CS297 Report

Improving the BM25F algorithm for use with OPIC based crawlers

Ravi Inder Singh Dhillon
ravi.dhillon@yahoo.com

Fall 2010

Advisor: Dr. Chris Pollett
San José State University
Department of Computer Science
One Washington Square
San Jose, CA 95192-0249
Ph: (408) 924-5145
http://www.cs.sjsu.edu/faculty/pollett/

Table of Contents

1. Introduction

The CS297/CS298 project aims at improving the BM25F algorithm for use with OPIC based crawlers. In contrast to algorithms which compute the page rank offline using a link matrix, OPIC does an online computation of Page rank while crawling the web. BM25 is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Sparck Jones, and others. BM25F is a modification of BM25 in which the document is considered to be composed from several fields (such as title, body, and anchor text) with possibly different degrees of importance. Thus the page relevance is based on weights assigned to these fields. The title and body of a document are termed as document fields. The anchor field of a document refers to all the anchor text in the collection pointing to a particular document. Thus if a lot of unimportant links are pointing to a document they can increase the page relevance of an important web page for unimportant word searches that are not relevant to it. Therefore we aim to improve BM25F by considering page rank computed by OPIC algorithm while computing weight for anchor field associated with a web page.

The CS297 report details the various activities that were carried out to prepare for the final CS298 project. The first task was to get familiar with the Yioop open source search engine which would be used to implement the project. This was accomplished by installing Yioop, making it perform a crawl and studying its source code. Second task was to gain an understanding of the workings of the Online page importance algorithm.

The algorithm was studied and then implemented from scratch using PHP. Third task was to deeply understand the functioning of BM25F algorithm. The algorithm was implemented from scratch using PHP. The first section of the report contains the details about the first task and the following sections have details about the second and third tasks. The last two sections contain the conclusion and the references.

2. Understanding YIOOP

This task was to understand the working of open source search engine YIOOP developed by Prof. Chris Pollett. While studying the YIOOP source code I fixed the following issues in its UI.

1) Restrict Sites By Url check box in Edit crawl options page was not working in Internet Explorer.

2) A logged in user was automatically logged out when search button was hit in the main search page.

2.2 Installation of Yioop and perform a crawl

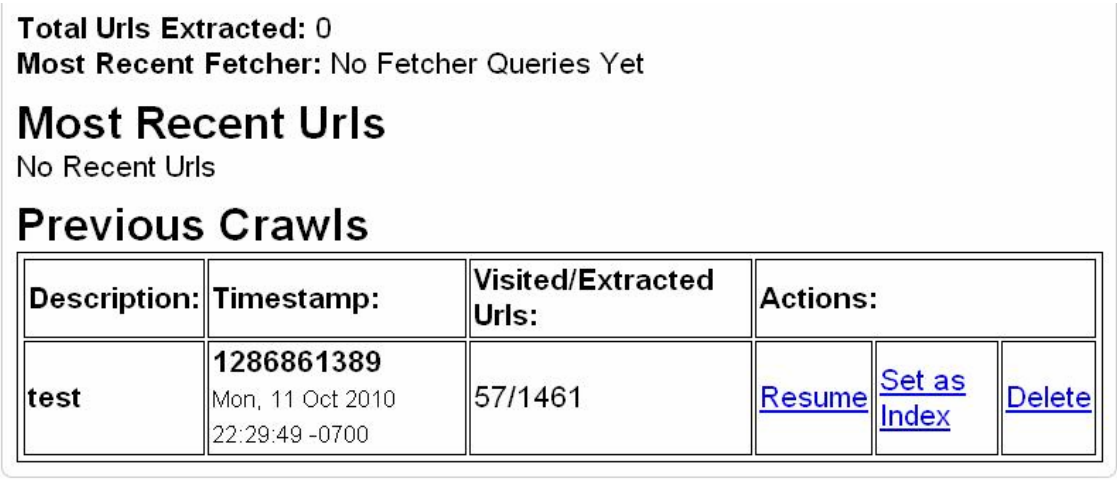
The machine used for installation of Yioop was a Windows XP Professional running on Intel Core2Duo with 2GB memory. The machine had the latest Apache, PHP and MYSQL bundle installed. The latest version of Yioop was downloaded from www.Seekquarry.com and placed in the document root of Apache server. The one time configuration was done and the data directory for the crawl was created. The steps for crawl were followed from www.seekQuarry.com and the queue_Server and fetcher scripts were started using the PHP command line. Queue_server.php maintains a queue of

urls that are going to be scheduled to be seen. It also keeps track of what has been seen and robots.txt info. Its last responsibility is to create the index_archive that is used by the search front end. Fetcher.php downloads batches of urls provided by the queue_server.

However the crawl was unsuccessful because the fetcher was unable to send information about the crawl back to the queue_server. This information was sent to the queue_server as POST data using curl. Careful observation and experimentation revealed that it was failing because the URL for queue_sever in the configuration setting was incorrect. A trailing slash should be appended to the path of queue server in the configuration. If the trailing slash is not present at the end of URL for queue server, it causes the redirection of page and the posted data is not sent once the redirect occurs.

i.e. it should look like **http://localhost/yioop/** rather than **http://localhost/yioop**

The URL was modified and crawl was started again. This time the fetcher successfully started sending information about crawled sites back to the queue server and it was stored in the data directory. The Yioop search screen started fetching results for user queries against the crawl data.



2.3 Description and Fix for First Issue in UI

The Restrict Sites By URL check box in the Edit crawl options page was dynamically displaying a text area in Mozilla firefox but not in Internet Explorer. The code was traced to Javascript inside basic.js file.The check box element was using an onchange DOM event to invoke a Javascript function which changed the visibility of text area element. The onchange event occurs when the content of a fileupload, select, text and textarea Javascript objects changes. Hence the event was not triggered in Internet explorer when the check box was selected. To fix the issue the onchange DOM event was changed to onclick DOM event. This DOM event is supported by checkbox element and triggered when element is clicked to change its state. After the change was made the issue was fixed for IE and also worked fine for mozilla.

2.4 Description and Fix for Second Issue in UI

A registered user can log into Yioop and perform admin tasks. A logged in user can also return to the main Yioop search page to use the search engine to get search results. When this user enters a query and hits the search button the search results are displayed for the query. However when this was done the user was automatically logged out of the system. To prevent CSRF attacks Yioop uses a token to authenticate activities performed by a user logged into the system. When a user logs in, this token is generated and placed as a hidden field inside the query form. This token is verified when query is submitted to check a valid user session. Since the token was missing inside the query form, user session was being terminated. Hence this hidden field was added to the query form to fix the issue. The code fragment added to form is given on next page.

<input type="hidden" name="YIOOP_TOKEN" value="<?php e($data['YIOOP_TOKEN']); ?>" />

3. Deliverable 2

3.1 Goal

The goal of this deliverable was to understand the workings of the OPIC algorithm and implement the algorithm in PHP from scratch.

3.2 OPIC Algorithm

The OPIC algorithm works on the principle as follows: Initially, some “cash” is distributed to each page and each page when it is crawled distributes its current cash equally to all pages it points to. Cash can be defined as the numerical value allotted to each page. The static nodes of the matrix represent the web pages. This is recorded in the history of the page. The importance of a page is then obtained from the “credit history” of the page.

The idea is that the flow of cash through a page is proportional to its importance. At each step, an estimate of any page **k’s** importance is **(H[k] + C[k]) / (G + 1)**, where **H[k]** represents history, **C[k]** represents cash and **G** is the total cash accumulated. The algorithm is executed over a decided number of iterations until the acceptable rate of convergence is achieved.

3.3 OPIC Simulator

As a part of the deliverable, a simulator was developed in PHP to simulate the workings of the Online Page Importance Calculation Algorithm (OPIC). This tool uses n × n static nodes (n can be configured in the configuration file) to depict the workings of a static network. A connection of nodes is activated or deactivated by checking or unchecking the radio button on the adjacency matrix.

The snapshot of the tool below shows the links between pages set for a 5 × 5 matrix

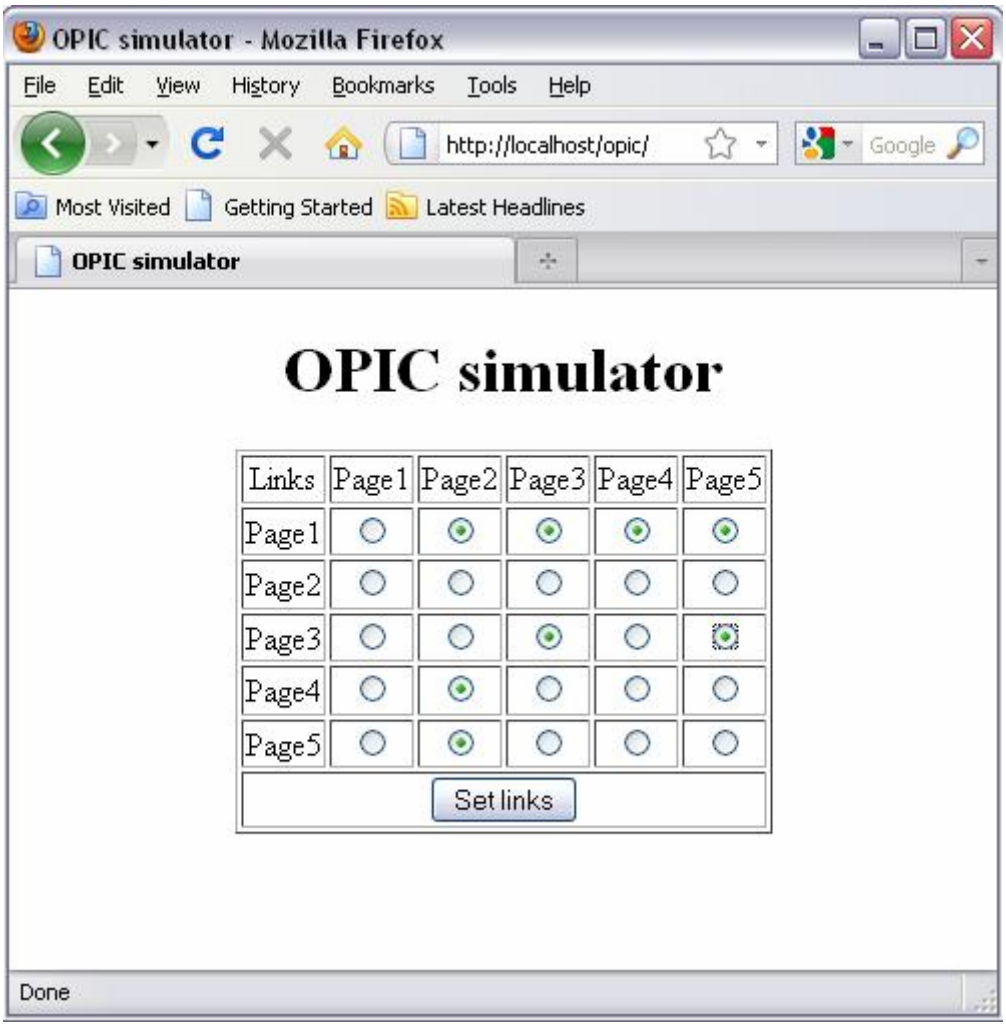
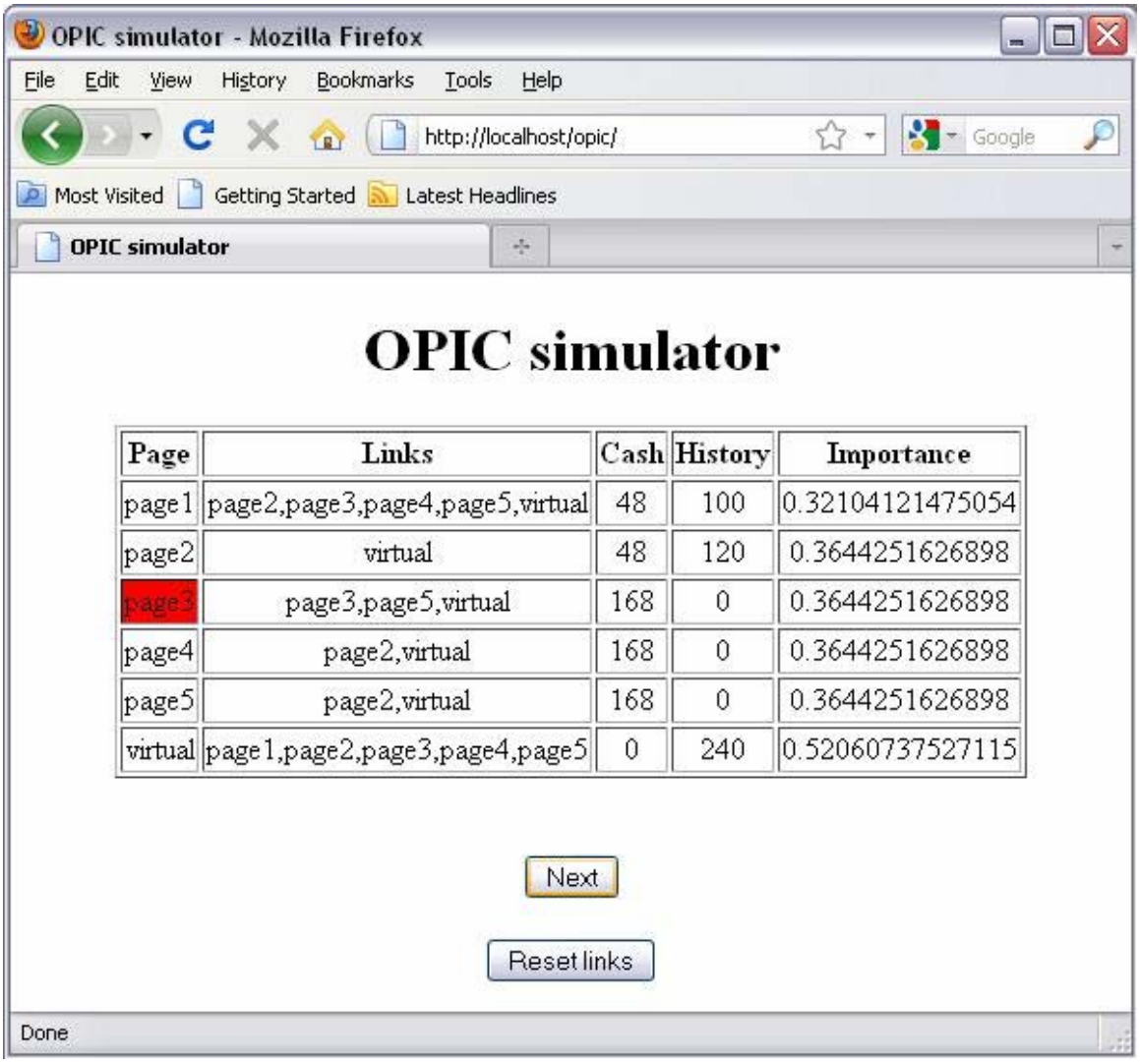


Figure 1: Setting up links for OPIC simulator
Source: SJSU CS Project portal. Retrieved Dec 6, 2010

After setting the links the user can iterate through the algorithm by clicking the next button. The table contains information about the pages, their links, cash, history and importance. The page with a red background is the page which is currently being visited by the algorithm.

The figure below shows the snapshot of one of the iterations.



Page	Links	Cash	History	Importance
page1	page2,page3,page4,page5,virtual	48	100	0.32104121475054
page2	virtual	48	120	0.3644251626898
page3	page3,page5,virtual	168	0	0.3644251626898
page4	page2,virtual	168	0	0.3644251626898
page5	page2,virtual	168	0	0.3644251626898
virtual	page1,page2,page3,page4,page5	0	240	0.52060737527115

Figure 2: Iteration of the OPIC simulator
Source: SJSU CS Project portal. Retrieved Dec 6, 2010

4. Deliverable 3

4.1 Goal

The goal of this deliverable was to understand the workings of the BM25F algorithm and to implement the algorithm in PHP from scratch.

4.2 BM25 Algorithm

In information retrieval, Okapi BM25 is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. BM25 ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. One of the most prominent instantiations of the function is as follows.

Given a query q, containing terms $t_1,...,t_n$, the BM25 score of a document d is:

$$R(q,d) = \sum_{t \in q} \frac{occurs_t^d}{k_1((1-b) + b\frac{l_d}{avl_d}) + occurs_t^d}$$

where $\boldsymbol{occurs_t^d}$ is the term frequency of $\boldsymbol{t}$ in $\boldsymbol{d}$; $\boldsymbol{l_d}$is the document length; $\boldsymbol{avl_d}$is the document average length along the collection; $\boldsymbol{k_1}$is a free parameter usually 2 and $\boldsymbol{b}$ belongs to [0,1] (usually 0.75). Assigning 0 to $\boldsymbol{b}$ is equivalent to avoid the process of normalisation and therefore the document length will not affect the final score. If $\boldsymbol{b}$ takes 1, we will be carrying out a full length normalisation.

$$idf(t) = log\frac{N - df(t) + 0.5}{df(t) + 0.5}$$

where ***N*** is the number of document in the collection and ***df*** is the number of documents where appears the term ***t***.

4.3 BM25F Algorithm

BM25F is a newer variant of BM25 that can take document structure and anchor text into account. First we obtain the accumulated weight of a term over all fields as

$$weight(t,d) = \sum_{c\ in\ d} \frac{occurs_{t,c}^{d} \cdot boost_c}{((1-b_c) + b_c \cdot \frac{l_c}{avl_c})}$$

where $\boldsymbol{l_c}$ is the field length; $\boldsymbol{avl_c}$ is the average length for the field ***c*** ; $\boldsymbol{b_c}$ is a constant related to the field length, similar to ***b*** in BM25 and $\boldsymbol{boost_c}$ is the boost factor applied to field ***c***. Next, a non-linear saturation is applied

$$R(q,d) = \sum_{t\ in\ q} \frac{weight(t,d)}{k_1 + weight(t,d)} \cdot idf(t)$$

and equal equation for ***idf(t)***

$$idf(t) = log \frac{N - df(t) + 0.5}{df(t) + 0.5}$$

where ***N*** is the number of document in the collection and ***df*** is the number of documents where appears the term ***t***.

4.3 BM25F Simulator

To simulate the BM25F algorithm a tool was created in PHP which takes as input 100 websites and extract all the terms that occur in them. It then computes the BM25F score for all these terms based on where they occur in the document. This information is stored in a table which is used by the tool which accepts a user query. Based on the query the tool generates the combined BM25F score of query for all the 100 pages. Below are the snapshots of the tool in use for a query.

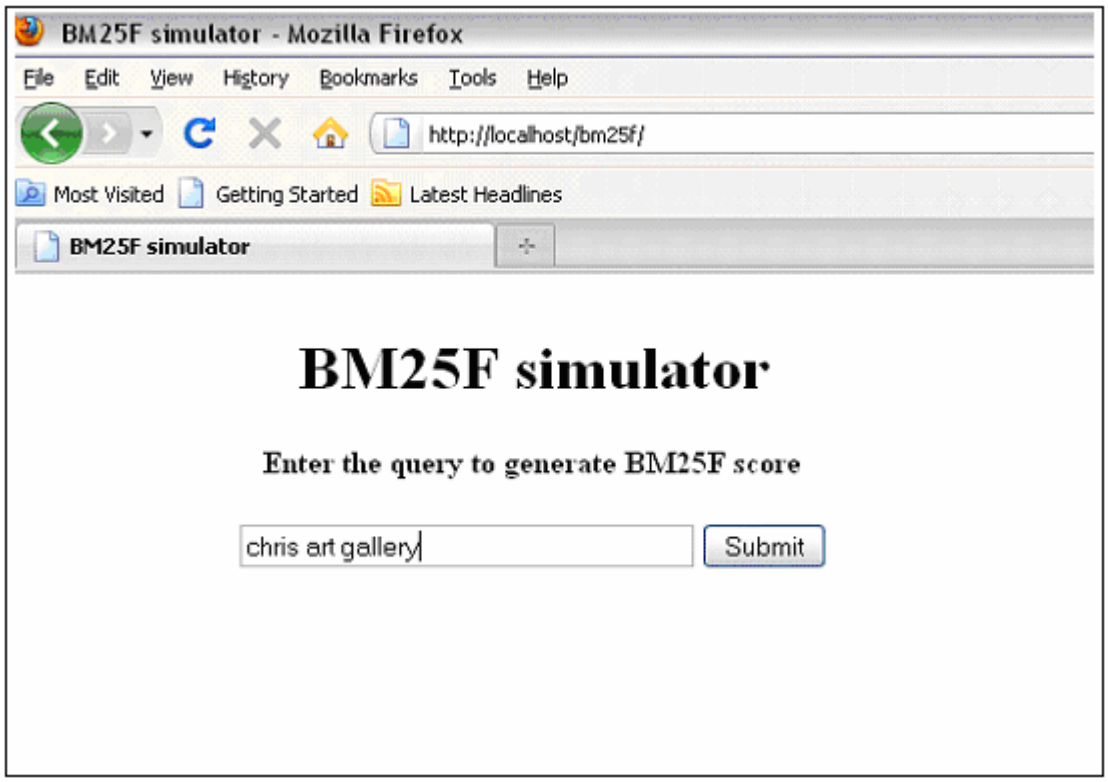
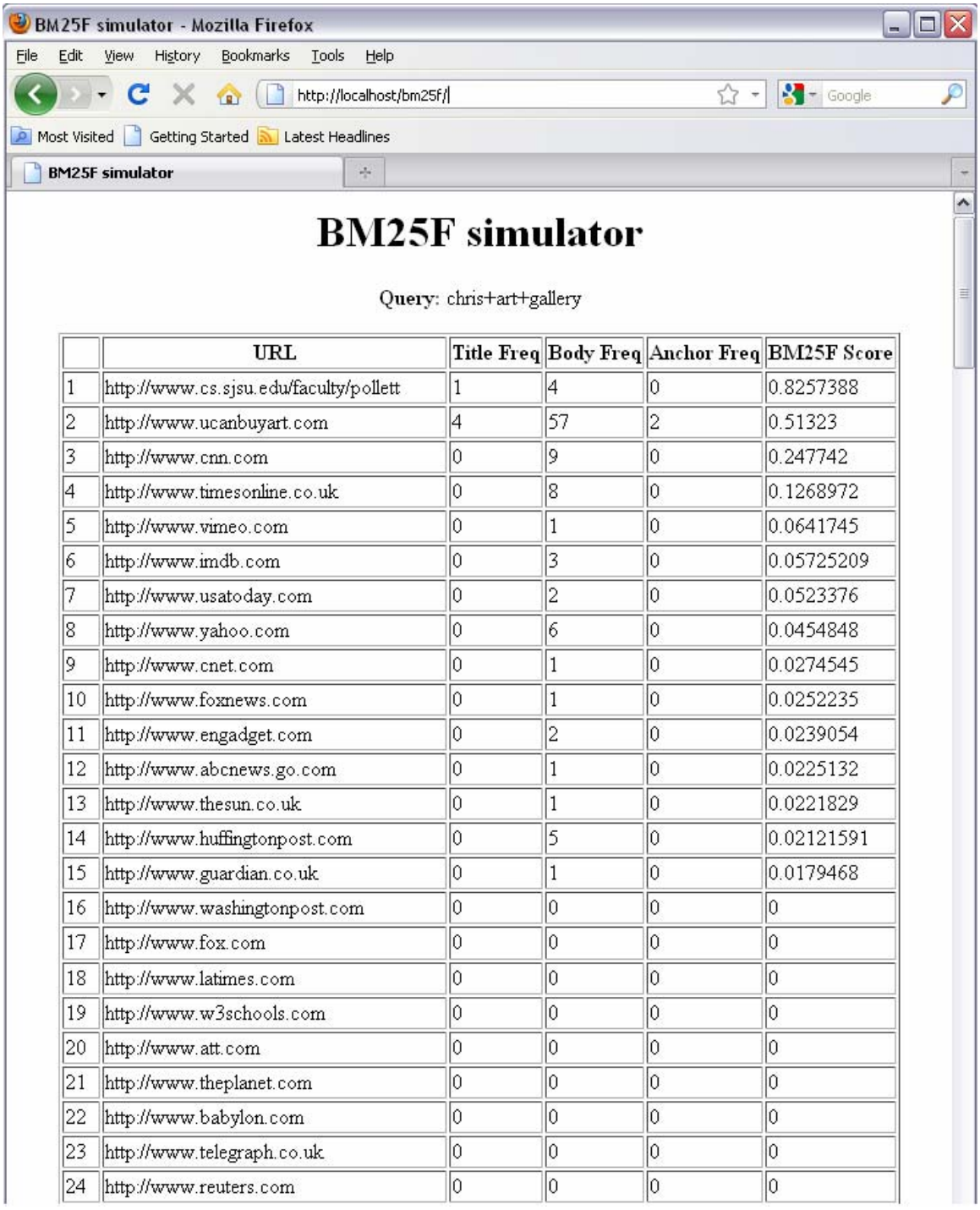


Figure 3: Query in BM25F simulator
Source: SJSU CS Project portal. Retrieved Dec 6, 2010



BM25F simulator

Query: chris+art+gallery

	URL	Title Freq	Body Freq	Anchor Freq	BM25F Score
1	http://www.cs.sjsu.edu/faculty/pollett	1	4	0	0.8257388
2	http://www.ucanbuyart.com	4	57	2	0.51323
3	http://www.cnn.com	0	9	0	0.247742
4	http://www.timesonline.co.uk	0	8	0	0.1268972
5	http://www.vimeo.com	0	1	0	0.0641745
6	http://www.imdb.com	0	3	0	0.05725209
7	http://www.usatoday.com	0	2	0	0.0523376
8	http://www.yahoo.com	0	6	0	0.0454848
9	http://www.cnet.com	0	1	0	0.0274545
10	http://www.foxnews.com	0	1	0	0.0252235
11	http://www.engadget.com	0	2	0	0.0239054
12	http://www.abcnews.go.com	0	1	0	0.0225132
13	http://www.thesun.co.uk	0	1	0	0.0221829
14	http://www.huffingtonpost.com	0	5	0	0.02121591
15	http://www.guardian.co.uk	0	1	0	0.0179468
16	http://www.washingtonpost.com	0	0	0	0
17	http://www.fox.com	0	0	0	0
18	http://www.latimes.com	0	0	0	0
19	http://www.w3schools.com	0	0	0	0
20	http://www.att.com	0	0	0	0
21	http://www.theplanet.com	0	0	0	0
22	http://www.babylon.com	0	0	0	0
23	http://www.telegraph.co.uk	0	0	0	0
24	http://www.reuters.com	0	0	0	0

Figure 4: BM25F score of Query for all pages
Source: SJSU CS Project portal. Retrieved Dec 6, 2010

5. Conclusion

CS297 was used to acquire understanding about the components which would be used to implement the final CS298 project. The YIOOP open source engine was studied which will incorporate the final implementation of the project. The major algorithms which would be required for implementation i.e. OPIC and BM25F were studied thoroughly and implemented using PHP.

CS298 next semester will use the learning's which were gained in CS297. The BM25F ranking function calculates the page relevance by assigning weights to document fields and the anchor field. The title and body of a document are termed as document fields. The anchor field of a document refers to all the anchor text in the collection pointing to a particular document. Thus if a lot of unimportant links are pointing to a document they can increase the page relevance of an important web page for unimportant word searches that are not relevant to it. In CS298 we will implement a modified BM25F by considering page rank computed by OPIC algorithm while computing weight for anchor field associated with a web page. This modified BM25F will provide a better estimate of the page relevance.

Currently, the project is still in the analysis phase and more information related to the specific implementation is to be researched and gathered. This includes creating small test models, researching related journal articles, analyzing reference guides, generating test cases etc. This will help to understand the overall processes involved in successfully implementing the modified BM25F algorithm.

Based on this analysis, the solution for the final project (CS 298) will be implemented. Future work will involve carrying on further research and analysis while using the findings from CS 297 to implement the final solution. This will involve using the PHP BM25F implementation as the basis for building the modified BM25F ranking algorithm. This will be tested using the YIOOP search engine and comparing the results with the earlier version of the algorithm. This evaluation will be helpful in comparing the effectiveness of the project.

6. References

[1]Serge Abiteboul and Mihai Preda and Gregory Cobena.(2003). *Adaptive on-line page importance computation.* In: Proceedings of the 12th international conference on World Wide Web. pp.280-290.

[2] Hugo Zaragoza, Nick Craswell, Michael Taylor, Suchi Saria, and Stephen Robertson. Microsoft Cambridge at TREC-13 (2004): *Web and HARD tracks*. In Proceedings of 3th Annual Text Retrieval Conference.

[3] Paolo Boldi and Massimo Santini and Sebastiano Vigna (2004). *Do Your Worst to Make the Best: Paradoxical Effects in PageRank Incremental Computations*. Algorithms and Models for the Web-Graph. pp. 168-180.

[4] Amy N. Langville and Carl D. Meyer (2006). *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press.

[5] Okapi BM25F. Wikipedia, the free encyclopedia.