第11卷 第34期 2011年12月
1671—1815(2011)34-8619-05

科 学 技 术 与 工 程
Science Technology and Engineering

Vol. 11 No. 34 Dec. 2011
ⓒ 2011 Sci. Tech. Engrg.

基于补偿的 Nutch 搜索引擎的设计与实现

马 睿 黄 穗
(暨南大学,广州 510632)

摘 要 Nutch 的排序机制使得一些传统的高质量的页面经常出现 Web 检索结果的前面,而新加入的高质量的 Web 页面很难被用户找到。针对传统基于链接算法中对新内容的迟钝性,提出了对网页的补偿算法,对 Web 上在一段时间内好的资源信息使其排名结果上升,而对于不好的资源使其下降,以此来加速优质内容的传播和网络内容的合理化。并最终把改进后的算法应用在基于 Nutch 的搜索引擎中,通过实验来验证和测试。

关键词 Nutch 排序策略 补偿机制
中图法分类号 TP391.3; **文献标志码** A

页面排序技术是搜索引擎的一项关键技术,因为检索的结果直接面向用户,影响用户的体验感受。Nutch 是一个开源的搜索引擎项目,为研究页面排序技术提供了很多便利[1]。

Nutch 的检索排序,属于典型的互联网搜索引擎体系。它主要有两部分构成,一种是基于全文检索的关键词匹配算法,这部分主要依赖于 Lucene 的索引检索机制,通过其自定义的相识度函数来实现。另一种是网页的重要性权重,通过网页之间的链接结构计算得到[2]。Nutch 的基于网页重要性算法是 optic,其基本原理和 PageRank 类似。在 Nutch 中把这个网页链接重要性量化为 Boost 值,最后通过 Lucene 的 setBoost 方法把这个值传递给最终的综合排序算法。虽然这种方法提高了搜索引擎的检索质量,但它妨碍了新加入 Web 上的高质量网页的内容传播。因为新的页面由于其链接数目比较少,很难出现在搜索结果的前面,从而很难被检索用户发现。

为此本文提出一种基于网页重要度的补偿排序策略,对在一段时间内重要的网页给予一定的奖励,使其排序结果与其在一定时间内的重要度挂钩。根据一段时间内真实的排名结果,来建立一个惩罚与奖励的制度,这样有利于信息的快速传播,也有利于信息的优胜劣汰。

2011年9月15日收到,9月26日修改

第一作者简介:马 睿,男,暨南大学信息科学与技术学院硕士研究生。研究方向:软件工程与理论。E-mail:ma01280802@163.com。

1 Nutch 排序机制

Nutch 的排序是基于索引和检索部分的核心是采用 Lucene, Lucene 的评分公式代码是在 org.apache.lucene.search.Similarity 类里实现的[3]。

$$S(q,d) = c(q,d)q(q)\sum_{t}(t(d)i(t)^2BN(t,d)) \quad (1)$$

式(1)中 $c(q,d)$ 表示的是这个文档 d 在查询词的项中出现的次数;文档中出现的项的次数越多,给予的分值也就越高。这个函数一般在查询的过程中计算的。$q(q)$ 是一个归一化函数,它使得不同查询之间具有可比性,它并不影响最终具体的排序结果。这个条件是在搜索的时候计算的。$t(d)$ 表示的是在查询条件中,每个项 t 在当前文档中出现的频率,频率越高其得分也就越高。$i(t)$ 表示的是反转文档的频率。它是指项 t 在所有文档中一共在多少个文档中出现过。出现的越少说明 t 对这个文档的贡献越大,也越容易定位,得分也就越高。B 是查询的时候对项 t 设置的权重值。$N(t,d)$ 得到的是在建立索引时得到的一些计算出的参数值,它封装

了一些列优先权和长度的数值。其计算公式如公式(2)。

$$N(t,d) = B_t(\,)l(f)\prod_f B_f(\,) \tag{2}$$

公式(2)中的 B_t 为文档的 boost 值,是建立索引的时候设置的全局文档的得分,文档域的 boost 是一个文档域添加到文档中时设定的字段得分。在公式(2)中可以看到,文档有全局属性 boost,这个 boost 值和 PageRank 算法中的 PageRank 算法中的 PageRank 数值具有相似性。在 Nutch 的代码中有一个基于超链接分析的工具类,它在 org. apache. nutch. tools. DistributedAnalysisTool 类中有具体的实现,这个类实现了一个类似的 PageRank 算法。Nutch 最终排序公式中的 PageRank 因子正是通过这个字段的值添加到 Lucene 检索中的。由于它的底层采用 Lucene 作为索引和检索的内核,Nutch 在把外部的链接情况分析完成后,把链接量化为一个 boost 值,利用 Lucene 的 setBoost 方法来实现最后的文档权重问题。

2 补偿机制

2.1 Nutch 算法的不足

由第 1 节的介绍可以看到,Nutch 中的网页重要性权重计算其原理和 PageRank 是一样的。下面是 PageRank 的公式[4]:

$$P(u) = d + (1-d)\sum_{v\in V(u)}\frac{p(v)}{N(v)} \tag{3}$$

公式(3)可以理解为一个网页的权重值来自两部分:一部分是来自跳转因子 d;另一部分来自父网页传递的权重值。PageRank 的本质体现了浏览 Web 的"随机访问模型",即用户随机从某个页面出发,下个页面被点击的概率即该页面的 PageRank 值。

Web 上有这样一个事实:很多不为人知的网页可能包含有极高的价值,而由于 PageRank 的计算方法,没有链接父网页的传递或链接到的网页很少,自然其 PageRank 值也就相对较低,从而在检索结果中很难被用户发现;另外一个网页最近更新了内容,其中很可能有比较重要的信息,但根据 PageRank 算法思想,只对链接结构进行迭代计算,无法考虑其重要性[5]。使得网页 D 即使更新了内容也很难在短时间内提高其权重值,从而让用户发现。补偿机制正是帮助这些"弱者"提升自己的重要性,用主动的方式加速有价值的信息传播。

2.2 补偿排序

基于上述因素,有必要对上述资源在计算其权重值的时候给予一定的补偿。算法的思想很简单,引入网页的修改时间作为补偿因子,一个网页的修改,有理由认为其进行了内容上的更新或者网页刚发布也是新的内容。一个网页的修改时间离当前时间越近,得到的补偿也就越多[6]。

在算法的设计上引入补偿因子,设网页的修改时间用 T_m 来表示,当前时间用 T_n 来表示,可以得到加入补偿因子的计算方法为公式(4)所示。

$$P(u) = d + (1-d)\left(\sum_{v\in V(u)}\frac{p(v)}{N(v)} + \frac{1}{T_m - T_n}\right) \tag{4}$$

但上面的公式还有一个问题,给所有的新内容都人为地加入了补偿,然而如果这个内容是一些无用信息或者垃圾信息,那不是做了无用功了吗? 为了解决这个问题,引入了网页的补偿机制。即:对加入补偿因子的网页如果其内容用户认为是好的,那么给予一定的奖励,再提高其 PageRank 值,使得用户更容易发现它,提高信息传播的速度;反之,用户认为加入补偿后的信息没那么重要,则给这个网页一定的惩罚,降低其 PageRank 值,同时也相当于提高了其他重要信息的排名。

对于一个搜索引擎来说,一般都是利用爬虫遍历并下载部分或全部的 Web 网页。为了保证搜索内容的更新,对于一个特定的爬虫来说,它完成了本次下载后,索引器对本次的下载进行索引并计算其权重值,然后爬虫循环进行下次的 Web 遍历。这样可以得到多个数据集合,其中每个数据集合中的 Url 由于时间的推移使其在互联网上的引用和访问情况也在发生着变化[7]。补偿机制的核心思想就是通过分析基于时间序列的评价值变化,判断在最近一次更新后,某个 URL 在过去那段时间内的重要度变化情况。如果在过去那段时间内其重要度上升了,则表示很多用户都在关注这个信息,也代表

这个信息是重要的。如果在过去那段时间内这个 URL 的重要度下降了,则说明这段时间内大家更多的是关注其他信息,也许这个信息是过时的信息了。把这个补偿机制作为一个参数,在用户检索时,搜索引擎将按照补偿后的 PageRank 值来决定一个 URL 在检索结果中的位置。

例如:V_1 为爬虫开始第一次下载的数据集合,其中包含了 N_1 个页面,通过某一个特定的搜索引擎排序算法(比如本文介绍的原始 *Nutch* 中的排序算法),对这 N_1 个页面进行排序。得到的排序结果页面集为:$D_1 = \{v_1, v_2, \ldots v_i, \ldots v_{N1}\}$,假设其中页面 k 的排序结果为 v_i,即网页 j 的排序结果为第 i 名。

对第二次下载的数据集合,其中包含了 N_2 个页面,同样通过对网页集 N_2 的权重值进行排序。这样得到排序后的结果页面集合为:$D_2 = \{v_1, v_2, \ldots v_j, \ldots v_{N2}\}$,其中页面 k 的排序结果为 v_j,即网页 j 的排序结果为第 j 名。

设 $C = i - j$,则 C 就代表了在这段时间内网页 k 的实际重要度变化情况,如果其为正数,则说明在这段时间内其重要度上升了;如果为负数则说明下降了,为 0 则是保持不变。

根据这个思想,引入补偿机制,把公式(4)变为如下形式:

$$P(u) = d + (1 - d)\left(\sum_{v \in V(u)} \frac{p(v)}{N(v)} + \frac{C}{T_m - T_n}\right) \tag{5}$$

在具体实现上,如果 $i - j$ 为 0 设置 $C = 1$,表示不进行补偿;如果页面 K 在第一个集合中没有出现,则表示是新网页,也设置 $C = 1$,也不对其进行补偿。

公式(5)看似合理,但仔细分析就会发现,计算过程中来自父网页传递的权重值和给予的补偿因素对这个网页的权重值贡献度是相同的。但在实际情况中来自父网页的传递应该是主要部分,所以要对算法中的来自父网页的传递和补偿因素的贡献进行区分。参考文献[5]的方法对补偿因素进行限制,引入最近一次下载的 URL 集总数 M,对公式(5)进行进一步的改进得到公式(6)。

$$P(u) = d + (1 - d)\left(\sum_{v \in V(u)} \frac{p(v)}{N(v)} + \frac{C}{(T_m - T_n)M}\right) \tag{6}$$

需要注意的是,用补偿机制算法计算出的 PageRank 值可以集成在搜索引擎的检索算法内,可以不针对于某种具体的排序算法。但该修正值在下一次更新并重新计算时,将不作为评价参数而直接使用补偿前的原始数据。这样做是为了使得对网页的惩罚和奖励在相对公平的情况下进行,原始数据在一定程度上真实地反映了链接情况。也就是说反映了用户对其的使用和评价情况,即对于好的信息用户会去主动地进行链接和传播,对于不好的信息会被人们慢慢地遗弃。

2.3　补偿机制加入 Nutch

由于在建立索引的时候 Nutch 就会对 boost 进行初始化并利用类似于 PageRank 的算法计算其值,最后调用 Lucene 的 setBoost( )方法对其进行修改。因此在其建立索引的时候加入算法因子来改变 Nutch 的算法。其中主要涉及三个类文件:Crawl、Indexer、IndexerMapReduce,另外还增加了一个配置文件 store 以用来保存需要的算法数据。

Crawl 类中解决爬行和建立索引的过程,当爬虫爬完所有网页后,调用 Indexer. index( )方法来建立搜索索引,然后调用 Indexer. index( )方法里的 IndexerMapReduce. initMRJob( )对数据进行初始化,最后在 IndexerMapReduce 里的 redece 方法里通过调用 ScoringFilter 的 indexerScore 来迭代计算 boost 值,最后通过 doc. setScore(boost)来修改最终平分公式中的全局 boost 值。

通过上面的分析,做如下改动来实现补偿算法:IndexerMapReduce 类和 Crawl 类的主要改动:调用 indexerScore 后,把实际信息进行处理并写入配置文件,供下次更新时使用;本次也要根据配置信息来计算其补偿因子并加入到算法中。对于网页的修改时间,在 Nutch 中是把抓取回来网页信息都在 reduce 组装好后以 < url, doc > 形式存储,其中 doc 里有如下几个 field:content(正文)、site(所属主地址)、title(标题)、host(host)、segement(属于哪个 segement)、digest(MD5 码,去重的时候用)、tstamp

(时间戳,即需要的时间数据)、url(当前 url 地址)。

程序部分代码如下:

```
HashMap < String,Object > hm = new HashMap < String, Object > ();
ArrayList < Float > al = new ArrayList < Float > ();
Float boost = 1.0f;
Long m =1;
//对初始 boost 进行迭代
Boost = indexerScore ( key, doc, dbDatum, fetchDatum, parse, inlinks, boost);
//对本次信息进行处理并保存
hm. put( doc. getFieldValue( "url") , boost);
al. add( boost);
//统计网络规模
//配置文件没数据,则新算法不起作用,先收集数据;
//否则计算实际名次差额
if( f. length( ) ! = 0) {
HashMap < String,Intger > temp = new HashMap < String, Intger > ();
read( f);//读配置文件
temp. put( url,Intger. valueOf( MingCi) );
//根据 url 的名次差额计算 boost
if( temp. get( doc. getFieldValue( "url") ! = null)
if( temp. get( doc. getFieldValue( "url") = = 0) )
boost = boost + 1/( Integer. parseInt ( datatime) - Integer. parseInt
( doc. getFieldValue( "tstamp") ) );
else
boost = boost + temp. get ( doc. getField ( "url") )/( m * ( Integer.
parseInt( datatime) - Integer. parseInt ( doc. getFieldValue ( "tstamp") ) ) )
else
boost = boost + 1/( Integer. parseInt( datatime) - Integer. parseInt( doc. getFieldValue( "tstamp") ) );
}
doc. setScore( boost);
……
```

注意:第一次系统爬行并建立索引时,配置文件需要搜集信息,以用来对网页进行惩罚和奖励。所以在第二次系统更新时,新算法才开始起作用。

3 实验结果与分析

3.1 抓取信息

为了方便实验,本文抓取了 4 个权威的农业网站信息,在配置爬虫的时加入权威的农业网站种子集,建立文件 url. txt,在这个文件中输入要爬行的入口网页地址,爬虫在开始的时候按照这些网页开始抓取,然后根据这些网页的链接关系向下爬取,直到达到了设定的爬行层数。本文采用了 4 个权威的农业网站作为入口,分别是:

表 1 4 个权威的农业网站作为入口

网络地址	网站名称
http://www. ny3721. com/	中国农业网址大全
http://www. moa. gov. cn/	中华人民共和国农业部
http://www. nongye. cn/	农业中国
http://www. shennong. com/	神农网

3.2 实验结果分析

本实验于 2011 年 7 月、8 和 9 月分别进行了两次层数为 3 的下载实验。

表 2 boost 排名变化

网站名称	7 月	8 月	9 月
http://www. ny3721. com/url/6043/	2 082	-212	-174
http://www. tjbshg. com/	1 733	-55	-82
http://www. china6419. com/	707	-17	1
http://www. cjj. moa. gov. cn/jsfg/	1 025	24	15
http://www. zzys. moa. gov. cn/fagui/	1 703	14	-76

在表 2 中是随机选取的 5 个网站,7 月的数据是其真实的 boost 排名;8 月的数据是 7 月和 8 月的真实 boost 排名之差;同理:9 月数据是 8 月和 9 月真实 boost 排名之差。并且用这个差额作为公式(7)中的补偿因子 C 来影响最终的排名结果。可以看到在 8 月一些网站的 boost 排名有些是上升的,说明过去那段时间其重要度上升了,很多用户都在关注这个信息,也代表这个信息是重要的;也有一些排名下降了。说明过去那段时间内这个 URL 的重要度下降了,这段时间内大家更多的是关注其它信息,这个信息也许是过时的信息了。最后显示经过补偿的排名。

在 9 月的数据中有些真实 boost 排名继续上升或下降,也说明算法能够起到信息的加速传播或下降。还有一些信息没有按照预期的上升或下降,这

个也不奇怪,每条信息都有其内在的价值[8],加速其排序结果的上升和下降,经过长期的维护和更新,使其稳定在其应有的位置。

4　结束语

补偿机制算法计算出的值可以集成在搜索引擎的检索算法内,可以不针对于某种具体的排序算法。在搜索引擎更新时主动的给这段时间内用户关注高的网页一定的奖励,从而达到信息的优胜劣汰,促使有价值的信息可以在 Web 上快速传播。

参　考　文　献

1　李村合,吕克强. Nutch 搜索引擎的页面排序修改方法研究. 计算机工程与设计,2009;30(6):1343—1345

2　Hatcher E, Gospodnetic O. Lucene in action. Manning Publication Co,2005

3　良小伟, 王申康. 基于 Lucene 的全文检索系统研究与开发. 计算机工程,2006;32(4):94—96

4　张　毅, 张冬梅. 搜索引擎 PageRank 算法的比较和改进. 科技创新导报,2008;21(1):18—20

5　张　岭, 马范援. 加速评估算法一种提高 Web 结构挖掘质量的新方法. 计算机研究与发展,2004;(1):98—103

6　王春花. 基于 Nutch 的农业搜索引擎检索结果排序策略的研究. [硕士学位论文]. 西北农林科技大学,2010

7　Zhang Dell, Dong Yisheng. An efficient algorithm to rank Web resources. Computer Networks, 2000; 33: 449— 455

8　张军华. 网络信息传播自由度的影响因素研究. 高校图书情报论坛,2007;6(2):48—51

Resign and Implementation on Compensation Algorithm of Search Engine Based on Nutch

MA Rui, HUANG Sui

(Jinan University, Guangzhou 510632, P. R. China)

[**Abstract**]　Nutch sorting mechanism leading to some of the traditional high-quality Web pages are difficultly to found than the newly added high-quality Web pages. Because of the traditional link algorithm's slow nature to the new content. The compensation algorithm was made for Web page. Period of time to enhance the good information, and the poor information to bring it down, in order to accelerate the spread of high-quality content.

[**Key words**]　Nutch　ordering policy　compensation mechanism

(上接第 8604 页)

The Control of Overlapping Solutions in NSGA-II Algorithm

WANG Guang-bo, HAN Qing, ZHONG Xiao-ping

(School of Aeronautics, Northwestern Polytechnic University, Xi'an 710072, P. R. China)

[**Abstract**]　The introduction of elitism strategy of NSGA-II, which can help preventing the loss of good solutions by choosing the best solutions from the merged population that merges the current population and the offspring population to construct the next population, enables easier production of overlapping individuals. The existence of overlapping individuals in the evolution populations means overlapping regions in the searching space, which makes the algorithm much less efficiently in exploiting new feasible region. In consideration of computational complexity and strengthen effectively of solution set. A remove strategy is presented to improve the NSGA-II algorithm. The numerical examples demonstrate that eliminating overlapping solutions make the NSGA-II algorithm more steady and gain a solution set with better distribution.

[**Key words**]　NSGA-II　overlapping individuals　remove　distribution