2010

A3CRank: an adaptive ranking method based on connectivity, content and click-through data

Ali M. Zareh Bidoki
University of Tehran

Pedram Ghodsnia
University of Tehran

Nasser Yazdani
University of Tehran

Farhad Oroumchian
University of Wollongong in Dubai, farhado@uow.edu.au

Publication Details

Zareh Bidoki, A. M., Ghodsnia, P., Yazdani, N, & Oroumchian, F. A3CRank: An adaptive ranking method based on connectivity, content and click-through data, Information Processing & Management, 46(2), 2010, 159-169

A3CRank: An adaptive ranking method based on connectivity, content and click-through data

Ali Mohammad Zareh Bidoki* zare_b@ece.ut.ac.ir
Pedram Ghodsnia* pedram@pedram.ir
Nasser Yazdani* yazdani@ut.ac.ir
Farhad Oroumchian** FarhadOroumchian@uowdubai.ac.ae

*ECE Department, University of Tehran, Tehran, Iran
** University of Wollongong in Dubai, Dubai, UAE

Abstract

Due to the proliferation and abundance of information on the web, ranking algorithms play an important role in web search. Currently, there are some ranking algorithms based on content and connectivity such as PageRank and BM25. Unfortunately, these algorithms have low precision and are not always satisfying for users. In this paper, we propose an adaptive method based on the content, connectivity and click-through data triple, called A3CRank. The aggregation idea of meta search engines has been used to aggregate ranking algorithms such as PageRank, BM25, TF-IDF. We have used reinforcement learning to incorporate user behavior and find a measure of user satisfaction for each ranking algorithm. Furthermore, OWA, an aggregation operator is used for merging the results of the various ranking algorithms. A3CRank adapts itself with user needs and makes use of user clicks to aggregate the results of ranking algorithms. A3Crank is designed to overcome some of the shortcomings of existing ranking algorithms by combining them together and producing an overall better ranking criterion. Experimental results indicate that A3CRank outperforms all other single ranking algorithms in P@n and NDCG measures. We have used 130 queries on University of California at Berkeley's web to train and evaluate our method.

Keywords: Web ranking, PageRank, BM25, TF-IDF, Aggregation, Reinforcement learning, OWA;

1. Introduction

Finding high quality web pages is one of the most important challenging issues for any web search engine. Ideally, the quality of pages is defined based on user preferences. Therefore, the problem of ranking is to sort web pages based on user requests or preferences. Definitely, to make the web more interesting and productive, we need a good and efficient ranking algorithm to present more appropriate results for users.

Usually, there are thousands or even millions of relevant pages for each query. Nevertheless, users typically consider only the top 10 or 20 results. Therefore, we have to focus on the most valuable and appealing pages. To do this, a better ranking criterion is required and a more efficient mechanism has to be used. This will enable the search engine to present the best related pages to the user in response to her queries. However, current ranking algorithms have low precision in average and are not adaptive to user needs. Obviously, we have to devise a solution to achieve a ranking algorithm with higher effectiveness that is also adaptive to page content and user preferences.

There are currently two major categories of ranking algorithms based on content (classical IR) and connectivity (the web graph).

In classical Information Retrieval (Baeza-Yates & Ribeiro-Neto, 1999), the system tries to find documents corresponding to the user query. IR Algorithms usually work based on matching words in documents. In other words, for each query the documents with the more similar content to the query will be selected as the more relevant ones. Examples of the content based ranking algorithms are TF-IDF (Salton & Buckley, 1988), BM25 (Robertson, Walker & HancockBeaulieu, 1995), etc. These algorithms are suitable for well formed and structured environments such as digital libraries and collections of scientific articles. In these environments queries are long and well specified and the vocabulary is small and relatively well understood.

However, the web consists of a large number of unstructured documents linked together, creating a massive graph. Furthermore, queries are generally short (2.4 terms in average (Zhang & Moffat, 2006)) and vocabulary is huge. This poses new challenges to IR. In addition, since the contents of the web are published in a distributed manner, this content is often inconsistent and includes a lot of misinformation. Therefore, application of classical IR methods to web content will result in problems such as low precision and recall, as well as the rank spamming problem (Henzinger, Motwani & Silverstein, 2002).

To remedy these issues, new connectivity-based algorithms have been proposed that use links between web pages in addition to content relevancy. Previous studies indicate that algorithms using hyperlinks for ranking yield satisfactory results (Henzinger, 2001). Their main strength comes from using the content of other pages to rank current pages. In other words, links carry information which can be used to evaluate the importance of pages and the relevancy of pages to the user query. Instances of connectivity based ranking algorithms are PageRank (Page, Motwani, & Winograd, 1998), HITS (Kleinberg, 1999) and DistanceRank (Zareh & Yazdani, 2007).

Although these algorithms are appropriate in some situations, on average their precision is low compared to content based algorithms (Najork, Zaragoza & Taylor, 2007). Furthermore, they suffer from shortcomings like the "rich-get-richer' problem (Cho, Roy, & Adams, 2005) that causes young high quality pages to take a long time to become popular. In other words, popular pages are ranked higher and have a higher chance to be browsed by users while young pages are likely to be neglected regardless of their quality. In web information retrieval, the user plays the most important role in the system and the basic objective is to satisfy him by a good ranking. However, in the above ranking algorithms, there is not any position for the user; directly or indirectly. Thus, there seems to be room for better algorithms that take the role of the user into account.

Hitherto we have identified three factors that can be used to produce high-quality rankings of web content, namely content, connectivity, and user behavior. In this paper, we propose an adaptive ranking algorithm based on this triple. For this purpose we use click-through data to implicitly take user behavior into consideration. We call this algorithm A3CRank in which the “A” stands for Adaptive and the 3 “C”s are Content, Connectivity and Click-through data. In this algorithm, we use click-through data to combine the results of both the content-based and the connectivity-based algorithms. For this combination, we have used some content-based methods such as TF-IDF , BM25 and DFR_BM25 (He & Ounis, 2005) in addition to PageRank as a connectivity-based ranking algorithm.

We use ideas from the concept of meta search engines with the difference that meta search engines merge the results of other search engines while here we merge the results of some ranking algorithms. Roughly speaking, we are going to aggregate some ranking algorithms using user click-through data, to achieve a better ranking algorithm and bring the higher quality pages to the top of the result list. A major property of our method is its adaptability. Depend on the user need and context, the method adapts itself with the environment to present an appropriate ranking for the user’s satisfaction.

We use a goodness factor for each ranking algorithm which shows the satisfaction degree of an average user for each algorithm. The goodness factor is computed via an iterative process, using user clicks and reinforcement learning (Sutton & Barto, 1998). Furthermore, we have used the OWA operator (Yager, 1988) to merge the results of the various ranking algorithms.

We have used 130 queries on the University of California at Berkeley’s web to evaluate our algorithm. The queries have been issued by 10 computer science students to the system for evaluation. Our algorithm outperforms all the other single ranking algorithms like PageRank, BM25, TF-IDF, etc in the standard criterions such as P@n and NDCG.

The next section discusses our solution, A3CRank. Experimental analysis and comparison with some of the well-known algorithms come in section 3. Section 4 reviews related work. Finally, our conclusion and future work agenda are presented in section 5.

2. A3CRank

To overcome some challenges in web search such as low coverage and low precision/recall rankings, Meta Search Engines (MSE) (Meng, Yu, & Liu, 2002) are sometimes used. In a MSE, the user query is sent to a number of search engines and then, the results from all of these engines are merged together and presented to the user. MSEs perform two important tasks including database (search engine) selection and document selection. In the former, depending on user goals, some search engines are selected and in the latter, some documents are selected for merging. Therefore one of the important duties of MSEs is the merging of results. Unfortunately, in a MSE, the various search engines are considered as black boxes and details about their internal features are not available.

In this paper, we apply the merging idea of MSEs to ranking. That is, we have a number of ranking algorithms instead of a number of search engine's results to merge. We combine the results from ranking algorithms such as PageRank (Page, Motwani, & Winograd, 1998), HITS (Kleinberg, 1999), BM25 (Robertson & Walker, 1994), TF-IDF (Salton & Buckley, 1988), etc. Figure 1 shows an overview of our idea.

Contrary to the situation with MSEs, here all detailed features of sources are available. As mentioned, the most important thing in a MSE is merging the results. In our method, the behavior of the user is utilized to increase the quality of the merging process. In other words, the algorithm adapts itself with user click-through-data through time. Thus, the system will satisfy users' needs to find high quality pages more conveniently. As mentioned earlier, we call this algorithm A3CRank, because it is an Adaptive algorithm based on the Content, Connectivity and Click-through data triplet.

Naturally, each ranking algorithm has its pros and cons. By combining different ranking algorithms, there is a chance to overcome their shortcomings and achieve a better algorithm. For example, PageRank which is a connectivity-based algorithm is based on page popularity which finds old high quality pages better. Although its sensitivity to rank spamming is low, it is susceptible to the "rich-get-richer" (Cho, Roy, & Adams, 2005) problem. While content sensitive algorithms (in classical IR) such as TF-IDF and BM25 are not sensitive to the "rich-get-richer" problem, they are prone to rank spamming. Thus with an appropriate aggregation, the weaknesses of each algorithm can be mitigated. We will show some of the benefits of combining algorithms in the experimental results presented in the next section.

Roughly speaking, we needed to find out how to aggregate the results by using user behaviour data to find the real quality of the pages and also to satisfy the user needs. Our problem is similar to a reinforcement learning problem (Sutton & Barto, 1998*)*. In the reinforcement learning problem, the learning is done from interaction. "The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment." The agent interacts with the environment by selecting some actions and the environment will present the agent with rewards dependent on the chosen action. The agent and environment interact together at a sequence of discrete time steps $t = 0,1,2,\ldots$. At each time step, the agent receives some representation of the environment's state $s_t \in S$, where S is the set of all possible states, and on that basis selects an action a_t. The agent is going to maximize received rewards from the environment.

In our method we consider the search engine as the agent and the users as the environment. Figure 2 shows the relation between the search engine and users. The search engine, the agent in response of user query (q), sends a ranked list of results as an action. Dependent on the quality of these results, the user will click on items in the ranked list as the reward. Furthermore, in each time step t the ranked list and the query together comprise the state s_t. Naturally, the search engine agent aims to maximize the number of clicks on its returned results. In other words, the agent must prepare more appropriate results for the user, in order to receive better rewards.

Our algorithm, A3CRank is composed of four stages:

1. Computing the goodness factor for each algorithm.
2. Utilizing the goodness factor of each algorithm to compute the weight of each resulting page from that algorithm.
3. Computing the OWA vector for aggregation.
4. Computing the final weight of each result using the OWA vector.

These stages will iterate until the convergence point is reached. That is, until the system parameters such as goodness factors and weights become stable. In the following, we explain these stages in detail.

In the first stage, we use a factor called the goodness factor (*gf*) which is computed for every algorithm by reinforcement learning (Sutton & Barto, 1998*)* using user click data. The goodness factor shows the degree of

suitability of each algorithm for users in average. This factor helps us find the real quality of pages in each algorithm. In other words, the algorithm with higher *gf* will return higher quality pages. We will use this factor in the next stage to compute the weight of each resulting page returned by each algorithm. The *gf* is computed as shown in equation 1 below, where α is learning rate of the system (equation 2) and the factor γ is the quality of the clicked pages and comes from equation 3.

The value of the learning rate α comes from equation 2 where *itr* shows time or iteration number and β is a static value to control the regularity of the learning rate. Experimentally, we found that if the learning rate α is properly adjusted, the system will converge and reach the stable state faster. In the initial state of the algorithm, the "*gf*"s of the algorithms are not known, so initially, we set α to one and, then, decrease it exponentially to zero. Furthermore, in the first iteration we set all goodness factors to 1/m equally where m is the number of the underlying ranking algorithms.

$$gf_i = (1-\alpha) * gf_i + \alpha * \gamma \qquad (1)$$

$$\alpha = e^{-\beta * \text{itr}} \qquad (2)$$

$$\gamma = \sum_{j=1}^{T} \frac{2^{r_j} - 1}{\log(1+j)} \ , \ r_j = \frac{1}{t_j} \qquad (3)$$

In equation 3 which is similar to the NDCG measure (Jarvelin & Kekalainen, 2000), T is the number of clicked pages and t_j depicts the order of clicking on a page with rank j (j^{th} page). Because we don't have r_j which is the relevancy of page j in the NDCG measure, we replace it with the order of clicks by the user. That is, we interpret that the pages receiving earlier clicks have higher relevancy. Naturally, we can use other factors such as P@n for optimization. After the learning process, the selected criteria will be optimized.

The second stage is to assign a weight to each result of every algorithm. The weight of resulting page d for algorithm i is dependent on its *gf* and the page's rank and is computed as depicted in equation 4, where n is the number of results for each query returned by algorithm i and R(d) shows the rank (order) of d in the results. Thus n, may be different for every algorithm and it depends on both the query and algorithm. Furthermore, R_i is the set of resulting pages from a query returned by algorithm i.

$$w_{id} = \begin{cases} gf_i * (1 - \dfrac{R_i(d) - 1}{n}) & if\ d \in R_i \\ 0 & else \end{cases} \qquad (4)$$

Therefore, we will have a matrix of weights containing the weight of each resulting page returned by each algorithm. In this matrix the rows represent the various algorithms and the columns represent the returned pages. Naturally, if resulting page d is not included in algorithm i then w_{id} is set to zero. Then we trace the matrix column by column to store a vector for each page. For example $V_d=\{w_{1d}, w_{2d}, \ldots, w_{md}\}$ is the vector of resulting page d. Afterwards, we sort all of the vectors in descending order.

The third stage is computing the OWA (Yager, 1988*)* weight vector for each resulting page vector. Currently, OWA is one of the best aggregation operators. This operator maps a vector of size n to a single value. For example if $A=[a_1 \ldots a_n]$ is our page vector and W=[OWA(1) , OWA(2) , … OWA(n)] is the OWA weight vector in which $\sum_j OWA_j = 1$, the result of aggregation is $f(a_1,...,a_n) = \sum_{j=1}^{n} OWA_j b_j$, where b_j is the j^{th} largest element in

vector A. As we see, the main property of the OWA operator is the reordering of A. So that each weight OWA_i is associated with i^{th} largest element in vector A instead of a_i.

We use the Optimistic Exponential OWA operator to find the weights of each vector as the following.

$$OWA(1) = \alpha$$

$$OWA(2) = \alpha(1-\alpha)$$

$$OWA(3) = \alpha(1-\alpha)^2$$

...

$$OWA(n-1) = \alpha(1-\alpha)^{n-1}$$

$$OWA(n) = (1-\alpha)^n$$

Where parameter α belongs to the unit interval $0 \le \alpha \le 1$. Now, we have an n dimensional OWA vector in that OWA(i) is associated with the i^{th} position in each sorted page vector. Experimentally, we have found that $\alpha = 0.3$ is suitable for the results aggregation.

The final stage is the weight computation of each resulting page as in equation 5, where m depicts the number of ranking algorithms that have been used.

$$w_d = \sum_{i=1}^{m} w_{id} OWA(i) \qquad (5)$$

At this point, the results are presented to the user in order of descending w_d.

The above four stages will iterate until we reach a convergence point and have the final *gf* for each algorithm. After that the learned goodness factors will be used for the combination of ranking algorithms. Naturally, gf is not static forever and web content in constantly prone to change. In response to these changes (detected during the crawling and indexing processes) and computation of the new rankings, gf will be computed again. But except in the first run, the initial *gf* for each algorithm is not reset to 1/m and the latest calculated value is employed.

This combined algorithm has many advantages such as:

- It is scalable, allowing for the addition of any new algorithm easily.
- It is adaptive with users' behavior and preferences.
- Straightforwardly, we can add user personalization with assigning *gf* to each user and each algorithm, separately.
- Instead of using coarse-grained features (ranking algorithms), we can use fine-grained features such as TF, IDF, in-link, etc. Obviously, each ranking algorithm is composed of these small features.
- The algorithm is designed to help overcome shortcomings like the "rich-get-richer" and rank spamming problems. We will try to assess this aim with more experimental analysis in the future.

3. Experimental Results

Because our algorithm is based on user clicking behavior and reinforcement learning, its evaluation is hard and complicated. We used University of California at Berkeley's web site with five million web pages to evaluate our algorithm. About 130 standard queries in two categories have been used. Some of these queries have been extracted from the AOL data set (Pass, Chowdhury, & Torgeson, 2006) of queries that users have used in the past to reach Berkeley's web. We asked 10 computer science students to enter these queries into the system. The selected well known algorithms for aggregation are PageRank, TF-IDF, BM25 and DFR_BM25. We used the Terrier information retrieval platform from the University of Glasgow (Terrier, 2007) to compute the value of content based ranking

algorithms. We found that the results of TF-IDF and BM25 are similar in this data set, so we did not use BM25 for combination.

For comparison, we have used two metrics "Precision at position n" (P@n) and "Normalized Discount Cumulative Gain" (NDCG) (Jarvelin & Kekalainen, 2000). Precision at *n* measures the relevancy of the top *n* results of the ranking list with respect to a given query (equation 6).

$$P@n = \frac{\#\text{of relevant docs in top n results}}{\text{n}} \qquad (6)$$

Since P@n can only handle cases with binary judgment "relevant" or "irrelevant", for better evaluation we also used NDCG which handles multiple levels of relevance. Equation 7 shows how NDCG is computed. r(j) is the rating (0=detrimental, 1=bad, 2=fair, 3=good, 4=excellent, and 5=definitive) at rank j. To compute NDCG, about 30 results from each algorithm, related to 130 queries, were judged in these 6 levels by users.

$$NDCG(n) = \sum_{j=1}^{n} \frac{2^{r_j} - 1}{\log(1+j)} \qquad (7)$$

For the evaluation of our algorithm we conducted two stages: training the system and testing the system. About 60 queries were used in the training phase and the rest were used for the testing phase. In the training phase, users worked with the system by issuing queries and clicking on the results. In the training stage, parameters like the goodness factor of algorithms were learned by the system. Figure 3 demonstrates the steps of the training stage:

1. Set the goodness factors for each algorithm equally to 1/m.
2. The system receives a query from the user.
3. Results of all ranking algorithms are sorted and merged by equation 5 (the final weight of each resulting page is computed).
4. The results are presented to the user, sorted by their weights.
5. Gather user click-through data for each query.
6. Compute the goodness factor of each algorithm using equation 1.
7. If any other query exists go to step 2.

The above steps iterate until all goodness factors are discovered. Now we start the test process using the remaining 70 queries. As noted earlier, two measures (NDCG and P@n) are used for evaluation. Naturally, we compute them for each query and then take the average in each dimension (n) for all queries.

Figure 4 and 5 show comparisons of the A3CRank algorithm with other algorithms in the P@n and NDCG measures respectively. As the figure shows, our learning algorithm outperforms the others. A3CRank achieves a 46% increase in P@n and a 36% increase in NDCG@n compared to DFR_BM25 which is the best one of the others. This achievement resides in the combination of both content-based and connectivity-based algorithms using user behavior data.

After convergence, we have 0.09, 0.38 and 0.53 values for the goodness factors of PageRank, TF-IDF and DFR_BM25 respectively. Also their averages during the learning process are 0.12, .35 and .53 respectively. Figure 6 shows the convergence track of the system to achieve final goodness factors. Thus, the BM25 algorithm has the highest and PageRank has the lowest value. Similar results have also been found in the literature for precision and NDCG values (Najork, Zaragoza & Taylor, 2007). Now we can be confident that by aggregation of some ranking algorithms controlled by user behavior we achieve more satisfactory results.

To show that the learning process used by A3CRank will find the high quality pages in earlier positions on average for different queries, we show the difference between the result quality of each query (equation 8) in figure 7 before and after the learning process. In this equation n is the number of top results showed to users and r_j

depicts the relevancy degree of the page with rank j (judged by the user). To compute the rank quality of query q, RQ_q, we sum the multiplication of relevancy degrees of items of the result list in their position values (1- (j-1)/n). For example for the first result of the query q, having the highest rank, its quality value is $r_1 * 1$ and for the second its quality value is $r_2 * \frac{n-1}{n}$, etc. We can observe in figure 7 that for most of the queries, the quality difference of results is positive. It means that our algorithm, averagely, will find the high quality pages of different queries in earlier positions of the list after the learning process.

$$RQ_q = \sum_{j=1}^{n} r_j (1 - \frac{j-1}{n}) \qquad (8)$$

To evaluate the quality of results, we compared 10 results from A3CRank and Google's ranking (results from Google search engine on 30 September 2007) for some queries in a subjective manner. We also forced Google engine to search only in the University of Berkeley's web site (.berkeley.edu). Tables 1 and 2 show the list of all titles and URLs of the top ten results for these two algorithms for **"**genetic drift" and "cross-language" queries respectively, ordered by each algorithm. Obviously, for the "cross-language" query both algorithms have similar results while for other query "genetic drift", because A3CRank is based on both content and connectivity, it produces superior results.

4. Background and Related Work

To the best of our knowledge, we have not seen any studies that use the combination of current ranking algorithms and user clicks for better ranking in search engines. Although, similar work has been done in the meta search engines area. For example in (Keyhanipour, Moshiri, & Kazemian, 2007), a solution for aggregation of results in meta search engines that uses click-through data and also the OWA operator for merging has been proposed. Also this method is adaptive and they have achieved interesting results through the aggregation of nine search engines.

In (Joachims, 2002) by utilizing the user click-through data and support vector machine (SVM) for training, a learning method for ranking has been proposed. It has been shown that this method effectively adapts with the retrieval function of a meta search engine to a particular group of users. After training, with a couple of hundred queries, it outperforms Google in terms of retrieval quality.

Similar to research presented here are the work done in (Shakery & Zhai, 2006; Qin, Liu, & Zhang, 2005). In these works, both content and connectivity (links) have been used without considering user clicks. They proposed a new general relevance propagation model for combining link and content. In their work instead of transmitting the rank of nodes in the web graph, relevancy between he query and the document (that node) will be propagated. It has been shown experimentally, that combining link and content generally results in better performance than using only content.

Also similar work has been done in (Richardson & Domingos, 2002) where they use a model called "intelligent surfer" for ranking. In this model instead of using a simple random surfer model used in PageRank, a probabilistic combination of link and content has been proposed. In their model, the weights of output links of nodes are not equal and are dependent on the user query. They have found that their algorithm outperforms the PageRank algorithm.

In (Xue, Zeng, & Chen, 2004) an iterative algorithm by utilizing the user click-through data to improve search performance has been proposed. This algorithm aims to find hidden relations and similarity between queries and documents to add these queries to documents' metadata (document extension). Experimentally, by using a large set of the MSN search log, they achieved significant improvements on search performance.

In (Agichtein, Brill, & Dumais, 2006) by using many click-through data features as user feedback in the ranking process both directly and indirectly, they found very interesting results. They used 3,000 user queries for evaluation and found a 31% increase in ranking quality in comparison to other ranking algorithms.

In the above methods that have used click-through data, user clicks are considered only as a feature for ranking, not for adaptability with the environment. While, in our method in addition to aggregation of some ranking algorithms by clicks, we have used the clicks to attain an adaptive ranking.

5. Conclusion and Future Work

Current ranking algorithms, whether content-based or connectivity-based, suffer from low precision and recall. Also they are not suitable for some situations and dependent on the context, they will work differently.

In this paper we have proposed a combined algorithm based on 3 "C"s: Content, Connectivity and Clicks. We have titled this algorithm A3CRank. A3CRank try to adapt itself with user needs using user click-through data. We have used the meta search engine idea of aggregation in our algorithm for ranking. While meta search engines aggregate the results from a number of search engines, our algorithm aggregates the results of a number of ranking algorithms. We have used a few ranking algorithms (both content and connectivity based) for combination such as TF-IDF, BM25 and PageRank**.**

For each ranking algorithm a factor called the goodness factor has been defined. It shows the degree of the satisfaction of the average user with that algorithm. Each goodness factor is computed using a reinforcement learning method according to the quality of clicked documents. Every clicked document's quality is computed by its rank and the order in which it was chosen for clicking.

Then, the results of all algorithms are merged by the OWA operator, which is one of the best known aggregation operators. The weights for resulting pages are computed using the goodness factor of each algorithm and the page's rank. Also, we have used the Optimistic Exponential OWA operator to finds the weights of the OWA vector.

We used University of California at Berkeley's Web and 130 related queries for evaluation of A3CRank. We have compared A3CRank with other single algorithms in P@n and NDCG measures and found interesting improvements.

The proposed algorithm has some features like scalability and adaptability. It is scalable in that we can add any new algorithm easily and also adaptive in that it adapts itself with user needs. Currently, we have selected coarse-grained features such as PageRank popularity and BM25 relevancy for merging. We plan to merge some fine-grained features like TF, IDF or in-degree that the above algorithms are composed from, using the mentioned method, as future work. Obviously, after the learning process and computation of goodness factors we will have a new ranking algorithm composed of some little features. Also the assessment of the susceptibility of our method to rank spamming (Henzinger, Motwani, & Silverstein, 2002) and "rich-get-richer" problems remains as future work. We also plan to add other ranking algorithms in the combination process such as HITS, DistanceRank, etc.

8. References

Agichtein, E., Brill, E., & Dumais, S.(2006). Improving Web Search Ranking by Incorporating User Behavior Information. In Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR).

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern Information Retrieval. ACM Press/ Addison-Wesley.

Cho, J., Roy, S., & E. Adams, R. (2005). Page Quality: In Search of an Unbiased Web Ranking. In Proceedings of ACM International Conference on Management of Data (SIGMOD).

He, B., & Ounis, I. (2005). A study of Dirichlet priors for term frequency normalisation. In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 465 – 471).

Henzinger, M. (2001). Hyperlink analysis for the web. IEEE Internet Computing, 5(1),45–50.

Henzinger, M., Motwani, R., & Silverstein, C. (2002). Challenges in web search engines, SIGIR Forum 36(2).

Jarvelin, K. , & Kekalainen, J. (2000). IR evaluation methods for retrieving highly relevant documents. In Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR).

Joachims, T. (2002). Optimizing Search Engine using Clickthrough Data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (pp. 132-142).

Keyhanipour, A. H., Moshiri, B., Kazemian, M., Piroozmand, M., & Lucas, C. (2007).Aggregation of Web Search Engines Based on Users' Preferences in Webfusion. Knowledge- Based Systems, 20(4), 321-328.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5),604–632.

Meng, W., Yu, C., & Liu, K.L. (2002). Building Efficient and Effective Metasearch Engines. *ACM Computing Surveys*, 34(1), 48–89.

Najork, M., Zaragoza, H., & Taylor, M. J. (2007). Hits on the web: how does it compare? In Proceedings of SIGIR'07 (pp. 471-478).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Computer Science Department, Stanford University.

Pass, G., Chowdhury, A., & Torgeson, C. (2006). A Picture of Search. In the First International Conference on Scalable Information Systems.

Qin, T., Liu, T.-Y., Zhang, X.-D., Chen, Z., & Ma, W.-Y. (2005). A study of relevance propagation for web search. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 408–415). ACM Press.

Richardson, M., & Domingos, P. (2002). The intelligent surfer: Probabilistic combination of link and content information in pagerank. In Advances in Neural Information Processing Systems.

Robertson, S. E., Walker, S., HancockBeaulieu, M. M., Gatford, M., & Payne, A. (1995). Okapi at TREC-4. In NIST Special Publication. The Fourth Text REtrieval Conference (TREC-4) (pp. 73—96).

Robertson, S., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In Proceedings of SIGIR (pp. 232-241)

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. Information Processing and Management, 24(5), 513–523.

Shakery, A., & Zhai, C. (2006). A probabilistic relevance propagation model for hypertext retrieval. In Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM) (pp. 550-558).

Sutton, R.S., & Barto, A.G. (1998).Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

Terrier (2007): http://ir.dcs.gla.ac.uk/terrier/.

Xue, G.-R., Zeng, H.-J., Chen, Z., Ma, W.-Y., Xi, W., Fan, W., & Yu, Y. (2004). Optimizing Web Search Using Web Click-through Data. CIKM (pp. 118-126).

Yager, R.R. (1988). On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Transactions on Systems, Man and Cybernetics,18(1).

Zareh Bidoki, A. M., & Yazdani, N. , DistanceRank: An intelligent ranking algorithm for web pages, Information Processing and Management (2007), doi:10.1016/j.ipm.2007.06.004.

Zhang, Y., & Moffat, A. (2006). Some Observations on User Search Behavior.11th Australasian Document Computing Symposium (pp. 1-8).

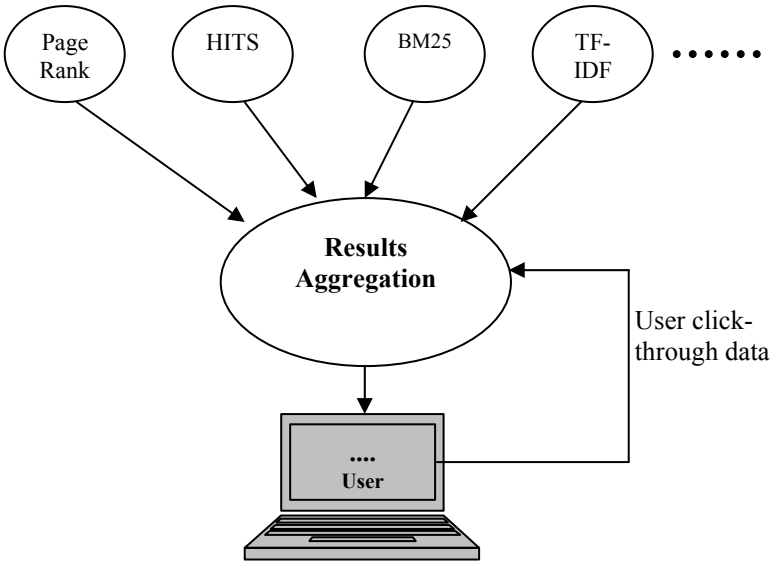

Figure 1: Using the aggregation idea in meta search to merge ranking algorithms.

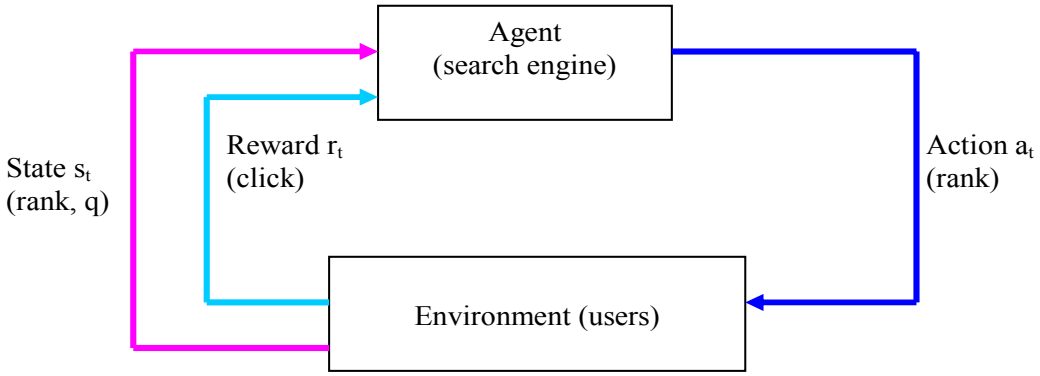

Figure 2: Relation between search engine and users as the agent and the environment.

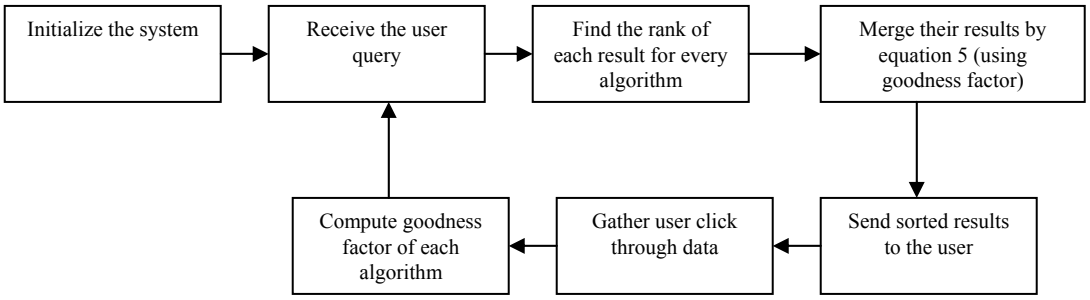

Figure 3: The steps of the system training stage.

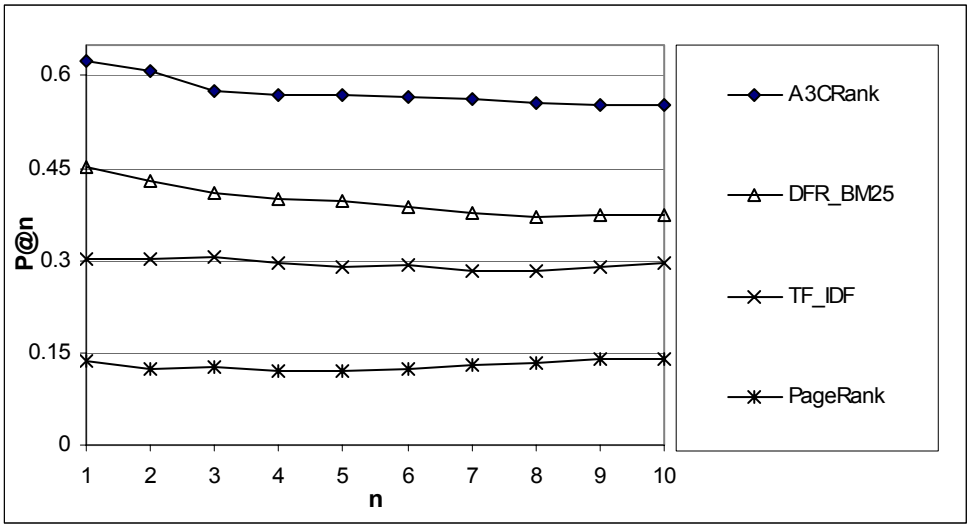

Figure 4: Comparison of A3Crank with BM25, TF-IDF and PageRank in the P@n measure.

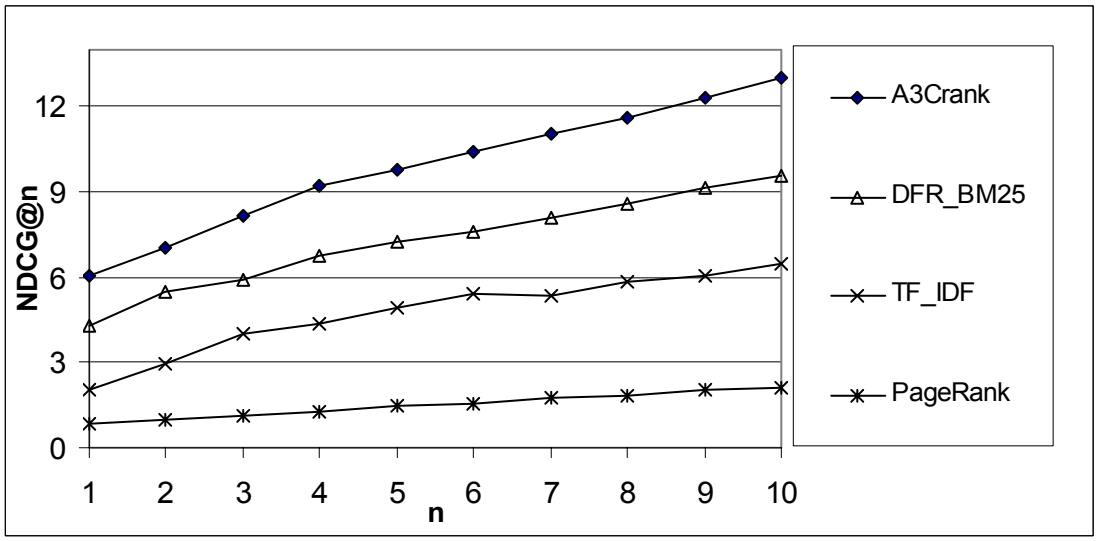

Figure 5: Comparison of A3CRank with BM25, TF-IDF and PageRank in the NDCG@n measure.

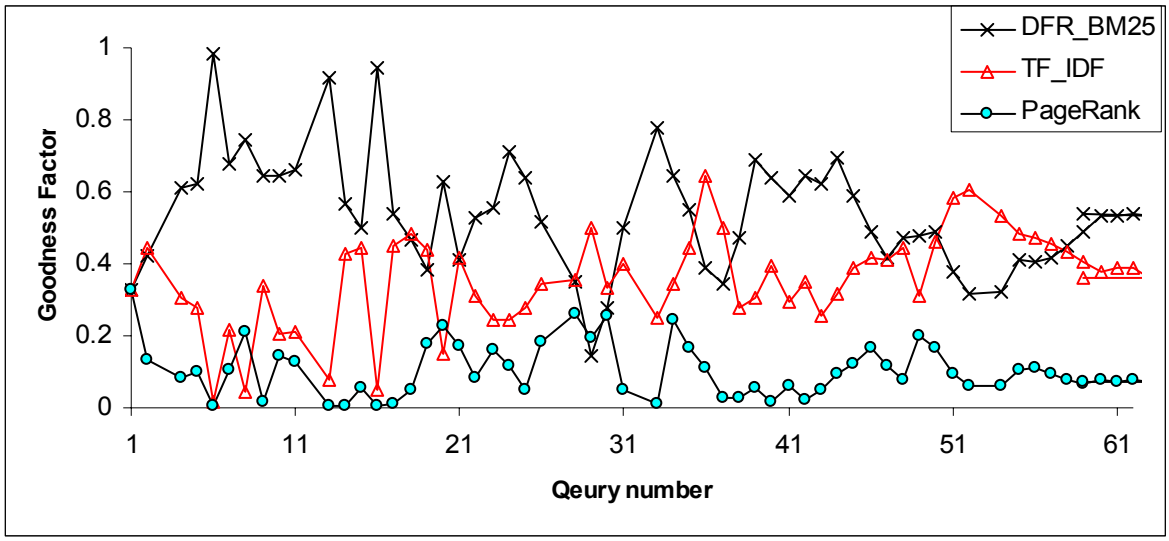

Figure 6: Convergence path of the system to find the goodness factors of ranking algorithms.

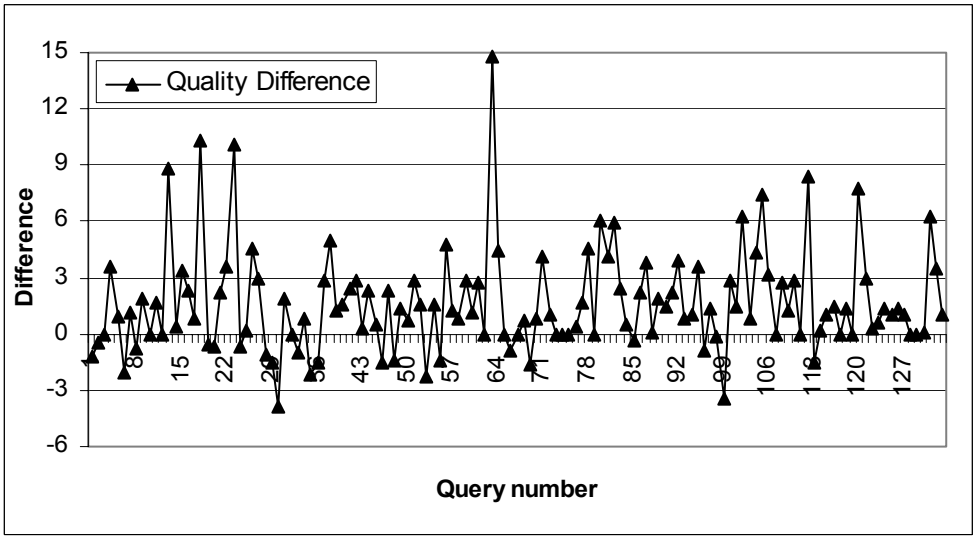

Figure 7: Difference between quality of each query result before and after learning (equation 8).

Table 1: List of all Titles and URLs from top ten results from Google and A3CRank algorithms (Query: cross-language).

Rank	A3CRank	Google
	Title &URL	**Title &URL**
1	**Metadata Research Program: Cross-Language Information Retrieval Home Page** http://metadata.sims.berkeley.edu/ResearchAreas/CrossLanguage.html	**Metadata Research Program: Cross-Language Information Retrieval Home Page** http://metadata.sims.berkeley.edu/ResearchAreas/CrossLanguage.html
2	**gey—papers** http://ucdata.berkeley.edu/personal/fred/my-papers/gey-papers.html	**Statistical tests of cross-language color naming** http://www.icsi.berkeley.edu/wcs/study.html
3	**Metadata Research Program Home Page** http://metadata.sims.berkeley.edu/papers/papers_bydate.html	**Fredric Gey UC DATA University of California Berkeley Information** ...

		http://ucdata.berkeley.edu/gey.html
4	**Metadata Research Program: Tides Project Abstract** http://metadata.sims.berkeley.edu/GrantSupported/tides_%20papers.html	**Metadata Research Program Home Page** http://metadata.sims.berkeley.edu/papers/papers_bydate.html
5	**Statistical tests of cross-language color naming** http://www.icsi.berkeley.edu/wcs/study.html	**Metadata Research Program: Tides Project Papers & Reports** http://metadata.sims.berkeley.edu/GrantSupported/tides_%20papers.html
6	**Macro Language** http://db.cs.berkeley.edu/postmodern/lecs/thomthom/sld008.htm	**Homepage Vivien Petras – Resume** http://people.ischool.berkeley.edu/~vivienp/resume.html
7	**Schema Concepts - Contents** http://www2.sims.berkeley.edu/academics/courses/is290-8/s04/lectures/6/toc.html	**gey—papers** http://ucdata.berkeley.edu/personal/fred/my-papers/gey-papers.html
8	**ActiveX** http://guir.cs.berkeley.edu/courseware/cs160/fall98/discussions/toolkitcomponent/sld022.htm	**296a3 Summary - Seminar Information Access Spring 2000** http://www2.sims.berkeley.edu/courses/is296a-3/s00/summary.html
9	**PPT Slide** http://trill.berkeley.edu/PhonLab/classes/ling110/PowerPoint/9sep/sld002.htm	**296a1 Summary - Seminar Information Access Fall 2000** http://www2.sims.berkeley.edu/courses/is296a-1/f00/summary.html
10	**Metadata Research Program: Seamless Project Home Page** URL: metadata.sims.berkeley.edu/GrantSupported/seamless.html	**296a1 Summaries - Seminar Information Access Fall 2002** http://www2.sims.berkeley.edu/courses/is296a-1/f02/summary.html

Table 2: List of all Titles and URLs from top ten results from Google and A3CRank algorithms (Query: genetic drift).

Rank	A3CRank	Google
	Title &URL	Title &URL
1	**Sampling error and evolution** http://evolution.berkeley.edu/evolibrary/article/side_0_0/samplingerror_01	**Metapopulation dynamics may facilitate cladogenesis** http://ib.berkeley.edu/courses/ib160/past_papers/vidigal-jones.html
2	**Genetic drift** http://evolution.berkeley.edu/evolibrary/article/side_0_0/evo_24	**Slide Summaries from the Biostatistics Workshop** http://allele5.biol.berkeley.edu/~diogo/ashi01/phylo2.html
3	**Effects of genetic drift** http://evolution.berkeley.edu/evolibrary/article/side_0_0/genesdrift_01	**NCSE Resource** http://www.cnr.berkeley.edu/~pts/Docs/Oklahoma_Disclaimer.html
4	**Mechanisms of microevolution** http://evolution.berkeley.edu/evolibrary/article/side_0_0/evo_39	**Questions on Evolution | Bio 1B Sections 105/106, Spring 2006** socrates.berkeley.edu/~akerr/bio1b/questions-evolution.html
5	**More on complex novelties (2 of 2)** http://evolution.berkeley.edu/evolibrary/article/side_0_0/complexnovelties_02	**2003 Bay Area Conservation Biology Symposium - Papers AG** http://nature.berkeley.edu/~alyons/conference_manager/sample_prog/papers_ag.html
6	**Peripatric speciation** http://evolution.berkeley.edu/evolibrary/article/side_0_0/speciationmodes_03	**ICSI | News: Scientists from ICSI, CAL-IT and Perlegen Sciences ...** http://www.icsi.berkeley.edu/news/2005/nb0502.html
7	**Variation and Selection** http://www.ucmp.berkeley.edu/education/dynamic/session3/sess3_variation2.htm	**[Gm-interest] provided** http://chess.eecs.berkeley.edu/gm/listinfo/gm-interest/2006-September/000397.html
8	**Variation within a Population** http://www.ucmp.berkeley.edu/education/dynamic/session3/sess3_variation1.htm	**Creating a Curriculum in Evolution** http://www.ucmp.berkeley.edu/education/events/padian1.html
9	**Modes of speciation** http://evolution.berkeley.edu/evolibrary/article/side_0_0/speciationmodes_01	**NCTE Resource Matrix, Curriculum** http://www.ucmp.berkeley.edu/ncte/resourcematrixcurr.html
10	**Natural selection** http://evolution.berkeley.edu/evolibrary/article/0_0_0/evo_25	**Publications** http://ib.berkeley.edu/labs/patton/publications.html