密级:_____

硕士学位论文

面向主题的文本挖掘技术研究与应用

作者姓名：魏永勇

指导教师：马俊才　　正高级工程师

中国科学院微生物研究所

学位类别：工学硕士

学科专业：计算机应用技术

研究所：中国科学院计算机网络信息中心

2013 年 5 月

Theme-oriented Text Mining Technology Research and Application

By

Wei Yongyong

A Dissertation/Thesis Submitted to

The University of Chinese Academy of Sciences

In partial fulfillment of the requirement

For the degree of

Master of Engineering

Computer Network Information Center

Chinese Academy of Sciences

May, 2013

声　　明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。就我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

作者签名：　　日期：

关于论文使用授权的说明

中国科学院计算机网络信息中心、中国科学院大学有权处理、保留送交论文的复印件，允许论文被查阅和借阅；并可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存该论文。

作者签名：　　导师签名：　　日期：

摘 要

世界微生物数据中心所建立的生物资源引用平台搜集了大量的生物资源类文献，并且分析了这些文献与微生物菌种之间的引用关系，一方面揭示了各菌种的应用领域，另一方面也为评价各个菌种保藏中心提供了间接的依据，当前该平台已被微生物研究领域的研究人员广泛使用。为了进一步分析这些文献资源的特征，结合文本挖掘技术，本文尝试从自然语义层面上分析文献与菌种之间的相关程度，完善该平台的功能，包括对文献进行自动分类，对文献打分，实现检索文献功能。

基于此目的，本文提出了如下的研究路线。首先利用自然语言处理工具，实现对文献资源的主题进行自动抽取，建立文献的中间表示状态。接着本文比较了贝叶斯分类与向量空间分类算法的实现效率，完成对文献资源的分类标注。

然后本文通过基于密度的聚类算法对文献资源做聚类分析，抽取引用菌种频率较高的文献集合，再从这些文献集合中提炼出菌种相关的主题词典，从而完成对文献资源的打分，评价与菌种的相关程度。然后通过对主题词的索引，实现了对文献资源的检索功能。在文章的最后，根据菌种出现在同一篇文献中的情况，对菌种进行了共现分析。

综合以上挖掘分析方法，本文取得了以下成果：

(1) 实现了对文献自动分类的功能，提供按类别进行统计的结果，使得文献与菌种之间的引用关系更加细化。

(2) 生成了菌种相关主题词典，完成了对文献与菌种之间相关关系进行打分的算法，不仅能辅助修正系统的错误，而且为检索结果的排序奠定了基础。

(3) 实现了基于主题的检索，使得研究人员能快速获取不同主题对应下的文献资源与菌种信息。

(4) 通过对菌种之间的关联关系的分析，为寻找菌种之间的共生关系提供了参考依据。

整体上看，各系统模块都已经顺利完成并整合到原来系统平台，提供了更为全面的参考信息供用户查询，但在一些细节的处理以及系统性能的优化方面，还有进一步的工作需要完善。

【关键字】文本挖掘 ，预处理，分类，聚类分析，检索

Abstract

World Data Center for Microbiology has built a platform called Analyzer of Bio-resource Citations, this platform has collected a large number of biological resources and analyzed the reference relationship between literature and strains, on the one hand showed the areas of strains application, and on the other hand also provide evidence for the evaluation of the Culture Collection. To further analyze the characteristics of the literature resources, combined with text mining techniques, this article attempts to analyze the correlation between literature and strains from the aspect of semantic level, so as to finish the function of the platform, including automatic classification, score the literature, and retrieve the relevant literatures.

For this purpose, we propose the following line of research. First, we used natural language processing tools to extract the theme of the literature resources automatically, and then used a middle state to represent those literatures. After this we compared the efficiency of the Bayes classifier and vector space model classifier and finished the classification task.

Then we used a clustering algorithm which based on density to cluster the literature resources and extracted those literatures which have a high frequency of strains reference, and then from those literatures we came up with a dictionary that is related to strains, with this dictionary, we can evaluate the relevance between strains and literatures. Next we built index for those literatures and finished the search module. At last, according to the strains appear in the same literature, we finished the strains co-occurrence analysis.

Use the method above, this article got the follow research result:

(1) Finished the job of literature automatic classification, provide the statistic information by category, which makes the reference relationship between literature and bacteria more refined.

(2) Generate a strain related dictionary , completed the scoring algorithm which can measure the relationship between literature and strains, not only can fix the system errors, and laid the foundation of sorting method.

(3) With a theme-based retrieval, researchers can quickly get the literature resources of different strains.

(4) Through the analysis of the relationship between the strains, the symbiotic relationship between the strains has powerful evidence.

Over all, each system module has been successfully completed and integrated into the original system platform, and provided more comprehensive reference information for uses, but in some details and the performance optimization, there is a further work to be perfected.

Keywords: Text Mining, preprocessing, Text Classification, Text Clustering, Retrieval

目 录

图目录

表目录

第一章 引言

1.1 研究背景及意义

随着信息爆炸时代的到来，各种各样的文本文献越来越多。从 Web 页面，期刊，会议文献，技术档案到电子邮件消息，都数不胜数，为了在这些文献中准确找到用户需要的资料，各大商业搜索引擎应运而生，比如 Google, Yahoo，国内也出现了一些有名的搜索引擎，比如百度，有道等。Google 在 2008 年称已经发现了 1T 的链接，也就是 1 万个亿的链接，并且每天链接增长速度是几十亿，而且还不包括不能被网络爬虫所抓取的那一部分页面。 在搜索引擎的帮助之下，用户能够从这些杂乱无章的网络环境中获取部分有用的信息，但由于信息总量确实太为庞大，并且搜索引擎只是对关键字进行检索，并不考虑语义环境，所以就产生了两个问题，其一就是不是所有的相关的结果都能被搜索引擎找出来，其二，搜索引擎返回的结果并不都是相关的，这两个标准就是用来评价搜索引擎质量的所谓查全率与查准率。因此，对文本进行语义分析就产生了实际的需要，这也是当前最新一代搜索引擎的研究方向，使得计算机能尽量理解人类语言，然后给出用户最想要的搜索结果。

具体到各个专业领域中，相关的资料与文献也是数不胜数，也出现一些针对某些专业领域的搜索引擎，又称为垂直搜索引擎，比如对于旅游，房地产信息等领域。又比如专门针对新闻的搜索引擎来说，通常需要把结果列表按政治，经济，体育等类别进行分类，产生了文本自动分类的需要。

因此逐渐诞生了文本挖掘技术，旨在利用计算机对文本信息进行自动挖掘分析，比如对文本自动分类，聚类，抽取文本中的实体，根据已有文本信息做出趋势预测等技术。文本挖掘借鉴了很多数据挖掘[1]的相关技术，但是二者又有明显的差别，数据挖掘的对象是存储在数据库中的高度结构化的数据信息，文本挖掘的对象是半结构化甚至是无结构的文本，比如 Web 页面或者纯文本数据。正因为如此，数据挖掘算法无法在文本挖掘中直接拿过来应用。

目前，互联网上的大部分信息都是非结构化的文本，比如网页，会议记录，各种存档等文件，存储于数据库中的结构化信息只是少部分。要想让计算机对无结构信息的文本进行分析处理，首要任务就是对其进行格式化，抽取重要的信息，过滤掉重要性不大的信息。文本挖掘的应用范围十分广泛，尤其是在面向各个专业领域的文献挖掘中，具有重要的科研与商业价值。比如 Feldman[2]对新闻信息的分析，Wuthrich[3]对股票的分析，都为决策者提供了丰

富的参考信息，在一定程度上驱动了文本挖掘技术的应用。

1.2 研究现状

因为文本挖掘是一个很宽泛的研究领域，它涉及到了数据挖掘，机器学习(Machine Learning)，自然语言处理(Natural Language Processing, NLP)，信息检索(Information Retrieval, IR)等多个相关学科的知识，所以文本挖掘的发展显得似乎有些零碎与无组织，缺乏统一性，但总体上看，文本挖掘的过程可以概括为三个过程。第一是文本预处理，第二是挖掘分析，最后是挖掘结果的展示[4]。其中挖掘分析技术[5]又包括了文本分类，聚类，以及文本中隐藏的趋势的探究过程。

对文本的预处理主要是因为纯文本信息结构散乱，无法直接用程序处理，预处理就是从文本中进行信息提取的过程。在预处理的基础之上，研究人员逐渐提出了结构化信息抽取技术，包括实体抽取与关键信息抽取，文本自动摘要生成技术。通常在文本中包含了丰富的结构化信息，比如电话号码，地址，电子邮箱等，这些实体通常都由多个词语组成，在很多实际的任务中都有识别文本中这些实体的要求，常用的实体识别的技术包括基于词典与基于固定模式的方法。例如在本课题的研究中，菌种就是一种实体。关键信息的抽取方法相对比较复杂，因为如何度量信息的重要程度没有固定的标准。

文本的分类技术是一个让计算机来代替人为的分类过程，给定一些标准的已知类别信息的文本，可以设计相关的算法，让计算机学习文本的特征，然后对未知文本打类别标签。当前分类技术已经比较成熟，分类的模型也比较多样化，适用于不同的应用场景。

聚类是一个天然的过程，因为不需要事先人为预定义类别，也不用给出标准的训练文本集合，程序会自动计算文献之间的相似度，并且把相似的文献聚合成为一个类别。

国内对于文本挖掘的研究起步比较晚，并且难度比较大，还处于发展阶段。由于中文与拉丁语系有巨大的差异，中文的自然语言处理技术尚不成熟，尤其是中文的分词问题，以及中文的语法灵活多样，使得对中文的预处理十分困难，中文词法分析是中文信息处理的基础与关键。

当前很多文本挖掘技术都已经进入商业化阶段，出现一些商用的文本挖掘产品，比如 Intelligent Miner[6]等工具。

1.3 研究目标

文本挖掘的对象主要是各种各样的纯文本信息，然后应用挖掘算法分析内在特征。在本文中，这些文本是指生物资源相关的文献，这些文献由世界微生物数据中心从各大网站与杂志，会议上搜集而来，并分析了这些文献资源中的菌种信息，供生物学的研究人员参考。本文中利用到的文本是这些生物资源文献，因此在后文中，将不再区别文档，文本，文献，它们都是同一个意思，实际指的是系统库中的生物资源文献。

本课题的主要研究目标是从文本语义的角度去分析这些生物资源类的文献与微生物学中菌种的相关性。对系统中的微生物资源文献，从自然语言角度，利用计算机抽取文献的主题并进行格式化存储。建立在此基础之上，对文献资源进行细化分类，完成各类别之间的菌种统计；然后用 DBSCAN 算法进行分析，获得主题词典，对文献打分，最后完成对文献的检索。

1.4 本文组织

全文共分为八个章节，按各自的内容不同与功能模块如下进行组织：

第一章：主要介绍本课题的技术背景，国内外的研究现状，所采用的方法线路。

第二章：介绍了本课题所依托的项目基础，当前已有的数据资源情况，已经完成的分析，还有当前的系统不足之处，并提出了解决方案的构想，所要完成的功能点。

第三章：介绍对文献进行预处理的方法，生成文献资源的主题向量，存储并为后文计算使用。

第四章：介绍对生物资源文献进行自动分类的办法，两种分类算法的效率比较，对文献分类的结果。

第五章：介绍生成主题词典的方法，利用主题词典对文献资源打分，评价与菌种之间的相关程度。

第六章：介绍对文献索引的办法，检索算法，以及如何与前端 Web 进行通信的方法。

第七章：介绍了菌种之间的共现关系挖掘办法。

第八章：总结全文工作情况，并提出了存在的不足之处，以及后续需要完善的方面。

第二章 课题系统介绍

2.1 项目背景

本课题的项目背景基于中国科学院微生物所的生物资源引用分析平台(Analyzer of Bio-resource Citations，ABC)，见 http://abc.wfcc.info。 如下图是ABC 的主页面，ABC 平台是 WDCM(World Data Centre for Microorganisms，世界微生物数据中心)其中的子课题。当前 WDCM 下属了近 600 个菌种保藏中心，分布在全世界各个国家，比较知名的比如有 ATCC(美国)，JCM(日本)，CGMCC(中国，微生物所)。各保藏中心都有保藏了相关的菌种以提供给科学家做科学研究。

图表 2-1 ABC 主页面

在 ABC 中搜集了大量的生物资源相关的论文，专利，基因组，核酸序列，并挖掘分析了这些文献与微生物菌种之间存在的引用关系。比如某篇文献中可能提到了 CGMCC 3.3928(微生物所，黑曲霉)这个菌种，我们就认为该文献与该菌种相关联。截止到目前(2013.03)为止，ABC 中收录的生物资源类文献数量已

达到 3219497 篇，其中有 173606 篇中都引用了相关的菌种，引用次数达 128381 次。在专利文献方面，当前与菌种信息相关的专利文献达到 12349 篇。以此类似，相关的基因组，核酸序列与菌种之间的引用数据在本平台中都有做统计分析。这些资料为生物学家提供了重要的参考信息，可以作为评价各个杂志，各个保藏中心的一个参考依据。

2.2 当前系统不足

当前在分析文献的时候是用基于关键字匹配的模式来实现的，比如对于“Actinobacillus succinogenes ATCC 55618 Fermentation Medium Optimization for the Production of Succinic Acid by Response Surface Methodology”这类似的文本，程序则认为这段文本引用了一个叫做“ATCC 55618”的菌种，因为这符合保藏中心缩写跟上一个编号这种模式。在大部分情况下，这样做没有问题，但有的时候，一个英语缩写有可能具备多种意思，可能不是一个保藏中心。

此外，对于大量文献的检索，其结果排序是一个难题。对于网页搜索，其结果排序的往往会利用到 PageRank 技术，指向一个网站的链接数目越多，则网站显得越重要，结果自然靠前。在纯文献的检索过程中，固然可以根据参考文献引用之间的关系确定文献的重要程度，比如 Google Scolar，但很多时候，系统中收录的文献，其引用指向关系不是特别明确，所以难以使用。

并且，ABC 平台本身是一个生物资源引用平台，主要分析的是文献资源与菌种之间的关系，因此，如何评价文献资源与菌种资源之间的关联性就显得最为重要。

当某篇文献中出现一个菌种时，其文章的主要内容有可能不是与菌种相关的，更有甚者，有时候匹配到的菌种模式并不是一个真的菌种，比如 ACM 2010 这种模式，因此，本文尝试从文章内容的自然语义的角度来度量文献与菌种之间的相关性。简单地说，通过计算机自动抽取论文的主题，去掉冗余信息，进行格式化处理，然后分析这些主题与菌种之间的相关度，作为一个辅助参考。

2.3 研究方案构想

基于以上的分析，利用文本挖掘的技术，提出如下的研究方案：

1) 对文献进行自动主题抽取

在文本挖掘中，如何对文本进行特征抽取是最重要的基础工作，抽取的特征需要能够尽可能的反应原文本的内容，在此，我们利用自然语言处理技术，

对文献中的名词短语(Noun Phrase, NP)进行抽取。比如对于这样的句子，“As we all know ,CGMCC is a culture collection.”，该句子的主题词就是“CGMCC”与“culture collection”。对文献中的所有主题进行抽取，作为文献的特征，然后建立文献的主题向量库。后续的相关挖掘工作都是对这些 NP 进行操作。

2) 对文献资源作分类处理

当前 ABC 系统中收集到的生物资源文献相对来说仍然比较笼统，没有对类别信息进行划分，但实际上微生物只是一个大的领域，除了细菌(菌种)外，还有真菌，真核生物等类群，引用菌种的可能只是其中一个类群，通过分类处理，能看出各类群之间引用菌种的差异性。

3)对文献进行聚类并抽取主题词典

因为目前系统中的文献数量已经上百万，如果对所有的文献都进行聚类操作，涉及的时间将会非常大，所以聚类分析只针对部分文献进行。选择适当的聚类算法，对这些文献进行聚类，然后对类簇的特征进行统计分析，抽取类簇的特征，统计该类簇引用了多少菌种。然后比较，筛选出引用菌种最多的一个类，并从中抽取主题，生成主题词典。

结合主题词典，然后对此文献的内容进行分析，判定该文献是否是微生物领域相关的，从而进行预测该文献引用菌种的可能性。

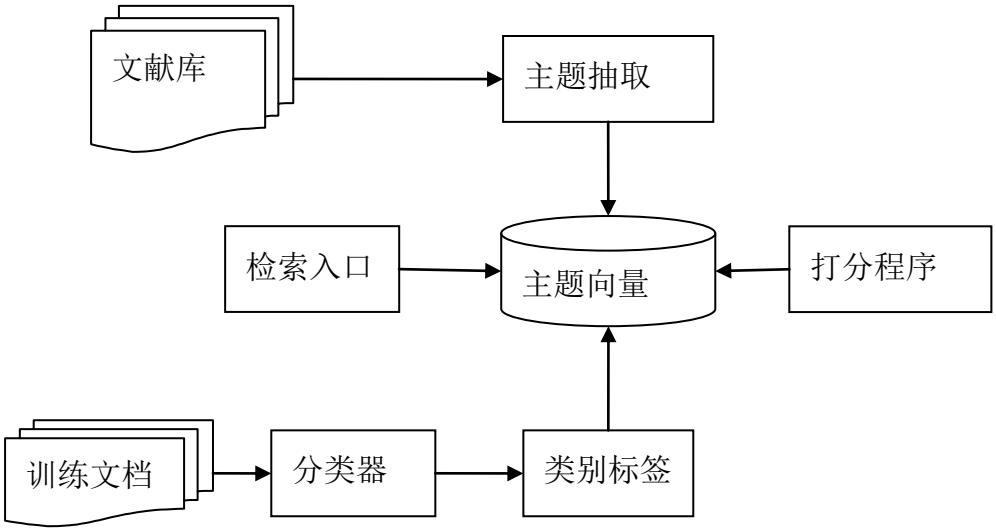
4) 基于主题的检索

为实现对文献资源的查找，可以利用信息检索技术，对这些主题建立倒排索引，提供快速的检索功能。与传统的全文检索技术比较，只索引主题词，大大降低了索引的大小，同时因为各文献的主题已知，方便对检索的结果进行进一步的挖掘工作。

5)菌种相关性分析

根据各个菌种出现在同一文献中的情况，分析菌种之间的相关性。

综合以上方案构想，系统总体的结构图如下所示：



图表 2-2 系统模块图

第三章 文献预处理及主题的抽取

3.1 自然语言处理

3.1.1 自然语言处理介绍

在进行文本挖掘的任务中，往往都会利用到自然语言处理(Natural Language Processing ,NLP)技术。它是语言学中的一个分支，这个分支的主要研究方向是用计算机来对语言进行建立模型。关于 NLP 的相关历史，Bates[7]在 1995 年中的一篇论文有详细的介绍，NLP 的发展受两个方面的推动作用，从科学的角度为了更好的理解人类语言，从实用的角度来说，为了更好的开发智能计算机系统。

NLP 起源于 20 世纪 40 年代末到 50 年代初之间的自动翻译项目[8]，该项目的爱好者认为根据词典进行词与词之间的转换的策略，能够提供粗略的翻译，再利用基本语法技巧，能生成较为精确的结果。但是该项目的结果是让人失望的，使人们认识到，即使是很简单的语言，即使是文盲都能理解的语言，要让计算机来理解也十分的困难，因为人类的常识在这里扮演了重要角色，而且这些日常常识非常难于进行编码或者用算法来进行实现。

从 90 年代开始，一方面出现了大量的词典，另一方面还有各种语料库相继出现，专门用于研究自然语言处理，而且计算机性能也有了很大的提高，使得 NLP 的研究进展有了巨大的突破。当前随着互联网技术与 Web 2.0 的发展，进一步推动了 NLP 的研究需要，各种基于 NLP 的网络产品应运而生，比如广为人知的 Google 翻译，虽然在准确度上有一定的欠缺，但依旧有重要的参考意义。

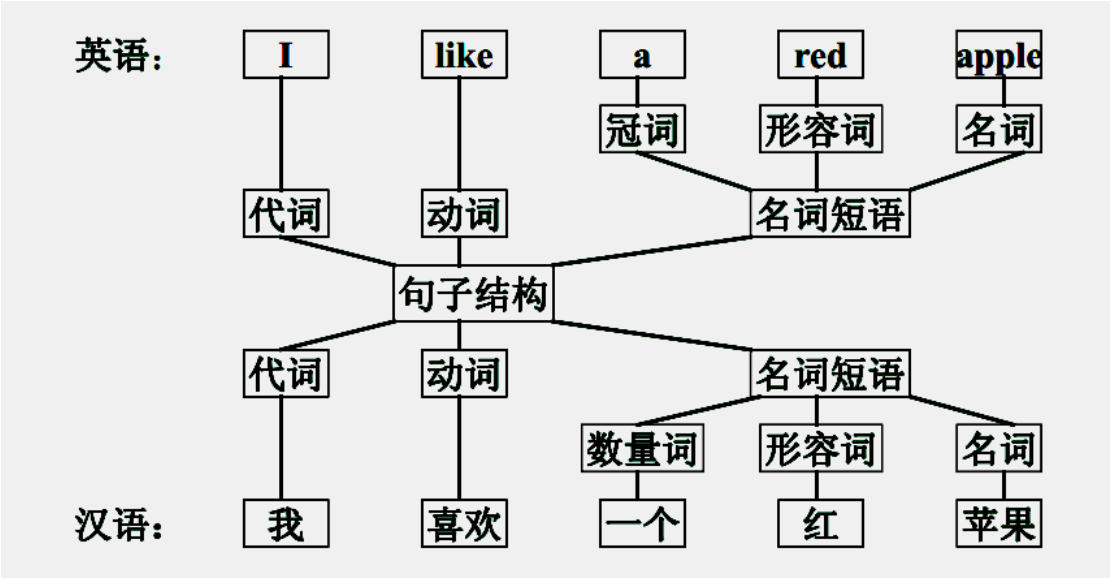
3.1.2 NLP 的相关流程

进行 NLP 主要包括词法分析，语法分析，及语义分析三个步骤。

词法分析：在语言的定义中，往往是由一个一个独立的词语构成的，比如以英语为例，就是由各单词来构成一个完整的句子，所以单词是英语中的最小的结构单位。每个语言都有自己的语法规则，这些词语按照相应的语法规则来形成一个完整的句子。进行词法分析的目的是要把文本首先分割成一个一个的

段落，然后把段落再切割成独立的语句，最后从句子中再抽取出单词出来。不同的语言词法分析方法都各不相同，比如在拉丁语系跟非拉语系之间存在很大的差异，英语中，单词之间用空格断开，划分单词的过程就相对比较容易，从而找出词汇也是比较容易的一件事情。但在汉语中，单词的切分就比较困难了，因为汉语的构成是汉字，再组成词语，所以就需要知道汉语的构词方法，还有一词多义的现象，有可能还需要解决歧义问题。

语法分析：语言的语法用来描述词语如何构成语句。比如对于编程语言来说，编译器就能够根据各语言的语法来进行分析，并生成语法树。不过自然语言的语法规则要复杂得多。经过语法分析，可以清晰的得到句子的成分结构：



图表 3-1　语法分析过程

语义分析：即使是一个句子的语法词法都对，但有时候这种句子不一定具有意义。另外，在不同的上下文环境中，同一个句子的语义也不一定相同。所以语义分析的难度相对比较大。

3.1.3　NLP 与文本挖掘

通常“计算语言学”被某些学者或者其他人当成是 NLP 的一个同义词。Hearst[9]认为，从大规模的文本库中寻找统计学模式并进一步得出 NLP 技术，比如词性标注，词语歧义消除，生成双语词典，这只是 NLP 中的一个分支，这就是说计算语言学也是一种文本挖掘的形式。所以，Hearst 还有 Liddy[10]都认为文本挖掘为 NLP 服务，而不是反过来，并且他们认为元数据是 NLP 与统计分析之间的桥梁。同时他们认为文本挖掘只是全面获取信息的一种方式。

在文本信息分析中的一个主要问题是“复指词的摇晃”，比如代词“这，那，你们，他们”往往都指示了前文中的信息。所以检测复指词搜索它指代的内容也是 NLP 研究中的一个热点。从语言的逻辑上来看，这被称为识别新论点在文本中的起点过程。

在文本挖掘的应用中，NLP 的一个重要作用便是进行主题识别。在科学和技术文档中，往往都自己附带了几个关键词，作为元数据部分，对文档的内容有一个简单的暗示。

主题识别过程大概如下。对于给定的文档，利用标点及词法技巧能获取到一部分候选的短语。然后，对每个短语，分别计算它们的特征，比如使用频率，是否在标题中出现，在训练集合中出现的情况等。一些机器学习算法可以用来对候选短语进行辅助打分，最后抽取得分最高的主题。

3.2 利用 NLTK 进行主题抽取

3.2.1 NLTK 介绍

NLTK(Natural Language Toolkit)最初是在 2001 年宾夕法尼亚大学计算机与信息科学学院的计算语言学课程上开发出来的。之后随着应用需要有大量的开发者加入进来，NLTK 不断地被丰富与发展。现在它已经被很多大学用在课程教学上，并且为很多科研项目服务。NLTK 由 python 开发，之所以会选择 python 是因为 python 具有简单的学习路线，语法也比较简单，并且是完全面向对象的一种编程语言。NLTK 自身附带了丰富的语料库，这些包括 Brown 语料库，Gutenberg 语料库以及常见停用词的语料库等。利用 NLTK，可以完成以下 NLP 相关的基础任务：

1) 断句：把一篇文档切分成为一个个独立的小句子。

2) 切分 token：把句子切分成最小的语义单元即单词。

3) 词形归一化：把单词的不同形式，例如大小写，单复数等形式进行归一化处理。

4) Part-of-speech(POS) Tag：对句子进行语法分析，并且标注单词的词性，比如名词(NN)，动词(VB)，形容词(JJ)等。

5) 语法解析：把句子的语法进行解析，并生成语法树。

围绕 NLP 的主题，NLTK 中还有很多的辅助工具，在此不一一列举。

3.2.2 用 NLTK 进行主题抽取

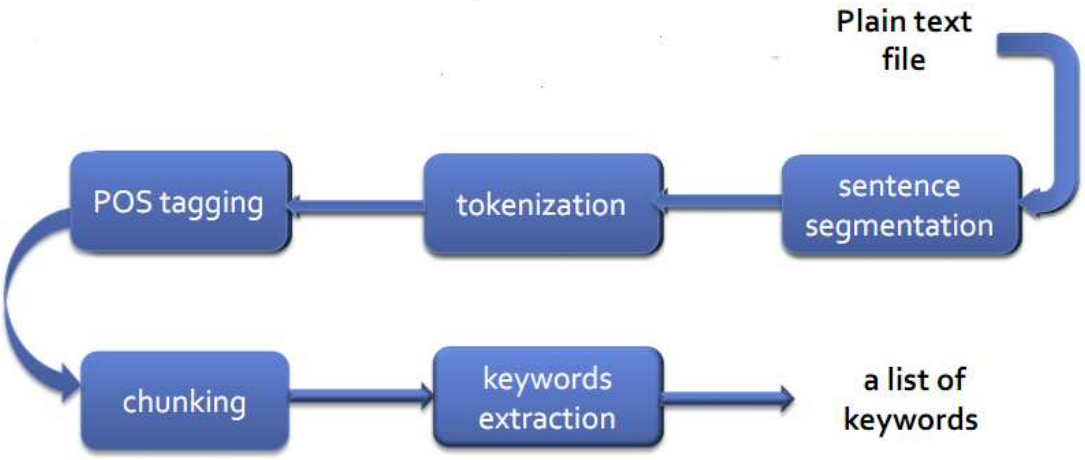
在平常获取信息的过程中，很多时候，我们都对文本信息的主题比较感兴趣，这些主题又通常由某些关键词来表示，通过这些关键词，我们能够快速知道文本的主要内容，是关于哪个领域的，所以在本文中，我们又把这些关键词称为主题词。所以可以看见，大多数学术期刊中，通常都要求作者写上中英文关键词。而在这些主题词中，基本都是一些名词性短语(Noun Phrase, NP)。在一篇学术文献中可以有多个名词短语，但往往只有一部分短语能够代表文献的主题，通常来说这些名词短语的出现频率都会比较高，能够把文章内容特征，相关的研究领域鲜明表示出来。通过对文献主题的自动抽取，不仅可以从直观上了解到文献的所属研究领域，而且便于开展后期的文本挖掘工作，主题抽取是文本挖掘中研究的热点之一，也是本课题中最重要的一步。至于如何抽取主题并形成主题词典，主要的技术方法有[11]：

(1) 从高频被用引文献截取高频短语作为主题;

(2) 利用语义的局部性思想来界定主题;

(3) 以词频的变化率处于突发状态的主题词语作为主题;

在本课题的实现中，采用了第一种方式，也就是高频率出现的名词短语作为主题词，故主要任务是如何自动发现文献中的名词短语(NP)。其中的 NP 可能是单个名词短语，也可能是一个二元组，还有可能是一个三元组。 利用 NLTK 中提供的基本工具，可以达到抽取 NP 的功能，NLTK 全称是 Natural Language Toolkit，是用 python 开发的一个自然语言处理工具包，处理流程如下图所示：



图表 3-2 NLTK 处理流程

整个步骤当中,POS tagging 的步骤最为关键,也就是对单词进行词性标注,进一步才能进行 chunking,抽取名词短语。在 NLTK 中,POS tagging 的方法可以有好几种,甚至可以基于机器学习的方法自行训练一个 Tagger。最简单的 Tagger 对每个词都赋予同样的词性,还有基于正则表达式的 Tagger,根据英语的单词习惯,看单词的后缀进行 Tag,比如以 ed 结尾的判定为一个动词,还有 Lookup Tagger,这种 Tagger 通过统计训练语料库里面最常用的词,最有可能出现的词性是什么,来进行词性标注。在 NLTK 中目前推荐的一个 Tagger 是 Standard Treebank POS tagger, 它来自于宾夕法尼亚的 Treebank 项目,这个 Treebank 经过了人工的注释,并且解析了超过 450 万美式英语单词的语料库,包括 Brown 语料库等。基于默认的这个 Treebank Tagger,可以使得 Tag 的准确率达到 99.57%.

Treebank Tagger 产生的主要词性标注如下图所示:

Tag	Description	Example
CC	conjunction, coordinating	*and, or, but*
CD	cardinal number	*five, three, 13%*
DT	determiner	*the, a, these*
EX	existential there	*there were six boys*
FW	foreign word	*mais*
IN	conjunction, subordinating or preposition	*of, on, before, unless*
JJ	adjective	*nice, easy*
JJR	adjective, comparative	*nicer, easier*
JJS	adjective, superlative	*nicest, easiest*
LS	list item marker	
MD	verb, modal auxillary	*may, should*
NN	noun, singular or mass	*tiger, chair, laughter*
NNS	noun, plural	*tigers, chairs, insects*
NNP	noun, proper singular	*Germany, God, Alice*
NNPS	noun, proper plural	*we met two Christmases ago*
PDT	predeterminer	*both his children*
PRP	pronoun, personal	*me, you, it*
PRP$	pronoun, possessive	*my, your, our*
RB	adverb	*extremely, loudly, hard*
RBR	adverb, comparative	*better*
RBS	adverb, superlative	*best*
RP	adverb, particle	*about, off, up*
SYM	symbol	*%*
TO	infinitival to	*what to do?*
UH	interjection	*oh, oops, gosh*
VB	verb, base form	*think*
VBZ	verb, 3rd person singular present	*she thinks*
VBP	verb, non-3rd person singular present	*I think*
VBD	verb, past tense	*they thought*
VBN	verb, past participle	*a sunken ship*

图表 3-3　Treebank 词性表

POS tagging 完成之后，可以对句子进行 chunking，产生相应的簇，进行 chunking 的时候，产生式如下：

```
Grammer="NP: {<NN>+}   #1 个或者多个普通名词
             {<NNP>+}    #专有名词
             {<JJ>*<NN>} #形容词（任意个）名词（1 个）"
```

对于文档中的每个句子，都进行这样的分析，并进行收集，得到每篇文档的关键词集合，以及相应出现的次数。

在本课题中，所有的文献来自于生物医学领域的各大知名杂志，很多都是 PDF 文件，对于这些 PDF 文件，为了便于处理，都事先转换成为文本文件。然后再接由 NLTK 进行分析。相关的一些处理流程如下：

(1) 文献初步过滤与还原

去除引用信息，行尾的连词符号，及所有的换行。算法如下：

```
def prepaper(paper):
        #first cut off the REFERENCES
        match=re.search("\nREFERENCES\n",paper,re.IGNORECASE)
        if match:
                paper=paper[:match.start()]
        #for those words span two lines flaged as "-"
        paper=paper.replace("-\n","")
        #replace all the "\n"
        paper=paper.replace("\n"," ")
        return paper
```

(2) 句子过滤

经过切分后的句子，有一些可能是不符合语法规定的，比如句子长度太小，或者单词数目太小，以及出现异常单词，应把这些句子过滤去除。

```
def filterSent(sent):
        #sentence length <20 ,pass
        if(len(sent)<20):
                return False
        #diffrent chars<5,exception ,pass
        chars=set(sent)
        if(len(chars)<5):
                return False
        #if tokens <5 ,pass
        tokens=nltk.word_tokenize(sent)
        if(len(tokens)<5):
                return False
        words=[w.lower() for w in tokens]
        #count the words >4
        longwords=[w for w in words if len(w)>4]
        if(len(longwords)<3):
                return False
        #if max length >35 ,pass
        wordlength=[len(word) for word in longwords]
        if (max(wordlength)>35):
                return False
        return True
```

(3) 获取 NP 短语树并从中提取 NP 串

在对句子进行语法分析后，形成语法树，从此语法树中提取出 NP 子树，并生成相应的串。

```
#for a tree,extract the NP(noun phrase),return a list of NP tree
def getNpTreeList(root):
	res=[]
	try:
		root.node
	except AttributeError:
			return res
	else:
		if root.node=='NP':
			res.append(root)
		else:
			for child in root:
				childlist=getNpTreeList(child)
				res+=childlist
	return res
#from the given list of NP tree,make a list of NP
def getNpStrings(nptreelist):
	result=[]
	for nptree in nptreelist:
		wordlist=nptree.leaves()
		wordlist=nltk.tag.untag(wordlist)
		wordstr=" ".join(wordlist)
		result.append(wordstr)
	return result
```

经过 NP 的提取，我们可以获取到每一篇文献的相关主题词集合，从这些主题词中，可以看出这篇文献论述的是什么内容。对每个主题词，还记录了它在文章中出现的频率，以此作为权重的参考因素，方便后续的挖掘工作。

预处理完毕，数据存储格式设计如下：

字段	描述
DocId	文档标志
Unigram	一元组名词，词:词频，例如”strain:22,acid:10,genus:6”
Bingram	二元组名词，词:词频，例如”type genus:1,gene sequence:4”

Trigram	三元组名词，词:词频；例如”cell wall amino:2”

比如，对于下面的一段文本信息[12]

Genetic transformation is a natural process during which foreign DNA enters a cell and integrates into the genome. Apart from its relevance for horizontal gene transfer in nature, transformation is also the cornerstone of today's recombinant gene technology. Despite its importance, relatively little is known about the factors that determine transformation efficiency. We hypothesize that differences in DNA accessibility associated with nucleosome positioning may affect local transformation efficiency. We investigated the landscape of transformation efficiency at various positions in the Saccharomyces cerevisiae genome and correlated these measurements with nucleosome positioning. We find that transformation efficiency shows a highly significant inverse correlation with relative nucleosome density. This correlation was lost when the nucleosome pattern, but not the underlying sequence was changed. Together, our results demonstrate a novel role for nucleosomes and also allow researchers to predict transformation efficiency of a target region and select spots in the genome that are likely to yield higher transformation efficiency.

通过进行 NP 的抽取，我们可以获取到如下的一些主题对该段文本信息进行量化表示：

Unigram	transformation:8;efficiency:6;nucleosome:4;genome:3;correlation:2;gene:2;sequence:1;process:1;accessibility:1;determine:1;technology:1;landscape:1;inverse:1;density:1;transfer:1;cell:1;role:1;relevance:1;pattern:1;today:1;nature:1;importance:1;cerevisiae:1;cornerstone:1;apart:1;novel:1;target:1;region:1
Bingram	transformation efficiency:5;cerevisiae genome:1;novel role:1;nucleosome pattern:1;gene technology:1;gene transfer:1;inverse correlation:1;nucleosome density:1;target region:1
Trigram	determine transformation efficiency:1

3.3 关于存在的问题分析与探讨

NLTK 中提供了相对比较多的一些自然语言处理工具，我们在此用到的仅仅是进行词法语法分析，从而抽取句子中的主题。但是因为这毕竟还只是一个机器工具，在智能方面有一定的缺陷，比如连词符号(son-in-law)，或者单词中间因为编辑原因无故断开(hel lo)，或者英语单词中的多种形式问题，这些细节问题在人为看来很容易区别，但如果在程序中未做处理，可能无法识别这些情

况，所以，在主题抽取的过程中，有可能会存在错误的情况。但是随着文本长度的增加，通常情况下大部分句子结构，大部分的单词都是规则的，所以只要我们抽取的主题涵盖了原文本的大多数主题就算是成功的。对于细节方面的完善，有待进一步加强与提高。

另外，在文本处理操作很耗费 CPU 时间，所以在预处理期间视文本长度大小通常会显得比较慢。此时一方面从软件层面尽量优化处理算法，另一方面在条件许可的情况下扩展到分布式结构中去，利用多台机器协作进行处理。

第四章 生物资源文献自动分类

4.1 分类概述

文本自动分类[13]是把自然语言文档根据它们的内容的不同分派给已经预定义的类别的过程。通常把已定义的类别集合又称为受控词典。文档自动分类技术是图书信息检索中具有悠久历史的传统技术。在此基础上发展起来的LCSH(Library of Congress Subject Headings)[14]就是一个综合性的并广泛使用的受控词典，用来对主题描述符定位类别，当前已经有五卷内容，每卷达 6000 多页，总共有两百万主题描述符。这主要是为了提供一个标准的分类目录，无论什么主题或者语言都已经包含在内。

在文本挖掘中的分类应用起源于 1990 年左右的“知识工程”，致力于人工的分类原则，然后编码实现为一个系统，用来对新文档进行自动分类。从那时起，机器学习技术在文本分类中开始被应用，并且是当前的研究热点。

分类过程中需要预处理操作，那就是特征选择，从训练集合中选择出一部分子集，然后在分类的过程中使用这个子集作为该类特征的代表。通常来说我们可以构造一个特征评估函数，然后根据此函数来选择合适的词语作为该类别的特征。特征评估函数包括以下几种形式[15]：信息增益（IG），词项与类别的互信息（MI），词项的 x^2 统计量（CHI），以及词条的期望交叉熵（CE）等。另外一种常见特征选择方法是基于频率的特征选择方法，也即选择那些在类别中出现频率较高的词语作为特征词，我们在本课题的实现中，就采用了此种办法，如果主题词出现的次数越多，则认为该词的权重越大，越具有代表性。

文本自动分类技术有很多实际的应用，包括文档检索中的索引，自动截取元数据，通过探测文档覆盖的主题消除单词的二义性，组织与维护大量 Web 资源的目录结构。在网站上，把新闻信息按照政治，军事，娱乐，体育等栏目进行自动分类，使得用户能够按照自己的兴趣进行选择性浏览。

4.2 常用分类算法

分类的模型有多种，常见的有基于朴素贝叶斯（Naïve Bayes，NB）模型的分类方法与基于向量空间模型的方法。

4.2.1 基于朴素贝叶斯分类

根据 NB 的学习方法，文档 d 属于类别 c 的概率计算方法如下：

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c)$$

这里，$P(t_k|c)$表示 t_k 出现在类 c 文档中的条件概率。P(c)表示文档 d 出现在类 c 中的先验概率。在 NP 分类中，需要找出文档最可能属于的类别，也就是具有 MAP(maximum a posteriori，最大后验概率)估计值的结果 c_{map}:

$$c_{map} = \arg\max \hat{P}(c|d) = \arg\max \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c)$$

对上式取对数，即得

$$c_{map} = \arg\max \hat{P}(c|d) = \arg\max \left[\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log \hat{P}(t_k|c)\right]$$

对于$\hat{P}(c)$与$\hat{P}(t_k|c)$的估计，可以使用最大似然估计法：

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(t_k|c) = \frac{T_{ct}}{\sum_{t' \in V} T'_{ct'}}$$

这里，N_c 表示 c 类包含的文档数，N 表示训练集合的文档总数。T_{ct} 表示 t 在训练集合中出现的词频，$\sum_{t' \in V} T'_{ct'}$表示文档的总长度。为了去掉零概率，可以加一平滑，即在每个数字上加 1，便有

$$\hat{P}(t_k|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (1 + T'_{ct'})} = \frac{T_{ct} + 1}{\sum_{t' \in V} T'_{ct'} + B}$$

以上在采用了多项式 NB 模型，还有另一种基于贝努利的 NB 模型，贝努利模型中在估计$\hat{P}(t_k|c)$的时候利用类 c 文档中包含 t 的文档数的比率来计算。

4.2.2 基于向量空间模型的分类

另一种主流的分类方法是以向量空间模型(Vector Space Model，VSM)为基础，该模型把文档量化表示成为一个向量，然后利用向量之间的相似性进行计算比较。基于向量空间模型的分类方法以 Rocchio 方法和 kNN 方法为代表。

Rocchio 方法利用质心(centroid)来定义分类边界，一个类 c 的质心指的是类中文档向量的平均向量，即

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

对于给定的一篇文档来说，计算该文档与各质心的距离或者相似度，取其最小的距离或者最大的相似度，并把该文档归入对应的类别。

kNN(k nearest neighbour)方法通过局部信息来确定类的边界。对于一篇给定的测试文档来说，计算并判定与该文档最近的 k 篇文档分别属于哪个类别，然后取出现次数最多的那个类作为该文档的所属的类别。

4.3 分类的实现与评价

4.3.1 建立训练文献库

在 Mesh(Medical Subject Headings)中，对微生物资源文献的分类情况如下图所示

图表 4-1　Mesh 微生物资源分类

Mesh(Medical Subject Headings) 是关于生物医学主题词的一套分类词表，该表由 NLM 在 1960 年编制出版，收集了 1.6 万多个主题词，并设立了各种参照和注释，副主题词 82 个。Mesh 是一部动态词表，每年都有一定的增删变动，它是 NLM 对生物医学文献标引的依据，也是目前最权威最常用的标准生物医学主题词表，可供免费下载。著名的中外生物医学数据库 PubMed, Medline, CBMdisc 等都采用 Mesh 词表作为主题词检索。

在 Mesh 的分类中，在 Microbiology 下面一共分为七个子类别：

Bacteriology：关于细菌学的分类入口；

Environmental Microbiology：有关环境中的微生物的分类入口，其中该类又分为 Air Microbiology(空气微生物)，Food Microbiology(食物微生物)，Soil Microbiology(土壤微生物)，Water Microbiology(水生微生物)，一共四个子类；

Genetics, Microbial：微生物遗传学；

Industrial Microbiology：工业微生物学；

Mycology：真菌学；

Plant Pathology：植物病理学；

Virology：病毒学；

在 PubMed 中，我们为每个类别选取一定数量的文档作为训练文档集合。数据

库设计如下表所示

表格 4-1 训练数据表

Field	Type	Null	Key	Default	Extra
Pmid	int(5)	NO	PRI	NULL	
Title	varchar(512)	NO		NULL	
Abstract	text	NO		NULL	
Category	varchar(128)	NO		NULL	
Title_hash	varchar(50)	NO		NULL	
Np	text	YES		NULL	
Note	varchar(50)	YES		NULL	

Pmid:文档的 pmid 号。

Title:文档标题。

Abstract:摘要数据内容。

Category:文档所属的分类。

Title_hash:标题的哈希值。

Np:文档的主题。对于从 PubMed 上获取的文本，需要利用前文介绍的预处理技术对 Abstract 进行预处理，然后抽取其中的主题，对文档进行量化表示，后文所有的计算工作都是建立在这个 Np 的基础之上。

我们从 PubMed 上分别获取各类别文献 200 篇，其中 100 篇用来作为训练集合，另外 100 篇作为测试集合。

4.3.2 多项式模型的贝叶斯分类

参考朴素贝叶斯的计算公式，计算系统中的文献属于每个类别的概率大小，然后选择较大值，理论上该类就是文献应该划分到的类。部分测试结过如下所示

pmid:22712411	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22712406	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22712403	source class:Genetics,Microbial	bayes class:
pmid:22712400	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22712398	source class:Genetics,Microbial	bayes class:Industrial Microbiology
pmid:22711117	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22711115	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22710156	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22707131	source class:Genetics,Microbial	bayes class:Genetics,Microbial
pmid:22977352	source class:Plant Pathology	bayes class:Plant Pathology
pmid:22973028	source class:Plant Pathology	bayes class:Plant Pathology
pmid:22971235	source class:Plant Pathology	bayes class:Plant Pathology
pmid:22970984	source class:Plant Pathology	bayes class:Plant Pathology
pmid:22969781	source class:Plant Pathology	bayes class:Plant Pathology
pmid:22969430	source class:Plant Pathology	bayes class:

图表 4-2　贝叶斯分类结果

对测试集合中的 700 篇已知类信息的文档进行了贝叶斯分析，大部分的结果都显示正确无误，但还是出现 43 篇文献的类信息无法判断的情况。此时属于每个类别的概率的可能性都为 0，所以没有被分到对应的类别中去。可能的原因与训练样本集合的选择有关系。

可以看出，利用朴素贝叶斯分类算法，其准确率：

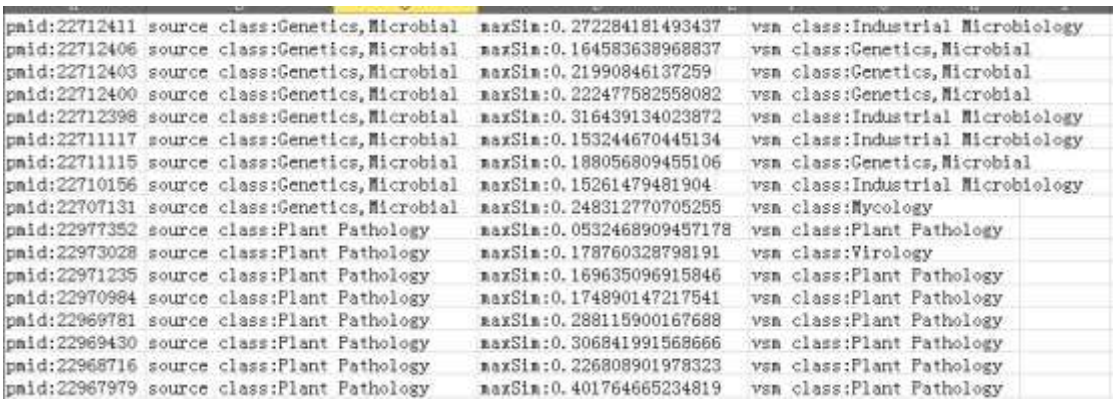
$$P=1-43/700=94.7\%,$$

所以，这是一种准确率比较高的一个分类算法。不过，其缺点在于计算量比较大，对每一个类别，都需要计算一次概率值。在计算后验概率的时候，对每一个 term，需要计算它的贡献大小，也即是需要多次的迭代进行乘除法运算。在实际运行中发现，60s 钟的时间内，只能对 3 篇文档进行分析，平均一篇文档的分析需要 20s。 当然这与机器的性能也有一定的关系，总的来看，进行朴素贝叶斯分类运算量较大，时间开销较大，但准确度比较高。

4.2.3 Rocchio 算法对文献分类

Rocchio 算法是基于向量空间模型的分类算法，在训练集合的建立阶段，直接采用了之前的样本集合。只是在判定具体类别的策略中，由概率值的计算，改为计算未知文档与类的平均向量之间相似程度，这里，平均向量指的是训练文献向量的平均值，然后取最大值的类作为未知文档的类。在同样的硬件设备环境下，对文档进行 Rocchio 分类。

测试结果如下图所示。



pmid:22712411	source class:Genetics,Microbial	maxSim:0.272284181493437	vsm class:Industrial Microbiology
pmid:22712406	source class:Genetics,Microbial	maxSim:0.164583638968837	vsm class:Genetics,Microbial
pmid:22712403	source class:Genetics,Microbial	maxSim:0.21990846137259	vsm class:Genetics,Microbial
pmid:22712400	source class:Genetics,Microbial	maxSim:0.222477682558082	vsm class:Genetics,Microbial
pmid:22712398	source class:Genetics,Microbial	maxSim:0.316439134023872	vsm class:Industrial Microbiology
pmid:22711117	source class:Genetics,Microbial	maxSim:0.153244670445134	vsm class:Industrial Microbiology
pmid:22711115	source class:Genetics,Microbial	maxSim:0.188056809455106	vsm class:Genetics,Microbial
pmid:22710156	source class:Genetics,Microbial	maxSim:0.15261479481904	vsm class:Industrial Microbiology
pmid:22707131	source class:Genetics,Microbial	maxSim:0.248312770705255	vsm class:Mycology
pmid:22977352	source class:Plant Pathology	maxSim:0.0532468909457178	vsm class:Plant Pathology
pmid:22973028	source class:Plant Pathology	maxSim:0.178760328798191	vsm class:Virology
pmid:22971235	source class:Plant Pathology	maxSim:0.169635096915846	vsm class:Plant Pathology
pmid:22970984	source class:Plant Pathology	maxSim:0.174890147217541	vsm class:Plant Pathology
pmid:22969781	source class:Plant Pathology	maxSim:0.288115900167688	vsm class:Plant Pathology
pmid:22969430	source class:Plant Pathology	maxSim:0.306841991568666	vsm class:Plant Pathology
pmid:22968716	source class:Plant Pathology	maxSim:0.226808901978323	vsm class:Plant Pathology
pmid:22967979	source class:Plant Pathology	maxSim:0.401764665234819	vsm class:Plant Pathology

图表 4-3 Rocchio 对文献分类

一共对 700 篇文档进行了分类分析，分类结果正确的占 585 篇，其正确率为

$$P=585/700*100\%=83.5\%$$

可以看出，在准确度方面，基于向量空间模型的分类算法比朴素贝叶斯算法低。但是 VSM 分类算法的时间开销方面比贝叶斯算法要低，基本达到 1s 钟一篇文档的速度（测试环境：Intel(R) Xeon(R) CPU E31220 @ 3.10GHz，4G 内存）。在文档数量比较多的时候，这是比较折中的一个选择。降低了准确度，但减少了程序的运行时间。如果要进一步提高速度，减少程序运行的时间，可以使用多线程技术，通过多个分析例程同时分析，来解决时间开销问题。

本系统中 PubMed 数据源的文献数量当前为 113051 篇，对其分类标注结果如下表所示：

表格 4-2 PubMed 数据源分类结果

Class	number	percent
Bacteriology	19993	17.7%
Environmental Microbiology	7471	6%
Genetics,Microbial	12348	10.9%
Industrial Microbiology	12739	11.3%
Mycology	31907	28.2%
Plant Pathology	5995	5.3%
Virology	6884	6.01%
Other	15714	13.9%

4.4 小结

在文本分类算法中，VSM 模型与贝叶斯模型是两个大的体系，在本文中，两种分类算法都有进行过实验，并比较了二者的差异之处。实验结果表明 VSM 分类算法在效率上要远远的领先于贝叶斯分类算法，在研究中发现贝叶斯分类的时间主要耗费在概率的计算上面，对于每篇文本，需要判断它的每个主题属于某个类别的概率值，然后相乘起来作为后验概率，再结合先验概率，从而判断该文本属于某类别的概率值。这之间存在大量的浮点运算操作。而对于 VSM 分类算法，把每篇文本用一个整数向量来表示，再计算向量之间的相似度，整数运算要比浮点运算快得多。从而 VSM 分类算法在速度方面更胜一筹。但是实验数据表明，VSM 算法的准确率不如贝叶斯算法。

在实际应用中，训练集合的确定对分类结果有重要的影响，训练集合应该能够尽量表现该类文档的主要特征，涵盖该类的主要主题词，并且尽量排除与类别无关的主题。当前的训练集合还有待进一步完善，需要在后面的应用中不断的补充。

第五章 聚类与主题词典的生成

5.1 聚类分析概述

文本聚类分析如何用于各种领域的文本挖掘与信息处理已经有了广泛的研究。最初研究人员探索了如何利用聚类来提高信息检索系统[16]中的精度与召回率[17]，这主要是通过查找文档的最近距离邻居来实现。近年来，聚类技术被提出用来组织检索引擎返回的结果集[18]。研究还发现，对于文献集合来说，对它们进行自然聚类的结果，可以用来辅助训练文本的分类器，比如 Yahoo 就利用到这一技术。

因为聚类的过程是致力于发现自然组合，所以能够反应结果组合中的主题的概览，在本课题中，就是利用到这一特性来进行抽取主题词典。在人工智能的研究领域中，这是一种没有监管的机器学习。一个好的聚类结果能把文档进行分组，组内的文档之间彼此都相似，而组与组之间的文档不相似。

与分类相区别的是，分类的时候，类的数目包括类的特征都是事先已知的，然后文档被指定归属到某个类中去。相对的，在聚类问题中，类的数目，特征，类之间关系都是未知的。

5.2 文献相似度的度量方法

在使用任何的聚类算法分析之前，都必须要确定如何度量文档之间的相似度[19]，或者说文档之间的距离。度量方法不仅仅要能反应文档之间的相似度大小，而且还需要能够准确的区分文档的特征。对于不同的聚类算法，选择的度量标准可能会有所不同，但计算文档相似度的函数总体上应该满足以下基本特征：

(1) 任意的两篇文档之间的距离都必须要是非负的，也即是 $d(x,y) \geq 0$;

(2) 如果出现两篇文档距离为 0 的特殊情况，当且仅当它们完全相同，也即$d(x,y) = 0 <=> x = y$;

(3) 相似度函数必须要满足对称性质，也就是说 x 到 y 的距离一定等于 y 到 x 的距离，$d(x,y) = d(y,x)$;

(4) 相似度函数必须要满足三角函数规则，即

$$d(x,z) \le d(x,y) + d(y,z)$$

常见的相似度度量方法有：

（1） 欧几里德距离(Euclidean Distance)

欧几里德距离是几何学上面距离的标准度量方法。它是两点之间的简单距离，在二维或三维空间中可以用数学公式轻易求出来，欧几里德距离常被用于聚类分析中作为度量文档之间相似度的参考标准。它满足相似度函数的几个基本特征，也是 K-means 算法中默认采用的相似度函数。给定两篇文档 d_a, d_b，分别用它们的特征词构成的向量进行量化表示，即$\overrightarrow{t_a}$，$\overrightarrow{t_b}$，二者之间的欧几里德距离定义如下：

$$D_E(\overrightarrow{t_a},\overrightarrow{t_b}) = \sqrt{\sum_{t=1}^{n} |w_{t,a} - w_{t,b}|^2}$$

这里特征词集合为 $T=\{t_1,t_2\ldots,t_n\}$, w 表示与之对应的权重。

（2）余弦相似度(Cosine Similarity)

当两篇文档都用特征词向量进行表示的时候，二者之间的相似度直接与这个向量有关，可以用二者夹角的余弦值来进行度量，所以又称为余弦相似度。余弦相似度是广为流行的一种相似度测量方法，广泛被用于信息检索与聚类算法之中。给定两篇文档 d1,d2, 它们的余弦相似度定义如下：

$$\text{CosineSimilarity}(d1,d2) = \cos\theta = \frac{\sum_{k=1}^{n} w_{1k} \times w_{2k}}{\sqrt{(\sum_{k=1}^{n} w_{1k}^2)(\sum_{k=1}^{n} w_{2k}^2)}}$$

(3) Jaccard 相似系数(Jaccard Coefficient)

Jaccard 相似系数主要度量两个对象之间的交集占二者之间并集的比重大小。对于文档来说，Jaccard 相似系数比较共同词汇的权重之和占非共同词的权重和的一个比例。如下所述定义:

$$SIM_J(\overrightarrow{t_a},\overrightarrow{t_b}) = \frac{\overrightarrow{t_a} \times \overrightarrow{t_b}}{|\overrightarrow{t_a}|^2 + |\overrightarrow{t_b}|^2 - \overrightarrow{t_a} \times \overrightarrow{t_b}}$$

Jaccard 相似系数的度量值在 0 到 1 之间，当$\overrightarrow{t_a}$等于$\overrightarrow{t_b}$的时候为 1，如果当$\overrightarrow{t_a}$与

$\overrightarrow{t_b}$不相交的时候为 0。

(4) 皮尔森相关系数(Pearson Correalation Coefficient)

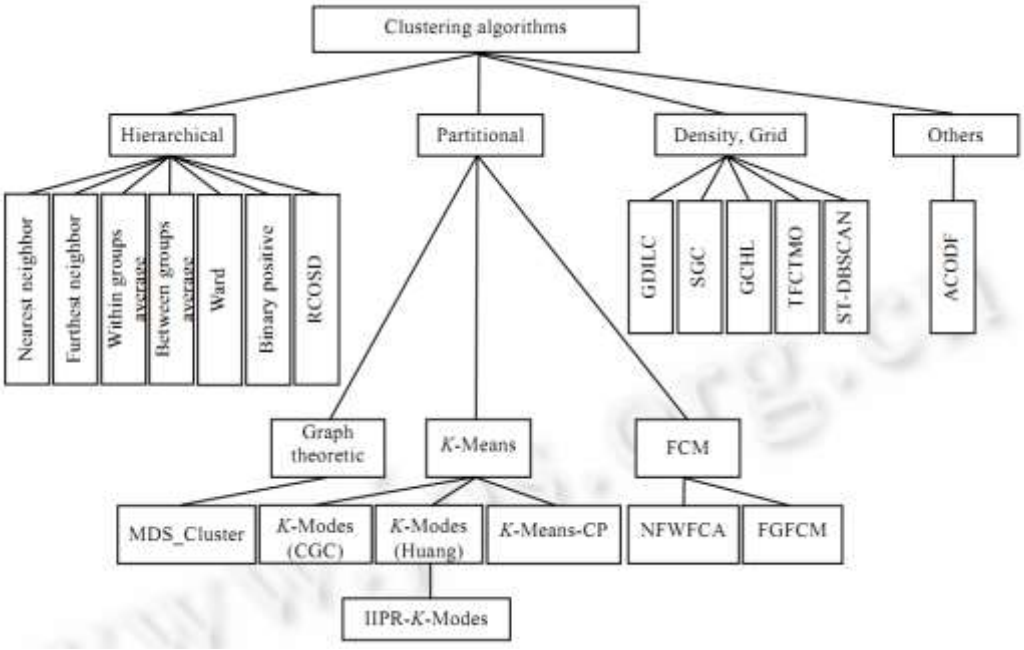
皮尔森相关系数是另一种度量方式，有不同形似的公式，给定词项集合 $T=\{t_1, t_2\cdots, t_n\}$，通常有如下形式：

$$SIM_p(\overrightarrow{t_a},\overrightarrow{t_b}) = \frac{n\sum_{t=1}^{n} w_{t,a} \times w_{t,b} - TF_a \times TF_b}{\sqrt{\left[n\sum_{t=1}^{n} w_{t,a}^2 - TF_a^2\right]\left[n\sum_{t=1}^{n} w_{t,b}^2 - TF_b^2\right]}}$$

这里$TF_a = \sum_{t=1}^{n} w_{t,a}$，$TF_b = \sum_{t=1}^{n} w_{t,b}$，与其他的几种度量方式不同的，皮尔森相关系数的结果范围是-1 到 1 之间，当$\overrightarrow{t_a}$等于$\overrightarrow{t_b}$的时候为 1。

5.3 基本聚类方法

根据孙吉贵[20]等人的研究，可以将聚类算法归为以下几类：



图表 5-1 聚类算法类别

1 基于层次的聚类算法(Hierarchieal)：

层次聚类算法可以按两种方式进行，第一种方式是从下到上，合并最相似的独立的实例，或者从上到下，初始的时候所有对象为一组，然后再细分成小组。以层次凝聚聚类(Hierarchical Agglomerative Clustering)为代表，该过程中每个小的分组都嵌套在一个更大的分组当中，然后不断重复，一直到所有的 N 篇文档合并在一起。根据合并时采用的策略的不同，可以分为单链接，全链接，平均链接三种方式。最后生成一颗系统树，显示了簇与簇之间的关系。

2 基于划分式的聚类算法(Partitional)：

与层次技术不同，划分聚类对数据点进行水平方向划分，以 K 均值算法为代表。1967 年，MacQueen[21]首次提出 K 均值算法（K-means 算法），以此为基础，后来衍生出了很多种类似的分析方法。算法流程如下：

(1)从已知的 n 个数据对象中选出 K 个对象作为初始聚类的中心;

(2)循环流程(3)到(4)，直到每个簇中心都不再变化为止；

(3)把每篇文档都指定到最近的那个中心；

(4)重计算新的类簇中心。

在第(2)步中，需要衡量簇的中心是否发生了变化，通常可以用 RSS(Residual Sum of Squares, 残差平方和)即所有向量到其质心的距离平方和来度量：

$$RSS_k = \sum_{x \in w_k} |\vec{x} - \vec{\mu}(w_k)|^2$$

$$RSS = \sum_{k=1}^{K} RSS_k$$

RSS 是目标函数，通过观察它的值来决定何时停止执行，一般来说，当 RSS 收敛或者低于某指定的阈值的时候结束。

K-means 算法的时间复杂度 O(tKmn)，其中 t 表示为迭代的次数，K 为聚类中心数，也是最终结果集合中类簇的个数，m 为特征属性数目，n 为待分类对象的数目，在通常情况下，K,m,t<<n。当数据量比较大的时候，比层次聚类要快得多。K-means 的不足之处在于初始聚类中心不太好确定，如果选择不当，迭代次数多，而且聚类结果也不好，并且 K-means 算法只适合于凸形数据集。

3 基于密度的聚类(Density – based Clustering)

基于距离进行划分的办法有两个显著的缺点，第一是难以发现不规则的形状，第二是难以确定簇的数量。根据簇性质启发，由于内部对象之间的相似性与密集性，引出了基于密度的聚类办法。

基于密度的聚类利用密度条件把样本集合中相邻的对象聚集成一类。该类算法基于样本空间的点分布情况，也就是样本的密度情况，样本空间被低密度区域划分开，并不需要预先知道类簇的数目。这类算法在法理论上可以发现任何形状的区域，尤其是高密度的区域，过滤低密度的区域。代表算法有：DBSCAN 算法、DENCLUE 算法等。

DBSCAN[22]，[23]算法是一种典型的基于密度的聚类算法，在本课题的相关工作中，主要是利用该算法进行生成主题词典，故对它做详细介绍。

该算法引入了两个参数，Minpts，Eps。

Eps：一个距离值，两个点在该距离范围内，就认为这两个点是相邻的。

Minpts：使点成为核心对象所需要的最小邻居数。

主要的定义如下：

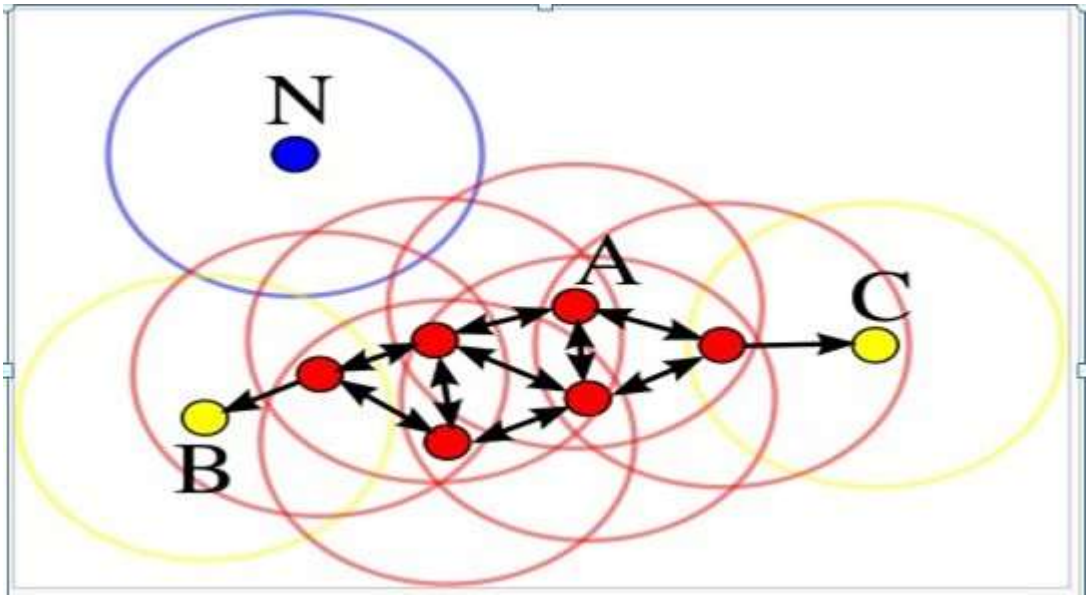
NEps(p): **{q|dist(p,q)<=Eps}**。

核心对象(Core Object): 如果**|Neps(p)|>=Minpts,** 则称 p 为一个核心对象。

如果 q∈NEps(p)且 p 是核心对象,则称 q 是从 p 直接密度可达的。

如果存在类似于 p1->p2，p2->p3,…,->pn 的直接密度可达链，则称 pn 从 p1 可达。

如果存在 o->p,o->q，则称 p,q 是密度相连的。



图表 5-2 核心对象与密度可达

例如在上面的示意图中，点 A 就是一个核心对象，点 B 和点 C 都是从点 A 密度可达的点，从而认为他们密度相连，所以归属于同一个类簇。但对于点 N 来说，它是一个孤立的噪点，既不是一个核心对象，也不能从任何核心对象密度可达（这里 Minpts=3）。

DBSCAN 算法的最终目的是搜索最大的密度相连的点集的过程。

算法伪代码入下：

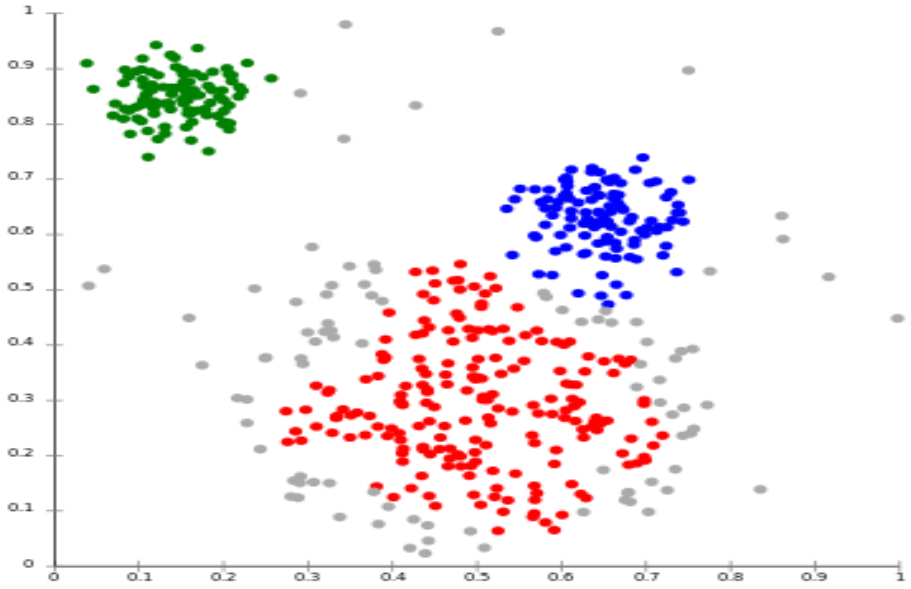
```
算法流程如下：
算法：DBSCAN 聚类算法
输入：文档集合 DocSet，最小相似度 Eps，领域内最少点的个数 Minpts.
输出：各文档 doc 所属的类。
DBSCAN(DocSet,Eps,Minpts)
      C=0
      For each unvisited doc d in DocSet
          d.visited=true
          d.neighborSet=getAllNeighbor(d,Eps)//获取 d 的 Eps 领域内的直
接可达对象
         if count(d.neighborSet) >Minpts
             d.keyPoint=true
             C=next cluster
            keyPointCluster(d,C) //发现核心对象，进行聚类处理
         else
             d.keyPoint=false
             print "d isolate point" //该点为孤立点

    keyPointCluster(d, C)  //对核心对象 d 进行聚类，递规处理
        add d to cluster C
        for each docpoint in d.neighborSet
           if docpoint.visited==false
              docpoint.visited=true
              docpoint.clusterId=C
              print "docpoint :C"
              if docpoint.keyPoint==ture
                 keyPointCluster(docpoint,C)

     getAllNeighbor(d,Eps)  //获取文档 d 的 Eps 领域内的所有直接密度可达点
        Set=empty
        For each doc in DocSet except itself
           Similarity=cosinesimilarity(d,doc)
           If Similarity>Eps
               Add doc to Set
        Return Set
```

通过 DBSCAN 进行聚类分析，我们可以抽取密集区域，过滤点集稀疏的区

域及噪点，降低干扰，这对于挖掘致密区域的模式特征有重要的意义。例如，对于下图对象区域，通过 DBSCAN 聚类，能够抽取出三个代表区域。在具体的应用场合中，可以把注意力集中到这三个区域中，作进一步的统计分析。



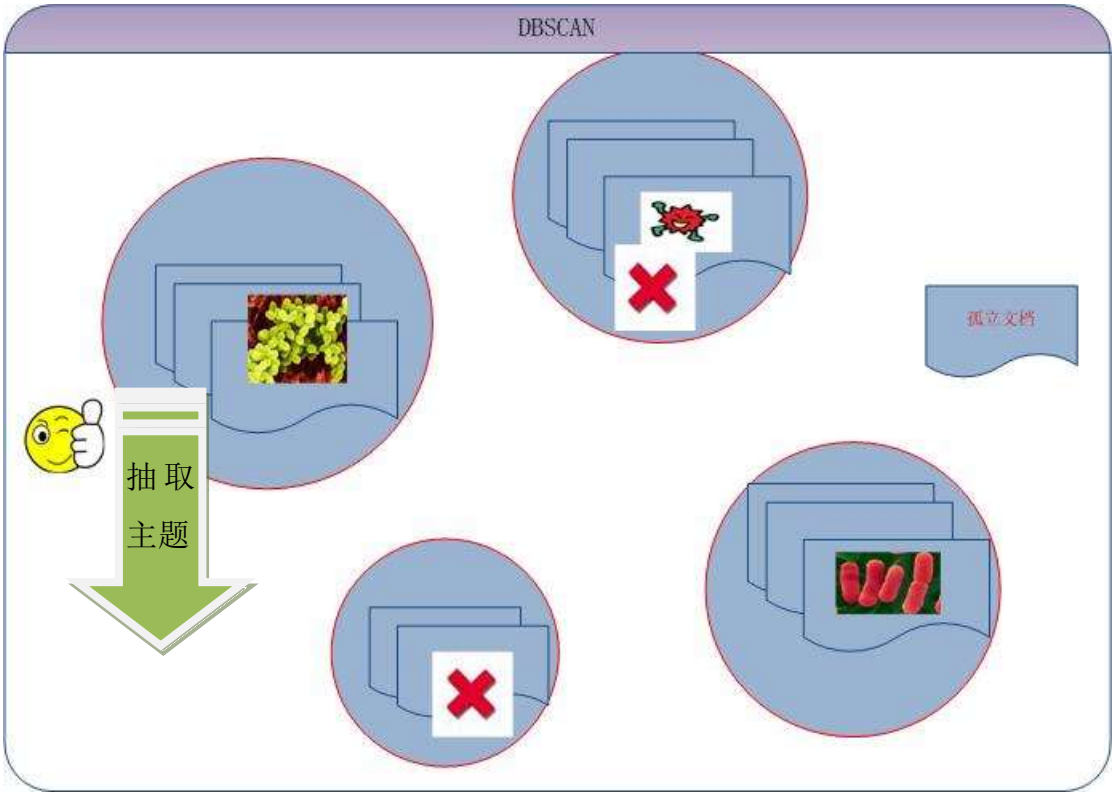
图表 5-3 DBSCAN 聚类分析

4 网格聚类（Grid-based Clustering）

网格聚类与传统的聚类算法有很大不同，它不关心数据点的情况，而是围绕在点周围的值空间。它最大的优势是能极大的降低计算复杂性，尤其是对于数据量比较大的情况。

5.4 主题词典生成策略

在本文中，我们需要分析菌种与主题之间的引用关系。如下图所示，在 ABC 的文献库中，有一些文献中引用了菌种，但更多的文献中并没有引用菌种。我们尝试通过聚类分析，把这些文献分隔成各个区域，挑选菌种出现频率最高的一个区域，再从该区域中抽取主题，对各个主题按频率赋予一定的权重，从而可以为作为对文档评分的标准。



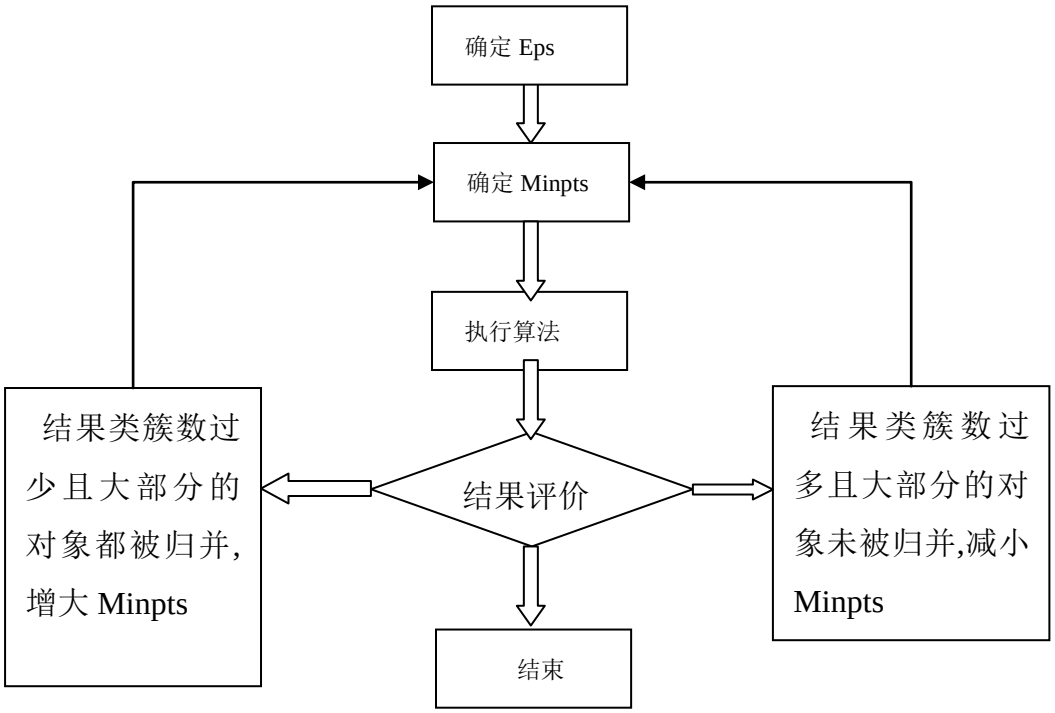
图表 5 -4　高频引用菌种文献抽取

因为基于密度的 DBSCAN 算法具有很好的降噪能力，排除孤立文档，提取高密度的文档区域，非常适合本课题中的任务。

5.4.1 DBSCAN 算法参数确定

DBSCAN 算法需要两个输入参数 Eps, Minpts，参数的选择对聚类分析的结果有很大影响。对于给定的 Eps, 如果 Minpts 取得过小，会导致核心对象数量过于庞大，把相关度不大的对象密度相连接，使得整个区域连成一片，无法区分开来；另一方面，如果是 Minpts 取得过大，核心对象的数量过小，可能使得本应该密度相连的对象没能连接在一起。

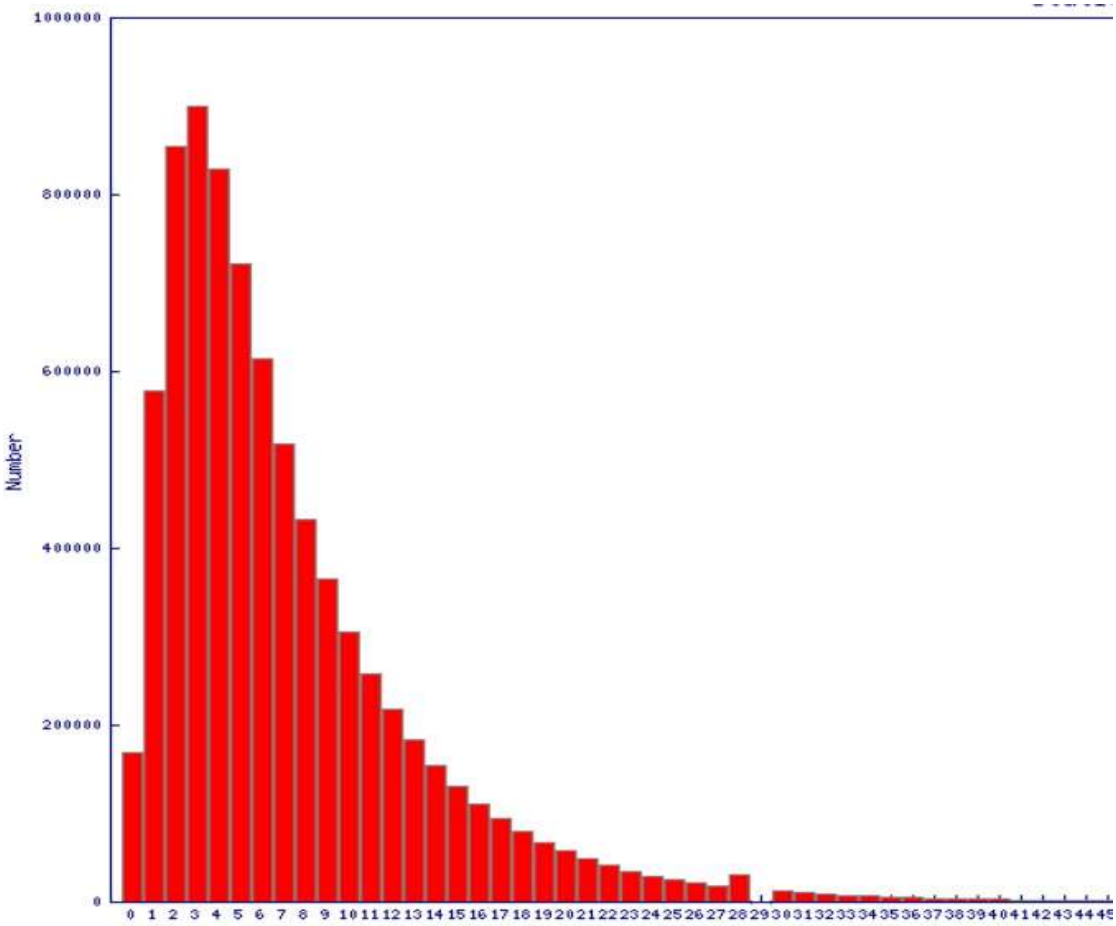
传统上对于 DBSCAN 算法的输入参数选取问题并没有一个标准的流程，主要是根据用户经验进行尝试，先锁定 Eps，再逐步确定 Minpts，根据聚类的结果，如果结果类簇个数过少且大部分的对象都被归并，则应该增大 Minpts；如果结果类簇的个数太多，并且每个类簇下的文档数不是太多的情况下，则需要适当缩小 Minpts。如下图所示步骤：



图表 5-5 DBSCAN 参数确定过程

从中可以看出，为了获取一个较为合适的聚类结果，需要多次调整参数。为了减少调整次数，可以利用统计分析技术，预先确定出 Eps 的范围，再分析 Minpts 的范围。其思路如下，因为 DBSCAN 算法是一类基于密度的算法，其核心思想就是对象集合的密集程度，因此我们可以先估算一下密度的分布情况（这里的密度在实际的计算中，用相似度来代替）。

把待分析的 N 篇文档，组成一个 N 维方阵，然后计算文档之间的两两相似度，并统计其分布情况。如下图的分布情况图，横标代表了相似度大小百分数（%），纵标数字说明在此相似度下的文档对数量。可以看出，相似度大多分布在 10%以下，10%到 20%之间有一小部分，大于 30%只占极少部分。因此估计 Eps 的范围在 0.3 左右比较合适。估计是否正确，需要结合程序的运行结果进行评估。



图表 5-6 相似度分布图

密度分布图为 Eps 的估计提供了一个参考值。对于一个给定的 Eps 值，分别计算每篇文档对象在这个 Eps 相似度下，有多少文档与它密度可达。可以选择一个居中的值作为参考，根据计算结果再进行调整。例如，在 Eps 取 0.3 的时候，Minpts 取 30，也即是说，如果有 30 篇以上文献与某文献的相似度在 0.3 以上，就算为一个核心对象。

取不同的参数，多次运行程序，实验结果如下：当 Eps 取 0.3，Minpts 取 50 的时候，结果有 3 个类簇，Minpts 增加到 70 的时候，结果只有 2 个类簇；当 Eps 取 0.4，Minpts 取 40 的时候，结果也为 3 个类簇。

DBSCAN 算法的聚类结果与参数的选择有很大影响，在实际的实验中可以不断调整。

5.4.2 主题词典的生成

在本课题中，为了对文档进行打分评价，我们需要生成一个菌种相关的主题词典。当然这个词典可以人工生成，但人工生成存在三个问题：第一是全面

性不够，容易漏词；第二是不方便对主题所占的权重进行打分；第三是人工维护的工作量太大。

为了方便，暂时采用程序生成加人工辅助修正的办法。具体的做法是从引用菌种频率比较高的文献中去抽取相关主题并汇总，并将它的频数作为权重。在当前 ABC 的文献库中，有如下的按杂志统计结果，

Order	Journal	Papers	Strains
1	International journal of systematic and evolutionary microbiology	2074 Papers cite	3310 Strains
2	Applied and environmental microbiology	237 Papers cite	506 Strains
3	Journal of clinical microbiology	85 Papers cite	217 Strains
4	Journal of bacteriology	65 Papers cite	133 Strains
5	Acta biochimica et biophysica: Academiae Scientiarum Hungaricae	62 Papers cite	77 Strains
6	Microbiology (Reading, England)	50 Papers cite	152 Strains
7	Journal of medical microbiology	32 Papers cite	193 Strains
8	Antimicrobial agents and chemotherapy	26 Papers cite	47 Strains
9	The Journal of biological chemistry	20 Papers cite	15 Strains
10	Antonie van Leeuwenhoek	15 Papers cite	23 Strains

MORE

图表 5-7 ABC 杂志菌种统计结果

可以看出，IJSEM（International Journal of Systematic and Evolutionary Microbiology）排在首位，文献数量最多，引用的菌种数目也最多。IJSEM 由普通微生物学学会(SGM)负责出版，它是发布新菌种与酵母的主要社区及记录细菌名称的官方杂志，当前的影响因子 2.268（2011 年）。

在实验中，利用来自 IJSEM 中的 2000 篇文档，对其进行聚类分析，结果得到了三个类簇：

表格 5-1 结果类簇

类簇	文档数量	菌种频数/文档数
Cluster Id0	47	0.51
Cluster Id1	162	0.45
Cluster Id2	103	6.5

实验结果表明，类簇三中的文献引用菌种的频率最高，与菌种的关系最为密切。我们对该簇文献中的主题进行抽取汇总，从而可以得到菌种相关词典。如下表所示：

表格 5-2 主题词典（部分）

strain	3625	temperature	404
acid	1741	atcc	393
growth	1407	optimum	379

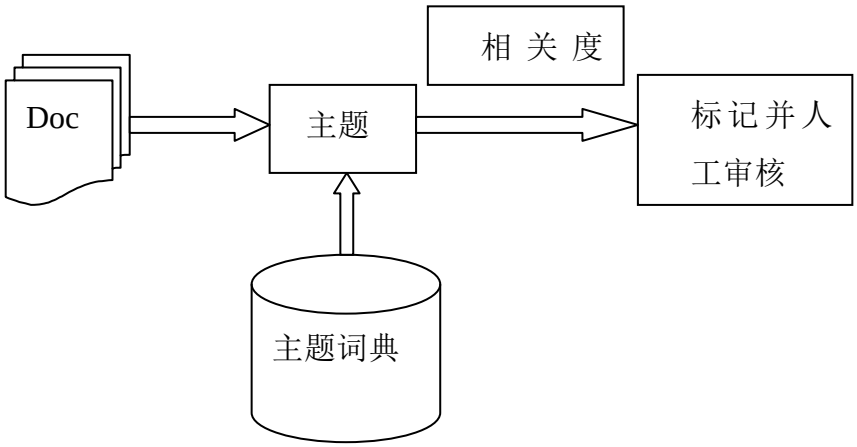
genus	1388	microbiology	370
sequence	1382	evolutionary	370
fatty	1302	carbon	312
gene	1298	bacteria	303
type	1244	cell	302
medium	777	tree	290
analysis	720	production	287
novel	551	description	281
international	497	family	254
journal	441	systematic	244
……		……	

5.4.3 利用主题词典修正系统错误

在 ABC 系统平台的菌种分析过程中，挖掘文献中菌种的时候，利用到的策略是 PERL 正则表达式匹配的过程。在文献中，利用到这样的正则模型“/(?<=[\W\d])($cc)\s+([\w\-\.]+\w)T?/”，在这里$cc(culture collection)表示一个保藏中心的缩写，比如 CGMCC（China General Microbiological Culture Collection，中国科学院微生物所），ATCC（American Type Culture Collection，美国模式培养物集存库），后面跟若干空白字符，再接上菌种编号，就可以提出菌种出来，比如 ATCC 25922（大肠埃希菌）这样的模式。

当前在 ABC 系统平台的数据中，收录了世界范围内 596 个菌种保藏中心的数据，这些保藏中心都由 WDCM(World Data Centre for Microorganisms，世界微生物数据中心)进行了统一的编号，并赋予了相应的简称缩写。这对于类似 ATCC，CGMCC 这样的知名保藏中心没有什么问题，但是有些保藏中心，比如“University of Queensland Microbial Culture Collection”，WDCM 对其的简称缩写是 ACM，这个缩写在英文中可能表示美国计算机协会，又例如“Centraalbureau voor Schimmelcultures, Fungal and Yeast Collection 其缩写为 CBS，也可能代表哥伦比亚广播公司。

所以通过正则表达式进行匹配出来的结果，有可能出现错误，所表示的并不是一个菌种，虽然目前为止发现错误的情况不是太多。利用前文生成的主题词典，可以对缩写有歧义的保藏中心的菌种进行全文分析，确定全文内容与菌种的相关度，对于相关度较小的文献，标记出来待人工进行审核，修正错误的模式，从而提高了系统的准确度。相关流程如下所示：



图表 5-8 错误修正流程

在实际的应用中发现，对于有二义性的保藏中心缩写，如果计算出的相关度值为零，此时符合菌种模式的字符串很可能就不是一个菌种。但因为这里面涉及到的人工审核的工作量比较大，数据还有待进行搜集，具体的效果需要对这些数据进行统计才能确定。

第六章 文献检索的实现

在 ABC 的文献数据库中，基于前期的菌种提取技术，对于给定的菌种，可以获知引用该菌种的文献情况。对于给定的文献，当然也能获知其中出现的菌种，这些主要都是利用数据库技术来实现，在数据库中有文献编号与菌种编号之间的关系模式。但在更多的情况下，用户并不能直接获取到文献在数据库中的编号情况，只能通过用户输入文档中的关键词来查找文献。为了实现这一功能，需要利用到信息检索技术[24]来完成。

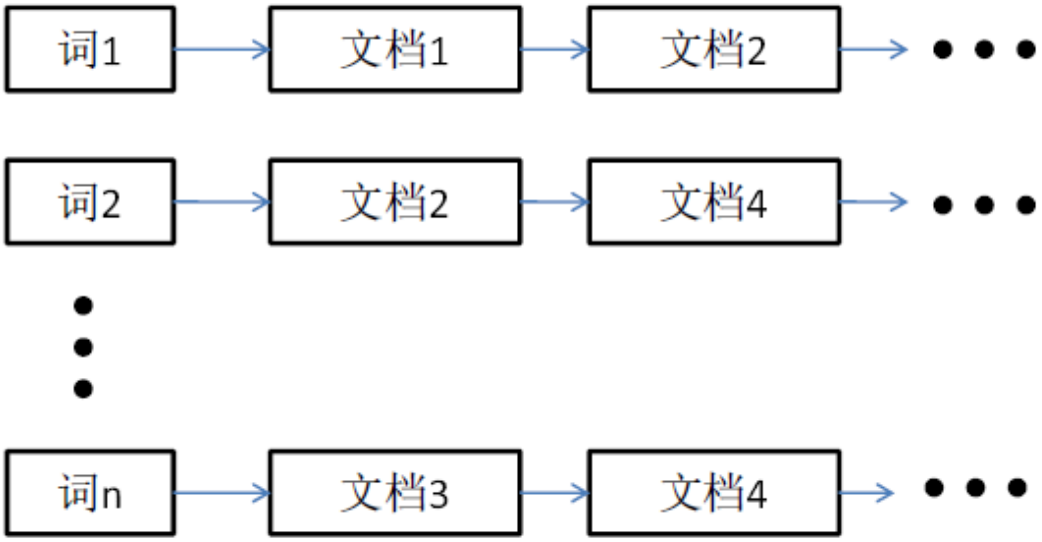
当前信息检索技术已经比较成熟，也出现了一些开源的检索工具包，以 Lucene 为代表。但是当前整个系统构架采用的 LAMP 模式，如果引入 Lucene 工具包，搭建 Java 运行环境，会对当前系统造成很大的额外负担。考虑到之前的预处理过程已经对文献进行了主题的抽取，在此基础上，对这些主题建立索引，实现对文献的检索。一方面能减小倒排索引的大小，另一方面也能降低检索的时间，当然，这样的不足在于可能会漏掉一些关键词。

6.1 检索的实现原理

6.1.1 倒排索引

在检索引擎的具体实现中，通常需要建立检索的模型，信息检索的模型[25]是指对文档进行表示的方法，及如何对用户的输入进行处理，从而找出与此相关的文档。常见的模型有向量空间模型，概率模型[26]等。

与此相应的，对文献信息进行索引的方式也各不相同，常用的是“倒排文档索引”技术。倒排文档索引可以被看成一个由链表组成的数组，每个链表的表头包含关键词信息，其后续单元则包括所有出现过这个关键词的文档标号，以及一些其他的辅助信息，比如词频，位置等。这些信息对检索结果的排序有重要的作用。



图表 6-1 倒排索引示意图

在处理查询请处的时候，对词典进行搜索，取得相应的倒排列表，对倒排表中的文档，计算它们与查询的相似情况，依此对结果进行排序。

6.1.2 检索结果评价

信息检索中评价检索结果最常用的两个指标是正确率和召回率。

正确率(Precision, P)是指查询返回的列表集合中相关的文献所占的百分比例，如下公式所示：

$$\text{Precision} = \frac{\text{返回结果中相关文档数}}{\text{返回结果总数}} = \text{P(relevent|retrieved)}$$

召回率(Recall, R)是指查询返回的结果列表中相关的文献占到文献库中所有相关文献的百分比，如下公式：

$$\text{Recall} = \frac{\text{返回结果中相关文档数}}{\text{所有相关文档数目}} = \text{P(retrieved|relevant)}$$

很多情况下，用户只希望第一页的所有结果都是相关的，也就是说，他们非常注重高的正确率。但是也有一些人，比如在寻找相关学术资料的时候，需要比较高的召回率，才能尽量找全相关领域的文献资料，故这个指标也不可忽略。于是存在一个折中的方案，F 值(F measure)同时融合了正确率与召回率两个指标，它是正确率与召回率的调和平均值，定义如下：

$$F=\frac{1}{\alpha^{\frac{1}{P}}+(1-\alpha)^{\frac{1}{R}}}=\frac{PR(1+\beta^2)}{R+P\beta^2}$$

其中，$\beta^2=\frac{1-\alpha}{\alpha},\alpha\in[0,1],\beta^2\in[0,\infty]$。默认情况下，平衡 F 值中正确率与召回率相等，即$\alpha=1/2$ 或者$\beta=1$，当$\beta=1$的时候，上式简化为

$$F_{\beta=1}=\frac{2PR}{P+R}$$

但是等权重并不是唯一选择，$\beta<1$表示强调正确率，$\beta>1$表示强调召回率。

6.2 索引结构设计

6.2.1 索引数据类型

为了充分减少索引文件所占用的磁盘空间，在生成索引文件的时候，主要利用到以下数据类型：

Byte: 一个 Byte 占用一个字节的磁盘空间，也即 8 位二进制位。

Int: 一个 Int 类型占用四个字节的磁盘空间，也即 32 位二进制位。

Long: 一个 Long 类型占用八个字节的磁盘空间，也即 64 位二进制位。

String: String 类型占用的磁盘空间由此 String 中包含的字符数确定。

Vint: 把一个 Int 型的数据用变长字节来表示，具体占多少字节由此 Int 的值大小决定。

Vlong:把一个普通 Long 型的数据用变长字节来存储，占多少字节由本身值大小决定。

事实上，用 Int 或者 Long 来存储数字类型在功能上就已经够用了，但还要设计出 Vint 与 Vlong，这完全是为了从节省磁盘空间来考虑，对于小于 127 的正整数，只用一个字节就足够了，大于或者等于 128 的需要两个字节或者更多，对于每个 Byte 的 8 位，其中后 7 位表示数值，最高位 1 表示是否还有另一个 Byte，0 表示没有，1 表示有。越前面的 Byte 表示数值的低位，越后面的

Byte 表示数值的高位。例如，150 的二进制表示为 1001,0110，一共需要 8 位，一个 Byte 表示不了，因而要用两个 Byte 来表示，第一个 Byte 表示后 7 位，并在最高位设置 1 来表示后面还有一个 Byte,所以为(1)0010110，第二个 Byte 表示第 8 位，并在最高位设置 0 来表示后面没有其它的 Byte 了，所以为，(0)00000001。

具体情况如下表所示：

表格 6-1 索引文件数据类型

Value	First byte	Second Byte	Third Byte
0	00000000		
1	00000001		
2	00000010		
127	01111111		
128	10000000	00000001	
16, 383	11111111	01111111	
16, 384	10000000	10000000	00000001
16, 385	10000001	10000000	00000001

向索引文件中写入一个 Vint 的算法如下所示：

```
void IndexOutput::writeVint(int32_t value){
        uint32_t ui=value;
        while((ui&~0x7f)!=0){
                this->writeByte((byte)((ui&0x7f)|0x80));
                ui>>=7;
        }
        this->writeByte((byte)ui);
}
```

与此结构相对应，读出一个 Vint 的算法如下所示：

```
int32_t IndexInput::readVint(){
        uint8_t b=readByte();
        int32_t i=b&0x7f;
        for(int32_t j=7;(b&~0x7f)!=0;j+=7){
                b=readByte();
                i|=(b&0x7F)<<j;
        }
        return i;
}
```

对于 Int 类型，在写入的时候，将其拆分成为四个字节写入到索引文件中，算法如下图所示：

```
void IndexOutput::writeInt(int32_t value){
        writeByte((byte)(value>>24));
        writeByte((byte)(value>>16));
        writeByte((byte)(value>>8));
        writeByte((byte)(value));
}
```

6.2.2 缓冲区的设置

缓冲区也叫缓存，驻留在内存中，传统的文件系统在把数据写入磁盘的时候，需要通过 CPU 的中断进行处理，使用缓冲技术之后，直接与内存进行数据交互，只有缓冲满的时候再请求 CPU 处理，如此能极大提高系统的交互效率。当需要把数据写入到文件系统中时，先把数据写入到缓存区，当缓冲区满的时候，再把这段内存块写入到磁盘中去，如此能够大大减少对磁盘访问的次数，降低时间开销。同样，当需要读取数据的时候，可以先从磁盘中读取一块到缓存区，然后再到缓存区中取数据，下一次需要读取磁盘的时候，只要是数据是相邻的，都能在缓冲区中找到。

在大多数操作系统中，系统对文件的读写，都已经使用了缓存区机制，但在建立索引的过程中，需要准确定位文件指针的位置，需要随时对缓冲区的数据进行操作，所以系统中默认的缓冲机制就无法有效利用，需要维护自己的缓存区。

实现写入索引类的设计如下：

```
class IndexOutput{
	private:
		char *m_fileName;
	protected:
		char *m_buffer;
		int m_buffersize;
		long m_bufferStart;
		int m_bufferPos;
		FILE *fp;
		bool isClose;
	public:
		IndexOutput();
		IndexOutput(char *m_file,bool create);
		void writeInt(int32_t value);
		void writeVint(int32_t value);
		void writeByte(byte b);
		void writeString(std::string &str);
		void writeLong(int64_t value);
		void writeVlong(int64_t value);
		void writeChars(const char *s,int64_t start,int64_t
len);
		int64_t getFilePointer();
		void close();
		void flush();
		void flushBuffer();
		virtual ~IndexOutput();
		void seek(int64_t pos);
};
```

6.2.3 索引文件结构

索引文件中最大的单元是段，通常在一个段中包含有多个文档，一个文档中包含多个域信息用来定义存储单位，文档的数据结构在程序中的组织情况如下图所示：

```
class Field{
	public:
		int type;//store type
		int is_store;//if store
		int token;//if cut
		std::string name;//field name
		std::string data;//field data
	public:
		Field();
		virtual ~Field();
};
class Document{
	public:
		std::map<std::string,Field *> m_fieldMap;
	public:
		Document();
		void addField(std::string,Field *field);
		int32_t getIntField(std::string fieldName);
		int64_t getLongField(std::string fieldName);
		std::string getStringField(std::string fieldName);
		bool containField(std::string fileName);
		virtual ~Document();
};
```

索引文件包括如下几个:

Segments 文件：记录索引段和段的状态，是否关闭。

Seg.fmn 文件：记录域信息。

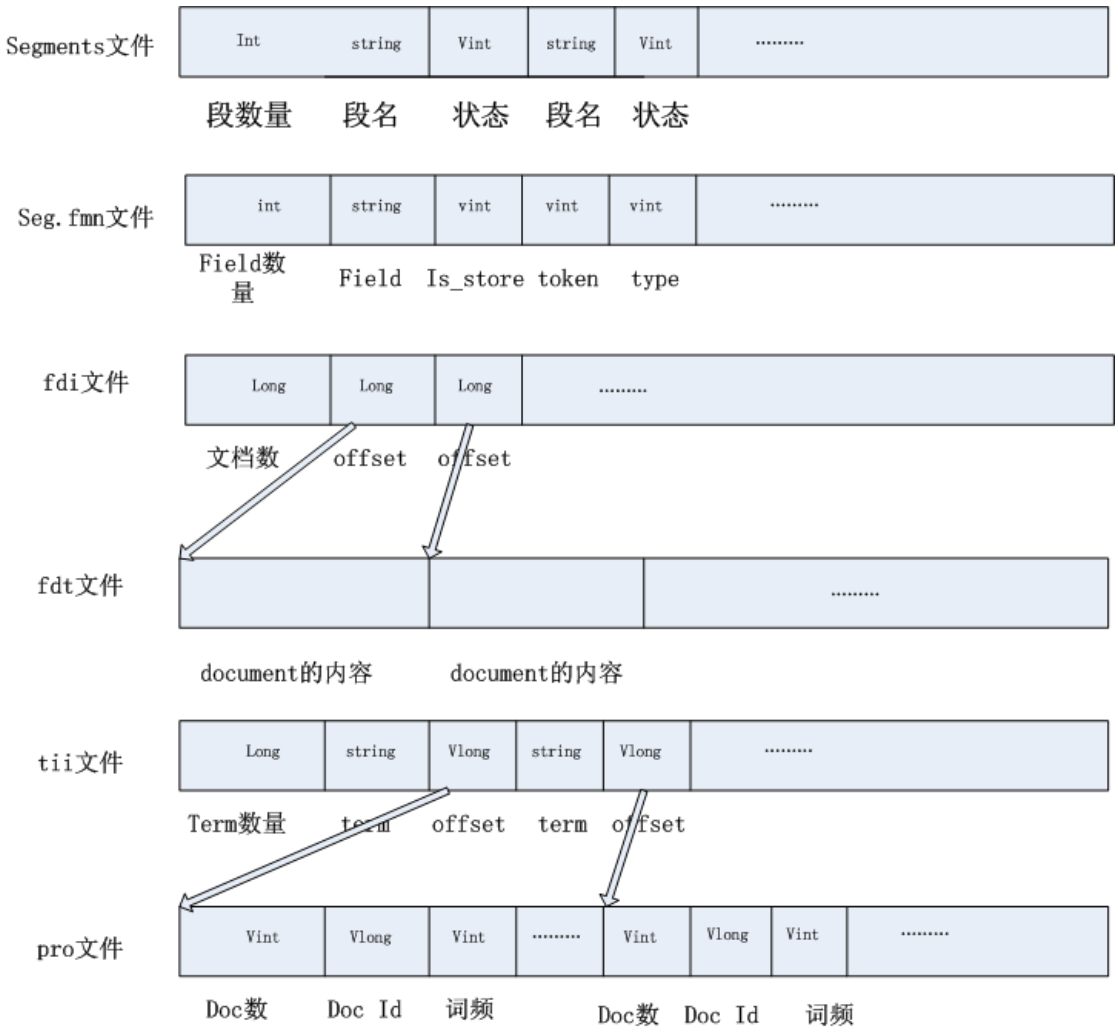
Fdi 文件：记录文档数量及各文档在 fdt 文件中的偏移量。

Fdt 文件：存储文档的内容信息。

Tii 文件：关键字列表文件，存储各关键字及在 pro 文件中的偏移量。

Pro 文件：记录关键字在哪些文档中出现及出现频率。

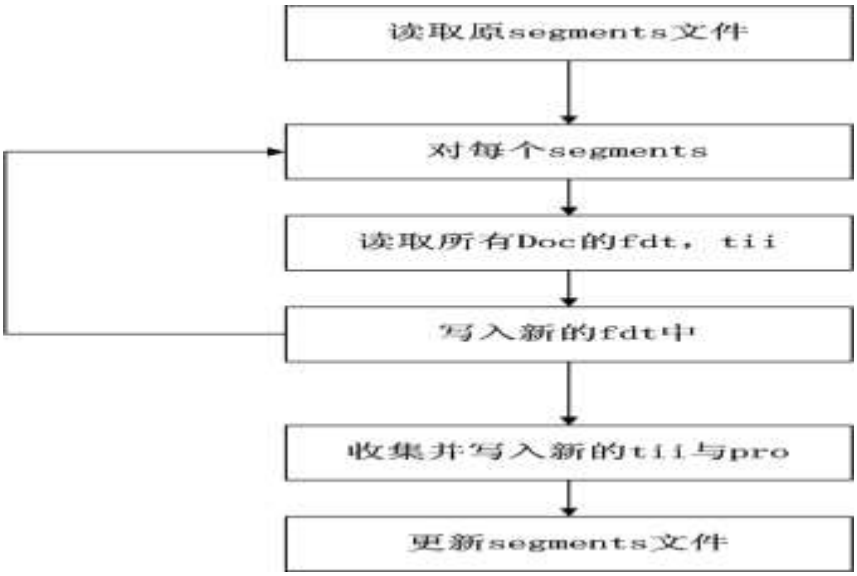
各文件之间的关系如下图所示：



图表 6-2 索引文件结构图

6.2.4 增量索引及索引合并算法

在建立索引的过程中，通常所涉及到的计算量很大，是一个很耗费时间的过程。另外，随着新的文档的加入，往往需要对索引文件进行更新操作，因此如何设计索引生成规则至关紧要。这也是在索引中引用段的作用。每次写索引的时候，都可以重新开辟一个段，最后再合并各个索引段。同理，为了加快建立索引的过程，也可以采用多线程的模式，每个子线程负责写一个索引段，当所有的子线程的任务都结束的时候，再对各个索引段进行一个合并，从而完成索引建立过程。合并索引段的流程如下图所示：



图表 6-3　索引合并流程

6.3 检索算法

检索端的目的就是根据提供的主题词，去读取索引文件，获取匹配到的文献的过程。为了方便，封装了一个 IndexReader 类，作为读取索引的工具类。其主要方法如下：

```
public:
        IndexReader();  /*构造函数*/
        IndexReader(char * indexPath); /*带索引路径的构造函数*/
        int64_t   getNumDoc();/*获取文档数量*/
        int64_t    getTermCount();/*获取词典大小*/
        std::vector<HitDoc> getTermDocs(std::string &term);
        /*对给定的关键词，获取对应的文档*/
        void   getTermDocs(std::string &term,std::vector<HitDoc>
&docVo); /*获取对应文档信息*/
        void    doc(int64_t docId,Document & doc); /*查找单个文档*/
        void   open(const char * indexPath); /*打开索引*/
        virtual   ~IndexReader();
private:
        void loadFmn();/*获取域信息*/
        void loadTerm();/*加载词典*/
        std::string getSegName();/*获取段名*/
```

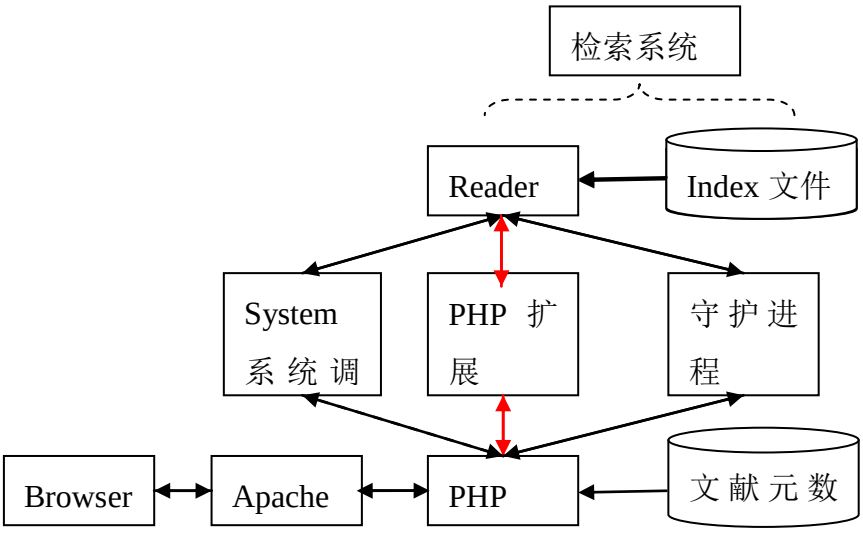
检索调用的接口主要是 getTermDocs 这个函数，有两种调用方式，第一种直接返回一个 HitDoc 向量，第二方式按一个引用参数返回。其实现方式如下：

```
void IndexReader::getTermDocs(std::string
        &term,std::vector<HitDoc> &docVo){
        std::map<std::string,int64_t>::iterator
        it=m_termMap.find(term);  /*在词典中搜索关键词*/
        if(it!=m_termMap.end()){
                int64_t pos=m_termMap[term];  /* 定位偏移量*/
                p_proInput->seek(pos);/*移动文件指针*/
                int32_t count=p_proInput->readVint();/*搜索到的数量*/
                for(int i=0;i<count;i++){  /*依次读次匹配上的文档*/
                        HitDoc doc;
                        int64_t docId=p_proInput->readVlong();
                        int32_t hits=p_proInput->readVint();
                        doc.setDocId(docId);
```

根据检索到的文档集合，可以再次从数据库获取到文档的元数据，包括标题，作者，杂志，年份，还包括文档所属的微生物类别，菌种相关性得分。菌种相关性得分反应了该篇文献与菌种的相关程度，得分较高的排在搜索结果的较前面。另外根据文档所属的类别，在结果中，我们还列出了一部分同类的，菌种相关性得分较高的文档。

6.4 有关 Web 端与后台检索接口讨论

在检索引擎的核心算法中，为了获得较高的效率，我们采用了 C++进行实现。所以在后台的程序中，实际上是一个本地的检索系统。但是当前系统的大平台是一个 Web 服务系统，用户通过浏览器发出请求，服务器接收相关数据，进行查询，然后需要将执行结果反馈给用户。当前系统平台采用的是经典的 LAMP 构架体系，也就是说操作系统 LINUX，web 服务器 Apache，数据库采用 MySQL，Web 脚本预言是 PHP。故在 PHP 与 C++之间需要建立一个交互系统，才能正常通信。



图表 6-4　前端后台交互方式

如上图所示，PHP 脚本语言与检索系统交互方式有三种，第一种方式是即简单的 System 系统调用，第二种方式利用 PHP 扩展来实现直接方式，第三种方式利用守护进程。

(1) 直接利用 System 系统调用

利用 System 系统调用方式是最为简单的交互方式。PHP 作为一种服务器端脚本语言，很多的时候不能直接完成某些任务，需要借助于操作系统提供的外部接口，通过命令行的形式进行呼叫，在 LINUX 系统中，由 SHELL 负责执行，然后结果返回给 PHP 脚本。PHP 中提供了系统库函数实现这个功能，比如 system()函数，执行给定的命令，然后会输出与返回相应的执行结果。但是这里我们并不需要输出结果，只要检索的返回结果，利用 exec 可以返回结果中的最后一行(把结果中的换行符号去掉，就能全部返回)。利用 exec()进行交互的关键代码如下所示：

```
$keyword=$_GET["searchInfo"];  /*获取检索关键词*/
$commandPath="/home/weiyy/textming/search/Search ";/*程序路径*/
$command=$commandPath."'".$keyword."'"; /* 拼接执行命令*/
```

利用这种系统调用的方式，好处是简单，缺点在于，每次调用的系统开销都很大。

(2) 利用 PHP 扩展模块

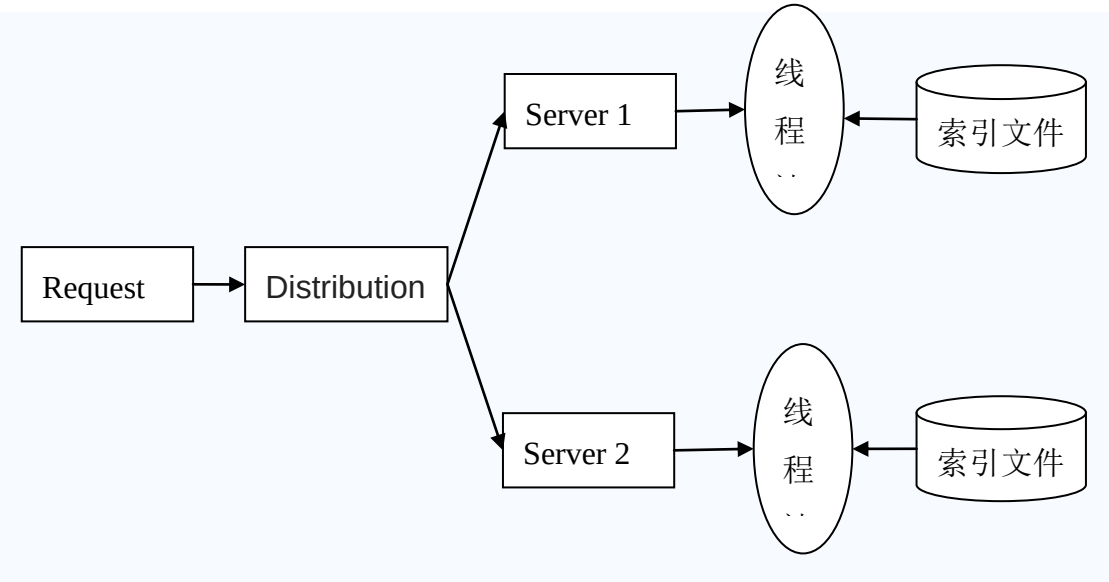
PHP 扩展是效率相对比较高的一种交互方式。PHP 本身是由 C 语言编写的一种脚本语言，因此在与 C 语言交互方面具有先天优势。PHP 源代码在发布的时候就包括了各种各样的 C 语言扩展包，这些扩展包中包含了众多的模块，包括数据库访问模块，网络模块等。还有很多模块比如 GD 图形模块未包含在它的发布包中，但在 PECL 上提供下载。正是因为 PHP 的各种扩展的存在，使得 PHP 的应用特别广泛，极大的提高了 PHP 与 C 语言之间的交互能力。在 PHP 的发布包中的 ext 目录下，有一个 ext_skel 文件，该文件提供了自动生成扩展骨架的功能，极大的减少了扩展编写的流程，只需要补充并添加检索相关的函数接口，然后编译生成模块文件(.so)文件，加载到 PHP 内核中，就可以直接调用检索函数，减少了系统开销。此方式的缺点在于技术性相对比较复杂，需要对 PHP 的内核知识比较熟悉，并且后期更新与维护比较烦琐。

在本项目中，通过利用 PHP 扩展方式，使得时间开销大大减少，最初直接系统调用的单次查询时间在 0.3 秒左右，减少到现在的 0.1s 左右。

(3) 利用后台守护进程

为了方便 PHP 与网络程序进行通信，PHP 在发布的时候提供了 sockets 扩展包(在 ext/sockets 目录下) ，在安装 PHP 的时候只要加上 --enable-socket 参数就可以直接把 sockets 客户端直接编译进 PHP 内核，然后在 PHP 中就可以直接调用 socket 函数与服务器端进行通信。

因此我们只需要把检索程序封装成一个 socket 服务器端程序，然后 PHP 就可以利用自身的 socket 函数连接到服务器，然后传递检索关键字，并从服务器端获取检索的结果。对于大规模的访问，为了提高效率，可以采用分布式服务的构架体系，并在每个服务内部采用线程池的方式，能极大的降低响应时间。如下图所示构架：

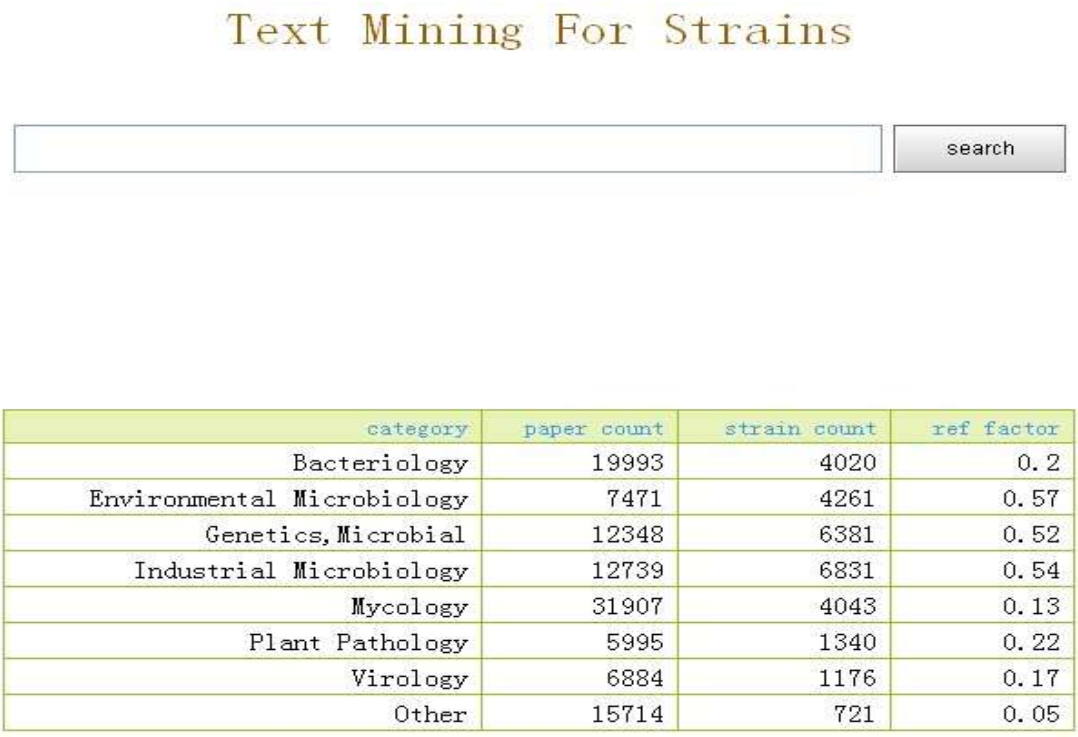


图表 6-5 守护进程方式

该方法的主要优点在于系统的吞吐量较大，响应时间短，在性能上有极大的优势，对于大规模并发访问，可以根据需要添加 Server 的数量，通过一个负载均衡模块把请求进分发到各个服务器中，各服务器都设置有自己的线程池，其中的各个子线程负责索引的查询工作。不过该方法是技术性最为复杂的一种办法，对硬件资源要求比较高，后期的维护也相对比较困难。

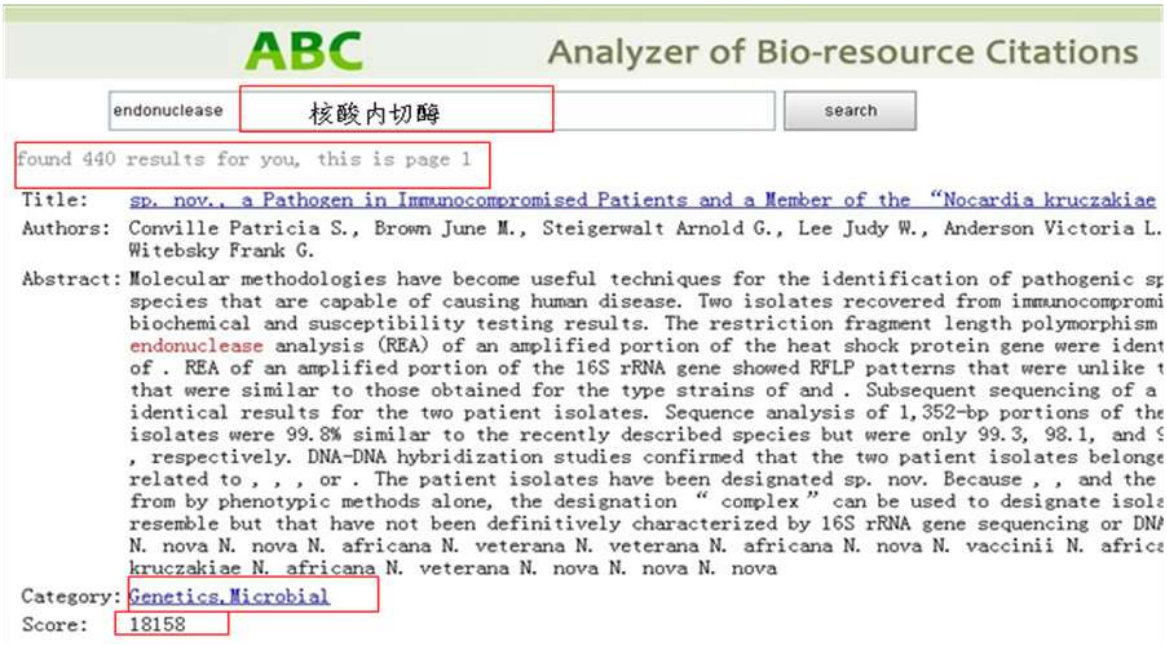
6.5 Web 前端检索界面

前端检索界面如下图所示，在首页面中介绍了当前系统的有关文献分类情况，及各类别下文献资源数量，对菌种引用的次数。根据这二者的比值，计算出一个引用因子。引用因子越大，说明该类文献对菌种的引用就越频繁。从图中可以看出，环境微生物(Environmental Microbiology)的引用因子最大，为0.57，也即是说，该类文献资源引用菌种的概率比较大。在检索框中，用户可以输入检索词，查询相关的文献资源。



图表 6-6 文献检索主页

检索的结果页面如下图所示：



图表 6-7 文献检索结果

这里，以 endonuclease(核酸内切酶)为例，可以查找到相关的文献资源 440

篇。结果中给出了标题，通过此标题可以跳转到详细信息页面，并列出了该文献的作者，摘要信息，关键字加红显示。Category 显示了该文献所属的类别信息。Score 是一个排序依据，Score 的计算利用到了第五章生成的主题词典，Score 值越大，表示该文献与菌种的关系越密切，故排在结果的前面。

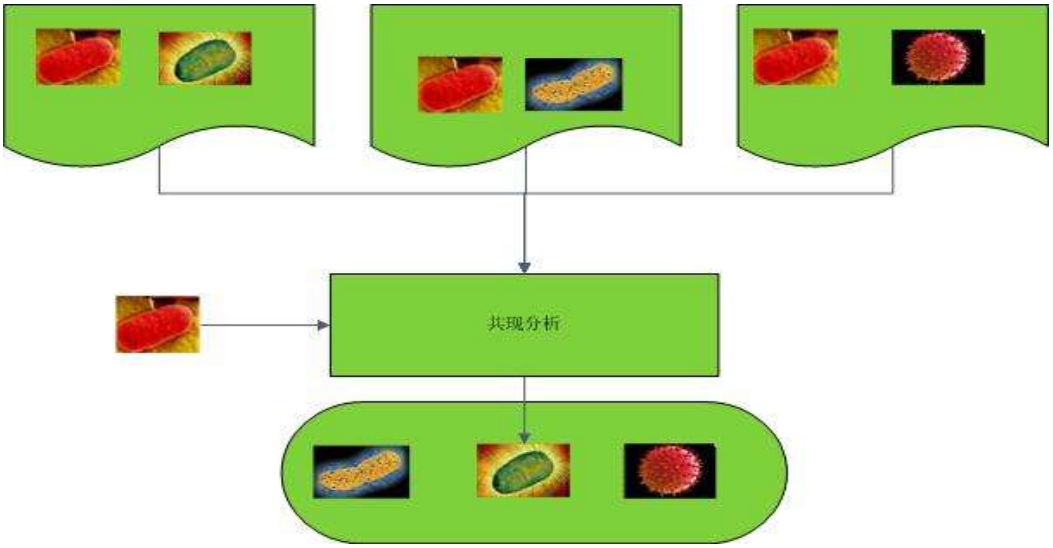
图中显示的文献属于“Genetics, Microbial”这个类别，这是一个超链接，打开它可以对检索结果进行过滤，只显示该类别下的文献资源。

第七章 菌种共现关系挖掘

共现关系的挖掘在数据挖掘中有广泛的应用，比如常见的，在社交网络平台中，根据对同一主题感兴趣的用户情况，发现用户群体并分组。根据用户之间好友关系的情况，向用户推荐好友等。在本课题的主要研究目标是挖掘菌种与文献之间的关系，通过文献资源作为桥梁，能够发现菌种之间的相关关系。

7.1 基本思路

进行菌种共现分析的基本思路如下图所示：



图表 7-1 菌种共现分析

在 ABC 平台的前期工作中，建立了文献与菌种之间的引用链接，由 Strain_Doc 表对这些记录进行查询。字段如下所示：

ReCordId：记录编号 ID；

DocId：系统中文献编号；

Title_Hash：文献标题的哈希值；

Strain_Org：菌种的保藏中心名称缩写；

Strain_Id： 菌种的编号；

Wdcm_Id： 保藏中心的编号；

菌种与文献之间的引用是一种多对多的关系，在某一篇文献当中，可能会引用到多个菌种，同一个菌种也会被多篇文献引用。

对于指定的文献，通过该数据表能迅速找到该文献中已经出现过的菌种，同时，对于某菌种，我们也能得出引用该菌种的所有文献。从文本挖掘的视角来看，菌种其实就是文献中出现的一个个实体。透过文献集合，能够进一步发现菌种与菌种之间的关系。

利用 Strain_Doc 表，首先获取所有的菌种名称。然后对每个菌种，分析它所对应的文献 ID 列表。对每两个菌种，分别扫描获取各自的文献 ID 列表，然后对此列表求交集，表示同时出现在同一篇文献中的情况。如果同时出现在同一篇文献的次数越多，表示这两个菌种之间的关系越密切。

通过对菌种关系之间的分析，对于保藏中心的工作人员来说，可以把关系相近的经常共同出现的菌种放在一起，便于取用，同时对于菌种采购人员，可以考虑一起采购，另外在生物学上，经常在一起出现的菌种之间可能具有对应的生物学意义，供研究人员参考。

7.2 结果展示

经过对 Strain_Doc 表的扫描与分析，可以得到菌种之间的相关关系。数据库表 StrainPair 设计如下表所示：

表格 7-1 菌种共现关系分析

Field	Type	Null	Key	Default	备注
StrainOne	varchar(20)	NO	PRI	NULL	菌种一
StrainTwo	varchar(20)	NO	PRI	NULL	菌种二
OneBrief	varchar(255)	NO		NULL	菌种一简称
OneNo	varchar(255)	NO		NULL	菌种一编号
TwoBrief	varchar(255)	NO		NULL	菌种二简称
TwoNo	varchar(255)	NO		NULL	菌种二编号
DocList	text	YES		NULL	共同文献列表
DocCount	int(11)	YES		NULL	共同文献数目

对于每两个菌种，该表存储了它们的共同文献列表及共同出现的次数。系统前端展示页面如下图所示：

Strain Name	Strain Name	cooccurrence relation
ATCC 25922	ATCC 27853	953
ATCC 25922	ATCC 29213	695
ATCC 25922	ATCC 25923	610
ATCC 29212	ATCC 29213	601
ATCC 25922	ATCC 29212	482
ATCC 22019	ATCC 6258	453
ATCC 27853	ATCC 29213	422
ATCC 25923	ATCC 27853	363
PCC 6803	PCC 7120	363
PCC 6803	PCC 7942	354

图表 7-2 菌种关系查寻

在前端页面展示中，给出了菌种与菌种之间的关系，最后一列表示了共同出现的次数。从中可以看出，大肠埃希氏菌与铜绿假单胞菌之间的关系就相对比较密切。通过查询框，输入某个菌种的代号可以查寻与之相关联的菌种信息。

通过输入“ATCC 25922”，可以查询与大肠埃希氏菌有关联的菌种，如下图所示：

Strain Name	cooccurrence relation
ATCC 27853	953
ATCC 29213	695
ATCC 25923	610
ATCC 29212	482
ATCC 35218	217
ATCC 49619	146
ATCC 700603	132
ATCC 49247	76
ATCC 6633	68
ATCC 43300	37
ATCC 29741	34
ATCC 8100	34
ATCC 6538	33
ATCC 33560	31

图表 7-3 共现关系查询结果

在数据表 StrainPair 中，不仅存储了共同出现次数，还存储了共同出现的文献 ID 列表，可以根据需要，在前端展示平台中进一步实现列出共同文献的功能，以便相关生物学研究人员查询使用。当前系统暂时还不支持这一功能，有待进一步研究。

第八章 总结与展望

8.1 工作总结

依托于 WDCM 的数据支持，本文探讨了文本挖掘技术在微生物相关文献挖掘中的应用。截止到目前为止，WDCM 已经注册了全世界近 600 个菌种保藏中心，并且还在不断的扩大。在数据资源方面，WDCM 的文献库中收录了来自 PubMed, Highwire 等大型数据库及 IJSEM 等微生物相关杂志中的文献信息，还有来自于 WIPO 的专利文献信息，然后挖掘分析这些文献中引用的菌种资源，从而建立起文献资源与菌种资源的一个桥梁，作为一个评价保藏中心菌种质量的一个参考标准。

在分析菌种与文献之间的引用关系的时候，主要利用到的是基于正则表达式的匹配过程，从而建立一一对应的关系模型，最终做出相关统计，以供生物学家作相关的参考。但这种基于关键字匹配的过程存在一定的错误，因为同一个英文的缩写可能代表多种含义，并不是指代某个保藏中心，为了能够做进一步的分析，本文提出了从文本的语义模型上挖掘相关度的思路。

因为计算机无法对纯文本信息进行直接处理，我们需要一个格式化过程，在此思路的基础之上，首先本文探讨了文本的预处理过程，利用到了开源工具 NLTK，对文献进行词性标注，然后进行名词短语的抽取，然后选择出现频率较高的短语作为该文献对应的主题词，从而量化表示，作为计算分析的基础。

预处理结束之后，各文献就有了对应的一个格式化之后的中间表示状态，也就是该文献的主题词集合。然后针对文献自动分类的需求，建立训练文献集合，然后比较了贝叶斯模型与 VSM 模型算法在实现中的效率，利用 Rocchio 算法对文献资源进行了类别标注，并统计各类别下菌种的数量。

接着利用 DBSCAN 聚类算法，对 IJSEM 杂志中的微生物文献进行分析，抽取了其中菌种引用频率较高的文献集合，利用该文献集合进一步抽取出了主题词典。利用主题词典，对文献与菌种之间的相关性作一个评分，从而辅助修正系统中的错误。

文章然后讨论了对主题进行检索的相关算法，重点分析了索引空间的压缩策略以及检索结果的排序方法。利用主题词典对文献的打分情况，高分文献排在前，同时召回相似的文献，从而提高检索的召回率。最后文章还讨论了检索程序与 web 前端的实现方法，以提高系统的响应时间与效率。

在最后，本文简单实现了对菌种的共现分析。

8.2 研究展望

经过本文所讨论的几个步骤的处理，在方法上，利用文本挖掘技术改善菌种文献引用关系的基本模型已经建成，为生物学家提供更多的信息支撑服务。但在细节上，还有许多需要进一步改进的地方，主要包括以下几个方面：

1） 预处理过程的改进。预处理的目的是进行主题的抽取，这是整个系统最为基础的一步，因为后面所有的分析工作都是建立在这一步的基础之上的。改进的方法主要包括提高词性标注的准确度，另外可以根据人工经验进行适当的过滤，使得所抽取的主题尽量贴近于原文献的研究领域。

2） 训练集合的完善。对于文献的分类来说，训练集合是分类的基础工作，分类结果是否准确在一定程度上取决于训练集合中的文献是否能代表该类别的特征集合。当前的类别定义及训练文献参考了 Mesh 对微生物的分类，其训练集合取自于 PubMed 中的文献。后期可以适当增加一定的人工工作量，对训练集合中的文献进行精挑细选。

3） 随着访问量的提高，检索效率需要进一步的优化。可以采取的策略包括分布式，多线程支持，通过并行技术，缩短系统的相应时间。

4） 优化用户界面，提高用户体验，使得系统更加符合用户的使用习惯。

参考文献

[1] David Hand, HeikkiMannila, PadhraicSmyth. 数据挖掘原理[M]. 张银奎，廖丽，宋俊. 北京：机械工业出版社，2003.186－207.
[2] Feldman R, Dagan I . Knowledge Discovery in Textural Databases(KDT) [C]. Montreal: Proc. Of the 1st Int' l Conf on Knowledge Discovery,1995.112-117.
[3] uthrich B, Permunetilleke D, Leung S, et al. Daily Predication of Major Stock Indices from Textual WWW Data[C]. New York: Proc.of the 4th Int' lConf. on Knowledge Discovery,1998.
[4]Jeffery L. Solka. Text Data Mining: Theory and Methods[J].Statistics Surveys,2008,2,94-112.
[5] 军鹏，朱东华，李毅，黄连宏，黄进.文本挖掘技术研究进展[J]. 计算机应用研究，2006，02.
[6] 张雯雯. 文本挖掘工具述评[J]. 图书情报工作,2012,56(8):26-31.
[7] Bates, M. Models of natural language understanding. Proceedings of the National Academy of Sciences, 92, 9977-9982,1995.
[8] Lenat, D.B. (1995) "CYC: A large-scale investment in knowledge infrastructure." Comm ACM,Vol. 38, No. 11, pp. 32–38.
[9] Hearst, M. (1999). Untangling text data mining. In Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20-26, 1999 (invited paper).
[10] Liddy, E. D. (2000). Text mining. Bulletin of the American Society for Information Science, 27. Online: http://www.asis.org/Bulletin/Oct-00/liddy.html
[11] 殷蜀梅,张智雄.医学文献集合的主题抽取和主题聚类实践[J],数字图书馆论坛,2008,09.
[12] Aslankoohi E, Voordeckers K, Sun H, Sanchez-Rodriguez A, van der Zande E, Marchal K, Verstrepen KJ. Nucleosomes affect local transformation efficiency. Nucleic Acids Res. 2012 Oct;40(19):9506-12.
[13] Sebastiani, F. (2002) "Machine learning in automated text categorization." ACM Computing Surveys, Vol. 34, No. 1, pp. 1–47.
[14] Witten, I.H. and Bainbridge, D. (2003) How to build a digital library. Morgan Kaufmann, San Francisco, CA.
[15] 申红，吕薄粮，内山将夫，进佐原均.文本分类的特征提取方法比较与改进[J]，计算机仿真，2006，23（3）.
[16] C. J. van Rijsbergen, (1989), Information Retrieval, Buttersworth, London, second edition.

[17] Erald Kowalski, Information Retrieval Systems – Theory and Implementation, Kluwer Academic Publishers, 1997.
[18] Oren Zamir, Oren Etzioni, Omid Madani, Richard M. Karp, Fast and Intuitive Clustering of Web Documents, KDD ’97, Pages 287-290, 1997.
[19] Anna Huang. Similarity Measures for Text Document Clustering[C]. the New Zealand Computer Science Reasearch Student Conference 2008.
[20] 孙吉贵，刘杰，赵连宇. 聚类算法研究[J], Journal of Software 2008,19(1):48-61.
[21] J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297
[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu (1996-). "A density-based algorithm for discovering clusters in large spatial databases with noise". In Evangelos Simoudis, Jiawei Han, Usama M. Fayyad. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9.
[23] Sander, Jörg; Ester, Martin; Kriegel, Hans-Peter; Xu, Xiaowei (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". Data Mining and Knowledge Discovery (Berlin: Springer-Verlag) 2 (2): 169–194. doi:10.1023/A:1009745219419.
[24] Christopher D. Manning, Probhakar Raghavan, Hinrich Schutze. Introduction to Information Retrieval[M]. 王斌，译.北京：人民邮电出版社，2010：1-45
[25] Amit Singhal. Modern Information Retrieval: A Brief Overview. IEEE Data Eng. Bull. 2001 , 24(4): 35-43.
[26] M. E . Maron and J. L . Kuhns. On relevance, probabilistic indexing and information retrieval. Journal of the ACM,7:216-244,1960.

致 谢

不知不觉，求学生涯二十载，即将走到尽头.回头遥望，虽然曾经认为长路慢慢，遥遥无期，今亦觉得光阴确实如箭，往者已不再可追.岁月一点点过去，历经沧桑，又一次到了人生的十字路口，开启一段新的路途.抬头，遥望，在兴奋之余，又似乎有几分迷茫，似乎未来是那么的不可捉摸.

回想当初，已经不记得是何时开始知道中国科学院研究生院这个地方，再到后来又如何得知计算机网络信息中心，再直到某天发现这里招收研究生.猛然发现，这里就是我要去的地方，暗自思量，却又增添几分忧愁，面对科学院的光环，自己实力未必能达到要求.不过总是需要争取一下，即使保送不成功还有考研的机会.于是就这样怀着试一试的心态，填写申请，从广州到北京单刀赴会，面试，最终，从乔老师手里拿到了一纸拟录取通知书，走进了研究生院的大门.

在研究生这三年的期间里，有许多需要感谢的人和事.首先需要感谢的是我的指导老师马俊才老师，感谢马老师提供的这个科研平台，给予的指导与支持，然后感谢实验室的刘翟老师，刘老师学识渊博，项目经验丰富，在毕业课题的选题与论文的撰写方面提供了宝贵的意见，还有孙清岚老师，感谢孙老师的日常辅导及在实验室生活中给予的关心与照顾，同时最后也感谢微生物所的全体员工，也祝你们工作顺利，全家幸福.

还要感谢已毕业的苏晓林师兄，吴军师兄，还有夏青师姐，初进实验室的那段时间，正是在你们的带领之下，才慢慢熟悉实验室的各种规章制度生工作流程.同时感谢张润与苏锦河同学，感谢你们的陪伴，在学习与工作中提供的帮助，还有王梓涵与王华进两位师弟为实验室带来的欢声笑语，最后感谢网络中心 2010 级亲爱的同学们，曾经一起上课，一起突击考试的情境依然历历在目.

最后感谢所有亲人与朋友们，谢谢你们多年来的关心，谢谢！

作者简介

【 基本情况 】

姓名：魏永勇 性别：男 民族：汉

出生日期：1987.03.08 籍贯：重庆梁平 政治面貌： 团员

【 学历及工作经历 】

2006.09-2010.06 中山大学 生物技术专业 理学学士

2010.09-2013.06 中国科学院大学/中国科学院计算机网络信息中心 计算机应用技术专业 工学硕士

【 论文发表情况 】

魏永勇，马俊才，刘翟，孙清岚 基于 NLTK 与 DBSCAN 聚类算法的趋势预测系统 计算机应用研究[2012 年 7 月录用]

【 参加科研项目情况 】

时间	科研项目	承担任务
2011/10-2012/01	Wipo 专利信息统计分析	负责 Web 系统开发
2012/03-2013/01	ABC,生物资源引用平台	负责数据库设计与维护,数据更新

【 攻读学位期间获奖情况】

2012.01 中国科学院计算机网络信息中心 年终考核三等奖

2012.07 中国科学院大学 “三好学生”

2013.01 中国科学院计算机网络信息中心 年终考核一等奖