基于拼音相似度的汉语模糊检索方法的研究

Research on Chinese Approximate Retrieval Methods Based on Pinyin Similarity

（申请清华大学工学硕士学位论文）

培 养 单 位 ：计算机科学与技术系

学　　科 ：计算机应用

研 究 生 ：曹 鞏

指 导 教 师 ：郑 方 研 究 员

联合指导教师 ：邬 晓 钧 助 研

二〇〇九年三月

基于拼音相似度的汉语模糊检索方法的研究

曹 曄

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘　要

随着互联网的发展，信息检索技术，特别是文本检索技术，已经成为与科学研究、日常生活和国民经济密切相关的重要技术。信息检索的主要功能是根据用户输入的信息，在信息集合中找出与用户需求最为相关的信息展现给用户。

传统的文本检索普遍采用以关键词匹配为核心的检索技术，将用户输入的查询文本划分为关键词，通过统计信息赋予关键词不同的权值，并且根据关键词在文档中的出现次数计算权值来决定文档和用户查询文本之间的相关性。当用户输入含有错误，或者与检索意图有偏差时无法很好地返回满足实际检索需求的结果。根据用户日志统计，拼音错误是用户最主要的输入错误情形。

本文引入基于拼音相似度的编辑距离来衡量汉字字符串之间的相似度，提出一种模糊检索方法，对用户的输入进行扩展，将扩展出的检索结果与原结果融合展现给用户。实验结果表明，新方法对于含有拼音错误的输入能较大程度提高召回率与准确率，对于无错查询只有轻微的负面作用。

基于倒排索引的索引归并是检索核心技术，是主要的检索耗时过程，索引归并的经典算法通常无法满足实际的用户需求。商业搜索引擎普遍采用折中的技术手段，牺牲召回率来大大加快归并过程。

基于用户通常只关注前 N 项检索结果这一事实，本文提出了一种可广泛适用于目前流行的检索模型的预测剪枝的索引归并算法，通过提前去除不可能进入排名前 N 的候选文档，以及在前 N 项结果已经找到并且排名全部确定时提前结束检索过程，可以在确保前 N 项结果不变的情况下加快检索过程。在合适的参数选择下，新算法能够使检索速度提高 6 倍左右。

关键词：信息检索　模糊匹配　向量空间模型　查询扩展　索引归并

Abstract

Information retrieval, especially text retrieval, has become an important technology closely related to scientific research, daily life and the national economy with the fast development of the Internet. The primary function of information retrieval is to identify the most relevant information from the whole collection to the needs of users based on the input query.

Keyword Matching is the basic technology of traditional text retrieval systems. The user's query is divided into different key words while each has a special weight judged by statistical information. The relatedness score between the document and the query is then calculated according to the times of the key words in the document. The search result will not meet the user's needs well if the query has errors in it or deviate from the user's intention. According the users log statistics, Pinyin errors are the most popular

In this paper, an edit-distance based on Pinyin is introduced to measure the similarity between two Chinese strings, on which an approximate retrieval method is further proposed so that the user's query can be expanded and more results are presented to the user with existing ones. Experiment results showed that the new method can greatly improve the precision and the recall rate for queries with Pinyin errors while only a slight drop for correct queries.

Based on inverted index, index merging is a key and most time-consuming process in the retrieval task. The classic multi-way merge algorithm is usually unable to meet the actual needs of the users and technical compromises are commonly used in commercial search engines to significantly speed up the merging process with decrease of the recall rate.

Considering that the users usually concerns about only the top N items in the search results, we propose a new index merging algorithm, in which the documents impossibly ranking top N are removed earlier and the merging process ends immediately if all the top N results have been found with the correct ranking. Given appropriate parameters, the merging process can be speeded up 6 times with the new algorithm.

Key Words: information retrieval approximate match query expansion vector space model index merging

目 录

第 1 章 引言 1
1.1 信息检索技术概述 1
1.1.1 信息检索的意义 1
1.1.2 广义和狭义的信息检索 2
1.1.3 信息检索的关键技术 2
1.1.4 现代信息检索面临的挑战和应对 4
1.2 模糊匹配技术概述 6
1.2.1 模糊匹配的定义 6
1.2.2 模糊匹配的意义 6
1.2.3 相似度度量标准 7
1.2.4 主流的模糊匹配算法和存在的问题 7
1.3 本文研究的主要内容 8
1.3.1 研究目标 8
1.3.2 各章内容简介 8
第 2 章 三种不同相似度度量下的汉语模糊匹配算法 11
2.1 三种相似度度量方式 11
2.1.1 基于汉字的编辑距离 12
2.1.2 基于拼音的编辑距离 12
2.1.3 基于拼音改良的编辑距离 12
2.2 索引与匹配算法 13
2.2.1 建立索引 13
2.2.2 汉字串近邻空间 14
2.2.3 查询扩展和模糊匹配 15
2.3 实验 16
2.3.1 实验设计 16
2.3.2 评测指标 17
2.3.3 实验结果和分析 17

2.4 本章结论 18
第 3 章 以模糊匹配为核心的检索方法 19
3.1 相关研究工作 19
3.1.1 文本相关性的研究历史 19
3.1.2 信息检索模型概述 20
3.1.3 布尔模型 22
3.1.4 向量空间模型 23
3.1.5 关键词赋权 29
3.1.6 向量空间模型的不足以及现有的解决方案 31
3.2 模糊检索方法 35
3.2.1 汉语的拼音输入 35
3.2.2 模糊检索的实现 36
3.2.3 分词边界的影响 38
3.2.4 offset 调权的应用 39
3.3 实验 40
3.3.1 实验设置 40
3.3.2 实验结果和分析 41
3.4 本章结论 42
第 4 章 检索过程的性能优化 45
4.1 相关研究 46
4.1.1 倒排索引 46
4.1.2 索引归并算法 49
4.1.3 商业搜索引擎的折中处理 50
4.1.4 其它的优化算法 52
4.2 索引归并中的剪枝算法 53
4.2.1 剪枝算法流程 53
4.2.2 *ChangeAddMode* 函数 55
4.2.3 *AccumTrim* 函数 55
4.2.4 *QuitJudge* 函数 56
4.2.5 剪枝频度的选择 57

4.2.6 剪枝算法在模糊检索系统中的应用.....57
4.3 实验.....57
4.3.1 实验设置.....57
4.3.2 实验结果和分析.....58
4.4 本章结论.....59
第 5 章 结论和展望.....61
参考文献.....63
致 谢.....67
附录 A 本文的检索系统采用的停用词表.....69
个人简历、在学期间发表的学术论文与研究成果.....71

第1章 引言

1.1 信息检索技术概述

1.1.1 信息检索的意义

随着社会的日益进步和科学技术的进一步发展，人类进入了一个知识大爆炸的年代：创造出来的知识、交互产生的信息，数量越来越多，种类越来越多，存储方式也越来越多。在这种情况下，如何快速、便捷地从浩瀚的信息“海洋”中获取所需要的知识和信息，已经成为一个亟待解决的具有重要实际意义的问题。

从科学研究的角度来讲，一种便捷有效的信息检索技术的存在，可以避免重复研究或者研究上走弯路，有助于节省研究人员的时间，同时也是研究人员获取新知识、了解新动态以及将自己的研究成果展示出来的捷径。

从人们实际的生活需要来看，进入本世纪之后，随着互联网的快速发展，对于信息检索技术提出了更加迫切的要求。网络已经成为人们工作、学习、娱乐的重要场所，成为人们与外界交互的重要通道。与电视、广播、报纸等传统媒体不同，互联网上的信息数量上更加庞大，组织上更加复杂，同时却缺乏良好的用户推送模式，用户获取信息从以前的被动接受为主变为主动寻求为主。从浩瀚的互联网信息海洋中获取需求的信息，已经成为与人们日常生活密切相关的一项技术。正因为这方面的迫切需求，各主流商业搜索引擎，已经成为当代网民联入互联网的主要入口。根据权威的互联网流量统计提供商 ALEXA[1]2008 年第 12 期统计数据，著名中文搜索引擎百度[2]已经成为访问量最高的中文网站。由此可见，信息检索技术已经成为整个互联网构架体系中的重要一环。

从对国民经济的影响来看，由信息检索技术衍生而来的竞价排名已经成为互联网经济乃至整个实体经济的一大亮点。竞价排名技术为无数中小企业提供了便捷快速低成本的营销手段，而这一切都是和信息检索技术的产生和发展密切相关的。

[1] http://www.alexa.com/
[2] http://www.baidu.com

由此可见，信息检索技术如今已经成为一门对科学研究、日常生活以及国民经济的发展有着重要影响和推动作用的关键性技术，对信息检索进行进一步的研究，使得它能够更好地帮助用户获取所需要的信息，有着切合实际的重要意义。

1.1.2 广义和狭义的信息检索

信息检索系统的功能是根据用户的输入信息，检索返回用户需求的目标信息[1]。

狭义的信息检索，用户的输入是文本串，用户检索的目标也是文本形式的，如科技文档或者网页文本。这种信息检索也被称为文本检索。

广义的信息检索，则是指搜索的目标信息以及用户的输入信息都可以是多种形式的。搜索的目标信息可以是文本、音频、视频等，用户的输入也可以不局限于文本，包含更多的多媒体因素。这种信息检索也被称为多媒体内容检索。

虽然从实际的应用角度来看，多媒体内容检索能够更好地满足用户的实际需要，但是因为技术上的原因，现今商业搜索引擎提供的主要服务网页检索是典型的文本检索，这方面的研究开展的比较早，商业应用也比较成熟。即便是音乐搜索、图片搜索、地图搜索，也都是以文本检索的方式完成的。以音乐搜索为例，它的用户输入依然是文本，而检索的目标确切来说，是音乐文件对应的文件名等文本描述信息，离真正的多媒体内容检索还相距较远。目前，哼唱搜索[2]是比较接近实用的多媒体内容检索：用户哼唱一段旋律，检索出这段旋律所属的音频。与此同时，图片内容搜索、视频内容搜索虽然在各大高校和科研机构中也得到了广泛的关注，并取得了一定的研究成果，但是离实际的商业化使用还有一段距离。

本文的研究关注于文本检索，以下如果没有特别说明，文中提到的“信息检索”都指狭义的信息检索。

1.1.3 信息检索的关键技术

从用户输入查询（查询）开始，到检索系统输出排序后的检索结果为止，整个信息检索系统的处理流程可以用图 1.1 来表示。

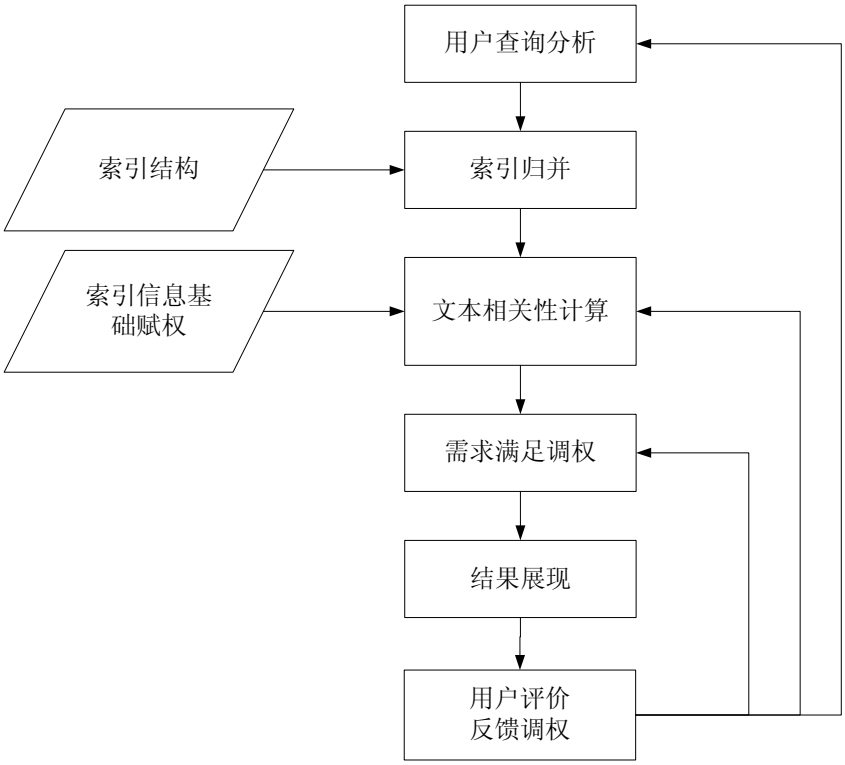

图 1.1 一个典型的信息检索系统

其中用户查询分析、文本相关性计算和需求满足调权[3]这三步都是直接影响检索结果的关键技术，而索引归并算法则直接影响信息检索系统的运行效率，它们都是信息检索中的关键性技术。下面，按照它们在整个检索流程中的先后顺序进行介绍：

查询分析技术根据用户的输入文本识别出用户的检索需求，它包括对用户查询进行分词，并且识别分词后各个关键词的表意能力和重要性，对于用户的查询进行必要的扩展，同时结合资源的情况进行适当的文本替换[4]。另外，对于用户的错误输入也要进行必要的纠错。

索引归并根据查询分析的结果从整个文档集合中找出符合用户需求的那些候选文档。归并算法的核心是在保持召回率的前提下得到尽可能高的性能。这方面的关键技术包括归并过程中的剪枝算法、索引结构的压缩处理以及磁盘 IO 的优化处理。

文本相关性计算根据一个有区分度的、可比的计算模型，计算索引归并中每个归并结果与用户查询的相关性得分。这方面的研究主要集中在确定一种好

的相关性计算模型上。

需求满足调权则是在相关性计算的基础上，对于每个归并结果的得分进行细化的调整和处理。需求满足调权包括以下几个维度的处理：文档质量调权、时效性调权、点击反馈调权和检索多样性调权。其中，文档质量调权是为了尽量展现自身质量较好的文档；时效性调权是为了对于“今年中秋是哪一天”和“N95 手机价格”这样有较强时效性需求的查询展示最近的结果；点击反馈调权则是根据用户对于本查询的原有点击结果，调整结果展现的位置；检索多样性调权则是对于类似“苹果”这样可能有电影、电器、水果多方面需求的查询，在展现的时候尽量满足不同用户的需求。

这四项关键技术在整个检索过程中是相互联系互相影响共同作用的。“查询分析”中对查询的分析结果以及“文本相关性计算”中选择的相关性计算模型会直接影响“索引归并”的算法性能，“需求满足调权”的调权方法又直接取决于“查询分析”的结果。

除此之外，商业搜索引擎对于用户日志的分析，和对用户评价和反馈的利用已经成为一项十分重要和基础的工作。对于用户查询和点击日志的分析可以帮助挖掘用户的行为，改善检索的效果。商业搜索引擎的查询纠错以及相关检索功能都是通过用户日志分析来完成的。比较遗憾的是，由于用户日志分析对于用户日志数据的规模有着较大的要求，一般的高校和科研机构在这方面的研究相对于商业搜索引擎还比较滞后。

1.1.4 现代信息检索面临的挑战和应对

随着互联网的发展以及用户的逐渐成熟，现代信息检索主要面临以下一些方面的挑战：

1. 随着文档集合规模的增大，对于索引结构、归并算法都提出了新的要求。以中文网页的数量为例，据《2005 年中国互联网络信息资源数量调查报告》，仅仅 2005 年，中文网页数量就增长了 227.7%，达到了 63 TB。到 2009 年，以最保守的估计，这一规模至少也应该增长 4 倍以上。在这样一个规模的数据集合上进行索引和检索，是一项非常有挑战性的工作。所以，主流商业搜索引擎普遍采用了分布式计算的方式，将文档集合分为不同的小库分别进行部署。对于用户的查询，先在各个小库上分别进行检索，再将不同库内部的检索结果相互融合后展现给用户。因而，对

于现代搜索引擎来说，架构设计已经成为一项越来越关键的技术。

2. 随着互联网的普及程度逐步提高，搜索引擎的用户数量也大幅度增加。因而，对于检索的耗时也提出了更高的要求。信息检索的消耗时间主要发生在索引归并这一步，为了提供更加快捷的服务，商业搜索引擎普遍采用了全与归并、高频词过滤、高频词粘接等方式，在部分牺牲相关性的基础上来得到更快的归并速度。效果和性能一直都是一个折中，虽然对于主流商业搜索引擎来说，由于检索了海量的网页文本，即使牺牲掉部分效果也依然能够满足用户的大部分需求，但是小规模的搜索引擎必须通过优化索引结构和归并算法来力争在相关性不受太大影响的情况下提高归并的效率。
3. 由于搜索引擎已经成为大部分网站的主要用户入口，因而在搜索引擎的检索中排名更加靠前，已经成为很多网站的迫切需求，并由此产生“搜索引擎优化”（Search Engine Optimization）技术，针对搜索引擎对网页的检索特点，让网站建设各项基本要素适合搜索引擎的检索原则，从而使搜索引擎收录尽可能多的网页，并在自然检索结果中排名靠前，最终达到网站推广的目的。其中，很多搜索引擎优化都是纯粹的作弊行为，通过隐藏文笔、链接工厂、桥页、跳页等来提升自身网页的排名，对于这类行为，各商业搜索引擎都通过完善网页分析、采用链接算法等方式来处理。
4. 由于用户的自身情况不同，相同的查询所表达的用户需求可能是不同的，返回相同的结果并不能总是很好地满足当前用户的需求。以地域性需求为例，同样是搜索“肯德基”，北京的用户更关注北京的肯德基的情况，湖北的用户自然更关心湖北的搜索结果。对于这类需求，可以通过用户的 IP 地址判断他的位置，返回更符合需求的结果。百度的主要收入来源竞价排名中，已经使用了按地域推送广告这一技术。除此之外，还有一些难以判断的硬性需求。例如，同样是搜索“苹果”，用户的需求可能是电影，可能是电器，也可能是水果，搜索引擎很难判断出用户的准确意图，只能通过分析最近一段时间的整体用户行为，判断这个查询下用户的主需求，然后尽力满足主需求。当然，随着对于用户行为更加深入的研究，可以根据用户之前的搜索行为推断实际兴趣所在，从而提供个性化搜索，也将是下一代搜索引擎的重要特征。

总之，随着技术的进一步发展、数据规模的增大以及用户行为的日渐成熟，搜索引擎研究的热点已经从单纯的相关性模型转到对结构的调整、用户需求的分析和更好的满足需求这些问题上来了。

1.2 模糊匹配技术概述

1.2.1 模糊匹配的定义

如果给定目标字符串 *str* 和模式串 *pat*，模糊匹配问题可抽象地描述为：给定正整数 *k* 作为阈值，找出目标字符串 *str* 中的所有子串 *s*，使得 $Dis(s, pat) \le k$ [5]。其中，$Dis(a,b)$ 用来衡量串 *s* 和串 *pat* 之间的相似度，值越小表明两者之间越相似，当这个值为 0 时表明两者完全一样。

显然，在阈值 k 为 0 时，模糊匹配退化为经典精确匹配问题，即给定目标字符串 *str* 和模式串 *pat*，在 *str* 中寻找 *pat* 的匹配位置。在字符串精确匹配算法方面，BM 算法及其改进算法（如 BMH[6]等）能达到极高的效率（子线性），被各种检索系统广泛使用[6][7]。BM 算法的“跳跃”策略，在模糊匹配的很多计算方法（如 TU 过滤算法）中，都得到了借鉴和应用。

1.2.2 模糊匹配的意义

现有的信息检索系统大部分采用基于关键词精确匹配的检索技术[3]。在实际应用中，由于表达能力有限，或者输入查询时发生错误（例如使用拼音输入法输入了同音的错字），用户的输入往往无法精确表述自己的检索意图；另一方面，检索系统索引的数据本身也可能存在错误[4]，这就导致单纯文字上的精确匹配可能无法满足用户的需求。

与此同时，聊天室、即时通信软件、手机短信也逐渐成为人们日常生活和工作中十分重要的部分。短信、聊天软件的文本与搜索引擎的用户查询的共同特点是：文本长度不长，数量十分巨大，可能含有比较多的错误，通常的文本处理方法难以奏效。我们将这类文本称为短文本。

由于短文本的广泛存在，对短文本进行过滤、分类和聚类有巨大的实际需求。这三类需求在技术解决方案上有一个共同点，就是计算两个文本之间的相似度，换个角度来说，就是在一定的阈值之内对短文本进行模糊匹配。研究文本的模糊匹配对于解决这类问题也有重大意义。

当然，除了信息检索和自然语言处理领域以外，模糊匹配方法还广泛用于其他领域，如入侵检测、信息过滤、基因检测等[8][9]。本文只关注模糊匹配技术在信息检索领域之中的应用，特别是专注于利用模糊匹配技术，来改善当用户输入存在偏差时的检索结果。

1.2.3 相似度度量标准

进行文本的模糊匹配，首先要有合适的文本相似度度量标准，也即合适的函数 $Dis(a,b)$。目前，普遍采用编辑距离来刻画两个字符串的差异[10]，度量它们之间的文本相似度。

设 A、B 为两个字符串，狭义的编辑距离定义为把 A 转换成 B 需要的最少删除（删除 A 中一个字符）、插入（在 A 中插入一个字符）和替换（把 A 中的某个字符替换成另一个字符）的次数，用 $ED(A,B)$ 来表示。直观来说，两个串互相转换需要经过的步骤越多，差异越大。

除了编辑距离以外，两个字符串的最长公共子串和最长公共子序列的长度也常常被用来衡量两个字符串的相似程度。最长公共子串可以看作最长公共子序列的一种特殊情况。当替换代价为无穷大的时候，求最长公共子序列就等同于求编辑距离。所以，在计算意义上，三种相似度度量方式可以统一。

但是，编辑距离最初应用于英文等字母文字，直接将它移植到汉语之中是否能够满足实际的需要还值得商榷。本文将在此方面做进一步的研究和探讨。

1.2.4 主流的模糊匹配算法和存在的问题

模糊匹配技术中计算编辑距离的主要策略有以下几种[10,11]：动态规划[12]、自动机[13]、位平行策略[14]。在实际的应用中，常常将它们结合使用以获得更高的时间效率[15]。但是，在文献[16]中已经证明，计算两个长度分别为 m 和 n 的字符串的最长公共子序列的时间复杂度不可能低于 $O(m\log n)$。从前面的分析可以知道，计算编辑距离的时间复杂度不会低于这个复杂度，在很多情况下这是无法满足实际应用需要的。文献[8]使用带有剪枝的动态规划算法和扩展的 TU 过滤算法进行汉语模糊匹配，取得了非常好的效果。在该文献的实验中，*str* 为 600 左右的歌手、乐队名和 5000 左右的歌曲名，*pat* 为用户日志中的一次用户输入，它们的长度大多在 10 个汉字以内，在 k 限制最小且算法运行最快的情况下，一次模糊匹配所需要的时间是 10 毫秒左右。如果待匹配的 *str* 进一步增加，模

糊匹配运行的时间也将相应线性增加。

主流商业搜索引擎，如百度、谷歌[3]等，检索的文本规模以 TB 计，而要求的响应时间通常不到 0.1 秒。如果直接采用上面介绍的算法，将用户的输入查询在文本集合上进行模糊匹配，是难以在短时间内完成整个匹配流程的。

注意到，上述的模糊匹配算法都是在原文上进行的。而在搜索引擎系统中，普遍采用索引来加快精确匹配的速度，有鉴于此，本文也探讨利用索引技术，来加快模糊匹配的速度，从而将它引入到信息检索系统中来。

1.3 本文研究的主要内容

1.3.1 研究目标

本文的研究目标是将文本的模糊匹配引入到信息检索系统之中，使得检索结果能够更好地符合用户的需求。本文对传统的基于关键词精确匹配的检索模型进行了扩展，并且通过建立倒排索引、对索引归并过程进行剪枝等方法，加快模糊匹配的运行速度，以满足实际的应用需求。

1.3.2 各章内容简介

本文后续的内容安排中，第 2 章是对必要的先验方法的探讨，第 3 章和第 4 章则是分别在检索效果和检索效率上的改进。具体介绍如下

现有的汉语相似度度量方法主要继承于英文方面的研究，第 2 章针对现有的度量方法的不足之处，提出三种不同的汉语相似度度量方式，并且以倒排索引的方式实现了在大容量文档集合上的快速模糊匹配。实验部分比较了这三种不同的相似度度量方式的准确率和召回率。

结合第 2 章的研究成果，考虑到目前拼音错误是信息检索系统用户查询错误的一个重要原因，第 3 章将第 2 章的相似度度量方法模糊匹配算法引入向量空间模型，针对用户实际输入的错误查询，采用基于拼音相似度的模糊匹配算法进行改进，通过实验比较改进前后的检索相关性。

而由于改进之后一次查询会变为多次查询，时间消耗也会有所增加。为了使得时间性能能够有所改善，更好地满足用户的实际需求，第 4 章对向量空间模

[3] www.google.cn

型中的倒排索引以及多路归并算法进行了时间性能上的优化，并通过实验对改进前后的检索效率进行比较。

第 5 章总结全文，对后续研究进行展望，同时，也对本文方法的适用范围进行了说明。

第2章 三种不同相似度度量下的汉语模糊匹配算法

正如第 1 章所介绍的，现有的主流商业信息检索系统大部分采用基于关键词精确匹配的检索技术，取得了一定的成果[1]。但是在实际应用中，用户的查询输入与检索系统数据库的构建都不可能完全正确[8]。用户对于搜索主题所处的领域不了解，采用不合适的查询词，会导致查询词的覆盖范围大大缩小[17]；在中文信息检索系统中，用户还常会输入同音或近音的错别字。模糊检索根据用户输入的模糊特征来检索匹配内容，可处理以上这些关键词精确匹配所无法解决的问题[5]。

在英文检索系统中，通常对用户输入的单词进行拼写纠错，就能解决大多数问题[5]。系统先搜索单词表，找出所有与查询串中单词的编辑距离（Edit Distance）在一定限度之内的所有词汇，再根据这些词汇来执行精确检索，即可在一定程度上实现模糊检索[11,18]。所以，英文模糊检索的研究主要集中在快速简捷的字符串模糊匹配算法方面[6,7]。考虑到查询串的整体性，文献[19]提出了块索引的方法，通过二步的模糊匹配过程，找到与查询串整体编辑距离在一定限度之内的串的位置。

汉语是典型的非字母语言，把任意两个汉字的差别都算成同一个值不够精确。绝大多数汉字都是表意单元，词语的搭配灵活多样，难以建立完整的词表用于纠错。所以汉语的模糊检索无法照搬英文中的方法，目前的研究主要集中在快速的汉字串模糊匹配算法方面[8,9]。其中，文献[8]的研究工作改善了模糊匹配的时空复杂度，在实际系统中算法的时间复杂度可以达到子线性。然而，遍历整个文本集来寻找相似串的出现位置，虽然可以比较准确地完成模糊检索任务，但随着数据集规模的进一步增长，时间开销依然难以接受。

与前述思路不同，本文提出一种基于汉语拼音的模糊检索方法，通过扩展原始的查询串，将模糊检索任务转化为若干精确匹配的检索任务，从而大大降低算法复杂度。

2.1 三种相似度度量方式

参考英文基于字母编辑距离的度量方式，本文提出 3 种汉字串相似度的度

量方式：基于汉字的编辑距离、基于拼音的编辑距离，以及基于拼音改良的编辑距离。

2.1.1 基于汉字的编辑距离

将单个汉字作为编辑距离中的距离度量单位，即：两个汉字串之间的距离，等于使它们完全一样所需的最少替换、插入或者删除的汉字个数。

2.1.2 基于拼音的编辑距离

由于汉语拼音输入法的广泛使用，大部分用户的输入错误都表现为同音字或者近音字的替换误用，对 Sogou 实验室所提供的用户日志的分析结果也证实了这一点。基于此，本文提出了基于拼音的编辑距离来衡量汉字串的相似度。如果把拼音串简单地看作广义的英文字母串，则替换、插入或者删除一个字母后，所得结果不一定是合法的拼音串。因此应从音节的角度来分析拼音串的差别。

对于一个单独的音节来说，它与另外一个音节的差异总可以分解为以下三种变化：声母变化、韵母变化和声调变化。声母、韵母和声调的可能取值都是有限的，可以枚举定义从一种取值变为另一种取值的编辑距离。所以，对于一个现有的音节，容易找到所有与它编辑距离为 n 的音节。例如，要找到所有与它编辑距离是 2 的音节，那么变化可能是声母改变 1 个距离单位，韵母改变 1 个距离单位，声调改变 0 个距离单位；或者声母改变 2 个距离单位，韵母和声调没有发生改变；等等。这只是一个排列组合的问题。

如果给所有音节编号，将音节整体看作一个特殊的单字，那么基于拼音的编辑距离可认为是基于汉字的编辑距离的细化，即不同的汉字之间根据拼音的近似程度有不同的距离，而不是笼统地将任意两个汉字的距离都计为 1。

2.1.3 基于拼音改良的编辑距离

根据前述基于拼音的编辑距离定义，音节/li3/和/ni3/的编辑距离是 1，/li3/和/pi3/的编辑距离也是 1。但是这 2 组音节的差异从发音机理角度来看，/li3/与/ni3/更加近似。类似的例子如：音节/lin3/与/ling3/的差异较小，而/lin3/和/lan3/的差异较大。

基于以上考虑，提出改良的编辑距离计算方式如下。

1） 发音相似（容易发生替换错误）的声母或韵母之间差异小于 1。例如，/l/和/n/、/z/和/zh/、/c/和/ch/等声母对，/in/和/ing/、/en/和/eng/等韵母对，对音节编辑距离的贡献小于 1（本文实验中赋予 0.5 的替换代价），这样的声母和韵母对一共是 9 对。对于其余的声母或者韵母对的替换代价的计算，则依然采用方法二中的使用字符编辑距离的计算方法。

2） 若同一音节的声母和韵母同时发生改变，则在计算编辑距离时给予一个正的惩罚值（本文实验中取值为 2）。根据这一计算方式，对于音节串 $A=I_1I_2I_3...I_n$（其中 I_i，$i=1,2,...,n$ 代表一个音节），若 I_2 在声母和韵母上同时发生差异为 1 的变化得到新的音节串 $B=I_1I_2'I_3...I_n$，I_1 和 I_3 各发生一个声母或韵母的差异为 1 的变化得到新的音节串 $C=I_1'I_2I_3'...I_n$，则 C 与 A 之间的编辑距离为 2，小于 B 与 A 之间的编辑距离 4。

3）音调变化导致的差异小于 1。由于音调错误比较常见和普遍，而且广泛使用的各种拼音输入法都不要求用户输入音调，所以可认为音调差异小于一般声母和韵母之间的差异。按照发音相似的韵母和声母对一样的处理方式，这种差异在本文实验中赋予 0.5 的替换代价。

在本章后面的实验中，由于将所有小于 1 的差异都赋值为 0.5，因此将所有的差异都乘以 2 之后，可以得到结果为整数的编辑距离，而这并不影响不同串之间相似度大小的比较。

2.2 索引与匹配算法

依据具体的距离度量方式，可以扩展原始的查询串，将模糊匹配转化成多个相关的精确匹配，实现检索任务，步骤是：首先对查询串进行编辑距离由小到大的扩展，然后对扩展出的查询串进行精确匹配，精确匹配的结果在去重之后再按照查询串的编辑距离由小到大进行排序，最后将排序的检索结果返回给用户。

2.2.1 建立索引

在本章的模糊匹配算法中，以离线方式对文本数据集中的单字或者音节建立索引。这是因为，在第 1 节中所提出的 3 种距离度量方式都以单字或音节作为最小的考察和计算单位。当采用 2 种基于拼音的距离度量方式时，要先将文

本逐句地转成拼音串（逐句进行拼音的自动标注可以更好地联系上下文处理多音字、变音字等拼音现象），然后再以音节为单位构建索引。

在文本数据集的索引表中，需要记录索引头（本索引对应的单字或者音节），以及索引头在文本数据集合中出现过的所有位置（文本号以及在文本中出现的具体位置）。

2.2.2 汉字串近邻空间

在对原始查询串进行扩展时，需要引入汉字串近邻空间的概念。

定义 1 在具体的汉字串相似度度量方式下，所有与目标串编辑距离为 m 的汉字串组成的集合，称为该目标串的 m-近邻空间。m-近邻空间中每一个汉字串，称为该目标串的一个 m 近邻串。

将查询串按编辑距离进行由小到大的扩展，其实质就是依次计算查询串的各个近邻空间。而近邻空间内汉字串的数量，直接影响遍历该近邻空间的时间复杂度。下面以基于汉字的编辑距离为例，对近邻空间进行分析。

令汉语体系全部汉字的集合为 Σ，则所有长度为 n 的汉字串总数为 $|\Sigma|^n$。对于一个长度为 n（n 远小于 $|\Sigma|$）的汉字串 X，它的 0-近邻空间显然只包含它自身。

对于 X 的 1-近邻空间，编辑距离可能由替换、插入和删除 3 种方式之一造成。以替换为例，长度为 n 的汉字串共有 n 个可能的替换位置，每个替换都有 $|\Sigma|-1$ 个候选的替换汉字，因此只考虑替换可得到 $n(|\Sigma|-1)$ 个与 X 编辑距离为 1 的汉字串。对插入和删除操作进行类似的分析可知：只考虑插入可以得到约 $(n+1)|\Sigma|$ 个汉字串，只考虑删除则可以得到 n 个汉字串。所以总的来说，1-近邻空间内所有汉字串的数目大约是 $(2n+1)|\Sigma|$。在实际应用中，n 一般不超过 10，$|\Sigma|$ 的大小至少在 10^3 数量级上，所以 1-近邻空间内汉字串的数目也至少是 10^3 的数量级。

对于 X 的 m-近邻空间内的串，一共发生了 m 处替换、插入或删除。可近似地认为串 X 共有 $(n+1)$ 个位置能够插入汉字，有 n 个位置的汉字能够删除或者替换。假设在 m 处变化中有 a 次插入，$m-a$ 次删除或替换，则可产生新的汉字串约 $C_{n+1}^{a}|\Sigma|^{a}\times C_{n}^{m-a}|\Sigma|^{m-a}$ 个（其中 C_{n+1}^{a} 表示组合数，下同），因此 m-近邻空间内所

有串的数目大约为 $\sum_{a=0}^{m} C_{n+1}^{a} C_{n}^{m-a} |\Sigma|^{m} = C_{2n+1}^{m} |\Sigma|^{m}$。

在实际应用中，虽然 m 的取值通常不超过 $n/2$，但 m-近邻空间内的串仍然数量巨大。

对于如此数量级的近邻空间，不可能逐一访问其中每个近邻串来进行检索。即使有可能在较小的时间开销内判断某个汉字串是否符合中文语法，从而去掉许多不合理的近邻串，这一判断过程需要进行的次数也是惊人的。因此，必须通过其它的方式来遍历近邻空间，完成对原始查询串的扩展。

2.2.3 查询扩展和模糊匹配

依然以基于汉字的编辑距离度量方式为例，介绍本章的模糊匹配算法进行查询扩展并匹配的过程。

令用户输入的查询串为 $A = a_1 a_2 a_3 a_4 a_5 ... a_n$，其中 $a_i (i = 1, 2, ..., n)$ 是汉字。在 A 的 1-近邻空间中，所有因 a_i 被替换所生成的串有 $B = a_1 a_2 a_3 x a_5 ... a_n$ 的形式，其中 $x \in \sum$ 且 $x \neq a_i$。B_i 实际上代表了 $|\Sigma| - 1$ 个不同汉字串，称 B_i 为通配串。在检索时，先分别精确地检索两个子串 $B_{i1} = a_1 a_2 ... a_{i-1}$ 和 $B_{i2} = a_{i+1} a_{i+2} ... a_n$ 在数据集中出现的位置，然后对两个子串的位置进行比较，找到串 B_{i1} 和串 B_{i2} 同时出现且 B_{i1} 在 B_{i2} 的 $i+1$ 个位置前出现的文本。由于替换可能发生在 n 个不同的位置，所以类似的通配串有 n 个。对于插入和删除两种改变方式，可构造类似的通配串扩展并检索。

分析上述过程的复杂度。考虑插入、删除和替换三种操作，则在 1-近邻空间内通配串的总数目大约是 $3n$，每个串通过至多 2 次的精确匹配实现模糊检索。考虑到精确匹配的结果可以被不同的通配串共享，实际上只需经过 $2n-1$ 次精确匹配即可实现 1-近邻空间内的所有串的检索，这相对于 1-近邻空间内串的总数量大大减少了。

对于 A 的 m 近邻空间（m>1），方法是类似的。在编辑距离为 m 时，可近似认为串有 $2n+1$ 个位置可能发生改变，相应通配串的数目约为 C_{2n+1}^{m}。由于进行精确匹配的通配串的子串实际上都是原输入串的子串，而原输入串的子串总数量为 $n(n+1)/2$，所以实际最多进行 $n(n+1)/2$ 次精确匹配。

随着 m 的增长（$m < n$），通配串的数目也逐渐增大。在实际应用过程中，可以设置返回给用户的查询结果的数量上限，以此作为算法结束的阈值。通常

在 m 还处于一个比较小的数值时，扩展就可停止。此外，由于实际检索时 n 不会很大，与扩展串精确匹配所需要的时间开销相比，计算扩展串本身的时间开销可以忽略，实验的相关数据也证实了这一点。

上述查询扩展是在基于汉字的编辑距离度量方式下实现的。当采用基于拼音的两种编辑距离度量方式时，总体思路类似，区别在以下两点：1） 在获取用户查询串之后，首先将它转换为拼音串；2） 在查询串的扩展过程中，替换生成的通配串 B_i 中的 x 不可能取所有音节，而是与 a_i 差异为 1 的音节集合 W 中的元素，因此在考察子串 B_{i1} 和 B_{i2} 的相对位置关系时，需要判断它们之间的那个音节是否属于集合 W。

对于以上得到的查询结果，首先要进行去重，然后按照它们被检索出来时所考虑的编辑距离，从小到大依次排列，反馈给用户。

综上所述，通过将用户查询扩展并分割成多个精确匹配的查询，避免了逐一遍历整个近邻空间的操作，仅仅在通常的精确检索的几倍时间内，完成模糊检索的任务。

2.3 实验

对于本章提出的查询扩展算法的正确性，也就是说查询扩展算法是否按照定义的距离度量方式从“近”到“远”地返回相应的文本，在算法层面上已经得到了证明，对于输出结果的抽查也验证了这一点。

本章提出的基于索引的匹配方法相比较常见的依次顺序匹配的方法，在时间复杂度上的差异十分明显。本章的实验目的在于考察所提出的三种不同的距离度量方式，是否能够很好满足用户的需求。

2.3.1 实验设计

实验中使用的文本数据来源于搜狐—清华大学联合实验室（Sogou，搜狗实验室）提供的网页文本数据集合，从中选取了约 4 万篇长度处于整个数据集合平均长度附近的文本。这些文本全部来源于真实网页，并且去除了 html 标引符号，只保留了文本内容。文本数据集的总大小为 79.3 MB，每个文本的平均长度为 930 个汉字。将这些数据作为实验中待匹配的文本。在实验过程中，由于没有标题信息，因而所有的文档只有一个“内容域”。

实验中一共采用了 400 组用户查询串作为匹配的目标串。每组查询串由一个错误查询和与之对应的正确查询组成。错误查询全部来源于 Sogou 实验室提供的用户查询日志，由人工选取其中一些明显有错的查询，并且人工进行了更正。为了不失一般性，在选取的过程中没有刻意挑选错误类型为拼音的查询串，因而这些错误查询能够在一定程度上代表信息检索系统实际的用户错误输入情况。经统计，这些错误查询的平均长度为 6 个汉字，其中 95%左右是拼音相关错误。

在实际的实验中，错误串被认为代表用户给予搜索引擎的错误输入，而人工更正则代表用户的真实意图。

2.3.2 评测指标

假设查询串 *B* 是错误的查询输入，串 *A* 是它的人工更正。对 *A* 进行精确匹配的检索，得到结果集包含有 *n* 个结果：$\Omega=\{W_1,W_2,W_3,...,W_n\}$，把$\Omega$视作本组查询的标准输出。

如果对串 *B* 进行精确匹配的检索，则无法得到正确的查询结果。在实验中，分别利用第 1 节中提到的 3 种编辑距离度量方式来对错误串进行模糊检索。令排序后输出的前 *p* 个文本构成集合$\Delta=\{X_1,X_2,X_3,...,X_p\}$。在这 *p* 个文本中，若有 *x* 个文本在集合Ω中，则认为模糊匹配系统 Top *p* 的准确率为 *x/p*，召回率为 *x/n*。这两个指标可以用来反映本文提出的模糊匹配系统对于用户意图的匹配程度。

2.3.3 实验结果和分析

实验所用的文本集合规模为 4 万篇文本，错误的查询串一共 400 个，它们得到的检索结果Ω平均含大约 7 篇文本。其中，包含结果数不多于 3 的查询有 31.25%，包含结果数不多于 10 的查询有 78.75%，包含结果数不多于 30 的查询有 97.5%。

我们共进行了 3 组对比实验，实验数据结果如表 2.1。其中，Top3、Top10 和 Top30 分别指在前 3 个、前 10 个和前 30 个匹配结果的集合上进行统计和计算的结果。

表 2.1 模糊匹配算法的准确率和召回率

距离度量方式	准确率 / %			召回率 / %		
	Top3	Top10	Top30	Top3	Top10	Top30
基于汉字	31.48	18.15	11.77	45.33	60.98	75.57
基于拼音	53.70	28.23	16.82	51.40	76.55	89.52
基于拼音的改进	60.42	34.17	19.62	54.31	84.45	91.70

从表 2.1 可以看出，在准确率和召回率这 2 项指标上，2 种基于拼音的度量方式都比基于汉字的度量方式有较大提高，这是由于拼音能够更好地刻画汉字串之间的相似程度。基于拼音改良的度量方式，在准确率和召回率上，都取得了最好的实验结果，说明引入语音学知识对性能提高有帮助。

从表 2.1 还可以看出，前 30 个查询结果已经能够达到 90%左右的召回率，而返回前 30 个结果，一般只需要对原始的错误查询串扩展出不多于 10 次的扩展。此外，对于 4 万个文本组成的共 79.3 MB 的文本集合，建立的索引（单级索引，未压缩）大小在 100 MB 左右，基于音节建立的索引与基于单字建立的索引相比，大小并没有显著增加。

2.4 本章结论

本章从英文中通常使用编辑距离来度量文本相似程度为出发点，结合汉语自身的实际情况，提出了基于拼音改良的编辑距离度量方式，来衡量两个汉语文本串之间的相似度。在实际的实验中，准确率和召回率都优于传统的编辑距离度量方式。同时，通过以检索代替匹配，离线对文本建立索引，所以本章使用的匹配算法在性能上也优于传统的文本匹配算法。

但是，本章提出的度量方式依然存在一些局限性。虽然在实际检索系统的输入中，拼音相关错误占有非常高的比例（在 Sogou 用户日志的统计中，这一比例超过了 90%），但是对于其它类型的错误，例如形近字和近义词带来的错误，本方法提出的距离度量方法并不能很好地进行度量和表征，依然有待进一步的研究。同时，本章的方法依然有较大的改进空间：可以引入语言学方面的知识，以及根据用户日志的实际统计结果，对于每一种改变赋予的权值采用更细致合理的规定。

第3章 以模糊匹配为核心的检索方法

信息检索的主要目的，就是从信息集合中返回与用户的需求最为吻合的那些信息作为检索的结果。具体到本文关注的文本检索领域，就是在文档集中为用户检出最相关的子文档集，并且按检出文档的相关程度进行排序，作为对检索用户所提出查询的回应。

在文本检索领域，用户的输入以及待检索的目标都是纯粹的文本，因而判断文档集合中的某个文档是否与用户的需求相满足，主要的判断依据就是文档与查询之间的文本相关性。在这种情况下，信息检索系统就面临一个问题：如何定义查询与文本的相关性，从而使得文本上的“相关性”能够真正体现对用户需求的满足。由此可见，在信息检索系统中，“相关性”是一个非常关键的概念，它的计算方式会直接影响结果对用户需求的满足程度，因而对于文本相关性的研究是十分重要和有价值的。

第 2 章提出的基于拼音的汉语相似度度量方法从拼音这个维度对汉语文本的相似度进行了衡量。在实际的应用中，由于拼音输入法已经成为用户主要的输入手段，因而，从拼音维度对相似度进行衡量是十分有意义的。

因此，本章将首先介绍在文本相关性方面已有的一些研究工作，然后在第 2 章文本模糊匹配的研究基础上，提出以模糊匹配为核心的检索方法，对经典的向量空间模型进行改进。最后，本章将通过实验来对这一检索方法的实际效果进行考察。

3.1 相关研究工作

3.1.1 文本相关性的研究历史

正如前文所述，“相关性”是信息检索技术的核心概念，国内外的学术界对相关性的研究持续了相当长一段时间，并且取得了众多的研究成果。

国外的信息检索领域实际上从 19 世纪 50 年代开始，就已经就“相关性”各个方面的问题展开了激烈的争论。1958 年国际科学信息会议（ICSI）上关于“相关性”方面的讨论就已经成为一个重要的话题。到了 60 年代，为了弄清相

关性的判断受何种因素影响，Cuadra and Katter 和 Rees and Schultz 分别进行了几次重要的实验性研究[20]。到上世纪 70 年代，关于“相关性”方面的框架工作也已经展开，如 Saracevic 对相关性的所有可能的层次框架做了细致的归纳[21,22]。80 年代之后，从认知的和动态的角度看待相关性成为一个新的研究要点，例如以知识表示（knowledge representation）为理论基础的观点认为，用户的判断实际上是在文档信息与头脑中已有的概念知识体系之间寻找匹配的过程。这种观点关注的是用户的内部概念体系与外部世界的相互作用以及内部知识与相关性判断的关系[23-26]。上世纪 90 年代的研究工作基本上延续了认知主义传统，而且研究得更加全面。以文献[27]代表的论述更加深入细致地剖析了相关性的概念意义，指出应该从系统(system)、认知(cognitive)、情境(situational)和动机(motivational)等多个角度来描述相关性。在这一段时期内特别值得关注的是，人机交互成为相关性研究的焦点之一。与此同时，Mizzaro 在 1998 年提出了一个重要的观点，认为时间也是相关性的一个维度，在人机交互过程中是无法回避的[28]。这一观点如今已经得到普遍认同。直到今天，相关性的讨论仍在热烈进行中[29]。

中文信息检索研究在很大程度上是在西方已有理论体系之内进行的，研究特别集中在汉语相对于西方语言的特殊性上，如分词的必要性、句法分析(parsing)以及索引(indexing)以什么为基本单位、汉语的自然语言处理等等方面，但对于像“相关性”应该如何定义和衡量这样的一些基础理论问题的研究则比较少。可以说，在这个问题上中文信息检索仍然处于一种直觉和感性的阶段，即更加注重于已有的检索模型如何在汉语中取得好的应用效果等技术层面。但是，汉语毕竟有着众多不同于字母语言的独特特征，所以从汉语的基本特点出发，在文本相关性的理论基础上进行更多的研究，应该是非常有意义的工作。

3.1.2 信息检索模型概述

信息检索的文本相关性研究始于从上世纪 50 年代，随着上世纪末本世纪初以 yahoo[4]和 google 为代表的商业搜索引擎的大规模的普及和实用化，信息检索的相关性计算体系已经有了比较成熟的发展，产生了很多不同的检索模型来刻画查询与文档之间的相关性，以便从文档集合中检索出符合用户需求的文档子

[4] www.yohoo.com

集。

所谓模型，是采用数学工具，对现实世界某种事物或某种运动的抽象描述。例如，检索模型就是使用数学工具，来刻画文档与查询之间的相关性。在理想情况下，面对相同的输入，模型的输出应能够无限地逼近现实世界的输出，例如，现在已经广泛使用天气的预测模型来根据现有的天气、气压、云图等输入参数计算出未来可能的天气变化。但是，模型并不等于实现，同样的一个模型可以用多种方法实现，例如，布尔模型可以用倒排文档(inverted file)实现，也可以用 B-tree 或者 bitmap 实现。

本章所讨论的信息检索模型，通常用这样一个四元组来表示：$[D,Q,F,R(q_i,d_j)]$，其中，D 是文档集的机内表示，Q 是用户需求也就是用户输入查询的机内表示，F 是文档表示、查询表示和它们之间的关系的模型框架(Frame)，而 $R(q_i,d_j)$ 表示查询 q_i 和 document d_j 之间的相关性得分。

信息检索模型决定了从什么样的视角去看待查询式和文档；基于什么样的理论去看待查询式和文档的关系；如何计算查询式和文档之间的相关性得分。

图 3.1 表示的是从代数论、概率论、集合论和人工智能这四个不同的角度来看待查询式和文档之间的关系并计算它们之间的相似度得分所衍生出的一系列不同的检索模型。

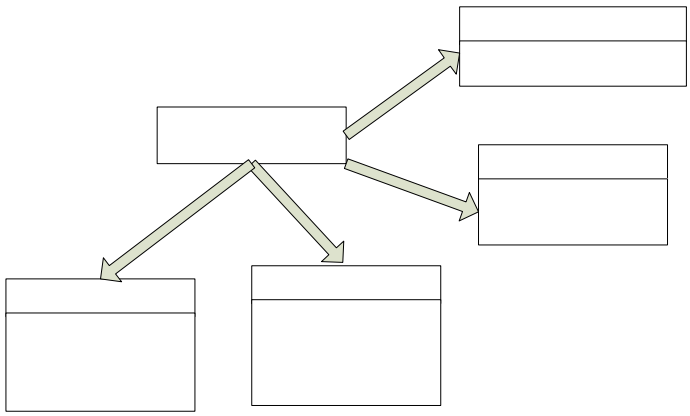

图 3.1 信息检索模型的分类

在这些检索模型中，布尔模型和向量空间模型，以及二者的结合是目前使用最为广泛、研究最为深入的检索模型，同时也是本章所关注和讨论的重点。本章将分别介绍这两种模型的定义、特点、实际的实现方式以及优缺点。

3.1.3 布尔模型

在布尔（bool）模型中，文档被表示为一系列关键词的集合，例如一篇文本为“今天天气不错，挺风和日丽的”，会被表示为“今天”、“天气”、“不错”、“，”、“挺”、“风和日丽”、“的”这些关键词的集合。

查询式则被表示为一系列关键词的布尔组合，例如一个查询可以被表示为“今天”&“天气”&（“不错”||“风和日丽”）。

在布尔模型中，检索流程——查询式和文档集合的匹配过程，就是寻求所有满足布尔查询式的文档的过程。例如，对于上面的那个查询，就是寻求所有“今天”和“天气”这两个关键词都同时出现，并且“不错”和“风和日丽”这两个关键词至少出现一个的文档。所有被匹配的文档，都会有相同的相关性得分，因而经典的布尔模型是无法对查询的结果进行排序的，在“扩展的布尔模型”中，通过对不同的关键词赋予不同的权重，可以实现检索结果按照相关性排序这一功能。

对于不同的布尔模型的实现，可以采用不同的分词或者 stemming 策略对文档和查询进行处理，同时也可以通过添加不同停用词过滤策略（stopword removal）策略来对结果进行细化调整。布尔模型可以有不同的实现方式，最简单的可以通过遍历文档集合中的每个文档判断它是否满足布尔查询式；或者对整个文档集合建立关键词倒排索引，对索引链根据布尔查询式进行“与/或”操作的链归并；另外，B-tree 和 bitmap 等数据结构也在布尔模型的实现中被广泛采用。

到目前为止，布尔模型依然是最为常用的检索模型之一。这是由于它的实现简单，而且查询式容易为用户所理解，经过初步训练的用户可以很方便地通过输入的文本查询式来实现较为复杂的查询意图，同时通过使用查询布尔表达式，可以很方便地控制查询结果。

但是布尔模型也被认为是功能最弱的检索模型。它的最主要问题在于不支持部分匹配，这意味着布尔模型进行的匹配过程会对结果造成很多的限制：这一匹配过程是非常刚性的，“与”意味着全部，“或”意味着任何一个。如果用户需要 n 个词中 m 个词同时出现的文档，就难以用布尔查询式进行表示。布尔模型很难表示用户复杂的需求，很难控制被检索的文档数量。原则上讲，所有被查询式匹配的文档都将被作为检索结果进行返回，如果不考虑索引词的权重，所有文档都以相同的方式和查询相匹配，无法对输出进行排序，。与此同时，很

难进行自动的相关反馈：如果一篇文档被用户确认为相关或者不相关，用户很难相应地修改查询式。此外在匹配过程中，布尔模型对于不同的关键词进行相同的处理，没有充分地利用关键词本身的属性来进行部分取舍。

以上这些缺点，导致经典的布尔模型难以在实际的系统中得到应用，但是布尔模型结构简单、实现方便、性能优异，它的检索思想依然被很多商业搜索引擎所借鉴。

3.1.4 向量空间模型

向量空间模型（vector space model）[30]，是主流的商业搜索引擎（如百度、google 等）进行相关性计算时的主要计算依据，为信息检索提供了建模的框架。研究者称向量空间模型为模型中的模型[31]。向量空间模型使得对查询向量中索引项权重的赋值成为可能，索引项权重模式改善了检索性能。利用计算得到的相似度可以对获取的文档按照相似度排序，可以使得与查询相关程度高的文档排在前面，从而有利于用户查找。向量空间模型的部分匹配策略使得文档的检索可以部分的匹配查询，从而能检索到更多的文档。由于向量空间模型实现容易，且性能明显优于大部分的信息检索模型[32]，当前的信息检索系统大多采用向量空间模型。

与布尔模型相同，向量空间模型的检索依然是基于关键词的，但是与布尔模型不同的是，在向量空间模型中，任何一个文本，不管是查询式还是待检索的目标文档，都由一个关键词列表来表示，而不是布尔查询式。例如，一个文本（可能是查询也可能是待检索的目标文档）是“今天天气不错，挺风和日丽的”，那么在某个向量空间模型中，它可能会被表示为<今天，天气，不错，挺，风和日丽，的>这样一个向量，同时向量中的每个关键词，都会有一个单独的权重，如<今天:0.5，天气:0.3，不错:0.1，挺:0.1，风和日丽:0.7，的:0>。这些权重可以由用户赋权，更一般的情况下，由模型根据统计信息予以赋权（具体的赋权策略，本文将在后面进行详细的介绍）。

在一个文档集合中，所有可能的关键词的集合被称为词表（vocabulary），它代表了这个文档集合所对应的应用中的重要词项。

如图 3.2 所示，对于一个专注于对数学领域的文档集合进行检索的信息检索系统，它选择了数学领域里一些专用名词作为检索的关键词词表。

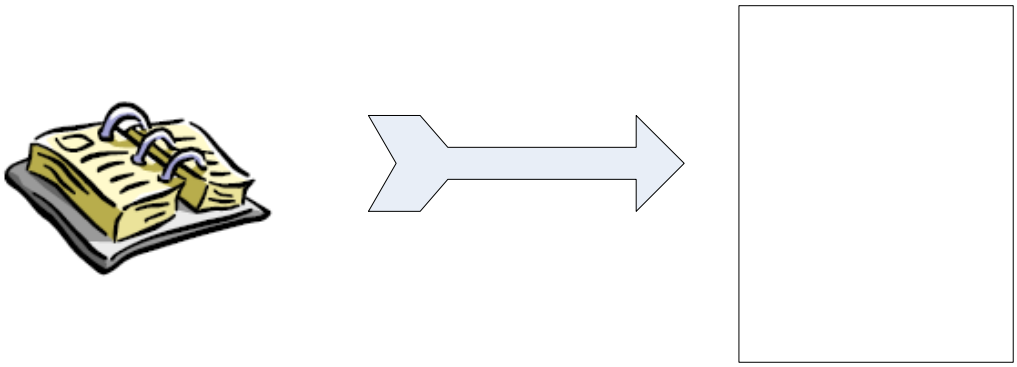

图 3.2 数学文档集中的词表

对于某个特定领域的文档检索，这样的词表选取方式可以很好地满足用户的需求，但对于没有限定领域，以所有互联网的网页文本作为检索目标的网页检索系统，所选取的词表通常只能选择所有词汇的全集。对于中文检索系统来说，这个词表的选择还受制于分词系统的输出，只有那些能够被分词系统识别出来的词汇，才有可能作为检索的关键词。

向量空间模型认为，词表中的这些关键词都是不相关的（un-correlated）或者说是正交的（orthogonal)，形成一个向量空间。如图 3.3 所示，词表中含有 N 个关键词，则这个向量空间就是 N 维的。

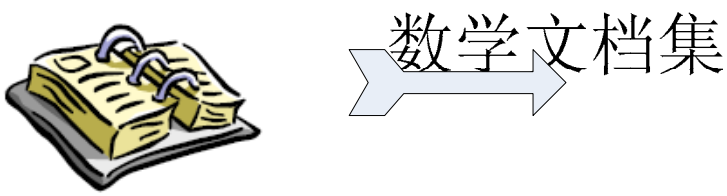

图 3.3 数学文档集对应的向量空间

但是实际上，这些关键词并不是完全独立的，它们之间依然是相互关联的。例如，当一个文档中存在“数学”，非常有可能同时存在“多项式”；当在一个文档中看到“数学”，有中等的可能性同时存在“物理”；当在一个文档中存在“数学”，只有很少的可能同时存在“爱情”，这种不同关键词之间的不同的共现概率，无法用经典的向量空间模型来刻画。

假如一个检索系统的词表含有 N 个关键词或索引项（关键词），那么一个文档或查询式可以表示为 N 个元素的线性组合，也就是一个 N 维的向量。特别的，这个检索系统中的所有 X 个文档，可以用图 3.4 中的 $X \times N$ 矩阵来进行表示。其中，m_{ab} 表明文档 a 中关键词 b 的权重，m_{ab} 越大，则这个关键词在这个文档中就越重要，当权重为 0 时，说明关键词在这个文档中没有出现或者忽略不计。

$$\begin{bmatrix} & T_1 & T_2 & T_3 & \ldots & T_N \\ D_1 & m_{11} & m_{12} & m_{13} & & m_{1N} \\ D_2 & m_{21} & m_{22} & m_{23} & & m_{2N} \\ D_3 & m_{31} & m_{32} & m_{33} & & m_{3N} \\ \ldots & & & & & \\ D_X & m_{X1} & m_{X2} & m_{X3} & & m_{XN} \end{bmatrix}$$

图 3.4 文档集合的矩阵表示

在查询和文档都用 N 维向量进行表示后，计算查询和文档之间的相关性就变为计算两个 N 维向量之间的相似度。图 3.5 表示一个词表数量为 3 的检索系统，它在向量空间模型下对应一个三维的向量空间。用户输入的查询式 Q 的向量化表示是<0，0，1>。文档集合中一共包含两个文档，其中文档 D_1 的向量化表示是<2，3，5>，文档 D_2 的向量化表示是<3，7，1>。这三个向量在向量空间中的状态如图 3.5 所示。对于这样一个用户输入 Q，信息检索系统的目的就抽象为查找 D_1 和 D_2 两个向量中与 Q 相似度最高的那个向量。

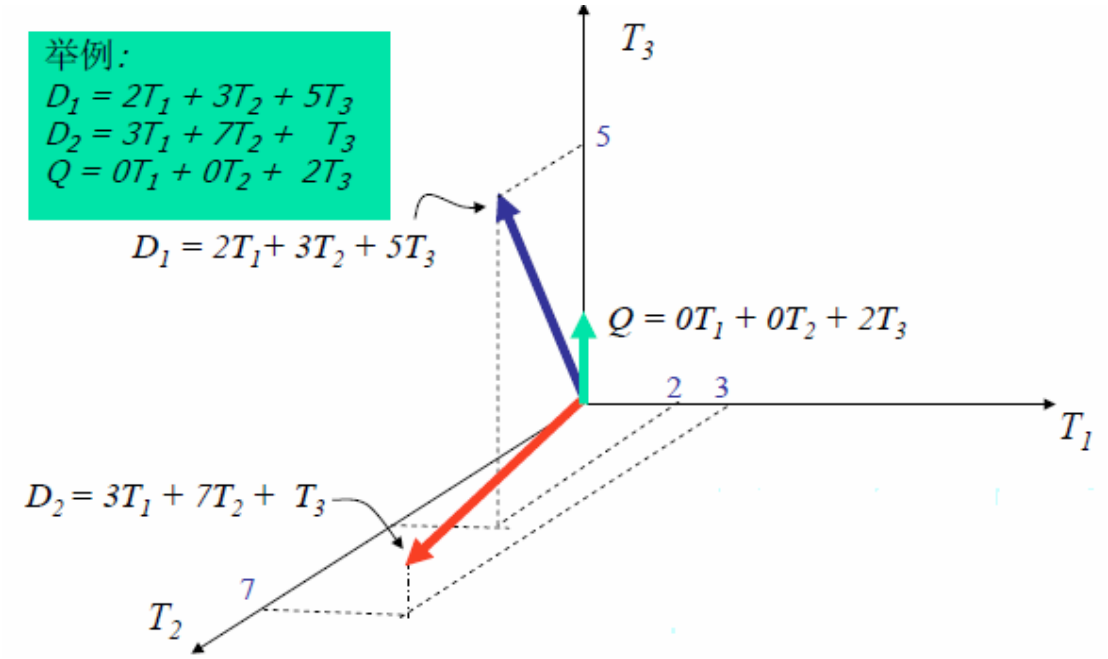

图 3.5 向量空间中的文档和查询式的向量表示

相似度是一个函数，它给出两个向量之间的相似程度，并由此刻画两个向量对应文本之间的相关性。由于查询式和文档都是向量，各类相似度存在于：两个文档之间、两个查询式之间、一个查询式和一个文档之间。其中，查询式和文档之间的相似度是信息检索系统所最关心的相似度计算。通过计算查询式和文档之间的相似度，可以根据预定的重要程度对检索出来的文档进行排序；也可以通过强制设定某个阈值，控制被检索出来的文档的数量；同时，检索结果可以被用于相关反馈中，以便对原始的查询式进行修正（例如：将文档向量和查询式向量进行结合）。

人们曾提出大量的相似度计算方法，但是不同的应用背景有着不同的需求，因此最佳的相似度计算方法并不存在。对于不同的应用需求，应该开发出不同的相似度计算方法。

内积（Inner Product）是一种广泛使用的相似度计算方法，它通过两个文本向量之间的内积，也就是一个向量到另一个向量上的投影的长度来衡量这两个向量所对应文本的相关性。内积越大，则这个投影长度就越大。

文档 D 和查询 Q 之间的内积相似度计算如公式（3-1）所示

$$Sim(D,Q)=\sum_{k=1}^{N}(d_k \bullet q_k) \qquad (3\text{-}1)$$

其中，d_k 表明关键词 k 在文档 D 中的权重，q_k 表明关键词 k 在查询 Q 中的权重。从这个公式可以看出，内积是查询式和文档中相互匹配的关键词的权重乘积之和。特别的，如果这个检索系统是一个二值的向量空间，关键词在向量中的取值只有 0（在文本中出现）和 1（在文本中不出现）两种，那么内积是查询式中的关键词和文档中的关键词相互匹配的数量。

内积的值越大，说明投影长度越长，两个文本越相似。在图 3.5 所示的例子中，查询 Q 和两个文档 D_1、D_2 的相似度利用内积来计算的结果就是：

$$Sim(D_1,Q)=2*0+3*0+5*2=10$$
$$Sim(D_2,Q)=3*0+7*0+1*2=2$$

可以看出，在用内积来衡量相似度的情况下，文档 D_1 与查询 Q 更加相似。

内积的相似度计算方式有它的一些独特属性：内积值没有界限，不像概率值要在(0,1)之间；内积的计算方法对长文档有利。内积用于衡量有多少关键词匹配成功，而不计算有多少关键词匹配失败，由于长文档包含大量独立词项，每个关键词均多次出现，因此一般而言，长文档和查询式中的关键词匹配成功的可能性会比短文档大。

除了两个向量的内积之外，两个向量的夹角在一定程度上也可以反映两个向量之间的相似度，如图 3.6 所示。在实际的计算中，通常使用余弦相似度来表征这一特点，两个向量的夹角余弦值越大，夹角就越小，可以认为两个向量更加相似。

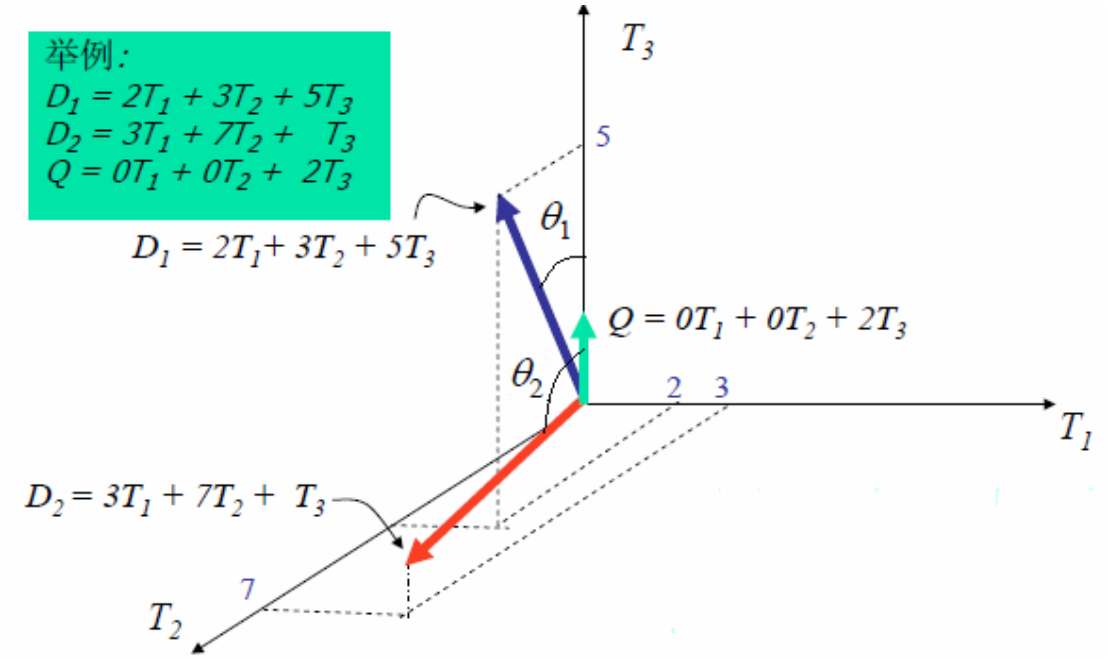

图 3.6 文档和查询式向量的夹角

余弦相似度的计算方法如公式（3-2）所示，它的实质是对内积的计算结果进行长度意义上的归一化。

$$CosSim(D,Q) = \frac{\sum_{k=1}^{N}(d_k * q_k)}{\sqrt{\sum_{k=1}^{N} d_k^2 \bullet \sum_{k=1}^{N} q_k^2}} \qquad (3\text{-}2)$$

在余弦相似度的计算体系下，示例中的两个文档和查询的相似度分别为：

$$CosSim(D_1,Q) = \frac{2*0+3*0+5*2}{\sqrt{(2^2+3^2+5^2)\bullet(2^2)}} = 0.811$$

$$CosSim(D_2,Q) = \frac{3*0+7*0+1*2}{\sqrt{(3^2+7^2+1^2)\bullet(2^2)}} = 0.130$$

从这个计算结果可以看出，虽然最终的计算结果都是 D_1 和 Q 更加相关，但是具体的相似度比例却不一样，在内积体系下，两者的相似度相差 5 倍，在余弦体系下，两者的相似度相差 6.24 倍。

内积和余弦相似度分别从投影和夹角这两个不同的几何意义上反映了两个向量之间的相似度，并据此衡量两个向量的相关性。除此之外，还有一些别的

相似度计算方法，如雅克比相关性（Jaccard Coefficient）等。

$$JacSim(D,Q)=\frac{\sum_{k=1}^{N}(d_k*q_k)}{\sum_{k=1}^{N}d_k^2+\sum_{k=1}^{N}q_k^2-\sum_{k=1}^{N}(d_k*q_k)} \quad (3\text{-}3)$$

实际系统中的相似度计算大都是在内积的基础上进行各种形式的归一化，很难讲它们之间哪一种效果更好，对于不同的应用场合，不同的文档集合，自有不同的最合适的相似度计算方法。而如何选择一个最切合的相似度计算方法，也是信息检索领域一个重要研究方向。

在实际的应用中，向量空间模型通常采用倒排索引与索引归并算法进行实现，算法的好坏直接决定了检索系统的性能，也关系到检索是否能够在时间效率上满足用户的查询需求。

3.1.5 关键词赋权

从 3.1.4 中的介绍可以看出，不管哪一种相似度计算模型，都极端依赖于 d_k 和 q_k 的取舍，也就是关键词在文档中的权重赋值，这种权重赋值的计算方法通常被称为权重赋值计算模型。

正如前面所提到的，在向量空间模型中，d_k 和 q_k（在不关心是文档还是查询式的情况下，下文中统一表示为 w_t）的赋值来源于文档和关键词的统计信息，最为主要的统计信息就是词频（term frequency），也被简写为 *tf*，它表明关键词在某个文档或者整个文档集合中的出现次数。除此之外，一些常见的统计信息还包括关键词出现的文档的总数，文档包含的独体关键词的总数等。

从这些统计信息出发，研究者们提出了一系列关键词赋权模型，如表 3.1[33] 所示：

表 3.1 各种关键词赋权模型

模型	权值计算公式
hits	$b_{q,t} \times b_{d,t}$
baseline	$f_{q,t} \times b_{d,t}$
tf	$f_{q,t} \times \frac{f_{d,t}}{dlf_d}$
idf	$f_{q,t} \times idf_t$
tf.idf	$f_{q,t} \times idf_t \times \frac{f_{d,t}}{dlf_d}$
log(tf)	$(1+\log(f_{q,t})) \times \frac{1+\log(f_{d,t})}{1+\log(avef_d)}$
log(tf).idf	$(1+\log(f_{q,t})) \times idf_t \times \frac{1+\log(f_{d,t})}{1+\log(avef_d)}$
log(tf).idf.dl	$(1+\log(f_{q,t})) \times idf_t \times \frac{1+\log(f_{d,t})}{1+\log(avef_d)} \times \frac{1}{avedlb + S \times (dlb_d - avedlb)}$
BM25	$f_{q,t} \times \log(\frac{N-n_t+0.5}{n_t+0.5})_t \times \frac{(K+1) \times f_{d,t}}{K \times \{(1-b)+b\frac{dlf_d}{avedlf}\} + f_{d,t}}$

q	查询
d	文档
t	关键词
N	文档集合中的文档综述
n_t	关键词 *t* 对应的文档总数
$b_{x,t}$	关键词 *t* 在文本 *x*（文档或查询，下同）中是否存在
$f_{x,t}$	关键词 *t* 在文本 *x* 中的出现次数
idf_t	关键词 *t* 的 *idf* 值
dlb_x	文本 *x* 的独立关键词数
dlf_x	文本 *x* 的所有关键词的词频之和
$avef_x$	文本 *x* 的所有关键词的词频的平均值
avedlb	文档集合中所有文档的 dlb_x 的平均值

在这么多的赋权模型中，影响最大、应用最为广泛的是 *tf-idf* 模型和 *BM25* 模型[33]。

idf 指关键词的逆文本频率，是该关键词在文档集合中分布情况的量化。*tf-idf* 模型由 Gerald Sahon 和 McGil 提出，它既考虑了索引项的重要性，又考虑了整个文档集合中文档之间的关系，给出的索引项权重较有代表性，取得了良好的检索效果。

BM25 模型由微软剑桥研究院提出，它基于概率模型，在 *tf* 的基础上引入了文档总数、关键词对应文档总数等多项统计信息来进行归一化和调权，在实际的应用中也取得了很好的效果。

3.1.6 向量空间模型的不足以及现有的解决方案

现有的主流商业搜索引擎以向量空间模型为基础，采用 *tf-idf* 以及它的改进作为关键词的赋权模型，在实际的应用中取得了良好的效果。但是，向量空间模型依然是以关键词匹配为核心，它的检索过程依然是严格意义上的文本精确匹配。

图 3.8 是一个典型的用户使用检索系统的流程图。

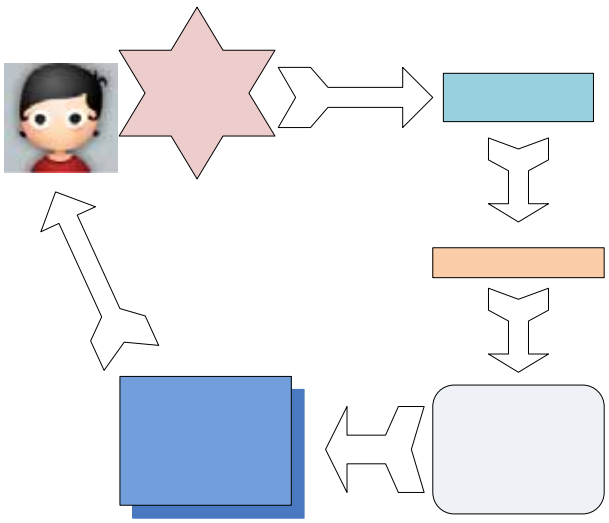

图 3.7 一个典型的检索流程

从用户具有一个检索需求开始，为了使用现有的以文档输入为核心的检索系统，用户需要在自己的脑海中将需求用文本表示（大多数情况下这个过程是

无意识的)。然后，要使用各种输入手段，例如拼音输入法、语音识别工具、手写板等将文本化的意图输入到电脑、手机等终端设备中，然后通过网络传输或者短信等手段将查询从用户终端传递给检索系统所在的服务器。检索系统在得到查询之后，会进行检索，得到的检索结果通过网络或短信等方式展现给用户。

从这个图中可以看出，从用户具有检索需求，到检索系统得到查询之间经过了三步：文本化、输入和传输。其中，传输这个过程相对而言比较稳定可靠的，而“文本化”和“输入”过程都有可能包含错误，使检索系统得到的查询最终与用户的意图发生偏差。

“文本化”过程中，当用户选择的词汇并不恰当时，会影响最终的检索结果。例如，检索系统的一个用户要在中关村购买计算机，他决定用“中关村计算机价格”这样一个文本来表示他的检索需求。但是在实际的现实生活中，大部分中关村的商家都习惯于使用“电脑价格”而不是“计算机价格”这个词汇来宣传或者描述自己的业务。可以预计，用户如此的输入会漏掉非常多的有价值的含有“电脑价格”而不是“计算机价格”的检索结果。这是一个同义词选取不当的例子，而用户还可能因为选择了过于宽泛的文本而影响检索结果对需求的满足。例如，在《苹果》这一部电影热映时，用户试图获取与电影相关的信息，因而选择“苹果”作为自己的文本化意图。但是，“苹果”包含水果、电影、数码用品等多个领域的含义，所以选择这样一个文本化意图最终将影响检索结果对用户需求的满足。

当用户输入自己的文本化意图时，也经常会发生错误，对检索结果造成影响。例如，一个用户试图搜索与操作系统有关的信息，他选择了“计算机操作系统”作为自己的文本化意图。但是，由于方言背景的影响，用户在使用拼音输入法时误将“作”的拼音由/zuo/拼成/zhuo/时，拼音输入法的结果就变成“计算机曹卓系统”了，并且用户自身并没有意识到这一错误就直接将查询传递给了检索系统。此时，系统给出的检索结果（图 3.9）虽然对于“计算机曹卓系统”是相关性良好的（所有的关键词全部命中），但与用户的实际意图有很大差距。


曹卓的博客 - 欧逸轩 - 网易博客
曹卓的博客 - 欧逸轩 - 网易博客 自立 自信 人人为我,我为人人 引用 ...(2)具有系统的专业基础理论和较广博的科技文化知识,精通本专业技术知识,实践...计算机技术、市政、市政工程、市政道路工程、概预算、结构、安全、讲师、统计师等...
caozhuo0629.blog.163.com/ 125K 2008-12-19 - 百度快照

基于S3C4510B的嵌入式网关的设计与实现--《计算机工程与设计》200..
《计算机工程与设计》2008年18期 加入...介绍了基于S3C4510B硬件平台和uClinux操作系统设计的嵌入式网关,完成了TCP/IP...1曹卓; 基于ARM9的嵌入式网关的设计 [D];大连海事大学; 2008年 2陈凯明; 基于Linux的家庭智能系统研究 [D];...
www.cnki.com.cn/Article/CJFDTotal-SJSJ200 ... 29K 2008-11-4 - 百度快照

三分仓回转式空气预热器计算机辅助计算 Computer Auxiliary Calcu..
回转式空气预热器密封系统改造 The Transformation of Sealing System of Round Air Pre-heater[科学之友 New Tempo Science] 王智磊 , 曹卓炫 , Wang Zhilei , Cao Zhuoxuan 回转式空气预热器非稳定换热的分析 Analysis of Unsteady ...
www.ilib.cn/Abstract.aspx?A=hzdl200104008 75K 2007-2-2 - 百度快照

【XLS】"2006年9月市直单位考聘专业技术人员进入面试人员名单...
文件格式:XLS/Microsoft Excel - HTML版
"信息管理与信息系统(本科)","0628011227",72,1 36,"..."计算机科学与技术 (本科)","0628010112",72,1 45,"...(本科)","0628014028",71,2 233,"曹雪","电视台",..."曹卓","临沂卫校","英语","0628011910",74,1 325,...
cl.tancheng.gov.cn:8080/webpage/default/p ... 72K 2006-11-7

【XLS】"南昌大学选课开课学生名单"
文件格式:XLS/Microsoft Excel - HTML版
(电力系统)" "6101105044","景国秀","二选","电自052班(电力系统)" "..."曹绍锦","二选","计算机043班" "6103104013","陈鹏","二选","计算机..."5502305003","曹卓","二选","光信息051班" "5502305009","陈秀","二...
jwc.ncu.edu.cn/admin/eWebEditor_jwc/Uploa ... 6152K 2007-9-28

图 3.8 用户的查询与意图不相符

以上两类偏差最终都会传递给检索系统“错误”的查询，造成最终用户满意度下降，成熟的商业搜索系统对此采取了一系列的应对措施。例如，通过日志分析，大部分商业搜索系统都提供了查询改写功能，如图 3.10 所示。用户再次点击搜索引擎的提示，或者自己改写输入的查询，都可以得到比较好的检索结果。

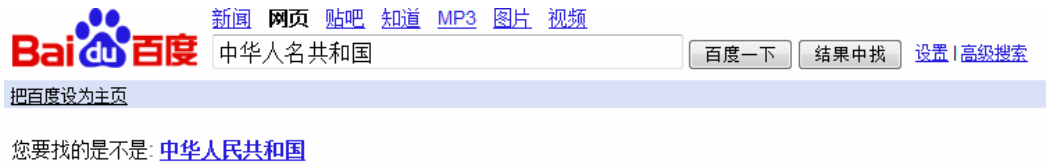

图 3.9 百度的查询改写功能

当然，很多时候由于分析技术的局限性，改写功能并不会完全正确。如图

3.11 所示，用户在制作 ppt 时需要获取一些图标的模板，因而选择“ppt 图标模板”作为自己的查询，但是检索系统却根据日志分析以及对资源的吻合程度，直接将查询改写成了“ppt 图表模板”，给用户体验带来了比较恶劣的伤害。

图 3.10 造成负面影响的查询改写结果

用户日志分析还有助于对用户的查询进行扩展，例如在《苹果》电影上映期间，将用户输入的“苹果”这种泛需求查询扩展成“苹果 电影”就可以对大部分用户的需求进行更好的满足（虽然这样的扩展以伤害小部分用户的体验为代价）。

除了根据用户日志进行分析，商业检索系统还普遍使用同义词替换来对查询的检索结果进行扩展，当然如何选取合适的替换词以及替换后的结果和原结果的结合方式，都需要做很多研究。

虽然商业搜索引擎采用上述处理手段解决了部分不良影响，但是对于很多应用来说，并没有足够的用户日志来完成日志分析。同时很多种应用，例如手机上的短信检索，要求用户改写查询是一项代价很大的行为，所以研究在这些应用情况下如何对用户的“错误”查询自动进行扩展，是十分有意义的。

3.2 模糊检索方法

3.2.1 汉语的拼音输入

随着电脑在普通老百姓生活中的普及，特别是易学易用的各种拼音输入方法的成熟和发展，拼音输入法已经成为图 3.8 中所描述的“输入”中的主要方法，拼音输入法的输出结果已经成为搜索引擎所处理的查询的主要来源。

与其它字母文字不同，汉语的读音和字形字义之间并没有直接的联系。尽管上世纪中期一度试图推行汉语的“拼音化”、“拉丁化”，但归根结底，汉字才是汉语表意的核心，而拼音始终只是一种辅助记忆和学习的工具。因此如图 3.12 所示，在使用拼音输入法时，从“文本化的意图”到“查询”之间，还有一个转化为“拼音”的过程。

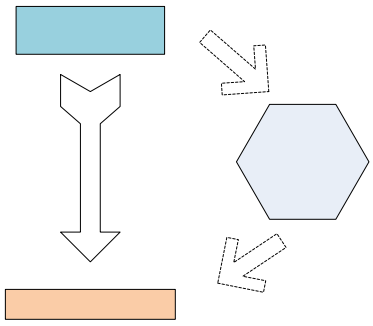

图 3.11 拼音输入流程

从“文本化的意图”到“拼音”这个过程，由于地域方言背景，或者自身拼音系统的不完善，导致生成的“拼音”含有错误。例如，南方方言区的人们普遍存在前后鼻音不分、鼻音边音混淆以及在卷舌音上的障碍，这些都极易导致拼音错误。这一类拼音错误的主要表现形式就是近音词的误用。

另一方面，从拼音到查询这个过程也会存在错误。同音字现象对于汉字来说十分普遍，即使是以词汇为单位输入，同音词也依然在汉语中广泛存在。当前的主流拼音输入法虽然根据统计信息以及用户自身的输入习惯对同音字同音词的顺序进行了一定的调整，但是依然无法避免用户不经意的错误输入。这一类拼音错误的主要表现形式就是同音词的误用。

从上面的分析可以发现，作为汉语信息检索系统的主要输入方式，在用户的“文本化意图”变成检索系统的“查询”这个过程中，拼音输入法会带来很

文本化的意图

输入

多错误。实际的抽样统计结果也证明这一观点。从搜狗实验室[5]提供的用户查询日志中随机抽取 2000 条用户查询日志进行查看，其中有 71 条明显含有拼音错误。如果能对这一类错误进行进一步的处理和分析，能够使检索系统更好地满足用户的需求。

3.2.2 模糊检索的实现

从前面的分析可以看出，拼音错误是汉语信息检索系统面临的一个重要问题。本文在第 2 章提出了基于改良的拼音相似度的模糊匹配算法，而检索过程的实质就是关键词的匹配，所以很自然地可以利用模糊匹配算法来对经典的检索模型进行改进。

最直接的想法是利用拼音的模糊匹配方法对关键词进行替换，这一替换过程可以用拼音对整个关键词表建立索引来实现。每一个关键词被替换成它的近音词或者同音词之后，就可以得到一个新的查询。接下来的工作，就变为对新的查询进行检索。当然，由于是替换之后的结果，所以需要对最终的相似度权值进行降权处理：

$$Sim(D,Q)=\max_{Q'\in extend(Q)} Sim(D,Q')Ext(Q,Q') \qquad (3\text{-}4)$$

在公式(3-4)中，$extend(Q)$ 是对 Q 进行关键词替换之后得到的扩展查询的集合（也包括 Q 本身），$Ext(Q,Q')$ 是扩展之后的查询 Q' 和原查询 Q 的相似度。它的计算遵循如下原则：

$$Ext(Q,Q')=1-\frac{Dis(Q,Q')*2}{Len(Q)+Len(Q')} \qquad (3\text{-}5)$$

在公式(3-5)中，$Dis(Q,Q')$ 函数表明在字符串相似度度量体系内，查询 Q 和查询 Q' 之间的距离；$Len(Q)$ 和 $Len(Q')$ 分别是 Q 和 Q' 的长度。$Ext(Q,Q')$ 的取值范围被归一化到 0 和 1 之间，$Dis(Q,Q')$ 越小，也就是在汉字串相似度度量体系内两者越接近，则 $Ext(Q,Q')$ 就越大，从而计算得到的相似度得分就越大。

由于部分正确的查询也有可能被扩展，所以为了避免最终扩展的结果影响

[5] http://www.sogou.com/labs/，由中文搜索引擎搜狗(http://www.sogou.com/)提供的数据资源、研究合作平台

原有的检索结果，需要将扩展之后的结果按照相关性分为三个档次，如图 3.13 所示。

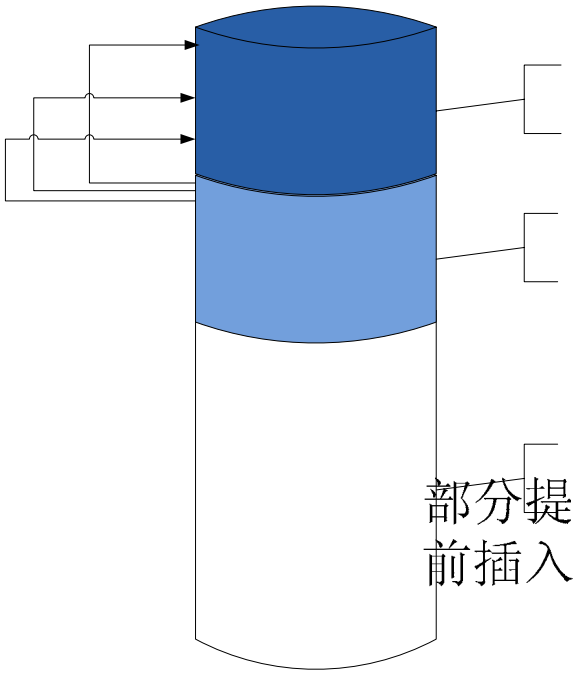

图 3.12 对检索结果进行扩展

第一档次是原查询的关键词全命中的结果，第二档次是扩展查询的关键词全命中的结果，第三档次是原查询的剩余检索结果。对于同一个文档，如果在多个档次中都出现，或者在同一个档次内部多次出现，则只保留其中的最高排名。当然，第二档次内部的检索结果相对于新查询是完全匹配，相对于旧查询实质上也是部分匹配，所以本文提出的模糊检索系统的实质，是将完全匹配的结果无条件提前，而对部分匹配的结果则引入拼音的模糊匹配，分两部分进行处理。同时，为了使得展现给用户的靠前的结果里面能够有部分来自于扩展的结果，所以，通过将第二档次排名最靠前的结果部分提前的方式，保证了在第二档次结果足够的情况下，检索结果的前 3 条，第 5-10 条和第 10-20 条三个结果档次中都至少有 1 条展现给用户的结果来源于第二档次，也就是查询扩展得到的结果。

需要特别说明的是，如果按照经典的向量空间模型，关键词全命中的得分有可能比关键词没有命中的得分要低，但是出于对最终检索结果的相关性考虑，

人为地对检索结果进行了控制。在第一档次和第三档次内部，文档的得分是按照向量空间模型中的相似度计算得到的。

3.2.3 分词边界的影响

由于输入含有拼音错误，所以分词的结果也会受到影响，特别是当分词的边界发生变化时，如表 3.1 所示：

表 3.2 拼音错误对分词的影响

正确输入	正确输入的分词结果	错误输入	错误输入的分词结果
中华人民共和国合同法	中华人民共和国、合同法	中华人名共和国合同法	中华、人民、共和国、合同法
篮球火	篮球火	蓝球火	蓝球、火
dnf 私服	dnf、私服	dnf 似乎	dnf、似乎
纳兰性德	纳兰性德	纳兰新德	纳兰、新德
十七届三中全会精神	十七届、三中全会、精神	十一届三宗全会精神	十一届、三宗、全会、精神
哪儿可以下载 wow	哪儿、可以、下载、wow	那儿可以下载 wow	那儿、可以、下载 wow
试衣服的窍门	试衣服、的、窍门	是衣服的窍门	是、衣服、的、窍门
江枫渔火对愁眠	江枫、渔火、对、愁、眠	江风鱼火对愁眠	江风、鱼、火、对、愁、眠

从上面的示例可以看出，一般而言，错误的拼音输入对于分词主要会造成两方面的影响：

1. 正确的关键词变成错误的关键词，但依然是一个关键词整体；
2. 正确的关键词由于输入错误，被切碎成两个或多个关键词。

除此之外，很少有因为输入错误导致原本两个或多个关键词被切成 1 个关键词的情况，这主要由于拼音输入法在用户输入时就是以词为整体进行处理的。

因而，在对原查询进行扩展时，除了依次尝试对每个关键词进行拼音扩展以外，还应尝试对前后相邻的 2 个或 3 个关键词合在一起，查看它们是否有可能扩展出一个完整的关键词。

为了保证最终扩展出来的查询与原查询有足够的相似度，查询扩展必须遵循以下原则：

1. 新查询与原查询之间只能通过一次替换扩展而来；
2. 替换后的汉字串必须是词表中存在的一个关键词；

3. 替换的新旧汉字串之间的距离必须小于 1 个韵母的差距。

遵循这些原则虽然会使得部分错误的查询无法变换成正确的查询，但是基本保证了替换之后的结果不会与原结果相差太远。

3.2.4 offset 调权的应用

由于第二档次内部的结果都是替换后的新查询全部关键词命中的结果，再继续套用相似度计算公式就不太合适，因而本文引入 offset 调权来对其中的结果进行排序。

offset 调权的思想是：对于一个查询（由关键词 *A*，*B* 和 *C* 组成），除了关键词的命中数量以外，关键词在文档中的相对位置也会影响到文档和查询的相似度。可以用一个例子来帮助理解：两个文档 D_1 和 D_2，它们的原始文本分别为 *ABCDEFEDEE* 和 *CBAJHKLLIOP*，它们都分别命中了 *A*，*B* 和 *C* 三个关键词各一次，按照经典的向量空间模型，这两个文档的得分必定是完全一样的。但是直观来看，文档 D_1 相比于文档 D_2 与原查询更加相似。这意味着除了关键词的匹配信息以外，关键词的位置分布对于相似度计算也是十分重要的。

正是基于以上思想，我们提出了 offset 排序的概念，offset 值的计算可以参考表 3.2 所示的例子，它的实质是关键词在查询和文档中的位置偏移的二次平均值。这个平均值越小，说明查询和文档中关键词的位置分布越相似，从而认为两者越相似。

表 3.3 offset 调权计算实例

	关键词 A	关键词 B	关键词 C
在查询中的位置信息	A_1	B_1	C_1
在文档中的位置信息	A_2	B_2	C_2
位置差值（diff）	$\lvert A_1 - A_2 \rvert$	$\lvert B_1 - B_2 \rvert$	$\lvert C_1 - C_2 \rvert$
位置差值的平均（avg）	$\dfrac{diff_A + diff_B + diff_C}{3}$		
二次平均值	$\dfrac{\lvert diff_A - avg \rvert + \lvert diff_B - avg \rvert + \lvert diff_C - agv \rvert}{3}$		

表 3.2 展示的是文档中每个关键词只被命中一次时的 offset 计算公式。通常情况下，一个文档内命中 1 个关键词的次数不止 1。这时需要遍历每一种可能的排列组合，以其中 offset 计算的最小值作为最终的 offset 计算结果。出于实际的

效率考虑，我们采用了一种折中算法，如图 3.14 所示，在计算完最开始的一个组合“*ACB*”的得分之后，再计算下一个 *A* 替换第一个 *A* 之后的组合的 offset 得分，因为一次只替换一个，所以只需要重新计算一个关键词的 diff，而 avg 也可以通过减去旧的 diff 再加上新的 diff 来得到。虽然这一算法最终无法得到全局最优解，但是把时间复杂度从指数级下降到了 O(n)级，大大加快了 offset 的计算速度。

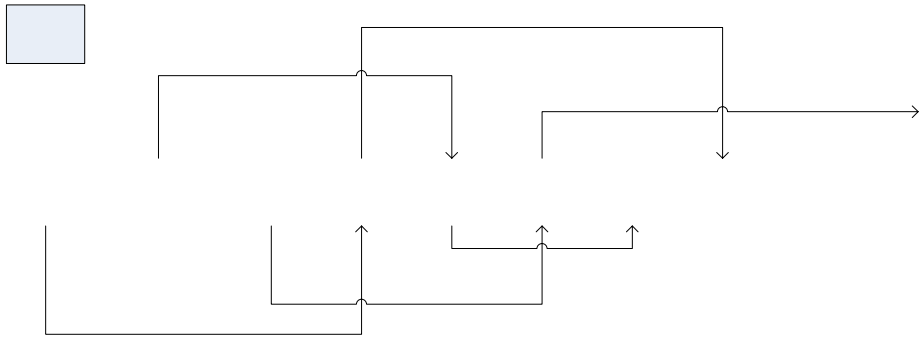

图 3.13 关键词命中多次时的 offset 计算

由于 offset 越小，位置相似度就越高，所以在第二档次内部，文档完全遵循 offset 得分按照从小到大的顺序进行排列。

3.3 实验

查询 ABC

3.3.1 实验设置

实验采用了搜狗实验室提供的网页文本数据，从中选取了 100 万个网页文本作为本章实验的文档集合，这些网页文本的平均长度大约是 1100 个汉字，整个文档集合在磁盘中的存储大小是 1.1 GB。实验一共选取了 100 个由于拼音输入导致的错误查询，作为检索系统的查询输入集合。

文档 A C B A

实验采用经典向量空间模型作为对比的检索系统。在对比系统和本文提出的模糊检索系统内部，都使用经典的 *tf-idf* 作为关键词赋权模型，采用内积作为相似度得分计算方式。

3.3.2 实验结果和分析

对于每一个查询，实验都采取人工评价的方法，对它的 Top 20 的检索结果进行相关性打分。打分分为两级，认为文档与用户需求相关，则这个结果得 1 分，不相关则得 0 分。计算所有 100 个查询对应的 2000 个检索结果的得分之后，就可得到最终的相关性得分。

实验对经典的向量空间模型、加入拼音扩展的模糊检索模型以及在第二档次内部再次加入 offset 调权的模糊检索模型，这三者的检索结果分别进行了人工评估，最终的评估结果如图 3.15 所示。

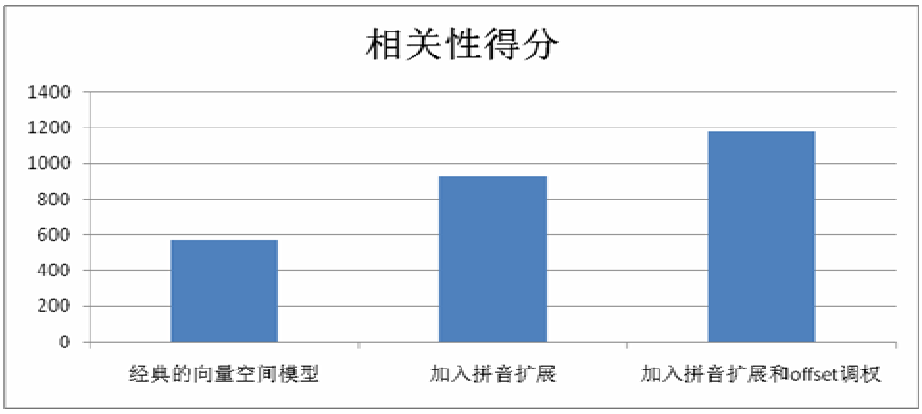

图 3.14 100 条错误查询的相关性得分评估结果

从这个结果可以看出，向原有检索结果中加入基于拼音相似度之后的扩展结果，会对检索结果的相关性有比较明显的提升。而在第二档次结果内部，采用 offset 得分来代替原有的相似度得分，对于最终展现出来的结果也是很有帮助的。

同时，为了考察模糊检索系统对于正确查询的影响，我们在随机抽取的 100 条正确查询上重复了上面的实验，实验结果如图 3.16 所示。

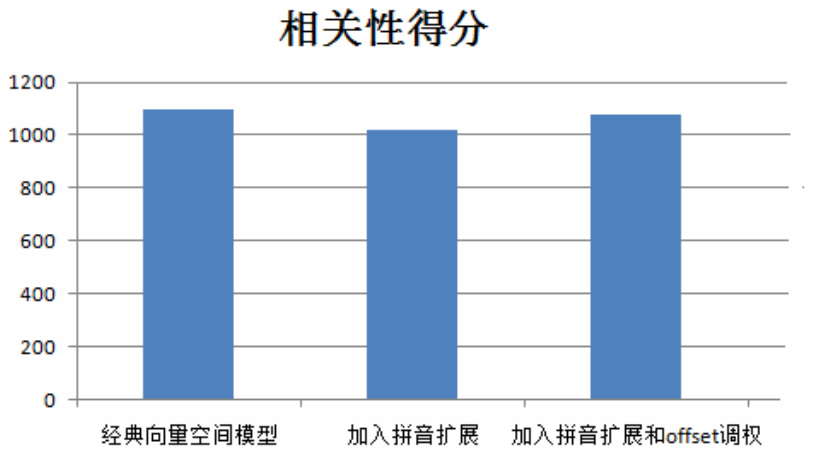

图 3.15 100 条正确查询的相关性得分评估结果

从这个实验可以看出，进行拼音扩展会对正确查询的检索造成部分负面影响，这是因为对于大部分查询来说，虽然已经保证全部匹配的结果排名最靠前，但是对于部分查询来说，它的“部分匹配”的检索结果中也有一些能够满足用户需求，但是在插入拼音扩展的结果之后，这部分结果会被推到第三档次，导致不能进入 Top 20。采用 offset 调权之后，由于扩展结果实际上也是“部分匹配”，所以调权实际上起到了对部分“部分匹配”的结果进行了调整，取得了较好的效果。

综合上述两个实验可以看出，虽然模糊检索系统会对大多数正确查询的检索结果略有不利的影响，但是它对于拼音错误的查询的处理结果有着大幅提升。

3.4 本章结论

本章从介绍目前使用最为广泛的向量空间模型出发，对整个检索系统的相关性评价体系进行了比较详细的介绍和梳理，同时详细分析了目前主流商业搜索引擎针对用户需求无法满足所做的特殊处理。针对目前普遍存在的拼音输入错误，结合第 2 章中的汉字相似度模糊匹配方法，对原有查询进行扩展，并将扩展查询的查询结果以合适的方式添加到原有检索结果中。同时，本文进一步采用 offset 调权的策略，对插入结果进行优化排序，进一步地改善检索结果的相关性。实验证明，本文提出的这一新的检索模型，相比较经典的向量空间模型，在检索结果上取得了明显的提升。

本文提出的模糊检索系统比较适合于应用在手机搜索平台或者语音输入搜

索平台这类具有大量拼音错误的检索场合，能够有效改善最终检索结果。

第4章 检索过程的性能优化

在第 3 章中，本文对于经典的向量空间模型进行了扩展，将一次查询变为了多次查询，因而会对时间性能造成一定的损失。本章将通过对检索过程中归并算法的优化，来力争提升检索的效率。

模型是对现实世界的抽象化处理和描述，同样一个模型，可以有不同的实现方法，不同的实现方法，也会有着不同的性能。

向量空间模型有多种不同的实现方法。直接从模型的意义出发，最为直观的实现方法就是遍历文档集合中每一个文档对应的向量，分别计算这些向量与查询向量之间的相似度，然后按照相似度排序，返回相似度最高的那些文档作为检索的结果。假设文档集合中所有文档的数目是 N，那么这个检索过程就需要完成 N 次向量相似度计算。基于向量空间模型的基本原则，如果查询与文档之间没有任何关键词匹配，那么它们的相似度得分就是 0。对于这类文档，在进行匹配的过程中完全没有必要计算它们的相似度。因而，仅仅计算与查询间至少有一个关键词重合的那些文档的相似度，就可以保证最后的检索结果的完备性。

出于这一考虑，现有的信息检索系统普遍采用倒排索引来记录关键词与文档之间的联系，并通过索引归并的方式找出所有包含查询中关键词的文档，并且计算它们的相似度得分。但是，即使进行了这一步过滤，当文档规模过大时，检索性能依然可能无法满足用户的需求，因而商业搜索引擎普遍采用部分牺牲检索结果召回率的折中方法，进一步提高检索过程的效率。

另一方面，对于大部分检索应用来说，用户只关心相似度最高的那些检索结果（Top N 的结果），因而如果在索引归并的过程中能够提前过滤掉最终得分不可能进入 Top N 的文档，并且在找到 Top N 的所有文档之后提前结束索引归并，就可以在保证返回给用户的前 N 个结果的准确性和完备性的条件下，大大地加快索引归并的过程，从而加快整个检索过程。

4.1 相关研究

4.1.1 倒排索引

采用向量空间模型的信息检索系统，普遍采用倒排索引（inverted index）来建立关键词到文档的映射关系。倒排索引是一种面向关键词的索引机制，利用它可以提高检索时的速度。如果使用正常的索引结构，建立的是“文档到关键词”的映射关系，在使用倒排索引技术后，建立的是“关键词到文档”的映射关系。

假设现在有两篇文档：文档 *A* 和文档 *B*。文档 *A* 的内容是：“今天天气不错”；文档 *B* 的内容是：“明天天气如何”。它们分词之后的结果分别为<今天、天气、不错>和<明天、天气、如何>。下面对这两个文档建立索引结构。

如果建立的是一般的索引结构，那么会有如表 4.1 所示的关系。

表 4.1 一般索引

文档编号	关键词	出现次数	关键词	出现次数	关键词	出现次数
A	今天	1	天气	1	不错	1
B	明天	1	天气	1	如何	1

从中可以看出，一般的索引结构是以文档为标准建立索引结构的，即它记录的是一篇文档中所有单词出现的情况。比如在文档 *B* 中“明天”、“天气”、“如何”均出现了一次。然而，用户在进行检索时，都是输入关键词进行查询，如果使用这种索引结构，在查询某一关键词时往往需要遍历所有的索引，当索引量非常大时，效率会成为一个很大的问题。

倒排索引恰恰解决了这个问题，它是以关键词为标准建立索引的，如表 4.2 所示。

表 4.2 倒排索引

关键词	文档编号	出现次数	文档编号	出现次数	文档编号	出现次数
今天	*A*	1				
天气	*A*	1	*B*	1		
不错	*A*	1				
明天	*B*	1				
如何	*B*	1				

从表 4.2 可以看出，倒排索引描述了一个关键词在所有文档中的出现情况，比如说关键词“天气”在文档 *A* 和文档 *B* 中分别出现了一次，而单词“不过”只在文档 *A* 中出现了一次。

通过比较可以发现，一般的索引结构建立的是一种“文档到单词”的映射关系，而倒排索引建立的则是一种“单词到文档”的映射关系。因为在日常的检索中，通常都是按照关键字进行搜索的，所以倒排索引可以更好地适合这种检索机制的需要。这也是倒排索引被大规模使用的原因。

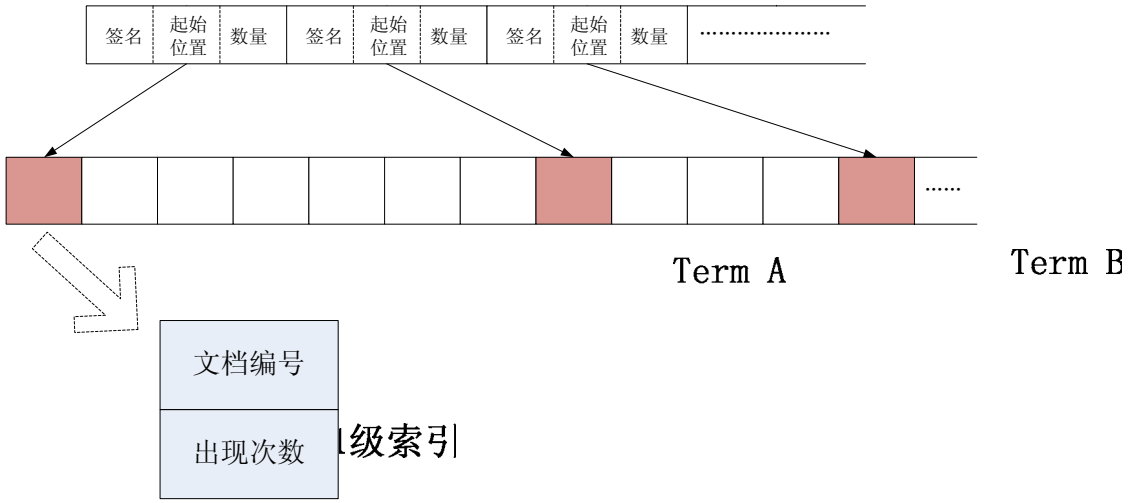

图 4.1 一个典型的 2 级倒排索引结构

在实际的检索系统的实现过程中，都采用多级索引结构以加快索引的定位和读取速度。

如图 4.1 所示是一个典型的 2 级倒排索引结构。在 1 级索引结构中，每一个关键词对应一个单元（下文称为结构体）；在 2 级索引结构体中，每一个关键词对文档的映射关系对应一个 2 级索引结构体。

1 级索引结构体记录的是每个关键词对应的 2 级索引信息，其中包括一个用于表征关键词唯一性的数字签名，本关键词对应的 2 级索引信息的起始位置以及含有的 2 级索引结构体的数目。选择数字签名而不是直接使用文本串表征关键词，是因为文本串不是定长的，会导致结构体变为不定长，从而对内存的分配以及结构体查找造成麻烦。数字签名是有序且内存大小固定的，因而可以方便地利用二分查找定位某个关键词的 1 级索引位置，并且据此定位它的 2 级索引位置。

1 个关键词对应多个 2 级索引结构体，每个索引结构体就代表关键词出现过的文档的信息，主要包括文档的编号以及关键词在文档中的出现次数等，对于部分检索系统，还会存储关键词在文档中的出现位置，以便实现 offset 调权。在同一个关键词对应的区域内部，2 级索引按照文档的编号进行顺序排列。

随着文档规模的增大、词表规模的增加，1 级索引的大小也会进一步增大。为了进一步加快索引的定位，商业搜索引擎通常会在 1 级索引上再添加一层 0

级索引，用于记录某个签名数字范围内的关键词对应的 1 级索引位置，形成典型的三层索引结构。

4.1.2 索引归并算法

索引归并是检索的核心过程，索引归并的目的就是根据倒排索引找出所有与查询有公共关键词的文档，文档和查询的相关性计算一般也是在这个过程中完成的。

当用户向检索系统输入查询之后，检索系统对查询进行分词和停用词过滤，得到一系列关键词，并根据多级索引结构读入每个关键词对应的 2 级索引。由于索引以文件形式保存在文件之中，所以读入索引涉及到内存与磁盘之间的 IO，包括索引的压缩与解压缩、应用层 cache 和系统 cache 的调度等技术，这些不是本文的关注重点。

得到查询中每个关键词的索引链之后，就可以开始索引归并过程。经典的索引归并算法如图 4.2 所示：

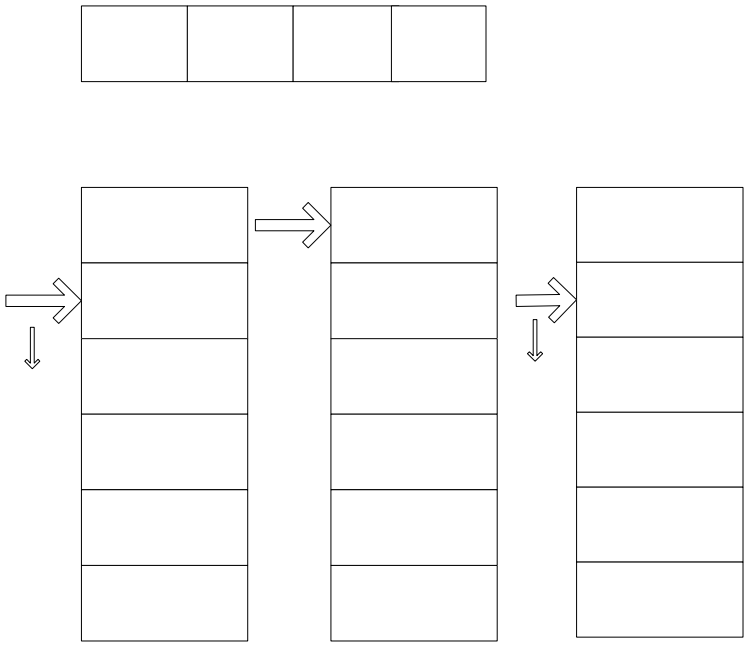

图 4.2 经典的倒排索引归并算法

图 4.2 所示对多个索引链进行“或”操作的过程与多路归并排序算法是非常相似的，它的具体运行步骤如下：

结果集合 tid=1 tid=2

1. 每个链上维护一个指针，指针所指的位置就是该链的当前位置。
2. 初始化将所有指针指向该链的第一个位置。
3. 通过最小堆、败者树等数据结构，找出所有链的当前位置中最小的文档编号（记为 *min_tid*）。
4. 根据相似度计算公式，计算 *min_tid* 对应的相似度得分（weight），并且把 *tid-weight* 二元组加入到结果集合中。
5. *min_tid* 对应的一个或多个链的指针向下移动一个位置。
6. 继续步骤 3，直到所有指针都移动到链尾部。
7. 对结果集合按照 weight 进行排序。

从这个运行步骤可以看出，假设所有关键词的索引链长度之和为 L，则不考虑最终的结果排序步骤，程序的时间复杂度为 O(L)，也就是要遍历完所有的关键词索引链。

注意到，由于一般只需返回 Top N 个检索结果，因而在索引归并的过程中，可以用一个大小为 N 的最小堆来维护结果集合，并且只把那些 weight 大于最小堆堆顶的 tid-weight 二元组加入到集合中。通过这种优化，可以加快最终的排序输出速度而并不影响实际的结果返回。但是，由于归并依然要进行到所有链遍历完成，所以归并过程的时间复杂度并没有得到改善。

4.1.3 商业搜索引擎的折中处理

检索模型的效果和效率是互相矛盾的：如果要保证结果的质量，则消耗的时间用户将无法接受；如果要减少消耗的时间，则通常以牺牲检索结果的质量为代价。

以最大的中文搜索引擎百度为例，它每天接受的用户搜索请求超过 10 亿，对于用户查询的平均响应时间保持在 100 ms 以内。百度索引的中文网页数量超过 80 亿，虽然已经对文档分库索引，检索时对每个小库的检索结果进行融合，但是由于机器的数量限制，每个小库的网页数量依然是很庞大的。一个典型的中文查询平均包含 2 到 3 个关键词，并且其中至少有 1 个关键词是比较常见的，也就是索引链长度比较长，这也就意味着有相当数量的文档含有查询中至少 1 个关键词。按照经典的索引归并算法，这些文档都需要计算相关性并且参与最终的排序输出，而这么多次计算对于实际的应用需求是无法接受的。

注意到，经典的索引归并算法其实质是求多个链的“或”操作，也就是只

要有一个关键词命中，文档也会被处理。但是从实际的相似度得分计算公式来看，命中关键词数量越多的文档，一般而言它的最终得分也就越高。因而商业搜索引擎普遍采用“与”操作来替代“或”操作完成最终的索引归并。“与”操作的实质就是只检索那些所有关键词全部命中的文档，使得最终匹配出来的文档数量大大减少。同时，由于“与”操作只需要找出那些在每个链上都出现的文档，而各条链都是按照文档编号有序排列的，因而在进行两条或者多条链的“与”操作时，可以采用“二分查找”或者“跳读”等技术手段完成这一操作。本文第 3 章在求第二档次的检索结果时，也是通过多个链的“与”操作，采用二分查找的手段来求扩展查询中关键词全命中的结果。

以图 4.3 为例，介绍“二分查找”完成两条链“与”操作的技术手段。

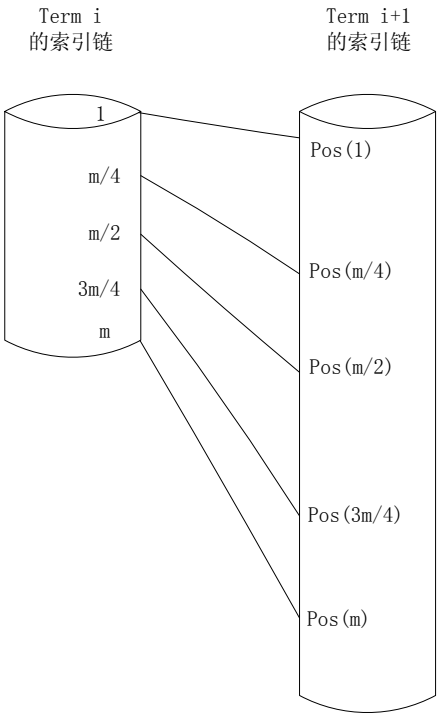

图 4.3 二分查找完成两条链的“与”操作

两个关键词 *i* 和 *i*+1 的链进行“与”操作，使用二分查找的实现细节如下：

假设关键词 *i* 的链长度为 *m*，也即含有 *m* 个文档（m 个 2 级索引单元），这些文档都是按照文档编号 tid 有序排列的。首先，在关键词 *i*+1 的链上，通过二分查找的方法寻找到关键词 *i* 的链中单元 1 内的 tid 的位置 Pos(1)和单元 *m* 内的

tid 的位置 Pos(*m*)，如果 1 和 *m* 对应的位置没有找到，则把二分查找最后的位置作为 Pos(1)和 Pos(m)的值。现在可以确定，由于链都是 tid 有序的，那么单元 *m*/2 内的 tid 的位置 Pos(*m*/2)必定在 Pos(1)和 Pos(*m*)之间，对它进行二分查找的时候，查找范围就缩小到了 Pos(1)和 Pos(*m*)之间。同样地，m/4，3m/4，m/8 等在进行查找时，都可以依次利用已有的结果，来逐步减少查找的范围。

“跳读”的思想与“二分查找”类似，都是利用链有序的特点来加快“与”操作的完成，它们的时间复杂度都低于对两条链进行遍历的时间复杂度。

采用“与”操作代替“或”操作，可以加快链归并的速度，同时减少最终匹配的文档数量，两者都可以加快索引过程。但是在文档数量非常巨大的情况下，即使采用“与”操作，时间消耗也可能依然无法满足需要，因而商业搜索引擎普遍采用“粘接”和“截断”策略来进一步减少检索的时间消耗。

所谓“高频词粘接”，是对那些出现频率很高链很长的关键词，在处理的时候将它和前一个或者后一个关键词合并成一个关键词进行处理。例如，用户的查询是“今天很凉快”，分词结果是<今天、很、凉快>。由于“很”是一个高频词，链必定很长，从而会导致检索过程耗时很多，因而在实际处理的时候，将“很”和“凉快”粘接到一起，形成一个新的粘接关键词，整个查询就变成了<今天、z_很凉快>。由于“z_很凉快”的链长度必定比“很”和“凉快”都要短，所以索引归并过程也会大大加快。当然，为了实现粘接功能，在对文档进行分词和建立索引时，也必须按照相同的策略对粘接关键词建立倒排索引。采用粘接策略会漏掉很多结果，例如如果文档中同时出现了“很”和“凉快”，但是两者位置并不相邻，这样的结果不会被匹配出来。

商业搜索引擎还采取“截断”策略，只读取部分最新的链，或者根据归并的结果数来提前终止归并过程。

不论是“或”操作变为“与”操作，还是“高频词粘接”策略或者是“截断”策略，都会漏掉部分结果。对于商业搜索引擎来说，它们的文档集合很多，即使溜掉了部分结果，展现给用户的还是非常丰富，因而可以接受这一损失，牺牲部分效果换取效率的提升。但是，对于部分文档集合并不十分庞大的情形，采用此类策略会大大地影响最终给用户的展现结果，并不可取。

4.1.4 其它的优化算法

Anh 和 Moffat 提出了一种检索过程中的剪枝策略[34]，Strohman 和 Croft 随

后对这一策略做了进一步的调整[35]，取得了非常明显的效果，在 TREC 2005 的数据集上，平均每秒完成的检索次数提升了 25 倍左右，同时对于内存的应用也有明显的改善。但是，他们的算法完全针对于 Moffat 在文献[36]中提出的一个整型化的关键词赋权模型。在这一模型中，关键词在文档中的得分被细化为 0 到 8 一共 9 个级别，对于同一个文档来说，分数越高则这个分数域内的关键词数量就越少，同时，在索引中也直接保存权值而不是 *tf* 这一统计信息。

因而，这一剪枝算法无法直接地应用到 *tf-idf* 和 *BM25* 这些主流的关键词赋权模型中。如果人为地对这些赋权模型进行整型化，一方面要对索引结构进行修改，另一方面整型化之后最终的检索结果也无法维持不变。

本文参考上述剪枝算法的思想，提出了一套新的预测剪枝策略，可以应用于 *tf-idf* 和 *BM25* 这些典型的赋权模型之中，在加快归并过程的同时，保证剪枝前后最终结果维持不变。

4.2 索引归并中的剪枝算法

4.2.1 剪枝算法流程

如图 4.4 所示，为了实现本文提出的剪枝算法，需要对 2 级索引的排序方式进行变化，不再以 tid 进行排序，而是以关键词在文档中的出现次数，也即 *tf* 进行排序。对于同一个文档来讲，由于它的 *idf* 相同，因而按照最终的关键词在文档中的得分进行排序。

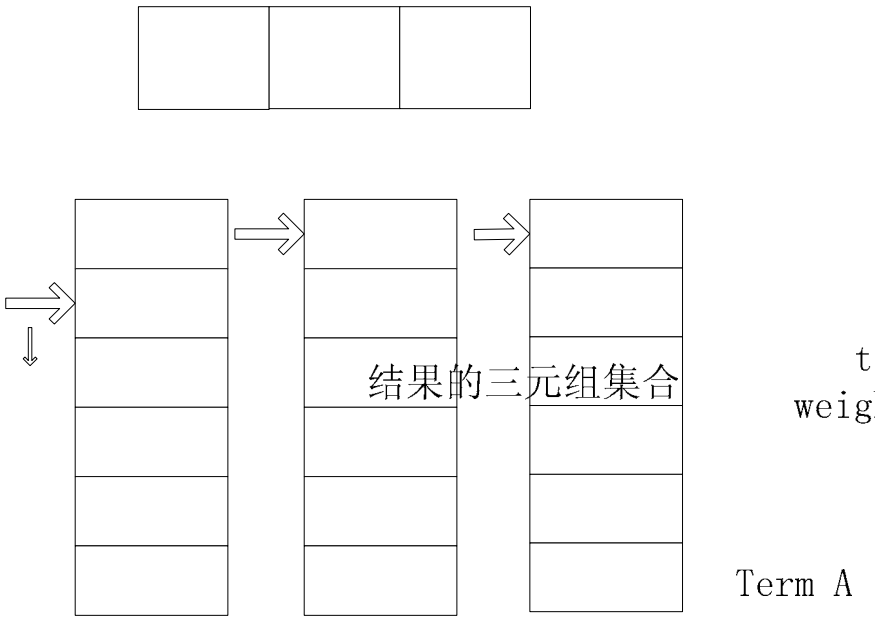

图 4.4 加入剪枝的倒排索引归并算法

在图 4.4 中的三元组可表示为 A_d <d, w, T >。其中，d 记录本三元组对应的那一个文档，w 表明目前的权值，集合 T 是当前 d 已经处理的关键词的集合。

具体的加入剪枝策略之后的检索算法如下：

1. 对查询进行分词。
2. 读入每个关键词的 2 级索引链。
3. 每个链上维护一个指针，指针所指的位置就是该链的当前位置。
4. 初始化，将所有指针指向该链的第一个位置，将 *AddMode* 置为真。
5. 计算链当前位置的每个关键词的 $d_k * q_k$ 得分，找出得分最高的那个关键词和它对应的文档的 tid，假设这个关键词是 A，tid 为 d。
6. 如果三元组结果集合 L 为空，则直接把三元组 A_d <$d, w_{q,t} \times w_{d,t}$,{A}>加入到 L 之中。
7. 如果三元组结果集合 L 不为空，判断 d 以前是否曾从 L 中被删除过，如果曾被删除过，则直接跳到步骤 5。
8. 如果 d 没有被删除过，判断 d 是否在 L 中。可以通过一个 bitmap，在 O(1)的时间内完成判断。
9. 如果 d 已经在 L 中，那么把权值 $w_{q,t} \times w_{d,t}$ 加入到已有的三元组的权值中，把文档 d 加入到三元组的关键词集合中。
10. 如果 d 不在 L 中，并且当前的 *AddMode* 是真，则直接把三元组 A_d

<$d, w_{q,t} \times w_{d,t}, \{A\}$>加入到 L 之中。

11. 调用 *ChangeAddMode* 函数判断是否需要修改 *AddMode* 的值。

12. 调用 *AccumTrim* 函数对 L 进行必要的剪枝。

13. 调用 *QuitJudge* 函数判断是否需要退出并跳到步骤 15。

14. 如果还有链没有归并完成，则回到步骤 5。

15. 对 L 按照权值排序，返回前 N 个结果。

与经典的链归并算法相比，剪枝功能主要体现在 *ChangeAddMode*，*AccumTrim* 和 *QuitJudge* 三个函数上，下面分别对它们的细节进行介绍。

4.2.2 *ChangeAddMode* 函数

Addmode 主要用来判断是否需要把新的文档加入到结果集合 L 中。如果可能进入前 N 的文档集合都已经在 L 之中，则可以将 *AddMode* 置为假，从而不再把新的文档加入结果集中。这一判断可以通过公式 4-1 来进行。

$$AddMode = (S_N \le p) \tag{4-1}$$

其中，S_N 表示 L 中第 N 大的权值，p 表明在最理想情况下，新的文档在剩余的归并过程中可能取得的最大权值。S_N 很容易得到，而 p 的计算遵从如下的公式：

$$p = \sum_{t \in q} p_t \tag{4-2}$$

其中，p_t 表明关键词 t 对应的链上当前指针下一个位置的文档 d 的 $w_{q,t} \times w_{d,t}$ 得分。$w_{q,t}$ 对于特定的关键词是保持不变的，$w_{d,t}$ 则与特定的文档 d 相关，随着归并的进行链向下移动，$w_{d,t}$ 会逐渐变小，因而 p_t 和 p 都是递减的，而 S_N 随着归并的进行是递增的趋势。

随着归并过程的进行，一旦不满足 $S_N \le p$，则说明即使新的文档得到了理论上的最高分，它的得分依然无法超过当前排名第 N 的文档的得分，显而易见，新的文档不需要加入到结果集合中了。

4.2.3 *AccumTrim* 函数

AccumTrim 函数的意义是去除已经在结果集合 L 中，但是最后却依然无法进入 Top N 的那些文档。判断 L 中的文档 d 是否需要被去除，可以用如下的公式：

$$S_d + p(d) < S_N \quad (4\text{-}3)$$

其中，S_d 是文档 d 的当前得分，S_N 的意义和公式(4-1)中的相同，$p(d)$ 表示在最理想情况下文档 d 在剩余的归并过程中可能获得的最大分数。$p(d)$ 的计算同样依赖于公式 4-2 中的 p_t，由于在 L 的三元组中已经保留了 d 对应的关键词集合 T，所以可以很容易地计算出 $p(d)$：

$$p(d) = \sum_{t \notin T} p_t \quad (4\text{-}4)$$

如果满足了公式 4-3，就说明文档 d 在接下来的归并过程中即使得到最大的得分，排名也无法进入 Top N，所以可以直接从集合中删除掉这个文档，并且以后也不再加入。

4.2.4 *QuitJudge* 函数

如果同时满足以下几个条件，则整个归并过程可以直接中止：

1. *AddMode* 是假，不再会有新的文档加入到 L 中。这同时表明所有可能进入 Top N 的结果都已经被加入到结果集合中了。
2. 在经过 *AccumTrim* 之后 L 中只剩下 N 个结果。这表明所有能够进入 Top N 的结果已经全部被找到了。
3. L 中剩余 N 个结果的顺序在剩余的过程中不会发生改变，这表明 Top N 的结果的顺序已经完全确定，可以直接中止归并过程，并把结果按顺序返回给用户。

在这三个条件中，第 1 和第 2 两个条件可以直接通过前述的两个函数进行判断。可以使用如下的函数来对第三个条件进行判断：

Function *CanQuit()*

For each document d_i in L:

If $S_{d_i} - S_{d_{i+1}} < p(d)$, return *false*

return *true*

这个函数表明，如果文档 d_{i+1} 在余下的归并过程中得到了最大的可能得分，可能超过排在它前面的文档 d_i，那么 Top N 个结果的顺序还有可能发生改变，归

并过程应该继续进行下去。

4.2.5 剪枝频度的选择

从上面的描述可以看出，剪枝根据索引是按照权值由高到低排序这一特点，通过对可能的最大得分进行预测，来避免不必要的计算过程。但是三个剪枝函数的判断本身也是需要消耗时间的，如果剪枝进行的频率过高，反而会影响最终的归并性能。

如果不对剪枝的频度进行限制，每一个 2 级索引结构体被选中时，剪枝就要进行一次。假设所有关键词的链长度总和是 l，那么在最坏情况下剪枝就将要进行 l 次。

为了降低剪枝的频率，可以每选出 n 个 2 级索引结构体时才进行一次剪枝。为此，本文定义了参数 a 对剪枝频度进行控制：

$$n = la \tag{4-5}$$

a 是一个 0 到 1 之间的正数，它的取值越大，表明剪枝进行得越频繁，剪枝的效果会越好，但是剪枝功能本身消耗得时间也会越多。本文将通过实验对 a 的选取进行探索。

4.2.6 剪枝算法在模糊检索系统中的应用

正如前面所分析的那样，“与”操作相比于“或”操作而言，算法复杂度要大大减小，特别是本文中“与”操作的结果并不是通过相似度得分而是通过 offset 得分来进行排序，所以本文提出的模糊检索系统在进行检索时，主要的时间消耗集中在经典的链归并之中。如果能够改进这一步骤的效率，对于本文提出的模糊检索系统的性能优化将很有价值。

4.3 实验

4.3.1 实验设置

在本章的实验中，所选取的网页文本来源于搜狗实验室提供的网页文本库，选取了其中 100 万个网页文本作为实验的文档集合，这些网页文本的平均长度大约是 1100 个汉字，整个文档集合在磁盘中的存储大小是 1.1 GB。

本次实验所选取的用户查询也来源于搜狗实验室提供的用户查询库，一共

选择了 10 万条实际的用户查询作为实验的查询集合。这些查询的平均长度是 7.6 个汉字，分词之后平均每个 qeury 含有 2.4 个汉字词（实验所用的分词系统是由哈尔滨理工大学信息检索实验室提供的开源的分词程序）。

实验在一台 8 GB 内存的 Linux 服务器上完成，为了排除磁盘 IO 的影响，所有的索引都全部载入到内存中。

实验采用经典的 *tf-idf* 模型作为关键词赋权模型，所采用的停用词列表参见附录 1。

4.3.2 实验结果和分析

由于剪枝算法的理论已经证明整个算法不会影响到最终的 Top *N* 结果，因而实验主要关注于在不同的 *N* 和 *a* 的取值下检索过程所消耗时间。

图 4.5 是实际的实验结果，其中纵坐标是 10 万条查询全部完成的时间（为了避免偶然性，10 万条查询一共完成了 3 次，取它们的平均值作为最终的结果）；横坐标是不同的 *a* 的取值，显然当 *a* 取值为 1 时，不进行剪枝。不同颜色和图案的曲线表示 *N* 不同取值情况下的时间消耗。

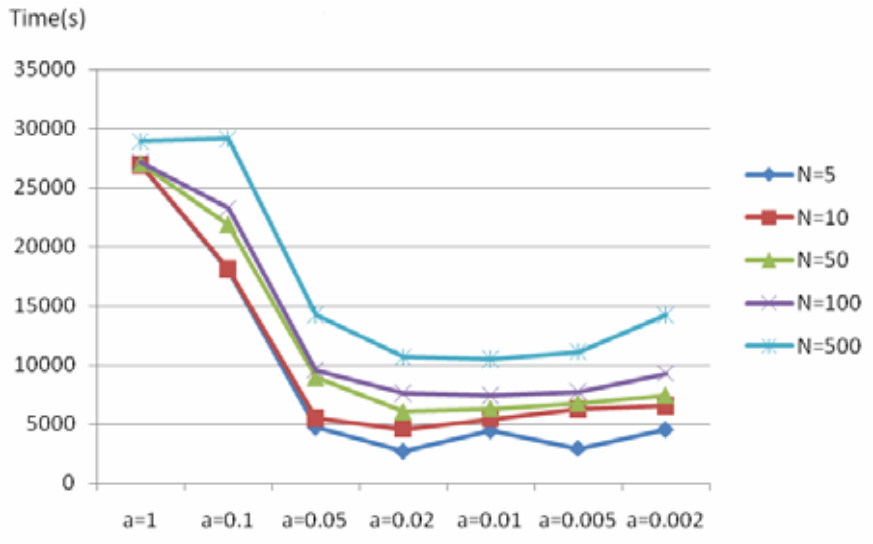

图 4.5 不同情况下的检索时间统计

从上面的实验结果可以得到以下结论：

1. 当 *N* 增长时，典型的不进行剪枝的归并过程时间消耗也发生增长，但增长并不明显。这是由于在 *N* 增大的情况下，归并过程中只有需要保留的

Top N 的结果增加了，归并的整体流程并没有发生大的改变。

2. 当 N 增长时，在不同的 a 参数下归并时间的增长是比较明显的。从剪枝算法可以看出，当 N 的值越小，剪枝的力度就越大，效果也就越明显。在极端情况下，如果 N 的值接近于链的总长度 l，那么剪枝几乎不会取得任何效果。
3. 剪枝频度参数 a 对于剪枝的效果影响非常明显。正如在 4.2.5 中所讨论的那样，a 太大会削弱剪枝的力度，a 太小则消耗在剪枝本身的时间将会大大增加。
4. 合适的 a 的选取对于归并过程的速度提升是非常明显的，能使速度加快 6 倍左右，而且保证对于经典模型下的检索结果不造成影响。

4.4 本章结论

本章主要针对检索模型的实现进行研究。针对现有索引归并的优化算法通常会影响最终结果这一问题，提出了新的基于得分预测的剪枝算法。实验证明，剪枝算法在不影响最终模型检索效果的情况下，能够大大加快索引归并的速度。但是，剪枝算法实际应用时需要对索引结构进行修改，因而，检索系统的索引程序也要为之发生改变。

第5章 结论和展望

本文研究基于拼音相似度的汉语模糊检索方法，对基于拼音的相似度度量方式、加入拼音扩展后的检索模型的构建以及检索过程的性能优化分别进行了探索，并通过在实际的网页文本和查询集合上的实验证明了所提出的检索模型和性能优化方法是有效的。

对于目前的信息检索系统而言，从结构体系设计，到相似度计算模型以及用户的个性化搜索需求，都面临着严峻的挑战，拼音输入造成的错误查询只是众多问题中的一个。不论是从商业利益还是从国家发展战略来讲，投入更多的人力物力对中文信息检索系统进行更加广泛和深入的研究，都是十分有价值的。希望本文能够在这一方面作出一点小小的贡献。

在本文的研究成果基础上，可进一步研究的内容包括：

- 结合文档集合中的词汇统计信息来辅助进行查询扩展，从而既减少查询扩展的次数，又能对扩展后的结果质量有进一步的保证
- 对拼音扩展的检索结果与原结果的结合做更深入的研究，尽量在少影响正确查询的情况下，对拼音错误查询进行良好的处理。
- 进一步研究剪枝频度参数 a 的取值，即如何从理论上计算选取与检索模型最适合的剪枝频度。

由于本文主要针对用户输入中的拼音错误进行研究和处理，所以，对于其它类型的错误如五笔错误、手写输入错误、选词不正确等无法进行良好的处理。另一方面，本文的检索系统在进行实际处理时只考虑了文档内容一个域，而没有对文档的标题、anchor 域等进行加权计算，同时，对于不同类型的文档之间的差别也没有深入地衡量，这些都是在实际运用中需要特殊考虑的地方。在实际的应用中，本文提出的模糊检索方法对于短信检索、语音检索这类重查代价太多或者拼音错误额外增多的应用环境能够取得比较好的效果。

本文提出的剪枝算法针对于向量空间模型中的全“或”归并操作，由于主流的商业搜索引擎都普遍牺牲相关性采用了“与”操作代替“或”操作，因而，本文的剪枝算法并不太适用于这类情况。

参考文献

[1] R Baeza-Yates , B Ribeiro-Neto. . Modern Information Retrieve . ACM press, 1999.

[2] Kornstadt, A., Themefinder: A Web-based Melodic Search Tool, in Melodic Similarity Concepts, Procedures, and Applications W. Hewlett and E. Selfridge-Field, Editors.1998, MIT Press: Cambridge.

[3] Manning C, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge University Press. 2008.

[4] Mitra M, Singhal A, Buckley C. Improving Automatic query Expansion. Proc of the 21 st Ann Int ACM-SIGIR Conference on Reser and Dev in Info Retrieval, 1998.

[5] G. NAVARRO A guided tour to approximate string matching. ACM Computing Surveys, 2001 , 33(1) : 31288.

[6] Tarhio J, Ukkonen E. Approximate Boyer-Moore string matching. *SIAM J on Computing*, , 1993, **22**(2): 243-260.

[7] Knuth D, Morris J, Pratt V. Fast pattern matching in strings [J]. *SIAM J on Computing*, 1977, **6**(2): 323-350

[8] 王静帆，邬晓钧，夏云庆等 中文信息检索系统的模糊匹配算法研究和实现. 中文信息学报 2007，21(6): 59-64

[9] 陈儒. 面向短信过滤的中文信息模糊匹配技术. 哈尔滨: 哈尔滨工业大学信息检索实验室 2003.

[10] Wu. S , Manber. U. Fast text searching allowing errors. Commun. ACM 35 , 1992 ,10 , 83291.

[11] R. Boyer , J. Moore. A fast string searching algorithm. Comm. ACM 1977 20 (10) :7622772.

[12] Horspool N. Practical Fast Searching in Strings. Software Practice and Experience ,1980 , 10.

[13] E. Ukkonen Algorithms for approximate string matching. Information and Control. 1985a. 64 , 1002118. Preliminary version in Proceedings of the International Conference Foundations of Computation Theory(LNCS , vol. 158 ,1983) .

[14] E. Ukkonen , Finding approximate patterns in strings. Algor. 1985b ,6 1322137.

[15] H Hyyro , G Navarro. Faster bit-arallel approximate string matching. Proc. 13th Combinatorial Pattern Matching (CPM’2002) , LNCS , 2002 2 Springer ,23273.

[16] A. Aho, D. Hirschberg, J. Ullman. Bounds on the Complexity of the Longest Common Subsequence Problem. J.ACM 23, 1, 1～12(1976)

[17] 黄永光，刘挺，车万翔 等. 面向变异短文本的快速聚类算法. 全国网络与信息安全技术

研讨会[C]. 北京，2005.

[18] Araujo M, Navarro G, Ziviani N. Large text searching allowing errors. Proc WSP97. Valparaiso, Chile: Carleton University Press. 1997, 8: 2-20.

[19] Baeza-Yates R, Navarro G. Block-addressing indices for approximate text retrieval. *J Am Soc Info Sci*, 2000, **51**(1): 69–82.

[20] Cuadra, C. A. and R. V. Katter. Experimental Studies of Relevance Judgments: Final Report. I: Project Summary (NSF Report No. TM-3520/001/00). Santa Monica, CA: System Development Corporation. 1967.

[21] Saracevic, T. The concept of 'relevance' in information science; a historical review. In T. Saracevic ed. Introduction to Information Science. New York: R. R. Bowker, pp.111-151. 1970.

[22] Saracevic, T. Relevance: a review of and a framework for the thinking on the notion in information science. Journal of American Society for Information Science 26(6): 321-343. 1975.

[23] MacMullin, S. E., and R. S. Taylor. "Problem dimensions and information traits." The Information Society 3(1): 91-111. 1984.

[24] Taylor, R. S. 1986. Value-Added Processes in Information Systems. Norwood, NJ: Ablex.

[25] Belkin, N. J., R. N. Oddy and H. M. Brooks. 1982. "ASK for information retrieval." Journal of Documentation 38(2): 61-71 and 38(3): 145-164.

[26] Dervin, B. 1983. An Overview of Sense-Making Research: Concepts, Methods and Results to Date. Paper presented at the International Communication Association Annual Meeting, Dallas, TX.

[27] Saracevic, T. Relevance reconsidered '96. In P. Ingwersen and N. O. Pors. Information Science: Integration in Perspective. Copenhagen: Royal School of Library and Information Science. 1996.

[28] Mizzaro, S. "How many relevances in information retrieval?" Interacting with Computers 10:305-322. 1998.

[29] Cosijn, E. and P. Ingwersen. Dimensions of relevance". Information Processing and Management 36: 533-550. 2000.

[30] Salon G.The SMART Retrieval System-Experments in Automatic Document Processing. Prentice Hall Inc,Englewood Cliffs,NJ,1971.

[31] J. Ponte and W.Croft. A language modeling approach to information retrieval.In SIGIR 1998.

[32] 孙建军，成颖，等. 信息检索技术[M].北京:科学出版社，2004.

[33] Makoto Iwayama,Atsushi Fujii，NorikoK ando,Yuzo Marukawa. An Empirical Study ox Retrieval Models for Different Document Genres:Patents and Newspaper Article.SIGIR' 03:251-258. Toronto, Canada 2003

[34] Anh V. N., Moffat A.: Pruned query evaluation using pre-computed impacts. In: SIGIR 2006, pp. 372–379. ACM Press, New York (2006)

[35] Strohman T., Croft W. B.: Efficient document retrieval in main memory. In: 30th Ann. Int.ACM 2007, pp. 175–182. ACM Press, Amsterdam (2007)

[36] Anh V. N., Moffat A.: Simplified similarity scoring using term ranks. In: SIGIR 2005, pp. 226–233. ACM Press, New York (2005)

致　谢

衷心感谢我的导师郑方教授对本人的精心指导。他敏锐的洞察力、渊博的知识、严谨的治学态度、精益求精的工作作风和对科学的献身精神给我留下了刻骨铭心的印象，这些使我受益匪浅，并将成为我终身献身科学和献身事业的动力。

也十分感谢邬晓钧老师对我学术上的指导，他的帮助是我硕士期间研究工作顺利进行的基础。

在香港中文大学电子工程系进行两个月的合作研究期间，承蒙李丹教授学术上细致指导与生活中的热心帮助，不胜感激。

同时也感谢清华大学信息技术研究院技术创新和开发部语音和语言技术中心全体老师和同学们的热情帮助和支持。

最后，衷心的感谢我的父母和其他亲朋好友对我的关心、支持和理解，没有他们对我的关心、鼓励和支持，我无法完成现在的硕士学业。

本课题在进行过程中大量使用搜狗实验室免费提供的数据，在此予以致谢。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：　　日　期：

附录 A 本文的检索系统采用的停用词表

说明：停用词后的数字表示停用词级别，0 级停用词无条件去除，1 级停用词则在去除停用词后关键词数量较少时，需要重新召回。停用词词表中最后两个停用词分别是全角和半角的空格。

啊 1
阿 1
吧 1
的 1
啦 1
吗 1
呢 1
是 1
呀 1
了 1
么 1
《 0
》 0
/ 0
， 0
。 0
* 0
！ 0
？ 0
、 0
～ 0
： 0
“ 0
” 0
‘ 0
’ 0
〔 0
(0
) 0
< 0
> 0
{ 0
} 0
（ 0
） 0
； 0
• 0
, 0
. 0
" 0
? 0
! 0
' 0
; 0
: 0
(0
) 0
` 0

~ 0 0

0

个人简历、在学期间发表的学术论文与研究成果

个人简历

1984 年 11 月 3 日出生于湖北省广水市。

2002 年 9 月考入清华大学计算机科学与技术系学习，2006 年 7 月本科毕业并获得工学学士学位。

2006 年 9 月免试进入清华大学计算机科学与技术系攻读硕士学位至今。

发表的学术论文

[1] Jiang CAO, Xiaojun WU, Yu Ting YEUNG, Tan LEE, Thomas Fang ZHENG. Automatic Collecting of Text Data for Cantonese Language Modeling. O-COCOSDA. Kyoto, Japan. 2008.

[2] 曹骅，邬晓钧，夏云庆，郑方. 基于拼音索引的中文模糊匹配算法. 清华大学学报（自然科学版）. 2009 年第 49 卷第 S1 期. 2009 年 (EI)

[3] 曹骅，邬晓钧，夏云庆，郑方. 基于索引过滤的汉语短文本模糊匹配计算方法. NCMSCC2009. 中国新疆. 2009.

[4] Jiang CAO, Xiaojun WU, Yunqing Xia, Thomas Fang ZHENG. A Pruning Algorithm for Index Merging in Common Relevance Models. AIRS2009.（EI，**审稿中**）

[5] 张利鹏, 曹骅, 徐明星, 郑方. 防止假冒者闯入说话人识别系统. 清华大学学报(自然科学版). 2008 年 S1 期.2008 年.(EI)