使用搜索引擎进行大部分手动文本挖掘时的场景，如图 20-1 所示。例如，我们都可以使用 Web 搜索引擎帮助我们访问 Web 上大量文本数据中最

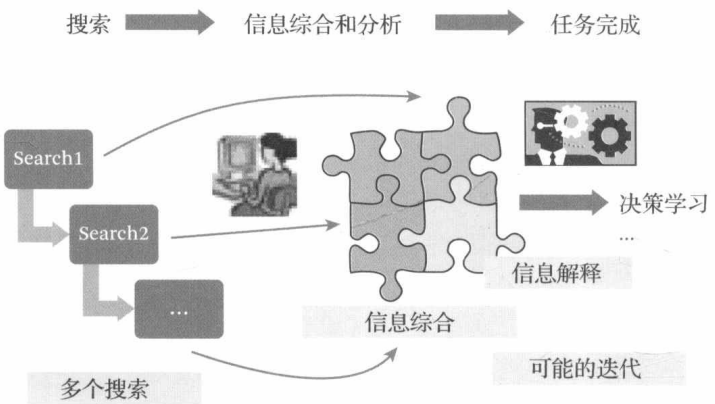


图 20-1 一个基于搜索引擎的人工分析

相关的信息，然后读取所有相关信息，这基本上是人工文本挖掘和分析。在这种情况下，搜索引擎起着两个重要的作用。首先，它使用户能够快速识别最相关的文本数据，这通常是我们解决问题真正需要的，从而避免处理大量不相关的文本数据。这种数据选择(简化)策略在逻辑上是我们在处理由于数据大小引起的可扩展性问题时应该采取的第一步。其次，它提供了知识来源，在这个意义上，它允许用户很容易地访问任何来源，以深入审查其可靠性。在这种情况下，计算机在支持文本分析上做得很少。然而，该系统(即搜索引擎)非常健壮，能够快速处理大量数据。 446

另一种极端情况是，系统可以尝试直接向用户提供任务支持，以尽量减少用户的工作，如图 20-2 所示。

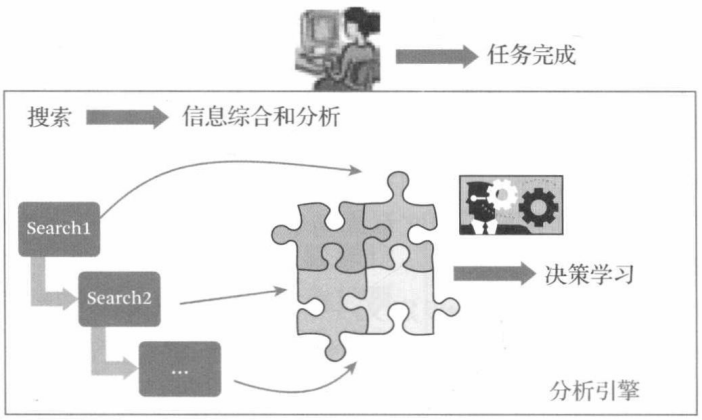


图 20-2　扩展搜索引擎来构建一个直接支持用户任务的分析引擎

由于计算机无法理解自然语言且缺乏对特定任务要求的了解，对所有或是大量的任务提供这种支持是不可能的。因此，我们只能为特定的问题建立非常特殊的预测模型，其特性一般都是由人来设计的。这种系统可以为决策任务提供最大限度的支持，但系统的功能不可避免地受到限制。由于预测建模一般需要使用机器学习技术，并结合从文本和非文本数据中提取的特征，因此显然需要在文本管理和分析之外提供更多的支持。然而，我们可以设想开发一个相对通用的文本分析系统的可行性，以帮助用户识别和提取特定预测任务的有效特征。为了确保通用性，这样的文本分析系统必须提供能够应用于许多不同问题的相对通用的文本分析操作；我们在本书中介绍的大多数算法都具有这种性质，因此可以作为通用运算符在系统中实现。然而，每个特定的文本分析应用都不可避免地有不同的要求，因此操作也必须标准化并且具有兼容性，以便它们可以灵活地组合起来，潜在地支持许多不同的工作流。请注意，文本数据获取功能，如查询、浏览和推荐，可以看作是特定操作的实例，可与其他操作(如分类、聚类或主题分析)“相互配合”。 447

这样一个基于操作的支持文本挖掘和文本分析的系统，其优点等同于将全套的汽车修理工具交给维修人员。为了诊断并解决汽车问题，维修人员需要使用不同的工具，并以某种方式组合打开汽车的元器件或利用探针进行电气测试。虽然实际的工作流程会因汽车的具体问题和型号而异，但一套通用的工具往往足以满足各种任务。同样，为了整合分散的信息，消化所有有关文本数据潜在的相关知识，分析人员也可以结合各种文本分析工具来建立实体间的复杂关系，挖掘文本中的潜在主题，识别有趣的模式以及为预测任务计算有用的特征。正如汽车维修的例子，虽然实际的工作流可能由于不同应用需求而大大不同(这取决于特定的应用)，但可扩展和可训练的一系列常用分析工具可能足以适应各种文本

分析应用。我们在本书中介绍的大多数算法都可以作为这样的分析工具，从而为开发一个统一的系统提供了基础，使分析人员能够以这样的方式执行文本分析。虽然设计和构造这样的系统还存在许多挑战，但这是一个值得去实现的目标。

20.1　文本分析操作

关系数据库能够通过查询语言(如 SQL)中定义的一组公共操作支持许多不同的应用程序。类似地，我们可以定制支持文本分析的一组通用操作。在本节中，我们将讨论定义文本分析操作的一些可能性。

448

第一，需要定义在处理过程中我们可能感兴趣的数据类型。当然，最重要的数据类型是 TextObject，它可以被定义为出现在词集合 V 中的词序列。显然，一个词、一个短语、一个句子、一个段落、一篇文章都可以视为 TextObject 的特定实例。我们还可以基于 TextObject 定义一些衍生的数据类型，包括像 TextObjectSet，它轻松捕获文章集合、句子集合等实例，其中涵盖了术语集合，在这种特殊情况中一个句子可能另包括一个词项。另一种可能是 TextObjectSequence，我们关心的是文本对象的顺序；TextObjectSequence 可以捕捉有趣的数据结构，如排好序的文章或句子列表。同样，一个特殊的情况是排好序的词项列表。

基于这些类型，我们还可以进一步定义一个主题 WeightedTextObjectSet，其中每个文本对象与一个数字权重关联。作为一种特殊情况，我们可以把词作为文本对象，从而可以用 WeightedTextObjectSet 表示一个词分布。WeightedTextObjectSequence 可以表示带有打分的搜索结果的排序列表。

第二，需要在各种数据类型之上定义操作。我们可能会有大量的操作，这取决于特定的文本分析算法。我们将简要讨论在本书的前几章中提到的几种最常用的算法，并展示即使用少量操作也可以用多种不同的方式将它们结合起来，灵活地支持特定应用场景中需要的不同工作流。

例如，在图 20-3 中，我们将展示以下操作。由于我们的目标主要是提出一些推测性的想法，所以以非正式、非严谨的方式呈现这些操作。

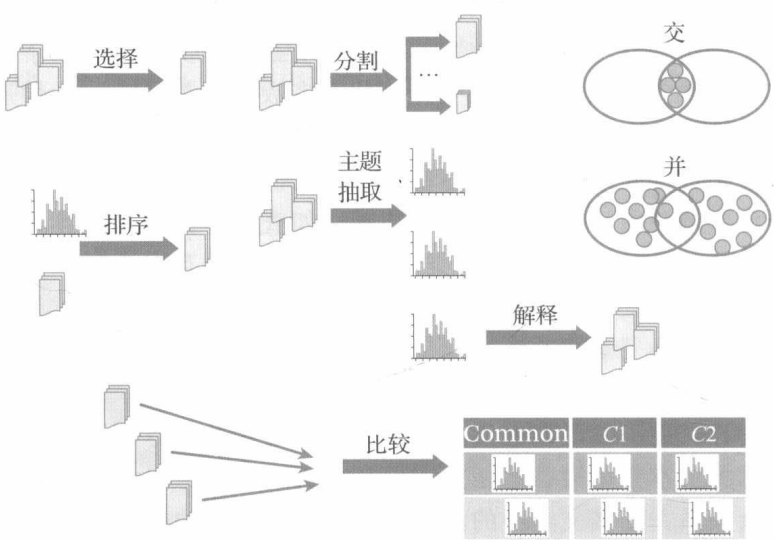


图 20-3　文本分析的可能操作示例图

选择(select)。TextObjectSet→TextObjectSet。选择操作把一组文本对象映射到一个文本对象子集上。查询、浏览和推荐都可以被看成选择的实例。我们可以额外地在每一个选择上进一步使用排序操作。

分割(split)。TextObjectSet→TextObjectSet，…，TextObjectSet。分割操作把文本对象集合映射到多个文本对象子集上。文本分类和聚类是分割操作的实例。

并(union)与交(intersection)。这是可以应用到任何集合上的标准集合操作。

排序(ranking)。(Weighted) TextObjectSet→TextObjectSequence。排序操作把一个加权的文本对象集合作为输入。这些文本对象指定排序方法和文本对象集，并且产生一个有序的文本对象序列作为输出。作为特例，Weighted TextObjectSet 可以是一个表示查询语言模型的词分布。

主题抽取(topic extration)。TextObjectSet→TopicSet。主题抽取操作把文本对象集合映射到一组主题上。

解释(interpret)。{Topic，TextObjectSet}→TextObjectSet。解释操作把一个主题和一个文本对象集合映射到另一个文本对象集合上。

比较(compare)。{TextObjectSet，…，TextObjectSet}→TopicSet，…，TopicsSet。比较操作把一个可比较的文本对象集合映射到一个能够覆盖所有可比较文本对象的通用主题集合以及特定上下文主题集合上。

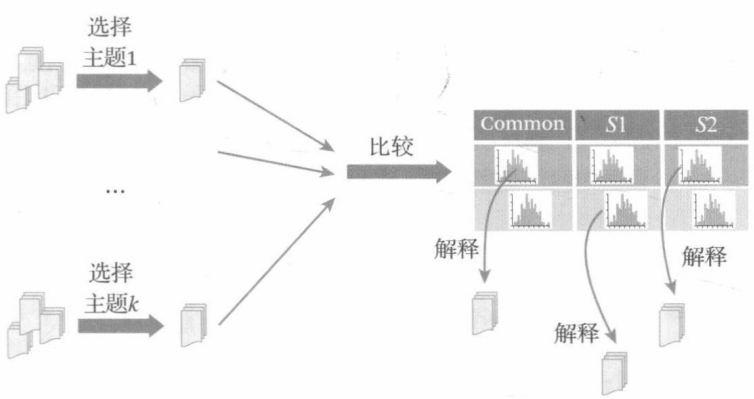
一些操作的形式化定义如图 20-4 所示，θ 表示加权词向量。即使只有这些操作，我们也已经形成了一些有趣的组合。例如，在图 20-5 中，我们看到了一个示例，先是主题选择操作，接着是比较操作，最后是解释操作。

在图 20-6 中，我们展示了另外一个例子，先是分割操作，接着是比较操作，然后是解释操作。很容易想象，有许多其他方法来组合这些操作。

- $C = \{D1, \ldots, Dn\}$; S, C的子集 $S1, S2, \ldots, Sk$
- 选择
  - 查询 (Q): $C \rightarrow S$
  - 浏览: $C \rightarrow S$
- 分割
  - 分类（有监督）：$C \rightarrow S1, S2, \ldots, Sk$
  - 聚类（无监督）：$C \rightarrow S1, S2, \ldots, Sk$
- 解释
  - $C \times \theta \rightarrow S$
- 排序
  - $\theta \times Si \rightarrow$有序的$Si$

图 20-4　文本分析操作的形式化

449
〜
450



Interpret(**Compare**(Select($T1$, C), Select($T2$, C), ..., Select(Tk, C)), C)

451

图 20-5　组合主题选择、比较、解释操作的例子

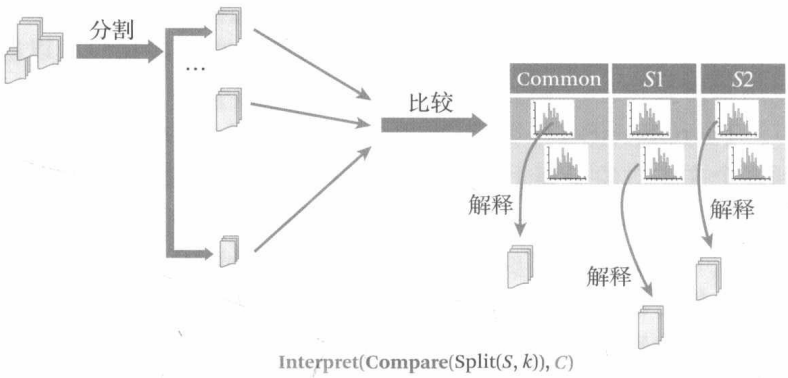


图 20-6　组合主题分割、比较、解释操作的例子

20.2　系统架构

总的来说，我们可以想象一个具有高层框架的用于支持文本管理和分析的系统，如图 20-7 所示，它提供给用户多层次的服务，包括：对自然语言处理的预处理，对多模式文本数据获取的低层服务(其中包括查询、浏览和推荐)，对文本数据分析的中层服务(其中包括通用的分析操作以及操作间彼此组合)，对高层应用的支持(其中包括支持特定应用的用户任务)。用户能够通过统一的交互界面访问所有的层级，还能访问针对特定用户和特定应用任务的个性化工作空间，这是很重要的。在这个图中，我们也展示了非文本数据的可用性，这通常需要一个数据库系统进行管理并为其他的模块进行服务，例如文本分析模块(在这个场景中，非文本数据可以被用来作为分析文本数据的上下文)。最后，我们看到
452 可以进一步应用常用的数据挖掘算法如 EM 以及预测模型来处理从分析人员获得的结果。

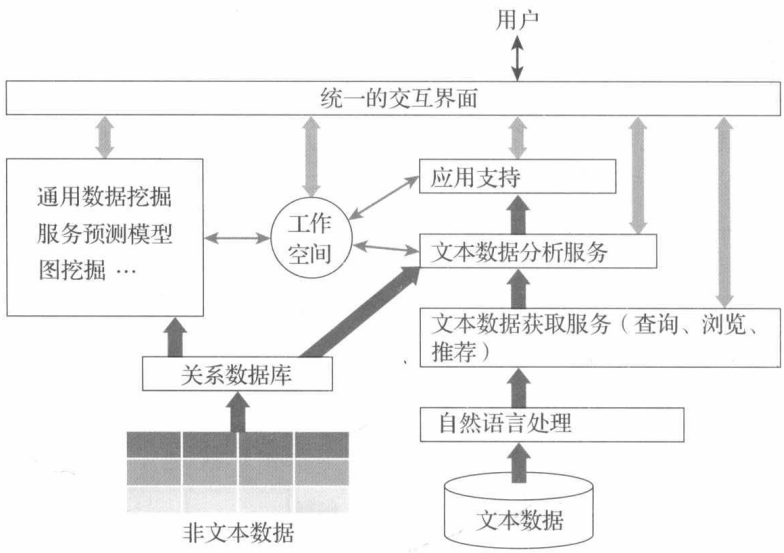


图 20-7　一个统一的文本分析系统的高层架构

举一个例子，考虑一个新闻挖掘应用程序。如果用户执行关键字搜索，则可能返回包含匹配文本的大量文档。如果我们在返回的文档集上运行主题分析，通过浏览结果能够使用户更有效地筛选数据。如果用户找到两个有用的聚类，可以将它们合并在一起，并用不同的关键字重新搜索。可以重复此过程，直到满足用户的信息需求为止。

20.3　MeTA 作为一个统一系统

在本节中，我们讨论将统一的系统 MeTA 扩展到一般的文本数据管理和分析系统的可能性。一方面，我们可以设想实现图 20-8 所示的架构，通过扩展搜索引擎来支持各种主题模型，从而进一步改进文本分类、文本摘要和文本聚类。

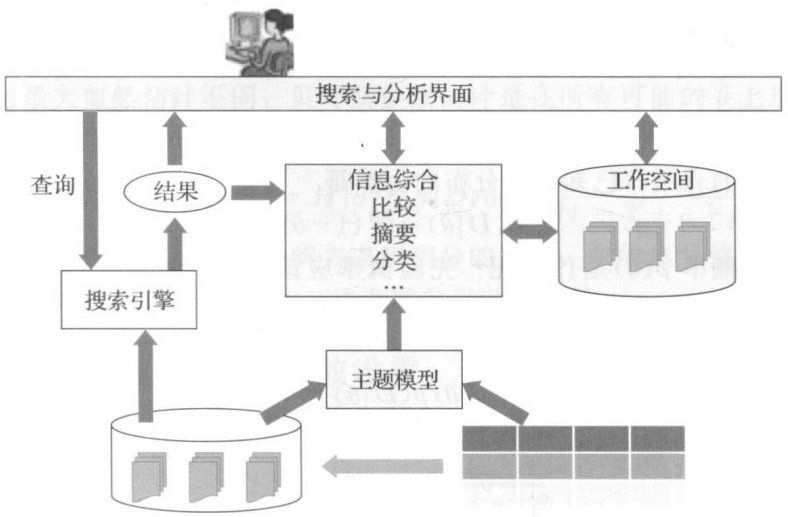


图 20-8　扩展搜索引擎得到一个通用的主题分析引擎

索引(正向和倒排)是 MeTA 中大多数文本挖掘应用程序共用的存储机制。分析器和分词器是首先应用于文本并创建各种特征以及分配词项 ID 的操作。如果词项不存储在索引中，则它们通常存储在 meta::sequence 中，它维护词项顺序以便进一步分析。此外，大多数文本挖掘应用都将索引作为输入。

所有任务的这种共有结构允许实现更高层的“封装器”功能，其作用与本章前面描述的方法相同。例如：

- TextObject 在 MeTA 中被表示成 **meta::corpus::document**。
- TextObjectSet 是 **meta::index::forward_index** 或者 **meta::index::inverted_index**。
- 一个 TextObjectSequence 的例子是 **meta::sequence::sequence**。
- 一个 WeightedTextObjectSequence 的例子是 **meta::index::ranker** 中得分函数的输出。

如果我们利用更高级别的函数将这些对象传递给不同的分析组件，将启用文本分析的统一视图。**meta::corpus::metadata** 允许为每个文档存储基本的结构化数据。如数据库中的更高级的结构化操作目前尚未实现，但支持简单的过滤和分组命令，例如过滤函数 **meta::index::ranker** 和 **meta::classify::classifier::knn**。

453 ~ 454

索引结构也有助于前面提到的想法：高级文本挖掘技术运行在整个数据集的较小相关子集上。为此，我们可以搜索倒排索引得到输出并在相关文档上运行主题模型算法。

因此，分析操作(如选择和分割)将 MeTA 的索引对象作为输入并且返回对象(例如序列)。该算法运行的选择操作可以是一个过滤器或搜索引擎，分割操作可以是 **meta::topics::lda_model** 或者一些其他的聚类算法。

455 ~ 456

附录 A

Text Data Management and Analysis：A Practical Introduction to Information Retrieval and Text Mining

贝叶斯统计

作为第 2 章和第 17 章的延续，本章将更为深入地介绍贝叶斯统计的相关知识。

A.1 二项估计和 Beta 分布

通过 2.1.5 节的介绍，已知二项分布的似然是

$$p(D|\theta)=\theta^H(1-\theta)^T \tag{A-1}$$

那么它对应的先验概率 $p(\theta)$是什么呢？先验概率应该反应一些关于模型参数的“先验知识”。对于抛硬币(即二项分布)来说，使先验概率与 θ 和$(1-\theta)$的幂成比例是合理的。因此后验概率也将与这些幂成比例：

$$\begin{aligned}p(\theta|D)&\propto p(\theta)p(D|\theta)\\&=\theta^a(1-\theta)^b\theta^H(1-\theta)^T\\&=\theta^{a+H}(1-\theta)^{b+T}\end{aligned}$$

所以需要找到某个具有 $P(\theta)\propto\theta^a(1-\theta)^b$ 形式的分布。幸运的是，有一种被称为 Beta 分布的概率分布，可以满足这种要求。如果 $x\in[0,1]$，则称 $x\sim\text{Beta}(\alpha,\beta)$。

$$p(x|\alpha,\beta)=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1} \tag{A-2}$$

这是 Beta 分布的概率密度函数(probability density function，pdf)。但什么是

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \tag{A-3}$$

$\Gamma(\cdot)$是 Gamma 函数，它可以被看作阶乘函数的连续版本。也就是说，$\Gamma(x)=(x-1)\Gamma(x-1)$。特别地，对于 $x\in\mathbb{Z}^+$，$\Gamma(x)=(x-1)!$。但这还是不能解释 $x^{\alpha-1}(1-x)^{\beta-1}$ 前面
457 那个常数的作用。

事实上，这个常数只是为了确保给定参数 α 和 β 后，Beta 分布在它的支撑集上积分和仍然为一。这是一个概率分布的必要条件。在数学上，也可以写成

$$\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{A-4}$$

注意到，x 支撑集上的和是这个常数的倒数。如果除以积分和(乘以它的倒数)，就会得到想要的结果：

$$\int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x=1 \tag{A-5}$$

如果熟悉微积分(或者知道如何使用 Wolfram Alpha 或者类似的工具)，容易证明这个式子。

关于 Beta 分布，另一个需要注意的地方是，它的期望值是

$$\frac{\alpha}{\alpha+\beta} \tag{A-6}$$

很快就可以明白这个式子的用处。通过重写对 $p(\theta|D)$的估计，可以观察到的数据是 H、T。此外，引入两个超参数 α 和 β 作为 Beta 分布的先验。之所以被称为超参数，是因

为它们是先验分布的参数。

$$\begin{aligned} p(\theta|H,\ T,\ \alpha,\ \beta) &\propto p(H,\ T|\theta)p(\theta|\alpha,\ \beta) \\ &\propto \theta^{H}(1-\theta)^{T}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{H+\alpha-1}(1-\theta)^{T+\beta-1} \end{aligned}$$

但这本身就是一个 Beta 分布，即

$$p(\theta|H,\ T,\ \alpha,\ \beta) = Beta(H+\alpha,\ T+\beta) \tag{A-7}$$

最后，可以得到贝叶斯参数估计的结果。最大似然估计得到的参数使得数据的似然函数达到最大；与最大似然估计不同，贝叶斯参数估计是在所有可能的 θ 上进行积分，计算给定数据上的期望值 $E[\theta|D]$。在这种情况下，“数据”是翻转的结果，以及超参数 α 和 β：

$$E[\theta|D] = \int_0^1 p(x = H|\theta)p(\theta|D)\mathrm{d}\theta = \frac{H+\alpha}{H+T+\alpha+\beta} \tag{A-8}$$

因为不是本书关注的重点，这里略去求解积分的细节，只给出最终的推导结果。这个结果适用于任何一个具有 Beta 先验的二项参数的贝叶斯估计。 458

A.2　伪计数、平滑处理和超参数设置

那么应该如何解释二项分布参数的贝叶斯估计结果呢？

$$E[\theta|D] = \frac{H+\alpha}{H+T+\alpha+\beta} \tag{A-9}$$

通过上面的讨论，可见 Beta 分布和二项分布是相似的。事实上，Beta 分布是二项分布的共轭先验。所有指数族中的概率分布都有共轭先验。共轭关系可以描述为：给定一个基于 X 分布的似然，选择 X 的共轭先验分布(假设它是 Y)将确保得到的后验分布也是 Y 分布。

对于抛硬币的例子，概率是二项分布。如果选择 Beta 分布作为先验，则后验分布最终也将服从 Beta 分布——这是因为选择了共轭先验！

在任何情况下，基于先验的所有推理，是我们在看到数据之前对参数进行的一些合理猜测。对于抛硬币这个例子，可能想要假设拥有一个“公平”的硬币。如果由于某种原因硬币可能是有偏的，那么也可以融入这些先验知识。

如果观察 θ 的估计，可以发现设置超参数能够影响预测的结果。回顾一下，θ 是出现正面的概率。如果希望出现正面的次数更多，可以设置 $\alpha > \beta$，因为这样可以增大 θ。这也与先验的平均值 $\frac{\alpha}{\alpha+\beta}$ 一致。将平均值设为 0.8 意味着我们的先验知识认为硬币会以 80% 的 459 概率抛出正面。

这可以通过设置 $\alpha=4$，$\beta=1$，或者 $\alpha=16$，$\beta=4$，亦或 $\alpha=0.4$，$\beta=0.1$ 来实现。但这些设置之间的区别是什么呢？图 A-1 显示了不同参数对应的 Beta 分布之间的比较。同样要记住，从 Beta 先验 $\theta \sim Beta(\alpha,\ \beta)$ 得到的值为我们提供了一个分布。尽管它是范围[0，1]上的单个值，我们仍然能够使用先验来产生一个概率分布。

拿一个均值为 0.8 的单峰 Beta 先验举例。从图 A-1 可以看出，α 和 β 越大，0.8 处的峰值就越高。下面看一下参数估计

$$\frac{H+\alpha}{H+T+\alpha+\beta} \tag{A-10}$$

可以将超参数看作伪计数——通过“已经完成的”实验结果获得的计数。超参数值越大，伪计数就越多，这意味着先验越“强”。随着实验总数的增加，$H+T$ 也会增加，这意味着对先验的依赖越少。

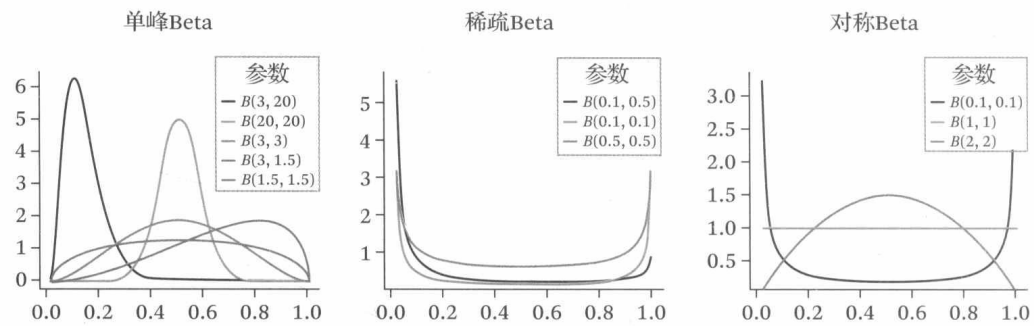


图 A-1　α 和 β 参数如何影响 Beta 分布的形状。从左到右分别是单峰 Beta(α，$\beta>1$)、稀疏 Beta(α，$\beta<1$)和对称 Beta($\alpha=\beta$)

最初 $H+T$ 相对较小时，先验在 θ 的估计中起着较强的作用。众所周知，少量的抛掷次数不能准确地估计真正的 θ——这里主要关心随着抛掷次数接近无穷大(或者某个“足够大”的值)，先验对于估计结果的影响。在这个意义上，先验可以起到平滑估计结果的作用。如果有一个较好的先验，估计就可以保持相对平稳，而不是初期波动很大。

如果先验是错误的，那么观察到的数据最终会掩盖超参数中的伪计数，因为 α 和 β 是保持不变的。

A.3　多项分布

此时，你可能已经理解狄利克雷先验平滑在信息检索语言模型和主题模型中起到的作用。但现在概率是定义在单词上的，而不再是定义在正面和反面这样的二元结果上。在讨论狄利克雷分布之前，首先讨论如何表示观察到词汇表中某个单词的概率。

为此，可以使用类别分布。在文本信息系统中，类别分布可以表示单个文档的一元语言模型。其中，类别总数 k 等于词汇表大小 $|V|$，下标为 i 的单词出现的概率是 p_i，所有
460 单词的概率和是 1。

类别分布与多项分布的关系就像伯努利(Bernoulli)分布与二项分布的关系。多项分布是在 n 次实验中观察每个单词 k_i 出现 x_i 次的概率。如果给定一个文档的计数向量，我们可以使用多项分布来计算观察到这个文档的概率(忽略位置)。其概率密度函数定义如下：

$$p(X_1=x_i,\ \cdots,\ X_k=x_k)\frac{n!}{x_1!\ \cdots x_k!}p_1^{x_1}\cdots p_k^{x_k} \tag{A-11}$$

其中

$$\sum_{i=1}^{k}x_i=n,\quad \sum_{i=1}^{k}p_i=1$$

它的概率密度函数也可以写成

$$p(X_1=x_i,\cdots,X_k=x_k)=\frac{\Gamma\left(\sum_{i=1}^{k}x_i+1\right)}{\prod_{i=1}^{k}\Gamma(x_i+1)}\prod_{i=1}^{k}p_i^{x_i} \tag{A-12}$$

很容易把更一般的多项分布与对应的二项分布关联起来。

A.4　狄利克雷分布

现在已经确定了具有 k 个结果的多项分布的似然函数。多项分布的共轭先验是狄利克雷分布。也就是说，如果使用狄利克雷分布作为先验，那么后验也将是狄利克雷分布。

与多项分布类似，狄利克雷分布是一个和为 1 的正向量上的分布。（“单纯形”是这些向量所在空间的名字。）与 Beta 分布类似，狄利克雷分布的参数是实数，其概率密度函数为：

$$p(\theta|\boldsymbol{\alpha}) = \frac{\Gamma\sum_{i=1}^{k}\alpha_i}{\prod_{i=1}^{k}\Gamma(\alpha_i)}, \quad \prod_{i=1}^{k}\theta_i^{\alpha_i-1} \tag{A-13}$$

这个式子中，我们获得了 $p(\theta|\boldsymbol{\alpha})$的形式，其中 θ 来自狄利克雷分布。在 Beta 分布里，它是二项分布中使用的参数。在这里，它是多项分布中使用的参数向量。从这个意义上来说，狄利克雷是一个生成分布的分布（Beta 分布也是这样！）。狄利克雷的超参数也是一个向量（为了强调用箭头表示），不同的是 Beta 分布中只有两个超参数，而狄利克雷分布有 k 个参数——每个多项概率都需要一个参数。 461

当选择先验设置时，通常没有任何关于狄利克雷中各项参数的具体信息，因此将它们设置为相同的值。所以将 $p(\theta|\boldsymbol{\alpha})$简写为 $p(\theta|\alpha)$，其中 α 是表示相同值向量的标量，从而 $\boldsymbol{\alpha}$ 是形如$\{0.1, 0.1, \cdots, 0.1\}$的向量。就像 $\theta \sim Beta(\alpha, \beta)$或 $\theta \sim Beta(0.4, 0.1)$一样，$\theta \sim Dir(\alpha)$和 $\theta \sim Dir(0.1)$也是常见的表示方法。

图 A-2 显示了 α 的取值对狄利克雷分布特性的影响。区域越高的点（代表一个向量）被选择的可能性就越大。

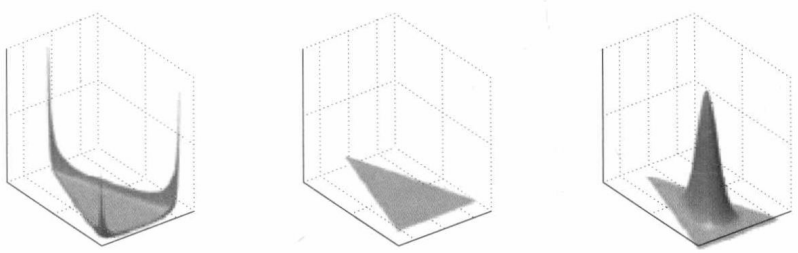


图 A-2　α 参数如何影响 3 个参数的狄利克雷分布的形状和稀疏度。从左到右分别是，$\alpha=0.1$，1，10。（摘自 Bishop[2006]）

下面来解释一下从狄利克雷分布生成一个点。在图 A-2 的最右侧子图中，如果选择的点是图中的峰值，将得到一个多项参数向量，其中三个分量都大致相等。例如，$\theta=\{0.33, 0.33, 0.34\}$。在 $\alpha=10$ 的情况下，很可能会得到这样的 θ。在中间的图中，不确定会得到什么样的 θ：均匀的合成、不均匀的合成，或者任何两者之间的组合都有可能获得。这被称为均匀先验——它可以表示没有关于先验分布的明确信息。最后，左边的图是一个稀疏先验（就像 α，$\beta<1$ 时的 Beta 分布）。

注意：均匀先验并不意味着能得到一个分量的均匀组合，它意味着有相同的可能性获得任何一种分量的组合。这可能会令人费解，因为得到的分布可能实际上就是一个均匀分布。

稀疏先验实际上在文本应用中十分重要。如果有一些维度的概率很高，其余维度可能性都相对较低，这实际上就是齐普夫定律本身的特性。可以用狄利克雷先验来实现每个主题的稀疏词分布（$\theta=\{0.9, 0.02, 0.08\}$）。在主题建模中，可以用狄利克雷分布来实现每个文档的稀疏主题分布。在大多数情况下，一个文档主要讨论少数几个主题，而其余主题大多没有 462
被讨论，就像 the、to、of 和 from 是常见的，而许多像 preternatural 这样的单词是罕见的。

A.5　多项分布参数的贝叶斯估计

下面对多项分布进行参数估计，并将其与之前的 Beta-二项分布相关联。对于最大似

然估计，可以获得

$$\theta_i = \frac{x_i}{n} \tag{A-14}$$

根据贝叶斯法则，可以将后验表示为似然(多项分布)和先验(狄利克雷分布)的乘积：

$$\begin{aligned} p(\theta|D,\alpha) &\propto p(D|\theta)p(\theta|\alpha) \\ &\propto \prod_{i=1}^{k} \theta_i^{x_i} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \\ &= \prod_{i=1}^{k} \theta_i^{x_i + \alpha_i - 1} \end{aligned}$$

可见它们是成比例的，因为我们省略了多项分布和狄利克雷分布中的比例常数(与 Gammas 的比例)。正如预期一样，现在可以观察到，由于共轭性，后验也是狄利克雷分布。

为了获得贝叶斯估计，我们需要将多项分布和狄利克雷分布全部代入后验的公式，并对 θ 进行积分以得到估计值。微积分相关的推导不是这里的重点，下面只展示最终答案

$$E[\theta_i|D] = \frac{x_i + \alpha_i}{n + \sum_{j=1}^{k} \alpha_j} \tag{A-15}$$

这看起来与二项分布估计非常类似！狄利克雷超参数也起到了伪计数的作用，从而平滑估计结果。

在信息检索中，可以使用如下的公式进行狄利克雷先验平滑：

$$p(w|d) = \frac{c(w,\ d) + \mu p(w|C)}{|d| + \mu} \tag{A-16}$$

所以有 $x = c(w,\ d)$ 和 $n = |d|$，它们分别是文档中当前单词的计数和文档的长度。然

463 后可得到 $\alpha_i = \mu p(w|C)$ 和 $\mu = \sum_{j=1}^{k} \alpha_j$，它们分别是单词 w 的伪计数的个数和伪计数的总数。现在知道查询似然平滑的超参数向量是什么了吗？它是

$$\boldsymbol{\mu} = \{\mu p(w_1|C),\ \mu p(w_2|C),\ \cdots,\ \mu p(w_k|C)\} \tag{A-17}$$

换句话说，这种平滑方法的狄利克雷先验与背景、文档集合语言模型成正比。

回顾加一(Add-1)平滑，我们可以将其视作狄利克雷先验平滑的一个特例。如果从狄利克雷中采样出均匀分布，就会得到

$$p(w|d) \frac{c(w,\ d) + 1}{|d| + |V|} \tag{A-18}$$

这意味着，每个单词都等可能地出现在文档集合语言模型中，这种情况是最不可能发生的。注意因为 $\boldsymbol{\mu} = \{1,\ 1,\ \cdots,\ 1\}$，所以 $|V| = k$ 或者 $\sum_{i=1}^{k} 1$。

A.6　结论

本节从简单抛硬币的伯努利分布开始，将它扩展成一套二项分布的试验。通过最大似然估计来探究参数估计，然后聚焦在贝叶斯方法。通过把贝叶斯的估计结果与基于伪计数的平滑方法对比，进一步了解了超参数如何影响分布。基于这个基础，继续研究了使用多维分布来刻画单个词汇的概率。我们讨论了 Beta-二项式模型与狄利克雷－多项式模型之间的关联，并且研究了如何在信息检索领域中的查询似然函数中应用狄利克雷先验进行语

464 言模型的平滑。

期望最大化

期望最大化(Expectation-Maximization，EM)算法是一种通用的优化算法，适用于解决最大似然估计中数据是“不完整的”或者似然函数涉及隐变量的问题。值得注意的是，“不完整数据”和“隐变量”的概念是相关的：当存在一个隐变量时，可以认为数据是不完整的，因为不能观察到隐变量的数值。类似地，当数据不完整时，也可以将一些隐变量与缺失的数据相关联。对于语言模型，期望最大化算法通常被用于估计混合模型的参数，其中生成数据点的来源是无法观察到的。

非正式地，期望最大化算法使用随机值初始化要估计的所有参数。然后迭代交替执行两个步骤，分别是期望步骤(即“E 步”)和最大化步骤(即“M 步”)。在 E 步中，EM 算法计算完整数据的期望似然函数(所谓的 Q 函数)，其中期望是指对于给定当前设置的参数和观察到的(不完整的)数据来计算隐变量的条件分布。在 M 步中，通过最大化 Q 函数来重新估计所有参数。一旦得到了新一轮估计的参数值，就可以重复 E 步和另一次 M 步。这个过程一直持续到似然函数收敛为止，从而达到局部最大值。直观地，期望最大化是通过“猜测”隐变量的值来迭代地补全数据，并通过假设猜测的值是真实数值来重新估计参数。

期望最大化算法是一种爬山算法，因此只能保证达到局部极大值。当有多个极大值时，能否达到全局最大值取决于起始点。如果从一个“正确的山”开始，将能够达到全局最大值。当存在多个局部极大值时，通常很难确定“正确的山”。有两个常用的策略来解决这个问题。第一个是尝试许多不同的初始值，并选择收敛后具有最高似然值的解。第二个是使用更简单的模型(理想情况下具有唯一的全局最大值)来确定更复杂模型的初始值。这个想法是，一个较简单的模型可以帮助粗略地定位全局最优解的区域，然后再从这个区域的某个值开始，使用更复杂的模型来寻找更准确的最优。

这里通过一个具体的问题来介绍期望最大化算法——估计一个简单的混合模型。有关期望最大化更深入的介绍，请参考 McLachlan 和 Krishnan[2008]。

B.1 简单混合一元语言模型

在基于混合模型的反馈方法[Zhai 和 Lafferty 2001]中，假设反馈文档集合 $\mathcal{F}=\{d_1, \cdots, d_k\}$从具有两个多项式组件模型的混合模型中“生成”。一个组件模型是背景语言模型 $p(w|C)$，另一个是待估计的主题语言模型 $p(w|\theta_F)$(其中，w 是一个单词)。这个想法是用 $p(w|C)$对 $\mathcal{F}$中的常见(没有区别力的)词进行建模，所以主题模型 θ_F 会吸收更多具有区别力的内容词。

该混合模型的反馈文档数据的对数似然是

$$\log L(\theta_F) = \log p(\mathcal{F}|\theta_F) = \sum_{i=1}^{k}\sum_{j=1}^{|d_i|}\log((1-\lambda)p(d_{ij}|\theta_F)+\lambda_p(d_{ij}|C))$$

其中 d_{ij} 表示文档 d_i 中的第 j 个词，$|d_i|$ 是文档 d_i 的长度，λ 是表示反馈文档中的“背景噪声”数量的参数，它根据经验设定。因此我们假设 λ 是已知的，并且想要估计 $p(w|\theta_F)$。

B.2　最大似然估计

估计 θ_F 的常用方法是最大似然(Maximum Likelihood，ML)估计，我们选择使 $\mathcal{F}$ 可能性最大的 θ_F。也就是说，估计的主题模型(由 $\hat{\theta}_F$ 表示)是

$$\hat{\theta}_F = \arg\max_{\theta_F} L(\theta_F) \tag{B-1}$$

466

$$= \arg\max_{\theta_F} \sum_{i=1}^{k} \sum_{j=1}^{|d_i|} \log((1-\lambda)p(d_{ij}|\theta_F) + \lambda p(d_{ij}|C)) \tag{B-2}$$

该方程的右侧可以看作是以 $p(w|\theta_F)$ 为变量的函数。为了求解 $\hat{\theta}_F$，原则上可以使用任何优化方法。但是由于该函数涉及各项和的对数，因此很难通过拉格朗日乘数法获得简单的解析解。所以一般情况下，我们需要采用数值算法。有很多种可供选择的算法，期望最大化算法只是选择之一。EM 算法可以很自然地应用，并且能够保证收敛到局部极大值。在我们的例子中，这个值也是全局最大值，因为似然函数只具有唯一的极大值。

B.3　不完整数据与完整数据

期望最大化算法的主要思想是用一些潜变量来“增广”(即补全)数据，从而“完整的”数据具有更简单的似然函数——更容易找到最大值，因此原始数据被视为“不完整的”。正如下面的内容所示，通过最大化完整数据的似然函数(因为它更容易最大化)，可以达到最大化不完整数据的似然函数(原始目标)。其中期望是通过计算隐变量的所有可能值得到的(因为完整数据的似然与原始不完整数据的似然不一样，前者包含隐变量)。

在这个例子中，为每次单词 w 的出现引入一个二元的隐变量 z，来指示该单词是从背景模型 $p(w|C)$ 还是从主题模型 $p(w|\theta_F)$ 中“生成”。令 d_{ij} 为文档 d_i 中的第 j 个词。引入一个相应的指示变量 z_{ij}，定义如下：

$$z_{ij} = \begin{cases} 1 & \text{如果 } d_{ij} \text{ 来自背景} \\ 0 & \text{否则} \end{cases}$$

因此，假设完整的数据不仅包含 $\mathcal{F}$ 中的所有单词，还包含它们对应的 z 值。因此，完整数据的对数似然是

$$L_c(\theta_F) = \log p(\mathcal{F}, z|\theta_F)$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{|d_i|} [(1-z_{ij})\log((1-\lambda)p(d_{ij}|\theta_F)) + z_{ij}\log(\lambda p(d_{ij}|C))]$$

注意 $L_c(\theta_F)$ 与 $L(\theta_F)$ 两者之间的不同：求和出现在 $L_c(\theta_F)$ 的对数之外是可能的，因为这里假设已知使用了哪个组件模型来生成每个单词 d_{ij}。

467

$L_c(\theta_F)$ 与 $L(\theta_F)$ 之间的关系是什么呢？一般来说，如果参数是 θ，原始数据是 X，并且用隐变量 H 来补全原始数据，那么有 $p(X, H|\theta) = p(H|X, \theta)p(X|\theta)$。因此，$L_c(\theta) = \log p(X, H|\theta) = \log p(X|\theta) + \log p(H|X, \theta) = L(\theta) + \log p(H|X, \theta)$。

B.4　似然函数的下界

在算法上，期望最大化的基本思想是使用初始猜测 $\theta^{(0)}$ 来初始化参数，然后迭代地搜索参数更优的解。假设当前的参数估计是 $\theta^{(n)}$，我们的目标是找到另一个可以提高似然 $L(\theta)$ 的 $\theta^{(n+1)}$。

下面考虑一个潜在更好的参数值 θ 与当前估计值 $\theta^{(n)}$ 的似然之间的差异，并将其与完整似然相应的差异关联起来：

$$L(\theta)-L(\theta^{(n)})=L_c(\theta)-L_c(\theta^{(n)})+\log\frac{p(H|X,\theta^{(n)})}{p(H|X,\theta)} \tag{B-3}$$

目标是最大化 $L(\theta)-L(\theta^{(n)})$，这等价于最大化 $L(\theta)$。现在使用给定数据 X 和当前参数估计 $\theta^{(n)}$ 隐变量的条件分布，即 $p(H|X,\theta^{(n)})$，对这个方程进行求期望，可得

$$\begin{aligned}L(\theta)+L(\theta^{(n)}) &= \sum_H L_c(\theta)p(H|X,\theta^{(n)})-\sum_H L_c(\theta^{(n)})p(H,\theta^{(n)})\\ &= \sum_H p(H|X,\theta^{(n)})\log\frac{p(H|X,\theta^{(n)})}{p(H|X,\theta)}\end{aligned}$$

请注意等式的左边保持不变，因为不包含变量 H。可以发现，最后一项是 $p(H|X,\theta^{(n)})$和 $p(H|X,\theta)$的 KL-散度，始终非负。因此可得

$$L(\theta)-L(\theta^{(n)})\geqslant\sum_H L_c(\theta)p(H|X,\theta^{(n)})-\sum_H L_c(\theta^{(n)})p(H|X,\theta^{(n)})$$

或者

$$L(\theta)\geqslant\sum_H L_c(\theta)p(H|X,\theta^{(n)})+L(\theta^{(n)})-\sum_H L_c(\theta^{(n)})p(H|X,\theta^{(n)}) \tag{B-4}$$

因此我们得到了原始似然函数的一个下界。期望最大化的主要思想是通过最大化这个下界从而达到最大化原始(不完整的)似然函数。注意这个下界的最后两项可以被视为常数，因为它们不包含变量 θ，所以这个下界实际上是第一项，也就是完整数据似然的期望，即由 $Q(\theta;\theta^{(n)})$表示的“Q 函数” 468

$$Q(\theta;\theta^{(n)})=E_{p(H|X,\theta^{(n)})}[L_c(\theta)]=\sum_H L_c(\theta)p(H|X,\theta^{(n)})$$

混合模型的 Q 函数如下

$$\begin{aligned}Q(\theta_F;\theta_F^{(n)}) &= \sum_{\boldsymbol{z}} L_c(\theta_F)p(\boldsymbol{z}|\mathcal{F},\theta_F^{(n)})\\ &= \sum_{i=1}^{k}\sum_{j=1}^{|d_i|}\left[p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})\log((1-\lambda)p(d_{ij}|\theta_F))\right.\\ &\quad \left.+p(z_{ij}=1|\mathcal{F},\theta_F^{(n)})\log(\lambda p(d_{ij}|\mathcal{C}))\right]\end{aligned}$$

B.5　期望最大化的一般步骤

显然，如果我们找到了 $\theta^{(n+1)}$ 使得 $Q(\theta^{(n+1)};\theta^{(n)})>Q(\theta^{(n)};\theta^{(n)})$，那么可得 $L(\theta^{(n+1)})>L(\theta^{(n)})$。因此，期望最大化算法的一般步骤如下。

1) 根据关于最佳参数值的任何先验知识，随机或启发式地初始化 $\theta^{(0)}$。

2) 通过以下两个步骤交替迭代地改进 θ 的估计：

a. E 步(计算期望)：计算 $Q(\theta;\theta^{(n)})$。

b. M 步(最大化)：通过最大化 Q 函数重新估计 θ：

$$\theta^{(n+1)}=\arg\max_\theta Q(\theta;\theta^{(n)})$$

3) 当似然 $L(\theta)$收敛时停止。

正如前文所述，因为隐变量的值假设已知，完整数据的似然 $L_c(\theta)$更容易最大化。这就是为什么 $L_c(\theta)$的期望——Q 函数——通常比原始似然函数更容易最大化。在数据中不存在潜变量的情况下，通常引入一个隐变量使得完全似然函数更容易最大化。

E 步主要是计算 $p(H|X,\theta^{(n)})$，有时会非常复杂。但在这个例子中，它很简单： 469

$$p(z_{ij}=1|\mathcal{F},\theta_F^{(n)})=\frac{\lambda p(d_{ij}|\mathcal{C})}{\lambda p(d_{ij}|\mathcal{C})+(1-\lambda)p(d_{ij}|\theta_F^{(n)})} \tag{B-5}$$

显然，$p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})=1-p(z_{ij}=1|\mathcal{F},\theta_F^{(n)})$。注意，通常情况下，$z_{ij}$可能依赖于 $\mathcal{F}$ 中

的所有单词。但在这个模型中，它仅取决于对应的单词 d_{ij}。

M 步涉及最大化 Q 函数，有时也会很难求解。但是在这个例子中，可以找到一个解析解。为了进行求解，我们使用拉格朗日乘数法，因为对参数变量 $\{p(w|\theta_F)\}_{w\in V}$ 具有以下约束，其中 V 是词汇表：

$$\sum_{w\in V} p(w|\theta_F) = 1$$

因此我们考虑下面的辅助函数：

$$g(\theta_F) = Q(\theta_F;\theta_F^{(n)}) + \mu\Big(1 - \sum_{w\in V} p(w|\theta_F)\Big)$$

并对每个参数变量 $p(w|\theta_F)$ 求导

$$\frac{\partial g(\theta_F)}{\partial p(w|\theta_F)} = \Big[\sum_{i=1}^{k}\sum_{j=1,d_{ij}=w}^{|d_i|}\frac{p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})}{p(w|\theta_F)}\Big] - \mu \tag{B-6}$$

令该导数为零，求解 $p(w|\theta_F)$ 的方程，可得

$$p(w|\theta_F) = \frac{\sum_{i=1}^{k}\sum_{j=1,d_{ij}=w}^{|d_i|} p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})}{\sum_{i=1}^{k}\sum_{j=1}^{|d_i|} p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})} \tag{B-7}$$

$$= \frac{\sum_{i=1}^{k} p(z_w=0|\mathcal{F},\theta_F^{(n)})c(w,d_i)}{\sum_{i=1}^{k}\sum_{w'\in V} p(z_{w'}=0|\mathcal{F},\theta_F^{(n)})c(w',d_i)} \tag{B-8}$$

注意，通过调整枚举方式，使得对于文档 d_i 中每个单词位置的累加和转变为对于词汇表中所有不同单词的累加和。这样的变换是可行的，因为 $p(z_{ij}|\mathcal{F},\theta_F^{(n)})$ 仅取决于相应的单词 d_{ij}。使用单词 w，而不是单词的出现 d_{ij} 来索引 z，可得

$$p(z_w=1|\mathcal{F},\theta_F^{(n)}) = \frac{\lambda p(w|\mathcal{C})}{\lambda p(w|\mathcal{C}) + (1-\lambda)p(w|\theta_F^{(n)})} \tag{B-9}$$

470 因此，可得下面简单混合模型的期望最大化更新公式：

$$p(z_w=1|\mathcal{F},\theta_F^{(n)}) = \frac{\lambda p(w|\mathcal{C})}{\lambda p(w|\mathcal{C}) + (1-\lambda)p(w|\theta_F^{(n)})}\text{(E)} \tag{B-10}$$

$$p(w|\theta_F^{(n+1)}) = \frac{\sum_{i=1}^{k}(1-p(z_w=1|\mathcal{F},\theta_F^{(n)}))c(w,d_i)}{\sum_{i=1}^{k}\sum_{w'\in V} p(1-p(z_{w'}=1|\mathcal{F},\theta_F^{(n)})c(w',d_i))}\text{(M)} \tag{B-11}$$

471 ~ 472 请注意我们不需要显式计算 Q 函数，只需要计算隐变量 z 的分布，然后直接得到使 Q 函数最大化的新参数值。

KL-散度和狄利克雷先验平滑

本附录详细地介绍了KL-散度，以及它与狄利克雷先验平滑在泛化查询似然平滑框架中的关系。第7章和第13章中曾简要介绍了KL-散度。

之前已经介绍过，给定两个概率质量函数 $p(x)$ 和 $q(x)$，p 和 q 之间的KL散度 $D(p \| q)$（或相对熵）被定义为

$$D(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

容易证明 $D(p \| q)$ 总是非负，当且仅当 $p=q$ 时为零。尽管不是分布之间的真实距离（因为它不是对称的，而且不满足三角不等式），将KL-散度视为分布之间的“距离”在很多情况下仍是有用的[Cover 和 Thomas 1991]。

C.1 使用KL-散度进行检索

假设查询 $\boldsymbol{q}$ 通过生成模型 $p(\boldsymbol{q}|\theta_Q)$ 产生，其中 θ_Q 表示查询一元语言模型的参数。类似地，假设文档 $\boldsymbol{d}$ 通过生成模型 $p(\boldsymbol{d}|\theta_D)$ 产生，其中 θ_D 表示文档一元语言模型的参数。如果 $\hat{\theta}_Q$ 和 $\hat{\theta}_D$ 分别是查询语言模型和文档语言模型的估计，那么 $\boldsymbol{d}$ 与 $\boldsymbol{q}$ 的相关程度可以通过负的KL-散度函数[Zhai 和 Lafferty 2001]来刻画：

$$-D(\hat{\theta}_Q \| \hat{\theta}_D) = \sum_w p(w|\hat{\theta}_Q) \log p(w|\hat{\theta}_D) + \left(-\sum_w p(w|\hat{\theta}_Q) \log p(w|\hat{\theta}_Q)\right)$$

值得注意的是，公式右侧的第二项是一个依赖于查询的常数，更具体地说，是查询模型 $\hat{\theta}_Q$ 的熵。在文档排序中可以忽略它。通常，上述公式的计算需要对 $p(w|\hat{\theta}_Q)$ 具有非零概率的单词累加求和。进一步，当 $\hat{\theta}_D$ 使用了某种通用平滑方法时，需要对 $p(w|\hat{\theta}_Q)$ 具有非零概率并且在文档 $\boldsymbol{d}$ 中出现的单词进行累加求和。这样的求和运算可以使用倒排索引来有效实现。

现在我们进行详细解释。假设通用的平滑方法如下：

$$p(w|\hat{\theta}_D) = \begin{cases} p_s(w|\boldsymbol{d}) & \text{如果词 } w \text{ 出现} \\ \alpha_d p(w|\mathcal{C}) & \text{否则} \end{cases}$$

其中 $p_s(w|\boldsymbol{d})$ 是在文档中出现某一单词的平滑概率，$p(w|\mathcal{C})$ 是文档集合对应的语言模型，α_d 是一个控制未出现的单词概率的系数，以使得所有单词概率和为一。一般来说，α_d 可能取决于 $\boldsymbol{d}$。事实上，如果给定 $p_s(w|\boldsymbol{d})$，一定有

$$\alpha = \frac{1 - \sum_{w;c(w;d)>0} p_s(w|d)}{1 - \sum_{w;c(w;d)>0} p_s(w|\mathcal{C})}$$

因此，不同平滑方法在 $p_s(w|\boldsymbol{d})$ 的选择上本质是不同的。

文档集合语言模型 $p(w|\mathcal{C})$ 通常由 $\frac{c(w,\mathcal{C})}{\sum_{w'} c(w',\mathcal{C})}$ 或者平滑后的版本 $\frac{c(w,\mathcal{C})+1}{V+\sum_{w'} c(w',\mathcal{C})}$ 进行估计，其中 V 是估计的词汇表大小（例如，文档集合中不同单词的总数）。平滑后的版本

具有一个优点：它永远不会赋予任何词项零概率，但是在检索性能方面，这两个版本不会有显著的差异，因为 $\sum_{w'} c(w', C)$ 通常显著大于 V。

可以看出，通过这种平滑方案，KL-散度评分公式本质上是(等式两侧对于排序文档是等价的)

$$\sum_{w} p(w|\hat{\theta}_Q)\log p(w|\hat{\theta}_D) = \left[\sum_{w;c(w;d)>0,p(w|\hat{\theta}_D)>0} p(w|\hat{\theta}_Q)\log\frac{p_s(w|\boldsymbol{d})}{\alpha_d p(w|C)}\right] + \log\alpha_d \qquad \text{(C-1)}$$

注意现在评分是根据 $p(w|\hat{\theta}_Q)$具有非零概率并且在文档中出现过的所有词项总和，
474 即所有“匹配”词项。

C.2 使用狄利克雷先验平滑

狄利克雷先验平滑是一种特殊的平滑方法，它遵循前面章节提到的通用平滑方法。特别地，

$$p_s(w|\boldsymbol{d}) = \frac{c(w,\ d) + \mu p(w|C)}{|d| + \mu}$$

并且

$$\alpha_d = \frac{\mu}{\mu + |d|}$$

将其代入方程(C-1)，可以看到经过狄利克雷先验平滑，KL-散度评分公式是

$$\left[\sum_{w;c(w;d)>0,p(w|\hat{\theta}_Q)>0} p(w|\hat{\theta}_Q)\log\left(1 + \frac{c(w,\boldsymbol{d})}{\mu p(w|C)}\right)\right] + \log\frac{\mu}{\mu + |d|}$$

C.3 计算查询模型 $p(w|\hat{\theta}_Q)$

现在，你可能想知道如何计算 $p(w|\hat{\theta}_Q)$。这正是 KL-散度检索方法相对于简单查询似然方法的优势——可以有不同的计算方法！最简单的方法是使用查询文本作为数据并通过最大似然估计量来估计概率，于是有

$$p_{ml}(w|\hat{\theta}_Q) = \frac{c(w,\ \boldsymbol{q})}{|q|}$$

使用这个估计值，应该很容易看出，KL-散度评分公式与 Zhai 和 Lafferty[2004]提出的查询似然检索公式本质上是相同的。

计算 $p(w|\hat{\theta}_Q)$的一个更有趣的方法是利用反馈文档。具体来说，可以将 $p_{ml}(w|\hat{\theta}_Q)$和反馈模型 $p(w|\theta_F)$结合起来，这里的反馈模型是由反馈文档估计而得。也就是

$$p(w|\hat{\theta}_Q) = (1-\alpha)p_{ml}(w|\hat{\theta}_Q) + \alpha p(w|\theta_F) \qquad \text{(C-2)}$$

其中 α 是需要根据经验设定的参数。请注意，这个 α 与平滑公式中的 α_d 是不同的。

475 当然，下一个问题是如何估计 $p(w|\theta_F)$。一种方法是假设反馈文档是由两组件混合模型生成，其中一个模型是 $p(w|\theta_F)$，另一个模型是 $p(w|C)$，即背景语言模型(或称为文档集合语言模型)。

$$\log p(\mathcal{F}|\theta_F) = \sum_{i=1}^{k}\sum_{w} c(w;d_i)\log((1-\lambda)p(w|\theta_F) + \lambda p(w|C))$$

其中，$F=\{d_1, \cdots, d_k\}$是反馈文档的集合，λ 是表示反馈文档中“背景噪声”程度的参数，需要根据经验设定。现在，给定 λ，反馈文档 $\mathcal{F}$和背景语言模型 $p(w|C)$，可以使用
476 期望最大化算法来计算 θ_F 的最大似然估计，详见附录 B。

参考文献

C. C. Aggarwal. 2015. *Data Mining - The Textbook*. Springer. DOI: 10.1007/978-3-319-14142-8. 296

C. C. Aggarwal and C. Zhai, editors. 2012. *Mining Text Data*. Springer. DOI: 10.1007/978-1-4614-3223-4. 296, 315

J. Allen. 1995. *Natural Language Understanding*. 2nd ed. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA. 54

G. Amati and C. J. Van Rijsbergen. October 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389. DOI: 10.1145/582415.582416. 87, 88, 90, 111

A. U. Asuncion, M. Welling, P. Smyth, and Y. W. Teh. 2009. On smoothing and inference for topic models. In *UAI 2009, Proc. of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada*, June 18-21, 2009, pp. 27–34. 385

R. A. Baeza-Yates and B. A. Ribeiro-Neto. 2011. *Modern Information Retrieval - the concepts and technology behind search*. 2nd ed. Pearson Education Ltd., Harlow, UK. http://www.mir2ed.org/. xvii, 18, 19

Y. Bar-Hillel, *The Present Status of Automatic Translation of Languages*, in Advances in Computers, vol. 1 (1960), pp. 91–163.

R. Belew. 2008. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press. 18

N. J. Belkin and W. B. Croft. 1992. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38. DOI: 10.1145/138859.138861. 84

C. M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ. 19, 37, 312, 385, 462

D. M. Blei, A. Y. Ng, and M. I. Jordan. March 2003. Latent Dirichlet Allocation. *J. of Mach. Learn. Res.*, 3:993–1022. 385

J. S. Breese, D. Heckerman, and C. Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, Morgan Kaufmann Publishers Inc. pp. 43–52, San Francisco, CA. http://dl.acm.org/citation.cfm?id=2074094.2074100. 235

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based N-gram Models of Natural Language. *Comput. Linguist.*, 18(4):467–479. 273, 288, 290, 291

C. Buckley. 1994. Automatic query expansion using smart: Trec 3. In *Proc. of The third Text REtrieval Conference (TREC-3*, pp. 69–80. 144

S. Büttcher, C. Clarke, and G. V. Cormack. 2010. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press. xvii, 18, 165

F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans. Web*, 5(1):2:1–2:33. DOI: 10.1145/1921591.1921593. 235

C. Campbell and Y. Ying. 2011. *Learning with Support Vector Machines*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. DOI: 10.2200/S00324ED1V01Y201102AIM010. 311

J. Carbonell and J. Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98,ACM, pp. 335–336, New York. DOI: doi=10.1.1.188.3982 321, 327

C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27. 58

J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, and D. M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, Curran Associates, Inc. 22, pp. 288–296. 272, 383, 384, 385, 410

K. W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29. http://dl.acm.org/citation.cfm?id=89086.89095. 273

T. Cover and J. Thomas. 1991. *Elements of Information Theory*. New York: Wiley. DOI: DOI: 10.1002/047174882X 37, 473

B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice*, 1st ed., Addison-Wesley Publishing Company. xvii, 18, 165

D. Das and A. F. T. Martins. 2007. A Survey on Automatic Text Summarization. Technical report, Literature Survey for the Language and Statistics II course at Carnegie Mellon University. 318, 321, 327

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874. 58

H. Fang, T. Tao, and C. Zhai. 2004. A formal study of information retrieval heuristics. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, ACM, pp. 49–56, New York. DOI: 10.1145/1008992.1009004. 129

H. Fang, T. Tao, and C. Zhai. April 2011. Diagnostic evaluation of information retrieval models. *ACM Trans. Inf. Syst.*, 29(2):7:1–7:42. DOI: 10.1145/1961209.1961210. 88, 90, 129

R. Feldman and J. Sanger. 2007. *The Text Mining Handbook - Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press. 18

E. A. Fox, M. A. Gonçalves, and R. Shen. 2012. *Theoretical Foundations for Digital Libraries: The 5S (Societies, Scenarios, Spaces, Structures, Streams) Approach*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers. DOI: 10.2200/S00434ED1V01Y201207ICR022. 80

W. B. Frakes and R. A. Baeza-Yates, editors. 1992. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 18

K. Ganesan, C. Zhai, and J. Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proc. of the 23rd International Conference on Computational Linguistics*, COLING '10, Association for Computational Linguistics, pp. 340–348, Stroudsburg, PA. 327

K. Ganesan, C. Zhai, and E. Viegas. 2012. Micropinion generation: an unsupervised approach to generating ultra-concise summaries of opinions. In *Proc. of the 21st World Wide Web Conference 2012, WWW 2012*, Lyon, France, April 16-20, 2012, pages 869–878. DOI: 10.1145/2187836.2187954 323

J. Gantz, and D. Reinsel. 2012. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, IDC Report, December, 2012. 3

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 1995. *Bayesian Data Analysis*. Chapman & Hall. 37

S. Ghemawat, H. Gobioff, and S.-T. Leung. 2003. *The Google file system*. In *Proc. of the nineteenth ACM symposium on Operating systems principles* (SOSP '03). ACM, New York, 29–43. 195

M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. A. Kipp. 2004. Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. *ACM Trans. Inf. Syst.*, 22(2):270–312. DOI: 10.1145/984321.984325. 84

D. A. Grossman and O. Frieder. Kluwer, 2004. *Information Retrieval - Algorithms and Heuristics, Second Edition*, vol. 15 of *The Kluwer International Series on Information Retrieval*. DOI: 10.1007/978-1-4020-3005-5. 18

G. Hamerly and C. Elkan. 2003. Learning the k in k-means. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS December 8-13, 2003*, Vancouver and Whistler, British Columbia, Canada], pp. 281–288. DOI: doi=10.1.1.9.3574 295

J. Han. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA. 296

D. Harman. 2011. *Information Retrieval Evaluation*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers. DOI: 10.1145/215206.215351 168, 188

M. A. Hearst. 2009. *Search User Interfaces*. 1st ed. Cambridge University Press, New York. 19, 85

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53. DOI: 10.1145/963770.963772 235

J. L. Hodges and E. L. Lehmann. 1970. *Basic Concepts of Probability and Statistics*. Holden Day, San Francisco. 36

T. Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Proc. of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, Morgan Kaufmann Publishers Inc., pp. 289–296, San Francisco, CA. DOI: 10.1145/312624.312649 370, 385

A. Huang. 2008. Similarity Measures for Text Document Clustering. In *Proc. of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, Christchurch, New Zealand, pages 49–56. 280

F. Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA. 30, 54

J. Jiang. 2012. Information extraction from text, In Charu C. Aggarwal and ChengXiang Zhai (Eds.), Mining Text Data, Springer, pp. 11–41. 19, 55

S. Jiang and C. Zhai. 2014. Random walks on adjacency graphs for mining lexical relations from big text data. In *2014 IEEE International Conference on Big Data, Big Data 2014*, Washington, DC, USA, October 27-30, pages 549–554. DOI: 10.1109/BigData.2014.7004272. 273

Y. Jo and A. H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, ACM, pp. 815–824, New York. DOI: 10.1145/1935826.1935932. 410

T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 25(2). DOI: 10.1145/1229179.1229181. 144

D. Jurafsky and J. H. Martin. 2009. *Speech and Language Processing*. 2nd ed. Prentice-Hall, Inc., Upper Saddle River, NJ. 19, 54

D. Kelly. 2009. Methods for Evaluating Interactive Information Retrieval Systems with Users. *Foundations and Trends in Information Retrieval*, 3(1-2):1–224. DOI: 10.1561/1500000012 168, 188

D. Kelly and J. Teevan. 2003. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37(2):18–28. DOI: 10.1145/959258.959260. 144

H. D. Kim, M. Castellanos, M. Hsu, C. Zhai, T. Rietz, and D. Diermeier. 2013. Mining causal topics in text data: iterative topic modeling with time series feedback. In *Proc. of the 22nd ACM international conference on Conference on information and knowledge management*, CIKM '13, ACM pages 885–890, New York, NY. DOI: 10.1145/2505515.2505612. 435, 438, 439, 440

J. M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632. DOI: 10.1145/324133.324140. 216

J. M. Kleinberg. 2002. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002*, December 9-14, 2002, Vancouver, British Columbia, Canada], pp. 446–453. http://papers.nips.cc/paper/2340-an-impossibility-theorem-for-clustering. 296

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press. 385

J. Lafferty and C. Zhai. 2003. Probabilistic relevance models based on document and query generation. In W. Bruce Croft and John Lafferty, editors, *Language Modeling and Information Retrieval*. Kluwer Academic Publishers. DOI: 10.1007/978-94-017-0171-6_1 87, 113

D. Lin. 1999. Automatic identification of non-compositional phrases. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, Association for Computational Linguistics, pages 317–324, Stroudsburg, PA. DOI: 10.3115/1034678.1034730. 273, 291

J.Lin and C. Dyer. 2010. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers. DOI: 10.2200/S00274ED1V01Y201006HLT007. 198, 216

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers. DOI: 10.2200/S00416ED1V01Y201204HLT016. 410

T.-Y. Liu. 2009. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331. DOI: 10.1561/1500000016. 216

Y. Lv and C. Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proc. of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, ACM, pp. 1895–1898, New York. DOI: 10.1145/1645953.1646259. 144

Y. Lv and C. Zhai. 2010. Positional relevance model for pseudo-relevance feedback. In *Proc. of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, ACM, pages 579–586, New York. DOI: 10.1145/1835449.1835546. 144

Y. Lv and C. Zhai. 2011. Lower-bounding Term Frequency Normalization. In *Proc. of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pp. 7–16. DOI: 10.1145/2063576.2063584 88, 110

P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L.L. Jordan, and J. Pal. 2003. How much information? http://www2.sims.berkeley.edu/research/projects/how-much-info-2003. 3

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA. 19, 54, 273

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York. xvii, 18, 165, 315

M. E. Maron and J. L. Kuhns. 1960. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7:216–244. DOI: 10.1145/321033.321035 87

S. Massung and C. Zhai. 2015. SyntacticDiff: Operator-Based Transformation for Comparative Text Mining. In *Proc. of the 3rd IEEE International Conference on Big Data*, pp. 571–580. 306

S. Massung and C. Zhai. 2016. Non-Native Text Analysis: A Survey. *The Journal of Natural Language Engineering*, 22(2):163–186. DOI: 10.1017/S1351324915000303 306

S. Massung, C. Zhai, and J.Hockenmaier. 2013. Structural Parse Tree Features for Text Representation. In *IEEE Seventh International Conference on Semantic Computing*, pp. 9–13. DOI: 10.1109/ICSC.2013.13 305

J. D. McAuliffe and D. M. Blei. 2008. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, eds., *Advances in Neural Information Processing Systems 20*, pages 121–128. Curran Associates, Inc. 386

G. J. McLachlan and T. Krishnan. 2008. *The EM algorithm and extensions*. 2nd ed. Wiley Series in Probability and Statistics. Hoboken, NJ., Wiley. http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+52983362X&sourceid=fbw_bibsonomy. DOI: 10.1002/9780470191613 466

Q. Mei. 2009. Contextual text mining. Ph.D. Dissertation, University of Illinois at Urbana-Champaign. 440

Q. Mei and C. Zhai. 2006. A mixture model for contextual text mining. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, ACM, pp. 649–655, New York. DOI: 10.1145/1150402.1150482. 423, 440

Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai. 2006. Generating semantic annotations for frequent patterns with context analysis. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, ACM, pp. 337–346, New York. DOI: 10.1145/1150402.1150441. 417

Q. Mei, C. Liu, H. Su, and C. Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proc.of the 15th international conference on World Wide Web (WWW '06)*. ACM. New York, 533–542. DOI: 10.1145/1135777 .1135857. 425, 426

Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. 2007a. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proc. of the 16th International Conference on World Wide Web*, WWW '07, ACM, pp. 171–180, New York. DOI: 10.1145/1242572.1242596. 410

Q. Mei, X. Shen, and C. Zhai. 2007b. Automatic labeling of multinomial topic models. In *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, August 12-15, 2007, pp. 490–499. DOI: 10.1145/1281192.1281246. 278

Q. Mei, D. Cai, D. Zhang, and C. Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, ACM, pp. 101–110, New York. DOI: 10.1145/1367497.1367512. 431, 432, 440

T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, September 26-30, 2010, pp. 1045–1048. http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html. 292

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, NV, pp. 3111–3119. 273, 292, 293

T. M. Mitchell. 1997. *Machine learning*. McGraw Hill Series in Computer Science. McGraw-Hill. 19, 37, 315

M.-F. Moens. 2006. *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)*. Springer-Verlag New York, Inc., Secaucus, NJ. DOI: 10.1007/978-1-4020-4993-4. 55

I. J. Myung. 2003. Tutorial on maximum likelihood estimation. *J. Math. Psychol.*, 47(1):90–100. DOI: 10.1016/S0022-2496(02)00028-7. 36

A. Nenkova and K. McKeown. 2012. A survey of text summarization techniques. In Charu C. Aggarwal and C. Zhai, eds, *Mining Text Data*, pp. 43–76. Springer US. DOI: 10.1007/978-1-4614-3223-4_3. 327

L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf. 216

B. Pang and L. Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135. DOI: 10.1561/1500000011 409, 410

J. M. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, ACM, pp. 275–281, New York, NY. DOI: 10.1145/290941.291008. 87, 90, 128, 427

J. R. Quinlan. 1986. Induction of Decision Trees. *Machine Learning*, 1(1):81–106. DOI: 10.1007/BF00116251. 301

D. R. Radev, H. Jing, M. Styś, and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938. DOI: 10.1016/j.ipm.2003.10.006. 327

D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, Association for Computational Linguistics, pages 248–256, Stroudsburg, PA. 386

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York. 324, 327

F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2010. *Recommender Systems Handbook*. 1st ed. Springer-Verlag New York, Inc. DOI: 10.1007/978-0-387-85820-3 235

C. J. Van Rijsbergen. 1979. *Information Retrieval*. 2nd ed. Butterworth-Heinemann, Newton, MA.

S. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146. 87

S. E. Robertson. 1997. Readings in Information Retrieval. In *The Probability Ranking Principle in IR*, San Francisco, CA, Morgan Kaufmann Publishers Inc. pp. 281–286. 84, 85

S. Robertson and H. Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4):333–389. DOI: 10.1561/1500000019. 88, 89, 129

S. Robertson, H. Zaragoza, and M. Taylor. 2004. Simple BM25 Extension to Multiple Weighted Fields. In *Proc. of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pp. 42–49. DOI: 10.1145/1031171.1031181 110

C. Roe. 2012. The growth of unstructured data: what to do with all those zettabytes? http://www.dataversity.net/the-growth-of-unstructured-data-what-are-we-going-to-do-with-all-those-zettabytes/. 3

R. Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*. 54

G. Salton. 1989. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley. 18

G. Salton and M. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill. 18

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620. 87

G. Salton and C. Buckley. 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297. 144

M. Sanderson. 2010. Test Collection Based Evaluation of Information Retrieval Systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375. 168, 188

M. Sanderson and W. B. Croft. 2012. The history of information retrieval research. *Proc. of the IEEE*, 100(Centennial-Issue):1444–1451, 2012. DOI: 10.1109/JPROC.2012.2189916. 85

S. Sarawagi. 2008. Information extraction. *Found. Trends databases*, 1(3):261–377. DOI: 10.1561/1900000003. 19, 55

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47. DOI: 10.1145/505282.505283. 315

G. Shani and A. Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, 2nd ed., pp. 257–297. Springer, New York, NY. DOI: 10.1007/978-0-387-85820-3_8. 235

F. Silvestri. 2010. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4:1–174. DOI: 10.1561/1500000013 144

A. Singhal, C. Buckley, and Mandar Mitra. 1996. Pivoted document length normalization. In *Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96,ACM, pp. 21–29, New York. DOI: 10.1145/243199.243206. 89, 106, 128

N. Smith. 2010. Text-driven forecasting. http://www.cs.cmu.edu/ñasmith/papers/smith.whitepaper10.pdf. 440

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proc. of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, ACM, pp. 623–632, New York. DOI: 10.1145/1321440.1321528. 185

K. Sparck Jones and P. Willett, eds. 1997. *Readings in Information Retrieval*. San Francisco, CA, Morgan Kaufmann Publishers Inc. 18, 85, 188

N. Spirin and J. Han. May 2012. Survey on Web Spam Detection: Principles and Algorithms. *SIGKDD Explor. Newsl.*, 13(2):50–64. DOI: 10.1145/2207243.2207252. 191

E. Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556. DOI: 10.1002/asi.v60:3 305

M. Steinbach, G. Karypis, and V. Kumar. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*. 296

J. Steinberger and K. Jezek. 2009. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275. 327

M. Steyvers and T. Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440. 385, 386

Y. Sun and J. Han. 2012. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers. DOI: 10.2200/S00433ED1V01Y201207DMK005. 440

I. Titov and R. McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proc. of the 17th International Conference on World Wide Web*, WWW '08, ACM, pp. 111–120, New York. DOI: 10.1145/1367497.1367513. 410

H. Turtle and W. B. Croft. 1990. Inference networks for document retrieval. In *Proc. of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '90, ACM, pp. 1–24, New York. DOI: 10.1145/96749.98006. 88

Princeton University. 2010. About wordnet. http://wordnet.princeton.edu. 395

C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths. 18

H. Wang, Yue Lu, and C. Zhai. 2010. Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach. In *Proc. of the 16th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, KDD '10, ACM, pp. 783–792, New York. DOI: 10.1145/1835804.1835903. 318, 327, 405, 406, 407, 408, 409, 410

H. Wang, Y. Lu, and C. Zhai. 2011. Latent Aspect Rating Analysis Without Aspect Keyword Supervision. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, ACM, pp. 618–626, New York. DOI: 10.1145/2020408.2020505. 318, 327, 405, 410

J. Weizenbaum. 1966. ELIZA—A Computer Program for the Study of Natural Language Communication Between Man and Machine, Communications of the ACM 9 (1): 36–45, DOI: 10.1145/265153.365168. 44

J. S. Whissell and C. L. A. Clarke. 2013. Effective Measures for Inter-document Similarity. In *Proc. of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, ACM, pages 1361–1370, New York. DOI: 10.1145/2505515.2505526. 279

R. W. White and R. A. Roth. 2009. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers. DOI: 10.2200/S00174ED1V01Y200901ICR003. 85

R. W. White, B. Kules, S. M. Drucker, and m.c. schraefel. 2006. Introduction. *Commun. ACM*, 49(4):36–39. DOI: 10.1145/1121949.1121978. 85

I. H. Witten, A. Moffat, and T. C. Bell. 1999. *Managing Gigabytes (2Nd Ed.): Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., San Francisco, CA. 18, 165

C.F J. Wu. 1983. On the convergence properties of the EM algorithm. *Ann. of stat.*, 95–103. 368

J. Xu and W. B. Croft. 1996. Query expansion using local and global document analysis. In *Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, ACM, pp. 4–11, New York. DOI: 10.1145/243199.243202. 144

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88. 315

C. Zhai. 1997. Exploiting context to identify lexical atoms—a statistical view of linguistic context. In *Proc. of the International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-97)*, pages 119–129. Rio de Janeiro, Brazil. 273, 291

C. Zhai. 2008. *Statistical Language Models for Information Retrieval*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers. DOI: 10.2200/S00158ED1V01Y200811HLT001. 55, 87, 128, 129

C. Zhai and J. Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, ACM, pp. 403–410, New York. DOI: 10.1145/502585.502654. 143, 466, 473

C. Zhai and J. Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214. 475

C. Zhai, P. Jansen, E. Stoica, N. Grot, and D. A. Evans. 1998. Threshold Calibration in CLARIT Adaptive Filtering. In *Proc. of Seventh Text REtrieval Conference (TREC-7)*, pp. 149–156. 227

C. Zhai, P. Jansen, and D. A. Evans. 2000. Exploration of a heuristic approach to threshold learning in adaptive filtering. In *SIGIR*, ACM, pp. 360–362. DOI: 10.1145/345508.345652. 235

C. Zhai, A. Velivelli, and B. Yu. 2004. A cross-collection mixture model for comparative text mining. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, ACM, pp. 743–748, New York. DOI: 10.1145/1014052.1014150. 423

D. Zhang, C. Zhai, J. Han, A. Srivastava, and N. Oza. 2009. Topic modeling for OLAP on multidimensional text databases: topic cube and its applications. *Stat. Anal. Data Min.* 2, 5–6 (December 2009), 378–395. DOI: 10.1002/sam.v2.5/6. 440

J. Zhu, A. Ahmed, and E. P. Xing. 2009. Medlda: Maximum margin supervised topic models for regression and classification. In *Proc. of the 26th Annual International Conference on Machine Learning*, ICML '09, ACM, pp. 1257–1264, New York. DOI: 10.1145/1553374.1553535. 386

G. K. Zipf. 1949. *Human Behavior and the Principle of Least-Effort*. Cambridge, MA, Addison-Wesley, 162

索　引

索引中的页码为英文原版书的页码，与书中页边标注的页码一致。

A

Absolute discounting(绝对折扣)，130
Abstract text summarization(抽象式文本摘要)，318，321-324
Access modes(获取模式)，73-76
Accuracy in search engine evaluation(搜索引擎评价的精确性)，168
Ad hoc information needs(临时信息需求)，8-9
Ad hoc retrieval(临时检索)，75-76
Add-1 smoothing(加一平滑)，130，464
Adjacency matrices(邻接矩阵)，207-208
Advertising(广告)，opinion mining for(观点挖掘)，393
Agglomerative clustering(凝聚聚类)，277，280-282，290
Aggregating(聚集)
  opinions(观点)，393
  scores(分数)，234
All-vs-all(AVA)method(AVA 方法)，313
Ambiguity(歧义)
  full structure parsing(完整结构分析)，43
  LARA(潜在方面评分分析)，406
  NLP(自然语言处理)，40-41，44
  One-vs-all method，(One-vs-ALL 方法) 313
  text Retrieval vs. database Retrieval(文本检索 vs. 数据库检索)，80
  topics(主题)，335，337
Analyzers in MeTA toolkit(MeTA 工具中的分析器)，61-64，453
analyzers:: filters namespace (**analyzers:: filters** 命名空间)，64
analyzers:: tokenizers namespace (**analyzers::tokenizers** 命名空间)，64
Anaphora resolution in natural language processing (自然语言处理中的指代消解)，41
Anchor text in web searches(网络搜索中的锚文本)，201
Architecture(架构)
  GFS，194-195
  MeTA toolkit(MeTA 工具)，60-61
  unified systems(统一系统)，452-453
Art retrieval models(最先进的检索模型)，111
Aspect opinion analysis(方面观点分析)，325-326
Associations，word(关联，词)，见 Word association mining
Authority pages in web searches(网络搜索中的权威页面)，202，207
Automatic evaluation in text clustering(文本聚类自动评价)，294
AVA (all-vs-all)method(AVA 方法)，313。
Average-link document clustering(平均链接文档聚类)，282
Average precision(平均准确率)
  ranked lists evaluation(排序列表评价)，175，177-180
  search engine evaluation(搜索引擎评价)，184
Axiomatic thinking(公理化思想)，88

B

Background models(背景模型)
  mining topics from text(文本主题挖掘)，345-351
  mixture model estimation(混合模型的估计)，351-353
  PLSA(概率潜在语义分析)，370-372
Background words(背景词)
  mixture models(混合模型)，141，351-353
  PLSA(概率潜在语义分析)，368-369，372
Bag-of-words(词袋)
  frequency analysis(频率分析)，69
  paradigmatic relations(聚合关系)，256
  text information systems(文本信息系统)，10
  text representation(文本表示)，88-90
  vector space model(向量空间模型)，93，109
  web searches(网络搜索)，215

Bar-Hillel report(Bar-Hillel 报告), 42
Baseline accuracy in text categorization(文本分类中的基线精确率), 314
Bayes, Thomas(贝叶斯, 托马斯), 25
Bayes' rule(贝叶斯法则)
  EM algorithm(EM 算法), 361-363, 373-374
  formula(公式), 25-26
  LDA(潜在狄利克雷分布), 383
Bayesian inference(贝叶斯推理)
  EM algorithm(EM 算法), 361-362
  PLSA(概率潜在语义分析), 379, 382
Bayesian parameter estimation(贝叶斯参数估计)
  formula(公式), 458
  overfitting problem(过拟合问题), 28-30
  unigram language model(一元语言模型), 341, 359
Bayesian smoothing(贝叶斯平滑), 125
Bayesian statistics(贝叶斯统计)
  binomial estimation and beta distribution(二项估计和 Beta 分布), 457-459
  Dirichlet distribution(狄利克雷分布), 461-463
  LDA(潜在狄利克雷分布), 382
  multinomial distribution(多项分布), 460-461
  multinomial parameters(多项参数), 463-464
  Naïve Bayes algorithm(朴素贝叶斯算法), 309-312
  pseudo counts, smoothing, and setting hyper-parameters(伪计数, 平滑, 设置超参数), 459-460
Berkeley study(伯克利研究), 3
Bernouli distribution(伯努利分布), 26
Beta distribution(Beta 分布), 457-459
Beta-gamma threshold learning(β-γ 阈值学习), 227-228
Bias, clustering(偏置, 聚类), 276
Big text data(大文本数据), 5-6
Bigram language model(二元语言模型)
  abstractive summarization(抽象式摘要), 323
  Brown clustering(布朗聚类), 290
Bigrams(二元词项)
  frequency analysis(词频分析), 68
  sentiment classification(情感分类), 394-395
  text categorization(文本分类), 305
  word tokenizer(分词器), 149
Binary classification(二元分类)
  content-based recommendation(基于内容的推荐), 223
  text categorization(文本分类), 303
Binary hidden variables in EM algorithm(EM 算法中的二元隐变量), 362-364, 366, 368, 467
Binary logistic regression(二元逻辑回归), 397
Binomial distribution(二项分布), 26-27
Binomial estimation (二项估计), 457-459
Bit vector representation(位向量表示), 93-97
Bitwise compression(按位压缩), 159-160
Blind feedback(盲目反馈), 133, 135
Block compression(块压缩), 161-162
Block world project(Block world 项目), 42
BM25 model(BM25 模型)
  description(描述), 88
  document clustering(文档聚类), 279
  document length normalization(文档长度规范化), 108-109
  link analysis(链接分析), 201
  Okapi(Okapi 算法), 89, 108
  popularity(流行度), 90
  probabilistic retrieval models(概率检索模型), 111
BM25-F model(BM25-F 模型), 109
BM25 score(BM25 分数)
  paradigmatic relations(聚合关系), 258-261
  syntagmatic relations(组合关系), 270
  web search ranking(网络搜索排序), 210
BM25 TF transformation(BM25 词频变换)
  description(描述), 104-105
  paradigmatic relations(聚合关系), 258-259
BM25+ model(BM25+模型), 88, 110
Breadth-first searches(广度优先搜索), 193
Breakeven point precision(平滑转折点准确率), 189
Brown clustering(布朗聚类), 278, 288-291
Browsing(浏览)
  multimode interactive access(多模式交互访问), 76-78
  pull access mode(拉取获取模式), 73-75
  support for(支持), 445
  text information systems(文本信息系统), 9
  web searches(网络搜索), 214
  word associations(词关联), 252
Business intelligence(商业智能)
  opinion mining(观点挖掘), 393
  text data analysis(文本数据分析), 243

C

C++ language(C++语言), 16, 58

Caching(缓存)
DBLRU(双桶 LRU), 164-165
LRU, 163-164
MeTA toolkit(MeTA 工具), 60
search engine implementation(搜索引擎实现), 148, 162-165
Categories(类别)
categorical distributions(分类分布), 460-461
sentiment classification(情感分类), 394, 396-397
text information systems(文本信息系统), 11-12
Causal topic mining(因果主题挖掘), 433-437
Centroid vectors(质心向量), 136-137
Centroids in document clustering(文档聚类中的质心), 282-284
CG(cumulative gain)in NDCG(NDCG 中的累计收益), 181-182
character_tokenizer tokenizer(**character_tokenizer** 切分器), 61
Citations(引用), 202
Classes(类)
Brown clustering(布朗聚类), 289
categories(类别), 11-12
sentiment(情感), 393-396
Classification(分类)
machine learning(机器学习), 34-36
NLP(自然语言处理), 43-44
Classifiers in text categorization(文本分类器中的分类器), 302-303
classify command(**classify** 命令), 57
Cleaning HTML files(清理 HTML 文件), 218-219
Clickthroughs(点击)
probabilistic retrieval model(概率检索模型), 111-113
web searches(网络搜索), 201
Clustering bias(聚类偏置), 276
Clusters and clustering(簇和聚类)
joint analysis(联合分析), 416
sentiment classification(情感分类), 395
text(文本), 见 Text clustering
Coherence in text clustering(文本聚类的凝聚度), 294-295
Coin flips, binomial distribution for(硬币投掷, 二项分布)26-27
Cold start problem(冷启动问题), 230
Collaborative filtering(协同过滤), 221, 229-233
Collapsed Gibbs sampling(吉布斯采样), 383
Collect function(**Collect** 函数), 197
Collection language model(文档集语言模型)
KL-divergence(KL-散度), 474
smoothing methods(平滑方法), 121-126
Common form of retrieval models(检索模型的常见形式), 88-90
Common sense knowledge in NLP(NLP 中的常识知识), 40
Common words(常用词)
background language model(背景语言模型), 346-347, 350-351
feedback(反馈), 141, 143
filtering(过滤), 54
mixture models(混合模型), 352-353, 355-356
unigram language model(一元语言模型), 345-346
vector space retrieval models(向量空间检索模型), 99, 109
Compact clusters(紧凑簇), 281
Compare operator(比较运算符), 450, 452
Complete data for EM algorithm(EM 算法完整数据), 467-468
Complete-link document clustering(全链文档聚类), 281-282
Component models(组件模型)
background language models(背景语言模型), 345, 347-350
CPLSA(上下文潜在语义分析), 421
description(描述), 143
EM algorithm(EM 算法), 359
mixture models(混合模型), 355-356, 358-359
PLSA(概率潜在语义分析), 370-373
Compression(压缩)
bitwise(位), 159-160
block(块), 161-162
overview(概述), 158-159
search engines(搜索引擎), 148
text representation(文本表示), 48-49
Compression ratio(压缩比), 160-161
Concepts in vector space model(向量空间模型中的概念), 92
Conceptual framework in text information systems (文本信息系统中的概念框架), 10-13
Conditional entropy(条件熵)
information theory(信息论), 33
syntagmatic relations(组合关系), 261-264, 270

Conditional probabilities(条件概率)
Bayes'rule(贝叶斯法则), 25-26
overview(概述), 23-25
Configuration files(配置文件), 57-58
Confusion matrices(混淆矩阵), 314-315
Constraints in PLSA(PLSA 中的约束), 373
Content analysis modules(内容分析模块), 10-11
Content-based filtering(基于内容的过滤), 221-229
Content in opinion mining(观点挖掘内容), 390-392
Context(上下文)
Brown clustering(布朗聚类), 290
non-text data(非文本数据), 249
opinion mining(观点挖掘), 390-392
paradigmatic relations(聚合关系), 253-258
social networks as(社交网络), 428-433
syntagmatic relations(组合关系), 261-262
text mining(文本挖掘), 417-419
time series(时间序列), 433-439
Context variables in topic analysis(主题分析中的上下文变量), 330
Contextual Probabilistic Latent Semantic Analysis (CPLSA, 上下文概率潜在语义分析), 419-428
Continuous distributions(连续分布)
Bayesian parameter estimation(贝叶斯参数估计), 28
description(描述), 22
Co-occurrences in mutual information(互信息中的共现), 267-268
Corpus input formats in MeTA toolkit(MeTA 工具语料输入格式), 60-61
corpusname.dat file(**corpusname.dat** 文件), 60
corpusname.dat.gz file(**corpusname.dat.gz** 文件), 60
corpusname.dat.labels file(**corpusname.dat.labels**文件), 60
corpusname.dat.labels.gz file(**orpusname.dat.labels.gz** 文件), 60
Correlations(相关)
mutual information(互信息), 270
syntagmatic relations(组合关系), 253-254
text-based forecasting(基于文本的预测), 248
time series context(时间序列上下文), 437
Cosine similarity(余弦相似度)
document clustering(文档聚类), 279-280
extractive summarization(抽取摘要), 321
text summarization(文本摘要), 325
vector measurement(向量度量), 222, 232
Coverage(覆盖率)
CPLSA, 420-422, 425-426
LDA, 380-381
topic analysis(主题分析), 332-333
CPLSA(Contextual Probabilistic Latent Semantic Analysis)(上下文概率潜在语义分析), 419-428
Cranfield evaluation methodology(Cranfield 评价方法), 168-170
Crawlers(爬虫)
domains(领域), 218
dynamic content(动态内容), 217
languages for(语言), 216-217
web searches(网络搜索), 192-194
Cross validation in text categorization(文本分类中的交叉验证), 314
Cumulative gain(CG)in NDCG(NDCG 中的累计增益(CG)), 181-182
Current technology(当前技术), 5

D

Data-driven social science research, opinion mining for(数据驱动的社会科学研究, 观点挖掘), 393
Data mining(数据挖掘)
joint analysis(联合分析), 413-415
probabilistic retrieval model algorithms(概率检索模型算法), 117
text data analysis(文本数据分析), 245-246
Data types in text analysis(文本分析中的数据类型), 449-450
Data-User-Service Triangle(数据-用户-服务三角形), 213-214
Database retrieval(数据库检索), 80-82
DBLRU(Double Barrel Least-Recently Used)caches(双桶最近最少使用缓存), 164-165
DCG(discounted cumulative gain)(折扣累计收益), 182-183
Decision boundaries for linear classifiers(线性分类器决策边界), 311-312
Decision modules in content-based filtering(基于内容过滤中的决策模块), 225

Decision support, opinion mining for(决策支持，观点挖掘)，393
Deep analysis in natural language processing(自然语言处理深层分析)，43-45
Delta bitwise compression(Delta 位压缩)，160
Dendrograms(树状图)，280-281
Denial of service from crawlers(拒绝爬虫访问服务)，193
Dependency parsers(依存句法分析)，323
Dependent random variables(依赖的随机变量)，25
Design philosophy, MeTA(设计原则，MeTA)，58-59
Development sets for text categorization(文本分类的开发集)，314
Dirichlet distribution(狄利克雷分布)，461-463
Dirichlet prior smoothing(狄利克雷先验平滑)
  KL-divergence(KL-散度)，475
  probabilistic retrieval models(概率检索模型)，125-127
Disaster response(灾害响应)，243-244
Discounted cumulative gain(DCG，折扣累计收益)，182-183
Discourse analysis in NLP(自然语言处理之篇章分析)，40
Discrete distributions(离散分布)
  Bayesian parameter estimation(贝叶斯参数估计)，29
  description(描述)，22
Discriminative classifiers(判别式分类器)，302
Distances in clusters(簇间距离)，281
Distinguishing categories(区分类别)，301-302
Divergence-from-randomness models(随机偏差模型)，87，111
Divisive clustering(分裂聚类)，277
Document-at-a-time ranking(逐个文档排序)，155
Document clustering(文档聚类)，277
  agglomerative hierarchical(凝聚层次)，280-282
  K-means(*K*-均值)，282-284
  overview(概述)，279-280
Document frequency(文档频率)
  bag-of-words representation(词袋表示)，89
  vector space model(向量空间模型)，99-100
Document IDs(文档 ID)
  compression(压缩)，158-159
  inverted indexes(倒排索引)，152
  tokenizers(分词器)，149
Document language model(文档语言模型)，118-123
Document length(文档长度)
  bag-of-words representation(词袋表示)，89
  vector space model(向量空间模型)，105-108
Documents(文档)
  filters(过滤器)，155-156
  ranking vs. selecting(排序 vs. 选择)，82-84
  tokenizing(分词)，148-150
  vectors(向量)，92-96
  views in multimode interactive access(多模式交互访问视图)，77
Domains, crawling(域名，爬虫)，218
Dot products(点积)
  document length normalization(文档长度规范化)，109
  linear classifiers(线性分类器)，311
  paradigmatic relations(聚合关系)，257-258
  vector space model(向量空间模型)，93-95，98
Double Barrel Least-Recently Used(DBLRU)caches(双桶最近最少使用缓存)，164-165
Dynamic coefficient interpolation in smoothing methods(平滑方法中的动态系数插值)，125
Dynamically generated content and crawlers(动态生成内容和爬虫)，217

E

E step in EM algorithm(EM 算法中的 E 步)，362-368，373-377，465，469
E-discovery(electronic discovery)(电子发现)，326
Edit features in text categorization(文本分类中的编辑特征)，306
Effectiveness in search engine evaluation(搜索引擎评价的有效性)，168
Efficiency(效率)
  database data retrieval(数据库的数据检索)，81-82
  search engine evaluation(搜索引擎评价)，168
Electronic discovery(E-discovery)(电子发现)，326
Eliza project(Eliza 项目)，42，44-45
EM algorithm(EM 算法). 见 Expectation maximization(EM)algorithm
Email counts (邮件计数)3
Emotion analysis(情绪分析)，394
Empirically defined problems(根据经验定义的问题)，82

Enron email dataset(安然电子邮件数据集), 326
Entity-relation re-creation(实体关系再造), 47
Entropy(熵)
  information theory(信息论), 31-33
  KL-divergence(KL-散度), 139, 474
  mutual information(互信息), 264-265
  PMI(点互信息), 288
  skewed distributions(偏态分布), 158
  syntagmatic relations(组合关系), 261-264, 270
Evaluation, search engine.(评价, 搜索引擎), 见 Search engine evaluation
Events(事件)
  CPLSA(上下文概率潜在语义分析), 426-427
  probability(概率), 21-23
Exhaustivity in sentiment classification(情感分类中的充分性), 396
Expectation-maximization(EM) algorithm(期望最大化算法)
  CPLSA(上下文概率潜在语义分析), 422
  general procedure(一般过程), 469-471
  incomplete vs. complete data(不完整 vs. 完整数据), 467-468
  K-means(*K* -均值), 282-283
  KL-divergence(KL-散度), 476
  lower bound of likelihood(似然下界), 468-469
  MAP estimate(MAP 估计), 378-379
  mining topics from text(文本主题挖掘), 359-368
  mixture unigram language model(混合一元语言模型), 466
  MLE, 466-467
  network supervised topic models(网络监督主题模型), 431
  overview(概述), 465-466
  PLSA, 373-377
Expected overlap of words in paradigmatic relations(聚合关系中词语重叠期望), 257-258
Expected value in Beta distribution(Beta 分布期望值), 458
Exploration-exploitation tradeoff in content-based filtering(基于内容过滤的探索利用平衡), 227
Extractive summarization(抽取摘要), 318-321

F

F measure(F 度量)
  ranked lists evaluation(排序列表的评价), 179
  set retrieval evaluation(集合检索的评价), 172-173
F-test for time series context(时间序列上下文的 F 值检验), 437
F_1 score(F1 参数)
  text categorization(文本分类), 314
  text summarization(文本摘要), 324
Fault tolerance in Google File System(谷歌文件系统容错性), 195
Feature generation for tokenizers(分词器特征生成), 150
Features for text categorization(文本分类特征), 304-307
Feedback(反馈)
  content-based filtering(基于内容的过滤), 225
  KL-divergence(KL-散度), 475-476
  language models(语言模型), 138-144
  overview(概述), 133-135
  search engines(搜索引擎), 147, 157-158
  vector space model(向量空间模型), 135-138
  web searches(网络搜索), 201
Feedback documents in unigram language model(一元语言模型反馈文档), 466
Feelings(感受), 见 Sentiment analysis
fetch_docs function(**fetch_docs** 函数), 154
file_corpus input format(**file_corpus** 输入格式), 60
Files in Google File System(谷歌文件系统中的文件), 194-195
Filter chains for tokenization(分词过滤链), 61-64
Filters(过滤器)
  content-based(基于内容), 221-229
  documents(文档), 155-156
  recommender systems(推荐系统), 见 Recommender systems
  text information systems(文本信息系统), 11
  unigram language models(一元语言模型), 54
Focused crawling(聚焦爬取), 193
forward_index indexes(**forward_index** 索引), 60-61
Forward indexes(前向索引, 正向索引)
  description(描述), 153
  k-nearest neighbors algorithm(*k*-近邻算法), 308
Frame of reference encoding(参照框架编码), 162
Frequency and frequency counts(频率和频率计数)
  bag-of-words representation(词袋表示), 89-90

MapReduce，197
METAanalyses(META分析)，68-70
term(词项)，97-98
vector space model(向量空间模型)，99-100
Frequency transformation in paradigmatic relations(聚合关系中的词频变换)，258-259
Full structure parsing(完整结构解析)，43

G

G-means algorithm(G-均值算法)，294
Gain in search engine evaluation(搜索引擎评价中的收益)，181-183
Gamma bitwise compression(γ位压缩)，160
Gamma function(γ函数)，457
Gaussian distribution(高斯分布)，22，404-405
General EM algorithm(通用EM算法)，431
Generation-based text summarization(基于生成的文本摘要)，318
Generative classifiers(生成式分类器)，309
Generative models(生成模型)
background language model(背景语言模型)，346-347，349
CPLSA，419，421
description(描述)，30，36，50
LARA，403，405-406
LDA，381
log-likelihood functions(对数似然函数)，343-344，384
mining topics from text(文本主题挖掘)，347
n-gram models(*n*元模型)，289
network supervised topic models(网络监督主题模型)，428-430
PLSA，370-371，380
topics(主题)，338-340
unigram language model(一元语言模型)，341
Geographical networks(地理网络)，428
Geometric mean average precision(几何平均准确率)(gMAP)，179
GFS(Google File System)(谷歌文件系统)，194-195
Gibbs sampling(吉布斯采样)，383
Google File System(GFS，谷歌文件系统)，194-195
Google PageRank(谷歌PageRank算法)，202-206
Grammar learning(语法学习)，252
Grammatical parse trees(语法解析树)，305-307
Granger test(格兰杰测试)，434，437
Graph Mining(图挖掘)，49
gz_corpus input format(**gz_corpus**输入格式)，60

H

Hidden variables(隐变量)
EM algorithm(EM算法)，362-364，366，368，373-376，465，467
LARA，403
Hierarchical clustering(层次聚类)280-282
High-level syntactic features(高级句法特征)，305-306
Hill-climbing algorithm，EM(爬山算法，EM)，360，366-367，465
HITS algorithm(HITS算法)，206-208
HTML files，cleaning(HTML文件，清理)，218-219
Hub pages in web searches(网络搜索中的中心页面)，202，207-208
Humans(人)
joint analysis(联合分析)，413-415
NLP，48
opinion mining(观点挖掘)，见Opinion mining as subjective sensors，244-246
unified systems(统一系统)，445-448
Hyperparameters(超参数)
Beta distribution(Beta分布)，458-460
Dirichlet distribution(狄利克雷分布)，461，463

I

ICU(International Components for Unicode)(Unicode编码国际组件)，61
Icu_filter filter(**Icu_filter**过滤器)，61
Icu_tokenizer tokenizer(**Icu_tokenizer**分词器)，61
IDF(inverse document frequency)(逆文档频率)
Dirichlet prior smoothing(狄利克雷先验平滑)，126
paradigmatic relations(聚合关系)，258-260
query likelihood retrieval model(查询似然检索模型)，122
vector space model(向量空间模型)，99-101
Illinois NLP Curator toolkit(Illinois NLP Curator工具箱)，64
Impact(影响)
CPLSA，426-427
time series context(时间序列上下文)，437
Implicit feedback(隐式反馈)，134-135

Incomplete data in EM algorithm(EM 算法中的不完整数据), 467-468
Incremental crawling(增量爬取), 193
Independent random variables(独立随机变量), 25
Index sharding(索引分片), 156-157
Indexes(索引)
  compressed(压缩), 158-162
  forward(前向), 153, 308
  k-nearest neighbors algorithm(*k*-近邻算法), 308
  MapReduce, 198-199
  MeTA toolkit(MeTA 工具箱), 60-61, 453-455
  search engine implementation(搜索引擎实现), 150-153
  search engines(搜索引擎), 147, 150-153
  text categorization(文本分类), 314
  web searches(网络搜索), 194-200
Indirect citations in Web searches(网络搜索中的间接引用), 202
Indirect opinions(间接观点), 391-392
Indri/Lemur search engine toolkit(Indri/Lemur 搜索引擎工具), 64
Inferences(推断)
  NLP, 41
  probabilistic(概率), 88
  real world properties(现实世界的属性), 248
Inferred opinions(推断的观点), 391-392
Information access in text information systems(文本信息系统中的信息获取), 7
Information extraction(信息抽取)
  NLP, 43
  text information systems(文本信息系统), 9, 12
Information retrieval(IR)systems(信息检索(IR)系统), 6
  evaluation metrics(评价指标), 324-325
  implementation(实现), 见 Search engine implementation
  text data access(文本数据获取), 79
Information theory(信息论), 31-34
Initial values in EM algorithm(EM 算法中的初始值), 466
Initialization modules in content-based filtering(基于内容过滤的初始化模块), 224-225
Inlink counts in PageRank(PageRank 中的链入计数), 203
Instance-based classifiers(基于实例的分类器), 302
Instructor reader category(指导者读者类别), 16-17
Integer compression(整数压缩), 158-162
Integration of information access in web searches(网络搜索中信息获取集成), 213
Integrity in text data access(文本数据获取完整性), 81
Interactive access, multimode(交互获取, 多模式), 76-78
Interactive task support in web searches(网络搜索中的交互式任务支持), 216
International Components for Unicode(ICU)(Unicode 编码的国际组件), 61
Interpolation for smoothing methods(插值平滑法), 125-126
Interpret operator(解释因子), 450-452
Intersection operator(交叉算子), 449-450
Intrusion detection(入侵检测), 271-273
Inverse document frequency(IDF)(逆文档频率)
  Dirichlet prior smoothing(狄利克雷先验平滑), 126
  paradigmatic relations(聚合关系), 258-260
  query likelihood retrieval model(查询似然检索模型), 122
  vector space model(向量空间模型), 99-101
Inverse user frequency(IUF, 逆用户频率), 232
inverted_index indexes(**inverted_index** 索引), 60
Inverted index chunks(倒排索引的块), 156-157
Inverted indexes(倒排索引, 反向索引)
  compression(压缩), 158
  k-nearest neighbors algorithm(*k*-近邻算法), 308
  MapReduce, 198-199
  search engines(搜索引擎), 150-153
IR(information retrieval)systems(信息检索系统), 6
  evaluation metrics(评价指标), 324-325
  implementation(实现), 见 Search engine implementation
  text data access(文本数据获取), 79
Iterative algorithms for PageRank(PageRank 迭代算法), 205-206
Iterative Causal Topic Modeling(迭代因果主题建模), 434-435
IUF(inverse user frequency)(逆用户频率), 232

J

Jaccard similarity(杰卡德相似度), 280
Jelinek-Mercer smoothing (Jelinek-Mercer 平滑),

123-126
Joint analysis of text and structured data(文本与结构化数据的联合分析), 413
contextual text mining, (上下文相关文本挖掘) 417-419
CPLSA, 419-428
introduction(简介), 413-415
social networks as context(社交网络上下文), 428-433
time series context(时间序列上下文), 433-439.
Joint distributions for mutual information(互信息联合分布), 266-268
Joint probabilities(联合概率), 23-25

K

K-means document clustering(*K*-均值文档聚类), 282-284
k-nearest neighbors (*k*-NN) algorithm (*k*-近邻算法), 307-309
Kernel trick for linear classifiers(线性分类器的核方法), 312
Key-value pairs in MapReduce(MapReduce 中的键-值对), 195-198
KL-divergence(KL-散度)
Dirichlet prior smoothing(狄利克雷先验平滑), 475
EM algorithm(EM 算法), 468
feedback(反馈), 139-140
mutual information(互信息), 266
query model(查询模型), 475-476
retrieval(检索), 473-474
Knowledge acquisition in text information systems (文本信息系统的知识获取), 8-9
Knowledge discovery in text summarization(文本摘要中的知识挖掘), 326
Knowledge Graph(知识图谱), 215
Knowledge provenance in unified systems(统一系统中的知识来源), 447
Known item searches in ranked lists evaluation(有序列表评价中的已知项目搜索), 179
Kolmogorov axioms(柯尔莫哥洛夫公理), 22-23
Kullback-Leibler divergence retrieval model (KL-散度检索模型). 见 KL-divergence

L

Lagrange Multiplier approach(拉格朗日乘数法)
EM algorithm(EM 算法), 467, 470
unigram language model(一元语言模型), 344
Language models(语言模型)
feedback in(反馈), 138-144
in probabilistic retrieval model(概率检索模型), 87, 111, 117
Latent Aspect Rating Analysis(LARA, 潜在方面评分分析), 400-409
Latent Dirichlet Allocation(LDA)(潜在狄利克雷分布), 377-383
Latent Rating Regression(潜在评分回归), 402-405
Lazy learners in text categorization(文本分类中的懒人学习), 302
Learners(学习器)
search engines(搜索引擎), 147
text categorization(文本分类), 302
Learning modules in content-based filtering (基于内容过滤系统中的学习模块), 224-225
Least-Recently Used(LRU) caches(最近最少使用缓存), 163-164
length_filter filter (**length_filter** 过滤器), 61
Length normalization(长度规范化)
document length(文档长度), 105-108
query likelihood retrieval model(查询似然检索模型), 122
Lexical analysis in NLP(NLP 词法分析), 39-40
Lexicons for inverted indexes(倒排索引词汇表), 150-152
LIBLINEAR algorithm(LIBLINEAR 算法), 58
libsvm_analyzer analyzer(**libsvm_analyzer** 分析器), 62
libsvm_corpus file(**libsvm_corpus** 文件), 61
LIBSVM package(LIBSVM 程序包), 58, 64
Lifelong learning in web searches(网络搜索终身学习), 213
Likelihood and likelihood function(似然与似然函数)
background language model(背景语言模型), 349-351
EM algorithm(EM 算法), 362-363, 367-368, 376, 465-469
LARA, 405
LDA, 378, 381-382
marginal(边缘), 28
mixture model behavior(混合模型行为), 354-357

MLE, 27
network supervised topic models(网络监督主题模型), 428-431
PLSA, 372-374
unigram language model(一元语言模型), 342-344
line_corpus input format(**line_corpus** 输入格式), 60
Linear classifiers in text categorization(线性文本分类器), 311-313
Linear interpolation in Jelinek-Mercer smoothing (Jelinek-Mercer 平滑中的线性插值), 124
Linearly separable data points in linear classifiers (线性分类器中的线性可分数据点), 312
Link analysis(链接分析)
HITS(HITS 算法), 206-208
overview(概述), 200-202
PageRank(PageRank 算法), 202-206
list_filter filter(**list_filter** 过滤器), 62
Local maxima(局部最大点), 360, 363, 367-368, 465
Log-likelihood function(对数似然函数)
EM algorithm(EM 算法), 365-366, 466-467
feedback(反馈), 142-143
unigram language model(一元语言模型), 343-344
Logarithm transformation(对数变换), 103-104
Logarithms in probabilistic retrieval model(概率检索模型中的对数), 118, 122
Logic-based approach in NLP(基于逻辑的自然语言处理方法), 42
Logical predicates in NLP(自然语言处理逻辑谓词), 49-50
Logistic regression in sentiment classification(基于逻辑回归的情感分类), 396-400
Long-range jumps in multimode interactive access (多模式交互获取中的长距离跳跃), 77
Long-term needs in push access mode(推送式信息获取中的长期需求), 75
Low-level lexical features in text categorization(文本分类中底层词汇特征), 305
Lower bound of likelihood in EM algorithm(EM 算法的似然下界), 468-469
LRU(Least-Recently Used) caches(最近最少使用缓存), 163-164
Lucene search engine toolkit(lucene 搜索引擎工具), 64

M

M step(M 步)
EM algorithm(EM 算法), 361-368, 373-377, 465, 469-470
MAP estimate(MAP 估计), 379
network supervised topic models(网络监督主题模型), 431
Machine-generated data(机器生成数据), 6
Machine learning(机器学习)
overview(概述), 34-36
sentiment classification methods(情感分类方法), 396
statistical(统计), 10
text categorization(文本分类), 301
web search algorithms(网络搜索算法), 201
web search ranking(网络搜索排序), 208-212
Machine translation(机器翻译), 42, 44-45
Magazine output(杂志输出), 3
Manual evaluation for text clustering(文本聚类人工评价), 294
map function(**map** 函数), 195-198
MAP(Maximum a Posteriori) estimate(最大后验估计)
Bayesian parameter estimation(贝叶斯参数估计), 29
LARA, 404-405
PLSA, 378-379
word association mining(词关联挖掘), 271-273
MAP(mean average precision)(算术平均准确率), 178-180
Map Reduce paradigm(MapReduce 范式), 157
MapReduce framework(MapReduce 框架), 194-200
Maps in multimode interactive access(多模式交互获取中的地图), 76-77
Marginal probabilities(边缘概率)
Bayesian parameter estimation(贝叶斯参数估计), 29
mutual information(互信息), 267
Market research, opinion mining for, (市场研究, 观点挖掘)393
Massung, Sean, biography(生平), 490
Matrices(矩阵)
adjacency(邻接), 207-208
PageRank, 204-208

text categorization(文本分类)，314-315
transition(转移)，204
Matrix multiplication in PageRank(PageRank 矩阵乘法)，205
Maximal marginal relevance(MMR)reranking(最大边缘相关重排序)
extractive summarization(抽取摘要)，320-321
topic analysis(主题分析)，333
Maximization algorithm for document clustering(文档聚类中的最大化算法)，282
Maximum a Posteriori(MAP)estimate(最大后验估计)
Bayesian parameter estimation(贝叶斯参数估计)，29
LARA，404-405
PLSA，378-379
word association mining(词关联挖掘)，271-273
Maximum likelihood estimation(MLE，最大似然估计)
background language model(背景语言模型)，346，350
Brown clustering(布朗聚类)，289
Dirichlet prior smoothing(狄利克雷先验平滑)，125-126
EM algorithm(EM 算法)，359-368，466-467
feedback(反馈)，141-143
generative models(生成模型)，339
Jelinek-Mercer smoothing (Jelinek-Mercer 平滑)，124
KL-divergence(KL-散度)，475-476
LARA，404
LDA，382
mixture model behavior(混合模型行为)，354-359
mixture model estimation(混合模型估计)，352-353
multinomial distribution(多项分布)，463
mutual information(互信息)，268-269
overview(概述)，27-28
PLSA，372-373，378
query likelihood retrieval model(查询似然检索模型)，118-119
term clustering(词项聚类)，286
unigram language models(一元语言模型)，52-53，341-345
web search ranking(网络搜索排序)，210
Mean average precision(MAP，算术平均准确率)，178-180
Mean reciprocal rank(MRR)(排序倒数均值)，180
Measurements in search engine evaluation(搜索引擎评价的度量)，168
Memory-based approach in collaborative filtering(基于记忆的协同过滤)，230
META toolkit(META 工具箱)
architecture(架构)，60-61
classification algorithms(分类算法)，307
design philosophy(设计原则)，58-59
exercises(练习)，65-70
overview(概述)，57-58
related toolkits(相关工具箱)，64-65
setting up(设置)，59-60
text categorization(文本分类)，314-315
tokenization(分词)，61-64
as unified system(统一系统)，453-455
Metadata(元数据)
classification algorithms(分类算法)，307
contextual text mining(上下文文本挖掘)，417
networks from(网络)，428
text data analysis(文本数据分析)，249
topic analysis(主题分析)，330
Mining(挖掘)
contextual(上下文)，417-419
demand for(需求)，4-5
graph(图)，49
joint analysis(联合分析)，413-419
opinion(观点)，见 Opinion mining; Sentiment analysis
probabilistic retrieval model(概率检索模型)，117
tasks(任务)，246-250
toolkits(工具箱)，64
topic analysis(主题分析)，330-331
word association(词关联)，见 Word association mining
Mining topics from text(文本主题挖掘)，340
background language model(背景语言模型)，345-351
expectation-maximization(期望最大化)，359-368
joint analysis(联合分析)，416
mixture model behavior(混合模型行为)，353-359
mixture model estimation(混合模型估计)，351-353
unigram language model(一元语言模型)，341-345
Mixture models(混合模型)
behavior(行为)，353-359

EM algorithm(EM 算法), 466
estimation(估计), 351-353
feedback(反馈), 140-142, 157
mining topics from text(文本主题挖掘), 346-351
MLE(最大似然估计), 见 Maximum likelihood estimation(MLE)
MMR(maximal marginal relevance)reranking(最大边缘相关重排序)
extractive summarization(抽取摘要), 320-321
topic analysis(主题分析), 333
Model-based clustering algorithms(基于模型的聚类算法), 276-277
Model files for MeTA toolkit(MeTA 工具箱中的模型文件), 59
Modification in NLP(自然语言处理中的修饰), 41
Modules in content-based filtering(基于内容过滤中的模块), 224-226
MRR(mean reciprocal rank)(排序倒数均值), 180
Multiclass classification(多元分类)
linear classifiers(线性分类器), 313
text categorization(文本分类), 303
Multi-level judgments in search engine evaluation(搜索引擎评价中的多级别判断), 180-183
Multimode interactive access(多模式交互获取), 76-78
Multinomial distributions(多项分布)
Bayesian estimate(贝叶斯估计), 463-464
generalized(泛化), 460-461
LDA, 380
Multinomial parameters in Bayesian estimate(贝叶斯估计中的多项参数), 463-464
Multiple-level sentiment analysis(多层次情感分析), 397-398
Multiple occurrences in vector space model(向量空间模型中的多次出现), 103-104
Multiple queries in ranked lists evaluation(多查询有序列表的评价), 178-180
Multivariate Gaussian distribution(多变量高斯分布), 404-405
Mutual information(互信息)
information theory(信息论), 33-34
syntagmatic relations(组合关系), 264-271
text clustering(文本聚类), 278

N

n-fold cross validation(*n*-折交叉验证), 314
n-gram language models(*n* 元语言模型)
abstractive summarization(抽象式摘要), 322-323
frequency analysis(词频分析), 68-69
sentiment classification(情感分析), 394-395
term clustering(词项聚类), 288-291
vector space model(向量空间模型), 109
Naive Bayes algorithm(朴素贝叶斯算法), 309-312
Named entity recognition(命名实体识别), 323
Natural language, mining knowledge about(自然语言, 知识挖掘), 247
Natural language generation in text summarization(文本摘要中的自然语言生成), 323-324
Natural language processing(NLP, 自然语言处理)
history and state of the art(历史与现状), 42-43
pipeline(流水线), 306-307
sentiment classification(情感分类), 395
statistical language models(统计语言模型), 50-54
tasks(任务), 39-41
text information systems(文本信息系统), 43-45
text representation(文本表示), 46-50
Navigating maps in multimode interactive access(多模式交互获取中的导航地图), 77
Navigational queries(导航类查询), 200
NDCG(normalized discounted cumulative gain)(标准折扣累计收益), 181-183
NDCG@k score(NDCG@k 评分), 189
Nearest-centroid classifiers(最近质心分类器), 309
Negative feedback documents(负反馈文档), 136-138
Negative feelings(负面感受), 390-394
NetPLSA model(NetPLSA 模型), 430-433
Network supervised topic models(网络监督主题模型), 428-433
Neural language model(神经网络语言模型), 291-294
News summaries(新闻摘要), 317
Newspaper output(报纸输出), 3
ngram_pos_analyzer analyzer(**ngram_pos_analyzer** 分析器), 62
ngram_word_analyzer analyzer(**ngram_word_analyzer** 分析器), 62
NLP, 见 Natural language processing(NLP)
NLTK toolkit(NLTK 工具包), 64
no_evict_cache caches(**no_evict_cache** 缓存), 60

Nodes in word associations(词语关联节点)，252
Non-text data(非文本数据)
context(上下文)，249
predictive analysis(预测分析)，249
vs. text(文本)，244-246
Normalization(规范化)
document length(文档长度)，105-108
PageRank，206
query likelihood retrieval model(查询似然检索模型)，122
term clustering(词项聚类)，286
topic analysis(主题分析)，333
Normalized discounted cumulative gain(NDCG，标准折扣累计收益)，181-183
Normalized ratings in collaborative filtering(协同过滤中的规范化评分)，230-231
Normalized similarity algorithm(规范化相似度算法)，279

O

Objective statements vs. subjective(客观与主观陈述)，389-390
Observed world，mining knowledge about(观察到的世界，知识挖掘)，247-248
Observers，mining knowledge about(观察者，知识挖掘)，248
Office documents(Office 文档)，3
Okapi BM25 model(Okapi BM25 模型)，89，108
One-vs-all(OVA)method(OVA 方法)，313
Operators in text analysis systems(文本分析系统中的操作)，448-452
Opinion analysis in text summarization(文本摘要中的观点分析)，325-326
Opinion holders(观点持有者)，390-392
Opinion mining(观点挖掘)
evaluation(评价)，409-410
LARA，400-409
overview(概述)，389-392
sentiment classification(情感分类)，见 OSentiment analysis
Opinion summarization(观点摘要)，318
Optimization in web searches(网络搜索优化)，191
Ordinal regression(有序回归)，394，396-400
Organization in text information systems(文本信息系统中的组织)，8
OVA(one-vs-all)method(OVA 方法)，313
Over-constrained queries(约束过强查询)，84
Overfitting problem(过拟合问题)
Bayesian parameter estimation(贝叶斯参数估计)，28，30
sentiment classification(情感分类)，395
vector space model(向量空间模型)，138
Overlap of words in paradigmatic relations(聚合关系中的词项重叠)，257-258

P

p-values in search engine evaluation(搜索引擎评价中的 *p* 值)，185-186
PageRank technique(PageRank 技术)，202-206
Paradigmatic relations(聚合关系)
Brown clustering(布朗聚类)，290
discovering(发现)，252-260
overview(概述)，251-252
Parallel crawling(并行爬取)，193
Parallel indexing and searching(并行索引和搜索)，192
Parameters(参数)
background language model(背景语言模型)，350-351
Bayesian parameter estimation(贝叶斯参数估计)，28-30，341，359，458，463-464
Beta distribution(Beta 分布)，458-460
Dirichlet distribution(狄利克雷分布)，461-463
EM algorithm(EM 算法)，363，465
feedback(反馈)，142-144
LARA，404-405
LDA，380-381
mixture model estimation(混合模型估计)，352
MLE，见 Maximum likelihood estimation(MLE)
network supervised topic models(网络监督主题模型)，429
PLSA，372-373，379-380
probabilistic models(概率模型)，30-31
ranking(排序)，209-211
statistical language models(统计语言模型)，51-52
topic analysis(主题分析)，338-339
unigram language models(一元语言模型)，52
Parsing(句法分析)
MeTA toolkit(MeTA 工具箱)，67-68
NLP，43
web content(网页内容)，216

Part-of-speech(POS)tags(词性标签)
MeTA toolkit(MeTA 工具箱), 67
NLP, 47
sentiment classification(情感分类), 395
Partitioning(划分)
Brown clustering(布朗聚类), 289
extractive summarization(抽取摘要), 319-320
text data(文本数据), 417-419
Patterns(模式)
contextual text mining(上下文文本挖掘), 417-419
CPLSA, 425-426
joint analysis(联合分析), 417
NLP, 45
sentiment classification(情感分类), 395
Pdf(probability density function)(概率密度函数)
Beta distribution(Beta 分布), 457
Dirichlet distribution(狄利克雷分布), 461
multinomial distribution(多项分布), 461
Pearson correlation(皮尔逊相关系数)
collaborative filtering(协同过滤), 222, 231-232
time series context(时间序列上下文), 437
Perceptron classifiers(感知机分类器), 312-313
Personalization in web searches(网络搜索中的个性化), 212, 215
Personalized PageRank(个性化 PageRank), 206
Perspective in text data analysis(文本数据分析的视角), 246-247
Pivoted length normalization(枢轴长度规范化), 89, 107-108
PL2 model(PL2 模型), 90
PLSA(probabilistic latent semantic analysis)(概率潜在语义分析)
CPLSA, 419-428
extension(扩展), 377-383
overview(概述), 368-377
Pointwise Mutual Information(PMI, 点互信息), 278, 287-288
Polarity analysis in sentiment classification(情感分类中的倾向性分析), 394
Policy design, opinion mining for(政策设计, 观点挖掘), 393
Pooling in search engine evaluation(搜索引擎评价中的池化方法), 186-187
Porter2 English Stemmer(Porter2 英语词形还原器), 66-67
porter2_stemmer filter(**porter2_stemmer** 过滤器), 62
POS(part-of-speech)tags(词性标签)
MeTA toolkit(MeTA 工具箱), 67
NLP, 47
sentiment classification(情感分类), 395
Positive feelings(正面情绪), 390-394
Posterior distribution(后验分布), 28
Posterior probability in Bayesian parameter estimation(贝叶斯参数估计中的后验概率), 29
Postings files for inverted indexes(倒排索引的记录文件), 150-152
Power iteration for PageRank(PageRank 能量迭代), 205
Practitioners reader category(实践者读者类), 17
Pragmatic analysis in NLP(自然语言处理之语用分析), 39-40
Precision(准确率)
search engine evaluation(搜索引擎评价), 184
set retrieval evaluation(集合检索的评价), 170-178
Precision-recall curves in ranked lists evaluation(排序列表评价中的准确率-召回率曲线), 174-176
Predictive analysis for non-text data(非文本数据预测分析), 249
Predictors features in joint analysis(联合分析中的预测器特征), 413-416
Presupposition in NLP(自然语言处理之预设), 41
Prior probability in Bayesian parameter estimation(贝叶斯参数估计先验概率), 29
Probabilistic inference(概率推断), 88
Probabilistic latent semantic analysis(PLSA, 概率潜在语义分析)
CPLSA, 419-428
extension(扩展), 377-383
overview(概述), 368-377
Probabilistic retrieval models(概率检索模型)
description(描述), 87-88
overview(概述), 110-112
query likelihood retrieval model(查询似然检索模型), 114-118
Probability and statistics(概率和统计)
abstractive summarization(抽象摘要), 322
background language model(背景语言模型), 346-349
basics(基础), 21-23
Bayes' rule(贝叶斯法则), 25-26
Bayesian parameter estimation(贝叶斯参数估

计)，28-30
binomial distribution(二项分布)，26-27
EM algorithm(EM 算法)，362-366
joint and conditional probabilities(联合和条件概率)，23-25
KL-divergence(KL-散度)，474
LARA，403
maximum likelihood parameter estimation(最大似然参数估计)，27-28
mixture model behavior(混合模型行为)，354-358、
mutual information(互信息)，266-270
Naive Bayes algorithm(简单贝叶斯算法)，310
PageRank，202-206
paradigmatic relations(聚合关系)，257-258
PLSA，368-377，380
probabilistic models and applications(概率模型和应用)，30-31
syntagmatic relations(组合关系)，262-263
term clustering(词项聚类)，286-289
topics(主题)，336-339
unigram language model(一元语言模型)，342-344
web search ranking(网络搜索排序)，209-211
Probability density function(pdf，概率密度函数)
Beta distribution(Beta 分布)，457
Dirichlet distribution(狄利克雷分布)，461
multinomial distribution(多项分布)，461
Probability distributions(概率分布)
overview(概述)，21-23
statistical language models(统计语言模型)，50-54
Probability ranking principle(概率排序原则)，84
Probability space(概率空间)，21-23
Producer-initiated recommendations(信息生成者发起的推荐)，75
Product reviews in opinion mining(观点挖掘中的产品评论)，391-392
profile command(**profile** 命令)，65-66
Properties(属性)
inferring knowledge about(推测知识)，248
text categorization for(文本分类)，300
Proximity heuristics for inverted indexes(倒排索引临近启发规则)，151
Pseudo counts(伪计数)
Bayesian statistics(贝叶斯统计)，459-460
LDA，381
multinomial distribution(多项分布)，463
PLSA，379，381
smoothing techniques(平滑技术)，128，286
Pseudo data in LDA(LDA 伪计数)，378
Pseudo feedback(伪反馈)，133，135，142，157-158
Pseudo-segments for mutual information(互信息中的伪片段)，269-270
Pull access mode(拉取获取模式)，8-9，73-76
Push access mode(推送获取模式)，8-9，73-76
Python language(Python 语言)
cleaning HTML files(清理 HTML 文件)，218
crawlers(爬虫)，217

Q

Q-function(Q-函数)，465，469-471
Queries(查询)
multimode interactive access(多模式交互获取)，77
navigational(导航)，200
text information systems(文本信息系统)，9
text retrieval vs. database retrieval(文本检索 vs. 数据库检索)，80
Query expansion(查询扩展)
vector space model(向量空间模型)，135
word associations(词关联)，252
Query likelihood retrieval model(查询似然检索模型)，90，113
document language model(文档语言模型)，118-123
feedback(反馈)，139
KL-divergence(KL-散度)，475-476
overview(概述)，114-118
smoothing methods(平滑方法)，123-128
Query vectors(查询向量)，92-98，135-137

R

Random access decoding in compression(压缩中的随机访问解码)，158
Random numbers in abstractive summarization(抽象摘要中的随机数)，322
Random observations in search engine evaluation(搜索引擎评价随机观察)，186
Random surfers in PageRank(PageRank 随机游走)，202-204
Random variables(随机变量)
Bayesian parameter estimation(贝叶斯参数估计)，28
dependent(相关)，25

entropy of(熵)，158，262-263，270
information theory(信息论)，31-34
PMI(点互信息)，287
probabilistic retrieval models(概率检索模型)，87，111，113
probability distributions(概率分布)，22
Ranked lists evaluation(有序列表的评价)
multiple queries(多查询)，178-180
overview(概述)，174-178
Rankers for search engines(搜索引擎排序器)，147
Ranking(排序)
extractive summarization(抽取摘要)，320
probabilistic retrieval model(概率检索模型). 见 Probabilistic retrieval models
vs. selection(选择)，82-84
text analysis operator(文本分析操作)，450-451
text data access(文本数据获取)，78
vector space model(向量空间模型)，见 Vector space(VS) retrieval models
web searches(网络搜索)，201，208-212
Ratings(评分)
collaborative filtering(协同过滤)，230-231
LARA，400-409
sentiment classification(情感分类)，396-399
Real world properties，inferring knowledge about(现实世界的属性，推测知识)，248
Realization in abstractive summarization(抽象摘要生成)，324
Recall in set retrieval evaluation(基于集合的信息检索评价中的召回率)，170-178
Reciprocal ranks(倒数排序)，179-180
Recommendations in text information systems(文本信息系统中的推荐)，11
Recommender systems(推荐系统)
collaborative filtering(协同过滤)，229-233
content-based recommendation(基于内容的推荐)，222-229
evaluating(评价)，233-235
overview(概述)，221-222
reduce function(**reduce** 函数)，198
Redundancy(冗余性)
MMR reranking(MMR 重排序)，333
text summarization(文本摘要)，320-321，324
vector space retrieval models(向量空间检索模型)，92
Regression(回归)
LARA，402-405
machine learning(机器学习)，34-35
sentiment classification(情感分类)，394，396-400
text categorization(文本分类)，303-304
web search ranking(网络搜索排序)，209-211
Regularizers in network supervised topic models(网络监督主题模型正则化)，429-431
Relevance and relevance judgments(相关性与相关性评价)
Cranfield evaluation methodology(Cranfield 评价方法)，168-169
description(描述)，133
document ranking(文档排序)，83
document selection(文档选择)，83
extractive summarization(抽取摘要)，321
probabilistic retrieval models(概率检索模型)，110-112
search engine evaluation(搜索引擎评价)，181-184，186-187
set retrieval evaluation(集合检索的评价)，171-172
text data access(文本数据获取)，79
vector space model(向量空间模型)，92
web search ranking(网络搜索排序)，209-211
Relevant text data(相关文本数据)，5-6
Relevant word counts in EM algorithm(EM 算法中的相关词计数)，364-365，376-377
Repeated crawling(反复爬取)，193
Representative documents in search engine evaluation(搜索引擎评价中的代表性文档)，183
reset command(**reset** 命令)，57-59
Retrieval models(检索模型)
common form(一般形式)，88-90
overview(概述)，87-88
probabilistic(概率)，见 Probabilistic retrieval models
vector space(向量空间)，见 Vector space(VS) retrieval models
Reviews(评论)
LARA，400-409
opinion mining(观点挖掘)，391-392
sentiment classification(情感分类)，394
text summarization(文本摘要)，318
RMSE(root-mean squared error)(根均方误差)，233
robots.txt file(**robots.txt** 文件)，193
Rocchio feedback(Rocchio 反馈)
forward indexes(前向索引)，157
vector space model(向量空间模型)，135-138

Root-mean squared error(RMSE)(根均方误差)，233
Ruby language(Ruby 语言)
cleaning html files(清理 html 文件)，218-219
crawlers(爬虫)，217
Rule-based text categorization(基于规则的文本分类)，301

S

Scalability in web searches(网络搜索的可扩展性)，191-192
Scanning inverted indexes(扫描倒排索引)，152
Scientific research，text data analysis for (科学研究，文本数据分析)，243
Scikit Learn toolkit(Scikit Learn 工具箱)，64
score_term function(**score_term** 函数)，154
Scorers(评分器)
document-at-a-time ranking(逐个文档排序)，155
filtering documents(过滤文档)，155-156
index sharding(索引分片)，156-157
search engines(搜索引擎)，147，153-157
term-at-a-time ranking(逐个词项排序)，154-155
Scoring functions(评分函数)
KL-divergence(KL-散度)，474
topic analysis(主题分析)，332
SDI(selective dissemination of information)(信息选择性传播)，75
Search engine evaluation(搜索引擎评价)
Cranfield evaluation methodology(Cranfield 评价方法)，168-170
measurements(度量)，168
multi-level judgments(多级别判断)，180-183
practical issues(实际情况)，183-186
purpose(目的)，167-168
ranked lists(排序列表)，174-180
set retrieval(集合检索)，170-173
Search engine implementation(搜索引擎实现)
caching(缓存)，162-165
compression(压缩)，158-162
feedback implementation(反馈实现)，157-158
indexes(索引)，150-153
overview(概述)，147-148
scorers(分数)，153-157
tokenizers(分词器)，148-150
Search engine queries(搜索引擎查询)
pull access mode(拉取获取模式)，74-75
text data access(文本数据获取)，78-79
Search engine toolkits(搜索引擎工具箱)，64
Searches(搜索)
text information systems(文本信息系统)，11
web(网络)，见 Web searches
Segmentation in LARA(LARA 中的方面细分)，405
Select operator(选择操作)，449-451，455
Selection(选择)
vs. ranking(排序)，82-84
text data access(文本数据获取)，78
Selection-based text summarization(基于选择的文本摘要)，318
Selective dissemination of information(SDI，信息选择性传播)，75
Semantic analysis in NLP(NLP 语义分析)，39-40，43，47
Semantically related terms in clustering(聚类中的语义相关词项)，187，285-287
Sensors(传感器)
humans as(人类)，244-246
joint analysis(联合分析)，413-415
opinion mining(观点挖掘)，见 Opinion mining
Sentence vectors in extractive summarization(抽取摘要中的句子向量)，319
Sentiment analysis(情感分析)，389
classification(分类)，393-396
evaluation(评价)，409-410
NLP，43
ordinal regression(有序回归)，396-400
text categorization(文本分类)，304
Separation in text clustering(文本聚类中的分离)，294-295
Sequences of words in NLP(自然语言处理中的词序列)，46-47
Set retrieval evaluation(集合检索的评价)
description(描述)，170
F measure(F 度量)，172-173
precision and recall(准确率与召回率)，170-173
Shadow analysis in NLP(NLP 浅层分析)，48，43-45
Short-range walks in multimode interactive access (多模式交互式获取中的短距游走)，77
Short-term needs in pull access mode(拉取式获取模式中的短期需求)，75
Sign tests in search engine evaluation(搜索引擎评价符号测试)，185

Signed-rank tests in search engine evaluation(搜索引擎评价符号排序测试), 185
Significance tests in search engine evaluation(搜索引擎评价显著性检验), 183-186
Similarity algorithm for clustering(聚类中的相似度算法), 276
Similarity in clustering(聚类中的相似度)
Similarity functions and measures(相似度函数和度量)
  extractive summarization(抽取摘要), 319, 321
  paradigmatic relations(聚合关系), 256-259
  vector space model(向量空间模型), 92, 109
  description(描述), 277
  document clustering(文档聚类), 279-281
  term clustering(词聚类), 285
Single-link document clustering(单链文档聚类), 281-282
Skip-gram neural language model(Skip-gram 神经语言模型), 292-293
sLDA(supervised LDA)(有监督 LDA 模型), 387
Smoothing techniques(平滑技术)
  Add-1(加一), 130, 464
  Bayesian statistics(贝叶斯统计), 459-460
  KL-divergence(KL-散度), 474-475
  maximum likelihood estimation(最大似然估计), 119-128
  multinomial distribution(多项分布), 463-464
  Naive Bayes algorithm(朴素贝叶斯算法), 310
  unigram language models(一元语言模型), 53
Social media in text data analysis(社交媒体文本数据分析), 243
Social networks as context(社交网络上下文), 428-433
Social science research, opinion mining for(社会科学研究观点挖掘), 393
Soft rules in text categorization(文本分类中的软规则), 301
Spam in web searches(网络搜索垃圾页面), 191-192
Sparse Beta(稀疏的 Beta), 459
Sparse data in Naive Bayes algorithm(朴素贝叶斯算法中的稀疏数据), 309-311
Sparse priors in Dirichlet distribution(狄利克雷分布稀疏先验), 461-462
Spatiotemporal patterns in CPLSA(CPLSA 时空模式), 425-426
Specificity in sentiment classification(情感分类中的具体性), 396
Speech acts in NLP(自然语言处理之言语行为), 47-48
Speech recognition applications(语音识别)
  application(应用), 42
  statistical language models(统计语言模型), 51
Spiders for web searches(网络搜索爬虫), 192-194
Split counts in EM algorithm(EM 算法中的划分计数), 374-375
Split operator for text analysis(文本分析划分操作), 449-452, 455
Stanford NLP toolkit(Stanford NLP 工具箱), 64
State-of-the-art support vector machines (SVM) classifiers(最先进的支持向量机分类器), 311-312
Statistical language models(统计语言模型)
  NLP, 45
  overview(概述), 50-54
Statistical machine learning(统计机器学习)
  NLP, 42-43
  text information systems(文本信息系统), 10
Statistical significance tests in search engine evaluation(搜索引擎评价中的统计显著性检验), 183-186
Statistics(统计), 见 Probability and statistics
Stemmed words in vector space model(向量空间模型中的词干提取), 109
Stemming process in META toolkit(META 工具箱的词干提取处理), 66-67
Sticky phrases in Brown clustering(布朗聚类中的固定搭配短语), 291
Stop word removal(去除停用词)
  feedback(反馈), 141
  frequency analysis(词频分析), 69
  META toolkit(META 工具箱), 62, 66
  mixture models(混合模型), 352
  vector space model(向量空间模型), 99, 109
Story understanding(故事理解), 42
Structured data(结构化数据)
  databases(数据库), 80
  joint analysis with text(与文本联合分析), 见 oint analysis of text and structured data
Student reader category(学生读者类别), 16
Stylistic analysis in NLP(自然语言处理之风格分析), 49
Subjective sensors(主观传感器)
  humans as(人类), 244-246

opinion mining(观点挖掘)，见 Opinion mining
Subjective statements vs. objective(主观与客观陈述)，389-390
Sublinear transformation(次线性变换)
term frequency(词项频率)，258-259
vector space model(向量空间模型)，103-104
Summarization(摘要)，见 Text summarization
Supervised LDA(sLDA)(有监督 LDA 模型)，387
Supervised machine learning(有监督机器学习)，34
SVM (state-of-the-art support vector machines) classifiers(支持向量机分类器)，311-312
Symbolic approach in NLP(自然语言处理之符号方法)，42
Symmetric Beta(对称的 Beta)，459
Symmetric probabilities in information theory(信息论中的对称概率)，32
Symmetry in document clustering(文档聚类对称性)，279-280
Synonyms(同义词)
vector space model(向量空间模型)，92
word association(词关联)，252
Syntactic ambiguity in NLP(自然语言处理之句法歧义)，41
Syntactic analysis in NLP(自然语言处理之句法分析)，39-40，47
Syntactic structures in NLP(自然语言处理之句法结构)，49
SyntacticDiff method(SyntacticDiff 方法)，306
Syntagmatic relations(组合关系)，251-252
Brown clustering(布朗聚类)，290-291
discovering(发现)，253-254，260-264
mutual information(互信息)，264-271
System architecture in unified systems(统一系统架构)，452-453

T

Tags，POS(词性标签)
MeTA toolkit(MeTA 工具箱)，67
NLP，47
sentiment classification(情感分类)，395
Targets in opinion mining(观点挖掘目标)，390-392
Temporal trends in CPLSA(CPLSA 时序趋势)，424-425
Term-at-a-time ranking(逐词排序)，154-155
Term clustering(词项聚类)，278
n-gram class language models(*n* 元类语言模型)，288-291
neural language model(神经语言模型)，291-294
overview(概述)，284-285
Pointwise Mutual Information(点互信息)，287-288
semantically related terms(语义相关词)，285-287
Term frequency(TF，词项频率)
bag-of-words representation(词袋表示)，89
vector space model(向量空间模型)，97-98
Term IDs(词项 ID)
inverted indexes(倒排索引)，151-152
tokenizers(分词器)，149-150
Term vectors(词项向量)，92
Terms，topics as(基于词项的主题)，332-335
Terrier search engine toolkit(Terrier 搜索引擎工具)，64
Test collections in Cranfield evaluation methodology (Cranfield 评价方法测试文档集)，168-169
Testing data(测试数据)
machine learning(机器学习)，35
text categorization(文本分类)，303
Text(文本)
joint analysis with structured data(与结构化数据联合分析)，见 Joint analysis of text and structured data
mining(挖掘)，见 Mining; Mining topics from text
usefulness(有用性)，3-4
Text annotation(文本标注)，见 Text categorization
Text-based prediction(基于文本的预测)，300
Text categorization(文本分类)
classification algorithms overview(分类算法概述)，307
evaluation(评价)，313-315
features(特征)，304-307
introduction(介绍)，299-301
k-nearest neighbors algorithm(*k*-近邻算法)，307-309
linear classifiers(线性分类器)，311-313
machine learning(机器学习)，35
methods(方法)，300-302
Naive Bayes(朴素贝叶斯)，309-311
problem(问题)，302-304
Text clustering(文本聚类)，12
document(文档)，279-284
evaluation(评价)，294-296
overview(概述)，275-276

techniques(技术), 277-279
term(词项), 284-294
Text data access(文本数据获取), 73
access modes(获取模式), 73-76
document selection vs. document ranking(文档选择与文档排序), 82-84
multimode interactive(多模式交互), 76-78
text retrieval vs. database retrieval(文本检索与数据库检索), 80-82
text retrieval overview(文本检索概述), 78-80
Text data analysis overview(文本数据分析概述), 241-242
applications(应用), 242-244
humans as subjective sensors(人作为主观传感器), 244-246
operators(操作器), 448-452
text information systems(文本信息系统), 8
text mining tasks(文本挖掘任务), 246-250
Text data understanding(文本数据理解). 见 Natural language processing(NLP)
Text information systems(TIS, 文本信息系统)
conceptual framework(概念框架), 10-13
functions(函数), 7-10
NLP, 43-45
Text management and analysis in unified systems(文本管理与分析联合系统), 见 Unified systems
Text organization in text information systems(文本信息系统中的文本组织), 8
Text representation in NLP(自然语言处理之文本表示), 46-50
Text retrieval(TR)(文本检索)
vs. database retrieval(数据库检索), 80-82
demand for(需求), 4-5
overview(概述), 78-80
Text summarization(文本摘要), 12
abstractive(抽象式), 321-324
applications(应用), 325-326
evaluation(评价), 324-325
extractive(抽取式), 319-321
overview(概述), 317-318
techniques(技术), 318
TextObject data type operators(**TextObject** 数据类型操作), 449, 454
TextObjectSequence data type operators(**TextObjectSequence** 数据类型操作), 449, 454
TF(term frequency)(词项频率)
bag-of-words representation(词袋表示), 89
vector space model(向量空间模型), 97-98
TF-IDF weighting(TF-IDF 权重计算)
Dirichlet prior smoothing(狄利克雷先验平滑), 128
probabilistic retrieval model(概率检索模型), 122-123
topic analysis(主题分析), 333
vector space model(向量空间模型), 100-103
TF transformation(词频变换), 102-105
TF weighting(词频权重计算), 125-126
Themes in CPLSA(CPLSA 主题), 420-422
Therapist application(临床应用), 44-45
Thesaurus discovery in NLP(自然语言处理之同义词典挖掘), 49
Threshold settings in content-based filtering(基于内容过滤中的阈值设置), 222, 224-227
Tight clusters(紧密聚类簇), 281
Time series context in topic analysis(时间序列上下文主题分析), 433-439
TISs(text information systems)(文本信息系统)
conceptual framework(概念框架), 10-13
functions(函数), 7-10
NLP, 43-45
Tokenization(分词)
MeTA toolkit(MeTA 工具箱), 61-64, 453
search engines(搜索引擎), 147-150
Topic analysis(主题分析)
evaluation(评价), 383-384
LDA, 377-383
mining topics from text(文本主题挖掘), 见 Mining topics from text
model summary(模型概要), 384-385
overview(概述), 329-331
PLSA, 368-377
social networks as context(社交网络上下文), 428-433
text information systems(文本信息系统), 12
time series context(时间序列上下文), 433-439
topics as terms(基于词的主题), 332-335
topics as word distributions(基于词分布的主题), 335-340
Topic coherence in time series context(时间序列上下文下的主题凝聚性), 436
Topic coverage(主题覆盖)
CPLSA, 420-422, 425-426
LDA, 380-381

Topic maps in multimode interactive access(多模式交互获取中的主题映射)，76-77
TopicExtraction operator (TopicExtraction 操作)，450
TR(text retrieval)(文本检索)
  vs. database retrieval(数据库检索)，80-82
  demand for(需求)，4-5
  overview(概述)，78-80
Training and training data(训练和训练数据)
  classification algorithms(分类算法)，307-309
  collaborative filtering(协同过滤)，229-230
  content-based recommendation(基于内容的推荐)，227-228
  linear classifiers(线性分类器)，311-313
  machine learning(机器学习)，34-36
  Naive Bayes(朴素贝叶斯)，309-310
  NLP，42-43，45
  ordinal regression(有序回归)，398-399
  text categorization(文本分类)，299-303，311-314
  web search ranking(网络搜索排序)，209-210，212
Transformations(变换)
  frequency(频率)，258-259
  vector space model(向量空间模型)，103-104
Transition matrices in PageRank(PageRank 转移矩阵)，204
Translation，machine(翻译，机器)，42，44-45
TREC filtering tasks(TREC 过滤任务)，228
tree_analyzer analyzer(**tree_analyzer** 分析器)，62
Trends in web searches(网络搜索趋势)，215-216
Trigrams in frequency analysis(三元词串频率分析)，69
Twitter searches(微博搜索)，83
Two-component mixture model(两组件混合模型)，356

U

Unary bitwise compression(一元按位压缩)，159-160
Under-constrained queries(约束不足查询)，84
Unified systems(统一系统)
  META as(META)，453-455
  overview(概述)，445-448
  system architecture(系统架构)，452-453
  text analysis operators(文本分析操作)，448-452
Uniform priors in Dirichlet distribution(狄利克雷分布均匀先验)，461
Unigram language models(一元语言模型)，51-54
  EM algorithm(EM 算法)，466
  LDA，381
  mining topics from text(文本主题挖掘)，341-345
  PLSA，370
Unigrams(一元词)
  abstractive summarization(抽象式摘要)，321-323
  frequency analysis(词频分析)，68
  sentiment classification(情感分类)，394
  words tokenizers(分词器)，149
Unimodel Beta(单峰 Beta)，459
Union operator(并操作)，449-450
University of California Berkeley study(加州大学伯克利分校研究)，3
Unseen words(未出现词)
  document language model(文档语言模型)，119-120，122-123
  KL-divergence(KL-散度)，474
  Naive Bayes algorithm(朴素贝叶斯算法)，310-311
  smoothing(平滑)，124，285-287
  statistical language models(统计语言模型)，52
Unstructured text access(无结构文本获取)，80
Unsupervised clustering algorithms(无监督聚类算法)，275，278
Unsupervised machine learning(无监督机器学习)，34，36
URLs and crawlers(URL 地址与爬虫)，193
Usability in search engine evaluation(搜索引擎评价中的可用性)，168
Utility(效用)
  content-based filtering(基于内容的过滤)，224-228
  text clustering(文本聚类)，294

V

Valence scoring(价态评分)，411
Variable byte encoding(变量字节编码)，161
Variables(变量)
  context(上下文)，330
  contextual text mining(上下文文本挖掘)，419
  CPLSA，422

EM algorithm(EM 算法), 362-364, 366, 368, 373-376, 465, 467
LARA, 403
random(随机), 见 Random variables
vByte encoding(vByte 编码), 161
Vector space(VS) retrieval models(向量空间检索模型), 87
bit vector representation(位向量表示), 94-97
content-based filtering(基于内容的过滤), 225-226
document length normalization(文档长度规范化), 105-108
feedback(反馈), 135-138
improved instantiation(改进实例化), 97-102
improvement possibilities(改进可能性), 108-110
instantiation(实例化), 93-95
overview(概述), 90-92
paradigmatic relations(聚合关系), 256-258
summary(总结), 110
TF transformation(TF 变换), 102-105
Vectors(向量)
collaborative filtering(协同过滤), 222
neural language model(神经语言模型), 292
Versions, MeTA toolkit(版本, MeTA 工具箱), 59
Vertical search engines(垂直搜索引擎), 212
Video data mining(视频数据挖掘), 245
Views(角度)
CPLSA, 420-422
multimode interactive access(多模式交互获取), 77
Visualization in text information systems(文本信息系统中的可视化), 12-13
VS retrieval models(向量空间检索模型), 见 Vector space(VS) retrieval models

W

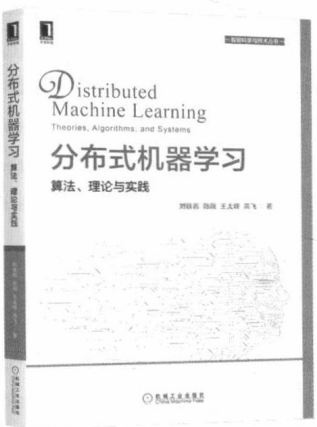
Web searches(网络搜索)
crawlers(爬虫), 192-194
future of(未来), 212-216
indexing(索引), 194-200
link analysis(链接分析), 200-208
overview(概述), 191-192
ranking(排序), 208-212
Weighted k-nearest neighbors algorithm(加权 k 近邻算法), 309
WeightedTextObjectSequence data type(**WeightedTextObjectSequence** 数据类型), 449
WeightedTextObjectSet data type(**WeightedTextObjectSet** 数据类型), 449
Weights(权重)
collaborative filtering(协同过滤), 231
Dirichlet prior smoothing(狄利克雷先验平滑), 127-128
document clustering(文档聚类), 279-280
LARA, 401-409
linear classifiers(线性分类器), 313
mutual information(互信息), 269-270
NetPLSA model(NetPLSA 模型), 430
network supervised topic models(网络监督主题模型), 431
paradigmatic relations(聚合关系), 258-261
query likelihood retrieval model(查询似然检索模型), 121-123
text categorization rules(文本分类规则), 301
topics(主题), 333, 335-336
vector space model(向量空间模型), 92, 99-103
Weka toolkit(Weka 工具箱), 64
whitespace_tokenizer command(**whitespace_tokenizer** 命令), 149
Whitespace tokenizers(空格分词器), 149
Wilcoxon signed-rank test(Wilcoxon 符号排序检验), 185
Word association mining(词关联挖掘)
evaluation(评价), 271-273
general idea of(基本思想), 252-254
overview(概述), 251-252
paradigmatic relations discovery(聚合关系的发现), 254-260
syntagmatic relations discovery(组合关系的发现), 260-271
Word counts(词计数)
EM algorithm(EM 算法), 364-365, 376-377
MapReduce, 195-198
vector space model(向量空间模型), 103-104
Word distributions(词分布)
CPLSA, 424-425
LARA, 405
topics as(主题), 335-340
Word embedding in term clustering(词项聚类中的词嵌入), 291-294
Word-level ambiguity in NLP(自然语言处理之词

汇级歧义)，41
Word relations(词语关系)，251-252
Word segmentation in NLP(自然语言处理之分词)，46
Word sense disambiguation in NLP(自然语言处理之词义消歧)，43
Word valence scoring(词价态评分)，411
Word vectors in text clustering(文本聚类中的词向量)，278
word2vec skip-gram (**word2vec** skip-gram 算法)，293
WordNet ontology(WordNet 本体)，294

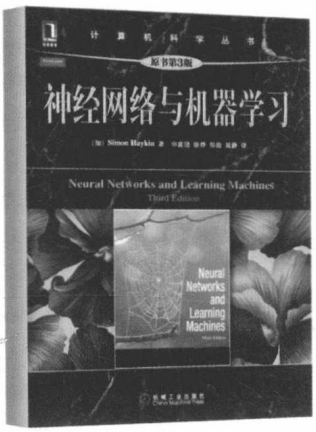
Z

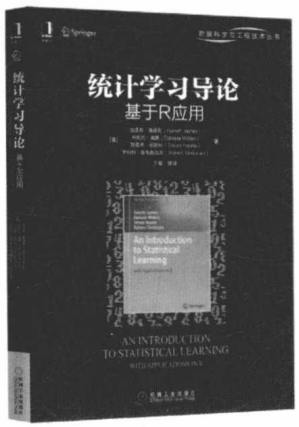
Zhai，ChengXiang，biography(翟成祥生平)，489
Zipf's law (齐普夫定律)
  caching(缓存)，163
  frequency analysis(词频分析)，69-70

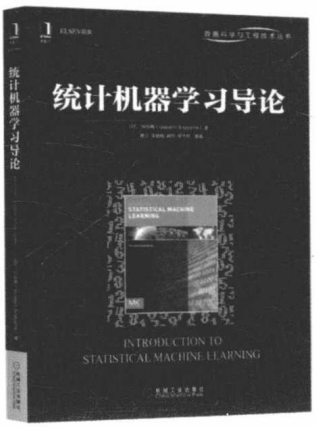
推荐阅读

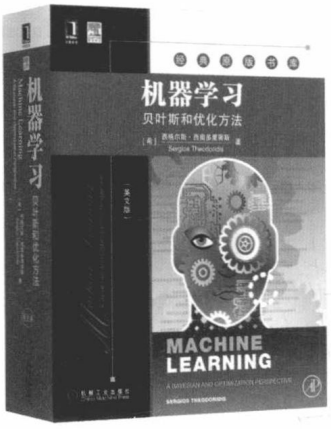












分布式机器学习：算法、理论与实践

作者：刘铁岩 陈薇 王太峰 高飞 ISBN：978-7-111-60918-6 定价：89.00元

深入理解机器学习：从原理到算法

作者：[以] 沙伊·沙莱夫-施瓦茨 等 ISBN：978-7-111-54302-2 定价：79.00元

神经网络与机器学习（原书第3版）

作者：（加）Simon Haykin ISBN：978-7-111-32413-3 定价：79.00元

统计学习导论——基于R应用

作者：[美]加雷斯·詹姆斯 等 ISBN：978-7-111-49771-4 定价：79.00元

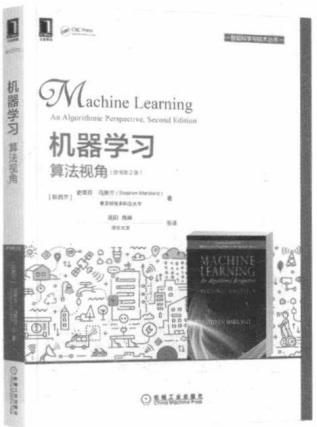
统计机器学习导论

作者：[日]杉山将 ISBN：978-7-111-59679-0 定价：89.00元

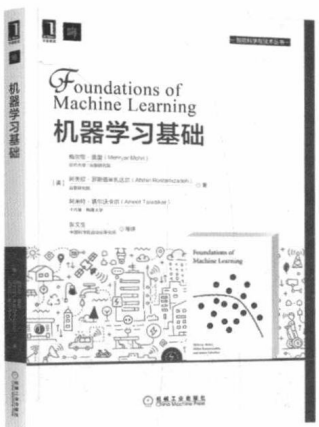
机器学习：贝叶斯和优化方法（英文版）

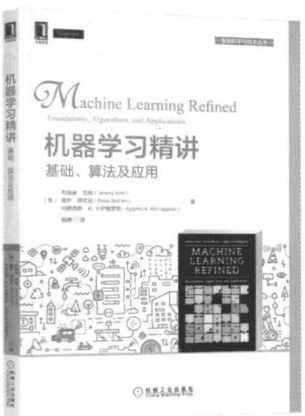
作者：[希]西格尔斯·西奥多里蒂斯 ISBN：978-7-111-56526-0 定价：269.00元

推荐阅读

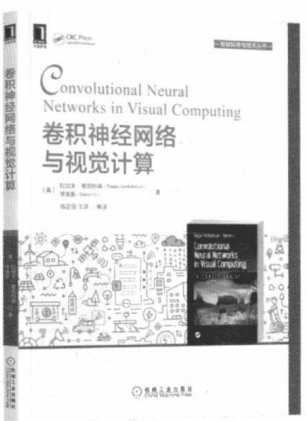












机器学习：算法视角（原书第15版）

作者：Stephen Marsland 译者：高阳 商琳 等 ISBN：978-7-111-62226-0 定价：99.00元

当计算机体系结构遇到深度学习

作者：Brandon Reagen 等 译者：杨海龙 王锐 ISBN：978-7-111-62248-2 定价：69.00元

机器学习基础

作者：Mehryar Mohri 等 译者：张文生 等 ISBN：978-7-111-62218-5 定价：待定

机器学习精讲：基础、算法及应用

作者：Jeremy Watt 等 译者:杨博 等 ISBN：978-7-111-61196-7 定价：69.00元

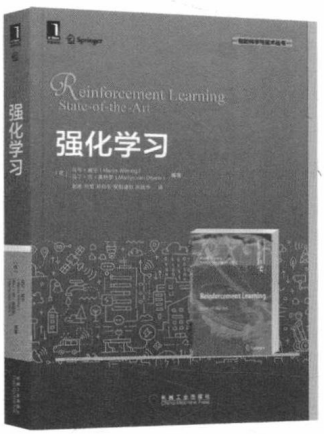
基于复杂网络的机器学习方法

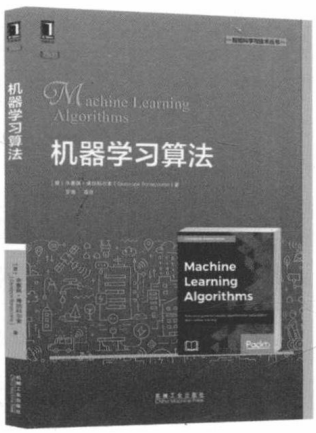
作者：Thiago Christiano Silva 等 译者:李泽荃 等 ISBN：978-7-111-61149-3 定价：79.00元

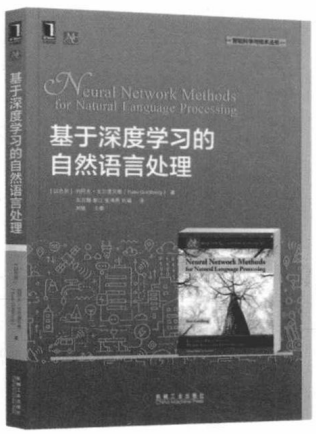
卷积神经网络与视觉计算

作者：Ragav Venkatesan 等 译者：钱亚冠 等 ISBN：978-7-111-61239-1 定价：59.00元

推荐阅读

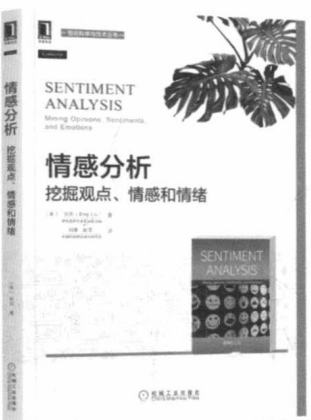












强化学习

作者：[荷] 马可·威宁 等 ISBN：978-7-111-60022-0 定价：119.00元

机器学习算法

作者：[意]朱塞佩·博纳科尔索 ISBN：978-7-111-59513-7 定价：69.00元

基于深度学习的自然语言处理

作者：[以色列] 约阿夫·戈尔德贝格 ISBN：978-7-111-59373-7 定价：69.00元

数据挖掘：实用机器学习工具与技术（原书第4版）

作者：[新西兰] 伊恩 H. 威腾 等 ISBN：978-7-111-58916-7 定价：99.00元

神经网络设计（原书第2版）

作者：[美] 马丁 T. 哈根 ISBN：978-7-111-58674-6 定价：99.00元

情感分析：挖掘观点、情感和情绪

作者：[美] 刘兵 ISBN：978-7-111-57498-9 定价：99.00元

文本数据管理与分析

信息检索与文本挖掘的实用导论

TEXT DATA MANAGEMENT AND ANALYSIS

A PRACTICAL INTRODUCTION TO INFORMATION RETRIEVA

本书是文本信息处理领域著名学者美国伊利诺伊大学香槟分校(University of Illinois at Urbana-Champaign，UIUC)翟成祥教授的倾力之作，主要基于作者在UIUC讲授数据管理与分析相关课程（如CS410文本信息系统，Text Information Systems）使用的材料。书中的大部分内容及结构与翟成祥教授在Coursera开设的两门在线课程“文本检索与搜索引擎”（Text Retrieval and Search Engines）和“文本挖掘与分析”（Text Mining and Analytics）一致，因此可作为两门在线课程的主要参考书。

全书内容以文本数据处理为核心，从理论到实践介绍了文本数据管理与分析的关键问题，广泛涵盖了信息检索和文本挖掘相关技术。

本书具体内容

- 文本信息获取与挖掘基础：统计与概率论、信息论等相关理论和文本数据理解技术。
- 文本信息获取关键技术：信息检索的模型、实现和评价，网络搜索以及推荐系统等。
- 文本挖掘关键技术：文档分类，文档聚类，文本摘要，主题分析，观点挖掘与情感分析，文本与结构化数据联合分析等。
- 文本管理和分析系统：整合信息检索与文本分析技术，结合配套软件工具META，构建统一的、人机结合的文本管理和分析系统。

本书特点

- 内容深入浅出，理论实践密切结合，可作为高等院校计算机科学、数据科学或相关专业本科生、研究生信息检索与文本挖掘课程的教材。
- 主题丰富，体系完整，逻辑清晰，便于与各种课程体系结合，可作为高等院校相关专业教师的参考书。
- 兼顾广度与深度，覆盖经典模型与方法并提供丰富的扩展阅读文献，可作为科研人员与工业界人士的参考书或技术手册。

上架指导：计算机\数据挖掘

ISBN 978-7-111-61176-9

9 787111 611769

定价：139.00元

投稿热线：(010) 88379604
购书热线：(010) 68326294
客服热线：(010) 88379426 88361066

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn