分类号_____ 密级_____
UDC _____ 编号_____

中国科学院研究生院

硕士学位论文

面向信息检索的语义计算技术

金千里

指导教师 徐波 研究员 赵军 副研究员
中国科学院自动化研究所
申请学位级别 工学硕士 学科专业名称 模式识别与智能系统
论文提交日期 2004.5 论文答辩日期 2004.6
培养单位 中国科学院自动化研究所
学位授予单位 中国科学院研究生院

答辩委员会主席_____

Semantic Computing in Information Retrieval

Dissertation Submitted to

Institute of Automation, Chinese Academy of Sciences

in partial fulfillment of the requirements

for the degree of

Master of Engineering

By

Qianli Jin

(Pattern Recognition and Intelligence System)

Dissertation Supervisor: Professor Bo Xu & Jun Zhao

独创性声明

本人声明所成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。尽我所知，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地说明并表示了谢意。

签名：_____导师签名：_____ 日　期：_____

关于论文使用授权的说明

本人完全了解中国科学院自动化研究所有关保留、使用学位论文的规定，即：中国科学院自动化研究所有权保留送交论文的复印件，允许论文被查阅和借阅；可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签名：_____导师签名：_____ 日　期：_____

注：本文的工作受国家自然科学基金 60372016 和 60272041 资助。

摘 要

信息检索，包括信息的组织、呈现、查询、存取等各个方面，为人们提供了快速、精确地获取所需信息的方式。信息检索通常是指文本检索，其核心是根据用户查询找到相关文本，包括“标引”和“相似度计算”两个关键技术。随着信息社会尤其是互联网的发展，人们对检索的要求越来越高。传统的基于关键词匹配的检索技术，往往存在查不全、查不准、检索质量不高的问题。因此，智能检索研究已经成为热点，并将是支撑下一代互联网的核心技术之一。

由于文本大多数是用非形式化的自然语言表述，因此实现智能检索的关键就是要在一定程度上理解自然语言，挖掘出隐藏在文本背后的“语义”。从研究现状来看，基于词汇的语义模型是一类比较理想的浅层语义表述方式，已经有了很多成功的实践。因此，在信息检索中引入智能技术的一种方案，就是在“标引”和“相似度计算”两个关键技术中引入词汇语义模型，用浅层语义来指导检索过程，提高检索的准确率。这正是文本的选题思路和工作重点。

本文首先简要介绍信息检索和语义模型的研究现状，说明两者结合的必要性和合理性。然后，论述三类语义模型（隐含语义标引、语义树、语义张量）在信息检索中的应用。最后，介绍模式识别国家重点实验室（NLPR）的信息检索系统框架、模块和实现；并利用 TREC 评测来测试系统的功能和性能。概括地说，本文主要有如下一些工作。

（1） 论述了语义模型与信息检索中两个关键技术（“标引”和“相似度计算”）的结合问题；
（2） 改进了隐含语义标引模型，提出弱指导的统计隐含语义标引模型，使语义空间分布更合理，效率也更高。这个模型可以小规模地应用于“查询主题词构造”技术；
（3） 提出了基于语义树的语义空间模型。语义空间不再是静态的，而是实时构建的，其灵活性和可操作性优于各种隐含语义标引模型。尤其在查询主题词扩展技术方面，性能超过了常见的扩展算法；
（4） 提出了语义张量的概念，并明确了其物理意义，归纳为两个核心思想。进一步，用窗口系列模型来表述这两个思想，并应用于查询和文本间的相似度计算。实验证明，这类模型比传统的矢量模型更有效；
（5） 构建了 NLPR 检索系统框架，并完成了模块设计和编程的工作。除了标引和相似度计算等与检索技术相关的模块外，还包含了汉语分词、英文词形还原等语言处理技术；
（6） 通过参加 2003 年的 TREC 评测（Robust Track 和 Novelty Track），测试了检索系统的功能和性能，并积累了一些文本检索的经验。其中，在 Novelty 检索任务中获得了较出色的成绩。

关键词： 信息检索 文本检索 语义模型 TREC 评测

Abstract

Information Retrieval, including information orgnizing, representation, inquiry, access, etc, supplies a series of technologies for obtaining information rapidly and accurately. Information retrieval systems, usually focusing on text retrieval, find relevant documents based on users'queries. There are two key technologies involved, "Indexing" and "Similarity Computing". The traditional retrieval methods based on the keyword matching often result in low precisions. With the development of information society and World Wide Web, the traditional retrieval methods can no longer satisfy users' requirement. Nowadays, intelligent retrieval has already become a hot-spot of research and will be a key technology in the next generation of World Wide Web.

In most cases, the contents of the text are represented by nature language. The key challenge of intelligent retrieval is the nature language understanding, which means to find out the meaning behide the text. We believe that semantic models based on words are suitable for representing the shallow meaning of the text. Therefore, we use word-based semantic models to supervise the process of information retrieval, in order to get improved retrieval performance.

Firstly, the thesis gives a brief introduction of the background of information retrieval and semantic models. Then, three kinds of semantic models (Latent Semantic Indexing series, Semantic Tree and Semantic Tensor) are proposed and evaluated in the field of information retrieval. After that, we present the NLPR IR System, including the architecture, module definitions and implementation. TREC evalutions are used to evaluate the system. In summary, the contributions of the thesis are as follows.

(1) It explains how to use word-based semantic models to supervise the process of information retrieval;

(2) Based on the former LSI and PLSI, weekly-supervised probabilistic latent semantic indexing (SPLSI) is presented and evaluated, which can get more reasonable semantic space and can be used in the process of indexing;

(3) Semantic Tree Model (STM) is developed to create a dynamic, flexible, controllable and real-time semantic space. As a new technology of indexing, STM outperforms most of the existing methods;

(4) Semantic Tensor is put forward as a new theory, which is expressed by two key notions. Three Window-based Models of this theory are developed to compute the similarities between documents and queries. The experiments show that they outperform the traditional word-based vector space models;

(5) We build NLPR IR System, including architecture designing, module definitions and implementation;

(6) We participate in 2003 TREC Evaluation (Robust Track and Novelty Track) in order to test NLPR IR system and get excellent results in Novelty Track.

Key Words： Information Retrieval, Text Retrieval, Semantic Model, Semantic Computing, TREC Evaluation

摘 要 ..... **I**

ABSTRACT ..... **II**

第一章 绪论 **1**

1.1 信息检索 1
1.1.1 信息检索简介 *1*
1.1.2 信息检索的分类 *1*
1.1.3 信息检索的两个关键技术 *2*
1.1.4 研究现状与热点 *3*
1.2 词汇语义模型 4
1.2.1 自然语言处理中的语义表述 *4*
1.2.2 两类常见的词汇语义模型 *5*
1.2.3 词汇语义模型在信息检索中的应用 *6*
1.3 论文的主要工作 6

第二章 基于隐含语义系列模型的信息检索 **8**

2.1 隐含语义标引模型（LSI-SVD） 8
2.2 统计隐含语义标引模型（PLSI-EM） 12
2.3 弱指导的统计隐含语义标引模型（SPLSI-EM） 14
2.4 面向信息检索的应用 18
2.5 实验评估 19
2.6 应用系统（GOOGLE-SHELL） 23
2.7 本章小结 27

第三章 基于语义树模型的信息检索 **29**

3.1 隐含语义标引系列模型的不足 29
3.2 从语义网到语义树 29
3.3 语义树模型 30
3.4 基于语义树模型的主题词扩展 33
3.5 实验评估 35
3.6 本章小结 38

第四章 基于语义张量模型的信息检索 **39**

4.1 信息检索中的矢量和张量 39
4.2 张量概念的模型化 41
4.3 张量模型的数学表述 – 窗口系列算法 42
4.3.1 数学模型一： 简单窗口模型 （Simple Window-based Model） *42*
4.3.2 数学模型二： 动态窗口模型 （Dynamic Window-based Model） *44*
4.3.3 数学模型三： 核心窗口模型 （Core Window-based Model） *47*
4.4 实验评估 49
4.4.1 总体准确率和召回率 *49*
4.4.2 窗口宽度分析 *51*
4.4.3 动态窗口模型参数分析 *51*
4.4.4 核心窗口模型参数分析 *52*

4.5 本章小结 52

第五章 信息检索系统与评测 53

5.1 NLPR 信息检索系统 53

5.1.1 NLPR 信息检索系统的框架 53

5.1.2 NLPR 信息检索系统的设计 54

5.1.3 NLPR 信息检索系统的具体实现 61

5.2 信息检索的基本评测指标 68

5.3 NLPR 信息检索系统在 TREC 评测中的表现 70

5.3.1 TREC 评测介绍 70

5.3.2 TREC-2003 中的鲁棒性全文检索评测 71

5.3.3 TREC-2003 中的新信息检索评测 72

5.3.4 TREC-2003 任务小结 76

第六章 总结与展望 77

6.1 基于语义模型的信息检索总结 77

6.2 基于语义模型的信息检索展望 77

参考文献 80

攻读硕士学位期间发表的论文 84

致 谢 85

第一章 绪论

1.1 信息检索

1.1.1 信息检索简介

信息检索起源于图书馆的参考咨询和文摘索引工作，从 19 世纪下半叶开始发展，至 20 世纪 40 年代，索引和检索已成为图书馆独立的工具和用户服务项目。随着 1946 年世界上第一台电子计算机问世，计算机技术逐步走进信息检索领域，并与信息检索理论紧密结合起来；脱机批量情报检索系统、联机实时情报检索系统相继研制成功并商业化，20 世纪 60 年代到 80 年代，在信息处理技术、通讯技术、计算机和数据库技术的推动下，信息检索在教育、军事和商业等各领域高速发展，得到了广泛的应用。

目前，广义信息检索技术的应用范围已经大大扩展了，除了传统的文本检索以外，还包含图片、音频、视频等多媒体信息检索。但是，文本所涉及到的内容，只是狭义的信息检索，也称为文本检索。

文本检索，包括信息的存储、组织、表现、查询、存取等各个方面，其核心为文本的索引和检索。从历史上看，信息检索经历了手工检索、计算机检索到目前网络化、智能化检索等多个发展阶段。目前，信息检索已经发展到网络化和智能化的阶段。信息检索的对象从相对封闭、稳定一致、由独立数据库集中管理的信息内容扩展到开放、动态、更新快、分布广泛、管理松散的 Web 内容；信息检索的用户也由原来的情报专业人员扩展到包括商务人员、管理人员、教师学生、各专业人士等在内的普通大众，他们对信息检索从结果到方式都提出了更高、更多样化的要求。适应网络化、智能化以及个性化的需要是目前信息检索技术发展的新趋势。

1.1.2 信息检索的分类

文本信息检索按照其内容和面向的应用来分，可以有以下一些分类方法：

(1) 按照数据库的不同，可以分为基于内部文本库的检索和基于互联网等开放数据库的检索。前者往往要求比较精确的匹配信息；而后者由于开放和动态的特性，存在大量数据模糊和冗余，因此仅要求保证一定的准确率即可。后者最热门的技术就是搜索引擎；两者之间的差别体现在以下五个方面。

①数据量：传统信息检索系统一般索引库规模多在 GB 级，但互联网网页搜索需要处理几千万甚至上亿的网页，搜索引擎的基本策略都是采用检索服务器群集，对大多数企业应用是不合适和不必要的，并不适用于企业应

用。②内容相关性：信息太多，查准和排序就特别重要，Google 等搜索引擎发展了网页链接分析技术，根据互联网上网页被连接次数作为重要性评判的依据。但企业网站内部的网页链接由网站内容采编发布系统决定，其链接次数存在偶然因素，不能作为判别重要性的依据。真正的企业应用的检索要求基于内容的相关性排序，就是说，和检索要求最相关的信息排在检索结果的前面，链接分析技术此种排序基本不起作用。③实时性：搜索引擎的索引生成和检索服务是分开的，周期性更新数据，大的搜索引擎的更新周期需要以周乃至月度量；而企业信息检索需要实时反映内外信息变化，搜索引擎系统机制并不能适应企业中动态性数据增长和修改的要求。④安全性：互联网搜索引擎都基于文件系统，但企业应用中内容一般均会安全和集中地存放在数据仓库中以保证数据安全和管理的要求。⑤个性化和智能化：由于搜索引擎数据和客户规模的限制，相关反馈、知识检索、知识挖掘等计算密集的智能技术很难应用，而专门针对企业的信息检索应用能在智能化和个性化方面走得更远。

(2) 根据所涉及到的语言不同，可以分为单语言的检索和跨语言的检索。通常，用户的查询和目标文本都是用同一种语言书写的；但是随着全球信息化的不断推广，因特网资源不再集中在英语等少数几种语言上，非英语因特网资源的比重不断增加；另一方面，非英语英特网用户的比重也在不断增加。根据预测，到 2005 年，英特网用户总量将达到 3 亿 4 千 5 百万，其中非英语用户将占到 78%。因特网资源的多语言性和因特网用户的多语言性，使得多语言文本检索变得迫在眉睫。例如：一个英汉日多语言信息检索系统中，用户用英文提问，系统除了反馈给用户英文的相关文本外，还可以反馈给用户中文和日文的文本。因为用户提问和文本可能是由不同语言表示的，语言之间的差异性给检索过程带来很多困难[4]。

(3) 根据检索的信息粒度不同，可以分为文本检索、段落检索、句子检索、词序列检索等等。颗粒度越小，信息定位越精确，技术难度也越大。对于文本来说，由于所包含的词比较多，因此可以利用的信息也比较丰富，检索起来相对容易。而句子一级的检索就困难很多，一方面是词比较少，另一方面简称、指代等现象可能在句子中大量出现。因此，在小颗粒度的检索中，主题词扩展和指代销解（Co-reference）就更加重要。

由于信息量的急速膨胀，对用户来说，要求的信息精度越来越高。对用户的一个查询，仅仅返回一堆文本显然是不能令人满意的，因为用户还是需要花费大量时间去浏览这些文本。所以，精确的信息定位越来越受到重视，这方面的研究也逐步成为一个热点，如各种问答系统。

1.1.3 信息检索的两个关键技术

文本信息检索过程是这样的：

用户希望看到关于某个话题(Topic)的一些文本，首先用一个提问(Query)对这个话题进行描述，检索系统从这个提问中衍生出标引条目；将这些标引条目与

文本库中每个文本的标引条目（文本库中的每个文本事先已经进行了类似的标引）进行匹配。然后，系统将匹配程度最好的文本序列返回用户。

从中可以看出检索系统的两个关键技术是：

（1） **标引**：包括提问标引和文本标引。用户给出的查询和文本库中的文本都是以自然语言方式表述的。标引的过程，就是把这两者用统一的数学模型表示出来。常用的方式是，将查询和文本都表示为词向量。为了使得构造的词向量能尽可能准确地反映用户查询或者文本原来的含义，发展出了一系列算法模型。如各种主题词扩展模型，相关反馈技术等等。

（2） **相似度计算**：通过计算用户查询标引和文本标引之间的距离，估计文本和查询之间的相似性，进而给出检索结果。这是任何一个检索系统的核心。如果前面采用了词向量的标引方式，那么在这部分就多采用矢量内积的方法来计算相似度，如 tf-idf[7][8]及其各种变形[12][27]。如果是互联网检索，还可以利用 HTML/XML 文件的重要字段信息，以及网页之间的链接信息来计算相似度。

信息检索系统的难点在于：由于查询和文本一般以词为单位来进行标引，词的一词多义问题(Polysemy)和一义多词问题(Synonymy)往往会严重影响系统效率。此外，通常采用的矢量内积方法，在某些情况下是不合理的[13]。

1.1.4 研究现状与热点

文本信息检索的研究主要可以分为两个大方向。

（1） 第一个是提高检索系统的时间效率，采用分布式多线程技术，在尽可能短的时间内，获得结果集；由于信息数量的剧增，尤其是面向互联网应用时，系统响应时间在某种程度上说是最重要的一个指标。

（2） 第二个是提高检索的准确率，其核心内容就是上一小节提到的两个关键技术。这部分是本文关注的重点。

在“标引”技术领域，除了各种传统的主题词扩展方法（如：基于 WordNet[11]的扩展，相关反馈[16]等等），也发展出一些现在比较流行的基于语义概念模型的方法（如基于 LSI 的扩展[1][2]、概念网络模型[14]等等）。但是，大多数基于语义的方法，都有一个相同的弱点，就是很难在真实系统中大规模应用。一方面是因为这种语义概念体系，无论是规则的还是统计的，都很难在非特定领域建立。另一方面是，把查询或者文本准确地映射到概念体系中，也是个很困难的课题。因此，“标引”技术在目前的情况下，多数系统还是采用传统的词向量模型，附带一些成熟的扩展方案。

在“相似度计算”技术领域，占主导地位的有两类模型。一类是 tf-idf

家族，包括 tf[7],tf-idf[8]、长度归一化模型[12]、BM25[27]等等。这类模型建立在词向量标引的基础上，有上百种不同的表述方式（或者说公式）。由于 tf-idf 系列模型简单，性能也不错，因此一直是最流行的。另一类是专门面向网络应用的模型，Google 所提出的 PageRank。简单地说就是，一个网页的重要性，决定于其他网页链接到它的次数。这是非常合理而有效的。除了这两类以外，其它还包括对一些格式文本（如 HTML,XML）的特殊字段，赋予不同权值等。

目前，比较成熟检索系统，多采用简单的词向量模型和 PageRank 模型，很少有涉及到语义层面的模型。这一方面是因为各种语义模型，虽然研究的很热，但极少有真正能大规模应用的。另一方面，由于商业检索系统对性能的苛刻要求，使得语义检索、个性化检索等知识密集型的技术很难应用。

随着信息社会的发展，人们对检索的要求越来越高。传统的全文检索技术基于关键词匹配进行检索，往往存在查不全、查不准、检索质量不高的现象，特别是在网络信息时代，已经越来越难满足人们检索的要求。因此，智能检索的研究已经成为热点，并且在可以预见的将来，能够应用于真实的系统。普遍认同的一个观点是，智能检索技术将是支撑下一代互联网的关键技术。

智能检索利用分词词典、同义词典，同音词典改善检索效果，比如用户查询“计算机”，与“电脑”相关的信息也能检索出来；进一步还可在知识层面或者说概念层面上辅助查询，通过主题词典、上下位词典、相关同级词典，形成一个知识体系或概念网络，给予用户智能知识提示，最终帮助用户获得最佳的检索效果，比如用户可以进一步缩小查询范围至“微机”、“服务器”或扩大查询至“信息技术”或查询相关的“电子技术”、“软件”、“计算机应用”等范畴。另外，智能检索还包括歧义信息和检索处理，如“苹果”，究竟是指水果还是电脑品牌，“华人”与“中华人民共和国”的区分，将通过歧义知识描述库、全文索引、用户检索上下文分析以及用户相关性反馈等技术结合处理，高效、准确地反馈给用户最需要的信息。

1.2 词汇语义模型

1.2.1 自然语言处理中的语义表述

用自然语言与计算机进行交流，获取合适的信息，得到满意的服务，是人们长期以来所追求的。自然语言理解是计算机科学领域与人工智能领域中的一个重要方向，它研究能实现人与计算机之间用自然语言进行有效交流的各种理论和方法，既重要的实际应用价值，也有重要的学术意义。人们可以用自己习惯的语言来使用计算机，而无需再花大量的时间和精力去学习不自然、不习惯的计算机语言；人们也可进一步了解人类的语言能力和智能机制。

目前的自然语言处理，在应用层面，基本还处于“字词处理”阶段。也就是，以字和词作为基本信息单元，利用一些简单规则或者统计信息对语言作一些粗浅的

分析。这和人们理想中的“理解自然语言”，还有比较大的差距。要做到理解，就要从非形式化的自然语言中，挖掘出隐藏在背后的“语义”，也就是人们真正想表达的含义。在这方面，学者们提出了很多句法理论和语义理论，试图用各种形式化方法去表述语义，获得了很多进展，但和实际应用还有很大的距离。自然语言处理中的语义表述，在大尺度上可以分为以下三类：

(1) 第一类方法在句法分析的基础上研究语义问题。这类方法从形式语法入手，首先进行短语或者句子的语法分析，然后再进行语义分析。比较典型句法理论有短语结构语法、依存语法、配价语法等，北京大学朱德熙教授、陆俭明教授、俞士汶教授在这方面做了很多有价值的工作[46]。
(2) 第二类方法从人的角度出发，认为思维的机制并不是先去分析语法，然后再分析语义，而是一种概念网络或者逻辑框架。对语义的理解，并不一定要建立在语法分析正确的基础上。这类方法比较典型的是，陆汝占教授提出的基于内涵模型的语义分析[44]和黄曾阳教授提出的概念层次网络理论（HNC）[45]。
(3) 第三类方法从词分析的角度入手，认为自然语言处理应该走词汇化的道路，黄昌宁教授曾多次表述过这个思想[43]。这类思想认为，很难有简单而普适的语法、语义规则，针对不同的词应该有不同的处理方式。在计算技术和计算能力越来越强的今天，这类方法逐步地为人们所关注。要从这条路实现对语义的理解，有个先决条件，就是要有某种词汇语义模型；对每个词的不同特性，以及词与词之间的概念关系有比较明确的定义。这方面已经有了一些比较成熟的方法，如 WordNet[11]、HowNet[5]、各种词的统计模型等等。现在比较热门的关于 Ontology 的研究，在某种程度上，也可以看作一种词汇语义模型。

词汇语义模型是把语义引入自然语言处理的一种简单可行的方式。因为，对一种语言来说，词汇量在短期内变化不大，有丰富的词典资源可以利用。本文的工作选择了词汇语义模型作为自然语言的语义表述，面向信息检索的应用，利用浅层信息对语义作一些初步分析。在下一节将简单介绍两类常见的词汇语义模型。

1.2.2 两类常见的词汇语义模型

词汇语义模型是以词为基本单位，描述词的特性以及词与词之间关系的一种体系。对目前已经较为成熟的两类词汇语义模型，简单描述如下：

(1) 第一类是人工编撰的语义词典。比较典型的是英文的 WordNet[11]和中文的 HowNet[5][6]。以 HowNet 为例，为反映每个词的特性，有“词类标记”、“义元标记”、“动态角色和属性标记”等，其中既有语法信息也有语义信息。此外，依据不同词的相似特性，构成了语义场，可以从各个层次了解词与词之间的关系。这种关系既可以是同义、反义、上下位关系，也可以用量化的相似度值来表达。
(2) 第二类是基于统计的词汇语义模型。如语义图模型[9][10]、隐含语义分析模型[1]、统计隐含语义分析模型[2]、概念层次网络模型[14]等等。这类模型，都是从大规模语料库中统计词的概率分布，采用各种聚类方法，

获得基于词的语义空间模型。在这类模型中，都可以通过计算两个词在语义空间中的距离，给出两者之间的相似度值。

1.2.3 词汇语义模型在信息检索中的应用

词汇语义模型能够描述词的特性，以及词与词之间的关系。而目前信息检索的常用技术都是基于词的，如关键词标引和相似度计算。因此，词汇语义模型在信息检索的应用中有很大的用武之地。具体的，对信息检索的两个关键技术，都有很大帮助。

(1) 标引技术：通常，标引技术是用一组关键词来表达一个查询或者一篇文本。引入词汇语义模型后，就能在语义层面上考察这组关键词是否可以合理而有效地表达原始文本的含义。
(2) 相似度计算：通常，相似度计算是计算查询词向量和文本词向量之间的接近程度，实际上每个词被赋予了一个独立的含义，词与词之间只有二元关系，要么是相同的，要么是不同的。自然语言中一个语义可以有很多种表述方式，因此这种二元关系显然是不够合理的。在引入了词汇语义模型后，词与词之间可以有一个相似度值来衡量两者的语义接近程度，能更好的反映真实情况。

1.3 论文的主要工作

在本文 1.1 节中提到，智能信息检索技术是这个领域研究的热点。从检索算法的角度出发，核心的研究集中在两个关键技术：“ 标引 ”和“ 相似度计算 ”。在随后的 1.2 节中说明了，词汇语义模型是一种比较理想的浅层语义表述方式。因此，在信息检索中引入智能技术的一种方案，就是在两个关键技术中引入词汇语义模型，用浅层语义来指导检索过程，提高检索的准确率。这正是文本选题的思路。我们相信，“ 基于语义模型的信息检索 ”能够有效地提高文本检索性能。

本文的结构和内容安排如下：

(1) 第一章，简要介绍了文本检索的背景和研究现状，以及已有的一些词汇语义模型；其中说明了文本信息检索中两大关键性的技术：“ 查询主题词的构造 ”和“ 相似度计算 ”，并提出语义模型如何与这两个关键技术结合的问题。从第二章到第四章，分别论述了三类词汇语义模型，以及它们在信息检索中的应用。
(2) 第二章，在前人隐含语义标引系列模型的基础上，论述了改进的 SPLSI 模型，使其语义空间分布更合理，效率也更高。面向的主要是“ 查询主题词构造 ”技术。
(3) 第三章，提出了一种构造语义空间的全新方法—语义树模型。面向的同样是“ 查询主题词构造 ”技术。语义空间不再是静态的，而是实时构建的，其灵活性和可操作性都非常好。尤其应用在查询主题词扩展技术上，有很

出色的表现，性能超过了常见的扩展算法。

(4) 第四章，提出了语义张量的概念，并明确了其物理意义，提出了两个核心思想。进一步，用窗口系列模型来表述这两个思想，并应用于查询和文本间的相似度计算。实验证明，这系列模型比传统的矢量模型更有效。面向的是“相似度计算”技术。

(5) 第五章，前半部分介绍了 NLPR 检索系统的构成、实现方式、模块定义、函数接口等等；其中包含了如中文分词、英文 Stemming 等算法的描述，也包含了工程实现的内容。第五章后半部分介绍了信息检索评测的基本指标和 TREC 评测的基本情况，以及结合语义模型技术的 NLPR 检索系统，参加 2003 年 TREC 评测的情况（Robust 检索和 Novelty 检索）。

(6) 第六章，文本内容的总结与展望。着重描述了本文的研究课题在未来可能的发展方向。

第二章 基于隐含语义系列模型的信息检索

2.1 隐含语义标引模型（LSI-SVD）

隐含语义标引模型（Latent Semantic Indexing - LSI）是 DEERWESTER 等在 1990 年提出的一种语义空间模型[1]。LSI 将词和文本之间的关系用一个相似度值来衡量，然后通过一系列的数学方法，生成一个词与词之间的相似度矩阵。从这个矩阵中，可以获得任意两个词之间的相似度值；因而可被用来作为信息检索中主题词扩展的依据。

LSI-SVD 算法的数学模型如下：

设 X 为一个实数 n 乘 p 矩阵，可以求得一个 n 乘 n 的正交矩阵 (orthogonal matrix) U 以及一个 p 乘 p 的正交矩阵 V，使得 X 分解为：

$$X = U^T \sum V$$

其中

$$\sum = diag(\sigma_1, \sigma_2, \cdots, \sigma_m)$$

满足

$$m = \min\{n, p\} \quad \sigma_1 \geq \sigma_2 \geq \cdots, \geq \sigma_m \geq 0$$

上式称为 X 的奇异值分解形式，纯量 $\sigma_1, \sigma_2, \cdots, \sigma_m$ 为 X 的奇异值，U 的行向量是 X 的左奇异向量(left singular vector)， V 的行向量为 X 的右奇异向量。当 $n \gg p$ 时，则不须储存全部的 U 矩阵，将 U 压缩成 U1 有 p 行向量，而

$$X_1 = U_1^T \sum_1 V_1$$

这称为 X 矩阵的奇异值压缩。

LSI-SVD 模型的具体实现过程如下：

首先根据训练语料，构造一个“ 文档—词 ”的相似度矩阵 C，矩阵中的每个元素是相应词在文档中出现的次数或者频度。根据矩阵分析中的奇异值分解（SVD）算法，得到：$C = U \sum V^T$，既分解为三个矩阵，如下图 2.1。

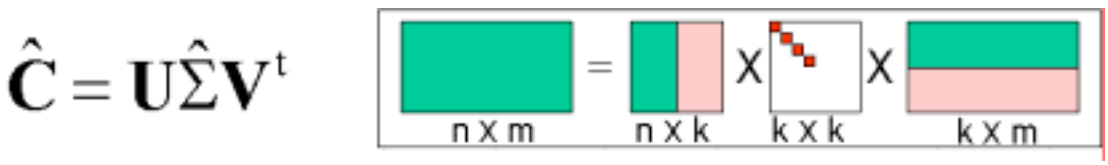


图 2.1 隐含语义标引的奇异值分解

其中，中间的 k x k 矩阵为对角阵，仅对角线上的元素取非零值，即为原矩阵 C 的奇异值,并且从大到小排列。矩阵的奇异值代表了矩阵本身的特征，而值比较大的那些奇异值代表了矩阵的主要特征。因此，可以把较小的奇异值忽略，从而压缩了大量的空间。如图 2.1 所示，当 k x k 矩阵仅取对角线上面一部分值的时候，矩阵 U 的右边部分和矩阵 V 的下半部分就自动缩减了。
在忽略了较小奇异值后，再把右边三个矩阵重新相乘，获得了新的“ 文档—词 ”相似度矩阵 $\hat{C}$ 。$\hat{C}$ 和原始的矩阵 C 比较有两个改进：

(1) 由于 $\hat{C}$ 忽略了较小的奇异值，因此也除去了原始矩阵 C 中的噪声；

(2) 原始 C 矩阵中的元素是“ 词在文档中出现的次数 ”，因而矩阵必然是稀疏的，很难直接利用。而 $\hat{C}$ 由于删除了较小奇异值，相当于做了平滑（smoothing），因此矩阵几乎在每个位置上都有值。也就是说，可以获得任意一个训练文档和任意一个词之间的相似度，无论这个词是否在这个文档中出现过。

基于 $\hat{C}$ 矩阵的新特性，将 $\hat{C}$ 乘以它自己的转置，即可获得“ 词-词 ”之间的相似度矩阵。这个矩阵就是语义空间的一种数学表述，可以用来进行主题词扩展，文本分类等一系列的工作。

以下是一个 LSI 具体例子。设有 c1-c5 以及 m1-m5 共 10 篇文本，各个词在文本中出现的次数见下图 2.2，既为矩阵 C。

Terms	Documents								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

图 2.2 词-文档 原始相似度矩阵

对这个矩阵 C 进行 SVD 分解 $C = U \quad \sum V^T$ ，得到以下三个矩阵: (U, $\sum$, V^T)

$$U=\begin{bmatrix}
0.22 & -0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41\\
0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11\\
0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49\\
0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01\\
0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27\\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05\\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05\\
0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17\\
0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58\\
0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23\\
0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23\\
0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18
\end{bmatrix}$$

图 2.3(a) SVD 分解后的 U 矩阵

$$\Sigma=\begin{bmatrix}
3.34 & & & & & & & & \\
& 2.54 & & & & & & & \\
& & 2.35 & & & & & & \\
& & & 1.64 & & & & & \\
& & & & 1.50 & & & & \\
& & & & & 1.31 & & & \\
& & & & & & 0.85 & & \\
& & & & & & & 0.56 & \\
& & & & & & & & 0.36
\end{bmatrix}$$

图 2.3(b) SVD 分解后的 Σ 矩阵

$$V^{T}=\begin{bmatrix}
0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06\\
0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24\\
0.46 & -0.13 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02\\
0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08\\
0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26\\
0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62\\
0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02\\
0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52\\
0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & 0.04 & -0.07 & -0.45
\end{bmatrix}$$

图 2.3(c) SVD 分解后的 V^{T} 矩阵

在中间 Σ 矩阵中，仅保留最大的两个奇异值，再把三个矩阵相乘，就可以得到更新过的 $\hat{C}$ 矩阵 $\hat{C} = U \hat{\Sigma} V^{T}$，如下图 2.4。

$\hat{C}$ = U $\hat{\Sigma}$ V^T

0.22 -0.11	3.34		0.20 0.61	0.46	0.54	0.28	0.00	0.02	0.08
0.20 -0.07		2.54	-0.06 0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.24 0.04									
0.40 0.06									
0.64 -0.17									
0.27 0.11									
0.27 0.11									
0.30 -0.14									
0.21 0.27									
0.01 0.49									
0.04 0.62									
0.03 0.45									

=

0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

图 2.4 生成更新过的$\hat{C}$ 矩阵

从上面的例子中可以明显的看到，原始矩阵 C 是稀疏的，而更新后的$\hat{C}$ 矩阵每个位置都有值。也就是说，我们可以获得任何一对“ 词-文档 ”之间的相似度值，无论这个词是否在这个文档中出现过。进一步来说，把$\hat{C}$ 矩阵乘以它自己的转置，就可以获得“ 词-词 ”的相似度矩阵，显然这个矩阵也是非稀疏的。从而，我们可以得到任何两个词之间的相似度值，无论这两个词是否在训练语料中一起出现过。利用这个“ 词-词 ”的相似度矩阵，就可以实现信息检索中的多项应用，具体见本章的 2.4 节。

SVD 算法的数学模型是完备的，但也存在一些缺点：

（1） SVD 算法不具备明确的物理意义，很难确切的表述矩阵特征值的缩减到底代表了什么。

（2） SVD 算法的过程不可控制，算法的时间和空间复杂度很大；这个特点使得 SVD 算法在目前的运算能力下很难大规模的应用。在本章 2.5 节的实验中有进一步的说明。

（3） SVD 算法最后得到的更新过的$\hat{C}$ 矩阵（图 2.4）中既有正值也有负值，也就是说“ 词-矩阵 ”之间的相似度值可能是负的。这和一般对相似度的理解有较大的冲突，很难给出这个矩阵一个明确的物理意义。并且，这种相似度值的不确定性，也给后续的应用带来困难。

LSI-SVD 算法的这些不足，正是以后 PLSI 模型（2.2 节）和 SPLSI 模型（2.3 节）试图改进的焦点。

2.2 统计隐含语义标引模型（PLSI-EM）

统计隐含语义标引（Probabilistic Latent Semantic Indexing - PLSI）模型是 Hofmann 在 1999 年提出一种语义模型[2]。在外在的表现形式上，PLSI 和 LSI-SVD 很相似，但内在的思想精髓是完全不同的。

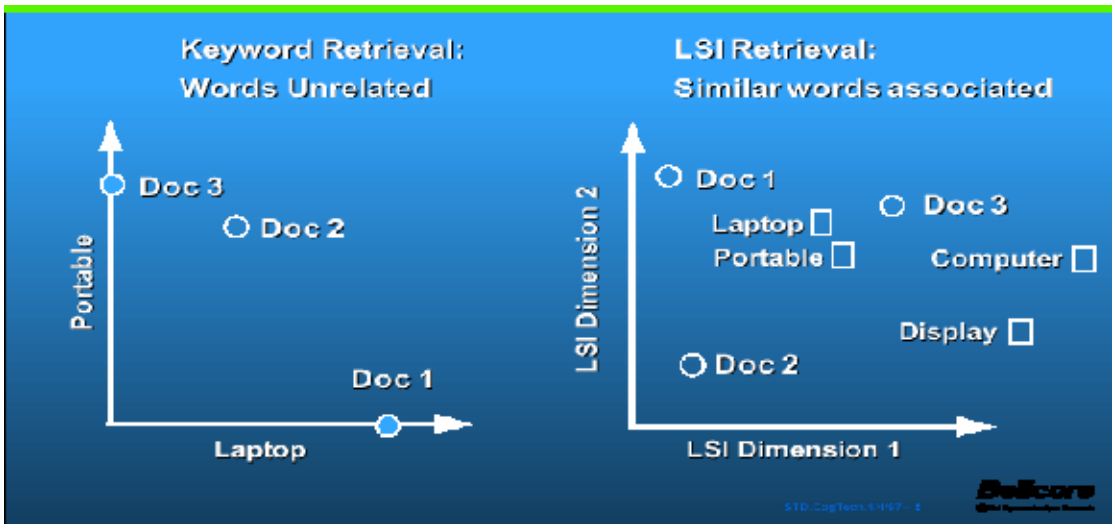


图 2.5 PLSI 模型的主要思想

PLSI 是一个概率模型，其主要思想见图 2.5。传统的自然语言分类模型中，词往往被作为一个个孤立的带有特定语义的实体来处理。如图 2.5 的左边部分，每个词被处理成独立的一维，所有的词构成了一个高维的语义空间；每个文档在这个语义空间中映射为一个点。这种传统的方法有两个缺点：

（1） 每个词都有一个独立的维，使得最后的语义空间维数很高。但实际上数学上并不需要这么高的维度来表达语义。

（2） 主要的，传统模型把每个词作为一维，割裂了词与词之间的关系。

PLSI 模型很好的克服了上面两个弱点。首先构造了一个维数不高的语义空间，将所有的“词”和“文档”同等看待，映射为这个语义空间中的一点，如图 2.5 右边部分。这样，既解决了维数过高不易处理的问题，也把词与词之间的关系体现出来了，语义上越接近的词在这个语义空间中几何上也越接近。而在构造这种映射的过程中，PLSI 模型采用的是迭代的 EM 算法，使得算法的效率更高。

PLSI 模型的主要算法流程如下：

（1）从概率的原则出发，建立文档 d 与词 w 之间的关系,既两者之间的概率相似度 P(w,d)。为了体现两者之间的内在联系（语义），以及压缩数据的需要，引入了隐含的多维变量 z，可以理解为语义空间，也可以理解为主题空间。这个空间的维度是事先决定的，一般可以是 20-100。

（2）根据条件概率公式，可得：

$$P(d,w)=\sum_{z\in Z}P(z)P(w\mid z)P(d\mid z)$$

其中，P(z)是语义空间是先验概率，P(w|z)是语义空间到词的条件概率，P(d|z)是语义空间到文档的条件概率。从数据中，无指导的训练出以上的参数，需要用 EM 算法。

（3）不同于 SVD 的分解，这里的决策函数是更有统计意义的最小熵原则，也就是最大化以下的函数：

$$L = \sum_{d \in D} \sum_{w \in W} n(d,w) \log P(d,w)$$

其中，n(d,w)是初始值是词 w 在文档 d 中出现的频度，P(d,w)是相应词与文档间的概率相似度。

然后，用一个标准的 EM 算法优化，

E-step:

$$P(z \mid d,w) = \frac{P(z)P(d \mid z)P(w \mid z)}{\sum_{z' \in Z} P(z')P(d \mid z')P(w \mid z')}$$

M-step:

$$P(w \mid z) = \frac{\sum_{d} n(d,w)P(z \mid d,w)}{\sum_{d,w'} n(d,w')P(z \mid d,w')}$$

$$P(d \mid z) = \frac{\sum_{w} n(d,w)P(z \mid d,w)}{\sum_{d',w} n(d',w)P(z \mid d',w)}$$

$$P(z) = \frac{1}{R} \sum_{d,w} n(d,w)P(z \mid d,w), \qquad R \equiv \sum_{d,w} n(d,w)$$

经过多次迭代，就可以求得比较准确的词与文档之间的概率相似度矩阵 P(w,d)。进一步，使用这个结果可以求得词与词之间的相似度矩阵 P(w,w)和文档与文档之间的相似度矩阵 P(d,d)，从而可以引用于信息检索，文本分类等，参见本章 2.4 节。

PLSI 的迭代算法，使得它比 SVD 算法更容易实现。并且语义空间 Z 有明确的物理意义，代表了隐含的主题；而其他的各个概率量也都有相应的物理意义。所以，相对于 SVD 来说，运算速度快，并且分类更合理（基于最小熵的原则）。图 2.6 是 Hofmann 给出的采用 PLSI 算法获得的部分词聚类结果。

"plane"	"space shuttle"	"family"	"Hollywood"
plane	space	home	film
airport	shuttle	family	movie
crash	mission	like	music
flight	astronauts	love	new
safety	launch	kids	best
aircraft	station	mother	hollywood
air	crew	life	love
passenger	nasa	happy	actor
board	satellite	friends	entertainment
airline	earth	cnn	star

图 2.6 PLSI 算法获得的部分词聚类结果

PLSI 模型在物理意义以及算法的运算时间方面都优于 LSI-SVD 算法。但是，PLSI 仍然有一些不足：

(1) PLSI-EM 是一种纯粹的无指导学习，算法迭代时收敛比较慢（在本章 2.5 节实验中有进一步表述）。

(2) 采用 PLSI-EM 算法获得结果，既得到 P(d,w)矩阵以后，如果又有了一批新的训练语料，想要更新获得的 P(d,w)矩阵，唯一的办法就是从头开始迭代算法，效率不高。

在下一节中，针对 PLSI 模型的这些不足，我们对 PLSI 进行了一些改进，称之为 SPLSI 模型。并在 2.4 节和 2.5 节中证明了在信息检索的应用中，SPLSI 模型更有优势。

2.3 弱指导的统计隐含语义标引模型（SPLSI-EM）

SPLSI 是“有指导的统计隐含语义标引”（Weakly-supervised Probabilistic Latent Semantic Indexing - SPLSI）的简称。与 PLSI 模型相同，采用了概率模型来衡量文档—词之间的关系，并引入了隐含空间的概念。这样，每一文本就被映射到一个语义空间，每个词也同样（见图 2.5 和图 2.8）。从而，无论是词与词之间还是文档与词之间，都通过这个语义空间来表达相似度，就有了相当明确的物理意义与可信度。在优化的决策函数方面，通用采用的是最大熵模型，聚类效果要好于 LSI 的最小二乘模型。

SPLSI 模型与 PLSI 模型的不同主要体现在以下方面：

(1) SPLSI 模型对文档进行了预聚类，从而可以指导 EM 迭代算法的初始值，使得收敛的更快。

(2) SPLSI 模型对 n(d,w)矩阵的定义和 PLSI 模型有所不同。采用了归一化的频度矩阵，使其物理意义更明确。

(3) 由于 EM 算法通常只能收敛在局部最优。因此，SPLSI 模型采用多初始

值，进行迭代，更容易收敛到最优值。

(4) SPLSI 模型提出了一种更新方案，使得新的训练语料能融入到已经训练好的“词-文档”概率相似度矩阵 P(w,d)中。

这四点都是试图弥补上节提到的 PLSI 模型的不足之处。

SPLSI 模型的算法主要流程如下：

(a) 汉语分词与文本粗分类

首先采用是已经较为成熟的最大概率分词方法对中文文档分词。然后，需要对双语文本进行人工的粗分类，这个过程主要是为了对后续迭代算法的初值进行弱指导。从算法本身来讲，粗分类不是必须的。此处，我们简单的将双语文本分为以下的 10 类：艺术、经济、军事、政治、新闻、生活保健、体育、教育、法律、科学。将同类型的文本排列在一起，供下一步使用。由于是粗分类，所以不要求很高的准确率。

(b) 构造“文档—词”索引矩阵 M(W,D)

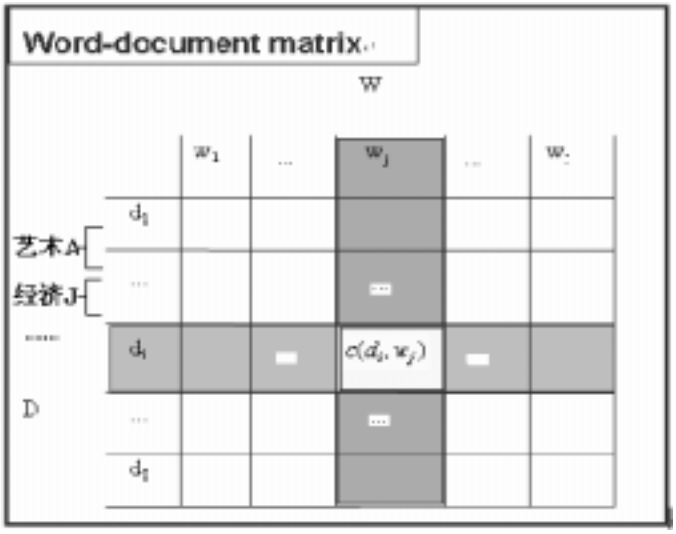


图 2.7 弱指导的初始“文档—词”索引]矩阵 M(W,D)

如图 2.7 构造初始的文档—词的索引矩阵 M(Word, Document)，其中的文档按照类型排序。矩阵 M 中元素的初始值 c(d,w)设为单词 w 在文档 d 中出现的次数。然后，需要进行归一化的操作，主要基于以下两个原因：

(1) 每篇文章中词的个数多少不同，因此一个词在短文章中出现一次的价值，显然应该大于在长文章中出现一次的价值；

(2) 一个很少出现的词，一旦出现在文档中，其价值应该大于普遍出现的词。事实上，类似于“the, 我们，的，of”之类的词几乎在任何文档中都会出现，因此其价值应该是趋向于零的。

所以，归一化的工作有两个。首先是根据一个停用词表（stoplist），把 M 矩阵中所有停用词对应的列去除。然后根据下面的公式进行归一化计算：

$$m(d,w)=\frac{c(d,w)}{Count\ \ (w)}\times\frac{\log\ \beta}{\log\ Length\ \ (d)}$$

其中，c(d,w)是矩阵 M 初始值（既词在文档中的出现次数），β 是系数，Count(w)是词 w 在所有文档中出现的总次数，Length(d)是文档 d 中所有非停用词数。

(c) 构造语义空间，确定映射初始值

构造 k 维的语义空间 Z，并且依据(a)中的粗分类结果给出语义空间的先验概率 p(z)。

具体的操作如下：设有 n 篇文档，文档共分为 t 种类型，其中第 1 篇到第 i 篇是同一类型的，那么有：

$$p(z_1)=\ p(z_2)=\ ...\ =\ p(z_{[k/t]})=[\frac{i}{n}\times\frac{1}{[k\ /\ t]}]$$

其中，‘ [ ] ’表示取整操作，k 值的选取依赖于经验，如果太小则无法把各类分开，如果太大则太敏感，容易引入噪声；在一般应用中可取 20 到 100。

有了语义空间后，需要分别构造 ”文档—主题 ” 的映射矩阵 P(D,Z)和 ”词—主题 ” 的映射矩阵 P(W,Z)，并给出初始值。设共有文档 n 篇，其中文档 d 属于第一类，而第一类的文档共有 i 篇，则：(其余部分可以依次类推)

$$p(z_1\mid d)=...=\ p(z_{[k/t]}\mid d)=\frac{1}{[k/t]};\text{其余}\ p(d,z)=0$$

这样做的目的是给每类文档分配语义空间（主题空间）中固定的若干维，使得不同类的文档在语义空间中初始就有比较大的距离，而同类的文档在语义空间中很接近。

而对矩阵 P(W,Z)，根据词 w 在哪类文档中出现次数最多，就按照这类文档的情况给出初始值，也就是说把词 w 安置在语义空间中这类文档所分配到的维度上，参见图 2.8 右边部分。需要注意的是，必须满足概率矩阵的条件，也就是任何一行的值之和必须是 1。

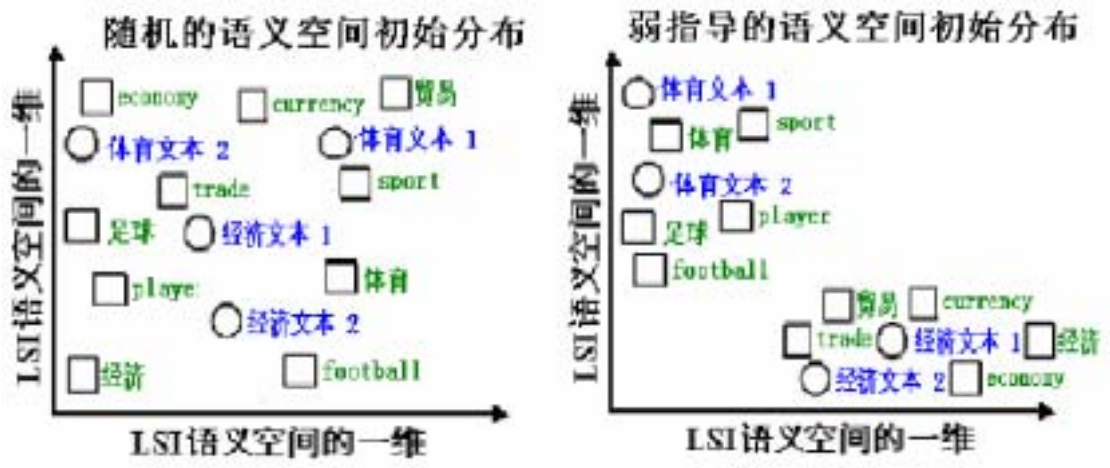


图 2.8 词和文档在语义空间中的初始分布

如图 2.8,采用上述弱指导的初始分布以后，相似的文本和词在语义空间中更接近，更趋近于合理，从而也更容易收敛到最优值。

(d) 采用 EM 迭代算法，求得结果

根据上述的结果，可以求得“ 文档—词 ”的相似度矩阵 P(W,D)初始值：

$$p(w,d) = \sum_{z \in Z} p(z \mid w)\, p(z)\, p(z \mid d)$$

然后，在最小熵的意义下，进行优化。即最大化以下决策函数（其中 m(w,d)是索引矩阵 M 中的元素）：

$$L = \sum_{w \in W} \sum_{d \in D} m(w,d) \log\ p(w,d)$$

对语义空间的每一维，采用标准的 EM 算法，进行迭代最优化。

E-Step:

$$p(z \mid w,d) = \frac{p(z)\,p(d \mid z)\,p(w \mid z)}{\sum_{z' \in Z} p(z')\,p(d \mid z')\,p(w \mid z')}$$

M-Step:

$$p(w \mid z) = \frac{\sum_{d \in D} m(w,d)\,p(z \mid w,d)}{\sum_{d \in D} \sum_{w' \in W} m(w',d)\,p(z \mid w',d)}$$

$$p(d \mid z) = \frac{\sum_{w \in W} m(w,d)\,p(z \mid w,d)}{\sum_{d' \in D} \sum_{w \in W} m(w,d')\,p(z \mid w,d')}$$

$$p(z) = \frac{\sum_{d \in D} \sum_{w \in W} m(w,d)\,p(z \mid w,d)}{\sum_{d \in D} \sum_{w \in W} m(w,d)}$$

反复应用上述 EM 公式，直到决策函数的变化量很小，即可认为达到了局部最大值。由于 EM 算法只能收敛到局部最优，所以在(c)中设定初值的基础上叠加不同的随机扰动，获得 10 组初始值；然后由这 10 种初始情况出发进行 EM 迭代。这样就大大增加了获得最优解的可能性。

在获得优化的 P(Z),P(W,Z),P(D,Z),P(W,D)矩阵后，利用以下公式，可以得到“ 词—词 ”相似度矩阵 P(W,W)。

$$p(w_1,w_2) = \sum_{d \in D} p(w_1 \mid d)\,p(d \mid w_2)$$

在 P(W,D)和 P(W,W)矩阵中，中文词和英文词的地位是完全等同的。

(e) 模型的动态扩展与优化

如果已经构造并且最优化了上述的一组 P(Z),P(W,Z),P(D,Z),P(W,D)矩阵，当有一些双语新文本 d ’到来的时候，我们拥有了新的知识。如何将这些新知识融合到这组矩阵中就成为一个问题。虽然 EM 算法本身是无法打断的，但可以有一种变通的办法来处理。
如果我们已经有了 P(W,Z),P(D,Z)等一组矩阵，当文档 d ’到来的时候，首先根据（b）中的步骤构造子 d ’的标引矩阵 M ’(W,D ’)，并把它拼接到原始的 M 中，形成一个新的 M 矩阵,参见图 2.9。这一步增加了文档的维度，而词的维度保持不变。然后,根据(c)中的步骤构造子矩阵 P ’(D ’,Z),同样的将其拼接到 P(D,Z)中。重新进行 EM 迭代，就可以扩展模型，加入新知识了。

图 2.9 标引矩阵 M 的更新

上述(a)-(e)五步就是 SPLSI 算法的主要流程。SPLSI 模型在四个方面改进了 PLSI 算法，使得花费更少的时间代价，就可以获得更好的“ 词-文档 ”概率相似度矩阵。此外，还引入了一个变通的更新数据的方法。在 2.5 的实验评估部分，将进一步阐述 SPLSI 算法的有效性。

2.4 面向信息检索的应用

在本章的前三节，描述了隐含语义标引系列算法的三种具体模型。三者的数学模型是不同的，但外在的表现形式很相似。输入都是由训练语料种统计得到的词在文档中出现的频度矩阵 c(d,w)，输出都是“ 词--文档 ”概率相似度矩阵 P(D,W)。因而，针对某种应用，可以约定输入输出，从而比较公正的比较三者之间的不同。

在信息检索领域，隐含语义标引系列算法的应用是非常广泛的，以下将具体说明应用方法。首先需要将输出的矩阵 P(D,W)乘以它自己的转置，从而获得“ 词—词 ”概率相似度矩阵 P(W,W)，参见下面的公式：

$$p(w_1, w_2) = \sum_{d \in D} p(w_1 \mid d)\, p(d \mid w_2)$$

这个 P(W,W)矩阵，在信息检索方面至少有三种方式的应用：跨语言（或者单语言）文本分类，跨语言（或者单语言）检索中相似度的计算，跨语言（或者单语言）检索中的关键词扩展。这里把文本分类也作为信息检索的应用，是因为以下两点理由：

（1） 信息检索的目标是要找出和查询主题相关的文本。换个角度也可以说是进行了文本分类，把文本分成“ 和主题相关 ”与“ 和主题无关 ”两类。
（2） 在信息检索的某些相关反馈技术中，需要对系统首次返回的文本进

行分类或者聚类，然后去修正关键词。也就是说，文本分类技术为相关反馈提供了支持。

这三种方式的应用描述如下：

(1) 跨语言(单语言)文本分类

文本分类问题的核心是计算文本之间相似度。设从文本 do 中抽取词向量 Wo，其维度等于 P(W,W)矩阵的行向量维度，其元素 Wo(word)为词 word 在文本中出现次数的归一化值；类似地，获得文本 dn 的词向量 Wn。利用 P(W,W)，得到文本 do 和文本 dn 相似度：

$$Similarity\ \ (do, dn) = Wo \bullet P(W, W) \bullet Wn^{T}$$

(2) 跨语言(单语言)文本检索

和（1）中相似，跨语言文本检索的核心问题是关键词向量和文本的相似度计算。把用户查询的关键词构造成词向量 Wq，其维度等于 P(W,W)矩阵的行向量维度。则查询关键词和文本 dn 相似度为：

$$Similarity\ \ (Query, dn) = Wq \bullet P(W, W) \bullet Wn^{T}$$

(3) 跨语言(单语言)查询关键词扩展

基于 SPLSI 的跨语言关键词扩展，实际上整合了机器翻译，词义消歧，语义扩展等多项功能。所有的工作综合起来，乘一个词间相似度矩阵即可完成。首先构造查询关键词向量 Wq,扩展后的关键词向量 We 为：

$$We = Wq \bullet P(W, W)$$

Wq 是相当稀疏的，而 We 几乎在每一项上都有值。这是符合设计思想的，任何词之间（包含中英文词或其他语言的词）都有一定程度的语义联系，区别仅仅在于这种联系的强弱。各种语言的词在 P(W,W)矩阵中都是相互平等的，完全可以看作一种语言来对待，所以跨语言的应用就和单语言的应用就完全相同了，不用任何额外的操作。

在下一节中，将对这些应用做各种评估，以了解隐含语义标引系列三种算法的具体表现。

2.5 实验评估

我们采用中英文双语文本对齐的语料库来进行实验，主要有以下两组数据：

a) 双语文本级对齐语料库 115 篇 新闻领域

b) 双语文本级对齐语料库 2041 篇 非特定领域

所使用的语料库包含在“中文语言资源联盟语料库”(www.chineseLDC.org）中。

根据上节提到的信息检索领域的三种应用，分别实验如下：

(1) 多语言文本分类的对比实验结果

采用以上的两个语料库，分别构建 LSI-SVD,PLSI,SPLSI 标引模型，然后应用所获得的词间相似度矩阵进行文本分类。同时，也实现了基于双语词典的关键词匹配（KW+Trans）等算法作为对比。实验结果如表 2.10。

训练语料库	算法	主题维度	关键词数	集内测试集文档数	集内准确率	集外测试集文档数	集外准确率
双语文本对齐 100 篇新闻领域	KW+Trans	--	8043	25	100.0%	15	80.0%
	LSI-SVD	20	8043	25	92.0%	15	73.3%
	PLSI	20	8043	25	96.0%	15	80.0%
	SPLSI	20	8043	25	96.0%	15	80.0%
双语文本对齐 1000 篇非特定领域	KW+Trans	--	33045	38	89.5%	41	82.9%
	LSI-SVD	时间和空间复杂度太大，暂无法计算					
	PLSI	50	33045	38	92.1%	41	85.4%
	SPLSI	50	33045	38	94.7%	41	87.8%

表 2.10 多语言文本分类对比实验结果

这里采用了两种不同规模的训练语料。第一种比较少，所以主题空间的维度设为 20；可以看到在训练语料比较少的情况下，隐含语义系列模型没有体现出什么优势，LSI-SVD 算法甚至比 Baseline 的效果更差。第二种训练语料比较充分，因此隐含语义系列算法体现出一些优势。从实验结果可以看出，SPLSI 算法是准确率最高，最有效的一种。值得注意的是，在训练语料达到一定规模以后，LSI-SVD 算法所需要的时间就很长，很难实际应用。

LSI-SVD,PLSI,SPLSI 三种算法，原始模型所获得的是“词—文档”概率相似度矩阵 P(W,D)。当它乘以自己的转置的时候，就得到了“词—词”概率相似度矩阵 P(W,W)；同理，当它的转置乘以它自身的时候，可以得到“文档—文档”概率相似度矩阵 P(D,D)。

为了给出一个直观的表述，用 100 篇双语文本分别训练出文档间相似度矩阵 P(D,D)，并用图 2.11 表现出来。

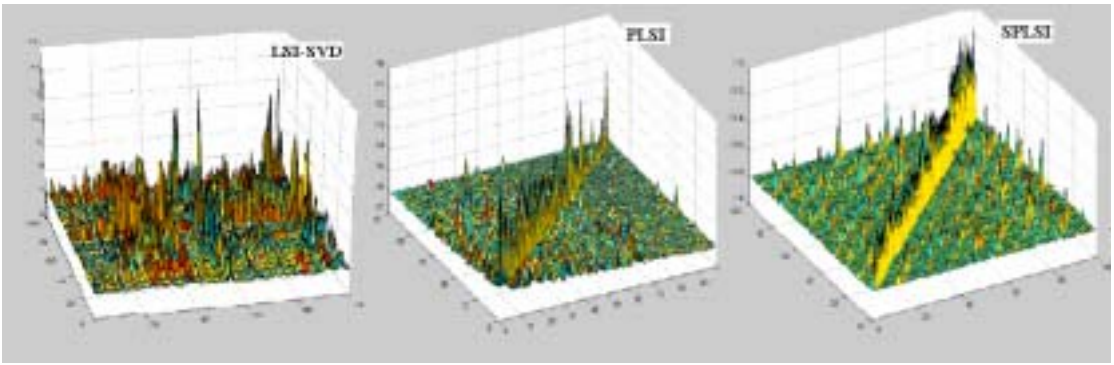


图 2.11 “ 文档—文档 ”概率相似度矩阵对比图

图中，x 和 y 轴代表的相应的文档，高度轴 z 代表了相应两个文档之间的相似度值。从左下到右上的对角线，显示了文档的自相似度值。显然，SPLSI 模型在这条线上的峰值更突出,体现了更强的物理意义 ;并且矩阵在整体分布上更为均匀，反映其分类更为合理。PLSI 模型虽然对角线也比较明显，但是整个矩阵看起来不够均匀。LSI-SVD 模型所获得的结果没有明显的对角线，这意味着一个文档和自己的相似度还比不上和某些其他文档的相似度，这显然是不合理的。

(2) 跨语言文本检索的对比实验结果

采用双语对齐的 1000 篇文本作为训练集,分别获得 PLSI 和 SPLSI 词间相似度矩阵 P(W,W)。然后，随机选择若干组查询关键词，在文本库中查找相关的文本。其中，KW+Trans 表示的是基于双语词典的关键词检索算法。

查询 关键词	算法	集内 准确率	集内 召回率	集外 准确率	集外 召回率
中国 贸易	KW+Trans	94.3%	61.1%	92.8%	56.5%
	PLSI	91.0%	86.2%	87.2%	83.1%
	SPLSI	92.7%	87.8%	90.3%	85.4%
football star match	KW+Trans	91.6%	63.7%	91.4%	58.3%
	PLSI	90.3%	88.2%	87.7%	84.7%
	SPLSI	92.5%	89.3%	91.9%	88.6%

表 2.12 跨语言文本检索的对比实验结果

如表 2.12,在随机抽取的用户查询测试中，无论是开放测试还是封闭测试，SPLSI 算法的综合指标比其他算法更出色，尤其在召回率方面，充分体现了语义扩展的强大功能。

在跨语言文本检索中，还有一个很重应用领域是：**译文检索**。采用与上述相同词间相似度矩阵 P(W,W)，进行译文检索；集内测试集采用的是直译文本，集外测试集采用的是意译文本。对比实验结果参见表 2.13。

算法	集内测试集 文档数	集内准确率 (直译)	集外测试集 文档数	集外准确率 （意译）
KW-Trans	24 * 2	95.8%	18 * 2	83.3%
PLSI	24 * 2	91.7%	18 * 2	88.9%
SPLSI	24 * 2	95.8%	18 * 2	94.4%

表 2.13 译文检索对比实验结果

由此可见，对于意译的文本，语义扩展算法的优势是相当明显的，比采用词典进行翻译的效果好很多。换一个角度来看，词间相似度矩阵也可以作为翻译概率矩阵来使用。一个中文词与一个英文词之间的概率相似度，可以看作它们之间互为翻译的概率。

(3) 跨语言检索中的关键词扩展

跨语言检索中的关键词扩展分为三个层次：翻译扩展、同义词扩展、语义扩展。前面反复用到的跨语言词间相似度矩阵，能够将这三个层次的扩展融为一体。利用上节描述的查询主题词扩展方法，采用 SPLSI 模型获得的部分跨语言扩展结果如表 2.14。

查询词	扩展词
自然 语言	natural language NLP 语法 linguistic
明星 文艺 唱歌	sing song 艺人 star 感谢 出售 制片
国民 经济 发展	develop economy 增长 trade economic
computer hardware	计算机 电脑 硬件 system 显示器 配件 CPU
football match	足球 比赛 player win 球员

表 2.14 跨语言主题词扩展

由此可见，SPLSI 模型所获得的跨语言的“词—词”概率相似度矩阵是非常有用的。最重要的是它把翻译的过程完全融合在数学模型里了，这是以前的模型所遇到的最大的困难。

在以上的三部分中，较为详细的描述了隐含语义标引系列三种算法在各种应用上的表现。总体上来看，LSI-SVD 模型的物理意义不明确，效果也相对比较差。而 PLSI 模型和 SPLSI 模型的效果比较接近，后者略优。

对这系列算法来说，如果进行上述的各种应用，在 On-line 过程中完全可以做到实时，因为所有需要做的仅是乘以一个矩阵。而在 Off-line 的部分，也就是训练过程，三种算法的区别就比较的明显，SPLSI 模型在此充分的显示了它的优势。由于 LSI-SVD 算法很难进行大规模的计算，所以主要对比了 PLSI 和 SPLSI 两种模型的效率。

软件平台	算法	预处理时间	内存占用（包括虚拟内存）	平均CPU利用率	EM算法迭代一次所需要的时间	EM算法平均迭代次数
Win2000 VC++6.0 SQL Server	PLSI	2 小时	2000 兆	5%	100 小时	8.2次
	SPLSI	2 小时	720 兆	10%	40 小时	4.3次
Linux GNU C MySQL	PLSI	0.5 小时	2000 兆	8%	60 小时	8.2次
	SPLSI	0.5 小时	720 兆	20%	20 小时	4.3次

表 2.15 时间与空间复杂度对比

如表 2.15 所示，在两种系统平台上做了实验。由于给出了更好的初值结构，并且在算法实现的过程中削减了内存的使用量，因而 SPLSI 算法所需的时间仅为

PLSI 算法的三分之一左右，平均的迭代次数也只有 PLSI 算法的一半左右。下面的图 2.16 简要的显示了两个算法的典型的迭代收敛过程（拟和示意图）。

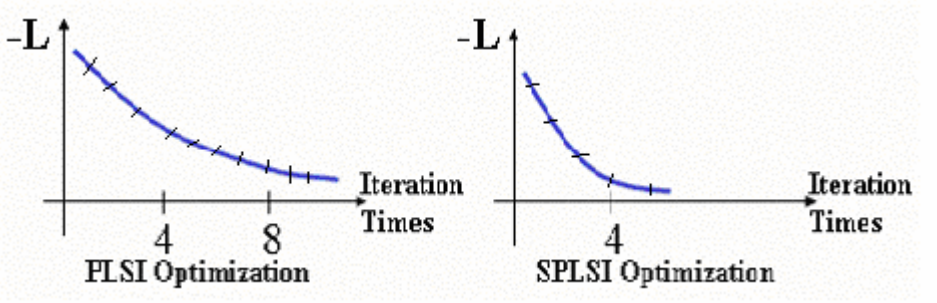


图 2.16 PLSI 和 SPLSI 的迭代过程（拟和示意图）

实验评估的一个总体结论就是，针对信息检索的应用，SPLSI 模型是隐含语义标引系列模型中最有效的模型，也是最高效的模型。

2.6 应用系统（Google-Shell）

目前，互联网已经给人们提供了许多方便：发邮件、看新闻、查资料、炒股票、玩游戏、远程教育、电子商务……但是，人们在享受网络通讯的种种便利之时，实际上都还揣着一个未能实现的共同美好愿望：凡发布信息，能及时到达所有需要者面前；凡寻求信息，能无一遗漏地自动来到自己的面前。换句话说，人们理想中的互联网，还应该是一个智能化的全球信息共享俱乐部。

由于目前的互联网是一个开放式的运行模式，既是动态改变的，又是多模态的。有用的信息往往分散在一系列相关的站点中，这些站点除了信息内容本身，并无任何的相似之处。这样就给自动收集，整理信息造成了很大的困难。为解决这一问题，人们发展出了搜索引擎技术。Google 就是一个非常优秀互联网数据的检索系统，用户可以提供一系列的查询关键词，Google 就能返回给用户包含这些关键词的相关的网页。在 Google 强大引擎的基础上，可以应用基于语义的信息检索技术，从而进一步提高互联网信息检索的有效性，这是我们的目标。

在上节的实验中，说明了 SPLSI 模型的有效性。在本节，利用 SPLSI 算法训练得到的‘词—词’概率相似度矩阵，实现了一个信息检索的应用系统：Google-Shell。该系统，正如其名称所显示的那样，是基于 Google 数据库来实现的，系统结构如下图 2.17 所示。

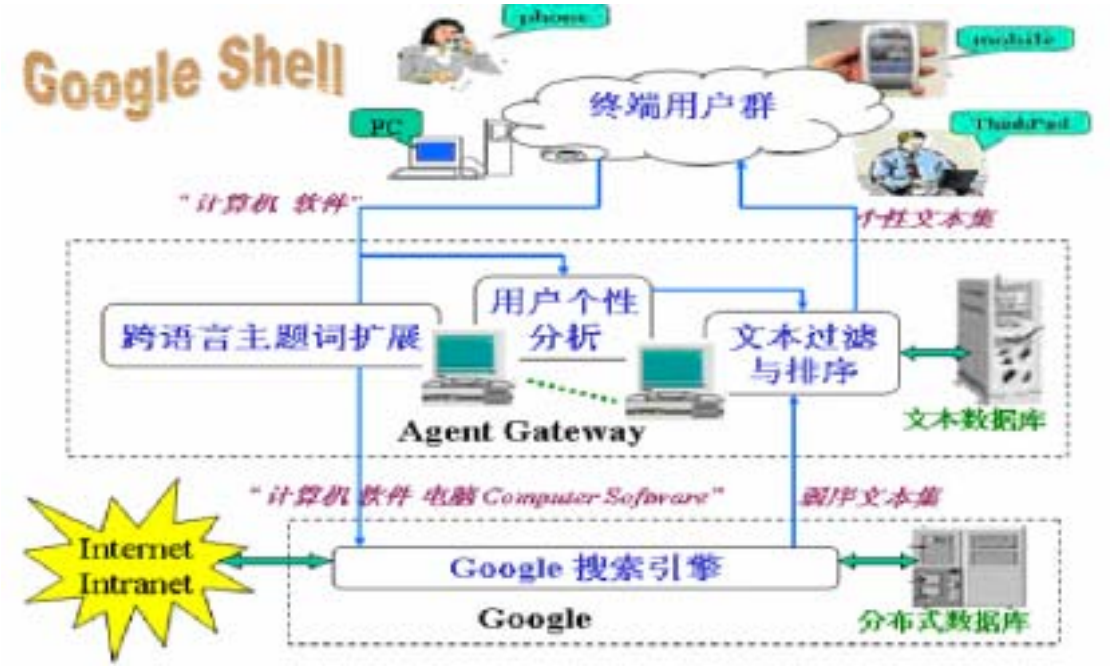


图 2.17 Google-Shell 系统的体系结构

Google-Shell 系统实际上是实现了一个互联网查询的代理（Agent）功能。用户把查询请求发送到这个系统，系统经过分析，扩展主题词以后，发送给 Google 搜索引擎。Google 返回的结果，经过个性化的整理以后，提供给用户。其中，涉及到本节上面提到的跨语言信息检索技术，是 Google-Shell 系统的核心技术之一。具体的实现过程，简单来说就是利用 SPLSI 模型得到的 P(W,W)矩阵来进行跨语言的主题词扩展。

在用户查询的时候，往往不能准确完全的给出查询的主题词。如，当要查“富士通公司”的时候，用户不知道这个词的英文具体拼写方法（应该是：“Fujitsu”），如果直接用 Google 搜索就不能获得英文标引的相关网页，但是 Google-Shell 采用了这种跨语言的关键词扩展之后就完全能够做到。

下面的图 2.18 是 Google-Shell 系统的界面。其中的输入框是用户输入查询的地方，如当想查找“北京的天气情况”，可输入：“Beijing weather”。输入框下面的四个可选项分别表示：

(1) Original：按照原始关键词检索
(2) Translation：进行关键词的跨语言扩展
(3) Synonymy：进行关键词的同意扩展
(4) Expansion：进行关键词的相似意扩展

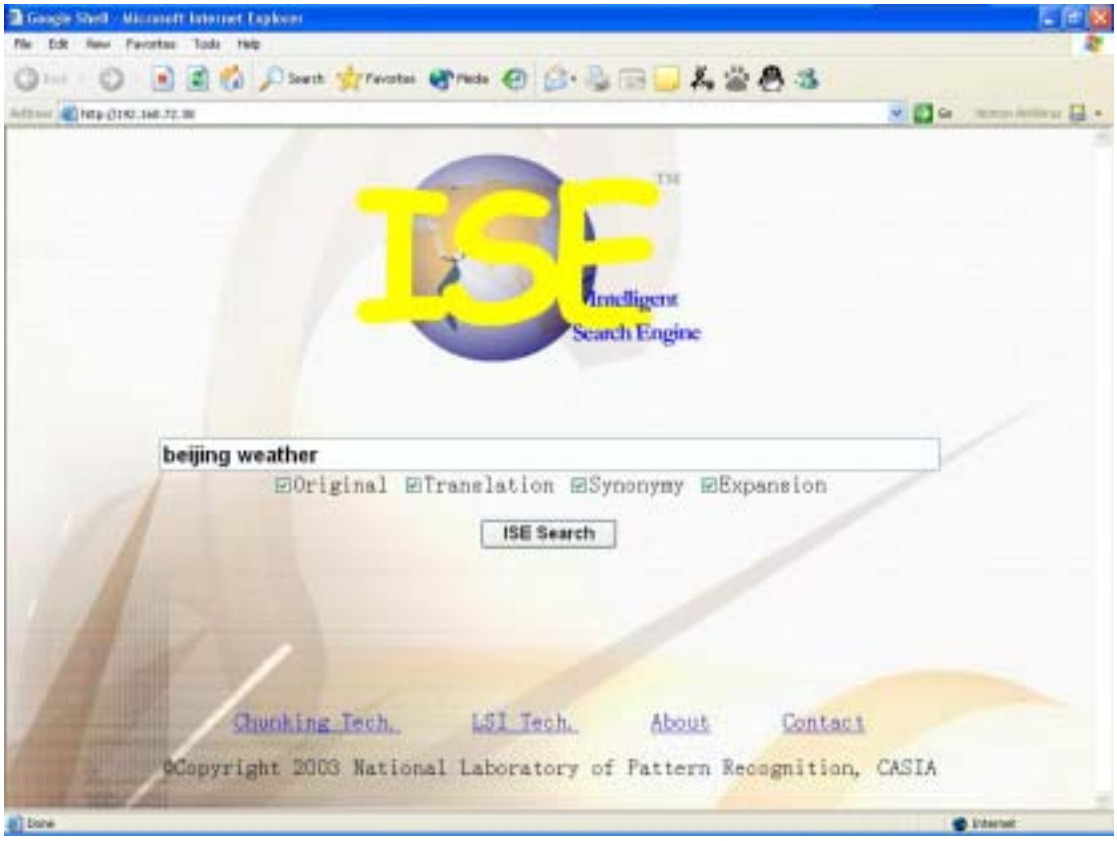


图 2.18 Google-Shell 系统的界面

当按下“ISE Search”(Intelligent Search Engine)按钮后，就开始互联网的检索过程。其查询关键词是：“Beijing weather”,查询选项都选上了。返回的部分结果如下：

（1） 如图 2.19，这是原始关键词查询所获得的部分结果。

图 2.19 Google-Shell 系统返回结果（一）

（2） 如图 2.20，这是跨语言扩展后，查询所获得的部分结果。扩展出了“北京 天气”。

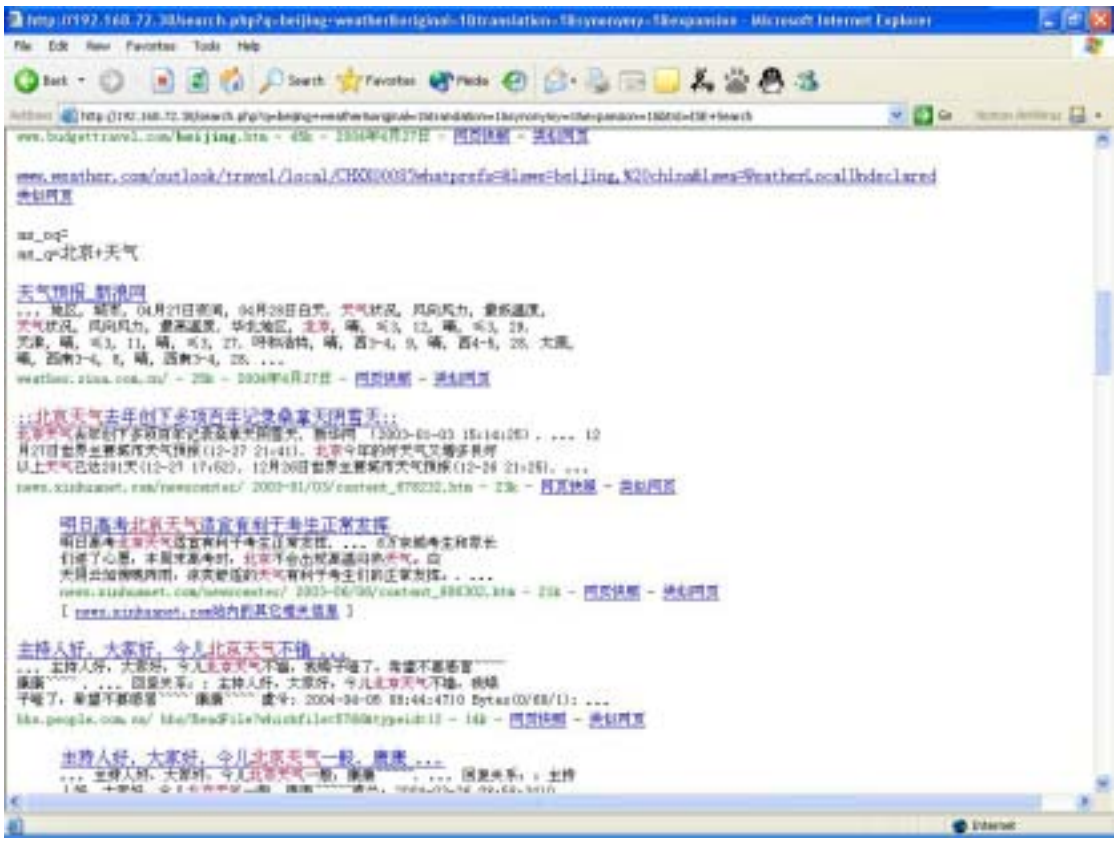


图 2.20 Google-Shell 系统返回结果（二）

（3） 如图 2.21，这是相似意扩展后，查询所获得的部分结果。扩展出了“ Peking weather ”。

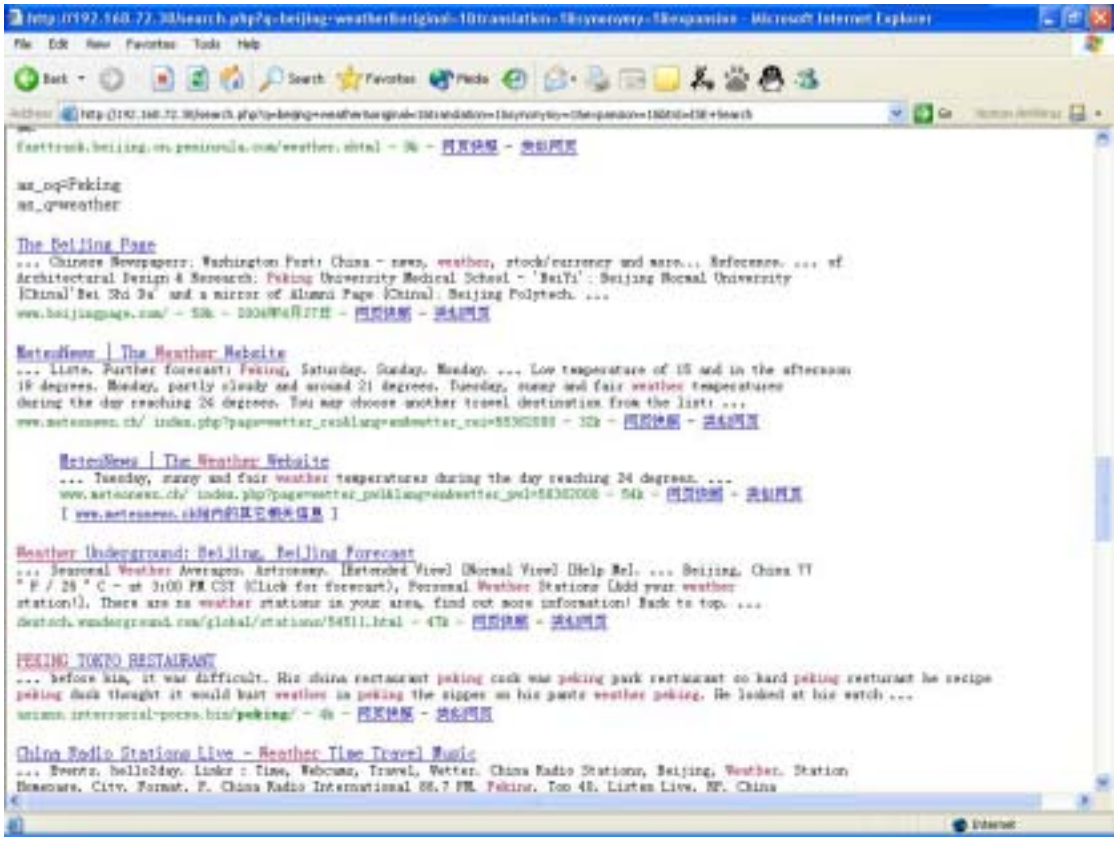


图 2.21 Google-Shell 系统返回结果（三）

由上面的结果可见，采用 SPLSI 技术进行的这种扩展，在现实的检索中非常的有效，能够找到很多以往被遗漏的相关网页。这个 Google-Shell 系统目前已经有了雏形，并在内部网上使用，获得了很不错的效果。

2.7 本章小结

在本章的前三节，详细描述了隐含语义标引系列三种算法模型的具体流程与实现方法。其中，第一种 LSI-SVD 模型和第二种 PLSI 模型是前人研究的成果，第三种 SPLSI 模型是本论文的创新点之一。相比前两个模型，SPLSI 有三个突出的优点：

（1） 物理意义明确
（2） 训练过程相对可控
（3） 算法效率相对比较高

本章的第四节，描述了隐含语义标引系列模型在信息检索领域的各种应用方法。提出了其数学模型。在随后的第五节中，分别用实验验证了各个模型在信息检索领域应用的成果。实验结果证明，在较小规模的应用中，SPLSI 模型是其中最有效，也是最高效的一个模型。

在本章的第六节,将 SPLSI 模型应用到实际的互联网检索代理系统 Google-Shell 中，得到了令人满意的效果。

本章可以得到的一个结论是，SPLSI 模型可以基于多语言语料，自动的获取跨语言的词语相似度矩阵，从而能够较好的解决语义约束和语言障碍的问题，在跨语言信息检索领域有广泛的应用。同时，SPLSI 也保留了这系列模型的共同缺点：时间和空间复杂度比较高，很难构造大规模的语义空间。

第三章 基于语义树模型的信息检索

3.1 隐含语义标引系列模型的不足

上面一章描述了隐含语义标引系列模型的算法，与其在信息检索中的应用。虽然，在数学上，三种模型都比较完美，但即使表现最好的 SPLSI 模型在信息检索的相关应用方面也有一些缺点。

(1) 训练过程的空间和时间复杂度比较高，大规模的“词--词”相似度矩阵很难得到。
(2) 训练过程可控制性不太好，新训练数据的融入比较困难。
(3) 对训练语料的要求比较高。最终获得的相似度矩阵的质量，很敏感的依赖于训练语料的优劣。
(4) 针对信息检索的应用中，如主题词扩展，SPLSI 模型引入了比较多的噪声。

考虑到这些不足，有必要面向信息检索的应用，来改造这些模型，克服上述的缺点。

3.2 从语义网到语义树

隐含语义标引系列算法的目标是构造“词—词”相似度矩阵，也可以称为是语义空间。在这个相似度矩阵中，任何两个词之间都有相似度；这一点反映在语义空间中就是任何两个节点之间都有连线。也就是说，这种语义空间是一个完全图，参见下图 3.1。

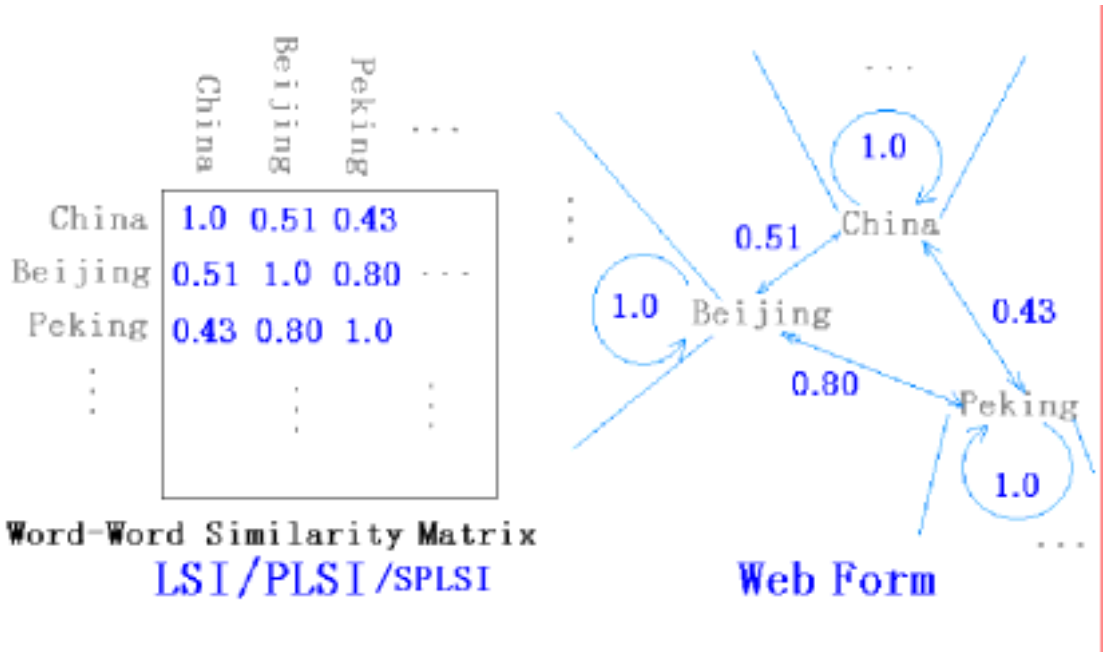


图 3.1 隐含语义标引系列模型的语义空间

图中的矩阵如果包含了三万词，那完全图的规模是相当可观的。现在的问题是，这么大的规模是否是必要的。我们考虑词与词之间的关系的时候，往往仅关心那些相似度比较大的词对，比如我们在关键词扩展的过程中，希望从“ Beijing ”扩展出“ Peking ”。如果两个词的含义几乎无关，在统计意义上也很少共现，那么给出它们之间的相似度值是没有实质上的意义的。

既然这么大规模的完全图并不是全有用，那么就应该考虑给它剪枝，也就是把那些相似度值小的路径删去，仅留下相似度值大的那些路径。进一步考虑，如果已经训练得到了完全图再去剪枝，显然意义并不是很大。理想的情况是，在训练的过程中，就自动能够保留那些有意义的路径。由于在数据结构方面，比“ 图 ”更简单更灵活的模型就是“ 树 ”，因此我们考虑采用树状的模型来构造语义空间，使得它能克服图状语义空间的不足。

这就是本章所要详细描述的“ 语义树模型 ”的最初思想来源。

3.3 语义树模型

本节将详细的描述语义树模型的定义和构建过程。虽然设计语义树模型的初衷是面向信息检索的应用，但它本身是一个完整的模型，可以通过它获得任何两个词之间的相似度，因此也可以同样地完成跨语言文本分类等工作。

与其他的语义空间模型相比，它有几下几个特点：

(1) 构成语义树模型的元素是预先构造好的，但是语义树本身是“ 即用即造 ”的。也就是根据特定的应用，实时的构造语义树，相当灵活。
(2) 实时构造的语义树可控性能非常好，可以仅仅包含几十个词，也可以包含上万个词，添加与删减词都很容易。
(3) 作为语义树的元素，其构造方法也很灵活，可以有多种数据来源，更新也很方便快捷。

因此，语义树模型的构造分为两大步，即：“ 构造语义树的元素(off-line) ”和“ 实时构造语义树本身（on-line）”，下面将详细说明。

构建语义树模型：

（1）建立语义树的元素

设 $PSim(q, p)$ 表示两个词 q 和 p 之间的先验相似度值。对任意一个给定的词 q，采用树状的模型来表达词 q 与所有其他词之间的关系，如下图 3.2 左边部分所示。

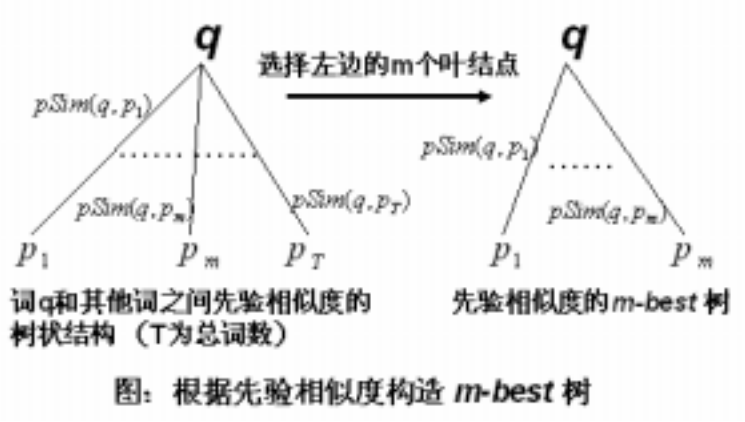


图 3.2 构造语义树的元素

显然，原词 q 是树根，而其余的词是树叶，两者之间树枝（也就是路径）的权值是两个词之间的相似度值。这里要注意的是，从 p1 到 pT 的所有词，是按照它们和原词 q 之间的相似度来排序的，和 q 相似度越高的词排在越左边。也就是要满足下式：

$$PSim\ (q, p_1) \geq ... \geq PSim\ (q, p_m) \geq ...$$

这里相似度 $PSim\ (q, p)$ 的计算有很多种方法，如：

（1） 两个词在训练语料中的共现频率
（2） 两个词在词典（如 WordNet[11]/HowNet[5]）中的距离
（3） 两个词共现的互信息
（4） 等等…（参见[6][23]）

然后，我们只保留最左边的 m 个树叶（对应最左边的 m 个词），并丢弃其余的树叶。如图 3.2 的右边部分，我们称之为原词 q 的 m-best 树。每个词都对应有它自己的 m-best 树，每个 m-best 树就是我们语义树模型的一个元素。这里 m 的选择依赖于特定的应用，经验取值是 10 到 100。

（2）实时建立语义树

前面已经提到，语义树模型是可以根据特定的应用来构造的。首先，我们需要给出一个初始的词向量，作为整个语义树的根节点，然后逐步展开构造整棵树。

设 $Q = (q_1, q_2, ..., q_i, ..., q_K)$ 表示初始的词向量，其中总共包含 K 个词，q_i 代表第 i 个词。从 Q 这个初始词向量出发，构造语义树 $TSTM\ (Q, v, m)$，其名称是 Term Similarity Tree Model 的缩写。其中，v 代表这个语义树的层数，m 代表组成这个语义树的每一个元素是 m-best 树，也就是每个相对根节点都连接着 m 个叶节点。构造好的语义树，参见图 3.3。

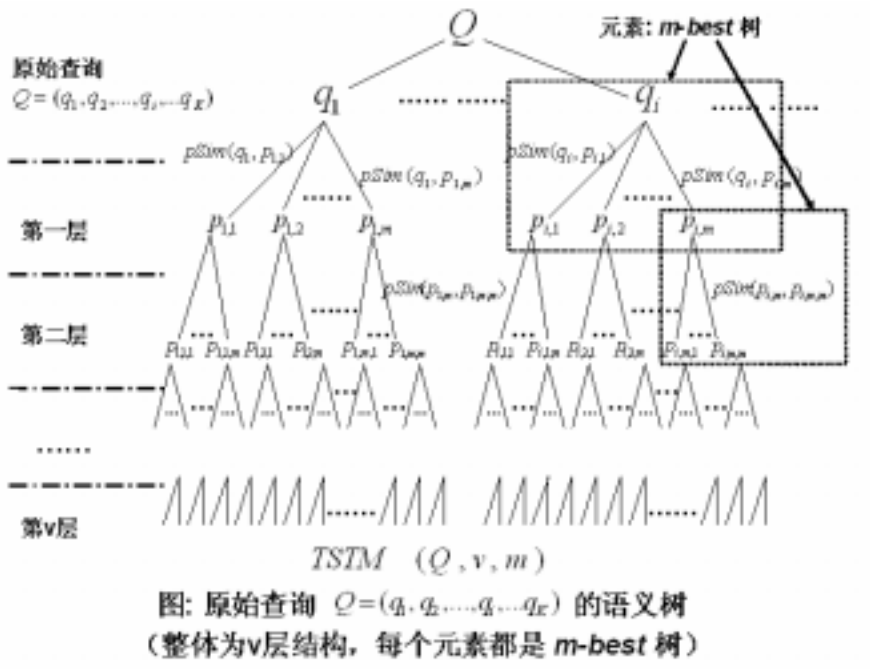


图 3.3 实时构造语义树

图中每个虚线的矩形框代表的是一个语义树元素（即一个 m-best 树）。事实上，整个语义树是由很多个 m-best 树在不同层次上搭建而成的。有了这个语义树之后，我们可以很容易的获得一个根节点词与一个叶节点词之间的相似度值，无论它们相隔的层数是多少。具体的定义如下：

(a)根节点 q_i 和它的叶节点 p 之间的**路径(path)**是指从节点 q_i 到节点 p 的树枝的路由。

(b)根节点 q_i 和它的叶节点 p 之间的**路径(path)的权值（weight）**是所有在路径上的树枝的权值相乘。树枝的权值在构造 m-best 树的部分已经给出了定义。

(c)根节点 q_i 和它的叶节点 p 之间的**最短路径(path)**是指它们之间拥有最大权值的路径。

(d)根节点 q_i 和它的叶节点 p 之间**相似度值**是指它们之间最短路径的权值。

这四个定义明确了两个词之间相似度的计算方法，如计算词 q_1 和词 $p_{1,i,j}$ 之间的相似度，可用以下公式：

$$\begin{aligned} & Sim\ (q_1, p_{1,i,j}) \\ & = weight\ -of\ -shortest\ -path\ (q_1, p_{1,i,j}) \\ & = PSim\ (q_1, p_{1,i}) \times PSim\ (p_{1,i}, p_{1,i,j}) \end{aligned}$$

事实上，任何一个根节点词与叶节点词之间的相似度都可以这么计算。而前面提到的根节点的词向量选择是任意的，所以任何两个词都可以依据语义树模型来计算相似度。这也就实现了一个语义空间模型的最主要的工作。

从语义树模型的构造过程我们可以看出：

(1) 一层的语义树模型（v=1）就是“局部共现模型”（Local Co-occurrence Model -- LCM）。所以，我们可以把语义树模型看作是局部共现模型的一个扩展。和 LCM 模型相比，语义树的优势主要体现在它有自动的语义聚类的功能。举个例子来说：词 computer 和词 hardware 经常在文本中一起出现，词 computer 和词 software 也经常一起出现，然而词 hardware 和词 software 没有一起出现过。这种情况下，在 LCM 模型中，词 hardware 和词 software 之间的相似度值就为零或者一个很小的值；但是基于语义树模型，两者之间就有一个合理的相似度值。

(2) 如果我们把语义树的层数 v 一直扩展下去，每层的叶节点数 m 扩展到词的总个数，语义树模型就接近了上章提到的隐含语义标引系列模型。而正式由于对层树 v 和叶节点数 m 的控制，使得语义树模型比 LSI 模型具有优势。一是时间和空间复杂度降低，容易计算；二是模型的可控性更强；三是更好的删除了噪声。

本节介绍了语义树模型的构造方法和优点，在下一节将说明如何使用这个模型来进行信息检索中的主题词扩展。

3.4 基于语义树模型的主题词扩展

基于语义树模型的主题词扩展分为两步，首先构造语义树，然后选择出符合要求的扩展词。具体的算法模型(TSTM 模型)如下：

设用户原始给出的主题词向量 $Q = (q_1, q_2, ..., q_i, ..., q_K)$ ，共包含 K 个词，其中 q_i 是第 i 个词。构造基于词向量 Q 的语义树 $TSTM\ (Q, v, m)$，其中 v 是语义树的层数，m 是指构成语义树的元素是 m-best 树。参见图 3.3。

当词 w 满足以下条件（公式）时，认为 w 是符合要求的扩展词：

$$\begin{cases} Sim\ (Q, w) = \sum_{i=1}^{K} Sim\ (q_i, w) \geq cv \\ Overlay\ \ (TSTM\ \ (Q, v, m), w) \geq\ percent\ \ \times K \end{cases}$$

其中 $Sim(Q, w)$ 是原主题词向量 Q 和词 w 之间相似度值；$Sim(q_i, w)$ 是词 q_i 和词 w 之间的相似度，可以按照上节描述的方法从语义树模型中计算得到。cv 是相似度的阈值。

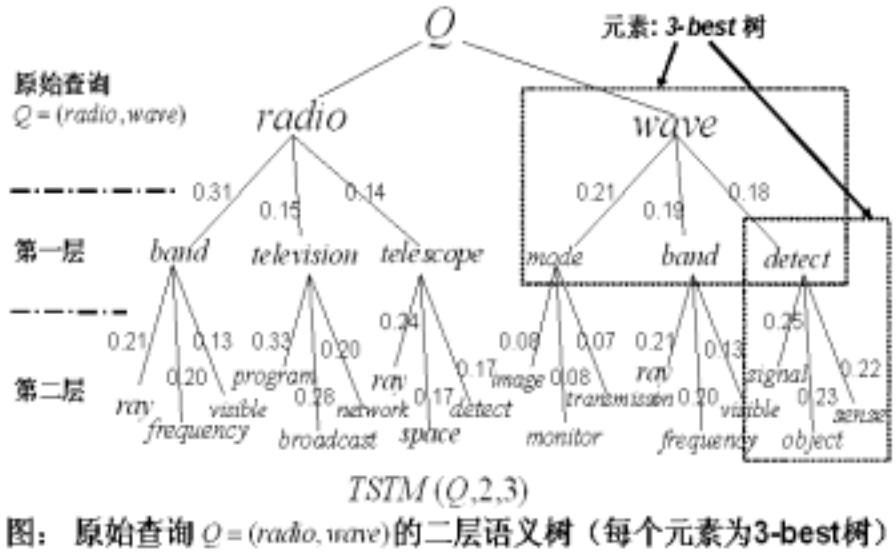
$Overlay\ (TSTM\ (Q,v,m),w)$ 是指词 w 在语义树 $TSTM(Q,v,m)$ 的子树中出现的次数。原主题词向量 Q 的总共有 K 个词，在语义空间中就有 K 个子树。$Overlay\ (TSTM\ (Q,v,m),w)$ 是指在这些子树中，有多少个包含了词 w，也可以说是有多少子树覆盖(Overlay)了词 w。*percent* 是覆盖度的阈值。

上面公式的第一个判别函数是用来评估词 w 和原每个主题词 $q_1,q_2,...,q_i,...,q_K$ 之间的相似度值。第二个判别函数是用来评估词 w 和真个查询主题 Q 之间的相似度。实际上，整个查询主题 Q(可能包含很多个词)可以看作是语义空间中的一个点，表达了用户的一个语义。我们希望找到的扩展词是在语义上接近于这个点，也就是接近整个查询主题 Q,而不是仅仅接近 Q 中的某些独立的词。因为，整个查询主题 Q 表达的意义是明确的，而 Q 中的单个词往往没有明确的意义。

举个例子来说，如果用户给出这样一个查询：

“radio, wave”

我们构造 2 层的语义树，每个元素是 3-best 树，参见下图 3.4。



图：原始查询 $Q=(radio,wave)$ 的二层语义树（每个元素为3-best树）

图 3.4 基于语义树的主题词扩展的例子

(1) 首先考虑词 ray 是否是合适的扩展词。在图中可以看到，从词 radio 到词 ray 有两条路径，分别是:

radio - band – ray

radio – telescope – ray

其中的最短路径（在本章 3.3 节中定义）是 radio-band-ray，所以我们可以计算两者之间相似度值如下：

$$\begin{aligned} &Sim(radio,ray)\\ &\quad = PSim(raido,band)\times PSim(band,ray)\\ &\quad = 0.31\times 0.21 = 0.0651 \end{aligned}$$

而词 wave 和词 ray 之间的最短路径是 wave-band-ray，所以我们可以计算两者之间的相似度值如下：

$$\begin{aligned} & Sim(wave, ray) \\ & = PSim(wave, band) \times PSim(band, ray) \\ & = 0.19 \times 0.21 = 0.0399 \end{aligned}$$

由此，针对词 ray，得到两个判别函数的值为：

$$Sim(Q(radio, wave), ray) = 0.105$$

$$Overlay(TSTM(Q,2,3), ray) = 100\%$$

(2) 完全采用相同的方法，考虑词 television，得到：

$$Sim(Q(radio, wave), television) = 0.15$$

$$Overlay(TSTM(Q,2,3), television) = 50\% .$$

(3) 如果我们采用的阈值是 *cv=0.08* 以及 *percent=100%*，就可以知道，词 ray 满足两个判别公式，而词 television 不满足。虽然说词 television 和原主题词 Q 有更高的相似度（和词 ray 比较），但是和 Q 的语义树覆盖度比较低，所以 television 并不是合格的扩展词。而 ray 由于满足了两个公式，被认为是符合要求的扩展词。

(4) 考虑了所有词以后，得到原查询扩展以后的主题词包括：

“radio, wave, band, detect, ray, frequency”

本节介绍了一个基于语义树模型的主题词扩展算法(TSTM 模型)，其核心思想是两个判别函数，分别考察扩展词与原查询的相似度和覆盖度。只有满足这两个条件的扩展词才被认为是符合要求的。在下一节中，将对这个算法模型进行实验评估。

3.5 实验评估

我们采用 TREC (http://trec.nist.gov) 的数据来测试算法。TREC 是 Text REtrieval Conference 的缩写，有关内容将在第五章中有详细的描述。共进行了两种类型的实验：

(1) 依据给定的查询主题，返回相关的句子；
   TREC 2002, 50 个查询主题, 56412 个待选的句子。
(2) 依据给定的查询主题，返回相关的文本；
   TREC 6-8, 150 个查询主题, 528155 个待选的文本。

3.5.1 整体的准确率

我们采用基于语义空间模型的主题词扩展算法（TSTM），扩展主题词，然后用

tf-idf 算法进行信息检索。同时，我们实现了其他的一些主题词扩展算法作为对比：(a)无扩展（BaseLine）；(b) 基于 WordNet 的扩展[11]；(c)基于 LCM 模型的扩展[23]；(d) 基于 SPLSI 模型的扩展[18]（参见 2.4 节）。

所有上述统计算法模型的训练语料都是 TREC 语料库中的 528155 个文本。

(1) 依据给定的查询主题，返回相关的句子
设置判别函数（见本章 3.4 节）的阈值为 cv=0.1，overlay percent=30%。语义树的参数为 m=20 ，TSTM 的层数 v=5。获得的结果如下图 3.5 所示。

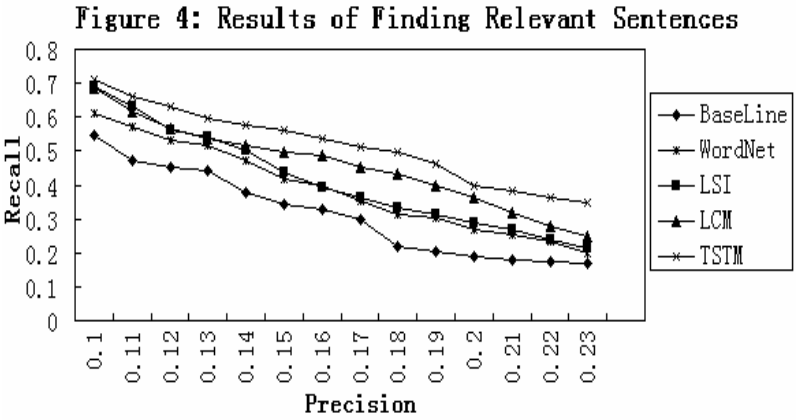


图 3.5 句子检索的对比实验结果

上图显示的是准确率和召回率曲线，从中可以看到准确率很难增长。主题词扩展在这类任务中是非常有效的，因为和文本相比，句子中的词数很少，特别需要这种主题词扩展的技术。显然，基于语义树模型的扩展方法表现是最好的，而基于隐含语义模型的效果甚至比 LCM 模型更差，主要是因为 LSI 引入了太多了噪声。

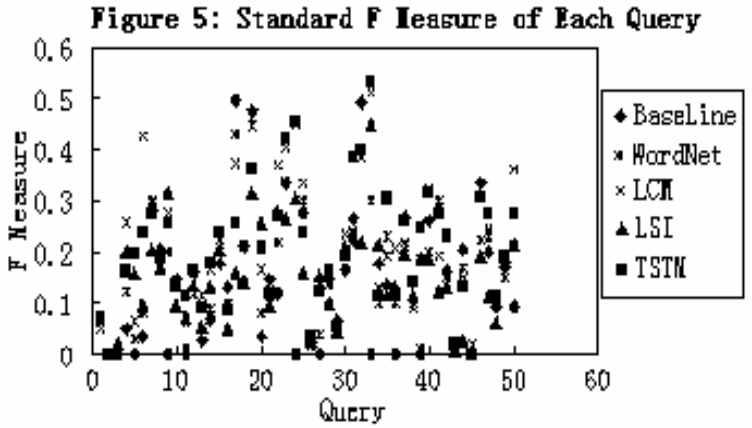


图 3.6 每一个主题的表现对比

图 3.6 显示了每一个主题的表现。可以看到，在大多数的情况下，基于语义树模型的扩展比其他算法获得更好的结果。但也有一些主题，LCM 模型的结果最好，这主要是因为 TSTM 模型仍然会引入少量的噪声。而 LSI 模型的表现几乎是最差的，这充分说明，在主题词扩展的应用中，需要非常小心的避免引入噪声。

(2) 依据给定的查询主题，返回相关的文本
设置判别函数（见本章 3.4 节）的阈值为 cv=0.1，overlay percent=75%。语义树的参数为 m=15 ，TSTM 的层数为 v=5。获得的结果如下表 3.7 所示。这里，我们只使用了 TREC 主题中的 Description 字段。

数据	算法	准确率	召回率
TREC 6	BaseLine	0.1582	0.1905
	WordNet	0.1534	0.2554
	LCM	0.1651	0.3117
	LSI	0.1510	0.3312
	TSTM	0.1773	0.3557
TREC 7	BaseLine	0.1726	0.1675
	WordNet	0.1667	0.2413
	LCM	0.1828	0.3616
	LSI	0.1674	0.3914
	TSTM	0.1865	0.3907
TREC 8	BaseLine	0.1853	0.2916
	WordNet	0.1815	0.3579
	LCM	0.2302	0.4717
	LSI	0.2017	0.4514
	TSTM	0.2430	0.4984

表 3.7 文本检索的对比实验结果

从上表可以看出，在实现了所有模型中，TSTM 模型在进行文本检索的主题词扩展中是最有效的。

3.5.2 TSTM 模型的参数分析

(a) TSTM 模型的层数 v 和叶节点数 m
TSTM 模型的层数 v 和叶节点数 m 显然会影响最终的结果。下图 3.8 显示了在句子检索的任务中，标准 F 值随 v 和 m 变化的情况。

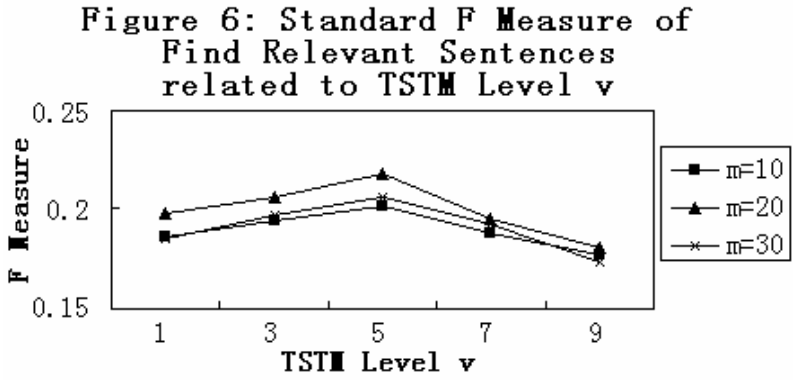


图 3.8 标准 F 值随参数 v 和 m 变化的情况

参数 v 和 m 的值越大，显示了扩展的程度越大。图 3.8 说明了，我们必须要优化

这两个参数，尽力避免过度的扩展。过度扩展甚至还不如不扩展。

(b) TSTM 模型的覆盖度

TSTM 模型的核心思想之一就是覆盖度的约束，它的阈值是用 *percent* 来表达的。如果 *percent=0*，意味着没有覆盖度约束。下图 3.9 显示了检索准确率随覆盖度阈值 *percent* 变化的情况。

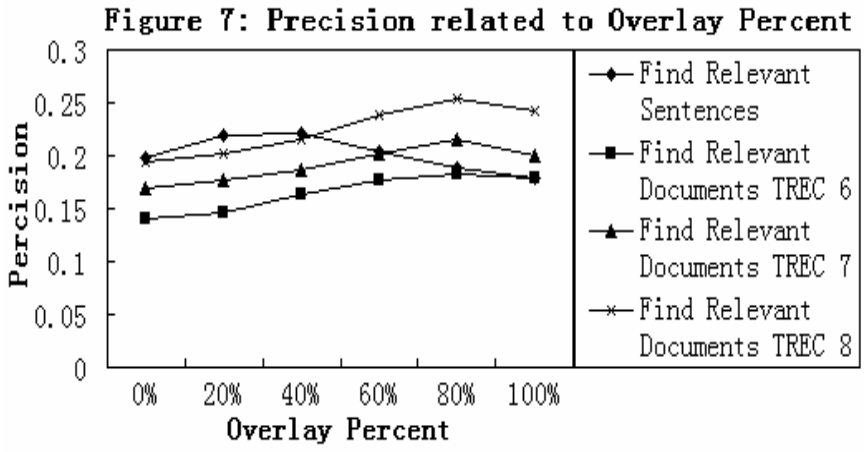


图 3.9 TSTM 模型覆盖度参数分析

考虑到句子检索使用的是长查询主题（包括 Title，Description 和 Narrative 三个字段），而文本检索使用的是短查询主题（仅使用 Description 字段），所以前者需要一个更小的覆盖度值。这是因为在短查询主题中，所有的主题词都很重要，只有被这些主题词都覆盖的词才是合理的扩展词。但是，对长查询主题来说，就不需要这么严格的覆盖度。

本节的实验评估证明了 TSTM 模型在主题词扩展中的有效性。

3.6 本章小结

本章从隐含语义标引系列模型的不足之处说起，把语义空间模型从图状的结构简化为树状结构，提出了完整的语义树模型。和其他语义空间模型相比，语义树模型主要有空间时间复杂度小，算法灵活可控等优势。

本章 3.4 节基于语义树的模型，提出了一种新的信息检索主题词扩展算法，称为 TSTM 模型。其核心是两个判别函数，分别对应语义上的相似度和覆盖度约束。3.5 节进行了实验评估，证明了在主题词扩展的应用领域，TSTM 模型比 LCM 和 LSI 等模型更有优势。

进一步考虑，我们会发现，语义树模型还有很多可以挖掘的地方。比如可以根据根节点词的不同特性，设置不固定的 v 和 m 参数值，使得语义树更加的灵活。另外，TSTM 主题词扩展模型还可以和相关反馈等技术结合起来，进一步提高检索的准确率。

第四章 基于语义张量模型的信息检索

4.1 信息检索中的矢量和张量

在第一章中曾提到，信息检索有两个核心技术：构建标引和计算相似度。第二、三章提出的两种语义空间模型，都有明确的物理意义，在数学上都是完备的，并在信息检索领域都找到具体的应用。然而，仔细的观察会发现，无论是隐含语义标引系列模型，还是语义树模型，都是面向构建标引这部分应用的。通俗一些来说，就是通过修正查询的关键词来提高检索的准确性。而对于计算相似度这部分，还是沿用了比较传统的矢量内积方法。本章将要提出的语义张量模型，试图在计算相似度的方面有所突破。

在信息检索中无法避免的一个过程就是计算查询与文本之间的相似度。其中使用最为广泛的技术就是矢量内积，也就是把查询主题词和文本中出现的词分别构造成词矢量，然后将两者相乘（内积），从而获得查询与文本之间的相似度。通过在词矢量中引入不同的因素，这种技术有很多个模型，如：tf[7]，tf-idf[8]，Length normalization[12]，BM25[27]等等。这类模型无论外在的形式如何变化，但有一个问题是无法回避的：矢量内积模型将词看成是孤立的元素。每个词在词矢量中占有一维，这意味着词与词之间是孤立的，相互之间没有关联。这一点是不合理的，因为词的简单堆砌（word bag）并不能明确的表达含义；而只有词的某种规则排列才能构成语义。矢量内积的方法显然忽略了这种结构信息。

那么，如何才能在计算相似度的过程中，保留这种结构信息呢。由于矢量是一维的，显然没有足够的空间来容纳这些信息。所以，一个自然的想法就是，将表达查询或者文本的一维矢量，扩充为高维的张量；利用附加的维度来表达词与词之间的结构信息。比如，当用户查询为：“ **World Cup 1998** ”时，这三个词独立的看都不能表达一个明确的语义，而只有当这三个词按照某种规则排列的时候才能体现含义。矢量(World,Cup,1998)显然不能体现这种结构信息，而张量可以，参见图 4.1。

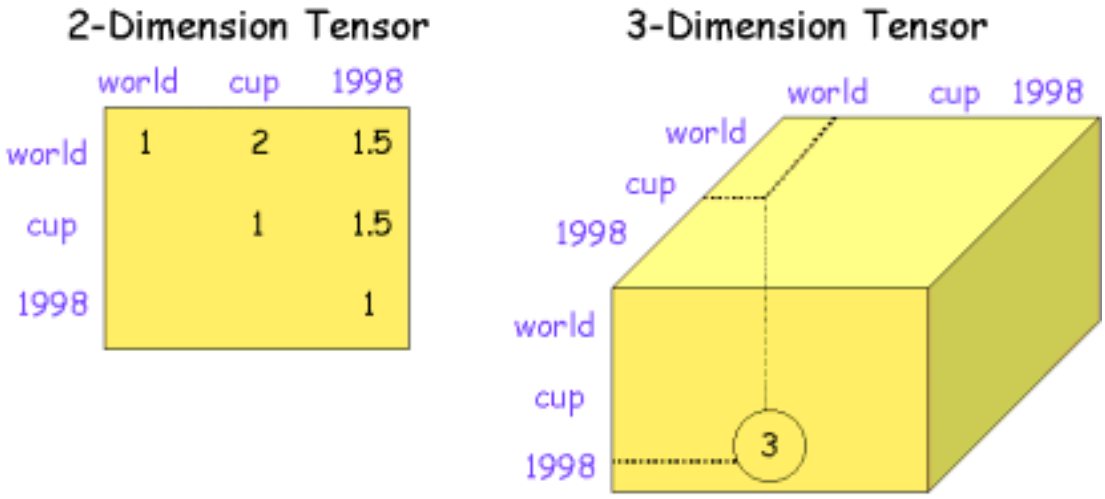


图 4.1 二维和三维张量的表达形式

上图中，二维张量是一个矩阵，每个元素表达的含义是，相应的两个词形成某种结构时，应给出的权值。这种结构就是上面提到的某种规则排列，可以是两个词在查询或者文本中的共现次数，也可以是两个词在查询或者文本中的距离，等等。类似的，三维张量是一个立方体矩阵，每个元素表达的含义是，相应的三个词形成某种结构时，应给出的权值。如果共有 N 个查询主题词，那么一个 N 维的张量可以完全表述这 N 个词所有可能出现的结构（规则排列）。

拿上面的例子来说，如果词“ World ”和“ Cup ”在一个文本中相距很远，那么不太可能是表达“ 世界杯 ”的含义。在使用矢量模型的计算中，无法避免给这个不相关文档以很高的相似度值，因为毕竟两个词在文本中都出现了。但是，在上述的二维张量中，我们可以规定，只有当“ World ”和“ Cup ”两个词按顺序连续出现（一种规则排列的特例）时，才能给很高的相似度值；如果两者距离很远，则不认为相关。这样，采用张量模型的优势就体现出来了，这个文本就会被判别为和主题不相关。

总之，用一句话概括张量模型的思想就是：在计算相似度时，不仅考虑主题词是否出现，还需考虑主题词如何出现（或者说以何种结构在文本中出现）。

在信息检索中，语义张量的优势是明显的，但是如何实现它却是个很大的挑战。虽然，在概念、逻辑和物理意义上，都已经非常明确了，但要想在具体的应用中实现张量的概念，还有两个问题需要解决：

（1） 当给出用户查询的时候，如何自动的构造一个张量；

（2） 如何进行张量之间的计算。

其中，第二个问题是比较容易解决的，数学上已经有了明确的定义。但是，第一个问题似乎非常困难。能够自动地构造张量，是实现这个概念的先决条件，但是如何才能知道主题词之间哪些规则排列是能够表达语义的呢？举上面的例子来说，我们知道“ World ”和“ Cup ”两个词按顺序连续出现，是代表世界杯的语义，

应该有比较高的权值；而“Cup”和“1998”两个词一起出现，显然并不代表什么明确的语义。人来区分是容易的，但是计算机如何区分呢？显然，要做到这一点需要对原始的查询真正达到语义上的理解，很难。

正是因为在构造张量上的困难，我们试图绕过这个问题，从信息检索的实践出发，采用一种变通的方法将张量概念模型化。从而使其能够达到应用的标准，这正是下一节将要表述的重点。张量概念的本身是很抽象的，可以用各种各样不同的模型去实现这个概念；而实现这个概念的本质，就是要定义一种词的规则排列模型。也就是说，要决定在哪些词的哪些排列出现时，认为是表达了语义的，应该给予高权值。

4.2 张量概念的模型化

本节将从信息检索的实践出发，将张量的概念模型化，也就是定义一种词的规则排列模型。

考虑以下的一个用户查询与两篇文档（A,B），其中文档中带下划线的词为查询主题词。

用户查询：

“**Can radio waves from radio towers or car phones affect brain cancer occurrence?**”

文档 A: “John claimed his brain cancer was caused by the wave from his cellular phone. That claim, put forth in a lawsuit, has no basis in accepted scientific fact.”

文档 B: “I was listening to the radio, when the tower collapsed. I ran several blocks before my brain kicked in, and saw that another wave of people started running towards a police car.”

（1）

我们非常清楚的知道，文档 A 和用户查询是相关的，而文档 B 和用户查询不相关。但如果采用矢量模型（如 tf,tf-idf 等）来计算相似度，会发现文档 B 和查询的相似程度超过了文档 A，文档 B 看起来是更相关的。

为什么会出现这种情况呢？虽然文档 B 包含了更多的主题词，但是这些词在文档 B 中是分散出现的，而一个独立的词很难表达一个明确的含义。通常，查询主题是由一组词组成的，整体上表达一个含义。例如上面例子中的“phone”和“brain cancer”，如果他们在文档的同一个句子中出现，该文档就很有可能和查询主题是相关的（如文档 A）。但如果他们在文档中分散出现，则很可能和查询主题不相关（如文档 B）。因此，我们把张量概念模型化的第一个核心思想表述如下：

Query words appearing closely in the document provide more

contributions to the similarity value than the ones appearing separately. The closer the query words in a document, the larger the similarity value between the query and the document.

查询主题词在文档中出现的越接近，应该给出越大的相似度值。

(2)

在传统的 tf-idf 方法中，不同的主题词根据倒排文档频率（inverse document frequency）给出不同的权值。因此，一些常用词就会被认为是不太重要的，权值比较小。这在通常的情况下，是合理的；但在某些情况下就不合理。在上面的例子中，“ radio wave ”和“ brain cancer ”显然比其他的词更重要，因为它们表达了这个查询的主要含义，然而这些常用词往往 idf 值很小。所以，我们需要改进主题词权值的确定方法。很自然想到的就是，一些命名实体和名词短语在查询中往往是比较重要的。基于这一点，张量概念模型化的第二个核心思想表述如下：

Some query words, like named entities and baseNP are called “Core Words”, while the other words are called “Surrounding Words”. “Core Words” are much more important than “Surrounding Words”, and should have special status in the retrieval processing (i.e. having larger weights).

某些查询主题词，如命名实体和一些名词短语，被称为“ 核心词 ”；其他的词被称为“ 环境词 ”。“ 核心词 ”在查询中起到了主要的作用，应该给予特殊的待遇（如：给比较大的权值）。

有了这两个核心思想，张量概念在某种程度上就明确化了。当然，张量概念还可以有其他各种表述方式，这里仅仅是面向信息检索应用的一种特定的表述。在模型化以后，离实用还有一个步骤，就是要用数学公式将上面两个思想体现出来，这正是下一节的主要内容。

4.3 张量模型的数学表述 – 窗口系列算法

基于上面提到的两个核心思想，我们提出了“ 窗口系列算法 ”，应用于信息检索。窗口系列算法，从简单到复杂，共有三个数学模型：“ 简单窗口模型 ”，“ 动态窗口模型 ”，“ 核心窗口模型 ”。在前两个模型中实现了第一个核心思想，在第三个模型中实现了第二个核心思想。

4.3.1 数学模型一： 简单窗口模型 (Simple Window-based Model)

就象第一个核心思想表述的那样，查询主题词在文档中出现的越接近，就应该有越大的权值。因此，我们引入了“ 窗口 ”的概念。如果，查询主题词在一定宽度的窗口（若干个词组成）中共现，就给出相对较大的权值。

首先，将文档中的所有词按顺序排列，如图 4.2。每一个长度为 d 的窗口，包含

了连续 d 个词组成的子串。

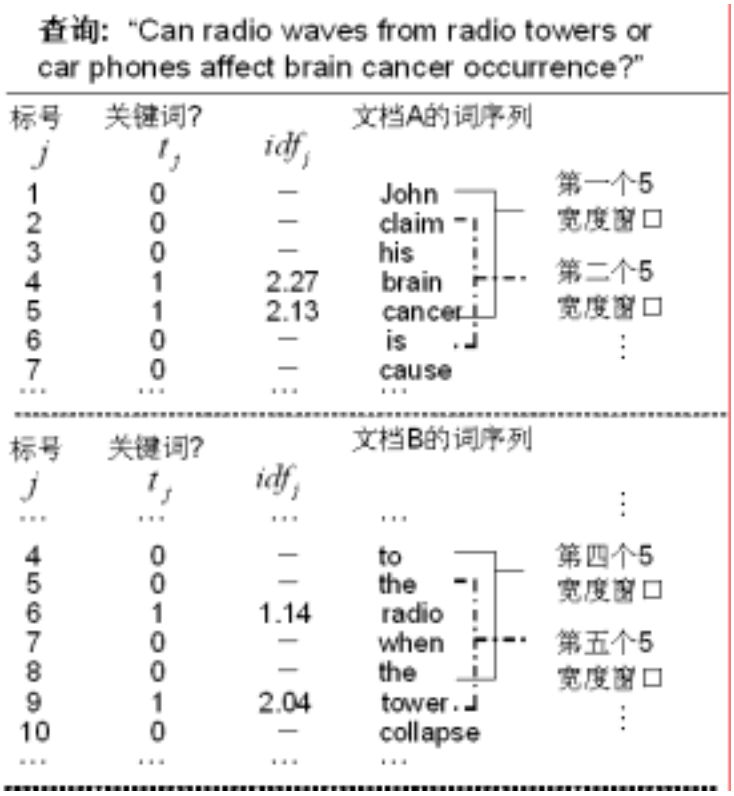


图 4.2 简单窗口模型

设文档的词序列，共有 N 个词；窗口宽度为 d。定义查询 Q 和文本 D 之间的相似度($Sim1(Q,D)$)为：

$$Sim1(Q,D)=\sum_{i=1}^{N-d} SWin\,(i,i+d)$$

$$SWin(i,i+d)=[\sum_{j=i}^{i+d} t_j * idf_j]*[\sum_{j=i}^{i+d} t_j]$$

其中，$SWin(i,i+d)$ 代表用户查询与文档的一个 d 宽度窗口之间的相似度值；这个窗口位于文档的第 i 个词到低(i+d)个词。t_j 是一个二元信号，如果文档词序列中的第 j 个词是查询主题词，则 t_j 为 1；否则为 0。idf_j 是文档中第 j 个词的倒排文档频率。

查询与文档之间最终的相似度，是查询和所有窗口间的相似度之和。查询和某个窗口之间的相似度 $SWin(i,i+d)$ 有两部分组成。第一部分 $[\sum_{j=i}^{i+d} t_j * idf_j]$ 和传统的

tf-idf 方法是相同的；第二部分$[\sum_{j=i}^{i+d} t_j]$给那些包含多个主题词的窗口，提供额外的权值。如果某个窗口含有 3 个查询主题词，那么这个窗口最终的权值就是传统 tf-idf 方法所获得权值的 3 倍。在窗口中的主题词越多，代表主题词之间越接近，所以最终的权值越大。

考虑 4.2 节中提到过的例子，一些主题词的idf_j值在表 4.3 中列出，是从大规模文本库中训练得到的。

Word	brain	cancer	radio	tower
idf	2.27	2.13	1.14	2.04
Word	phone	wave	Car	
idf	1.75	1.74	1.25	

表 4.3 一些主题词的 idf 值

如果采用传统的 tf-idf 方法，可以计算得到两个文档 A 和 B 各自和用户查询的相似度：

$$Sim(query, documentA) = 6.15$$

$$Sim(query, documentB) = 8.44$$

这意味着文档 B 与查询主题更相关。

而如果采用简单窗口模型（宽度为 5），得到的结果是：

$$Sim1(query, documentA) = 46.08$$

$$Sim1(query, documentB) = 43.56$$

这意味着文档 A 与查询主题更相关。虽然文档 A 中出现的主题词比较少，但是它们相互之间比较接近，因此文档 A 与查询有相对较大的相似度值。这和人的判断是相似的，因此简单窗口模型比 tf-idf 更合理一些。后面 4.4 节的大规模实验支持了这个结论。

4.3.2 数学模型二： 动态窗口模型 （Dynamic Window-based Model）

在简单窗口模型中，如果窗口中包含多个查询主题词，就给出相应比较大的权值。但是并没有考虑主题词在窗口中的分布情况。查询主题词在窗口中可以是连续出现，也可以是分散出现；如果在窗口中连续出现，就很有可能形成一个词组。一般来说，词组所表达的含义是明确具体的；而词有相对比较多的歧义。所以，我们应该给这种主题词连续出现的窗口，再附加格外的权值。

另外，在实际的应用中，很难去定量地确定简单窗口模型中窗口的宽度，因为对于不同类型的查询，最优的窗口宽度变化是比较大的。

为了解决上面的两个问题，在第一个模型的基础上，提出了动态窗口模型。在这个模型中，定义了一个“窗口紧密度”（TightWin）来改进静态的窗口宽度。

定义：窗口紧密度是指，能够包含原始窗口中所有查询主题词的最小子窗口宽度。

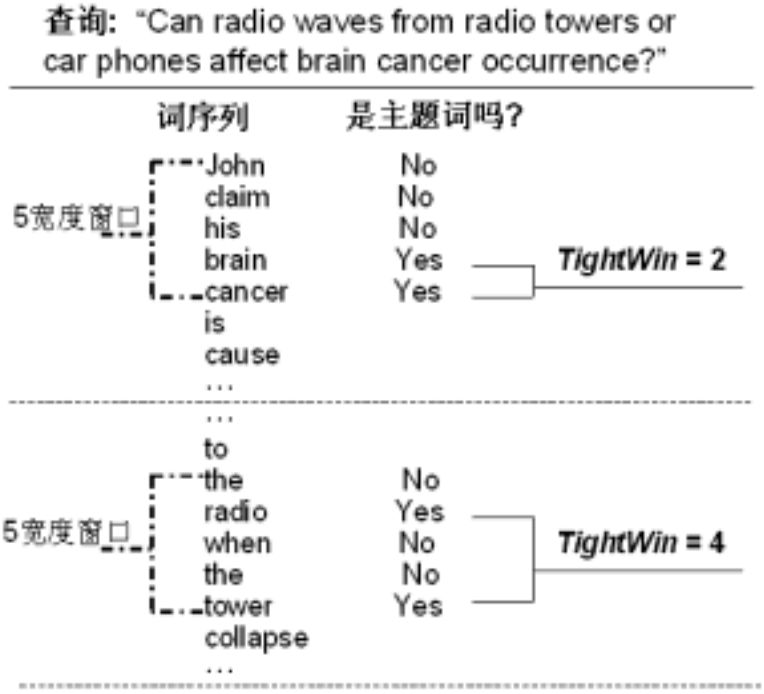


图 4.4 窗口紧密度（TightWin）定义

在图 4.4 中，给出了窗口紧密度的几个例子。如果主题词在窗口中分散出现，TightWin 的值就大，反之则小。所以，当 TightWin 值小的时候，应该给窗口更大的权值。没有对上图进行解释，别人看不清楚。

设 N 为整个文档词串的长度，d 为窗口的宽度。动态窗口模型的查询 Q 与文档 D 之间相似度值 $Sim2(Q,D)$ 定义为：

$$Sim\ 2(Q,D) = \sum_{i=1}^{N-d} DWin\ (i, i+d)$$

$$DWin(i,i+d) = SWin(i,i+d) * (\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p$$

$$= [\sum_{j=i}^{i+d} t_j * idf_j] * [\sum_{j=i}^{i+d} t_j] * (\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p$$

其中，$DWin(i,i+d)$ 代表用户查询与文档的一个 d 宽度窗口之间的相似度值；这个窗口位于文档的第 i 个词到低(i+d)个词。t_j 是一个二元信号，如果文档词序列中的第 j 个词是查询主题词，则 t_j 为 1；否则为 0。idf_j 是文档中第 j 个词的倒排文档频率。

TightWin 是上面定义的窗口紧密度；p 是大于零的一个参数。

和简单窗口模型对比，动态窗口模型的公式里多了一项：$(\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p$，正是这项改进了静态的窗口宽度。举个例子来说，当三个查询主题词在窗口中连续出现，这项的值为 $(\frac{3}{3})^p = 1$；如果这三个词在宽度为 5 的窗口中分散出现，则这项为 $(\frac{3}{5})^p = 0.6^p$，始终是小于 1 的。所以，窗口中连续出现的主题词，会给这个窗口带来更多的相似度值。

还是用上面的例子来说，采用宽度为 5 的窗口，设参数 p=1.0，动态窗口模型得到的结果如下：

$$Sim2(query, documentA) = 46.08$$

$$Sim2(query, documentB) = 37.20$$

两个值之间有了一定的差距，说明文档 A 比文档 B 更接近于查询主题。这是合理的。

总的来说，在这两个模型中，相似度值不仅决定于查询主题词在文本中是否出现，还决定于查询主题词在文本中的分布情况。其中，模型一描述了主题词在窗口距离下，大尺度上的分布情况；模型二进一步加入了主题词在窗口中的分布模型。因此，这两个模型实现了我们的第一个核心思想，既：查询主题词在文档中出现的越接近，应该给出越大的相似度值。

4.3.3 数学模型三： 核心窗口模型 （Core Window-based Model）

在上面两个模型中，查询主题词在文档中出现的越接近，给出越大的相似度值。这在某些情况下，可能会带来问题。考虑以下这个查询：

“Can radio waves from radio towers or car phones affect brain cancer occurrence?”

如果主题词“radio waves”和“brain cancer”在文本中出现的很接近，这个文本就很可能是相关的。但如果是“car phone”和“affect”出现的很接近，就不能说明什么问题。

因此，基于我们的第二个核心思想，将查询的主题词分为两类，定义如下。

定义:

(1) 代表查询主要含义的主题词,如一些命名实体和名词短语,被称为 “核心词（Core Words）”；
(2) 一个查询主题词，如果它不是核心词，则称为“环境词（Surrounding Word）”；
(3) 一个窗口如果包含核心词，则该窗口称为“活跃窗口（Active Window）”；否则称为“非活跃窗口（Non-Active Window）”。

显然，核心词比环境词更重要一些，所以活跃窗口应该有比较大的权值。

设 N 为整个文档词串的长度，d 为窗口的宽度。动态窗口模型的查询 Q 与文档 D 之间相似度值 $Sim3(Q,D)$ 定义为：

$$Sim3(Q,D)=\sum_{i=1}^{N-d} CWin(i,i+d)$$

$$CWin(i,i+d)=DWin(i,i+d)*[\sum_{j=i}^{i+d} t_j^*]$$

$$=[\sum_{j=i}^{i+d} t_j * idf_j]*[\sum_{j=i}^{i+d} t_j]*(\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p *[\sum_{j=i}^{i+d} t_j^*]^m$$

其中，$CWin(i,i+d)$ 代表用户查询与文档的一个 d 宽度窗口之间的相似度值；这个窗口位于文档的第 i 个词到低(i+d)个词。t_j 是一个二元信号，如果文档词序列中的第 j 个词是查询主题词，则 t_j 为 1；否则为 0。idf_j 是文档中第 j 个词的倒排文档频率。t_j^* 是另一个二元信号，如果文档词序列中的第 j 个词是核心词，

则 t_j^* 为 1；否则为 0。*TightWin* 是窗口紧密度；p 和 m 是大于零的参数。

和模型二相比，核心窗口模型有一个额外的项 $[\sum_{j=i}^{i+d} t_j^*]^m$，用来表述核心词的思想。举个例子来说，如果 3 个核心词在某个窗口中出现，这个窗口就是活跃窗口，这个额外项的值为 3^m。而当窗口中没有核心词出现时，这个窗口就是非活跃窗口，这项为 0。因此，只有活跃窗口才对最后的相似度值有贡献，窗口中的核心词越多，贡献越大。

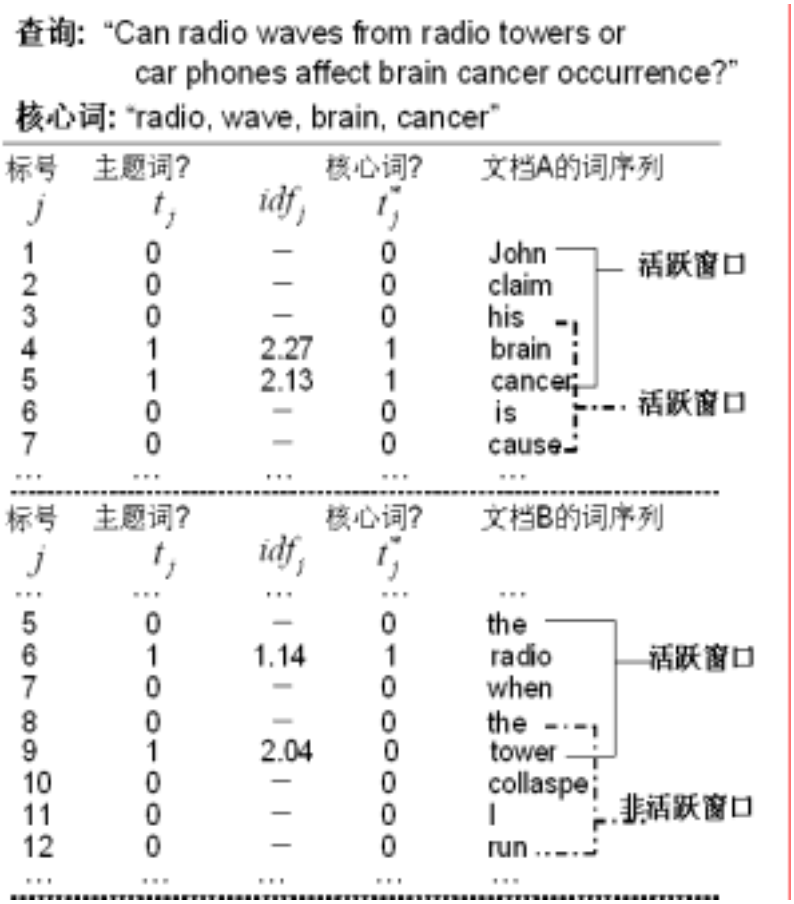


图 4.5 核心窗口模型

在上图 4.5 中，给出了一些核心词和活跃窗口的例子。其中“radio wave”和“brain cancer”被设定为核心词。采用核心窗口模型，可以计算得到相似度如下（参数 p=m=1.0）：

$$Sim3(query, documentA) = 72.53$$

$$Sim3(query, documentB) = 18.48$$

比较这两个值，发现文档 A 的相似度大大高于文档 B。这是合理的，因为事实上，

文档 A 和主题相关，而文档 B 不相关。

这里还有一个问题没有解决，就是如何确定核心词。虽然在定义上提到命名实体和一些名词短语，但目前命名实体识别和基本名词短语的识别还不能做到百分百准确，并且识别过程需要一些时间。因此，确定的核心词也并不都正确的，只能说尽量少的引入噪声。在我们系统中的实际做法是：

(1) 英语

(a) 首字母大写的词一般认为是命名实体，定义为核心词；

(b) 时间和数量词定义为核心词；

(c) 采用 Eric Brill 的 tagging 工具(http://www.cs.jhu.edu/~brill/)对查询标注词性，连续出现的名词认为构成名词短语，也作为核心词。由于查询的句子一般不很复杂，因此出现的错误不会很多。

(2) 中文

(a) 进行命名实体识别，只把概率大的人名、地名、机构名，定义为核心词；

(b) 时间和数量词定义为核心词；

(c) 采用我们自己的分词标注工具对查询标注词性，连续出现的名词认为构成名词短语，也作为核心词。由于查询的句子一般不很复杂，因此出现的错误不会很多。

至此，我们的第二个核心思想，也在核心窗口模型中被实现了。窗口系列的三个模型从简单到复杂，其优于 tf-idf 模型的根本原因在于考虑了查询主题词之间的关系。我们通过一种变通的方法，把张量概念模型化和实用化了，在下一节的实现评估中，将证明这种方法的有效性。

4.4 实验评估

依然采用 TREC (http://trec.nist.gov) 数据来测试算法，其中包括：
(1) 包含 528155 个 XML 格式文件的文本库；
(2) 200 个查询主题(TREC topic 301-450 and 601-650)。

我们仅采用 TREC topic 中的 description 字段，称为短查询。

4.4.1 总体准确率和召回率

采用这 200 个查询主题进行文本检索，实现了窗口系列的三个算法；并用 tf-idf [7][8]和 BM25[27](k1=1.2, b=0.75, k3=1000)算法作为对比。

设定窗口宽度为 5，参数值为 p=m=1.0。模型三中核心词的确定，采用了上节描述的三个步骤的方法。总体的准确率，参见下图 4.6。

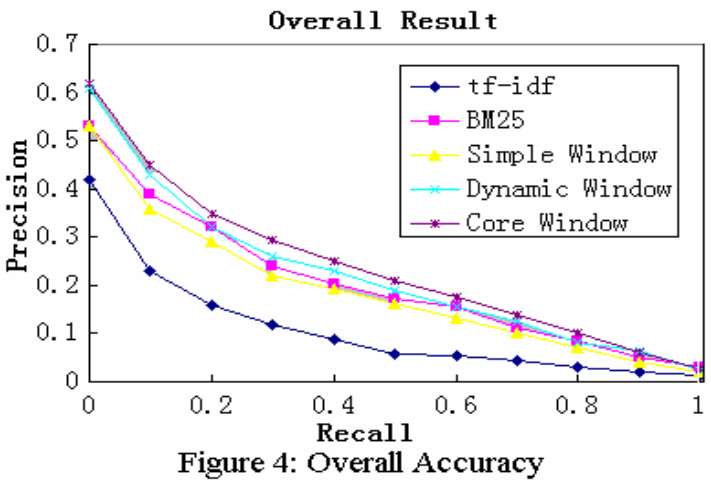


图 4.6 总体准确率-召回率曲线

从上面的结果可以看出，窗口系列模型表现好于 tf-idf 模型，并且核心窗口模型是其中最出色的。

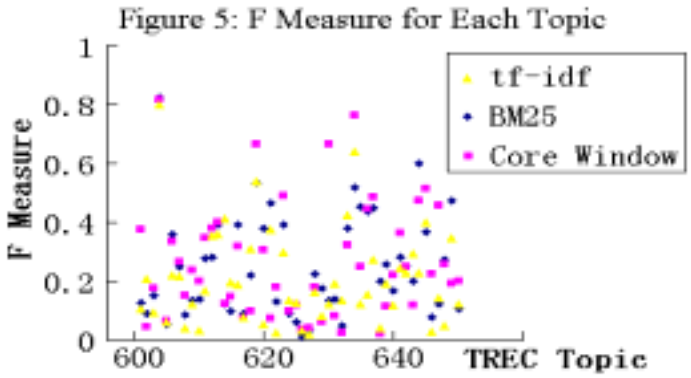


图 4.7 每个主题的表现（标准 F 值）

图 4.7 显示的是每个查询的表现（TREC topic 601-650）。在大多数情况下，核心窗口模型都是表现最好的，但也有一些例外。进一步的研究表明，这些例外多是由于在选择核心词时不合理造成的。某些部重要的词被选了进来，而忽略了一些真正重要的词。如果用人工调整核心词的选择，那么大多数的这类例外就不会发生。这充分证明了窗口系列模型本身的有效性和对不同查询的鲁棒性。

4.4.2 窗口宽度分析

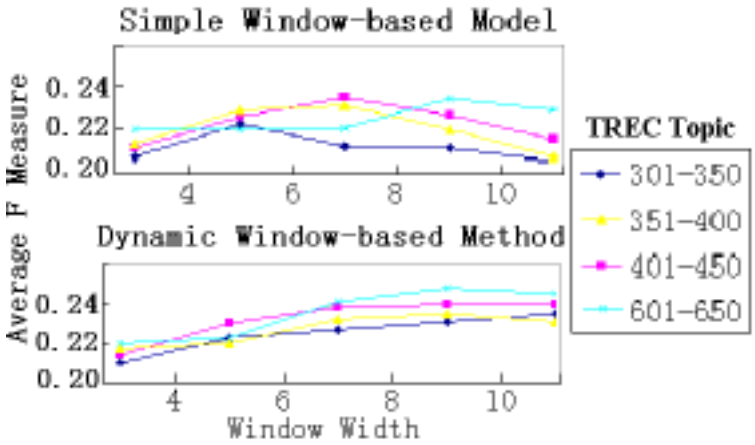


Figure 6: Average F Measure with Window Width

图 4.8 不同窗口宽度下的平均 F 值

上图显示了在不同窗口宽度下，简单窗口模型和动态窗口模型的表现。其中，每一条曲线代表了 50 个查询的平均 F 值。在上半部分的图中，可以发现很难给简单窗口模型选择一个最优的窗口宽度，因为它对不同的查询主题太敏感。在下半部分图中，所有的曲线有类似的形状和顶点位置，说明动态窗口模型比简单窗口模型有更好的鲁棒性。当窗口宽度大于 7 时，表现相当的稳定，这对一个真实系统很重要，很容易确定一个动态窗口的最优宽度。此外，当把窗口宽度设为 2 或者 3 时，窗口系列模型就非常类似于信息检索中二元标引（bi-gram indexing）或者三元标引（tri-gram indexing）模型[32]。

4.4.3 动态窗口模型参数分析

在动态窗口模型中，有一个大于零的参数 p。如果 p=0，那么模型二就简化成模型一了。下图 4.9 显示了在不同窗口宽度和不同 p 值下，动态窗口模型的平均 F 值。

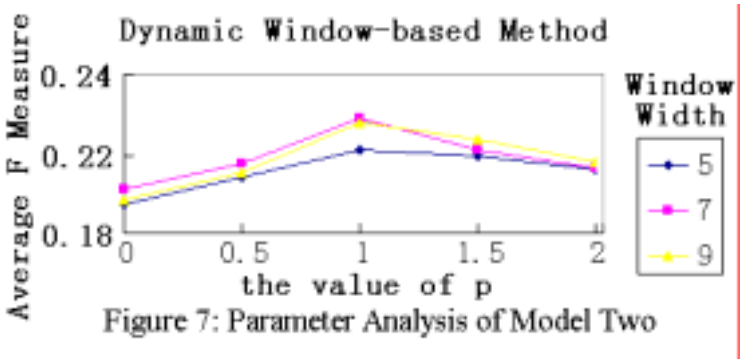


Figure 7: Parameter Analysis of Model Two

图 4.9 不同窗口宽度和 p 值下动态窗口模型的表现

上图中的所有曲线形状都很类似，顶点出现的位置也相同，这说明了在动态窗口

模型中很容易就能确定参数 p 的最优值。

4.4.4 核心窗口模型参数分析

在核心窗口模型中，有两个参数 p 和 m 对最终的结果有影响。我们试图最优化这两个参数。下面的表 4.10 显示不同的 p 和 m 值下，200 个查询的平均 F 值。

p \ *m*	0.5	1.0	1.5	2.0
0.5	0.247	0.251	**0.263**	0.248
1.0	0.252	**0.270**	0.254	0.241
1.5	**0.265**	0.260	0.254	0.242
2.0	0.240	0.241	0.237	0.236

表 4.10 在不同的 p 和 m 值下核心窗口模型的表现

所获得的结果，似乎暗示着当满足 p+m=2.0 时，检索系统可以获得最优的结果，这也是一个经验公式。因此，在核心窗口模型中，也可以很容易的确定最优化的参数 p 和 m。

4.5 本章小结

在本章中，首先探讨了矢量检索模型的不足，提出了张量检索的概念，并在理论上说明其正确性。然后，通过两个核心思想，把张量概念明确化，使其具有了可操作性。在本章的第三节，提出了数学化的窗口系列模型，从简单到复杂分别是：简单窗口模型、动态窗口模型、核心词窗口模型；并简述了实现过程。在第四节中，用较大规模的实验，验证了窗口系列模型的有效性。

窗口系列模型还有比较大的扩展空间，比如可以精确的定义查询主题词在文档中的距离等等；也可以和前面提到过的主题词扩展，相关反馈等技术结合起来，进一步提高信息检索的准确率。

第五章 信息检索系统与评测

5.1 NLPR 信息检索系统

基于第三章和第四章提到的技术，我们构建了一个 NLPR 信息检索系统，其中包含了常用的计算语言学算法，以及完整的文本检索系统构架。下面的部分将详细描述这个系统，并利用 TREC 评测来评估其性能。

5.1.1 NLPR 信息检索系统的框架

NLPR 信息检索系统的整体框架，如下图 5.1 所示。

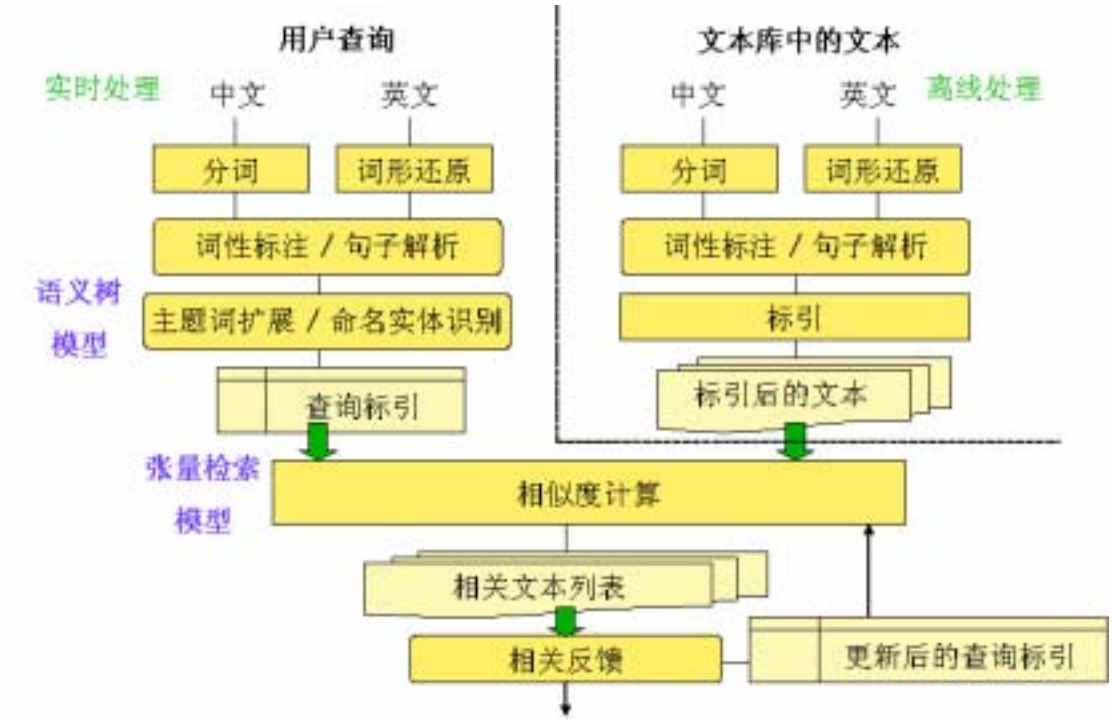


图 5.1 NLPR 信息检索系统框架

整个系统从功能上来说，分为三个部分：离线部分（Off-line）、在线部分（On-line）、语言资源支持部分。

(1) 离线部分是上图的右上角，主要是为文本库中的文本建立索引。对中文和英文来说，不同的主要是预处理部分。中文需要先分词，然后进行词性标注；英文需要先词性标注，然后进行词形还原（stemming）。最后在统一的框架下建立索引。此外，如果原始文本是某种标记格式的，（如 TREC 数据多是 XML 格式文本），还需要首先进行 XML 解析。

(2) 上图其余部分都属于检索的在线处理部分。首先，还是对中英文进行不同的预处理；采用语义树模型进行主题词扩展。然后，用张量检索模型进行文本检索。如果需要的话，获得第一次的结果后，还可以用相关反馈技术修正查询。

(3) 语言资源部分没有在图 5.1 中表示出来，这部分主要是为上述的两个主要功能提供语言资源。包括：词共现的概率统计、停用词过滤、数据库调用、XML 解析等等。

采用这个 NLPR 信息检索系统，可以完成各种算法下的文本检索、句子检索、相似度计算等功能。这三个大的模块由于其调用的功能有很大一部分是重叠的，所以在系统实现的过程中，设计了 12 个面向功能小模块，每个小模块完成一些特定的任务。它们组合起来就构成了整个系统。下一节将对这 12 个小模块做详细的描述。

5.1.2 NLPR 信息检索系统的设计

NLPR 信息检索系统由 12 个相互独立，相互调用的小模块组成，都是用标准 C 或者标准 C++语言编写的。以下将分别介绍：

(1) 中文分词与词性标注模块

- 功能：对输入的中文文本进行分词和词性标注
- 源文件：StateMachine.h StateMachine.cpp
  Source.h Source.cpp
- 模块实现的描述：
  模块的两个大部分，分别采用两个 C++类来实现。
  1) StateMachine: 分词和词性标注二层状态机的实现
  2) Source: 处理分词要用到的词典，词频等信息的存取，以及训练

  这两个类的相互关系是：在训练过程中只用到 Source 类；在分词和词性标注过程中用到 StateMachine 类（当需要资源时，向 Source 类请求）。

  StateMachine 类的实现
  分词系统处理的三种数据类型：
  A) 中文串： “ 分词词性标注系统 ”
  B) 数字串： “ 1,000 1,234.56 一二·九 ”
  C) 全角/半角标点，英文字母：“ ,。 ？ ”
  这样，有利于处理数字以及标点，而且程序结构比较清晰。具体实现时，我们采用二层状态机的结构。

  初始化状态变量为 0，进入状态机，当缓冲区读完时，自动退出。标* 的表示有二层状态机。其中状态 2，3，4 比较类似，在下图 5.2 中合并表示。

  二层状态机说明：
  A) state 2,3,4 -> state 1: 处理数字连接标点的情况，如 1,000 1,234.56
  B) state 2,3,4 <-> state 0: 处理数字和中文字联合组词的情况，

如 “ 第一，一次 ”

C) state 0 -> state 0: 这部分完成中文串分词的主要工作，采用概率模型来消歧

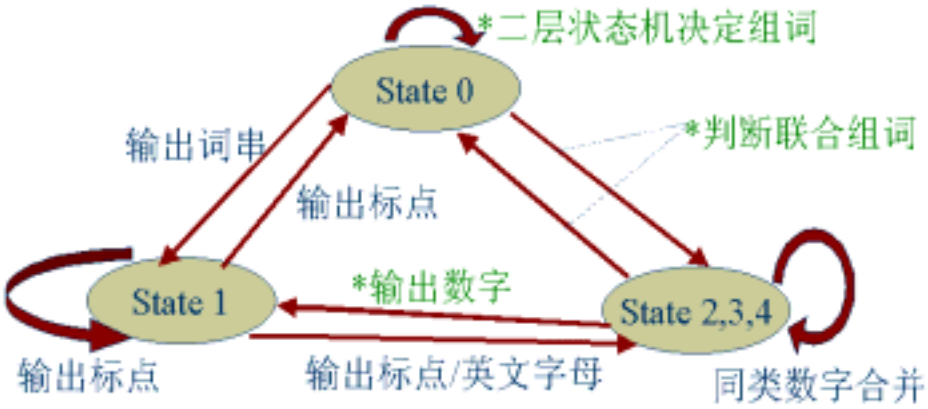


图 5.2 中文分词系统的状态机结构

其中系统的核心是 State 0->State 0 的二层状态机，采用动态规划的方法选择最优的分词路径。分词完成后，再使用三元概率模型，通过动态规划算法标注词性。

Source 类的实现方法

Source 类主要完成的工作是，根据语料库训练统计模型的参数，并给 StateMachine 类提供数据的支持。实际上是完成一个资源的收集，储存，发布的功能。

Source 类的资源组织形式如下：

下面图 5.3 的三部分分别是：按内码排列的中文字表，按内码排列的中文词表，每一个词有一个词性链表。其中第一部分主要是用作检索的标引，提高检索的效率；第二部分存储着最主要的词频等信息，第三部分是储存词性的频度信息。三部分的具体数据结构如下：

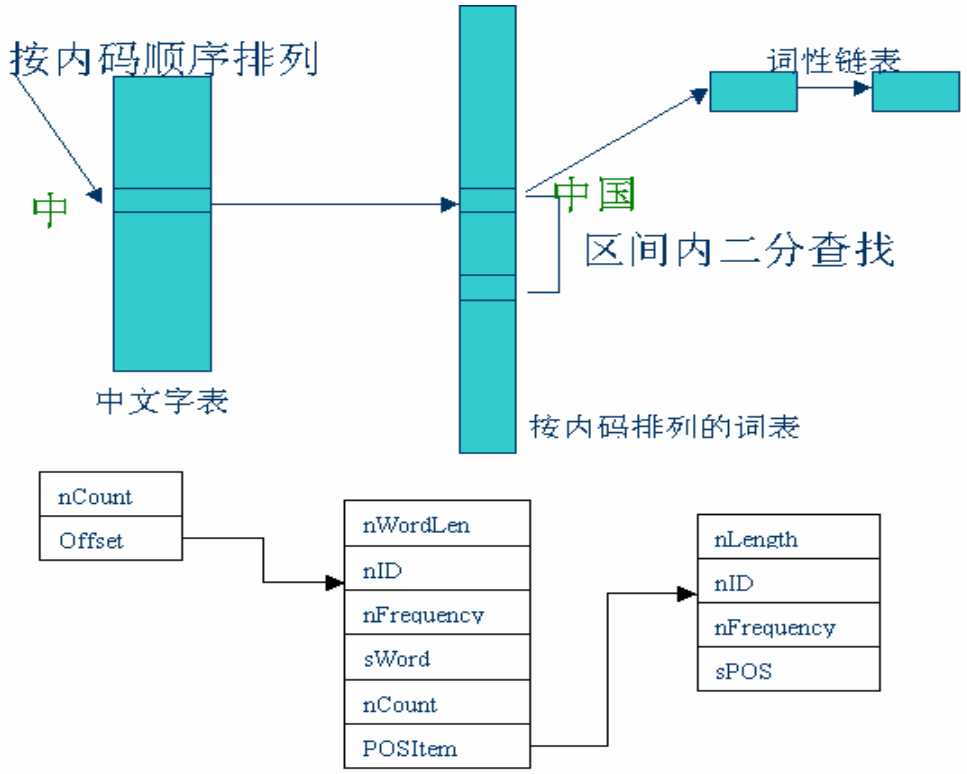


图 5.3 资源的组织结构

中文分词与词性标注模块，训练语料采用人民日报 1998 年 1 月-6 月的数据，词性标注集采用北大标准。测试结果如下：

测试种类	分词准确率	词性标注准确率
封闭测试	96.87%	92.42%
开放测试	95.89%	91.26%

表 5.4 中文分词与词性标注准确率

(2) 英文词性标注模块

- 功能：对文本中的每个英文词标注词性
- 源文件：Erik Brill 的免费软件(http://www.cs.jhu.edu/~brill/)
- 模块实现描述：
  输入是原始的英文文本，输出是标注完成的文本。

(3) 英文词形还原（Stemming）

- 功能：对输入的英文单词，进行还原操作，去掉后缀等等。
- 源文件：Stemming.h Stemming.c
- 模块实现描述：
  以 B. Frakes, C. Cox, C.Fox 的 Stem 程序[33]为基础，对规则进行一些修正与改进,并加入了不规则词的还原。主要目标是将英文词后的复数后缀 s,动词后缀 ed, ing 等等去掉，把名词，动词等还原为标准形式。这项工作是必要的，因为在关键词匹配的时候，如果不进行词形还原，

会造成同一个动词，由于时态不同而匹配不上等多种不合理的情况。整个 Stemming 的过程，是将原英文词按顺序经过 4 组规则的过滤：

（1） 不规则动词和名词的词形还原

这部分是通过查找一个对应词表来实现的，这个词表共有词对 4418 个。以下是部分的例子（表 5.5）：

原词	词形还原后的词
aardwolves	aardwolf
abaci	abacus
abetted	abet
abetting	abet
abhorred	abhor
abhorring	abhor
abode	abide
aboideaux	aboideau
aboiteaux	aboiteau
abought	aby
… …	… …

（2） 名词复数还原规则（表 5.6，‘ — ’表示删除，下同）

原词的结尾部分	修改后
sses	ss
ies	i
ss	ss
s	--

（3） 动词还原规则（表 5.7）

原词的结尾部分	修改后
eed	ee
ed	--
ing	--
at	ate
bl	ble
iz	ize
bb	b
dd	d
ff	f
gg	g
mm	m
nn	n
pp	p
rr	r
tt	t
ww	w

xx	x
y	i

(4) 形容词和名词后缀还原规则（表 5.8）

原词的结尾部分	修改后
ational	ate
tional	tion
enci	ence
anci	ance
izer	ize
abli	able
alli	al
entli	ent
eli	e
ousli	ous
ization	ize
ation	ate
ator	ate
alism	al
iveness	ive
fulnes	ful
ousness	ous
aliti	al
iviti	ive
biliti	ble
icate	ic
ative	--
alize	al
iciti	ic
ical	ic
ful	--
ness	--
Al	--
ance	--
ence	--
er	--
ic	--
able	--
ible	--
ant	--
ement	--
ment	--
ent	--

sion	s
tion	t
ou	--
ism	--
ate	--
iti	--
ous	--
ive	--
ize	--
e	--
ll	l

(4) XML 解析模块

- 功能：对 XML 文本进行解析
- 源文件：LibXML 免费库(http://www.xmlsoft.org)
- 模块实现描述：
  输入是原始的 XML 格式文本，输出是解析成树状结构的内容。

(5) 构建文本索引

- 功能：为文本库中每个文本建立索引
- 源文件：trec_robust.h trec_robust.c
- 模块实现描述：
  利用 Mysql 数据库，实现根据词来索引文档的功能。

(6) 查询主题词扩展

- 功能：利用第三章构造的语义树模型来扩展查询主题词
- 源文件：trec_robust.h trec_robust.c
  trec_novelty.h trec_novelty.c
- 模块实现描述：
  根据第三章表述的过程，利用输入的原始查询主题词，构造一棵语义树。然后，根据两个约束规则，来决定哪些词被扩展。由于前文已经详细描述了这个过程，所以不再赘述。

(7) 计算相似度

- 功能：实现了查询主题与文档（或句子）之间各种相似度的计算方法
- 源文件：trec_robust.h trec_robust.c
  trec_novelty.h trec_novelty.c
- 模块实现描述：
  实现的相似度计算方法包括：
  (1) 矢量模型 tf-idf[7][8]；
  (2) 矢量模型 tf-idf 长度归一化[12]；
  (3) 矢量模型 BM25[27]；
  (4) 张量检索模型，三个窗口系列算法。
  这部分的算法，在第四章中已经详细描述了，不再赘述。

(8) 相关反馈

- 功能：取一些和主题相关的文本，作为训练语料，来修正查询主题词
- 源文件：trec_robust.h trec_robust.c
  trec_novelty.h trec_novelty.c
- 模块实现描述：
  首先，利用信息检索系统，获得第一次的检索结果。得到，按照和原查询的相似度大小排序的一些文本集或句子集。假设排在前面的若干个文本（一般 10 到 100 个）和查询主题是非常相关的，把它们作为训练语料，去修正查询主题词。同理，假设排在最后的若干个文本和查询最不相关，也把它们作为反面的训练语料，修正主题词。最后，用修正过的主题词，再一次进行信息检索。这就是相关反馈（Relevant Feedback）[16]的基本思想。

  在系统中基本按照上述的过程来进行相关反馈。首先，获得第一次的检索结果后，选取相似度最高的若干篇文本（可选参数），统计这些文本中各个词的出现情况，如果一个词在这些文本中出现的概率大于某一数值（如 70%，可选参数）时，就把这个词作为额外的主题词，加到查询主题词列表中去，其权值一般给基本值 1。

(9) 停用词（stoplist）过滤

- 功能：将英文中不表达含义的冠词、介词等过滤掉。
- 源文件：stopper.c stop.h stop.c strlist.h strlist.cpp
- 模块实现描述：
  英文中有一些冠词、介词等不表达含义的常用词，需要把它们从查询的关键词表中除去；否则会影响检索的准确率。这个模块，主要用了一些索引技术，使其能够快速的查找停用词表，从而判断一个词是否属于停用词。总共要经过两轮的过滤：
  （1） 缩写指代词过滤
  如：aren't can't couldn't 等等，共有 54 个。
  （2） 普通停用词过滤
  如：no of the only often for 等等，共有 472 个。

(10) 词的概率统计

- 功能：统计词在文本库的所有文本中，出现或者共现的次数
- 源文件：trec_robust.h trec_robust.c
- 模块实现描述：
  前面提到的各种算法，有很多需要一些统计数据的支持。如 tf-idf 算法，需要预先统计各个词的 idf 值；而构造语义树的时候需要预先知道，词与词之间的共现频率。这个模块就是预先统计了这些信息，供其他模块需要时调用。具体来说，就是统计了以下两种信息：
  （1） 统计词在文本库中的倒排文档频率 idf，并存放在数据库中，格式为：
  “序号” “词” “该词出现在多少篇文本中”

(2) 统计词在文本库中的与其他词一起出现的次数，并存放在数据库中，格式为：
“序号” “词 1” “词 2” “两个词一起出现在多少篇文本中”

(11) 数据库 Mysql 调用

- 功能：封装了 Mysql 数据库的常用功能调用
- 源文件：mysql_c.h mysql_c.c mysql.h
- 模块实现描述：
  前面提到词的概率统计中，很多数据是存放在 Mysql 数据库里的，并且要求实时能获得这些数据。因此，利用 Mysql 本身提供的一些 C API(http://dev.mysql.com/doc/mysql/en/C.html)，我们封装了其主要读写功能，使得用户能够很方便的通过 C 语言函数来进行数据库操作，同时避免了一些格式错误的发生。具体来说，主要包括以下的功能：
  (1) 数据库的建立，打开，关闭等操作；
  (2) 数据表的建立，索引，查询，添加记录，删除记录等操作；
  (3) 返回数据集的顺序读取操作；
  (4) 任意 SQL 语句操作等等。

(12) 平台定义与基本算法资源

- 功能：定义并封装了一些基本字符串与常用算法函数
- 源文件：define.h PlatDef.h NLP_DB.h NLP_DB.c
- 模块实现描述：
  (1) 某些常用的字符串函数，在 Window 和 Linux 平台上有各自不同的函数界面；因此通过宏定义，将它们统一了起来。
  (2) 某些排序算法，标点符号判别等自然语言处理的常用功能函数，由于经常需要被调用，因此统一实现在这个模块里。

下一节，将介绍系统的源文件构成；以及这 12 个小模块中，主要的 API 函数的实现。

5.1.3 NLPR 信息检索系统的具体实现

系统的源文件共有 26 个，不包括额外调用的 LibXML 库和 MySQL 库。共可分为三类：

(a) 头文件 12 个。包含平台定义，类定义，函数调用定义等等。
define.h PlatDef.h libxml.h
NLP_DB.h mysql_c.h Source.h
StateMachine.h Stemming.h stop.h
strlist.h trec_novelty.h trec_robust.h

(b) C/C++源文件 11 个。包含系统实现的主要代码，以及系统功能调用的实例。
NLP_DB.c mysql_c.c Source.cpp
StateMachine.cpp stemming.c stop.c
stopper.c strlist.c trec_novelty.c

trec_robust.c main.c

(c) 语言资源文件 3 个。包含停用词表，不规则动词和名词转换表。

input.txt output.txt stop.wrd

此外，还通过 Mysql 的 c 接口，调用了 Mysql 数据库中的一些资源表，主要包含词在文本库中的共现频率等信息。

下面，将详细介绍一些主要模块的 API 函数接口，及调用方法。

(1) 表 5.9

函数定义	void Segmentation_POS_file(char* in, char* out)
函数功能	对输入的文本文件进行分词与词性标注
源文件	StateMachine.cpp
输入	char* in: 输入文本文件名
输出	char* out: 输出文本文件名
备注	标注集采用北大标准；所有类型的文本均可以处理，对未登陆字符串一律标注为 “ /nx ”。

(2) 表 5.10

函数定义	void Segmentation_POS_dir(char* in, char* out)
函数功能	对目录中的文件进行分词与词性标注
源文件	StateMachine.cpp
输入	char* in: 输入文件存放的目录
输出	char* out: 输出文本存放的目录
备注	标注集采用北大标准；所有类型的文本均可以处理，对未登陆字符串一律标注为 “ /nx ”。若输出目录不存在，则自动建立。

(3) 表 5.11

函数定义	int stop(const char* word)
函数功能	判断一个词是否是停用词
源文件	stopper.c
输入	char* word: 待判断的词
输出	返回值: 输入词是停用词返回 1， 输入词非停用词返回 0， 内部错误返回-1。
备注	出现内部错误时，向标准输出报告错误原因。

(4) 表 5.12

函数定义	int irregular(char** in, char** out, int n,char *s)
函数功能	不规则动词词形还原 不规则名词复数词形还原

源文件	stemming.c
输入	char** in: 不规则动词和名词输入词表 char** out: 不规则动词和名词还原后的词表 int n: 上述表格的长度 char* s: 待还原的词
输出	char* s: 还原后的词 返回值: 正常返回 1， 内部错误返回 0。
备注	如果输入词在不规则词表中不存在，则不做任何修改即返回。出现内部错误时，向标准输出报告错误原因。

(5) 表 5.13

函数定义	int stem(char* word)
函数功能	规则的词形还原
源文件	stemming.c
输入	char* word: 待还原的词
输出	char* word: 还原后的词 返回值: 正常返回 1， 内部错误返回 0。
备注	如果输入词不需要还原，则不做任何修改即返回。出现内部错误时，向标准输出报告错误原因。

(6) 表 5.14

函数定义	int db_init(MYSQL &conn, char* DB_HOST, char * DB_NAME,char * DB_USER,char* DB_PASS)
函数功能	打开一个 Mysql 数据库
源文件	mysql_c.c
输入	MYSQL &conn: 一个 Mysql 连接实例 char* DB_HOST: Mysql 服务器名 char* DB_NAME: 需打开的数据库名 char* DB_USER: Mysql 连接用户名 char* DB_PASS: Mysql 连接密码
输出	MYSQL &conn: 建立的 Mysql 连接 返回值: 正常返回 0， 错误返回-1。
备注	数据库不存在，或者用户名密码错，均返回-1。

(7) 表 5.15

函数定义	int db_close(MYSQL &conn)
函数功能	关闭一个 Mysql 数据库连接
源文件	mysql_c.c
输入	MYSQL &conn: 一个 Mysql 连接实例
输出	返回值: 正常返回 0，

	错误返回-1。
备注	数据库不存在，或者不能正常关闭，均返回-1。

(8) 表 5.16

函数定义	int db_exec(MYSQL &conn, MYSQL_RES &res, char *fmt, ...)
函数功能	执行一个标准 SQL 语句
源文件	mysql_c.c
输入	MYSQL &conn: 一个 Mysql 连接实例 MYSQL_RES &res: 一个 Mysql 结果集实例 char* fmt, …: 待执行的 SQL 语句
输出	MYSQL_RES &res: 如果执行的 SQL 语句是 select， 则返回结果集；否则返回 NULL。 返回值: 正常返回 0， 错误返回-1。
备注	数据库不存在，或者不能正常访问，均返回-1。

(9) 表 5.17

函数定义	double getWeightIDF_mysql(char* word)
函数功能	获得一个词的倒排文档频率（idf 值）
源文件	mysql_c.c
输入	char* word: 输入的词
输出	返回值: 返回输入词的倒排文档频率， 内部错误返回-1。
备注	如果输入词是未登陆词，则返回默认的倒排文档频率 1.0。

(10) 表 5.18

函数定义	int build_idf_mysql(MYSQL &conn,char* table, char* word,int docnum)
函数功能	建立倒排文档频率（idf 值）标引
源文件	mysql_c.c
输入	MYSQL &conn: 打开的 Mysql 数据库连接 char* table: Mysql 数据表名 char* word: 输入的词 int docnum: 包含该词的文档标号
输出	返回值: 正常返回 0， 内部错误返回-1。
备注	数据库不存在，或者不能正常访问，均返回-1。

(11) 表 5.19

函数定义	int ExpansionQuery_STM(MYSQL &conn, int* swc, char** words,double* weight, int n, int e_no)

函数功能	基于语义数模型，扩展主题词
源文件	trec_novelty.c
输入	MYSQL &conn：打开的 Mysql 连接 int* swc：原始的主题词数 char** words: 原始主题词表 double* weight：原始主题词权值 int n: 主题词表容量 int e_no：需要扩展的主题词数
输出	int* swc：扩展后的主题词数 char** words: 扩展后的主题词表 double* weight：扩展后主题词权值 返回值: 正常返回 0， 内部错误返回-1。
备注	特别注意，要给 words 和 weight 分配足够的内存，使其能够存的下扩展后的关键词表。数据库连接错误返回-1。

（12）表 5.20

函数定义	int do_relevant_feedback(char*** res, int* res_n, int topic_n, double** score, char*** topicwords, int* twc, double** topicweight, int tn, double frate)
函数功能	实现相关反馈
源文件	trec_novelty.c
输入	char*** res: 检索结果集（相关文本的词表） int topic_n：结果集中的主题数 int* res_n：每个主题的相关文本数 double** score：每个相关文本的相似度 char*** topicwords：每个查询的主题词表 int tn: 主题词表长度 int* twc：每个查询的主题词数 double** topicweight：每个查询的主题词权值 double frate：相关反馈度
输出	char*** topicwords：相关反馈后的主题词表 int* twc： 相关反馈后的主题词数 double** topicweight：相关反馈后的主题词权值 返回值: 正常返回 0， 内部错误返回-1。
备注	特别注意，要给 topicwords 和 topicweight 分配足够的内存，使其能够存的下扩展后的关键词表。数据库连接错误返回-1。

（13）表 5.21

函数定义	double similarityTFIDF(MYSQL &conn, char** words,double* weight,int swc, char** topicwords, double* topicweight, int twc)
函数功能	用 tf-idf 方法计算相似度
源文件	trec_novelty.c
输入	MYSQL &conn: 打开的 Mysql 连接 char** words：文本词表 double* weight：文本词权值（tf） int swc：文本词个数 char** topicwords：查询主题词表 double* topicweight:查询主题词权值 int twc：查询主题词个数
输出	返回值: 正常返回计算得到的文本和查询之间的相似度值，内部错误返回-1。
备注	数据库连接错误返回-1。

（14）表 5.22

函数定义	double similarityWindow(MYSQL &conn, char** words,double* weight,int swc, char** topicwords, double* topicweight, int twc int width, int MODEL, double m, double p)
函数功能	用窗口系列算法计算相似度
源文件	trec_robust.c
输入	MYSQL &conn: 打开的 Mysql 连接 char** words：文本词表 double* weight：文本词权值（tf） int swc：文本词个数 char** topicwords：查询主题词表 double* topicweight:查询主题词权值 int twc：查询主题词个数 int width: 窗口模型中的窗口宽度 int MODEL: 特定的窗口模型 1 - 简单窗口模型 2 – 动态窗口模型 3 – 核心窗口模型 double m/double p: 窗口模型参数
输出	返回值: 正常返回计算得到的文本和查询之间的相似度值，内部错误返回-1。
备注	数据库连接错误返回-1。

(15) 表 5.23

函数定义	int buildIDF(MYSQL &conn, char *dir,char *filelist)
函数功能	建立语料库的倒排文档(idf)标引
源文件	trec_robust.c
输入	MYSQL &conn: 打开的 Mysql 连接 char* dir：语料库文件存放的目录 char *filelist：语料库文件列表
输出	返回值: 正常返回 0， 内部错误返回-1。
备注	数据库连接错误返回-1。

(16) 表 5.24

函数定义	int ParseXMLDoc(char *filelist, char *dir,char* dest)
函数功能	解析 XML 文件，获得纯文本
源文件	trec_robust.c
输入	char *filelist： XML 文件列表 char *dir：XML 文件存放的目录 char* dest： 输出文件的目录
输出	char* dest：输出的纯文本文件，文件名和原文件相同，扩展名改为“ .txt ”。 返回值: 正常返回 0， 内部错误返回-1。
备注	解析出 XML 文件所有字段中的文本，按顺序排列。

(17) 表 5.25

函数定义	int relevant(char* in, char* out) int novelty(char* in, char* out)
函数功能	TREC 2003 Novelty 句子检索任务的实现
源文件	trec_novelty.c
输入	char *in： 所有输入文件存放的目录 char *dir：所有输出文件存放的目录
输出	返回值: 正常返回 0， 内部错误返回-1。
备注	具体的输入输出文件格式，参见 TREC 网站: http://trec.nist.gov

(18) 表 5.26

函数定义	int relevant_eva(char* in, char* out) int novelty_eva(char* in, char* out)
函数功能	TREC 2003 Novelty 评测的实现

源文件	trec_novelty.c
输入	char *in： 所有输入文件存放的目录 char *dir：所有输出文件存放的目录
输出	返回值：正常返回 0， 内部错误返回-1。
备注	具体的输入输出文件格式，参见 TREC 网站： http://trec.nist.gov

（19）表 5.27

函数定义	int robutst(char* in, char* out)
函数功能	TREC 2003 Robust 检索任务的实现
源文件	trec_robust.c
输入	char *in： 所有输入文件存放的目录 char *dir：所有输出文件存放的目录
输出	返回值：正常返回 0， 内部错误返回-1。
备注	具体的输入输出文件格式，参见 TREC 网站： http://trec.nist.gov

（20）表 5.28

函数定义	int robutst_eva(char* in, char* out)
函数功能	TREC 2003 Robust 评测的实现
源文件	trec_robust.c
输入	char *in： 所有输入文件存放的目录 char *dir：所有输出文件存放的目录
输出	返回值：正常返回 0， 内部错误返回-1。
备注	具体的输入输出文件格式，参见 TREC 网站： http://trec.nist.gov

5.2 信息检索的基本评测指标

文本信息检索系统的功能是为用户服务，根据查询来找到相关的信息，可以文本，段落，句子等等。最终是用户来评价一个检索系统的优劣，这一方面需要大量的人力资源；另一方面由于人所具有的主观性，不可避免的会有偏差。因此，在开发系统的过程中，设计了一系列的评测指标，来相对公平的优化和检验系统性能。一个评价指标，往往都是针对某一类的应用来设计的，有它自身所关注的重点。所以，对于一个检索系统来说，采用哪一个评价指标来优化，就与其应用场合密切相关。

下面列出的是这个领域普遍认可的一些评测指标，并且都在 TREC 评测中使用。

(1) 准确率（Precision)。
这可以说是检索系统最常见的评价指标，其定义为：

$$\text{准确率 } P = \frac{\text{系统返回的正确文本数}}{\text{系统返回的全部文本数}}$$

也就是用户在结果集中找到有用信息的概率。这个指标多用于冗余库的检索，如互联网。在这类检索中，只要找到相关信息即可，并不要求把所有的信息都找到；因为有大量重复或者类似的信息存在。

(2) 召回率（Recall)。
相关信息被返回的概率：

$$\text{召回率 } R = \frac{\text{系统返回的正确文本数}}{\text{系统中全部的正确文本数}}$$

多应用于需要把信息查全的情况。

(3) F 值。
这是准确率 P 和召回率 R 的一个折衷，也就是既要求返回的文本是相关的，又要求尽可能找到所有相关信息。其定义为：

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

其中 P 和 R 分别为准确率和召回率，β 是参数。大多数系统都采用标准的

F 值，也就是 β=1 的特殊情况，此时 F=2PR/(P+R)。

(4) TOP N 准确率。
大多数用户仅有耐心察看检索系统返回的最前面的 N 项结果，这个 N 一般是 10 到 30。因此，一个好的检索系统必须确保在最相关的结果，排列在最前面。TOP N 准确率这个评价指标，就是测试系统的这方面性能。
此外，如果要应用相关反馈技术，那么系统首次返回结果的前面若干项，就必须是和查询相关的；否则采用相关反馈后，不仅没有帮助，还可能使得二次检索结果更差。因此，一个系统，只有当它的 TOP N 准确率指标达到一定数值以后，才能应用相关反馈的技术。

(5) 最差 x%个查询主题的准确率
如果要关注系统的鲁棒性（Robustness)，也就是要求系统对所有查询都有相对比较好的性能；那么就要采取这个评测指标。如果我们确定了若干个查询（如 100 个)，来评测检索系统的性能。那么，对不同的系统来说，

准确率最低的那些查询是不同的。一个系统对自身最差的 x%（如 10%）个查询主题的性能，可以比较好的反映出这个系统对不同查询的兼容能力。

上述的这些指标，可以从不同的方面评价检索系统的性能。NLPR 的信息检索系统，通过参加 2003 年的 TREC 评测，来相对公正的检测这些指标。

5.3 NLPR 信息检索系统在 TREC 评测中的表现

5.3.1 TREC 评测介绍

TREC 是 Text REtrieval Conference 的简写，每年举办一次，主办者是 NIST (National Institute of Standards and Technology)，其网址是：

http://trec.nist.gov

TREC 评测是完全自愿，免费参加的；其目的在于，通过严格定义文本库、查询输入、结果输出的格式，来比较各个检索系统的性能。每年年初，TREC 会公布本年度的若干项任务，以及每个任务涉及到的文本库和评价指标。到 5 月份左右公布查询主题，每个检索系统独立的从文本库中寻找答案，然后过一至两个月提交统一格式的结果。由于输入输出，文本库都是统一的，并且不允许采用额外的训练语料，因此这个评测能够比较公正的反映出各个检索系统的算法性能。

由于 TREC 评测的相对公正性，国内外许多大学和研究机构都通过参加感兴趣一些任务，来测试自己的检索系统。包括：CMU, MSRA, Google, IBM, 清华大学，复旦大学，中科院计算所等等。TREC 的任务每年都有一些小的变化，但总体上还是比较稳定的；2003 年在文本检索领域主要有以下几项。

(1) Web 检索：检索和主题相关的网页，采用统一的网页库。这里除了用文本检索技术外，还可以使用网页的字段信息，相互之间的链接信息等。
(2) Robust 检索：这个任务是传统文本检索任务的一个延续。主要内容没有变化，还是根据查询在文本库中检索出相关的文本；但是评测指标不再是准确率或者 F 值，而采用了上面 5.2 节提到的第（4）和第（5）个评测指标，主要关注系统的鲁棒性。
(3) Novelty 检索：这项任务分成两个子项。第一项是句子检索，也就是根据查询主题，检索出相关的句子返回，使用户不用察看冗长的文本就能获得所要的信息；由于句子中的信息量比较少，因此这个任务比传统的文本检索要困难一些。第二项是冗余信息过滤，也就是把第一项任务返回的结果集中，过滤除去重复的信息，使得用户能够获得准确而不冗余的信息。
(4) 问答系统评测：输入是自然语言提问，输出是几个词组成的精确回答。除了检索技术外，还需要语法、语义等技术的支撑。
(5) Hard 交互式检索：信息检索系统首先返回一次结果集，然后由用户判断返回的哪些文本是相关的，哪些是不相关的；这些信息将反馈给检索系统，用来进行训练，然后再次给出一个更新后的结果集。这种交互式检索，也是相关反馈的一种方式。

由于中科院自动化所的 NLPR 检索系统，刚刚初步成型，缺乏经验和积累。所以，在 2003 年的 TREC 任务中，我们选择了两个相对比较容易的任务：Robust 检索和 Novelty 检索，分别对应与上面的第二和第三项。下面将分别介绍，NLPR 检索系统在这两项任务中的表现。

5.3.2 TREC-2003 中的鲁棒性全文检索评测

Robust 检索的基本任务就是全文检索。输入是特定的 TREC 查询主题，共有 50 个旧查询（TREC Topic 301-450 中选出）和 50 个新查询（TREC Topic 601-650）组成；文本库由 55 万多篇 XML 格式的文本组成。目的是要从文本库中找出和查询相关的文本。前面已经提到，这个任务的评测指标比较特别，主要关注的是那些性能比较差的查询主题，以此来反映系统的鲁棒性。采用 NLPR 信息检索系统来完成这项任务，其大致的框架在 5.1 节中已经作了详细的说明。针对鲁棒性检索的特殊评测指标，着重考虑了以下两点：

(1) 考虑到表现最差的查询，也需要有个令人满意的准确率。因此需要采用鲁棒性比较好的相似度计算方法。
(2) 虽然规定了最多可以返回 1000 个文本的结果集。但是，正如上面所提到的，多数用户往往只察看排在前面的若干项结果（2-3 个屏幕）。所以，对每个查询来说，系统仅返回排在最前面的几十个文本。

在 5.1 节描述的 NLPR 检索系统框架下，采用了以下四项比较有特色的技术，来完成 Robust 检索的评测。

(A) 基于语义树模型的查询主题词扩展，详见第三章及参考文献[15]；
(B) 采用窗口系列模型计算文本与查询主题之间的相似度，详见第四章及参考文献[13]；
(C) 文本长度归一化，见参考文献[12]；
(D) 动态阈值。由于不同查询的特性很不一样（如：不同得主题词数），因此采用单一固定的阈值显然不是很合理。在这里，我们利用了返回结果集的统计特性来获得动态的阈值。简单来说，就是和查询主题相似度最高的文本作为基准，取这个最大相似度值的某个百分比作为这个查询的动态阈值。

评测得到的 TREC 官方结果如下表：
（更多结果参见 TREC 官方网站 http://trec.nist.gov ）

ID Tag	算法	返回的文本数	平均准确率	前 10 个返回文本均错误的概率
NLPR03vb25	动态窗口模型	25	0.1516	7%
NLPR03vb10	简单窗口模型	10	0.1055	7%

NLPR03w16	核心窗口模型	16	0.1153	10%
NLPR03vb50	动态窗口模型 动态阈值	50	0.1770	7%
NLPR03w49	核心窗口模型 动态阈值	49	0.2434	10%

表 5.29 Robust 评测结果

这里的平均准确率是指，召回率分别在 0% 10% 20% … 90% 100% 时准确率的平均值。从上面的结果可以看到，所得到的平均准确率是很低的。这主要是因为对每个查询，仅返回了几十个文本（而不是 1000 个），因此在召回率比较高的点，准确率都为零，拉低了平均值。这里所获得的结果，可以反映出窗口系列模型的有效性。

5.3.3 TREC-2003 中的新信息检索评测

Novelty 评测的目的是检验系统定位相关信息，以及新信息的能力。输入是 50 个 TREC 查询主题，文本库是和这 50 个主题相关的 1250 篇文本（每个主题 25 篇）；第一个子任务的输出是和查询相关的句子集，第二个子任务的输出是和查询相关并且没有重复信息的句子集。

整体上，还是用 5.1 节的 NLPR 检索系统框架；然后针对这个特定的句子检索任务，整合了一些新的技术，下面将对两个子任务分别说明。

第一个子任务：Relevant 检索

(A) 两阶段的查询主题词扩展。第一阶段，首次检索前，采用的是基于语义树模型的扩展（见第三章）；第二阶段，第一次检索后，采用相关反馈技术来扩展，通常排在前 20%的句子集被用来进行相关反馈。由于句子中包含的词数要远远少于文本，因此这种大范围的主题词扩展是必要的，结果证明也是有效的。

(B) 两种计算相似度的方法。一种是 tf-idf，另一种是窗口系列模型（见第四章）。事实上，窗口系列模型是 N-gram 标引模型扩展。

(C) 句子长度的归一化，采用的是 pivoted document length normalization 技术，详见参考文献[12]。

(D) 动态阈值。由于不同查询的特性很不一样（如：不同得主题词数），因此采用单一固定的阈值显然不是很合理。在这里，我们利用了返回结果集的统计特性来获得动态的阈值。简单来说，就是和查询主题相似度最高的句子作为基准，取这个最大相似度值的某个百分比作为这个查询的动态阈值。

第二个子任务： Novelty (new) 检索

在检索到相关句子集的基础上，第二个任务就是要过滤除去重复的信息。这里定义了一个“新信息度”的概念（New Information Degree – NID），来衡量一个句子和前面的某个句子相比，包含多少新的信息。如果 NID 值比较大，那么这个句子就被保留下来；否则就作为重复的信息而删除。

我们采用了两种方法来定义一个句子相对于它前面某个句子的“新信息度”（NID）：

$$NID\_1 = 1 - \frac{\text{两个句子中都出现的词的 idf 值之和}}{\text{后面的句子中出现的所有词的 idf 值之和}}$$

$$NID\_2 = 1 - \frac{\text{两个句子中匹配上的二元词序列（bi-gram）个数}}{\text{后面的句子中总共的二元词序列个数}}$$

通常，如果 NID 值大于 10%-20%，就认为后面的句子包含有新的信息，将被保留下来。

TREC 2003 中，Novelty 检索共有 4 个 Task。第一个 Task 是没有训练语料的情况下，进行相关检索和新信息检索；第二个 Task 是已知相关句子集的情况下，过滤得到新信息句子集；第三个 Task 是有部分训练语料的情况下，进行相关检索和新信息检索；第四个 Task 是有部分训练语料的情况下，从已知相关句子集，过滤得到新信息句子集。以下的一系列表是这四个 Task 的 TREC 官方评测结果：

缩写含义如下：

Dyn: 动态阈值　　**Sta:** 静态阈值
Win: 核心窗口模型　　**RF:** 相关反馈
Leng: 长度归一化　　**Tf-idf:** tf-idf 相似度计算
NID_1 / NID_2: 新信息度模型　　**QE:** 语义树模型的主题词扩展

Task 1 Table

ID TAG	Algorithms	Relevant Results Average F Measure	Novelty Results Average F Measure
NLPR03n1w1	Dyn-Win-RF	0.510	0.425
NLPR03n1f1	Tf-idf-leng-RF	0.477	0.399
NLPR03n1f2	Tf-idf-leng-RF	0.407	0.349
NLPR03n1w2	Dyn-Win-RF	0.391	0.325
NLPR03n1w3	Dyn-Win-RF	0.330	0.279

表 5.30 Task 1 的评测结果

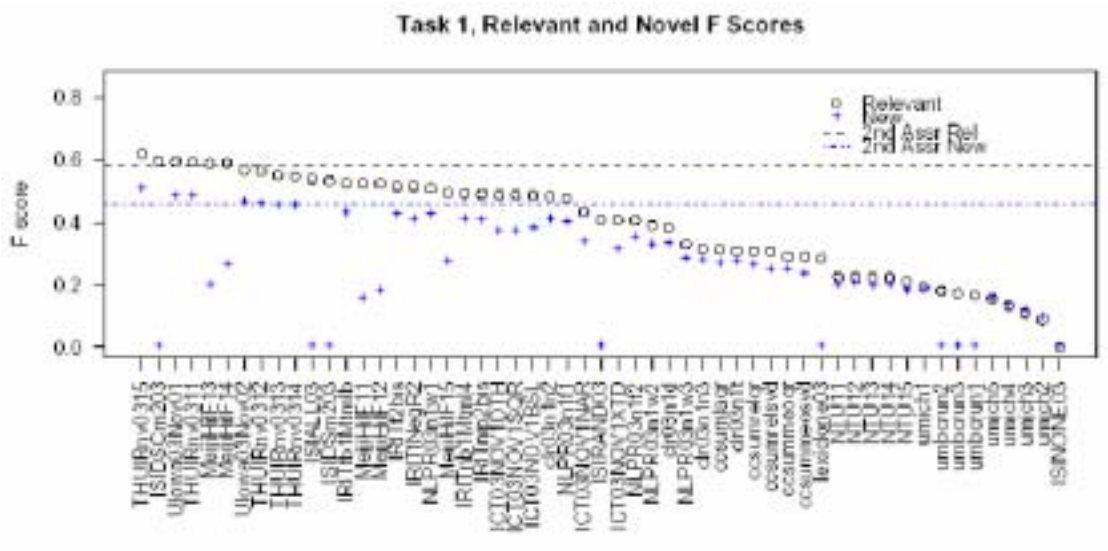


图 5.31 Task 1 的对比评测结果

Task 2 Table

ID TAG	Algorithms	Average F Measure
NLPR03n2d1	NID_1, Dyn	0.807
NLPR03n2s1	NID_1, Sta	0.819
NLPR03n2d2	NID_2, Dyn	0.808
NLPR03n2s2	NID_2, Sta	0.817
NLPR03n2d3	NID_1+2, Dyn	0.803

表 5.32 Task 2 的评测结果

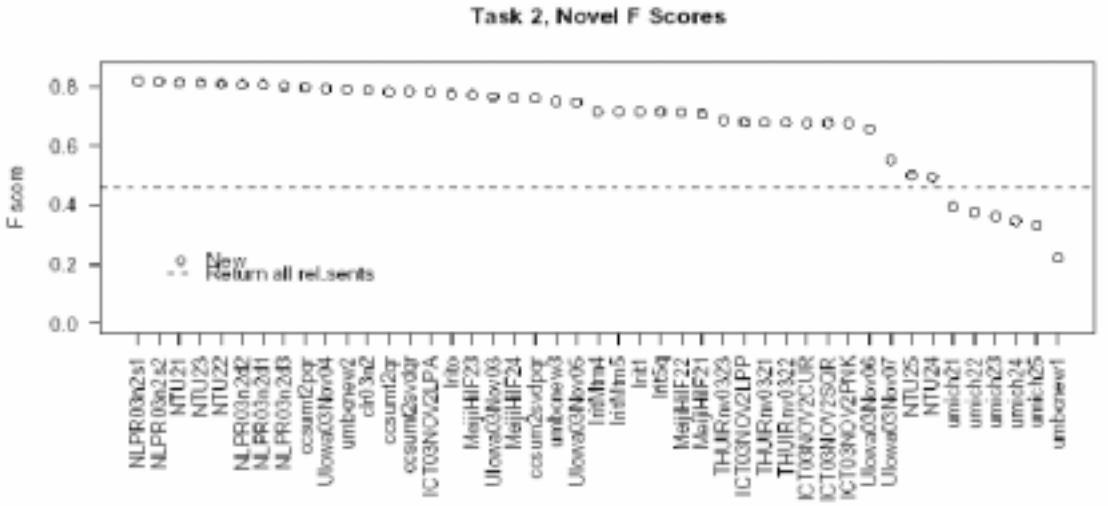


图 5.33 Task 2 的对比评测结果

Task 3 Table

ID TAG	Algorithms	**Relevant** Average F	**Novelty** Average F
NLPR03n3d1	RF, Win, leng, NID_2, Dyn	0.687	0.518
NLPR03n3s1	RF,Win, leng, NID_1, Sta	0.677	0.532
NLPR03n3d3	RF,Win, leng, NID_1, Dyn	0.674	0.509
NLPR03n3d2	QE,RF,tf-idf, leng, NID_2,Dyn	0.618	0.472
NLPR03n3s2	QE,RF, tf-idf, leng, NID_1,Sta	0.624	0.489

表 5.34 Task 3 的评测结果

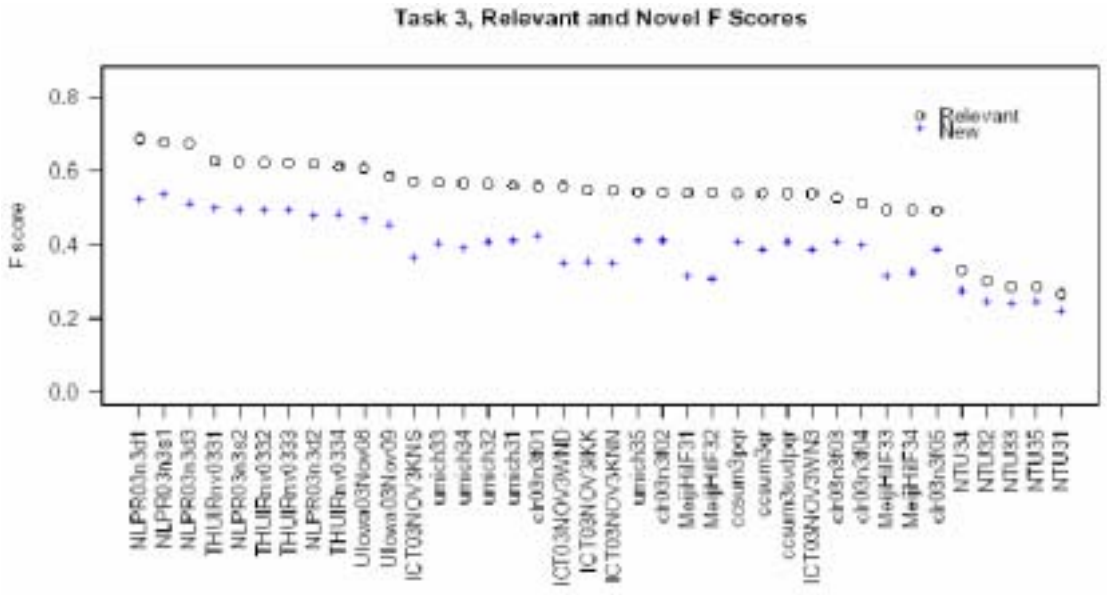


图 5.35 Task 3 的对比评测结果

Task 4 Table

ID TAG	Algorithms	Average F Measure
NLPR03n4d1	NID_1,Dyn	0.775
NLPR03n4s1	NID_1, Sta	0.789
NLPR03n4s2	NID_1+2, Sta	0.794
NLPR03n4s3	NID_2, Sta	0.796
NLPR03n4d2	NID_2. Dyn	0.773

表 5.36 Task 4 的评测结果

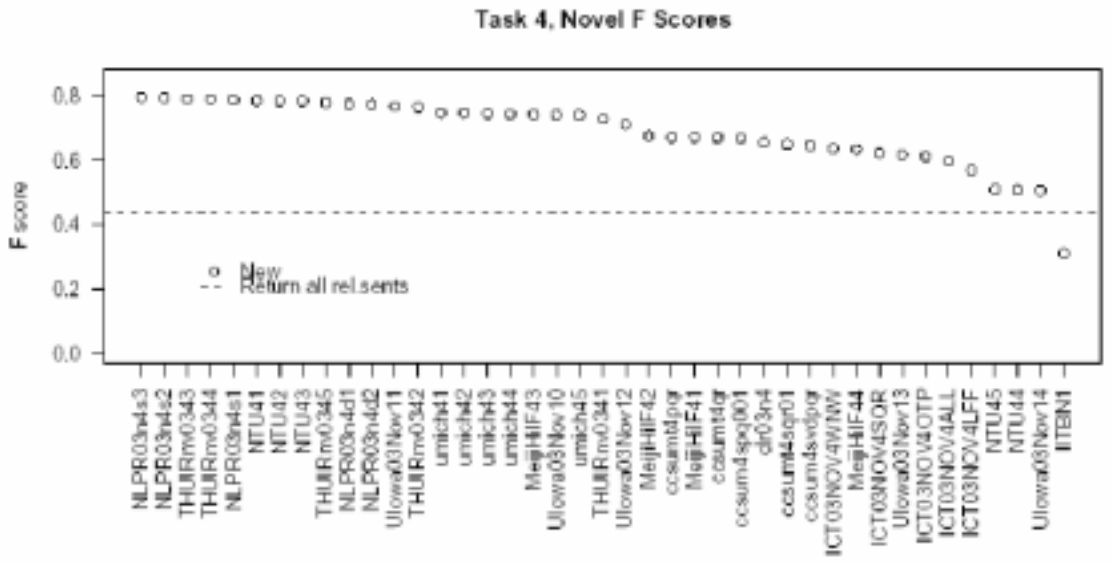


图 5.37 Task 4 的对比评测结果

从结果中可以看到，核心窗口模型以及相关反馈等技术对句子检索任务来说，是相当有效的。在 Task 2,3,4 中，都获得了比较高的 F 值，但是在 Task 1 中的 F 值比较小。这主要是因为在 Task 1 中，我们采用了 TREC 2002 的 Novelty 数据来训练检索系统，而这部分数据中，相关句子非常少，造成训练得到得阈值很大。然而，TREC 2003 的数据很不相同，相关程度很大，许多查询甚至相关的句子占了总句子数的 50%还多，因此在高阈值的情况下，准确率非常高，但召回率很低。由于这两者的不平衡，使得 F 值不是很高。在 Task 2，3，4 中，我们及时调整阈值，获得了比较理想的 F 值，在后三个 Task 中排名第一。

5.3.4 TREC-2003 任务小结

在 2003 年的 TREC 评测中，NLPR 检索系统参加了其中的两项 Robust 和 Novelty。通过评测，检验了系统的各项功能和性能。应该说，在系统层面上已经基本成型，可以满足各种类型的文本检索需求。在算法方面，进一步验证了我们所提出的几个语义模型的有效性，并在检索的实践中获得了很好的效果。

此外，在文本检索方面积累了一些经验。整理一下，有以下几点：

（1） 基于语义树模型的查询扩展是有效的。对于句子检索来说，由于所包含的词很少，所以需要大规模地扩展主题词；但对于文本检索来说，这种扩展需要非常谨慎，避免引入噪声。

（2） 静态阈值和动态阈值各有其适合的应用背景，很难说哪个更合理。

（3） （张量）窗口系列模型的相似度计算，比传统的矢量模型（如 tf-idf）有较大的优势，能大大提高检索的准确率。

（4） 在文本检索中，采用相关反馈技术一般都能提高检索性能。

（5） 在英文检索中，词形还原（Stemming）对提高召回率是至关重要的。但是，对某些查询来说，可能会导致准确率迅速下降。这就需要一个合理的折衷。

NLPR 检索系统参加 TREC 2003 评测的报告，可在以下网址下载：

http://trec.nist.gov/pubs/trec12/papers/chinese-acad-sci.novelty.robust.pdf

第六章 总结与展望

6.1 基于语义模型的信息检索总结

本文的主题是基于语义模型的文本信息检索。在第一章中简要介绍了文本信息检索的背景和研究现状，以及已有的一些语义模型；其中说明了文本信息检索中两大关键性的技术：“标引”和“相似度的计算”，并提出语义模型如何与这两个关键技术结合的问题。从第二章到第四章，分别详细描述了三类语义模型，以及它们在信息检索中的应用。其中，第一类隐含语义标引系列模型是建立的在前人研究的基础上，提出了模型的改进方式，面向的主要是“查询主题词构造”技术。第二类语义树模型，是一种构造语义空间的全新方法，面向的同样是“查询主题词构造”技术，实验证明比隐含语义标引模型更有效。第三类是窗口系列模型，这是语义张量概念的一种具体表述，无论从概念还是模型上都是全新的，面向的是“相似度计算”技术。第五章前半部分介绍了 NLPR 检索系统的构成，实现方式，模块定义，函数接口等等；其中包含了如中文分词、英文 Stemming 等算法的描述，也包含了工程实现的内容。第五章后半部分介绍了信息检索评测的基本指标和 TREC 评测的基本情况；以及 NLPR 检索系统，结合语义模型技术，参加 2003 年 TREC 评测的表现（Robust 检索和 Novelty 检索）。第六章是文本内容的总结与展望。

总体上来说，文本的工作可归纳为以下几点：

（1） 论述了语义模型和信息检索中两个关键技术的结合问题。

（2） 改进了隐含语义标引模型，建立了 SPLSI 模型，使其语义空间分布更合理，效率也更高。这个模型可以小规模的应用于“查询主题词构造”技术。

（3） 提出了新的基于语义树的语义空间模型。语义空间不再是静态的，而是实时构建的，其灵活性和可操作性都非常好。尤其应用在查询主题词扩展技术上，有很出色的表现，性能超过了常见的扩展算法。

（4） 提出了语义张量的概念，并明确了其物理意义，提出了两个核心思想。进一步，用窗口系列模型来表述这两个思想，并应用于查询和文本间的相似度计算。实验证明，这系列模型比传统的矢量模型更有效。

（5） 构建了 NLPR 检索系统框架，并完成了细化设计和编程的工作。

（6） 通过参加 TREC 评测，测试了检索系统的功能和性能，并积累了一些文本检索的经验。其中，在 Novelty 检索任务中获得了比较出色的成绩。

6.2 基于语义模型的信息检索展望

基于语义模型的信息检索，应该说正逐渐成为一个研究热点。一方面，自然语言的研究在现阶段还很难做到真正理解的层面，尤其是中文。因此，利用一些表层信息，如本文提到的各种词汇语义模型，来实现对语义的初步理解，是现阶段比较实用的方式。另一方面，互联网的飞速发展和人们对信息的渴求，也使检索技

术受到了前所未有的重视。目前普遍的看法是，检索技术将会是互联网的下一个高速增长点和盈利点，Google 的奇迹就是很好的证明。进一步提高检索的性能，也就成为了当务之急。综合这两点来看，我认为基于语义模型的信息检索，是非常有前景的一个研究方向。

文本工作所涉及到的 NLPR 的检索系统，以及多种语义模型还有很大的扩展与改进的空间。作为与后续研究者的一些交流，下面将列出我的一些想法和建议；它们是按照算法实现的难度排列的。

(1) 隐含语义标引系列模型，是所有模型中数学表述最完美的；但由于空间和时间复杂度太大，使得目前还无法大规模的应用。但是，应该看到这个缺点并不是不可克服的。如果能够有类似大型机这样的计算环境，那么使用 SPLSI 模型去构造一个完整的语义空间，可能会有出乎意外的良好性能。

(2) 语义树模型是相当高效和灵活的，每个根节点所连结的树枝个数和扩展层数，都可以根据不同词的不同特性来决定。这方面的研究，应该有深入下去的价值。

(3) 语义张量是新提出的一个概念。这个概念从逻辑和物理意义上来说都是正确而完整的，但如何实现是一个难题。所提出的窗口系列模型，只是语义张量概念的一种模型化方式，而且仅仅是一种变通的实现方法。因此，寻找一种切合语义张量概念的实用的数学模型，也是很有价值的研究方向。

(4) 目前已经有一些人工构造的词汇语义模型，如 WordNet[11]、HowNet[5]等等；也有一些基于统计模型的语义表述，如本文提到的各种语义模型。那么，两者之间到底是一种什么关系？它们的相同点和不同点在哪里？还没有一个令人满意的回答。唯一可以确定的似乎是，在面向应用的时候，两者的特性显然有比较大差距，也就是说很难相互替代使用。
如果，我们能够了解清楚两者之间的关系，那么就可以相互指导对方的构造与更新过程。由于两者的知识来源不同，因此这种类似于 Co-training 的训练过程，将有助于提高性能。此外，更理想的模式是，将两者统一起来，采用一致的语义空间框架。

(5) 目前，Ontology 是一个研究热点，既有研究者从心理学、语言学出发来构造概念模型，也有研究者从统计的角度出发去构造。两者各有优势与不足。简单来说就是前者需要人工构造，很难大规模的实现；后者缺乏监督，自动获得的结果可信度不高，可控性也不好。
如果第（4）项中描述的两种词汇语义模型，能够相互融合起来，那就为两种 Ontology 的构造方式之间寻找到了一个桥梁。一种可能的方式就是，人工构造 Ontology 的顶部，而用统计方法构造每个小领域的 Ontology，两者之间有一定重叠，然后利用这种融合技术将两者联系起来，形成统一的知识概念架构。从而，能够真正实现 Semantic Web 的构想。（互联网的 HTML 技术解决了显示一致性的问题，Semantic Web 的 XML 技术将解决内容一致性问题。）

目前，基于大规模数据库和互联网的检索技术已经给人们提供了许多方便：发邮件、看新闻、查资料、炒股票、玩游戏、远程教育、电子商务……但是，人们在享受这种种便利之时，实际上都还揣着一个未能实现的共同美好愿望：凡发布信

息，能及时到达所有需要者面前；凡寻求信息，能无一遗漏地自动来到自己的面前。换句话说，人们理想中的检索技术，应该能够实现一个智能化的全球信息共享俱乐部。让我们一起期待这一天的到来！

参考文献

[1] DEERWESTER S.,DUMAIS S.T.,FURNAS G.W.,LANDAUER T.K., and HARSHMAN R., *Indexing by latent semantic analysis*. Journal of the American Society for Information Science, 1990.

[2] THOMAS HOFMANN, *Probabilistic Latent Semantic Indexing* ,Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval,1999.

[3] Jianfeng GAO, Jian-yun NIE, Endong XUN, Jian ZHANG, Ming ZHOU and Changning HUANG, *Improving Query Translation for Cross-Language Information Retrieval using Statistical Models*, SIGIR, 2001.

[4] Jun ZHAO, *The Approaches and a Framework of Translingual Text Processing*, Chinese- Japanese Natural language Processing Proseminar (2nd), 2002.

[5] Zhendong Dong,Qiang Dong.Introduction to HowNet. www.keenage.com, 1999.

[6] Qun Liu, Sujian Li.*Computing Word Similarities based on HowNet*. Computational Linguistics and Chinese Language Processing, Vol.7, No.2 pp.59-76, 2002.

[7] Salton, G. And C. Buckley. *Term-weighting approaches in automatic text retrieval*. Information Processing & Management 24(5),513-523, 1988.

[8] Kenneth W.Church, William A.Gale, *Inverse Document Frequency(IDF): A Measure of Deviations from Poisson*. AT&T Bell Laboratories, 1995.

[9] Qin*g Ma, Mi*n Zhang, Ming Zhou, Changning Huang, Hitoshi Isahara1 . *Emergence of Chinese Semantic Maps from Self-Organization*. The 8th International Conference on Neural Information Processing (ICONIP2001), Shanghai, China, November, 2001.

[10] Qing Ma, Min Zhang, and Ming Zhou . *Self-Organization of Chinese Semantic Maps Using TFIDF Term Weighting*. The Second Workshop on Natural Language Processing and Neural Networks (NLPNN2001), the post-conference of NLPRS2001, Tokyo, Japan, November, 2001.

[11] Miller G A,et al. 1990. *Introduction to WordNet: an on-line lexical database. International Journal of Lexicography*,1990,3(4):235-312.

[12] Amit Singhal, Chris Buckley, Mandar Mitra. *Pivoted document length normalization*. In SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 21-29, 1996.

[13] Qianli Jin, Jun Zhao, Bo Xu. *Window-based Method for Information*

Retrieval. The First International Joint Conference on Nature Language Processing (IJCNLP-04), Hainan Island, China, 2004.

[14] Junfang Zeng, Yiping Yang. *Information Retrieving Based on Conceptual Network*. International Conference on Nature Language Processing and Knowledge Engineering (NLPKE), Beijing, China, 2003.

[15] Qianli Jin, Jun Zhao, Bo Xu. *Query Expansion based on Term Similarity Tree Model*. International Conference on Nature Language Processing and Knowledge Engineering (NLPKE), Beijing, China, 2003.

[16] Rocchio, J.J. *Relevant Feedback in Information Retrieval*. Chapter 14, pages 313-323. Prentice-Hall Inc, 1971.

[17] Qianli Jin, Jun Zhao, Bo Xu. *NLPR at TREC 2003 – Novelty and Robust Track*. Text REtrieval Conference (TREC-12), NIST, Maryland, USA, 2003.

[18] Qianli Jin, Jun Zhao, Bo Xu. *Weekly-supervised Probabilistic Latent Semantic Indexing and its applications in Cross-lingual Information Retrieval*. Joint Symposium on Computational Linguistics (JSCL-2003), 2003.

[19] Ponte J. and and Croft W. *A language modeling approach to information retrieval*. In Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval, pages 275-281,1998.

[20] Rila Mandala, Takenobu Tokuanga, Hozumi Tanaka, *Combining multiple evidence from different types of thesaurus for query expansion*. SIGIR,1999.

[21] Hong-zhao HE, Pi-lian HE, Jian-feng GAO and Chang-ning HUANG, *Query Expansion Based on the Context in Chinese Information Retrieval*. Journal of Chinese Information Processing, Vol 16. No 6, pages:32-37,2002.

[22] Lisa Ballesteros and W. Bruce Croft. *Dictionary Methods for Cross-lingual Information Retrieval*. In Proceedings of the 7th International DEXA Conference on Database and Expert Systems Applications, 791-801,1996.

[23] Ming ZHANG, Ruihua SONG, Chuan LIN, Shaoping MA, Zhe JIANG, Yijiang LIU and Le ZHAO, *Expansion-Based Technologies in Finding Relevant and New Information*. Text REtrieval Conference (TREC 11), 2002.

[24] Lide Wu, Xuanjing Huang, etc., *FDU at TREC-9: CLIR, QA and Filtering Tasks*. In: The Ninth Text REtrieval Conference (TREC 9), 2000.

[25] Kevyn Collins-Thompson,Paul Ogilvie, Yi Zhang, Jamie Callan, *Information Filtering, Novelty Detection, and Named-Page Finding*. Text REtrieval Conference (TREC-11), NIST, 2002.

[26] Karen Sparck Jones. *A statistical interpretation of term specificity and its application in retrieval*. Journal of Documentation, 28:11 – 21, 1972.

[27] S.E. Robertson, S.Walker. *Okapi/Keenbow at TREC-8*. Text Retrieval Conference (TREC-8), NIST Specail Publication 500-246, 1999.

[28] Warren R. Greiff. *A theory of term weighting based on exploratory data analysis*. In Proceedings of SIGIR, 1998.

[29] Hiemstra, D. *A probabilistic justification for using tf.idf term weighting in information retrieval*. International Journal on Digital Libraries 3(2), 131-139, Springer-Verlag, Berlin Heidelberg, 2000.

[30] Feifan Liu, Qianli Jin, Jun Zhao, Bo Xu. *Bilingual Chunk Alignment based on Interactional Matching and Probabilistic Latent Semantic Indexing*. The First International Joint Conference on Nature Language Processing (IJCNLP-04), Hainan Island, China, 2004.

[31] Fujita, S. *Notes on Phrasal Indexing JSCB Evaluation Experiments at NTCIR AD HOC*, in Proceedings of NTCIR-1 workshop, 1999.

[32] Tokunaga Takenobu, Ogibayasi Hironori, Tanaka Hozumi. *Effectiveness of complex index term in information retrieval*. The 6th RIAO Conference. Pp.1322-1331, 2000.

[33] http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes/bow_diff/stem.c

[34] Smadja. *Retrieving Collocations from Text: Xtract*. ACL, 1993.

[35] Qiang ZHOU, *Chinese Chunking and Tagging*.Chinese Information Transaction, 1997.

[36] Dias. *Using LocalMaxs Algorithm for Extraction of Contiguous and Non-contiguous Multiword Lexical Units*, 9-PCAI, 1999.

[37] Wei WANG and Ming ZHOU. *Structure Alignment Using Bilingual Chunking*, Colin 2002.

[38] Dekai WU, *Stochastic inversion transduction grammars and bilingual parsing of parallel corpora*. Computational Linguistics, 1997.

[39] David, Andrew and Michael. Latent Dirichlet Allcocation, Berkeley and Stanford, 2002.

[40] Gene, Knut and Paul. *Computing the SVD of a general matrix product/quotient*, SIAM, 2000.

[41] CHEN, ZHAO, XU and LIU, *Bilingual chunk processing based on spoken language, Chinese Information Transaction*. 2002.

[42] Cong LI and Hang LI. *Word Translation Disambiguation Using Bilingual Bootstrapping*. ACL, 2002.

[43] 黄昌宁,高剑峰,李沐．*对自动分词的反思*．《语言计算与基于内容的文本处理》,第七届全国计算语言学联合学术会议论文集, 2003.

[44] 陆汝占．*概念、语义计算及内涵逻辑*.《中文信息处理若干重要问题》,徐波,孙茂松,靳光瑾主编，2003 年第一版。

[45] 黄曾阳．*语义及概念体系在 NLP 中的作用*.《中文信息处理若干重要问题》,徐波,孙茂松,靳光瑾主编，2003 年第一版。

[46] 朱德熙．*《语法讲义》*,商务印书馆，1982 年版。

攻读硕士学位期间发表的论文

1．Qianli Jin, Jun Zhao, Bo Xu. *Window-based Method for Information Retrieval*. Lecture Notes on Artificial Intelligence / The First International Joint Conference on Nature Language Processing (IJCNLP-04), China, 2004.
[SCI 检索]

2．Feifan Liu, Qianli Jin, Jun Zhao, Bo Xu. *Bilingual Chunk Alignment based on Interactional Matching and Probabilistic Latent Semantic Indexing*. Lecture Notes on Artificial Intelligence / The First International Joint Conference on Nature Language Processing (IJCNLP-04), China, 2004.
[SCI 检索]

3．Qianli Jin, Jun Zhao, Bo Xu. *NLPR at TREC 2003 – Novelty and Robust Track*. Text REtrieval Conference (TREC-12), NIST, Maryland, USA, 2003.

4．Qianli Jin, Jun Zhao, Bo Xu. *Query Expansion based on Term Similarity Tree Model*. International Conference on Nature Language Processing and Knowledge Engineering (NLPKE), Beijing, China, 2003.

5．金千里，赵军，徐波. *弱指导的统计隐含语义标引及其在跨语言信息检索中的应用*. 《语言计算与基于内容的文本处理》, 第七届全国计算语言学联合学术会议论文集，哈尔滨，中国，2003.

致　谢

三年的研究生生涯即将结束了，这段时间在印象中是短暂的，但却在我的生命中留下了不可磨灭的烙印。从二十三岁到二十六岁，学会了奋斗，学会了坚持，学会了包容。一路走来，无论成功或失败，消沉或振奋，总有一些老师和同学在默默的关心和帮助，他们是我成长的动力。

首先，衷心感谢我的导师徐波研究员和赵军副研究员。是他们把我带入了自然语言处理的领域，帮助我明确了研究方向，并一直给予我殷切的关怀和细心的教导。他们严谨的科研态度，广阔的思维，渊博的知识，在我的研究工作中，起到了关键性的作用。作为学生，我从他们那里学到的不仅仅是学术知识，更重要的是学到了很多做人的道理。特别是赵军老师，他孜孜不倦的工作精神，谦虚谨慎的作风，宽厚优雅的人格魅力，都对我产生了潜移默化的影响，并将是我一生中学习的楷模。

我还要感谢黄泰翼研究员，宗成庆研究员，刘文举副研究员，张树武副研究员，博士生谢国栋、胡日勒、曹文洁、刘非凡、吴友政、段湘煜、周玉，硕士生刘丁、陈克利、徐晋，以及其他朝夕相处的中文信息处理组的同学们，与我分享了许多烦恼与喜悦，在学习和生活中给了我很大的帮助。点点滴滴的交流与探讨，往往能启发新的概念与思想。

特别感谢黄泰翼研究员，潘越研究员，孙茂松教授，宗成庆研究员，审阅了论文并提出了宝贵的意见。

最后，我还要感谢我的父母以及我的女朋友，在我成长的过程中一直给予我希望和鼓励。他们的关怀和无私的爱是我前进的最大动力。

我祝愿每个人在未来的日子里，不论身在何处，都能幸福快乐！

金千里
二零零四年五月