Blog Quality Analysis[1]

Xiaorui Chen

***Abstract*: This paper presents a combination of several techniques (RSS Feed, Lucene, and MySQL) that constituted a powerful, efficient system to acquire, parse, and optimize data from Blogs, and then based on analyzing TF (Term Frequency) and Links we make a contribution to Similarity Analysis and Influence Analysis by proposing another two novel Algorithms which are Similarity Score and Influence Score. Hence it becomes much easier and more effective to rank the related and authoritative Blogs under the comparison of Scores.**

Keywords: Term Frequency, Vector Distance, Penalization

I. Introduction

The explosive development of the internet and the massive utilization of social media have dramatically increased the popularity of blogs. As it is claimed in [4] that "over 50 million blogs are reported to exist, and around a hundred thousand new blogs are created everyday". According to the same estimation in [5], it reported that "blogging activity is doubling in size every two hundred days or about once every six and a half months". In additional, blog contains a wealth of information about a variety of events due to more bloggers bloging on more diverse topics including their personal lives, product reviews, political opinions, technology trends, tourism experiences, and the entertainment industry [2], that is, a set of keywords will be correlated and aggregated together, which cause blog spam. However when such topics and events recede, the keyword clusters dissolve automatically since such clusters are temporal, transient, and are associated with a specific time period [5].

Consequently, it becomes difficult to find out which blogs are in highly similar, which blogs are the most important or authoritative (can be reflected in several ways: content, title, and links) due to the frequent changing of blogs over time.

Hence it is significant and urgent for us to build a blog analyzing system to overcome these problems. And accordingly the motivation of this paper is to employ effective existing techniques (RSS, Lucene, and MySQL) and present Score Functions (Similarity Score and Influence Score) with computationally efficient in order to develop a specialized Java application for discovering correlated outlinks, mining hot keywords, distinguishing similarity of blogs, and identifying the influential authoritative websites.

II. Related Work

(1) RSS Feed

Due to the massive development of current web techniques, several APIs (Application Programming Interfaces) have been provided for individuals to view, update, and obtain data from a variety of blogs, and there are two of them are the most popular. One is the Google data APIs ("GData" for short) and the other is RSS (Really Simple Syndication) feed [3] technique, a method [3] that uses XML instead of HTML to distribute web content and fetch updated website information faster by using a RSS aggregator (a site or program that gathers and sorts out RSS feeds) which has been currently widely spread and adopted due to its effective and efficient application.

However, due to its applicable limitation, that is, GData is a protocol based on the Atom 1.0 and RSS 2.0 syndication formats [2] that make GData a far narrower applicable field (constrained by both syndication formats). In addition, the Blogger Data API only supports the standard Google Data API query parameters, namely, this sort of APIs are only applicable for Authenticated Google Blogs [2]. What if the blogs without GData APIs? What if the Bloggers have not authenticated Google Accounts? In this case, GData feed becomes helpless.

Moreover, Dare Obasanjo, a famous experienced Program Manager in Microsoft Corp. suggested that "GData less than a Web General Purpose Editing Protocol set too much restriction and becomes sorts of less compatible with Microsoft". He also claims that "XML data might be lost at times when downloading original 'atom:entry' due to the fact that GData is less supportive for updated entries". Consequently the applicable limitation of GData forces us to use RSS feed in this work.

(2) Lucene

Lucene is a free/open source information retrieval library. At the core of Lucene's logical architecture is the idea of a document that contains several fields of text, hence it allows Lucene's API to be independent of file formats [1], that is, text from PDFs, HTML, Microsoft Word documents, as well as many others can all be indexed and searched flexibly so long as their textual information can be extracted [1].

(3) MySQL

MySQL is one of the most popular open source databases in terms of the ease of use, scalability, and performance, we choose MySQL as the storage environment in this work. Lucene provides the same function as well; however, due to the

[1] Supported by Applications and Algorithms Research Group in Comlab, Oxford University; the grand project of the Science and Technology Commission of Shanghai Municipality (No.06DZ11202); Shanghai Leading academic Discipline Project (No. T0602); and the Subject Foundation of Shanghai Maritime University (No.XL0101-1).

Xiaorui Chan (1982-), Master, MSc in Computer Science, Oxford University, United Kingdom, focus on algorithms design & analysis and network security. E-mail: sherrychan82@yahoo.com.

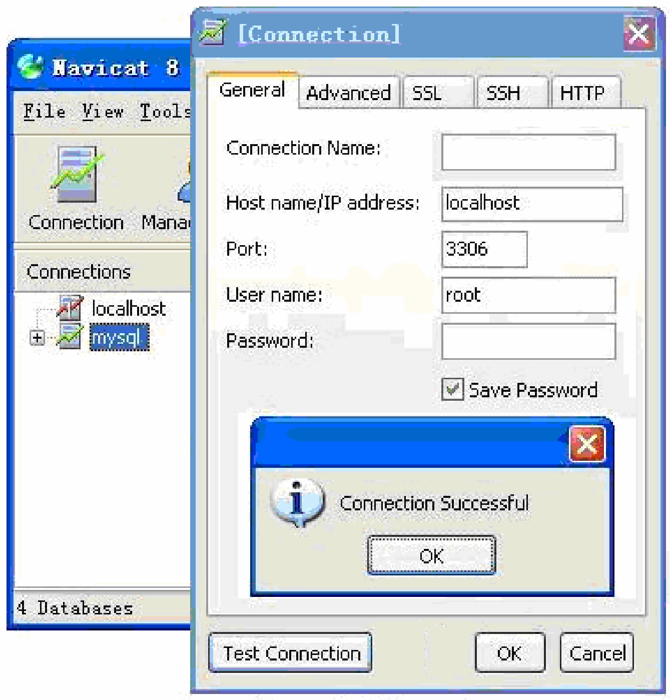
fact that the data storage function performs less professional and shows ineffective compared with MySQL, hence it is only adopted as a semantic parser rather than all.

III. Implementation

This work is carried out under Windows XP system on IBM R61 laptop of Intel (R) Core (TM) 2 Duo CPU, T 7100 & 1.80 GHz, 2G memory, using MyEclipse 6.0.1 as Java platform and Navicat 8 as MySQL development tool. The version of MySQL applied in this work is 5.0.51a.

Most of the researches with large number of data operations would be based on database management due to its easy control. Hence, database becomes more indispensable and critical for many current projects.

(1) Create Connection
After completing the installation of Navicat and MySQL successfully, then we have to connect them together; otherwise they are just separated software without data transportation and communication. Click the “Connection” icon in tool bar of Navicat and it pops a window of Navicat Connection. In my work, “User name” and “Password” are “root” and “Null” respectively, and other parameters are defaults. If it is a successful connection, Navicat pops a window showing “Connection Successful” message when clicking “Test Connection” button. Otherwise, it is being impossible to connect Navicat with MySQL database. See Figure 3.1.



Connection Name:	
Host name/IP address:	localhost
Port:	3306
User name:	root
Password:	

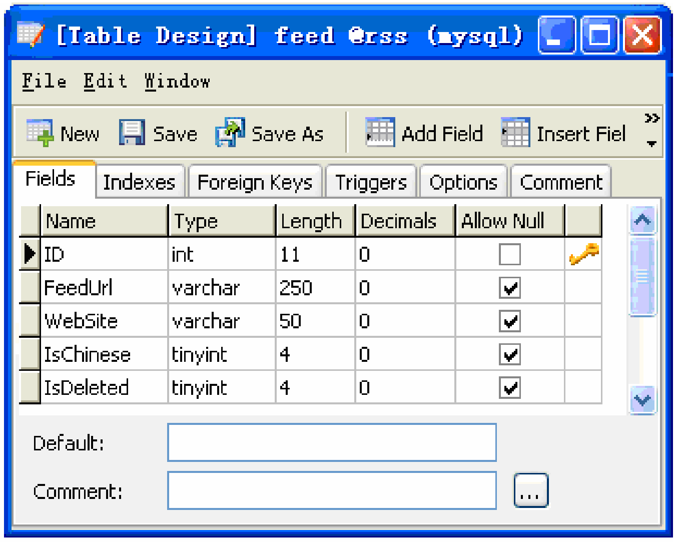
✓ Save Password

Connection Successful

OK

Figure 3.1 Connection

(2) Create Tables
As long as Navicat has been connected to MySQL successfully, then we have to design data models and create tables in MySQL. Figure 3.2 used to store the basic details of URLs of given blogs. “ID”, the series number of each blog, should keep unique. “FeedURL” is the specified URL of the corresponding blog. “WebSite” is defined as a constraint on outlinks, that is, if links appearing in URL containing sub-string equal to “WebSite”, then they are no longer deemed as outlinks. “IsChinese” filed is “1” if the URL contains Chinese character. “IsDeleted” means re-fetch (whether to obtain data again), that is, if “IsDelete” is “1”, the program no longer obtain new records from the URL.


[Table Design] feed @rss (mysql)
File Edit Window
New Save Save As Add Field Insert Fiel
Fields Indexes Foreign Keys Triggers Options Comment

Name	Type	Length	Decimals	Allow Null
ID	int	11	0	☐
FeedUrl	varchar	250	0	✓
WebSite	varchar	50	0	✓
IsChinese	tinyint	4	0	✓
IsDeleted	tinyint	4	0	✓

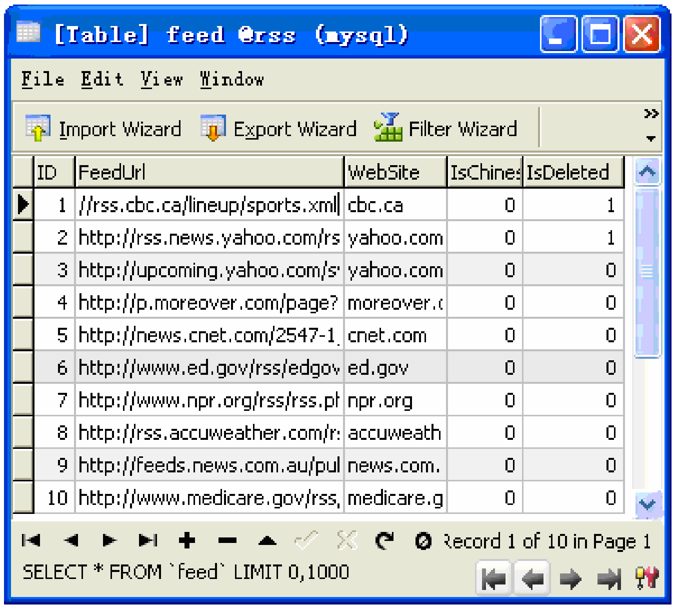
Default:
Comment:

Figure 3.2 Feed Table A

In the work, we rank the Similarity Score and Influence Score by comparing the results from the following ten different URLs,and the corresponding code format of inserting such URLs into MySQL are as follows.

insert into feed values(1,'url','website',0,0);

Figure 3.3 shows the result of “Feed table” after inserting URLs.


[Table] feed @rss (mysql)
File Edit View Window
Import Wizard Export Wizard Filter Wizard

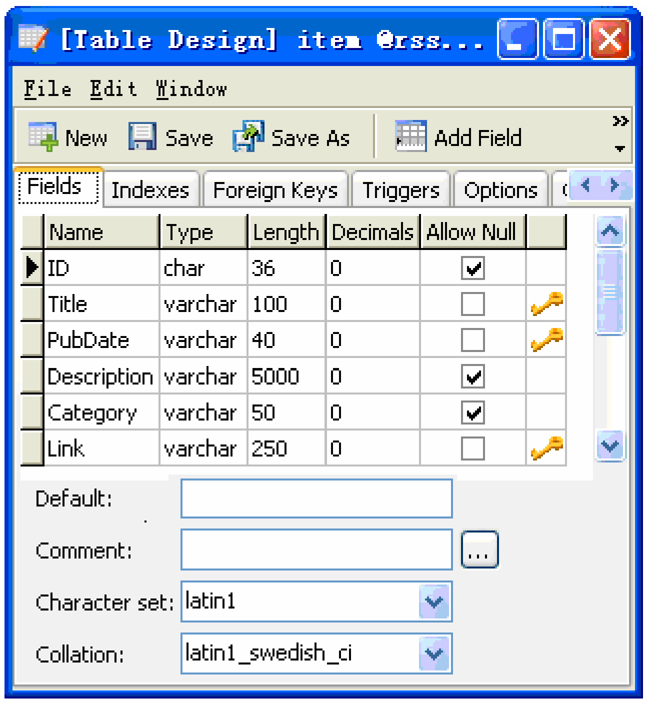
ID	FeedUrl	WebSite	IsChines	IsDeleted
1	//rss.cbc.ca/lineup/sports.xml	cbc.ca	0	1
2	http://rss.news.yahoo.com/rs	yahoo.com	0	1
3	http://upcoming.yahoo.com/s	yahoo.com	0	0
4	http://p.moreover.com/page?	moreover.c	0	0
5	http://news.cnet.com/2547-1	cnet.com	0	0
6	http://www.ed.gov/rss/edgov	ed.gov	0	0
7	http://www.npr.org/rss/rss.pl	npr.org	0	0
8	http://rss.accuweather.com/r	accuweath	0	0
9	http://feeds.news.com.au/pul	news.com.	0	0
10	http://www.medicare.gov/rss,	medicare.g	0	0

Record 1 of 10 in Page 1

```
SELECT * FROM `feed` LIMIT 0,1000
```

Figure 3.3 Feed Table B

"Item" table is designed to store all the original raw data parsed from the URLs, and the structure of "Item" table in Navicat after running codes is shown as Figure 3.4.



Name	Type	Length	Decimals	Allow Null	
ID	char	36	0	✓	
Title	varchar	100	0		🔑
PubDate	varchar	40	0		🔑
Description	varchar	5000	0	✓	
Category	varchar	50	0	✓	
Link	varchar	250	0		🔑

Default:
Comment:
Character set: latin1
Collation: latin1_swedish_ci

Figure 3.4 Item Table A

"PubDate" is used to store the date that it has been published and it is a primary argument in the function of Influence Score; "Description" field provides some abstracts of each post, usually the first paragraph of each post. "Insertdate" registers the time of each record that has been inserted into Databases.

However, due to the fact that the term weights and Vector Distance (will be discussed later) are different when calculating the Scores, hence we also have to divide up the raw data and store them into each corresponding tables. Accordingly, we design another three tables named "contentword", "titleword", "outlink" separately, all of which are used to store the information of content terms, title terms, and outlinks respectively. Hence after parsing raw data from the "Item" table, each term and link will be stored into each corresponding tables rather than aggregated them together in single "Item" table.

"contentword" and "titleword" tables have the same structure as follows, where, "Quantity" field calculates the total number of each term that occurs (frequency) in all contents records.

Table3.1. ContentWord table

Name	**Type**	**Length**
ID	char	36
ItemID	char	36
Word	varchar	50
Quantity	int	11

Table 3.2 lists the structure of Outlinks table.

Table3.2. Outlinks table

Name	**Type**	**Length**
ID	char	36
ItemID	char	36
Outlink	varchar	200

After successfully creating the infrastructure of MySQL, then we can build a program to obtain data from the given URLs and store them into each corresponding table.

(3) Build Java Platform
Apart from constructing the infrastructure in MySQL Databases, we also have to build a corresponding java program in MyEclipse environment with methods of acquiring data from URLs, and then parsing, inserting data into the aforementioned MySQL tables, and then calculate the Scores.

Step 1: Connect to MySQL
MySQL Databases infrastructure and Java program environment are two separated projects only until we bridge them together by loading the MySQL driver.

First create a java class named "DBConn", second add a java class package "mysql-connector-java-5.0.8-bin.jar" provided by Sum Corporation into the java library; and then import a java connector named "java.sql.Connection", and load a MySQL driver named "com.mysql.jdbc.Driver" into the program. Due to the fact that the Username and Password of MySQL are "root" and "Null" respectively, hence the bridge can be set up as: "Connection con = DriverManager.getConnection (temp, "root", ""), where, "temp" is a String type and equals to "jdbc:mysql://127.0.0.1:3306/rss?useUnicode=true&character Encoding=utf8"; "jdbc" is the querying mode; "127.0.0.1" is the bind address, which also is equivalent to "localhost"; "3306" is the default port; "rss" is the name of MySQL Database in this work, and "utf8" is the encoding character. Then we can query data from MySQL via Java platform by calling a "jdbc" query class and the codes are as follows:

```
Connection connection = DBConn.getConnection()
connection.createStatement().executeQuery
("SELECT * FROM tablename");
```

Step 2: Acquire Data
In MyEclipse environment we set two pointers point to URLs before fetching data so that the program knows which ones are going to be analyzed. Codes are as follows.

```
URL feedUrl = new URL(feedVO.getFeedUrl());
```

Where, feedVO is a class which stores the basic information of the given URLs, and the structure of it is constituted according to the fields listed in the Feed Table, which includes ID, FeedURL, Website, and IsChinese variables; and getFeedUrl() method returns "this.feedUrl".

However, the pointers do not know what kind of data (there are severalt types of information in each URL such as Publish Date, Category, and Abstract etc.) should be obtained since so far they have only been notified which URLs should be visited. Hence we introduce an "XPath" object (XPath, a language for selecting nodes from an XML document, is based on a tree representation and provides the ability to navigate around the tree) to fetch the aimed data from the given RSS feeds as follows,

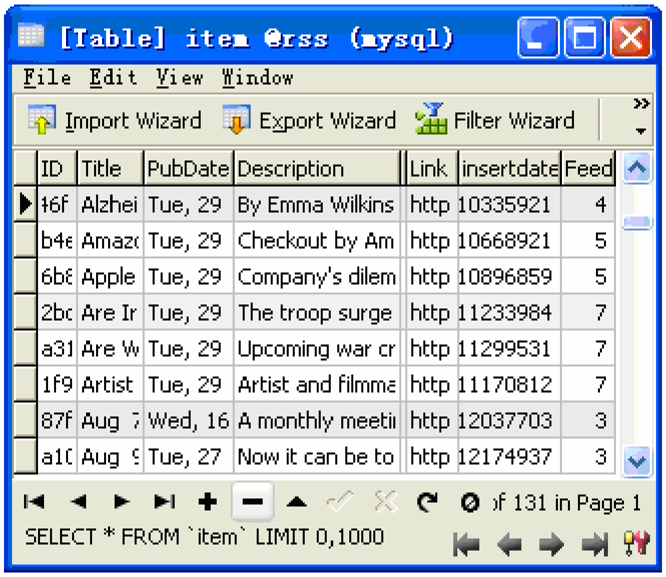
XPath xPath = XPathFactory.newInstance().newXPath();

Combined with XPath, we also have to adopt NodeList object (the NodeList interface provides the abstraction of an ordered collection of nodes without defining or constraining how this collection is implemented) correspondingly in order to fetch the matched data from the given URLs, codes are as follows,

*NodeList tableNodeList = (NodeList) xPath.evaluate (
"/rss/channel/item", URL, XPathConstants.NODESET);*

Each tag in such RSS documents has several corresponding nodes in the tree representation in terms of "rss/channel/item"; XPathConstants.NODESET" is the nodeset data type which maps to "NodeList", then we can obtain each type of raw records such as "title", "abstract", "pubdate", "link".

temp = xPath.evaluate("abstract ", tableNodeList.item(i));



ID	Title	PubDate	Description	Link	insertdate	Feed
46f	Alzhei	Tue, 29	By Emma Wilkins	http	10335921	4
b4ε	Amaz(	Tue, 29	Checkout by Am	http	10668921	5
6b8	Apple	Tue, 29	Company's dilem	http	10896859	5
2bc	Are Ir	Tue, 29	The troop surge	http	11233984	7
a31	Are W	Tue, 29	Upcoming war cr	http	11299531	7
1f9	Artist	Tue, 29	Artist and filmma	http	11170812	7
87f	Aug 7	Wed, 16	A monthly meeti	http	12037703	3
a1(	Aug 9	Tue, 27	Now it can be to	http	12174937	3

of 131 in Page 1
SELECT * FROM `item` LIMIT 0,1000

Figure 3.5 Item Table B

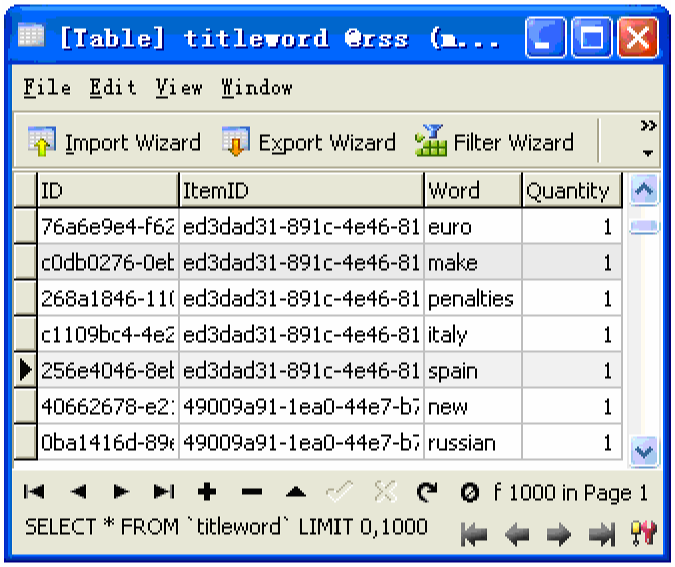
Step 3: Parse Data
Utilize Lucene to filter the aimed data (title terms, content terms, and outlinks) from the Item table. First define an analyzer:

*Analyzer analyzer = new StandardAnalyzer ();
TokenStream ts = (TokenStream) analyzer.tokenStream ("", Reader);*

Where, StandardAnalyser() is a popular Analyzer provided by Luence. Reader, an abstract class for reading character streams. analyzer.tokenStream() function is one of the methods in Analyzer class with fix format, and returns tokens. Analyzer plays as a chopping machine which cuts sentences into several single words. Codes for Outlinks acquisition are as follows,

*if (URL.getHost().toLowerCase().endsWith
(feedVO.getWebSite())) continue;*

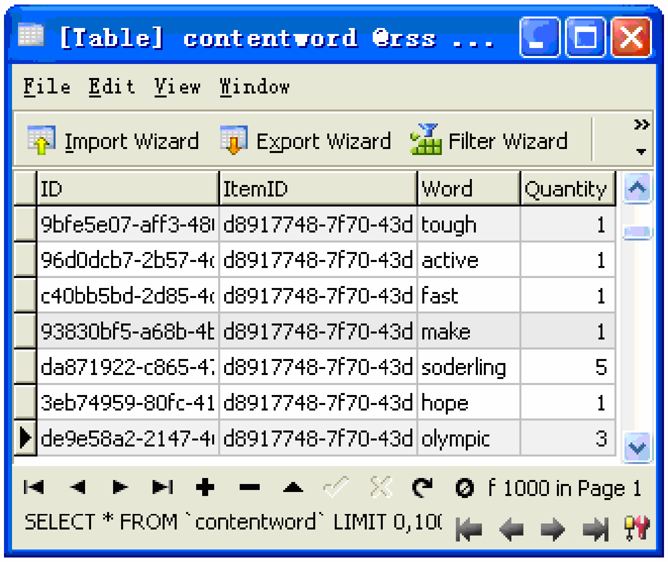
feedVO class (described before) provides the basic information of given URLs. "getHost()" gets the host name of this URL. "toLowerCase" sets all URLs lowercase. "endsWith()" checks if a link contains a sub-string that equals to "WebSite" variable. If links do contain such a sub-string， program will discard it and loop to the next record.



ID	ItemID	Word	Quantity
76a6e9e4-f62	ed3dad31-891c-4e46-81	euro	1
c0db0276-0et	ed3dad31-891c-4e46-81	make	1
268a1846-11(	ed3dad31-891c-4e46-81	penalties	1
c1109bc4-4e2	ed3dad31-891c-4e46-81	italy	1
256e4046-8et	ed3dad31-891c-4e46-81	spain	1
40662678-e2:	49009a91-1ea0-44e7-b7	new	1
0ba1416d-89ε	49009a91-1ea0-44e7-b7	russian	1

f 1000 in Page 1
SELECT * FROM `titleword` LIMIT 0,1000

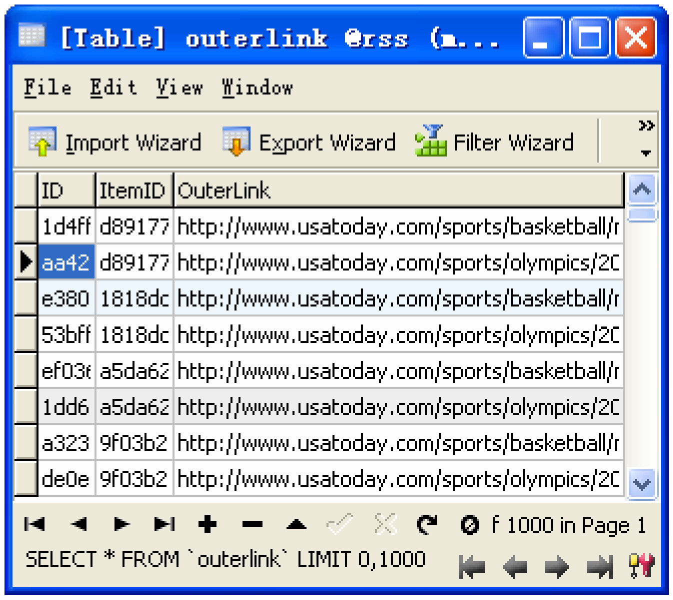
Figure 3.6 TitleWord Table



ID	ItemID	Word	Quantity
9bfe5e07-aff3-48	d8917748-7f70-43d	tough	1
96d0dcb7-2b57-4(	d8917748-7f70-43d	active	1
c40bb5bd-2d85-4(	d8917748-7f70-43d	fast	1
93830bf5-a68b-4t	d8917748-7f70-43d	make	1
da871922-c865-4:	d8917748-7f70-43d	soderling	5
3eb74959-80fc-41	d8917748-7f70-43d	hope	1
de9e58a2-2147-4(	d8917748-7f70-43d	olympic	3

f 1000 in Page 1
SELECT * FROM `contentword` LIMIT 0,10(

Figure 3.7 ContentWord Table



ID	ItemID	OuterLink
1d4ff	d89177	http://www.usatoday.com/sports/basketball/r
aa42	d89177	http://www.usatoday.com/sports/olympics/2C
e380	1818dc	http://www.usatoday.com/sports/basketball/r
53bff	1818dc	http://www.usatoday.com/sports/olympics/2C
ef03(	a5da62	http://www.usatoday.com/sports/basketball/r
1dd6	a5da62	http://www.usatoday.com/sports/olympics/2C
a323	9f03b2	http://www.usatoday.com/sports/basketball/r
de0e	9f03b2	http://www.usatoday.com/sports/olympics/2C

```
SELECT * FROM `outerlink` LIMIT 0,1000
```

Figure 3.8 OutLinks

IV. Algorithm and Experiment

Due to the spatial and temporal limitation, we select 1000 terms which occur most frequently to participate in this work

(1) Similarity Analysis
Similarity Analysis is a research on searching for "connected blogs" -- similarity in topic and similarity in links. That is, given two URLs, we exploit a novel technique to suggest the similarity of blogs by applying the observation that how many terms are in common or overlap, how many times a term occurs (we name it "TF (Term Frequency)"), and how many outlinks are similar. First, we define (VD) Vector Distance as follows:

$$VD = \frac{\sqrt{(tfA1 - tfB1)^2 + ... + (tfA1000 - tfB1000)^2}}{m}$$

Where, tfAi means the TF of the ith keyword of blog A, as the same, tfBi means the TF of the ith keyword of blog B. m is the penalization factor, which equals the total number of terms in the whole universe set.

Hence we define the ideas of VD_of_Title (Vector Distance of terms in title), VD_of_Content (Vector Distance of terms in content), and VD_of_Link as follows.
*VD_of_Title = VD (tfAi/tfBi are from titleword table).
*VD_of_Content = VD (tfAi/tfBi are from contentword table).
*VD_of_Link = the number of common outlinks in two URLs.

Then we define the function of Similarity Score as follows:

$$Score = wt * (\frac{1}{VD\_of\_Title}) + wc * (\frac{1}{VD\_of\_Content}) + wl * (VD\_of\_Link)$$

Where, wt, wc, wl are the weights of each VD, and set 15, 12, and 10 respectively in this work.

From the Similarity Score function, we know that there is an inverse relationship between the Score and the VD_of_Title, that is, the more similar the terms are, the smaller the VD_of_Title is, and hence a bigger Score we get. And it is as the same as the VD_of_Content. However, the Score goes in proportional relationship with the VD_of_Link, the more pairs of outlinks are in common, the bigger Similarity Score it is. However, ten selected blogs deem themselves as authoritative pages that the similar OutLinks in this experiment are all zero.

For 10 different Feeds there are: $C_{10}^{2} = \frac{10*9}{2*1} = 45$ pairs of choices. However, due to the spatial limitation, we only list the results of some significant and representative pairs of blogs here and Table 4.1 shows a clear comparison of these RSS Feeds.

Table 4.1: Some Pairs of Similarity Scores

No.	Each pair of URLs	Quantity of common terms (title/content)	Score
1.	Blog A: 1 (same order with Blog B: 2 Feed Table)	120 / 701	80.3
2	Blog A: 7 Blog B: 9	5/275	30.8
3	Blog A: 3 Blog B: 7	1/185	5.95
4	Blog A: 2 Blog B: 3	4/185	5.25
5	Blog A: 4 Blog B: 7	0/141	2.49

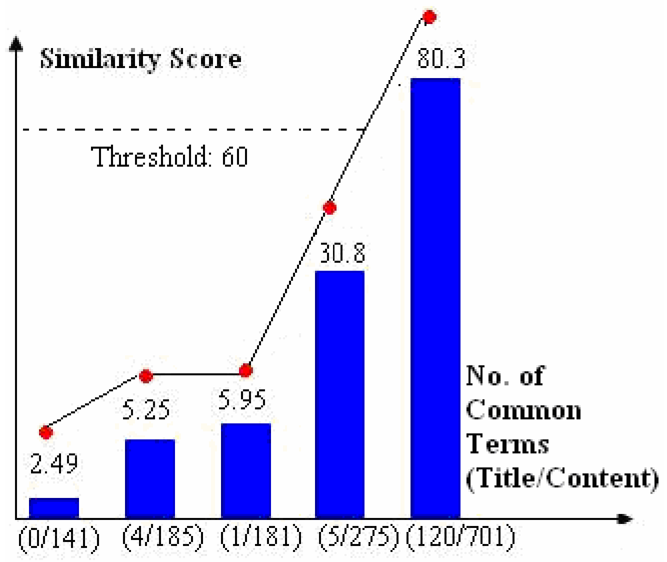


Figure 4.1 Comparison among Similarity Scores

Figure 4.1 shows a curve for the comparison among some pairs of blogs. Due to the fact that the scores of most unrelated pairs of URLs are under 60, hence it is meaningful to defined the threshold of Similarity Score as 60, that is, URLs with Similarity Scores higher than 60 are deemed as similar blogs.

(2) Influence Analysis

Influence Analysis is based on determining whether a blog mentions a keyword/link earlier than another by adding directionality and searching Influence Relation among blogs. And we defined these blogs as "authority blogs".

With extending the functions of VD_of_Title, VD_of_Content, and VD_of_Link, we describe another argument named PTPE (the percentage of overlapped terms whose first occurrences have been published earlier in Blog A than in Blog B). Then we give the definition of Influence Score function as follow.

$$\text{Score} = \text{wt} * PTPE\_\text{T} * \left(\frac{1}{VD\_of\_Title}\right) + \text{wc} * PTPE\_\text{C} * \left(\frac{1}{VD\_of\_Content}\right) + \text{wl} * PTPE\_\text{L} * (VD\_of\_Link)$$

Where, PTPE_T, PTPE_C, PTPE_L are the PTPE of Titles, Contents, and Outlinks respectively; wt, wc, wl are the weights of each corresponding PTPE and different from the ones in Similarity, we set wt = 3, wc = 1, wl = 2.

Table 4.2: Some Pairs of Influence Scores

No.	URLs	Quantity of Title terms (early/overlap)	Quantity of contents terms (early/overlap)	Score
1.	Blog A: 1 Blog B: 2	51/120	275/701	6.52
2.	Blog A: 2 Blog B: 1	69/120	426/701	8.48
3	Blog A: 7 Blog B: 9	4/5	196/275	4.04
4	Blog A: 1 Blog B: 9	3/3	187/267	3.32
5	Blog A: 2 Blog B: 3	3/4	67/185	0.57

From Table 4.2, in first comparison, there are 51 out of 120 overlapped terms are published earlier from Title, and 275 out of 701 from Content and the Influence Score of blog A on B is 6.52112078643748. However, Influence Score is a bidirectional function, after switching blog A and blog B, then that of blog B on A showing in the computer is 8.47899304383688 due to the fact that the occurrences of overlapped terms published earlier in blog A are 69 out of 120 and 426 out of 701 in Title and Content respectively. Therefore, it is evident to draw the conclusion that Blog 1 is a slight more authoritative than Blog 2. Accordingly, Figure 4.2 shows the Influence Score for the comparison among some pairs of blogs.

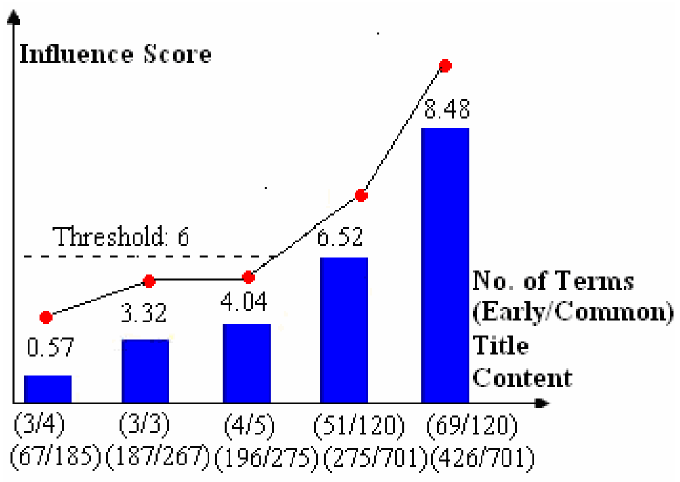


Figure 4.2 Comparison among Influence Scores

From Figure 4.2, we find out that the scores of less influenced pairs of URLs are less than 6; hence it is reasonable to set the threshold of Influence Score 6 in this work, that is, Influence Scores bigger than 6 are considered as influential pairs and Blog 2 are considered to be authoritative blog.

The Web is not a static environment and changes constantly. Influence Blogs in the past may not be quality pages now or in the future. Hence it is meaningful to study Influence Analysis to find out which blog with popular significant terms is published first.

V. Conclusions and Future Work

We construct a Blog Quality Analysis system by employing techniques with computationally efficient such as RSS Feed, Lucene, and MySQl in order to discover correlated terms, mine hot keywords, and enhance ranking functions.

Blogs quality analysis is a novel academic study; in which, RSS and Lucene have supplied several successful elegant and delicate solutions for related problems. In any case, Blogs quality analysis technology develops continuously; solutions to problems in these areas will correspondingly find their way from theory into practice quickly.

References

[1] (Basic introduction of Lucene) http://en.wikipedia.org/wiki/Lucene
[2] (Online sources, basic introduction of GData) http://code.google.com/apis/gdata/overview.html
[3] (Online RSS introduction) http://www.w3schools.com/rss/rss_intro.asp
[4] Nilesh Bansal, Nick Koudas. Searching the Blogosphere. In Proceedings of the 10th international Workshop on Web and Databases, WebDB 2007, (co-located with SIGMOD) Beijing, China, June 15 2007.
[5] Nilesh Bansal, Fei Chiang, Nick Koudas, Frank Wm. Tompa. Seeking Stable Clusters in the Blogosphere. In Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, Sept 23-28 2007.