

An Implicit-Feedback Based Ranking Methodology For Web Search Engines

Shahram Rahimi, Raheel Ahmad, Bidyut Gupta, Kaushik Adya
Department of Computer Science, Southern Illinois University
Carbondale, IL 62901
[rahimi, rahmad, bidyut]@cs.siu.edu
Tel: 618-453-6033

Abstract

The World Wide Web (WWW) is a fast growing network of information covering nearly every possible topic. With the input of a few keywords a search engine can return a list of relevant web pages by querying its index. However, it is quite common to witness irrelevant results being presented to the user. One way to improve the ordering of the search results is by incorporating user feedback in ranking the documents for relevancy. We present a model for search engine enhancement by using implicit feedback in the form of ClickThrough data from the users. The order of the links returned as the query result is re-arranged for the future queries based on the choices made by the majority of the users. An algorithm, with its implementation, is presented and then evaluated to demonstrate its capability as an add-on component for enhancement of the current ranking algorithms.

Keywords: Search Engine, Ranking Algorithms, Clickthrough Data, Implicit Feedback.

1 Introduction

The task of ranking the documents, according to some predefined criteria, falls under the responsibilities of the *ranking algorithms*. A ranking algorithm is one of the most crucial components of any search engine and usually requires much attention during the engine's development. Different search engines use different classes of ranking algorithms with varying degree of effectiveness and efficiency. Intuitively, a good information retrieval system should present relevant documents higher in the ranking, with less relevant documents following them. Although the ranking algorithms, following the search process, strive hard to achieve this goal, it is common to witness many irrelevant among the relevant queried information. This sub-optimal result has led to several researches in the area of search engine ranking algorithms.

This work presents a system for automatically rearranging the query results of an arbitrary search engine using the implicit feedback obtained from users in the form of what is known as "ClickThrough" data. Such ClickThrough data is easily available and can be recorded at a very low cost. In the following sections, we discuss the inadequacies with the current ranking approaches, the proposed approach along with some implementation details, and the results. This will be followed by the evaluation of the implementation and conclusions.

2 Document Ranking

Ranking algorithm is one of the crucial components of any search engine and plays an important role in its effectiveness. The satisfaction of the user lies on the links to the documents returned by this ranking algorithm. Different search engines use different ranking algorithms and most of them are proprietary and a well-kept secret. These algorithms include certain assumptions in order to rank the documents. Google, one of the most popular search engines, uses its own PageRank [3] algorithm. The most crucial aspect of the PageRank algorithm is that it interprets a link from page A to page B as a vote, by page A, for page B [4]. Along with this many other criteria are combined for the ranking procedure. TF x IDF is another ranking methodology used by search engines such as WebCrawler and Lycos [5]. This makes use of the term frequency (TF) in a document and how often the term is used in the collection of documents (IDF). Some of the other ranking methodologies are Boolean Spread Activation, Most-Cited and Vector Spreading Activation discussed in [6].

2.1 Drawbacks of Current Approaches

As seen in the above algorithms, the importance of a page is based on the metrics such as interest, popularity, location, etc. In almost all the ranking algorithms, the facts considered are the linkage, keywords, format of the words, position of the words, depth of the page in the domain, etc. In general the search engine user should be able to garner the information from the top links returned. But the user may not always find the most relevant links among the top few results. This situation occurs

because the user's choice has not been given any importance. One way to overcome this would be to use the feedback from users to re-rank the pages.

One can observe that none of the ranking algorithms listed above consider the feedback of the users to project the results. One evident reason for this lapse could be that users cannot be depended on to leave enough proper feedback explicitly, no matter how easy it is. One way is to get implicit feedback, based on the behaviors of the users. In this scenario, ClickThrough data is extremely useful to learn about the users' choice. In the next section, we will see present ClickThrough data and how it can be used to enhance search engines results.

3. ClickThrough Data and the Proposed Model

ClickThrough data in search engines can be thought of as triplets (q, r, c) , consisting of the query q , the ranking r presented to the user, and the set c of links the user clicks on. Clearly, users do not click on the provided links at random, but make a somewhat informed choice. While ClickThrough data typically consist of some amount of noise and clicks are not perfect relevance judgment metrics, they are likely to convey overall reliable information regarding the relevancy of pages to a search query.

3.1 Incorporating the ClickThrough Data from a Single User

ClickThrough data can be recorded with little overhead and without compromising the functionality of the search engine. The query q and the returned ranking can be easily recorded whenever the resulting ranking is displayed to the user. For recording the clicks, a simple proxy system is employed as shown in Figure 1, which is discussed later.

There are strong dependencies between the three parts of the triplet (q, r, c) . The presented ranking r depends on the query q as determined by the retrieval function implemented in the search engine. Furthermore, the set c of clicked links depends on both the query q and the presented ranking r . A user is more likely to click first on a link if it is relevant to q [7, 8]. Therefore, in order to get interpretable and meaningful results from ClickThrough data, it is necessary to consider and model the dependencies of c on q and r appropriately.

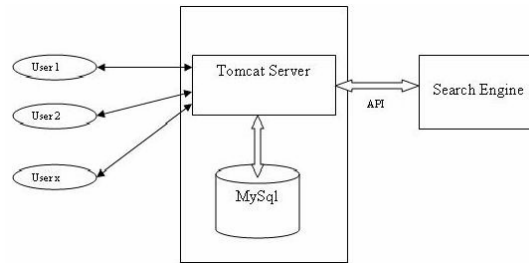


Figure 1. The Proposed Model

The strategy of utilizing user feedback to re-arrange the ordering of the query results could be given as (in this paper $link_k$ is also represented as l_k):

If a user clicks on links l_i and l_j in r , where $i < j$ and there exists no other clicked links in between i and j then {
If $i < j-1$ then,
in r_k the partial order will be $\dots l_i, l_j, l_{(i+1)}, \dots, l_{(j-1)}, l_{(j+1)}, l_{(j+2)} \dots$
Else maintain the same order as in r
}
Else maintain the same order as in r .

Following the above algorithm, the ordering based on the users' choice can be produced.

3.2 The Proposed Multi-User Feedback Approach

The proposed approach makes use of the algorithm presented in the previous section. For a query, q , we obtain the r_i for a considerable number of users, say ' n '. After obtaining the required data, the average displacement of the documents is calculated according to the feedbacks obtained in the above fashion. Once n such instances are collected, we re-order the documents for query q with r^* ordering. The procedure to calculate r^* is explained next:

For a ranking function F and a document set D , the query phrase q results in the ranking r : $(l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8 \dots, l_k)$, such that

$$F(q) = r: l_1 <_r l_2 <_r l_3 <_r l_4 <_r l_5 <_r l_6 <_r l_7 <_r l_8 <_r \dots <_r l_k$$

Rank of a document d , for the query q , can be defined as the corresponding position of its link in the ranking function $F(q)$. Δ is an integer array of size k , initially set to 0. This is used to store the displacements of the links in r^* with respect to r . Displacement of the document d in r_i compared to r is given by the change in position of its link l . It can be either negative or positive. Every time a user queries for q using F and clicks on some of the returned links, a new r_i is generated. For each

one of these r_i s the relative changes in the positions of the links are calculated and then inserted into the corresponding position in Δ (for r_i it would be $\Delta[i]$). After n such instances for q , the re-ranking procedure is started. The average displacement (AD_q), for each link in the array, will be calculated as $AD_q[i] = \Delta[i] / n$.

Next, the original rank of each link is added to the corresponding displacement, AD_q . This is to increase the weight of the original ordering (r) to downgrade the possible noise for small n . The final step in the re-ranking procedure is to re-rank the links for q , in the increasing order of their $(AD_q + r)$ values. From here onwards, for query q the newly ranked list will be shown to the users. By following this algorithm, the outcome of each user query is arranged in such a way that the results expected by the user moves toward the top of the ranking and the relative order for the un-clicked links is retained as per the ranking function. In the coming sub-section, for illustration purposes, the presented procedure is used to rearrange the results of a query with feedback from four users.

4. Implementation and Verification

As was mentioned before, instead of developing a search engine (SE) from scratch, we have come up with an architecture that utilizes an already existing SE. A web-based interface is developed that takes the results from an existing SE and projects them to the user while recording ClickThrough data and executing the proposed algorithm. The server manages an intermediate database, in which it stores the ClickThrough data and uses them to rearrange the results for the future query, using the proposed algorithm. The architecture for the implementation of the proposed model was shown earlier in Figure 1.

4.1 Implementation

The major component of the implementation is a proxy server that serves as an interface between users and Google. In order to decide whether a query should be directly answered by the search engine or intercepted by the proposed algorithm, a simple mechanism is used: if the terms were queried for less than ' n ' times (where n is a threshold value set by the administrator), the proxy server utilizes the external SE's resources and the server stores the results in the intermediate database; otherwise the server looks for results that are processed and available locally in the database. Each time the results are sent back to the user, the interface keeps track of the users' activities regarding their responses. It is during this interaction with the displayed results, that the ClickThrough data is recorded and at the end of a session, the recorded data is sent back to the proxy server, which executes our algorithm. The technologies used for the implementation are JSP, JavaBeans, JavaScript, MySQL 4.0 database, MySQL-java-connector and Google API. Tomcat server 5.5 and NetBeans were used for main application development.

4.2 Processing of the ClickThrough Data

After the successful implementation of the proposed model and the proxy server, for evaluation and verification purposes, data was collected from several interacting users. Concurrently, information was gathered regarding the keywords and the links returned by Google in response to the queries. The ClickThrough data was then processed to calculate the displacement in the position of links for each user and recorded in the intermediate database.

In this subsection, a simple example is given (Table 1). Suppose for query q , the original ranking from F is given by r , where $r: (l_1, l_2, l_3, l_4, l_5, l_6)$, and the ranking which might have best suited each of the users is given by $r_i (1 \leq i \leq 4)$ as depicted in Table 1. For each r_i , the algorithm calculates the displacements for the links, when compared to the original r , and then form Δ by adding these displacements together. Then for each link, the average displacement (represented by AD_q) is calculated and added to its original position in $r (AD + r)$. Finally, the links are re-arranged based on the increasing order of $(AD + r)$ to get the order $r^*: (l_1, l_3, l_2, l_4, l_5, l_6)$. A more illustrative example with more links is shown in Table 2.

Table 1. First Example of ranking rearrangement

r:	Displacement for the links					
	I1	I2	I3	I4	I5	I6
	Init:	0	0	0	0	0
1) r1:	l_1	l_3	l_5	l_2	l_4	l_6
2) r2:	l_3	l_4	l_1	l_2	l_5	l_6
3) r3:	l_1	l_3	l_2	l_4	l_5	l_6
4) r4:	l_1	l_2	l_4	l_3	l_5	l_6
	Δ	2	5	-3	-2	-2
	ADq $\div 5$	0.4	1	-0.6	-0.4	-0.4
	r	1	2	3	4	5
	r + AD	1.4	3	2.4	3.6	4.6
	r*	l_1	l_3	l_2	l_4	l_5

Table 2. 2nd Example of ranking rearrangement

r	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
	1	2	3	4	5	6	7	8	9	10
r1	1	3	4	2	5	8	6	7	9	10
r2	2	3	4	1	5	6	8	7	9	10
r3	1	3	5	2	4	6	7	8	9	10
r4	1	4	2	3	5	6	7	8	9	10
r5	1	2	3	4	5	6	7	8	9	10
r6	2	1	3	5	4	6	7	8	9	10
r7	1	2	4	3	5	6	7	9	10	8
r8	1	2	6	3	4	5	7	8	9	10
r9	1	3	2	4	5	6	7	8	9	0
r10	1	5	2	3	4	6	7	8	9	10
Displacement										
	0	2	-1	-1	0	1	1	-2	0	0
	3	-1	-1	-1	0	0	1	-1	0	0
	0	2	-1	1	-2	0	0	0	0	0
	0	1	1	-2	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
	1	-1	0	1	-1	0	0	0	0	0
	0	0	1	-1	0	0	0	2	-1	-1
	0	0	1	1	1	-3	0	0	0	0
	0	1	-1	0	0	0	0	0	0	0
	0	1	1	1	-3	0	0	0	0	0
Average Displacement										
	0.4	0.5	0	-0.1	-0.5	-0.2	0.2	-0.1	-0.1	-0.1
Avg Disp + original rank										
	1.4	2.5	3	3.9	4.5	5.8	7.2	7.9	8.9	9.9
Reordered links r*										
	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10

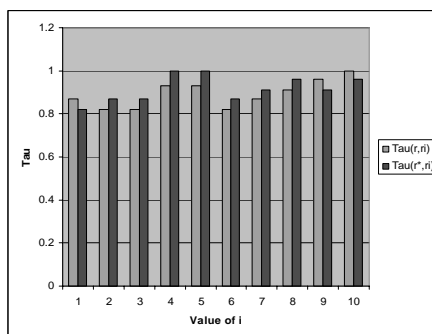


Figure 2. Comparison of Tau's from Table 3

4.3 Verification

τ can be calculated for the sets of displacements, using r and r^* separately. If our model is affective, then the $\tau(r^*, r_i)$, Tau value calculated with r^* and r_i , should be larger than the Tau calculated with r , $\tau(r, r_i)$. Saying that, it is not always possible that $\tau(r^*, r_i) > \tau(r, r_i)$ is valid for every i ($1 \leq i \leq n$) for obvious reasons such as noise and less reliable ClickThrough data collected from some users. Therefore, if $\tau(r^*, r_i) > \tau(r, r_i)$ materializes for the majority of r_i values then the objective of the presented approach is verified.

Consequently, Kendall's Tau was applied to the data in Tables 2 and 3, collected by our proxy server, in order to verify the advantage of the arraignments provided by the presented model. For Table 2, Tau values were calculated for the original ranking (r) and the reordered ranking (r^*) against r_1, r_2, \dots, r_{10} and the outcome was compared in Table 3.

Table 3. Values of Tau for Table 2

Value of i	1	2	3	4	5	6	7	8	9	10
Tau(r,ri)	0.87	0.82	0.82	0.93	0.93	0.82	0.87	0.91	0.96	1
Tau(r*,ri)	0.82	0.87	0.87	1	1	0.87	0.91	0.96	0.91	0.96

As it is illustrated, $\tau(r^*, r_i)$ is larger than $\tau(r, r_i)$ for seven cases out of ten, which indicates better ordering by the proposed system in 70% of the cases. Figure 2 illustrates the difference between the Tau values for each r_i reflected in Table 3.

Fifteen such data sets (as in Tables 1 and 2) were collected for fifteen different sets of keywords and over 20 users interacted with the proxy server for each set of keyword. In an average of 73% of the cases the proposed system provided better ordering compared to the original ordering of Google which is a considerable superiority. This outcome is not surprising. This is obviously for the reason that in the calculation of r^* , the users' choices were collected and considered in the rearrangement process.

5. Conclusion

In this paper, a new approach for enhancing the ordering of the query results, applicable to any search engine as an add-on module, is presented. The developed system is general enough to be applied to any search engine to improve the quality of its results. The initial experimental results are very encouraging. Since this system forms the query results based on the users' implicit feedback (ClickThrough data), it is bound to get better results as its usage increases. The model is designed to give more weight to the choices made by a group of people with the largest number of users, for a given set of keywords. The main advantage of this work over other similar models is our approach for combining the collected ClickThrough data, from multiple users, to rearrange the query results, that provides a better ordering. Moreover, in this class of ordering algorithms, there is some noise associated with the feedbacks that is amplified if the number of users is small; the model becomes stable as the number of interacting users increases.

REFERENCES

- [1] Berenens-Lee, T., Cailliau, R., gruff, J., and Pollermann B., "World Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, 1(2), 1992
- [2] J. Kleinberg, "Authoritative sources in a hyperlinked environment", In *Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms*, ACM Press, New York, 1998, pp 668-677
- [3] L. Page, S. Brin. "The PageRank citation ranking: Bringing order to the web", Technical report, Stanford Digital Library Technologies, 1998
- [4] "Google Technology," July 2005, <http://www.google.com/technology/>
- [5] Budi Yuwono, Dik L. Lee. "Search and ranking Algorithms for locating Resources on the World Wide Web", *Data Engineering, Proceedings of the Twelfth International Conference*, March 1996, pp:164 - 171
- [6] A Arasu, J Cho, H Garcia-Molina, A Paepcke, "Searching the Web" - *ACM Transactions on Internet Technology*, 2001 - portal.acm.org
- [7] T. Joachims, "Optimizing Search Engines using Clickthrough Data", July 2005, <http://www.joachims.org>
- [8] T. Joachims. "Unbiased Evaluation of retrieval quality using clickthrough data". Technical report, Cornell University, Department of Computer Science, 2002. <http://www.joachims.org>
- [9] Kendall. "Rank Correlation Methods". Hafner, 1955, pp 86-87