中山大学

硕士学位论文

基于查询相似性的直推式排序学习算法研究

姓名：蔡奕伦

申请学位级别：硕士

专业：计算机软件与理论

指导教师：汤庸

20100603

论文题目： 基于查询相似性的直推式排序学习算法研究
专业： 计算机软件与理论
硕士生： 蔡奕伦
指导教师： 汤庸教授

摘 要

在这个信息爆炸的时代,信息检索系统的出现无疑给人们在因特网上寻找自己感兴趣的内容提供了莫大的帮助。对于系统返回的大量文档来说,用户们最关注的就是这些文档的顺序。一个好的检索系统应该把最接近用户查询意愿的文档排在返回列表的最前面。

随着当前互联网上可得到的信息量快速增长,研究人员越来越意识到一个好的排序算法对检索系统的重要性。在过去的几十年里,各国学者提出了很多信息检索的技术,但是与日趋庞大的信息量相比,这些技术存在着一些共同的问题(如模型参数难以手工调整和过度拟合等现象),而且也很难把不同的模型结合起来形成一个更有效的新模型。

另一方面,机器学习领域的研究学者们已经发展出了一套成熟的理论用于解决前面提到的这些问题。近年来,一些学者开始利用机器学习技术来指导信息检索中的排序过程,并取得了比较大的突破。从而,"排序学习"作为信息检索领域的一个新分支迅速成为了当前热门的研究方向。目前多数流行的排序学习算法都是采用全监督的方式学习一个单一的排序模型来进行的,这种方法对于用户提交的多种多样的查询来说存在着一定的局限性。因为排序算法在训练的过程中有可能会在不同的查询之间采取折中的模型,从而导致排序准确率的下降。

在本文中我们提出了一种基于查询之间相似性信息的直推式排序学习方法来为每一个查询生成一个独立的排序模型。首先,本文利用一个查询的相应文档特征的标准差定义了一种全新的查询特征,并提出了一种与 Kendall's τ 距离类似的方法来度量两个查询之间的相似性;其次,本文采用这些方法从训练集中寻找与测试集上的查询相似的那些查询,并利用它们为训练集和测试集分别产生了多个额外特征。通过将这些特征添加到原来的训练集与测试集中得到新的数据集;最后,我们利用目前流行的多种全监督排序学习算法在新的数据集上进行训练与预测,并将实验所得

的结果与原始的全监督排序学习算法结果进行了比较。由于这些新添加的特征能够更好地表达出相应的查询,我们认为这个新的直推式学习框架将有可能提高排序的准确率。

我们在 LETOR 数据集上采用 RankBoost 与 SVM^{map} 算法对文中提出的直推式学习框架进行了实验,并给出了多个信息检索评价标准(MAP 和 NDCG)的比较结果和相应的分析。实验结果表明,本文提出的改进算法能够更加有效地提高文档排序的精度。

关键词: 排序学习,直推式学习,查询相似性

Title: Transductive Learning to Rank Using Query Similarity Information

Major: Computer Software and Theory

Name: Cai Yilun

Supervisor: Prof. Tang Yong

Abstract

In the information-explosion era, there is no doubt that the introduction of information retrieval (IR) systems provide great help on finding useful information on the Word Wide Web. For those large amount of documents returned by the systems, users usually pay more attention to the top documents. The ranking of each document, therefore, becomes the most crucial part to the user's searching experience. A good IR system should rank those most relevant documents on the top of the returned list.

As the amount of readily available information growing rapidly, it is increasingly important to be aware of the ways to perform better ranking task over the retrieved documents. Many IR techniques have been proposed by international researchers in the past decades. However, comparing to the enormous growth rate of the amount of information, it seems that existing IR techniques suffer the problem of tuning model parameters, even overfitting. Also, it is difficult to combine two different models to make a better prediction.

However, in the field of machine learning, maturate theories have been developed to solve those problems described previously. Recently, a great number of IR researchers leveraged machine learning methods into the ranking task of information retrieval and achieved significant improvements. Hence, "Learning to Rank for IR (LR4IR)", a new branch of information retrieval, is becoming more and more popular among IR research groups. Nowadays, most of the learning to rank algorithms conduct a supervised learning fashion to learn a single predict model for ranking, which may lead to lower accuracy since the single model would make compromise among different query cases.

In this paper we proposed a transductive learning framework which exploited the query similarity information to generate more adaptive training model for each query in the test set. Firstly, we proposed a new method to find similar queries by defining query

features as the standard variance of those corresponding documents' features, and using a novel metric similar to the Kendall's τ distance to measure the similarity of each query pair. Then we combined these methods to find similar queries of each query over the test set, and used them to generate additional document features for both the training set and the unlabeled test data. After that we learned predict functions on the derived training and test data via traditional supervised algorithm. We assumed this framework had the potential to improve the ranking accuracy since the new features better adapted to the queries on the test set.

We conducted the transductive learning framework with two state-of-the-art supervised learning algorithms RankBoost and SVM^{map} on the LETOR benchmark data collections, and made detailed analysis over our experimental results. The comparison of IR evaluations (MAP and NDCG) showed that our transductive methods outperformed the corresponding supervised algorithms.

Keywords: Learning to Rank, Transductive Learning, Query Similarity

论文原创性声明

本人郑重声明：

所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：蔡奕伦

日期：2010年5月16日

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

保密论文保密期满后，适用本声明。

学位论文作者签名：蔡奕伦　　导师签名：

日期：2010年 5月 16日　　日期：2010年 5 月 17 日

第1章 引言

随着互联网的快速发展,人们可以接收到的信息也越来越多。信息检索就是为了更好地帮助用户从这些海量信息当中搜索自己需要的信息而兴起的一门技术。而在检索系统返回的信息中,人们更关心的是那些与自己的需求更具相关性的信息。于是,研究如何更好地确定信息与查询之间的相关程度的学术分支便成为了当前信息检索领域的一个热门研究方向。近年来,越来越多的机器学习方法被引入传统的信息检索领域,并取得了比较大的突破。本章将从介绍信息检索开始,对排序算法在搜索引擎中的作用与意义进行阐述,并提出本文研究的思路与创新点,最后给出本文的组织结构。

1.1 信息检索与排序学习

"信息检索"(Information Retrieval,IR)一词最早出现于1948年MIT的学生Calvin N. Mooers的硕士论文中,并从1950年开始在公开文献中被使用。所谓信息检索,指的就是将信息按一定的方式组织和存储起来,并根据用户的需要找出有关信息的过程和技术。也就是说,信息检索包括了"存"和"取"两个环节和内容。狭义的信息检索特指后半部分,即从信息集合中找出所需要的信息的过程,也就是我们常说的信息查询。

早期的信息检索技术只应用于图书馆里面文献的检索。在计算机发明以后,人们开始尝试使用计算机手工为一些科技商业文献的摘要建立文本检索系统,并在索引模型、匹配策略以及文档内容的表示等方面取得了丰富的成果。其中一个里程碑式的工作出现于1957年——任职于IBM的H. P. Luhn提出了利用单词作为索引单位来为文档构建索引的方法,并提出把度量文档间单词的重叠程度作为检索评价的一个标准。这种方法就是目前常用的倒排索引技术的雏形。这一时期研究人员还提出了包括向量模型以及概率检索模型在内的多种理论模型[1],并成功开发了Lexis-Nexis、Dialog以及MEDLINE等大规模商用数据库检索系统。

19世纪后期,随着计算机和互联网技术的迅速普及,各种电子信息大量地涌入我们的日常生活当中。据估计,截至2010年3月20日,互联网上被索引的网页数量已经超过了200亿[1]。信息的总量在以惊人的速度高速膨胀,手工检索所有这些信息显然是一种不现实的做法,我们需要一些工具来帮助我们处理这些海量的数据,从中

[1] http://www.worldwidewebsize.com

提取出我们感兴趣的内容。信息检索技术早期在图书文献检索方面的成就让研究学者们看到了它在互联网时代的前景。进入19世纪90年代,网络搜索引擎开始出现并迅速走红。1990年,加拿大麦基尔大学计算机学院开发了世界上第一个网络搜索工具:用于FTP搜索的Archie。Archie可以根据用户输入的文件名返回该文件所在的下载地址。1994年,斯坦福大学的两位博士生David Filo和杨致远开发了Yahoo!系统,而在1998年,同样是斯坦福博士生的Larry Page和Sergey Brin开发了Google搜索引擎,并提出了著名的PageRank[1]算法。与此同时,Ringo和Amazon等提供推荐服务的公司也悄然兴起。

无论是在搜索引擎中还是在推荐系统中,排序过程都是其核心的组成部分。在本文中我们只关注那些基于文档检索的排序学习过程。近年来,由于有了更多可用的训练数据集,研究人员开始考虑把成熟的机器学习理论引入传统的信息检索过程来建立更加有效的排序模型。这种研究方法一般被称为用于信息检索的排序学习(Learning to Rank for IR,LR4IR)。在最近的这些年里,排序学习一直是信息检索领域的热门研究方向,学者们提出了很多有创意的算法来解决这个难题。

1.2 排序算法与搜索引擎

当前,网络应用日趋丰富,网络信息量与日俱增。海量的信息在给人们提供了更丰富的资料来源的同时也加重了人们检索有效信息的难度。搜索引擎以其日趋准确的服务大大提升了互联网用户的体验,在互联网应用中的用户规模和市场规模都在快速地增长。每一个网络用户几乎在自己每天上网时都会或多或少地使用搜索引擎搜索自己感兴趣的内容。

下面两则不同的调查分析报告从一个侧面反映了目前国内外搜索引擎市场的发展状况:

2010年1月15日,中国互联网络信息中心(CNNIC)发布了第25次《中国互联网络发展状况统计报告》[2]。报告中的数据显示,截至2009年12月,中国网民网络应用使用率排名前三甲的分别是网络音乐(83.5%),网络新闻(80.1%)以及搜索引擎(73.3%)。其中搜索引擎的使用率较2008年增加了5.3个百分点,超过了即时通信成为网民使用互联网的第三大应用。目前中国搜索引擎用户规模达到2.8亿人,年增长率为38.6%。

2010年3月10日,美国知名的数字测评公司comScore在它对美国搜索市场的

[2] http://www.cnnic.net.cn/uploadfiles/pdf/2010/1/15/101600.pdf

comScore qSearch 每月分析报告[3]中指出:仅仅 2010 年 2 月份一个月美国人就进行了近 145 亿次核心搜索,其中 Google 公司以 65.5% 的市场份额遥遥领先于排名第二的 Yahoo! 公司 48.7 个百分点。

目前比较著名的搜索引擎包括国外的 Google、Yahoo!、Bing 以及国内的百度等。一个典型的搜索引擎体系结构如图1-1所示。

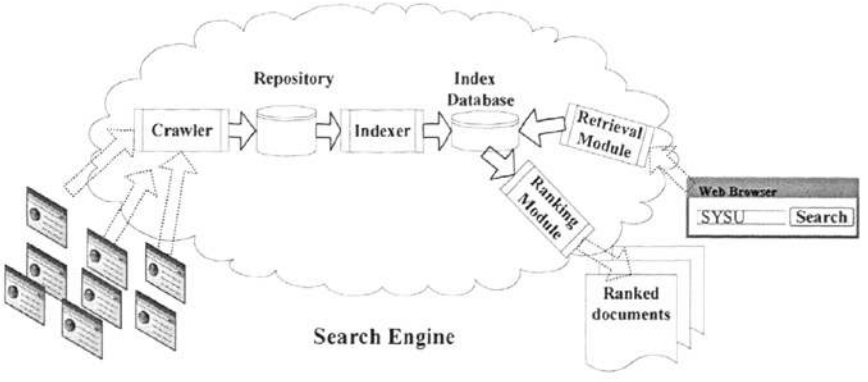

图 1-1 搜索引擎的体系结构

从图1-1中我们可以看到,搜索引擎体系中除了爬虫(Crawler)模块、索引(Indexer)模块、排序模块以及检索模块这四个核心模块之外,还包括了用于交互的用户界面以及用于作为爬虫模块输入的大量互联网网页。

爬虫模块 爬虫模块一般用于在庞大的互联网中抓取网页并分析保存,以便用于后续的索引过程。传统上一个爬虫模块包括了用于解释服务器 URL 地址的 DNS 解释器、用于使用 HTTP 协议抓取网页的模块、用于抽取网页内容和链接的分析器以及用于去除重复网页的模块。

索引模块 搜索引擎在接收到爬虫模块抓取的网页后,将使用索引模块抽取出索引项,用于表示文档以及创建文档库的索引列表。目前大多数的搜索引擎采用的都是倒排索引(Inverted Index)技术。对于目前庞大的互联网信息量来说,索引模块需要将信息存储到由数以千计的计算机组成的集群中。

检索模块 当用户通过搜索引擎提供的界面输入查询之后,检索模块用于快速地在索引数据库中检索出与查询相关联的文档。目前大多数的搜索引擎都提供了诸

[3] http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Releases_February_2010_U.S._Search_Engine_Rankings

如模糊搜索、逻辑查询以及范围限定等高级搜索功能，检索模块需要根据用户的输入正确地返回与查询需求相关的文档列表。

排序模块 通常与一个查询相关的文档的数目可能是成千上万的，而用户需要的可能只是那些更相关的几百个甚至几十个文档。因此，搜索引擎需要根据用户的需求对检索模块返回的文档列表进行排序，将更加相关的文档排在返回列表的更前面的位置。

从前面的介绍我们可以看出，排序模块在搜索引擎体系中起着很大的作用。作为与用户交互的最后一个环节，排序算法得到文档列表的速度与质量影响着用户对该搜索引擎的满意程度。在检索模块返回的数以千万计的相关网页中，如何将用户最感兴趣的那些网页排在靠前的位置，对所有搜索引擎来说都是一个极大的挑战。一个高质量的排序算法能带来用户体验的大幅度提高，从而使得那些搜索引擎提供商能够进一步拓宽自己的用户群体，提高自己的市场占有率。

当前很多学术机构与商业公司都投入了大量的人力物力用于改善搜索引擎中的排序模块。在传统的搜索技术(像 Google 的 PageRank 算法，百度的超链分析技术等)基础之上，研究人员还提出了各种用于支持“个性化搜索”的技术。而最近，随着排序学习研究理论的兴起，研究学者们将机器学习算法应用于排序模型的训练之中，提出了很多新的算法[2]。这些成果中有很大一部分来自于商业公司的研究院[3, 4, 5, 6, 7, 8]。

1.3 研究思路和创新点

当前大多数的排序学习算法都是采用全监督的方式进行的，也就是说对于训练集中的每一个样本都有一个人工标注好的相关程度。近年来，半监督学习凭借其在减少人工标注代价和提高机器学习性能方面的突出优势，吸引了众多国际研究人员的兴趣，他们对半监督学习方法进行了广泛而深入的研究。目前，许多半监督学习技术已经被广泛地应用于分类与回归问题。最近有研究学者提出了一些用于排序学习的半监督学习算法，并取得了比较好的排序效果。Massih Reza Amini 等人[9] 提出了一种基于 Boosting 的推理式学习算法来学习一个排序预测函数，Kevin Duh 等人[10] 提出了一种利用测试集中每一个查询的模式来为其生成更适合的排序模型的直推式学习算法，而 Shivani Agarwal[11] 则提出了一种基于图的拉普拉斯矩阵的直推式排序学习算法。

与之同时，部分研究学者也尝试从查询数据之中挖掘更有用的信息来指导排序

学习过程。用户的查询需求通常是多种多样的,对于这些不同类型的查询采用相同的排序预测模型可能会带来排序准确率的下降。In Ho Kang 等人[12] 对查询的多样性以及查询分类进行了研究,并根据查询的不同分布等信息把查询分成了主题相关、主页查找以及服务查找三类;而 Xiubo Geng 等人[13] 则在此基础上提出了利用 KNN 算法来学习依赖于查询相似性的排序预测函数的算法。

本文的研究是在前人半监督排序学习和查询相似性工作的基础上进行的。我们提出了一个利用查询相似性信息来为测试集中的每一个查询学习一个单独预测函数的直推式学习框架。在 LETOR 数据集上的实验结果显示,我们的算法比目前的全监督学习算法具有更好的排序准确率。

1.4 排序学习的形式化表示

为了后面章节叙述的方便,在介绍本文的组织结构之前,我们先介绍一些后面将会用到的符号,并同时给出排序学习的形式化表示。

在本文中,我们用 x 表示一个文档(Document),或者样本(Instance)。$\mathcal{X}$ 是一个称为域(Domain)或者样本空间(Instance Space)的集合。$\mathcal{X}$ 中包含了我们感兴趣的用于排序的对象 x,即 $x \in \mathcal{X}$。

$\mathcal{F}$ 是一个被称为文档特征空间(Feature Space)的集合,里面的元素被称为文档特征 (Feature)f。在实际应用中,$\mathcal{F}$ 是实数集的一个子集,即 $\mathcal{F} \subset \mathbb{R}$。一个文档中包含了若干个文档特征 f。通常我们用特征向量(Feature Space) $\langle f_1, f_2, \cdots, f_n \rangle$ 来表示一个具有 n 个文档特征的文档 x: $x = \langle f_1, f_2, \cdots, f_n \rangle$。

令 $\mathcal{Q}$ 是一个查询空间(Query Space)的集合,里面的元素被称为查询(Query) q。对于一个查询 q,我们用 X 表示检索系统对 q 返回的文档的集合。通常在训练集中,对于每一个文档 x,都有一个与其相应的标注(Label) y,表示该文档与给定的查询之间的相关程度。令 $\mathcal{Y}$ 表示标注空间(Label Space)的集合,则 $y \in \mathcal{Y}$。一般的,$\mathcal{Y} \subset \mathbb{Z}$。对于一个具有 m 个文档的文档列表 $X = \langle x_1, x_2, \cdots, x_m \rangle$(有时也简写为 $X = \{x_i\}_{i=1,\cdots,m}$),我们用符号 $Y = \langle y_1, y_2, \cdots, y_m \rangle$(有时也简写为 $Y = \{y_i\}_{i=1,\cdots,m}$)来表示与 X 对应的相关性标注。

在排序学习的设置中,训练集 S 通常由许多个查询及这些查询的检索文档构成,每一个文档都有相应的相关性标注。一个具有 U 个查询的训练集可以形式化为 $S = \{(q_i, X_i, Y_i)\}_{i=1,\cdots,U}$。而测试集一般则由若干个查询以及这些查询的检索文档

组成,这些文档都是没有标注的。我们通常用 $T = \{(q_j, X_j, Y_j)\}_{j=1,\cdots,V}$ 来表示一个具有 V 个查询的测试集。

给定一个函数假设空间(Hypothesis Space) $\mathscr{H}$,排序学习算法的主要任务就是通过对训练集 S 中的数据进行训练,找到一个假设空间 $\mathscr{H}$ 中的函数 h 用于排序文档 $h: \mathscr{X} \rightarrow \mathscr{Y}$,这个函数 h 使得在测试集 T 上进行预测时的错误率最小。

除了上面的表述外,本文里面还有另一个常用的记号—— $[[\pi]]$ 表示一个指示函数(Indicator Function)。当命题 π 成立时, $[[\pi]]$ 的值等于 1,否则为 0。

1.5 论文的组织结构

本文共分五章。各个章节的主要内容和组织结构如下:

第一章从信息检索的介绍开始,详细探讨了排序学习对搜索技术的重要性。接着结合当前用于信息检索的排序学习的研究现状提出本文的研究思路与创新点,并给出了排序学习的形式化表示。

第二章详细介绍了用于信息检索的排序学习的相关工作。首先对排序学习的发展历程进行了介绍,并结合本文的研究内容对有关半监督排序学习以及查询相似性信息的研究进行了深入的阐述,给出了本文研究思路的来源。

第三章里我们提出了一个利用查询之间相似性信息的半监督排序学习算法。在这一章的开头我们首先介绍了算法的基本框架,然后对里面关键的特征生成步骤进行了详细的说明,最后介绍了实验中用到的两个全监督算法。

第四章对实验设置以及结果分析进行了探讨。针对当前比较流行的全监督排序学习算法,本章通过在公开数据集 LETOR 上的实验结果验证了我们提出的算法的有效性,并结合排序学习评价标准对实验的结果进行了分析。

第五章总结了全文的工作,并进一步提出了未来可以继续进行的工作。

第2章 相关工作

本章首先介绍了排序算法的发展历程,并对其中一些重要的模型与算法进行了阐述。结合本文的研究方向,我们在本章中着重对半监督学习算法以及查询相似性的研究现状进行了介绍,并仔细探讨了这些理论应用于提高排序学习结果准确率的可能性。

2.1 排序学习算法的发展

2.1.1 传统的排序算法

作为目前一个热门的研究方向,研究学者们提出了多种用于信息检索的排序模型。传统的排序模型主要可以分为依赖于查询的(Query-dependent)模型和独立于查询的(Query-independent)模型两种[14]。

依赖于查询的模型

早期的排序模型是基于查询在文档中是否出现来进行的,一个典型的例子就是布尔模型(Boolean Model)[1]。布尔模型只能判断一个文档是否跟一个查询相关,但不能给出具体的相关程度,这大大限制了它的发展。但是由于布尔模型能够带给用户更多的可控制性,在今天的搜索引擎的"高级搜索"中仍然可以看到它的身影[1]。

为了度量查询与文档的相关程度,人们引入了向量空间模型(Vector Space Model)[1]。在这种模型中查询与文档都被看成了欧氏空间里的向量。具体来说,文档对每一个词(Term) t 均计算出一个权值作为文档在该词维度上的值,这个值通常都是 t 在文档中出现的频率——词频(Term Frequency)的函数(一个常见的函数就是 $tf\text{-}idf$)。而对于查询来说,如果一个查询中包含了词 t,则这个查询在该词维度上的值为1,否则为0。通过这样的方式将文档与查询映射成向量空间中的向量,然后采用两个向量之间的欧氏距离或者是夹角余弦就可以计算出它们的相似程度,也就是文档与查询之间的相关程度。在向量空间模型中,不同的词之间是被视为相互独立的,研究人员们进一步提出了隐性语义索引(Latent Semantic Indexing)[1] 来消除模型中的这个假设。

在传统模型中,除了向量空间模型之外另一个更有名的模型就是概率检索模型

[1] http://www.google.com/advanced_search

(Probabilistic Retrieval Model)[1] 了。概率检索模型的基本思想是通过估计查询与文档相关的概率值来对文档进行排序。假设文档与查询之间的相关性只有 +(相关)与 −(不相关)两种值。其中查询 q 与文档 x 相关的概率为 $r = P(+|q,x)$,不相关的概率为 $\bar{r} = P(-|q,x)$。概率检索模型通过相关概率与不相关概率的比值 $\log(r/\bar{r})$ 来计算查询与文档之间的相关程度,从而最终实现对文档的排序。概率检索模型的优点就在于它具有良好的数学理论基础,在过去的几十年里,研究学者们提出了多种基于概率排序原理的模型(BM25 模型[1] 以及 LMIR 模型[1] 就是其中的代表),这些模型取得了比向量空间模型更好的结果。

独立于查询的模型

除了前面讲到的那些依赖于查询的模型之外,研究人员们还从相反的研究角度提出了独立于查询的排序模型。这些模型有的考虑了超链接的结构,有的则考虑了文档内容之间的相似性关系。尽管研究的侧重点不同,但是这些模型都有一个共同地方,那就是它们都是通过文档自身的相关程度来进行排序的,并且这个相关程度是独立于查询的。

在所有独立于查询的排序模型之中,最著名的莫过于 Google 公司的 PageRank[1] 算法了。这种算法通过文档的超链接之间的引用关系来对文档进行排序,有效地利用了互联网庞大的超链接结构。PageRank 算法主要基于我们生活中的一个直觉:如果一个网页(或者说文档)被很多重要的网页所引用,那么它就具有很高的重要性,否则它的重要性就相对不那么高。假如把引用看成一种"投票"行为,网页 A 指向了网页 B 便可以被视为是 A 对 B 的重要程度进行了投票。PageRank 算法根据 B 所得到的"票数"来计算 B 的重要程度。当然,仅仅看"票数"(引用数)是远远不够的,PageRank 在计算中还加入了"投票人"(网页 A)的重要程度信息,重要性高的"投票人"所投的票的权值相对也较高。网页 B 最终所得到的"加权票数和"即相当于它的重要程度,即为相关程度。这个看似简单的算法在实际的应用中取得了巨大的成功。值得注意的是,由于 PageRank 算法仅仅考虑了文档自身的重要性,而没有考虑文档与查询之间的关联,因此有可能出现 PageRank 针对一个查询返回一堆重要性很高但是根本没有联系的文档的现象,这显然是没有意义的。Google 公司为此使用了增强的文本匹配技术,使它的 PageRank 算法得能够检索出重要性高而且正确的页面。

纵观 PageRank 算法,我们发现在计算网页 B 的重要性时 PageRank 算法忽略了 B 自身的引用信息,而这个信息有可能会帮助我们提高排序的准确率。也正是基于

这一点,Jon Kleinberg 提出了 HITS(Hyperlink Induced Topic Search)算法[1]。HITS 算法为每一个网页定义了两个值:中心值(Hub Value)与权威值(Authority Value)。这种定义方法来源于现实生活中网页的构造过程。有一些网页包含了某些信息的权威来源(如世界卫生组织关于 H1N1 流感的信息),一般被称为权威网页,具有较高的权威值;而某些网页则被手工编辑提供了到一些权威页面的链接(如联合国主页中包含了各下属机构的链接),一般被称为中心网页,具有较高的中心值。网页的中心值和权威值是互相递归定义的。一个网页的权威值取决于所有链接到它的网页的中心值的和,这个和值越高,网页的权威性就越高。另一方面,一个网页的中心值则由所有链接到它的网页的权威值的和决定,这个和值越大,那么网页的中心性越高。HITS 算法的思路就是通过迭代计算来得到每一个文档的权威值从而按照该值对文档进行排序。

当然,除了以上介绍的 PageRank 算法以及 HITS 算法之外,研究人员还提出了包括 Google TrustRank 等在内的多种独立于查询的算法模型。

2.1.2 用于信息检索的排序学习

随着网络上可以获得的信息量不断增大,传统的排序算法在实际应用时遭遇到了一定的困难[14]。例如,BM25 以及 PageRank 等的模型中都具有模型参数,为了获得更好的性能,我们通常需要使用校验集来调节这些参数。调整参数并不是一件容易的事,有时还可能会出现过度拟合(Overfitting)的现象,即一个在校验集上面调得很好的模型在测试集上的表现却很差。但是由于传统排序模型大都比较简单,参数的个数也比较少,所以即使是手工调整也是可以接受的。但是随着研究的深入,研究学者们提出的新的排序模型中的参数也越来越多,手工调整这些参数已经不能适应这样的任务了。其次,目前研究人员们已经提出了很多种排序模型,一个自然的想法就是可否把这些模型组合起来形成一个更有效的排序模型,虽然看起来很简单,但实现起来却很棘手。

就在研究学者们被这些传统排序模型中的各种问题所困扰之时,机器学习以其成熟的理论展示了它在模型自动调整参数和避免过度拟合方面的巨大威力,同时还能有效地组合多个模型。将机器学习的方法引入传统的排序模型中所能带来的改进效果让很多研究人员倍感期待。

在最近几年里,越来越多的机器学习方法被引入到排序模型之中,使得用于信

息检索的排序学习成为当前一个极其活跃的研究方向。一个典型的排序学习算法流程如图2-1所示。

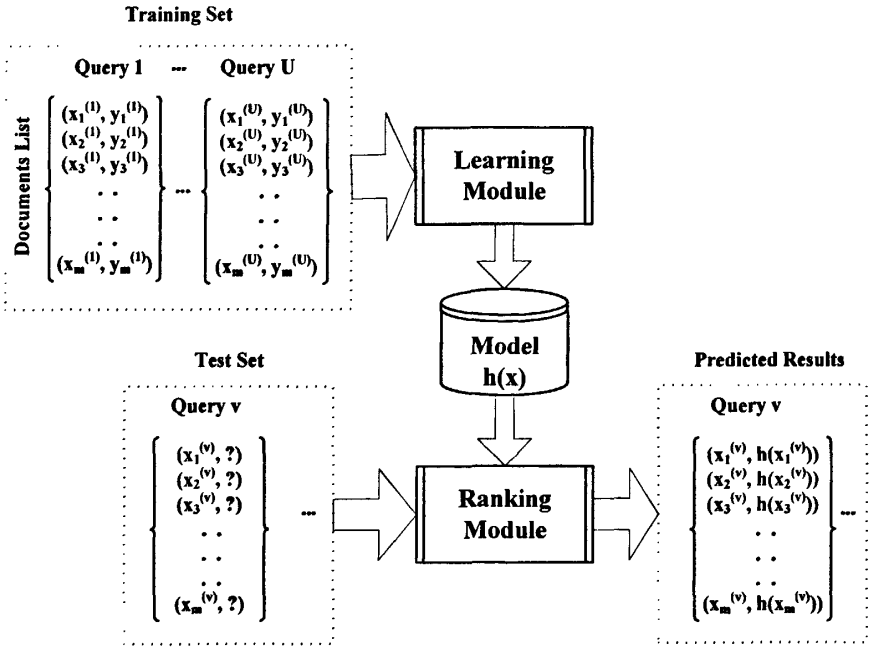

图 2-1 排序学习算法的典型流程

从图2-1中我们可以看到,排序学习算法首先对从查询-文档对(Query-Document Pair)里面抽取出来的文档特征采用机器学习的方法进行训练,得到一个在训练集上排序错误率最小的排序模型,然后再用该模型对测试集中的数据进行预测排序。根据训练过程中的样本对象构造方法的不同,排序学习算法可以分为 Pointwise 型,Pairwise 型以及 Listwise 型三类[14],这些不同类型的算法定义了不同的输入输出空间与损失函数。下面将对这三类排序模型进行详细的介绍。

Pointwise 型排序学习算法

把机器学习方法引入到排序学习中的一种直接方法就是用排序模型直接来预测每一个文档的相关程度,这种方法被称为 Pointwise 型排序学习算法。Pointwise 型算法一般把排序问题转化为传统机器学习中的分类与回归问题来求解。根据采用的机器学习技术的不同,Pointwise 型算法可以进一步分为基于回归的算法、基于分类的算法以及基于次序回归(Ordinal Regression)的算法[14]。

基于回归的 Pointwise 型算法通过学习一个回归函数为每一个文档输出一个实数值,这个实数值代表了文档的相关程度。这种算法的代表有 Polynomial Regression Function[15] 以及 Subset Ranking with Regression[16]。这两种算法的损失函数的定义存在着一个共同的问题,即当且仅当预测函数的值与实际标注完全一样时,损失函数才为 0。以二值预测为例,当且仅当预测函数在一个相关样本上的输出值为 1 时,这种算法的损失函数值才为 0。即使预测函数对相关样本给出了一个大于 1 的实数(可能某种程度上意味着这个文档的相关性更高),损失函数仍然不为 0。不相关样本的情况与之类似。

基于分类的 Pointwise 型算法的排序模型输出的是无序的类型(Non-ordered Categories),这样在一定程度上避免了上面出现的问题——因为分类的类型中不再涉及具体的预测函数值大小。这种算法的代表有 Discriminative Model for IR[17] 以及 Multi-class Classification for Ranking (McRank)[18]。其中 Discriminative Model for IR 中把相关文档当成了正样本,把不相关文档当成了负样本,并引入了最大熵(Maximum Entropy, ME)以及支持向量机(Support Vector Machine, SVM)两种经典分类模型对其进行分类。而 McRank 则引入了一个多类型分类器(Multi-class Classifier)来学习分类模型,并通过逻辑函数(Logistic Function)来将分类结果转化成排序分数。

基于次序回归的 Pointwise 型算法的排序模型输出的是有序的类型(Ordered Categories)。假设给定文档一共有 K 个类型,这种算法学习的排序模型可以很容易地通过$(K-1)$个阈值 $\{b_i\}_{i=1,\cdots,K-1}$ 来确定每个文档所属的类型。典型的代表有 Perceptron based Ranking (PRank)[19] 以及 Ranking with Large Margin Principle[20]。PRank 把文档映射到不同的区间,并通过一系列的迭代过程来修改阈值,从而最终得到以 w 为参数的排序预测函数。Ranking with Large Margin Principle 则引入了 SVM 模型来学习参数 w 和阈值 $\{b_i\}_{i=1,\cdots,K-1}$,并提出了固定间隔策略(Fixed Margin Strategy)与间隔求和策略(Sum of Margin Strategy)两种不同方法来最大化分类间隔。

由于 Pointwise 型排序学习算法的预测模型单独给出了每一个文档的预测分数,忽略了文档之间的相互次序以及文档与查询之间的关系,同时也忽略了文档的位置所带来的影响,这使得它具有一定的局限性。为了解决这个问题,研究人员进一步提出了 Pairwise 型和 Listwise 型排序学习算法。

Pairwise 型排序学习算法

Pairwise 型排序学习算法不再是直接给出每个文档的预测分数,而是通过训练学习得到文档之间的先后关系。这样排序学习算法的目标就转化为最小化排错的文档对的数目,研究学者提出了许多不同的算法来实现这个目标。这类排序学习算法的典型代表包括了 RankNet[7],RankBoost[21] 以及 Ranking SVM[22]。

RankNet 是第一个用于商业引擎的排序学习算法[14]。通过定义两个文档之间先后顺序的概率,RankNet 算法构造了一个交叉熵(Cross Entropy)损失函数,并通过神经网络与梯度下降法(Gradient Descent)来学习排序模型。在 RankNet 的基础上,研究人员进一步提出了 FRank[23] 改进模型,该模型使用量子信息科学中的保真度(Fidelity)理论构造了一个保真度损失函数。

RankBoost 算法是基于传统的 AdaBoost[24] 分类算法提出来的一个排序学习算法。类似于 AdaBoost,RankBoost 算法通过维护文档对的分布来代表不同文档对在学习中的不同重要程度,并在迭代过程中不断地修正那些重要的(通常是在上一次迭代中排序错误的)文档对,最终获得一个由若干个弱学习器(Weak Learner)线性组成的排序预测模型。

Ranking SVM 算法采用了 SVM 的原理来分类文档对,与普通 SVM 分类的约束条件建立在文档上不同,Ranking SVM 的约束条件是定义在文档对之上的。另一方面,由于 Ranking SVM 是基于 SVM 的,它继承了 SVM 许多好的特性,具有良好的泛化能力,同时可以引入核函数方法(Kernel Trick)来解决非线性分类的问题。

当前,基于文档对的研究方法是排序学习中的一个主要研究方向。Pairwise 型排序学习算法基于文档对的分类特点使得它可以很好地利用目前很多成熟的分类框架。但是,完全基于文档对也使得这种模型存在着一定的问题[14]。首先,不同的查询之间构造出来的文档对数目可能会相差很大,我们在评价的时候直接对结果求平均值可能会掩盖某些文档对数目较少的查询的排序效果很差的事实。其次,这种方法没有考虑文档排序位置与文档具体相关程度之间的联系。一般来说,一个更相关的文档应该排在列表更前面的位置,而基于文档对的方法只计算了文档之间的先后顺序,并没有考虑到文档的具体相关程度与它所排的位置之间的关系。因此,在基于文档对的算法中可能会出现一个较好的排序结果和一个较差的排序结果可能具有相同的评价结果的现象。

Listwise 型排序学习算法

为了弥补 Pairwise 型排序学习算法的不足,研究学者们提出了 Listwise 型排序学习算法。这种算法通过在排序的文档列表上直接定义损失函数或者直接优化 IR 评价标准的方法来学习排序模型。典型代表包括了 SVM^{map}[25]、AdaRank[26]、RankConsine[27] 以及 ListNet[28] 等。

一般来说,诸如 MAP、NDCG 等评价标准是不连续并且不可微的,要直接使用优化算法来对其进行优化具有很大的挑战性。SVM^{map} 建立在 SVM^{struct}[29] 的基础上,通过把 IR 评价标准 MAP 嵌入到 SVM 框架的约束条件之中来优化 MAP 的一个连续可微的凸上界。而 AdaRank 则是把 IR 评价标准 MAP 和 NDCG 嵌入到传统的 Boosting 优化框架之中,用这些 IR 评价标准来更新 Boosting 算法中的分布并且选择权值。这些直接优化 IR 评价标准的方法目前尚且存在着一些值得深入探讨的地方。例如,SVM^{map} 中的上界是否足够紧凑?不同的 IR 评价标准之间并不完全一致,在优化时应该依据什么进行选择?这些都是下一步研究的目标。

采用定义在文档列表上的损失函数是另一种 Listwise 型排序学习算法。RankCosine 使用了事实标注与预测分数两个向量之间的余弦相似性来定义损失函数并采用梯度下降法进行优化。这里定义的损失函数既不会受查询文档的数量影响,又很好地强调了排在前面的文档的重要性。ListNet 则是利用了文档列表的排序列表与其排列概率分布(Permutation Probability Distribution)是一一对应的这个事实来将预测排序列表映射到一个排列概率分布当中,通过 K-L 散度(K-L Divergence)来计算它与事实标注的排列概率分布之间的差异并定义损失函数,最后通过神经网络和梯度下降法进行优化求解。这些定义列表损失函数的方法大都是建立在实验结果之上的,目前尚缺乏严格的理论基础。

从上面的介绍可以看出,Listwise 型排序学习算法由于考虑了文档位置以及文档之间的相互联系,比 Pointwise 型和 Pairwise 型排序学习算法更有可能获得好的排序效果,而实验结果也印证了 Listwise 型算法的排序结果确实大大超过了前面两种类型。但是当前的 Listwise 型排序学习算法也不是非常完美的,有些算法存在着训练复杂度过高的问题(例如 ListNet 算法,Fen Xia 等[30] 提出了 ListMLE 算法来解决这个问题),而有些则对位置信息的考虑不够全面,没有很好地区分出前后位置的不同重要性(例如 ListNet 和 ListMLE 算法)。

2.2 半监督排序学习学习算法

全监督学习方式存在着一个比较严重的缺陷[31]:我们一般很难得到大量人工标注好的数据来进行训练。因为人工标注这些数据的成本很高,而且非常耗时间;此外,标注这些数据是需要该领域的专家来进行的,有些甚至还需要特殊的策略与技巧。这样在训练样本数目比较少的时候全监督学习的方法就会失效了。

与之相反,未标注的数据却是可以很容获得的,而且数量巨大。半监督学习吸引人的一个地方就是它可以利用已经标注的和未标注的数据来获得比全监督学习更好的性能。或者从另一个方面说,半监督学习可以获得与全监督学习相同的性能,但需要的已标注训练数据更少。这样一来就大大降低了人工标注的成本。

把未标注数据加入训练集对于学习过程是否真的有帮助呢?这种想法乍听之下似乎令人感觉非常意外。对于我们的学习过程来说,我们要从训练集的样本中学习一个预测函数 $h: \mathscr{X} \to \mathscr{Y}$ 来把样本 x 映射到标注 y 上,而未标注的数据并不能反映出从 $\mathscr{X}$ 到 $\mathscr{Y}$ 的这种映射关系,因为它的数据中根本不存在 $\mathscr{Y}$ 的信息。答案就在于人们假设了未标注数据的分布与其最终的标注之间是有联系的[31]。图2-2以一个简单的一维分类问题阐述了半监督学习中的这个假设的可能性。在这个问题中我们假设每一个样本都是只有一个样本特征 $f \in \mathbb{R}$ 的,并且每一个样本要么是正样本(Positive Sample),要么就是负样本(Negative Sample)。

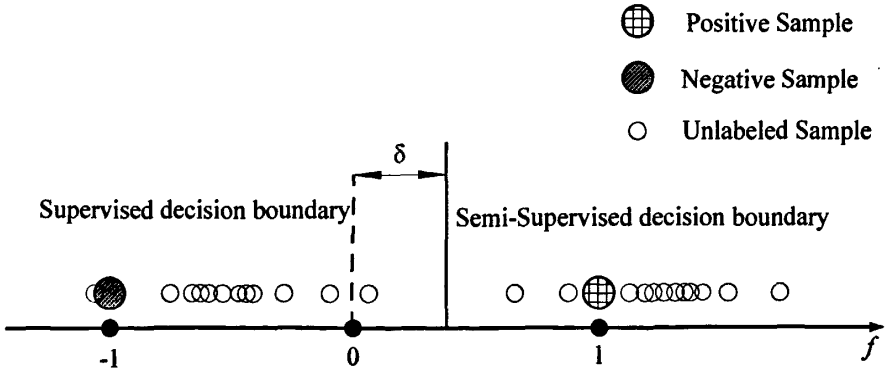

图 2-2 一个用于介绍半监督学习的一维分类例子

在全监督学习中,由于我们的例子里面只提供了一正(+)一负(−)两个样本,分别表示为 $(1,+)$ 与 $(-1,-)$。显然,最佳的决策边界(Decision Boundary)是 $f=0$(如

图2-2中虚线所示)。那些特征 $f \geq 0$ 的样本将被标注为正样本,而特征 $f < 0$ 的那些则被标注为负样本。

而在半监督学习中,除了已经标注的两个样本之外,我们还获得了许多未标注的样本。通过仔细观察,我们发现所有的样本恰好形成了两个比较明显的分类。假设属于同一个类别的样本是来自于同一个高斯分布的,那么我们就可以知道这个分类问题中正负两类样本的决策边界应该选择 $f \approx 0.4$(如图2-2中实线所示)更加合适。如果我们的这个假设是正确的,那么我们就获得了这个问题的一个更加可靠的决策边界。

通过这个例子我们看到,训练集中少量的样本显然不足以代表整个样本空间的真实情况,大量的未标注样本通过一个更加接近真实情况的分布帮助我们更好地去确定具有相同标注的区域,而这个区域中少数的已经标注好的样本则给出了这个区域的真实标注。

一般而言,半监督学习分为直推式(Transductive)与推理式(Inductive)两类。这两种方法都是利用了未标注的信息来获得更好的学习效果的,不同之处就在于两者对未标注数据的选择。推理式半监督学习方法中的未标注数据不包括测试集中的样本,它学习的预测模型是用于预测那些目前尚未见到的测试集中的数据的;而直推式半监督学习方法则是把测试集中的数据当成未标注数据来进行学习,它学习的预测模型反过来又用于对测试集进行预测。

当前,在机器学习领域已经有很多利用半监督学习算法研究分类与回归问题的成功例子[31],但是在排序学习领域方面的尝试却还比较少。

Massih Reza Amini 等[9] 提出了一个推理式半监督学习的算法框架。为了利用未标注样本的信息,他们提出了一个假设,即认为未标注数据集中的每一个样本与其在已标注样本中的相似样本应该具有相同的标注。一个简单的做法就是先从未标注数据集中选择样本并逐个添加标注,然后将其增加到原有的已标注训练集中得到一个新的训练集,最后再在这个新的训练集上利用传统的全监督排序学习算法进行排序预测。这种方法有一个缺陷:那些用于训练排序模型的未标注样本的标注并不是百分百正确的。这样一来,排序的准确率将严重地依赖于每一个未标注样本所被赋予的标注的正确性以及所选择的全监督排序学习算法的抗干扰性。为了避免这个问题,Amini 们提出了一种新的学习策略。在这种策略下排序学习算法的训练目标是使得排序模型在原始已标注样本训练集以及未标注样本训练集这两个独立的训练集上的错误率分别最小化,而不是把未标注样本添加到原始训练集中去。实验证明这

种方法在 TREC-09 以及 Reuters-21578 数据集上的排序结果比其它两种用于分类的半监督的方法有了显著性的提高。通过这些实验,Amini 的论文中还提出了一个观察结论:一个分类器的分类能力并不能完全标志出它在排序方面的性能,选择那些更好的分类器不一定能带来排序准确率的提高。

Shivani Agarwal[11] 提出了一种用于在图上进行排序学习的直推式学习算法。在这种算法下文档被看成了图中的顶点,而文档之间的相似性则被编码成相应顶点之间的边。Agarwal 的算法借鉴了近期基于图的正则化理论(Regularization Theory)以及相应的基于拉普拉斯矩阵的分类算法(Laplacian-based method for Classification)。通过一系列的变换,Agarwal 将这种基于图的排序学习归约为带拉普拉斯矩阵的二次规划问题,而这个问题可以用 SVM 进行求解。在 Agarwal 的论文中分别介绍了这种算法在无向图以及有向图上的应用,并进一步证明了它可以看成是在基于图的拉普拉斯矩阵(Graph Laplacian)上生成的再生核希尔伯特空间(Reproducing Kernel Hilbert Space,RKHS)的正则化过程。这样,从理论的角度上看,这个证明使得 Agarwal 的算法可以更好地利用那些在 RKHS 上进行正则化的算法所具有的算法稳定性与泛化属性;而从应用的角度上看,这个结果使得算法的实现过程可以大大地借鉴那些在将 SVM 应用于大规模数据集时所使用的成熟技术。最后,Agarwal 还在论文中提出了将这种直推式的半监督学习算法扩展成推理式的半监督学习算法的可能性,使之成为一个完整的半监督排序学习理论。

Kevin Duh 等人[10] 提出了一种利用测试集中每一个查询的模式(Pattern)为其独立训练一个排序模型的直推式排序学习算法。Duh 等通过 Kernel PCA 的方法发现每个查询的低维模式并将该查询以及训练集映射到这个模式上形成新的训练集以及待预测查询,然后再用传统的全监督排序模型在新的数据集上进行预测。这种基于测试集查询的排序学习算法的思路来自于数据集中的文档在不同特征维上的分布。例如,图2-3与图2-4显示了 TD2004 数据集上两个不同查询(查询 ID 分别为 97 和 192)的文档分布情况。其中,"•"代表了查询的相关文档,而"×"则代表了不相关文档。坐标系中包含了排序学习中两个比较重要的文档特征,其中横坐标(x 轴)表示文档标题的 BM25 分数,纵坐标(y 轴)表示文档的 HITS 中心(HITS Hub)分数。

在图2-3中,文档的相关程度在 y 轴(HITS 值)上具有明显的区分度;而在图2-4中,文档的相关程度则在 x 轴(BM25 值)上有明显的不同。显然,单个排序模型很难同时在这两个查询上都获得比较好的排序效果。如果采用不同的排序模型来对这两个查询的文档列表分别进行排序,将会获得更加好的排序效果。例如用一个排序

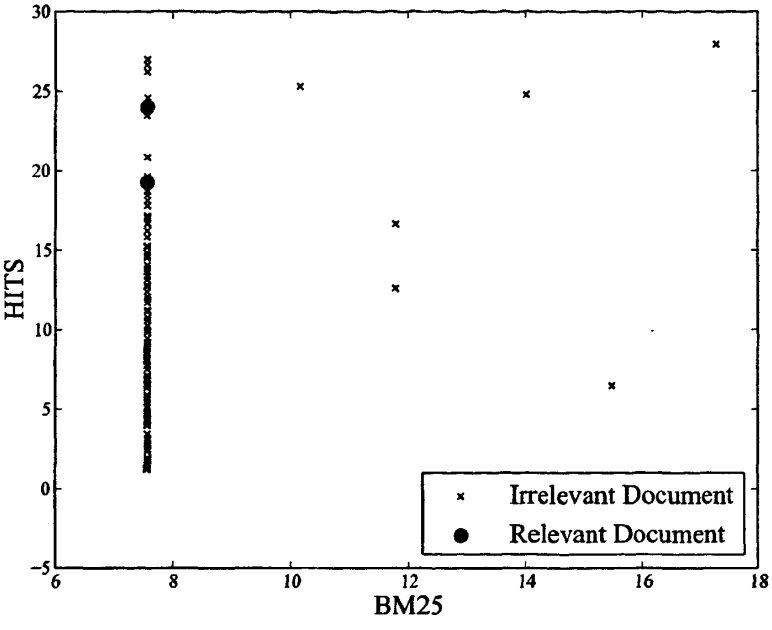

图 2-3 TD2004 数据集上查询 ID 为 97 的关联文档分布图

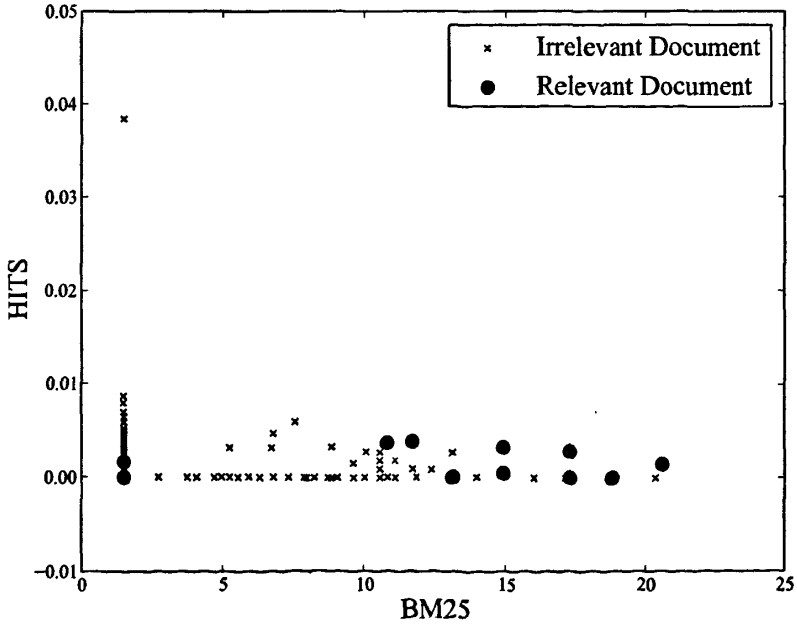

图 2-4 TD2004 数据集上查询 ID 为 192 的关联分布图

器以 [[HITS score > 19]] 为预测函数来对 ID 为 97 的查询的相关联文档列表进行排序,另一个排序器则以 [[BM25 score > 11]] 为预测函数来 ID 为 97 的查询的相关联文档列表进行排序。

本文的研究工作主要建立在 Kevin Duh 等人的工作之上,与他们将查询通过 Kernel PCA 方法映射到另一个低维向量空间上不同,我们的方法是通过利用查询与训练集的相似性信息来添加新的更加适合于该查询的特征,从而形成新的数据集进行训练。这种做法相当于把原来的查询映射到一个高维向量空间上,这在一定程度上增加了算法的时间复杂度,但实验结果表明我们的方法取得了比较好的效果。

在本文中我们采用的是一种与 Duh 等类似的直推式半监督排序学习框架。在后面的章节中,在不引起歧义的情况下我们将交替使用直推式排序学习算法与半监督排序学习算法这两个概念。

2.3 查询相似性信息

在网页搜索中,如何有效地利用用户给出的查询信息为用户提供更加精确的搜索结果,是当前研究人员面临的一个巨大挑战。

在真实的网络世界中,用户提交的查询是多种多样的。这些查询有长有短,有些查询可能会比较热门,检索系统返回的文档列表中会有很多的相关文档,而有一些查询可能比较冷门,只有很少的相关文档。对这些不同的查询使用同一个目标函数可能会导致预测性能的降低。目前存在的大部分全监督排序学习方法都是学习一个单一的目标函数,这对诸如文档排序等信息检索的任务来说或许并不是最好的,对不同的查询有针对性地采取不同的策略可能会带来更好的效果。

利用查询信息的一个比较直观的方法就是对查询进行归类。Uichin Lee 等[32]根据用户搜索目的和搜索需求的不同,将查询分成了导航型和信息型两类。对于导航型查询来说,用户事先在脑海中已经初步有了这个查询相对应的结果网页并希望被导航到这个页面去;而对信息型的查询来说,用户则是希望得到那些给出了包含自己查询背景信息的那些页面。In Ho Kang 等人[12]根据查询的分布以及查询在锚(Anchor)文本中的使用率等信息将用户的查询分成了主题相关型、主页查找型以及服务查找型三类。主题相关型的查询其实就是信息型查询,而主页查找型的查询就相当于导航型查询,服务查找型的查询则是一种事务查询,用户希望得到那些包含了自己在查询里面描述的服务的页面。

为了进一步推动研究人员在这方面的努力,国际上开展了许多这方面的比赛。

其中,KDD CUP 2005[33] 的比赛主题就是如何对互联网的搜索查询进行高效分类[2]。来自香港科技大学的 Dou Shen 等[34] 通过搜索引擎对查询和类别同时进行扩展再进行分类匹配,取得了本次大赛最好的成绩。

但是,简单地对查询进行分类并不能很好地发掘出查询的信息,而且查询分类本身也并不是一件容易的事。首先,自然语言中的词汇往往具有多义性,有一些单词可能同时属于多个类别;其次,用户的查询词通常都比较短,在有限的长度之内很难提取出有用的信息;最后,用户对同一个查询可能具有不同的目的性,单凭查询分类难以判断真实的用户需求。在排序模型的训练中引入查询关联文档的信息更有可能带来排序性能的提高。

Xiubo Geng 等[13] 通过对 TREC 2004 Web Track[35] 中的三类查询采用主成分分析法(Principle Component Analysis, PCA)进行降维后得到的二维散点图如图2-5所示。其中"•"代表命名网页查找(Named Page Finding),"*"代表主页查找(Homepage Finding),而"+"则代表主题提取(Topic Distillation)。

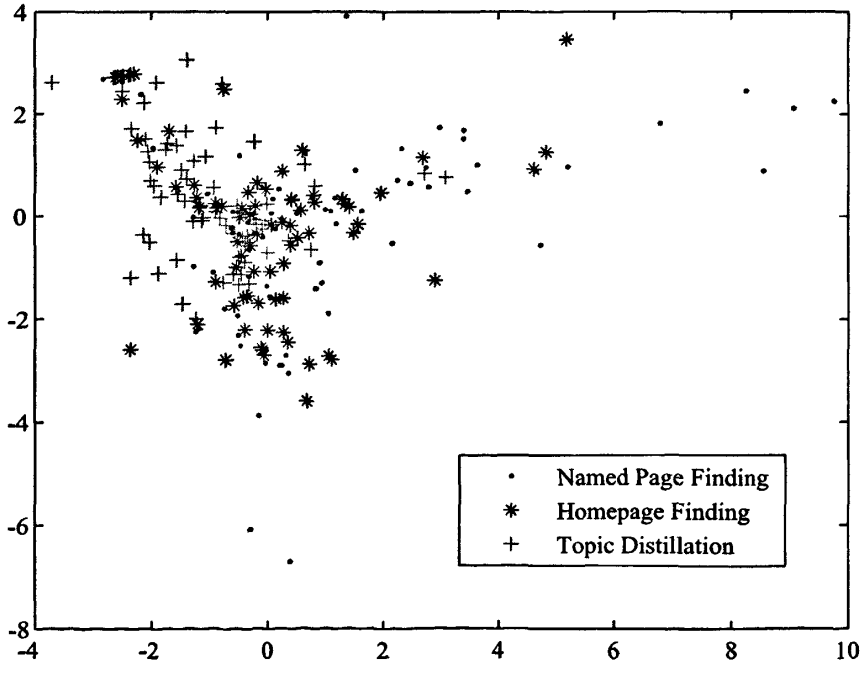

图 2-5 TREC2004 上的查询分类散点图[13]

[2] http://www.sigkdd.org/kdd2005/kddcup.html

从这个图中我们可以看到,不同的查询之间并不存在着明显的界限。但是,同一类的查询却往往是聚集在一起的。也就是说,一个查询的类别一般跟它周围的相邻查询的类别是一样的。这个性质被 Xiubo Geng 等称为查询的"局部性(Locality)"。基于这个观察,他们提出了一种基于 K 近邻(K-Nearest Neighbor,KNN)的查询依赖排序学习算法[13]。在他们的算法中,训练集中每一个查询都被映射到了特征空间中的一个点上。对于测试集中的每一个查询,通过把它映射到特征空间上去并找出与其相邻的 K 个训练集中的点来组成一个小规模的训练集对其进行训练。Xiubo Geng 等认为,这种依赖于查询的排序学习方法与传统的排序学习方法相比,存在着两方面的优势。首先,每一个查询的训练过程中都有针对性地利用了相似查询的信息,而且排除了不相似查询的干扰;其次,算法中采用了聚类的思想,并且由相似查询组成的训练集是动态生成的。实验结果表明,这种方法比使用单一排序模型具有显著的提高。

受查询"局部性"性质的启发,我们利用查询之间的距离来作为它们之间的相似性的度量。在 Xiubo Geng 等的欧氏距离之外,我们提出了一种与 Kendall's τ 距离[36] 类似的度量方法。这套度量方法主要用于比较两个查询之间不一致的特征维的数目。此外,我们还利用统计学里面的标准差的概念提出了一种新的表示查询向量的方法,这些方法与 Xiubo Geng 等提出的方法一起构成了本文利用查询相似性信息的主体。

第 3 章　基于查询相似性的直推式排序学习算法

正如上一章所讲的,查询之间的相似性信息对于排序模型的训练起着很大的作用。基于这个前提,在本章中我们提出了一种基于查询相似性的半监督排序学习算法,并对相关的细节进行了介绍。首先本章将介绍算法的基本框架以及基本的步骤;然后对算法里面提到的特征生成步骤作进一步的的阐述,包括了如何通过文档特征来定义查询特征,如何度量两个查询之间的相似性以及如何生成新的数据集;最后本章将会对实验里面用到的两个全监督排序学习算法进行详细的介绍。

3.1　算法框架

在本文中,我们提出了一种基于查询相似性的直推式排序学习算法。它的主要原理就是为测试集中的每一个查询学习一个独立的排序模型。这种方法关键的思想就是利用查询之间的相似性信息来为特定的查询样本产生更有助于排序的特征。为了描述的清晰性,下面先给出该算法的基本框架。

输入: 训练集 $S = \{(q_i, X_i, Y_i)\}_{i=1\cdots U}$
　　测试集 $T = \{(q_j, X_j)\}_{j=1\cdots V}$
　　全监督排序学习算法 **Learn( )**
　　相似查询子集的个数 K
输出: 测试集中每一个查询的排序预测 $\{Y_j\}_{j=1\cdots V}$

for $(q_j, X_j) \in T$ **do**
　从 S 中选择与查询 q_j 相似的查询组成 K 个不同的子集 $\Omega = \{\omega_k\}_{k=1\cdots K}$
　for $\omega_k \in \Omega$ **do**
　　利用 ω_k 训练出一个排序模型 h_k
　　用 h_k 分别为 (q_j, X_j) 和 S 产生新的特征 f_{k1} 和 f_{k2}
　end
　将 $\{f_{k1}\}_{k=1,\cdots,K}$ 和 $\{f_{k2}\}_{k=1,\cdots,K}$ 分别加入 (q_j, X_j) 与 S 形成 (q'_j, X'_j) 与 S'
　在新的数据集上利用算法 **Learn( )** 学习预测函数 $H_j(\cdot) = \text{Learn}(S')$
　利用预测函数 H_j 进行预测 $Y_j = H_j(q'_j, X'_j)$
end

算法 3-1 基于查询相似性的直推式排序学习算法框架

在算法3-1中,对于测试集中的每一个查询 q_j,算法通过各种不同的方法在训练集中寻找出那些与 q_j 相似的查询,并将这些相似的查询组合成 K 个不同的集合(第2行)。把这 K 个不同的集合分别作为训练集产生不同的排序学习模型(第4行),然后使用这些不同的模型分别为查询 q_j 以及训练集 S 产生新的特征(第5行),从而得到一个扩展的查询 (q'_j, X'_j) 和训练集 S'(第7行)。这个新的训练集 S' 是利用 q_j 的查询相似性信息构造的,那些新的特征能够更好地表达出查询 q_j。基于上一章的讨论,我们相信这个新的训练集训练出来的模型能够在查询 q_j 上获得更好的排序预测结果。

3.2　特征生成

在算法3-1中最重要的步骤就是针对测试集中的每一个查询 q_j 生成一些额外的特征,从而为 q_j 构造一个新的训练集 S'。这是本文提出的算法框架的一个创新点。正如我们在算法3-1里面第2行至第6行所描述的,在本文中我们利用查询之间的相似性信息来生成这些额外特征。

在详细介绍构造新的训练集之前,我们首先必须解决的问题就是如何为查询 q_j 选择它的相似查询。或者换句话说,如何对这些查询进行聚类。这里面涉及到了如何表示这些查询以及如何度量两个查询之间的相似性这两个关键的子问题。不同的查询表示方法以及不同的相似性度量方法组合在一起,形成了各种不同的选择相似查询的方法。

3.2.1　查询特征的表示

当我们要为一个查询 q 选择它的相似查询时,查询就成为了我们的研究对象。与其他的研究过程类似,我们要弄清楚的首要问题就是如何来有效地表述我们的研究对象。

在经典的排序学习实验设置中,数据集中所提供的查询信息在排序算法的训练过程中所起到的作用仅仅是用于标识两个不同文档是否有联系,即是否属于同一个查询。于是通常在排序学习的数据集只是简单地用一个整数 ID 来表述一个查询。这样的信息在我们依赖于查询的算法中是远远不够的,我们需要更多的有效性信息来表达数据集中的查询。

一个容易想到的方法就是利用数据集中与查询相关联的那些文档信息来表述一个查询。仿照传统排序学习设置中把一个文档映射到 n 维文档特征空间中一个点

的惯例,我们利用一个查询的相关联文档信息把这个查询映射到 n 维查询特征空间的一个点上。在本文中我们使用两种映射方法来达到这个目标。

第一种方法是由 XiuBo Geng 等人[13] 提出的。这种方法启发式地采用了 BM25 值作为一个推理模型(Inference Model)来选择一个查询 q 的前 ℓ 个排序文档,然后把这些文档的特征向量的平均值作为查询 q 的特征向量。更具体地讲,把与查询 q 相关联的文档列表 $X = \{x_i\}_{i=1,\cdots,m} = \{\langle f_1^{(i)}, f_2^{(i)}, \cdots, f_n^{(i)}\rangle\}_{i=1,\cdots,m}$ 里面的文档按照 BM25 特征值从大到小进行排序,然后再选择排在最前面的 ℓ 个文档 $X' = \{\langle f_1^{(i)'}, f_2^{(i)'}, \cdots, f_n^{(i)'}\rangle\}_{i=1,\cdots,\ell}$。这样查询 q 的特征向量就可以表示为:

$$q = \langle \frac{1}{\ell}\sum_{i=1}^{\ell} f_1^{(i)'}, \frac{1}{\ell}\sum_{i=1}^{\ell} f_2^{(i)'}, \cdots, \frac{1}{\ell}\sum_{i=1}^{\ell} f_n^{(i)'}\rangle \tag{3-1}$$

在本文中我们提出了另一种新的映射方法来表达一个查询,这种方法利用了统计学中的标准差这个统计量。一般来说,在排序学习模型中一个好的文档特征有助于我们把相关与不相关的文档有效地区分开来。正如 Kevin Duh 等[10] 的研究表明,不同的特征对于区分一个数据集中不同的查询所起到的作用是不同的。这就是说,一个查询的相关联文档的特征维的区分度有效地指示了该查询所属的分类。基于这个观察,我们选用了文档特征维的标准差这个统计量来表述一个特征的区分度,一个文档特征如果具有更大的标准差值,那么这个特征就对它相应的查询具有更好的区分度。我们认为如果两个查询在某个特征维上的区分度相似,那么这两个查询就越有可能落在同一个分类中。于是,文档特征维的标准差值便可以被视为一个查询的特征值。更具体地说,第 i 个文档特征 f_i 的平均值为 $\overline{f_i} = \frac{1}{m}\sum_{k=1}^{m} f_i^{(k)}$,相应的标准差为 $\sigma(f_i) = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(f_i^{(k)} - \overline{f_i})^2}$。于是查询 q 的特征向量就可以表示为:

$$q = \langle \sigma(f_1), \sigma(f_2), \cdots, \sigma(f_n)\rangle \tag{3-2}$$

必须注意的是,由于不同的文档特征值之间的差异是很大的,有些甚至相差几个数量级。如果某一个文档特征的值普遍偏大,那么即使这个特征值对于查询而言并没有很好的区分度,由这些值计算出来的标准差值还是会比较大。因此,我们必须对原始数据进行一些处理。通过归一化 (Normalize) 操作把文档特征值映射到 $[0,1]$ 区间,这样计算出来的标准差才更加合理,更具有可比性。

3.2.2 查询相似性的度量

一个定义在集合 $\mathscr{X}$ 上的度量 (Metric) 是一个函数 $d: \mathscr{X} \times \mathscr{X} \to \mathbb{R}$,有时候也简称为距离函数或者距离。$(\mathscr{X}, d)$ 称为度量空间或者距离空间。

反过来，一个定义在 $\mathscr{X}$ 上的函数 d 要成为一个度量，必须满足以下四个性质[37] $(\forall x_i, x_j, x_k \in \mathscr{X})$：

- $d(x_i, x_j) \geq 0$;
- $d(x_i, x_j) = 0 \iff x_i = x_j$;
- $d(x_i, x_j) = d(x_j, x_i)$;
- $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$。

在任意一个非空集合上都可以定义距离，使之称为一个度量空间。常见的度量包括了数学中常用的欧氏距离以及计算机科学中用于编码检错与纠错的海明距离。

在本文我们使用的第一种度量就是欧氏距离。假设一个查询 q 的特征向量为 $q = \langle \dot{f}_1, \dot{f}_2, \cdots, \dot{f}_n \rangle$，那么查询 q_i 与 q_j 之间的距离就是：

$$d_1(q_i, q_j) = \sqrt{(\dot{f}_1^{(i)} - \dot{f}_1^{(j)})^2 + (\dot{f}_2^{(i)} - \dot{f}_2^{(j)})^2 + \cdots + (\dot{f}_n^{(i)} - \dot{f}_n^{(j)})^2} \tag{3-3}$$

显然，两个查询特征向量之间的距离 d_1 越小，相似性越高。

除此之外，我们还引入一种类似 Kendall's τ 距离[36] 的度量。Kendall's τ 距离通过计算两个排列之间不一致的元素对数目来度量两个排列的相似程度。在计算机科学中，Kendall's τ 距离有时也被称作冒泡排序距离，因为 Kendall's τ 距离与使用冒泡排序算法把一个排列转换成另一个排列所需要的交换次数相等。

假设有排列 τ_1 和 τ_2，如果元素 i 与 j 在排列 τ_1 与 τ_2 中的先后顺序不一致，那么我们称元素对 (i, j) 在 τ_1 和 τ_2 上不一致。例如对于排列 $\tau_1 = \langle 4, 5, 3, 1 \rangle$ 和 $\tau_2 = \langle 1, 4, 3, 5 \rangle$，在 τ_1 中元素 4 排在了 1 的前面，而在 τ_2 中则相反，我们称元素对 $(4, 1)$ 在 τ_1 和 τ_2 上是不一致的。同理可以知道元素对 $(4, 5)$ 是一致的，并且可以算出 τ_1 与 τ_2 之间的 Kendall's τ 距离为 4。显然，Kendall's τ 距离越大，τ_1 与 τ_2 之间的相似度越低。

具体到查询特征向量的问题，我们定义两个查询之间的距离为两个查询特征向量之间不一致的特征对的数目。一个特征对在两个查询上是一致的当且仅当特征对中的两个特征在两个查询中具有相同的大小关系。即：

$$d_2(q_i, q_j) = \left| (\dot{f}_s, \dot{f}_t) : s < t, (\dot{f}_s^{(i)} - \dot{f}_t^{(i)}) * (\dot{f}_s^{(j)} - \dot{f}_t^{(j)}) < 0 \right| \tag{3-4}$$

其中 $|\cdot|$ 是一个规模函数(Size Function)，表示里面的元素的数目。

例如对查询特征向量 $q_1 = \langle 0.58, 0.49, 0.03, 0.88 \rangle$ 和 $q_2 = \langle 0.26, 0.11, 0.39, 0.31 \rangle$，特征对 $(\dot{f}_1, \dot{f}_2)$ 就是一致的，因为在 q_1 中第一个特征值 $\dot{f}_1^{(1)} = 0.58$ 比第二个特征值

$\dot{f}_2^{(1)} = 0.49$ 大，而在 q_2 中也是这样（$\dot{f}_1^{(2)} = 0.26 > \dot{f}_2^{(2)} = 0.11$）。同理可以得出特征对 $(\dot{f}_2, \dot{f}_3)$ 是不一致的，并可以算出 q_1 和 q_2 的距离为 3。两个查询特征向量之间的距离 d_2 越大，相似性越低。

3.2.3 生成特征

前面我们已经介绍了两种表述查询特征向量的方法与两种度量查询特征向量相似度的方法。实际上，这些方法之间可以相互任意的组合。于是，对于一个查询 q，我们一共就有了 $4 = 2 \times 2$ 种为它挑选那些相似查询的方法。

在半监督算法3-1中，我们利用上面提到的 4 种挑选相似查询的方法从训练集 S 中为测试集 T 中的每一个查询 q 挑选出 K 个由它的相似查询组成的集合，显然这些集合都是 S 的子集。接着，我们用这 K 个子集分别训练排序模型来为 q 排序，从而得到了与 q 相关联的文档列表 X 的 K 个排序预测分数 $\{\hat{Y}_k\}_{k=1,\cdots,K}$。最后，我们把 X 中每个文档的每一个预测分数值都作为该文档的一个额外特征添加进该文档的特征向量中。从而我们得到了一个扩充的文档列表 $X' = \{x_i'\}_{i=1,\cdots,m} = \{\langle f_1^{(i)}, f_2^{(i)}, \cdots, f_n^{(i)}, f_{n+1}^{(i)}, \cdots, f_{n+K}^{(i)} \rangle\}_{i=1,\cdots,m}$。这个操作的结果就是查询 q 的每一个关联文档都增加了 K 个特征。为了能够在算法3-1的最后利用全监督排序学习算法 **Learn**( ) 来为查询 q 进行预测，我们也必须用同样的排序模型来为训练集中的所有文档都添加 K 个相应的额外特征。

最终我们得到了一个扩充后的训练集 S' 与查询 q'，由于我们添加的 K 个额外特征是针对 q 利用查询之间的相似性信息生成的，我们认为这个新的训练集 S' 能够提高对 q' 排序的准确率，也即是提高了查询 q 的排序准确率。

3.3 RankBoost 与 SVM^{map}

从理论上讲，任何一个全监督的排序学习算法都可以用在算法3-1中的 Learn( ) 函数中（第 8 行）。为了验证我们提出的半监督排序学习算法的有效性，我们采用了两个代表当前全监督排序学习水平的算法：RankBoost 和 SVM^{map}。

3.3.1 RankBoost

RankBoost[21] 是一种基于 Boosting 的排序学习算法，更具体的说，RankBoost 是基于 Freund 和 Schapire 的 AdaBoost[24] 算法以及 Schapire 和 Singer 提出的后续改进算法[38] 的。它是当前最好的 Pairwise 型的排序学习算法之一。

RankBoost 的核心思想就是把排序学习问题形式化成基于样本对的分类问题，然后利用 Boosting 算法迭代进行求解。与 AdaBoost 基于样本的分类不同，RankBoost 是基于样本对的。

对于训练集值 $S=\{(q_i,X_i,Y_i)\}_{i=1,\cdots,U}$ 中的任意一个文档对 (x_0,x_1)，RankBoost 引入了一个反馈函数(Feedback Function) $\Phi:\mathscr{X}\times\mathscr{X}\to\mathbb{R}$ 来度量文档 x_1 是否应该排在 x_0 的前面。其中，$\Phi(x_0,x_1)>0$ 意味着 x_1 应该排在 x_0 的前面而 $\Phi(x_0,x_1)<0$ 则相反；当 $\Phi(x_0,x_1)=0$ 时说明在文档 x_0 与 x_1 之间没有先后关系。一个好的排序模型的排序结果应该尽量与反馈函数的结果一致。

为了度量排序结果与反馈函数的一致性，RankBoost 在 S 中所有正的文档对(即反馈函数的输出结果为正数的那些文档对)上面定义了一个分布 $D:\mathscr{X}\times\mathscr{X}\to\mathbb{R}$：

$$\sum_{x_0,x_1}D(x_0,x_1)=1,\ \text{where}\ D(x_0,x_1)=c\cdot max\{0,\Phi(x_0,x_1)\} \tag{3-5}$$

其中 c 是一个使得 D 成为一个分布的正常数。分布 D 体现了一个文档对在排序中的重要程度：D 在一个文档对上赋予的值越高，则在下一次排序时弱学习器排正确这个文档对的重要性就越高。

RankBoost 算法的最终目标就是使得它的排序结果中文档对错误率最小化。这个文档对的错误率可以通过一个损失函数来体现。假设 RankBoost 算法最终学习到的排序预测函数为 $h:\mathscr{X}\to\mathbb{R}$，结合前面的定义，我们可以给出该排序模型的排序损失函数 L 如下：

$$L^D(h)=\sum_{x_0,x_1}D(x_0,x_1)[[h(x_1)\le h(x_0)]]=\Pr_{(x_0,x_1)\sim D}[[h(x_1)\le h(x_0)]] \tag{3-6}$$

显然，损失函数 $L^D(h)$ 描述了当文档对 (x_0,x_1) 服从分布 D 时排序模型 h 排错的文档对的“加权数量”。这个定义也体现了上面所说的分布 D 的意义，它使得 RankBoost 在下一次迭代时会挑选那些把上一次排序错误的文档对给排正确的弱学习器，从而在多次迭代之后降低文档对的排序错误率。

RankBoost 算法的基本框架如算法3-2所示。

从算法3-2中我们可以看到，在每一次迭代中 RankBoost 学习一个新的弱学习器 $h_t:\mathscr{X}\to\mathbb{R}$ 来获得一个弱排序结果，虽然叫做弱学习器，它的功能与一般的排序预测模型是一样的，都是给出了对文档列表排序的预测信息。根据当前得到的弱排序结果，RankBoost 算法对分布 D_t 进行更新。

输入: 初始化的分布 D

训练集中的文本对

输出: 排序预测函数 h

初始化 D_1 为 D

for $t \leftarrow 1$ **to** T **do**

　利用权值分布 D_t 训练出一个弱学习器 h_t

　采用 h_t 进行预测 $h_t : \mathscr{X} \rightarrow \mathbb{R}$

　选择系数 $\alpha_t \in \mathbb{R}$

　更新权值分布 $D_{t+1} = \frac{1}{Z_t} \cdot D_t(x_0, x_1) \cdot \exp(\alpha_t(h_t(x_0) - h_t(x_1)))$，其中 Z_t 是使得 D_{t+1} 成为一个分布的正规化因子

end

输出最终的预测函数: $h(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

算法 3-2 RankBoost 排序学习算法

假设 (x_0, x_1) 是上一次迭代时被标记为重要的文本对，我们希望的结果是把 x_1 排在 x_0 的前面。假设当前系数 $\alpha_t > 0$，那么在当前这一轮迭代中，如果 h_t 给出了 x_0 与 x_1 的正确排序关系，即 $h_t(x_1) > h_t(x_0)$，那么 $D_{t+1}(x_0, x_1) < D_t(x_0, x_1)$，文档对的重要性就下降了。相反，如果 h_t 给出了错误的排序预测 $h_t(x_1) < h_t(x_0)$，那么 $D_{t+1}(x_0, x_1) > D_t(x_0, x_1)$，文档对的重要性继续被提高了。最终 RankBoost 会越来越注重这些很难分正确的文档对的排序。

Yoav Freund 等[21] 证明了损失函数 $L^D(h)$ 存在着一个上界:

$$L^D(h) \leq \prod_{t=1}^{T} Z_t \tag{3-7}$$

由公式(3-7)我们知道，要使得预测函数 h 的错误率最小，只需要在算法的每一次迭代中选择一个系数 α_t 来最小化正规化因子 Z_t:

$$Z_t = \sum_{x_0, x_1} D_t(x_0, x_1) \cdot \exp(\alpha_t(h_t(x_0) - h_t(x_1))) \tag{3-8}$$

在 Yoav Freund 等[21] 的论文中介绍了三种方法来解决这个问题。

由于 RankBoost 是在 AdaBoost 的基础上发展起来的，因此，RankBoost 继承了 AdaBoost 的许多好的性质:特征选择，训练时收敛以及一定程度上泛化的能力。这使得 RankBoost 在排序学习领域有着很广泛的应用。

3.3.2 SVM^{map}

SVM^{map}[25]是建立在通用SVM框架上的一种排序学习算法。它采用了SVM^{struct}[29]框架来直接优化IR评价标准MAP。SVM^{map}是当前最好的Listwise型排序学习算法之一。

假设$X=\{x_i\}_{i=1,\cdots,m}$是与训练集中查询q相关联的文档列表,它相应的标注为$Y=\{y_i\}_{i=1,\cdots,m}$(其中$y_i=1$表示相应的文档被标注为相关,而$y_i=0$则表示被标注为不相关)。与所有的SVM算法类似,SVM^{map}的目标就是学习一个(文档列表)输入空间$\mathcal{X}^m$与(标注列表)输出空间$\mathcal{Y}^m$之间的映射函数$h:\mathcal{X}^m\to\mathcal{Y}^m$。为了量化排序模型的质量$\hat{Y}=h(X)$,SVM^{map}引入了一个损失函数$\Delta:\mathcal{X}^m\times\mathcal{X}^m\to\mathbb{R}$。$\Delta(Y,\hat{Y})$给出了一个应该标注为$Y$的文档列表被排序模型标注为$\hat{Y}$时的惩罚。通过在这个损失函数中加入IR评价标准就可以直接在训练过程中对它们进行优化。在SVM^{map}中Yisong Yue等[25]引入了MAP作为一个性能指标来定义损失函数:$\Delta_{map}(Y,\hat{Y})=1-\mathrm{MAP}(Y,\hat{Y})$,其中$\mathrm{MAP}(Y,\hat{Y})$为一个正确标注列表为$Y$的文档列表$X$被排序模型标注为$\hat{Y}$时所得到的MAP值。

基于上述定义,SVM^{map}算法的优化目标就是找到一个函数h,使得排序模型在训练集S上的风险(也就是损失的期望值):

$$L_S^{\Delta}(h)=\frac{1}{U}\sum_{i=1}^{U}\Delta_{map}(Y_i,h(X_i)) \tag{3-9}$$

最小化。其中U为训练集中查询的数量。

假设h是一个以权值w为参数的函数,于是SVM^{map}的目标就转化为寻找一个参数w使得$L_S^{\Delta}(w)\equiv L_S^{\Delta}(h(\cdot;w))$最小,这可以通过学习一个判别式函数(Discriminant Function)$F:\mathcal{X}^m\times\mathcal{Y}^m\to\mathbb{R}$来实现。对于一个给定查询$q$的文档序列$X$,我们可以挑选一个预测$\hat{Y}$使得$F$最大化:

$$h(X,w)=\arg\max_{\hat{Y}\in\mathcal{Y}^m}F(X,\hat{Y};w) \tag{3-10}$$

进一步的,SVM^{map}假设判别式函数F是一些用于表达输入输出的组合特征函数(Combined Feature Function)的线性函数:

$$F(X,\hat{Y};w)=w^T\Psi(X,\hat{Y}) \tag{3-11}$$

其中组合特征函数 Ψ 被定义为:

$$\Psi(X,\hat{Y}) = \sum_{i:x_i \in X^r} \sum_{j:x_j \in X^{\bar{r}}} [(\hat{y}_i - \hat{y}_j)(x_i - x_j)] \tag{3-12}$$

这里 X^r 表示 X 中相关文档的集合,而 $X^{\bar{r}}$ 则表示 X 中不相关文档的集合。

于是,SVM^{map} 的目标问题可以转化为求解如下的优化问题:

$$\min_{w,\xi \geq 0} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{3-13}$$

$s.t.\quad \forall i, \forall Y \neq Y_i$

$$w^T\Psi(X_i,Y_i) \geq w^T\Psi(X_i,Y) + \Delta(Y_i,Y) - \xi_i \tag{3-14}$$

这是一个经典的 SVM 优化问题,SVM^{map} 采用了 SVM^{struct} 算法中的切割平面(Cutting Plane)算法[39] 来解决这个问题。

输入: 训练集 $S = \{(q_i, X_i, Y_i)\}_{i=1,\cdots,U}$
　　参数 C
　　允许误差 ϵ
输出: 排序预测函数 H

初始化所有的工作集 $\mathcal{W}_i$ 为空集: $\mathcal{W}_i \leftarrow \emptyset$ for all $i = 1, \cdots, U$
repeat
　for $i \leftarrow 1$ **to** U **do**
　　$H(Y;w) \equiv \Delta(Y_i,Y) + w^T\Psi(X_i,Y) - W^T\Psi(X_i,Y_i)$
　　计算 $\hat{Y} = \arg\max_{Y \in \mathcal{Y}^m} H(Y;w)$
　　计算 $\xi_i = \max\{0, \max_{Y \in \mathcal{W}_i} H(Y;w)\}$
　　if $H(\hat{Y};w) > \xi_i + \epsilon$ **then**
　　　$\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{Y}\}$
　　　$w \leftarrow$ 在集合 $\mathcal{W} = \bigcup_i \mathcal{W}_i$ 上求解优化问题(3-13)得到的结果 w
　　end
　end
until 在本次迭代过程中所有的 $\mathcal{W}_i$ 都没有被改变;

算法 3-3 SVM^{map} 排序学习算法

算法3-3给出了 SVM^{map} 的算法流程。SVM^{map} 优化问题中的目标函数具有着数目为指数级别的约束条件(如公式3-14所示)。为了解决这个问题,SVM^{map} 维护了一个工作集 $\mathcal{W}$,这个工作集仅仅包含了那些具有最大违反的约束(Most Violated Constrains)。其中,违反(Violation)被定义为:Voilation $\triangleq \Delta(Y,\hat{Y}) + w^T\Psi(\hat{Y},X)$。由于优化过程仅仅在工作集上进行,于是算法3-3可以很快地找出那些具有最大违反的约束。这个步骤里面运用到了 MAP 评价标准的一个性质:如果一个序列的每一个位置的相关程度都是固定的,那么无论什么文档出现在这个位置,MAP 的值都不会变。更进一步地说,如果把这些文档按照排序预测分数降序排序的话,那么在 MAP 值不变的情况下, $w^T\Psi(\hat{Y},X)$ 会取得最大值。根据这个性质,SVM^{map} 设计了一个时间复杂度为 $O(m\log m)$ 的算法来迅速找出具有最大违反的约束,其中 m 为与查询 q 相关联的文档的数目。

实验结果证明,SVM^{map} 至少可以具有与其他基于 SVM 的排序学习算法同样的性能,而且可以很好地扩展到其他不同的 IR 评价标准中。最近,一些研究人员在 SVM^{map} 的基础上设计出了诸如 SVM^{ndcg}[40] 与 SVM^{mrr}[40] 之类的算法。这些算法尽管在损失函数定义、寻找最大违反约束等方面与 SVM^{map} 存在着不同之处,但总体的算法框架与原理都是类似的。

3.4　离线优化

在算法3-1中,对于测试集 T 上的查询 q_j,我们从训练集 S 中为其选择了 K 个不同的相似查询子集 $\Omega = \{\omega_k\}_{k=1\cdots K}$,并且用这些子训练集 ω_k 分别为 (q_j,X_j) 和 S 中的所有文档生成了新的文档特征,最后再在那些添加了新文档特征的数据集上面进行学习与预测。

这是一种在线(Online)学习的策略。对于给定的查询 q_j,我们利用查询之间的相似性信息将其与相应的训练集中的查询映射到一个更高维的特征向量空间中进行学习预测的做法无疑降低了算法的效率,从而增加了对 q_j 进行预测的响应时间,而这个时间在实际应用中通常是不可接受的。为此,我们可以将一些能够提前计算的过程(Procedure)进行预处理以降低对给定查询的预测响应时间,提高用户体验。

算法3-1的第 4 行和第 5 行中利用相似查询子集 ω_k 作为训练集来学习排序模型 h_k 并分别为 (q_j,X_j) 和 S 进行预测。显然,这一步里面的模型学习以及利用 h_k 为 S 生成新特征的过程都是可以预先计算的。假设一个子集 ω_k 里面包含了 ℓ_k 个 S 中的查询(ℓ_k 一般是一个比较小的正整数),我们可以预先计算出 S 中所有由 ℓ_k 个查询

组成的子训练集相应的排序模型，并利用这些模型来得到 S 的那些相应的新文档特征 f_{k2}。这样对于一个给定的测试集里面的查询 q_j，我们只需要找出那些相似查询组成的子集 $\{\omega_k\}_{k=1\cdots K}$（这一步所需要的时间相对于训练过程来说是非常少的，可以忽略不计），就可以很快地定位出相应的预测模型 $\{h_k\}_{k=1\cdots K}$ 和 $\{f_{k2}\}_{k=1\cdots K}$，完成算法3-1中第 4 行和第 5 行的主要运算过程。

同样的优化思路还可以进一步应用于算法3-1的第 8 行中。由于训练集 S 中的那些新添加的文档特征 f_{k2} 是可以预先计算的，那么由这些新文档特征扩展后的训练集 S' 所对应的排序模型也是可以进行预处理的。这样一来对于一个给定的查询 q_j，我们就可以很快地通过找到它对应的训练集的排序模型来得到它的预测结果。

输入　：训练集 $S = \{(q_i, X_i, Y_i)\}_{i=1\cdots U}$
测试集 $T = \{(q_j, X_j)\}_{j=1\cdots V}$
全监督排序学习算法 **Learn( )**
相似查询子集的个数 K
相似查询子集包含的查询个数 $\{\ell_k\}_{k=1,\cdots,K}$

输出　：测试集中每一个查询的排序预测 $\{Y_j\}_{j=1\cdots V}$

预处理: ▷ 计算出 S 中所有由 ℓ_k 个查询组成的子训练集相应的排序模型
▷ 利用这些模型为 S 生成相应的新文档特征 f_{k2}
▷ 利用算法 **Learn( )** 得到所有添加了 K 个新文档特征的训练集 S 的那些排序模型

1 **for** $(q_j, X_j) \in T$ **do**
2　从 S 中选择与查询 q_j 相似的查询组成 K 个不同的子集 $\Omega = \{\omega_k\}_{k=1\cdots K}$
3　**for** $\omega_k \in \Omega$ **do**
4　　从预处理结果中定位出 ω_k 对应的排序模型 h_k 与文档特征 f_{k2}
5　　用 h_k 为 (q_j, X_j) 生成新的文档特征 f_{k1}
6　**end**
7　将 $\{f_{k1}\}_{k=1,\cdots,K}$ 加入 (q_j, X_j) 形成新的待预测查询 (q'_j, X'_j)
8　从预处理结果中定位出加入了 $\{f_{k2}\}_{k=1,\cdots,K}$ 的 S 相对应的预测函数 H_j
9　利用预测函数 H_j 进行预测 $Y_j = H_j(q'_j, X'_j)$
10 **end**

算法 3-4 基于查询相似性的离线直推式排序学习算法框架

算法3-4给出了我们提出的离线(Offline)学习策略的算法框架。这种离线学习的策略实际上是对算法3-1的时间与空间效率的一种折中 (Trade-Off)。通过缓存算法3-1中一些可以预处理得到的排序模型来减少算法的实时响应时间,从而得到一个比较好的时间效率。虽然算法3-4在离线预处理部分所需要的时间与空间复杂度都比较高,但是考虑到在实际应用(如搜索引擎)中,系统对离线训练部分的性能要求远远低于在线预测部分,我们提出的离线算法框架在实际应用中还是可行的。

第4章 实验及分析

本章将对实验部分进行详细的介绍,并通过实验结果的比较与分析来验证前面提到的基于查询相似性的半监督改进算法的有效性。首先我们将介绍用于实验的公开数据集 LETOR,接着是用于实验结果分析的评价标准 MAP 和 NDCG,然后是实验细节的设置,最后是实验结果的比较与分析。

4.1 数据集

本文在微软亚洲研究院发布的 LETOR[41] 数据集上进行实验[1]。LETOR 数据集是在多个被广泛用于信息检索领域的资料集以及查询集的基础上构造出来的。这些资料中的文档在一些仔细设计的策略下被采样,接着通过抽取文档中的特征和元数据(Metadata)形成一系列的查询 -文档对。

LETOR 中的文档来自于两个资料集:"Gov"数据集(也称为 TREC 数据集)以及 OSHUMED 数据集。

4.1.1 TREC 数据集

TREC 数据集中的文档来自于爬虫程序在 2002 年 1 月份对".gov"域的网页进行抓取的结果,这里面包含了 TREC2003 和 TREC2004 两个子集。LETOR 中的 TD2003 和 TD2004 数据集分别来自 TREC2003 和 TREC2004 的主题提取(Topic Distillation)任务以及相应的相关性标注。表4-1 显示了 TD2003 和 TD2004 中的一些统计数据。

表 4-1 TD2003 以及 TD2004 数据集的统计数据

数据集	查询数目	检索文档数目	相关文档数目
TD2003	50	49,175	516
TD2004	75	74,170	1,600

TD2003 和 TD2004 数据集对每一个文档都分别抽取了 44 个同样的特征。这些特征包括了以下四类:

1. 低级内容特征:这类特征包括词频(tf),反向文档频率(idf),文档长度(dl)以及

[1] http://research.microsoft.com/users/tyliu/LETOR/

这几者的结合(如 *tf-idf* 等)。对于每一个这样的特征,在数据集中都有4个相应的特征值,分别对应于网页的4个域:主题,锚,标题和URL。

2. 高级内容特征:这类特征包括BM25算法以及LMIR算法作用在文档上的输出值。通过采取不同的文档域及不同的平滑方法(DIR,JM,ABS),TD2003和TD2004对每一个文档都抽取了9个相同的语言模型特征。
3. 超链接特征:这类特征包括了PageRank,HITS以及它们的变种(HostRank,Topical PageRank和Topical HITS)。其中HITS和Topical HITS这两种特征又分别具有Authority和Hub这两种分数,于是TD2003和TD2004一共抽取了7个超链接特征。
4. 混合特征:这类特征包含了内容和超链接两者的信息。例如:基于超链接的相关性传递(Hyperlink-based Relevance Propagation)以及基于网站地图的相关性传递(Sitemap-based Relevance Propagation)。

每一个查询-文档对都有一个手工标注的二值相关性标注:相关(Relevant)与不相关(Irrelevant)。经过采样抽取特征后的数据集格式示意图如图4-1所示。数据集中的每一行分别描述了一个查询-文档对信息,这些信息从左到右依次为相关性标注、查询ID、44个特征ID与相应的值以及文档ID。

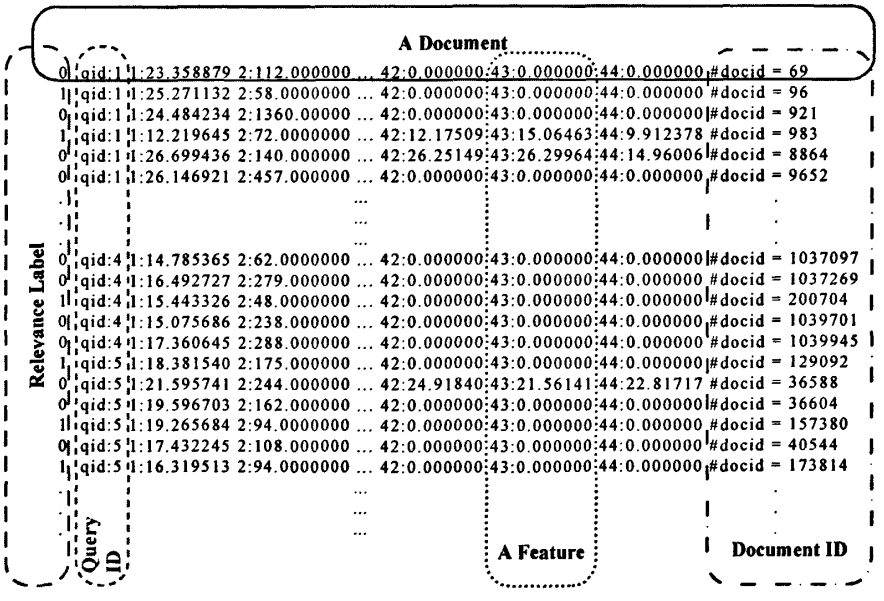

图4-1　TD2003(TD2004)数据集的示例图

4.1.2 OHSUMED 数据集

OHSUMED 数据集中一共包含了近 1.6 万个来自医学数据库 MEDLINE 的记录,这些记录对应于 106 个描述医学搜索需求(Medical Search Need)的查询。

OHSUMED 数据集中的每一个文档包含了 25 个文档特征,其中低级内容特征 20 个,高级内容特征 5 个。这些文档由人工标注为 3 个相关程度:完全相关,部分相关以及不相关。

由于 OHSUMED 数据集中每一个查询所包含的关联文档数目都比较少,根据本文提出的算法框架很难生成有效的额外特征来加强排序效果。所以我们只是在 TREC 数据集上对本文提出的算法进行了实验。

4.1.3 交叉校验

在 LETOR 数据集中,TD2003(TD2004)的所有文档被分成了五个子集 S_1, S_2, S_3, S_4 和 S_5。这五个子集被用于进行五重交叉校验(5-Fold Cross Validation),从而避免过度拟合等问题。用于交叉校验的五个子集划分如表4-2所示。

表 4-2 用于交叉验证的 TD2003(TD2004) 数据集划分

子目录	训练集	测试集	校验集
Fold1	$\{S_1, S_2, S_3\}$	S_4	S_5
Fold2	$\{S_2, S_3, S_4\}$	S_5	S_1
Fold3	$\{S_3, S_4, S_5\}$	S_1	S_2
Fold4	$\{S_4, S_5, S_1\}$	S_2	S_3
Fold5	$\{S_5, S_1, S_2\}$	S_3	S_4

在每一次的实验中,三个子集被当成训练集用于训练,一个子集用于校验,一个子集用于测试。训练集用于训练排序模型,校验集用于调整排序模型中的参数(如神经网络学习算法的迭代次数,支持向量机算法的目标函数中的系数等),而测试集则用于检验排序模型的性能。这样对于一个原始训练集,我们做了五组不同的实验。将这五组实验的结果进行平均我们就得到了排序学习模型的最终性能评价结果。

进行交叉校验有助于更好地评价排序模型的泛化能力,特别是在训练数据集规模比较小或者是参数数目比较多的情况下更加有效。通过采取交叉校验的方法,我

们可以估计一个排序模型与那些和用于训练模型的数据集相互独立的数据集之间的拟合水平。

4.2 评价标准

对检索结果的评价是信息检索领域的一个关键问题。借助于查询与文档相关性的判断,人们提出了多种方法用于评价排序模型的性能。

目前在信息检索领域存在着几个已经被广泛接受的评价标准,最基本的就是查准率(Precision)和召回率(Recall)。在这两个评价标准的基础上,人们进一步提出了MRR(Mean Reciprocal Rank),MAP(Mean Average Precision),RC(Rank Correlation)以及NDCG(Normalized Discounted Cumulative Gain)等用于对排序结果进行评价的标准。在LETOR数据集中使用的是MAP和NDCG这两个评价标准。

4.2.1 MAP

MAP是反映排序模型在全部的相关文档上的性能的单值标准。如果一个排序模型的排序结果将更多的相关文档排在前面,那么该排序模型将可能获得更高的MAP值。

对于一个查询 q_i,排序模型的排序结果 $X_i = \langle x_1^{(i)}, x_2^{(i)}, \cdots, x_m^{(i)} \rangle$ 在位置 j 的查准率 $\mathrm{P}(i,j)$ 定义为在 X_i 前 j 个文档中查询 q_i 的相关文档所占的比例:

$$\mathrm{P}(i,j) = \frac{\sum_{k=1}^{j} [[x_k^{(i)} \in X_i^r]]}{j} \tag{4-1}$$

其中 X_i^r 是由查询 q_i 的所有相关文档组成的集合,$X_i^r \subset X_i$。

相应的,查询 q_i 的平均查准率就定义为排序模型的输出结果在所有相关文档位置上的查准率的均值:

$$\mathrm{AP}(i) = \sum_{j=1}^{m} \frac{\mathrm{P}(i,j) * [[x_j^{(i)} \in X_i^r]]}{|X_i^r|} \tag{4-2}$$

例如,对查询 q_i,排序模型的结果为 $\langle +, -, -, +, +, - \rangle$("+"代表相关,"−"代表不相关),$|X_i^r| = 3$。那么 $\mathrm{P}(i,1)$ 到 $\mathrm{P}(i,6)$ 分别为 $\langle 1.0, 0.5, 0.33, 0.5, 0.6, 0.5 \rangle$,所以查询 q_i 的平均查准率 $\mathrm{AP}(i) = (1.0 + 0.5 + 0.6)/3 = 0.7$。一个查询的理想排序结果的AP值为1。

MAP即为所有查询的平均查准率的平均值:

$$\mathrm{MAP} = \frac{\sum_{i=1}^{|Q|} \mathrm{AP}(i)}{|Q|} \tag{4-3}$$

其中 $|Q|$ 为查询的数目。

4.2.2 NDCG

经验表明,MAP 在排序模型性能评价方面具有很好的区分度和稳定性。但是,MAP 评价标准也存在一个致命的缺陷,那就是它把查询跟文档的相关性视为简单的二值关系,即一个查询和一个文档要么相关,要么不相关。而在现实生活当中,我们可能会赋予一个文档与查询不同的相关程度,如不相关,部分相关,相关,很相关,完全相关等。我们希望排序模型能够区分出不同的相关程度,同时将相关程度更高的文档排在更前面。显然,MAP 不能完全胜任这种排序模型的性能评价。于是研究人员们引入了一种新的评价标准——NDCG[42],它对传统的二值评价标准进行了改进。

NDCG 的改进主要基于以下两个原则:

1. 相关程度很高的文档比一般相关的文档具有更高的价值;
2. 排得越靠后的相关文档的价值越低,因为用户一般不会访问到它。

由于综合考虑了相关性等级和排序位置的影响,对查询 q_i,排序结果 X_i 在位置 j 的 NDCG 值为:

$$\mathrm{NDCG}(i,j) = Z_j \sum_{k=1}^{j} \frac{2^{r(i,k)} - 1}{log(1+k)} \tag{4-4}$$

其中 $r(i,k) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ 为文档 $x_k^{(i)}$ 相对于查询 q_i 的相关程度,Z_j 是一个归一化因子,它使得最优的排序检索结果在位置 j 处计算得到的 $\mathrm{NDCG}(i,j)$ 为 1。

与 MAP 类似,在位置 j 的 NDCG 值为所有的查询在位置 j 的平均值,即:

$$\mathrm{NDCG}(j) = \frac{\sum_{i=1}^{|Q|} \mathrm{NDCG}(i,j)}{|Q|} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} Z_j \sum_{k=1}^{j} \frac{2^{r(i,k)} - 1}{log(1+k)} \tag{4-5}$$

下面以一个简单的例子来说明 NDCG 值的计算过程。

假设排序模型将文档与查询的相关程度分为 3 个级别:0、1 和 2(0 代表不相关,1 代表部分相关,2 代表相关)。一个文档序列的排序结果 X 对应的相关程度为 $V = \langle 2,0,1,2,1,0 \rangle$。

每个位置的相关程度可以用来衡量 X 在该位置的增益(Gain)。一般来说,相关程度为 x 的文档贡献的增益定义为 $(2^x - 1)$。采用这个定义,文档序列 X 对应的增益序列为 $G = \langle 3,0,1,3,1,0 \rangle$。

X 在位置 j 的增益可以与排在其前面的所有文档的增益进行累加,从而得到位置 j 的累加增益(Cumulative Gain,CG)。假如位置 j 的文档累加增益用 $\mathrm{CG}[j]$ 表示,

那么 CG 可以递归的定义为:

$$\mathrm{CG}[j] = \begin{cases} \mathrm{G}[1] & \text{if } i = 1, \\ \mathrm{CG}[j-1] + \mathrm{G}[j] & \text{if } i > 1. \end{cases} \tag{4-6}$$

对于上文中的 G,我们可以计算得到 $CG = \langle 3, 3, 4, 7, 8, 8 \rangle$。

但是,上面 CG 的计算方法显然没有体现 NDCG 的改进原则中的第二点——排得越靠后的相关文档的价值越低。具有相同的增益的不同文档,排在越后面的文档用于累加的增益应该越少。一个简单的方法就是把文档的增益除以其位置的对数函数后再进行累加,从而得到位置 j 的文档折扣累加增益(Discounted Cumulative Gain, DCG)。DCG 值的定义如下:

$$\mathrm{DCG}[j] = \begin{cases} \mathrm{G}[1] & \text{if } i = 1, \\ \mathrm{DCG}[j-1] + \mathrm{G}[i]/log(j+1) & \text{if } i > 1. \end{cases} \tag{4-7}$$

对应的,我们可以求得 $DCG = \langle 3, 3, 3.5, 4.79, 5.18, 5.18 \rangle$

最后,为了便于比较,我们需要将 DCG 值进行归一化处理,使其统一到区间 $[0, 1]$。这只需要将每一个位置的 DCG 值除以最优排序时的 DCG 值即可。V 的最优排序结果为 $V' = \langle 2, 2, 1, 1, 0, 0 \rangle$,相应的 $DCG' = \langle 3, 4.89, 5.39, 5.82, 5.82, 5.82 \rangle$。所以文档序列 V 对应的 NDCG 值为 $NDCG = \langle 1.000, 0.613, 0.649, 0.823, 0.890, 0.890 \rangle$。

4.3 实验设置

在这一节中我们将对本文提出的算法3-1的实验设置作进一步的说明。

在我们的算法框架中,我们采用了 4 种方法来挑选查询 q 的相似查询。而对每一种方法,我们简单地挑选 5 个由 q 的相似查询组成的不同集合。于是,我们一共得到了 $K = 4 \times 5 = 20$ 个相似查询集合。采用这 20 个集合我们可以为 q 与相应的训练集生成 20 个额外特征。对于 TD2003 和 TD2004 数据集来说,新的训练集与测试集中的每一个文档将由 64 个文档特征组成。

我们采用两个代表当前全监督排序学习水平的算法 RankBoost 与 SVM^{map} 分别来验证本文提出的半监督学习框架的有效性。仿照一般排序学习算法的实验过程,我们采用这两种算法在 TD2003 与 TD2004 数据上进行五重交叉校验来挑选最好的参数,以获得更接近真实情况的性能。其中,在采用 RankBoost 算法作为最后一步的排序模型 **Learn( )** 时,我们在前面的特征生成步骤中也相应地使用了 RankBoost 作

为训练模型。类似的,在 SVM^{map} 作为最后一步的排序模型时,我们也相应地使用了 SVM^{map} 来生成额外特征。

4.4 实验结果

图4-2显示了在 TD2003 与 TD2004 数据集上采用 RankBoost 算法作为最后一步的排序模型 **Learn( )** 时所得到的 MAP 结果的比较。其中,baseline 表示原始的全监督算法,而 transductive 则表示本文提出的直推式算法(后面的图示中类似)。图4-3则显示了 SVM^{map} 作为排序模型 **Learn( )** 时相应的 MAP 结果的对比。

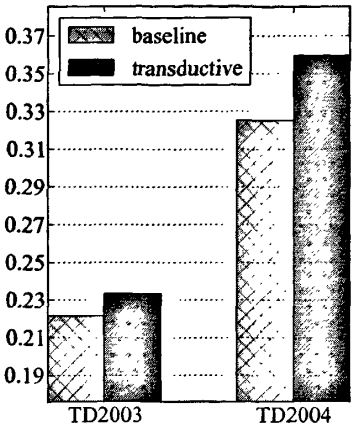

图 4-2 采用 Rankboost 算法时的 MAP 值对比

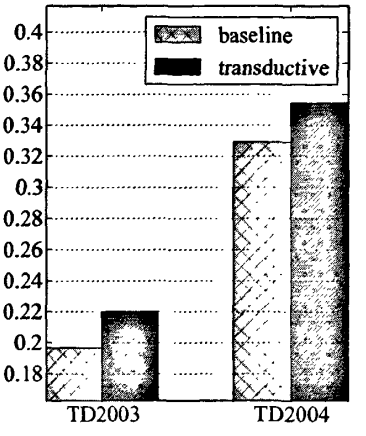

图 4-3 采用 SVM^{map} 算法时的 MAP 值对比

图4-4到图4-7分别显示了采用 RankBoost 算法和 SVM^{map} 算法在 TD2003 与 TD2004 数据集上的几个主要 NDCG 值结果的对比。

从这些关于 MAP 值与 NDCG 值的对比图示中我们可以得出以下两点关于实验结果的简单结论:

- 采用 RankBoost 算法时,除了在 TD2003 数据集上 NDCG@1 的值保持不变外,我们提出的半监督 RankBoost 算法比全监督 RankBoost 算法在其它评价标准上都有不同程度的提高,其中在 TD2004 数据集上的改进效果更加明显;
- 采用 SVM^{map} 算法时,我们提出的半监督算法在所有的评价标准上都取得了提高。而且,与采用 RankBoost 算法时相反,采用 SVM^{map} 算法时在数据集 TD2003 上的改进效果比在 TD2004 上更加明显。

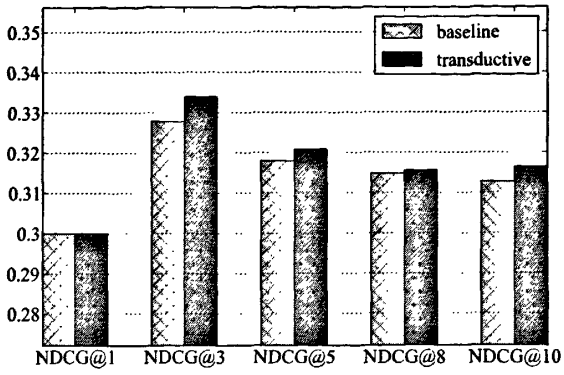

图 4-4 采用 RankBoost 算法时在 TD2003 上的 NDCG 值对比

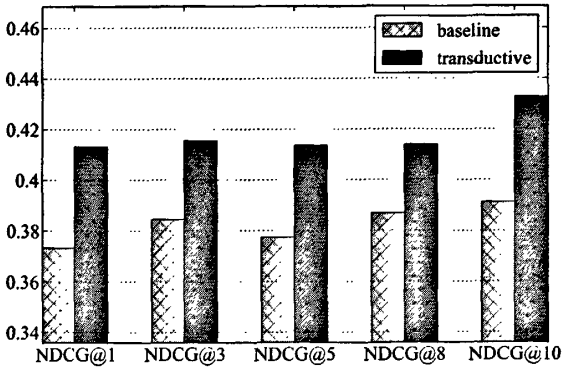

图 4-5 采用 RankBoost 算法时在 TD2004 上的 NDCG 值对比

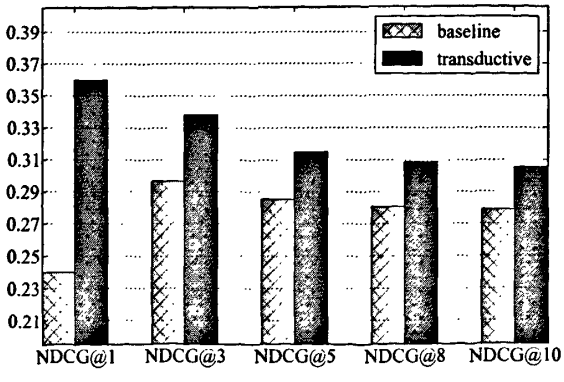

图 4-6 采用 SVM^{map} 算法时在 TD2003 上的 NDCG 值对比

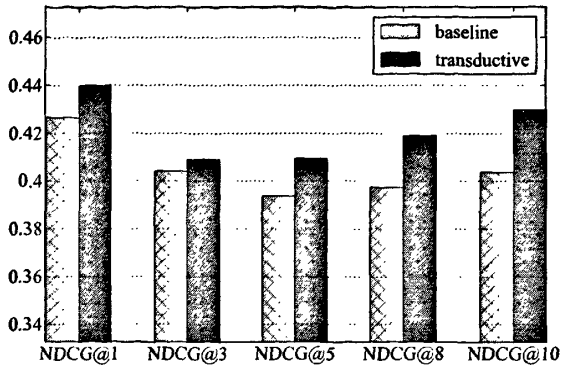

图 4-7 采用 SVM^{map} 算法时在 TD2004 上的 NDCG 值对比

4.5 实验结果分析

在上一节中我们给出了利用查询相似性信息的直推式排序学习算法在 TD2003 以及 TD2004 数据集上的实验结果的柱状图表示。这些实验结果图示向我们展示了本文提出的算法的直观的改进效果。在这一节中,为了更好地验证本文提出的算法的有效性,我们将对实验的数值结果做进一步的分析。

表4-3与表4-4给出了与前面实验结果柱状图相对应的数值结果[2]。其中,括号里面的数字指出了我们的半监督排序算法框架比相应的全监督算法的改进百分比。

表 4-3 半监督与全监督 RankBoost 算法在 TREC 数据集上的实验结果

数据集	算法	MAP	N@1	N@3	N@5	N@8	N@10
TD2003	全监督	0.2216	0.3000	0.3278	0.3180	0.3148	0.3128
	半监督	0.2335	0.3000	0.3340	0.3208	0.3156	0.3164
		(5.35%)	(0.00%)	(1.88%)	(0.89%)	(0.26%)	(1.14%)
TD2004	全监督	0.3252	0.3733	0.3845	0.3774	0.3870	0.3913
	半监督	0.3597	0.4133	0.4155	0.4136	0.4139	0.4327
		(10.60%)	(10.71%)	(8.07%)	(9.58%)	(6.96%)	(10.58%)

[2] 由于版权的原因,我们并不能获得 RankBoost 算法的原始实现代码,所以我们按照 RankBoost[21] 论文中的原理自己实现了这个算法。虽然我们这里给出的全监督 RankBoost 算法的 MAP 值与 NDCG 值与 LETOR 数据集中公布的结果不太一样,但是我们的结果与 LETOR 中的结果不相上下。

表4-4 半监督与全监督 SVM^{map} 算法在 TREC 数据集上的实验结果

数据集	算法	MAP	N@1	N@3	N@5	N@8	N@10
TD2003	全监督	0.1968	0.2400	0.2967	0.2854	0.2807	0.2789
	半监督	0.2200	0.3600	0.3380	0.3150	0.3087	0.3054
		(11.79%)	(50.00%)	(13.92%)	(10.38%)	(9.98%)	(9.49%)
TD2004	全监督	0.3295	0.4267	0.4042	0.3938	0.3974	0.4038
	半监督	0.3546	0.4400	0.4090	0.4094	0.4191	0.4299
		(7.62%)	(3.13%)	(1.19%)	(3.97%)	(5.46%)	(6.47%)

从表4-3中我们可以看到:半监督的 RankBoost 算法在 TREC 的两个数据集上的 MAP 值都有了明显的提高(分别超过了 5% 与 10%)。其中在 TD2004 数据集上面半监督的 RankBoost 算法的 NDCG 评价值比全监督算法有近 10% 左右的提高,而在 TD2003 上的提高率则为 1% 左右。

在表4-4中,半监督的 SVM^{map} 算法在 TD2003 与 TD2004 上的 MAP 值同样有着明显的提高(分别超过了 11% 与 7%)。其中在 TD2003 数据集上面半监督的 SVM^{map} 算法的 NDCG 评价值比原始算法有了超过 10% 的提高(其中在 NDCG@1 上面尤为显著,改进率达到了 50%),而在 TD2004 数据集上面则是达到了 4% 左右。

从这两个表中我们可以看到,本文提出的利用查询相似性信息构造的半监督排序框架在排序学习领域的基准数据集 LETOR 上取得了很好的效果。其次,原始的 RankBoost 与 SVM^{map} 算法在 TD2003 与 TD2004 数据集上面的不同表现也从另一个侧面反映了数据集的多样性,很好地体现了本文前面关于使用同一个模型很难对不同的数据集取得同样好的排序结果的假设。而本文提出的半监督算法由于更好地利用了查询之间的相似性信息生成那些更有助于训练过程的额外特征,使得这两个不同的排序模型在同一个数据集上的结果更加一致。

前面我们给出的结果都是基于数据集里面所有查询的平均值的,下面我们将从每一个查询的改进情况入手,具体探讨本文的半监督学习框架对数据集里面不同查询的影响。在这里我们以采用 SVM^{map} 算法时在 TREC 数据集上的 MAP 值改进情况为例做一些简单的讨论。

图4-8与图4-9分别给出了在 TD2003 以及 TD2004 数据集上那些 MAP 值改进与降低的查询数目柱状图。在每一个图中,右边的条形从上到下分别表示比原全监督

算法提高了超过1%,超过20%与超过50%的查询的个数,而左边的条形从下到上则分别表示比原来降低了超过1%,超过20%以及50%的查询个数。

在TD2003数据集的50个查询中,半监督的SVM^{map}算法有29个查询的MAP值比原来有了超过1%的提高,并且有15个查询的提高率超过了50%。相反,降低超过1%的查询数目只有17个,降低超过50%的查询有8个。

在TD2004数据集的75个查询中,半监督的SVM^{map}算法有37个查询的MAP值比原来有了超过1%的提高,并且有16个查询的提高率超过了50%。而降低超过1%的查询数目有25个,其中降低超过50%的查询只有6个。

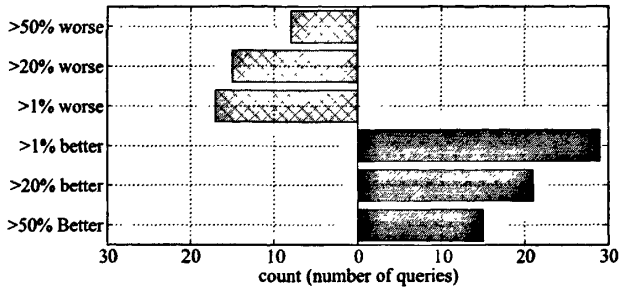

图4-8　采用SVM^{map}时在TD2003数据集上所有查询的MAP改进性能统计

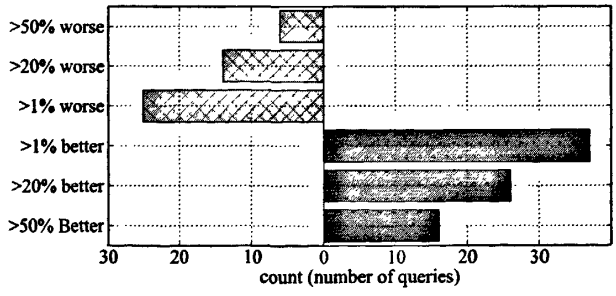

图4-9　采用SVM^{map}时在TD2004数据集上所有查询的MAP改进性能统计

通过这些统计数据我们可以观察到,我们提出的半监督排序学习算法并不是在每一个查询的评价标准上都有提高。但是总体上而言,比原来全监督算法的评价值提高的查询的数目远大于降低的查询的数目。

为进一步获得采用SVM^{map}算法时本文提出的半监督排序学习模型在各查询上的MAP值改进效率的直观理解,图4-10与图4-11分别给出了TD2003与TD2004数据集上所有查询在两个不同算法下的散点图。其中"o"代表MAP值取得改进的查询,"×"代表MAP值降低的查询,而"□"则表示那些MAP值没有改变的查询。

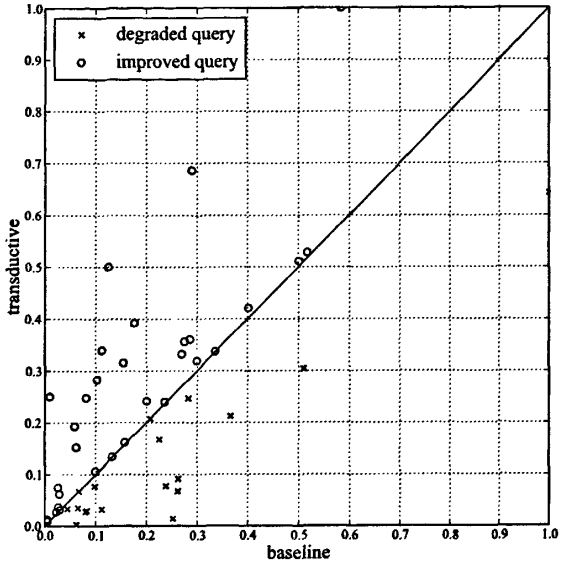

图 4-10 采用 SVM^{map} 时 TD2003 数据集上各查询的 MAP 散点图

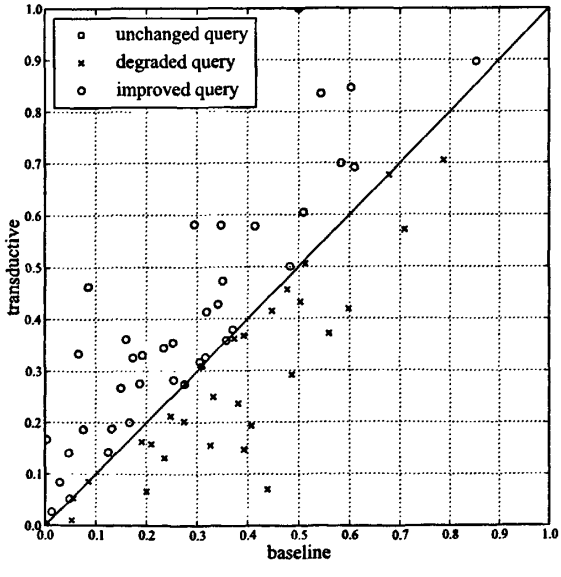

图 4-11 采用 SVM^{map} 时 TD2004 数据集上各查询的 MAP 散点图

从TD2003数据集上的查询分布可以看出,当查询在全监督 SVM^{map} 算法下的MAP值在区间 $[0, 0.2]$ 时,本文提出的半监督 SVM^{map} 算法的改进效果最好(无论是在改进的查询数目上还是在各个查询的改进百分比上),在TD2004数据集上这种现象更为明显。而在所有其它的MAP值区间中,无论是TD2003数据集还是TD2004数据集,半监督排序学习算法的改进程度就没有那么显著了。

一种可能的解释就是原始数据集中的那些很难排序的查询所包含的文档特征不具备较好的区分性,这导致了它们在原有的全监督算法下很难被排序正确。而我们的半监督排序学习算法利用查询之间的相似性信息为这些查询生成了更适合于排序该查询的额外文档特征,这些特征很显然在后面的排序模型被加以利用并取得了较好的排序改进效果。

第5章 总结与展望

在这一章中我们将对本文的研究工作进行总结,并针对研究过程中出现的一些问题以及后续工作作进一步展望。

5.1 论文工作总结

本文的研究主题是基于查询相似性的直推式排序学习算法。当前,研究学者们已经提出了很多全监督的排序学习模型,但是在半监督方面的工作却还是比较少。本文在前人的工作之上提出了一个基于查询相似性的半监督排序学习框架,并取得了相对较好的结果。

在本文中,我们首先通过对半监督学习算法与查询相似性的介绍探讨了这两种策略应用于排序学习算法之中提高排序准确率的可能性,接着详细地介绍了我们的主要研究工作,即基于查询相似性的直推式排序学习算法框架。

算法的主要原理是通过测试集上的查询与训练集之中的查询的相似性信息来为原始文档数据生成额外的特征,并为测试集中的每一个查询学习一个独立的排序模型。我们认为这种依赖于查询的学习方式将会大大提高排序的准确率,因为我们生成的额外特征能够更好地表达出测试集中的每一个查询。

在算法的实现细节上,我们提出了一种基于文档特征值标准差的查询特征向量定义方法,并给出了一种与 Kendall's τ 距离类似的用于度量两个查询之间相似性的新颖方法。通过这些方法的组合,我们在训练集上为测试集中的每一个查询挑选其相似的查询来生成一些额外的文档特征,从而得到一个更加适应于预测该查询的数据集。最后,我们在新的训练集上使用全监督排序学习算法来学习查询的预测模型。

同时,由于本文提出的半监督排序学习算法将文档特征映射到一个更加高维的向量空间,并且为每一个查询均训练一个独立的排序模型,这样无疑降低了算法的时间效率。于是我们提出了一种离线优化的半监督排序学习框架,通过预先计算好排序模型来提高对查询的预测效率。

在 LETOR 数据集上的实验结果表明,无论是当前比较好的 Pairwise 型排序学习算法算法还是 Listwise 型排序学习算法,我们的算法框架都能使其比原始的全监督排序策略有着明显的提高。

5.2 论文工作展望

从实验的设置与实验结果的分析中,我们可以得到一些有关本文提出的基于查询相似性的半监督排序学习框架的后续研究思路。

首先,算法框架的核心之处在于额外特征的生成。如何表达一个查询的特征向量以及度量两个不同查询之间的相似性,直接决定了生成的这些额外特征的质量。未来我们可以进一步探讨一些更适用于排序学习算法的定义方式。

其次,我们在文中给出了一种离线优化的半监督排序学习框架用于改进算法的性能,这是一种采用缓存策略(Cache Strategy)的优化方法。接下来我们可以考虑其它的优化策略。例如引入近似算法(Approximation algorithm)来提高算法的性能,或者考虑将特征生成过程与最终的排序训练过程这两个步骤结合起来统一优化。

最后,从 RankBoost 与 SVM^{map} 的实验结果中我们可以看到,本文的算法框架在同一个数据集上采用不同的算法的改进效果是不一样的。以后我们可以进一步研究这些算法与那些额外特征之间的联系,根据不同的算法自动选择最佳的额外特征生成方法,使得这个半监督排序学习框架采用同一个算法在不同的数据集上面能取得同样好的改进效果。

参考文献

[1] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[2] 刘铁岩、徐君、李航、马维英. 为搜索引擎学习最优的排序模型. 中国计算机学会通讯 *(Communications of CCF)*, 3(10):41–45, 2007.

[3] N. Ailon and M. Mohri. An efficient reduction of ranking to classification. In *To appear: The 21st Annual Conference on Learning Theory (COLT), Helsinki, Finland,* 2008.

[4] N. Ailon. A Simple Linear Ranking Algorithm Using Query Dependent Intercept Variables. *Advances in Information Retrieval*, pages 685–690, 2009.

[5] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web,* pages 1–10. ACM, 2009.

[6] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research,* 9:203–233, 2008.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, page 96. ACM, 2005.

[8] Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, page 193. ACM, 2006.

[9] M.R. Amini, T.V. Truong, and C. Goutte. A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *Proceedings of the 31st annual*

international ACM SIGIR conference on Research and development in information retrieval, pages 99–106. ACM, 2008.

[10] K. Duh and K. Kirchhoff. Learning to rank with partially-labeled data. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 251–258. ACM, 2008.

[11] S. Agarwal. Ranking on graph data. In *Proceedings of the 23rd international conference on Machine learning*, page 32. ACM, 2006.

[12] I.H. Kang and G.C. Kim. Query type classification for web document retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, page 71. ACM, 2003.

[13] X. Geng, T.Y. Liu, T. Qin, A. Arnold, H. Li, and H.Y. Shum. Query dependent ranking using k-nearest neighbor. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122. ACM, 2008.

[14] T.Y. Liu. *Learning to rank for information retrieval*. Now Pub, 2009.

[15] N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems (TOIS)*, 7(3):183–204, 1989.

[16] D. Cossock and T. Zhang. Subset ranking using regression. *Learning Theory*, pages 605–619, 2006.

[17] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, 2004.

[18] P. Li, C. Burges, Q. Wu, JC Platt, D. Koller, Y. Singer, and S. Roweis. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in Neural Information Processing Systems*, 2007.

[19] K. Crammer and Y. Singer. Pranking with ranking. *Advances in neural information processing systems*, 1:641–648, 2002.

[20] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. *Advances in neural information processing systems*, pages 961–968, 2003.

[21] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4:933–969, 2003.

[22] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, page 142. ACM, 2002.

[23] M.F. Tsai, T.Y. Liu, T. Qin, H.H. Chen, and W.Y. Ma. FRank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, page 390. ACM, 2007.

[24] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[25] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, page 278. ACM, 2007.

[26] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, page 398. ACM, 2007.

[27] T. Qin, X.D. Zhang, M.F. Tsai, D.S. Wang, T.Y. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing & Management*, 44(2): 838–855, 2008.

[28] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, page 136. ACM, 2007.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[30] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[31] X. Zhu and A.B. Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool Publishers, 2009.

[32] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th international conference on World Wide Web*, page 400. ACM, 2005.

[33] Y. Li, Z. Zheng, and H.K. Dai. KDD CUP-2005 report: Facing a great challenge. *ACM SIGKDD Explorations Newsletter*, 7(2):99, 2005.

[34] D. Shen, J.T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, page 138. ACM, 2006.

[35] R. Song, J.R. Wen, S. Shi, G. Xin, T.Y. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, and W.Y. Ma. Microsoft research asia at web track and terabyte track of trec 2004. In *Proceedings of the Thirteenth Text REtrieval Conference Proceedings (TREC-2004)*, 2004.

[36] M.G. Kendall and J.D. Gibbons. *Rank correlation methods*. Oxford University Press, 1990.

[37] S.J. Axler. *Linear algebra done right*. Springer Verlag, 1997.

[38] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[39] T. Joachims, T. Finley, and C.N.J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[40] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 88–96. ACM, 2008.

[41] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007.

[42] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):446, 2002.

致 谢

首先诚挚的感谢我的导师汤庸教授。在协同软件实验室的两年里,汤老师为我提供了一个宽松自由的学术环境,让我得以顺利完成论文的工作。汤老师对工作兢兢业业,对学生和蔼宽厚的态度给我留下了深刻的印象。

衷心感谢实验室信息检索小组的潘炎老师。在加入小组之后,潘老师在如何做好研究上给了我很多的教育与指导,使我得以一窥学术殿堂的深奥,受益匪浅;潘老师对待学术严谨的态度以及在技术上渊博的知识也深深地感染和激励着我。很感谢潘老师在学习和生活上对我的诸多关心与建议。

感谢实验室信息检索小组的成员们:陈国华师兄、彭泽武师兄、罗烨敏师兄、林鹭贤师兄、吴桂宾师兄、戚洪睿同学以及罗海霞师妹。每一次与你们共同讨论学术上的问题都让我获益良多,你们在学习和生活上也给我提供了许多的关怀与帮助。

感谢所有在实验过程中给我提供了帮助的那些素为谋面的研究人员们。谢谢华盛顿大学的 Kevin Duh、康乃尔大学的 Yisong Yue,微软亚洲研究院的 Tao Qin 等人,你们的回复给我提供了宝贵的意见,同时也解决了我在实验过程中遇到的一些疑问,使我得以顺利地完成论文的相关实验。

感谢在中山大学学习的六年里认识的那些同学和朋友们,你们的陪伴让我的大学生活变得更加丰富多彩,这将是我人生中最值得回忆的时光。

最后,谨以此文献给我挚爱的双亲。他们在背后默默的支持与鼓励陪伴着我一路走过人生的各种酸甜苦辣,也给了我不断进取的无穷动力。

蔡奕伦

二零一零年五月

于 中东至善园

基于查询相似性的直推式排序学习算法研究

作者： 蔡奕伦
学位授予单位： 中山大学