校园搜索引擎Sixer

王苜子 计51 2015011387
王颖 计51 2015011387

1. 项目目标

综合运用搜索引擎体系结构和核心算法方面的知识，基于开源资源搭建搜索引擎，用于http://news.tsinghua.edu.cn的网页内容搜索。

- 使用Heritrix抓取“清华大学新闻网”站点的所有内容
- 实现基于概率模型的内容排序算法；
- 实现基于HTML结构的分域权重计算,并应用到搜索结果排序中；
- 实现基于PageRank的链接结构分析功能,并应用到搜索结果排序中；
- 采用便于用户信息交互的Web界面；

2. 数据抓取

尝试使用heritrix1.14和heritrix 3.2两个版本抓取数据。对于1.14，可以完全按照实验要求给出的参考设置选项。可能会碰到种子抓取失败的问题，解决方案是缩短过滤规则表达式，然后等排队网页增多的时候使用完整的过滤规则。可以设置抓取间隔的最小值和最大值，在速度和抓取量之间做一个平衡。设置的过滤规则首先是拒绝图书馆资源相关url，然后拒绝无用文件，最后接受清华新闻域名下的网页。

3. 内容索引

Lucene是一个优秀成熟的、简单易用，用于全文检索和索引的Java开源工具包，在最近的几年广受欢迎。本次实验尝试使用了4.0，这是能够很好支持IKAnalyzer V2012分词工具的最高版本。在图片搜索实验的基础上进行了修改。首先对图片搜索实验中自己的实现与内置BM25标准模型进行对比，两者表现相同。同时尝试了VSM模型，但是在后续的实验中发现效果不好，这里不再赘述。

文档评分机制是搜索引擎中极为重要的一部分，决定了各个文档在页面中的出现顺序。本实验中采用了组合html分域权重和Page Rank得分的BM25概率模型。

首先离线计算Page Rank得分。简单起见本实验用java重新实现了实验3中的PageRank算法。

PageRank得分作为一个网页的权重$boostDocument$。

其中一个问题是非HTML文档的得分如何计算。首先其中的出度极其少(主要是一些参考文件，往往不是清华新闻相关的网页)，根据入度计算效果还不错。之后的调整中有给它加上了一个偏移因子，提高了文档的得分，使得最终的效果中能够搜索到很相关的文档。

计算之后的PageRank很正常，排名靠前的是中英文新闻主页以及各个子主页(比如图说清华，头条新闻，媒体清华)。最初没有调整，因为我认为需要一部分好的新闻网页包含多条新闻内容的索引。但是之后的效果不

够好，仔细分析发现主页作为每个新闻网页的头部，都会存在一个链接，不能凭此认定它的内容最好。

另外索引的时候对分数进行了调整，主要是因为一些网页之间得分相差两个数量级，导致评分的时候其余的因素影响都比不过PageRank得分的影响，这是不正常的。比如搜索生僻新闻的时候，内容相关的网页可能因为PageRank得分太低而被排在后面。然后对分数进行了取对数处理，缩小得分差距，但是保持大小关系不变。

然后解析网页内容，根据HTML结构下的不同标签作为不同的域，比如标题，关键字等。每个域有一个域的权重$boostField$。

分析了网页的html特点，对以下内容进行了索引：

- title：对应新闻标题内容。该信息是作为区分文档最重要信息。
- h：大多数情况下对应新闻的标题或者小标题，该信息的重要性仅次于title。
- p：对应新闻正文内容，是该页面较重要的信息。
- a：可以利用锚文本信息。链向一个网页P的超链接中会包含描述性文本a，a就可以作为网页P的一个补充文本。
- keyword：出现不多且区分度不够好，没有利用
- 加强标签，如b，strong不太常见，没有使用。

对于非HTML文档，其中没有这种结构，于是简单处理文档名字等价于标题，文档内容等价于正文。

域权重是超参数，需要通过实验进行调整。类似实验1，构造查询样例，构造pooling进行结果分析和比较，最终确定了各个域的参数。

4. 内容检索

主要利用Lucene框架，在图片搜索的基础上进行了改造。前端做了很大的修改，并且规范了前端文件的存放。由于实现了扩展以及索引文件较大，查询过程较慢，经过了多次内存优化，现在能够在占用内存800M左右的时候实现查询的快速响应(显示结果时间小于0.5s)。

5. 扩展

(1) 相关推荐

打算做成类似百度搜索右侧的那部分，为用户提供与其该次搜索相关的关键词，以便用户进行深入搜索找到理想网页，或获取其他意想不到的信息。当前基础下没有办法实现用户搜索行为信息进行推荐。可以利用的信息只有网页内容。自己设计实现了算法，从两个方面计算两个关键词的相关分数，每次推荐相关分数前N高的关键词。

首先是从内容方面，基于$TF*IDF$模型思想实现。TF定义为用户查询词k和待推荐关键词m同时出现在一篇文档中的次数。但是问题就是常用词TF异常高，于是引用IDF思想。DF定义为m出现在多少文档中，IDF是其倒数。剩下的问题就是生僻词的IDF特别高，造成了相关分数也很高。于是重新定义 $$ IDF = \frac{1}{\sqrt {\alpha DF}} $$ α是一个超参数，用来控制平衡常用词和生僻词的干扰问题，

最终确定$\alpha=2$。相关分数为 $$ r(k,m)=\frac{TF(k,m)}{\sqrt {\alpha* DF}} $$ 实现过程非常非常痛苦，需要有根据文档查找关键词的数据结构和根据关键词查找文档列表的数据结构。在线计算分数不现实，于是决定离线保存每一关键词最相关的N个词项。但是同样需要解决数据结构复杂带来的内存问题，这个实现花了几乎4天时间不停的优化，最后能够在电脑不卡死(我电脑4G内存)情况下尽快计算完毕。

(2) 自动补全

文本补全的原理是寻找到与当前查询词有相同前缀的查询词汇。利用了Lucene中suggest模块的部件FSTCompletion。建立好索引之后，将content域中的每个token（即文本经过分词后得到的词汇）取出，并将每个查询词的重要性置为其在各个文档中的出现频度。然后交由FSTCompletion模块进行处理。

该模块的原理是对输入的词汇构造有限状态自动机。在进行查询时，在查询词的子自动机中寻找终态，每找到一个终态就把当前路径对应的词汇添加进备选词汇列表中。在获得足够的备选词汇后，按照其重要性及其字母序进行排序，然后按顺序高低将指定数目词汇返回作为补全结果。

前端使用bootstrap插件实现显示效果，配合Ajax技术，在用户输入查询词的时候能够实时的将数据通过Ajax传送给后台，后台查询补全结果返回。前端进行显示部分更新。

从效果图中可以看到，无论中文、英文，一个字符、多个字符，我们的搜索引擎都能为用户提供，以用户键入的序列为前缀的，频度较高的词。这种设计使得用户在输入的时候不必输全所有的内容，就能获得搜索的关键词，简化了用户的搜索步骤，提高了搜索的效率。

6. 开源工具

Lucene 4.0

IKAnalyzer 2012

Jsoup

pdfbox 1.8.13

poi 3.16

7. 实验总结

本次实验让我们熟悉了多种方便开发且功能强大的开源软件，例如用于数据抓取的工具Heritrix，搜索引擎整体框架的搭建Lucene，HTML解析工具Jsoup，PDF解析工具pdfbox，M.S Office解析工具poi等等。当然在使用这些工具中我们也遇到了一些配置上的小问题，例如heritrix参数的选择等。

巩固了BM25概率模型，PageRank算法等相关知识，并学会将这些理论相结合应用于实际的工程中，并在实际工程的开发中，结合抓取数据的独特性，对原始算法内容进行改进以更好地适应数据的情况。除此之外，我们在完成了基础功能之后，对数据进行了全面的分析，结合之前课程所学知识，独立设计了包括相关词推荐、文本补全一系列扩展功能。

在算法的设计、实现和应用中我们遇到了许多小问题，在不断地讨论，查找相应资料中一步步地进行了解决，通过这一次的实践，也让我们再一次深刻地认识到理论指导实践的同时，从理论到实践依然是一个富有挑战性的过程，这一过程也是整个过程中最有成就感的部分。

感谢老师和助教在这次大作业中提供的建议和指导！

参考资料

课件及实验要求
各个开源工具的官方文档 https://www.ibm.com/developerworks/cn/opensource/os-cn-heritrix/
http://blog.csdn.net/wy_kath/article/details/9385015
http://blog.wuzx.me/archives/368
http://www.cnblogs.com/shiyu404/p/6344591.html
http://blog.csdn.net/java85140031/article/details/18969467