A Revised SimRank Approach for Query Expansion

Yunlong Ma, Hongfei Lin, and Song Jin

Information Retrieval Laboratory, School of Computer Science and Engineering,
Dalian University of Technology, Dalian 116023, China
kevinma@mail.dlut.edu.cn, hflin@dlut.edu.cn,
jinsong@mail.dlut.edu.cn

Abstract. Query expansion technologies based on pseudo-relevance documents have been proven to be effective in many information retrieval tasks. One problem with these methods is that some of the expansion terms extracted from feedback documents are irrelevant to the query, which may hurt the retrieval performance. In this paper, we proposed a normalized weight SimRank (NWS) approach for query expansion, with query logs collected by a practical search engine. Analyzing the relationship between queries and URLs, we create a query-click graph, and a term-relationship graph is constructed by several transformations. In order to reduce the computational complexity of NWS, strategies of pruning and radius limit were used to optimize the algorithm. Experimental results on two TREC test collections show that our approach can discover the qualified terms effectively and improve queries' accuracy.

Keywords: Search Engine, Query Expansion, Query Logs, SimRank.

1 Introduction

In general, most web search engines discover relevance documents for a query by term matching. Unfortunately, in most cases, the original query can't express user's information need accurately, and people often use different words to describe concepts in their queries from authors use to describe the same concepts in their documents. In order to help a user in such a case, the query expansion approach becomes very important which focus on generating new queries by adding words to the original query.

Query expansion is the approach of adding words to the original query to improve retrieval performance in information retrieval operations. It can discover some related property between words and original query automatically by analyzing a collection of documents or some resources. There are two main challenges in the query expansion process. One is where should the additional words be found. As an external resource, a query log is a rich information source [1], which including billions of submitted queries, viewed search results, and clicked URLs. Thus the huge amount of search engine log data offers excellent opportunities for additional words mining. The other one is what types of association are useful to help improve the queries quality. In the paper [2], we look into association and similar patterns at the level of terms through analyzing the relations of terms inside a query log in a real search engine, and use them as candidate terms.

P.-J. Cheng et al. (Eds.): AIRS 2010, LNCS 6458, pp. 564–575, 2010.

In this paper, we focus on query expansion base on the recent history of queries. We try to generate a historical query-click graph that records the clicks that were generated by URLs when a user inputs a query. The query-click graph is a weighted bi-partite graph [3], with queries on one side and URLs on the other. Then, the term-relationship graph which was obtained by several transformations from the query-click graph could reflect the direct connections of the terms in query logs. And our approach is based on the notion of SimRank [4], which can compute term similarity based on not only directly connections but also indirectly relationships in the term-relationship graph. However, in our case we need to extend SimRank to take into account the weights of the edges in the term-relationship graph, and we call the revised algorithm normalized weight SimRank (NWS). Term pairs with high similarity scores are with high confidence and then used for query expansion.

In the next section, we discuss the related work of query expansion and query logs analysis. Section 3 we describe how we construct the query-click graph and term-relationship graph from query logs. Section 4 describes the original SimRank algorithm and our normalized weight SimRank algorithm. Section 5 shows the candidate terms section and our query expansion strategy. Section 6 contains the experimental results based on the AOL query log, as well as a discussion of those results and the last section is conclusion.

2 Related Work

2.1 Query Expansion

In query expansion, the query is expanded using words or phrases with similar meaning to those in the query and the chances of matching words in relevant documents are therefore increased. One of the earliest studies of query expansion was carried out by [5] who clustered words based on co-occurrence in documents and used those clusters to expand the queries. The techniques that have been used recently can be described as being based on either global or local analysis of the documents in the corpus being searched.

The global analysis technique we describe here is one of the first techniques to produce consistent effectiveness improvements through automatic expansion. Other researchers have developed similar approaches, i.e., using LSI (Latent semantic indexing) or LDA (Latent Dirichlet allocation), and have also reported good results [6]. Among all the local analysis approaches, pseudo-relevance feedback (PRF) exploiting the retrieval result has been the most effective [7]. It has been implemented in different retrieval models: vector space model [8], probabilistic model [9], and so on. Recently, the PRF principle has also been implemented within the language modeling framework [10]. In these works, the top ranked documents were assumed to be relevant as a special case of relevance feedback.

Compare with PRF approaches a crucial question is the expansion terms determined in traditional ways from the pseudo-relevant documents are not all useful [1]. So, some studies focus on using an external resource for query expansion. Several external collection enrichment approaches have been proposed, such as search engine query logs [2], WordNet [12], Wikipedia [13] etc. Our work follows this strategy of a query expansion approach using query logs as a resource of query expansion terms.

2.2 Query Graphs

Baeza-Yates [14] identifies five different types of query graphs based on query logs. In all cases, the nodes are queries, a link is introduced between two nodes respectively if: (1) the queries contain the same word(s) (word graph), (2) the queries belong to the same session (session graph), (3) users clicked on the same URLs in the list of their results (URL cover graph), (4) there is a link between the two clicked URLs (URL link graph), (5) there are l common terms in the content of the two URLs (link graph). Baeza-Yates and Tiberi [15] study a weighted version of the cover graph. Their analysis provides information not only about how people query but also about how they behave after a query and the content distribution of what they look at. Moreover the authors study several characteristics of click graphs, i.e., Query-click graphs (also known as query-document graph) are introduced by [3][11], are bipartite graphs of queries and URLs, where a query and a URL are connected if a user clicked on the URL that was an answer for the query. This framework is used to infer semantic relations among queries and to detect multi-topical URLs, i.e., URLs that cover either several topics or a single very general topic. Query-click graphs are presenting more in detail in Section 3.

3 Basic Graph Model

3.1 Query-Click Graph

Query-click graph (also known as query-document graph) is introduced by [3]. It is a weighted bi-partite graph which the nodes in the one set are queries and the nodes in the other set are URLs which appears in query logs. An edge appears between a query q and a URL u if the user that issued the query, clicked on the URL in the list of results. Let $G_{QC} = \{V_{QC}, E_{QC}\}$ be a bi-partite graph such that $V_{QC} = \{v_q, v_u\} = Q \cap U$ and $E_{QC} = \{e_{qu}\} \subseteq Q \times U$, where Q a set of all queries in query logs and U a set of URLs clicked for those queries, v_q is a node of query and v_u is a node of URL in the graph. Let $\omega(e_{qu})$ be a weighting function for an edge $e_{qu} \in E_{QU}$, for this instance it is the number of clicks there are for a query q on a URL u.

Specifically, the query-click graph constructed by the following four steps:

1. Remove all noisy records and stopwords from the query logs (details in Section 6.1).
2. Add a new query node to the query-click graph for every unique query in the query logs.
3. In the same way, add a new URL node to the graph for every unique URL in the query logs.
4. Make a directly connection edge form every query-URL node, if they appear in the same record, and the weight of edge is assigned 1. If the edge already exists, add 1 to the weight.

3.2 Term-Relationship Graph

A term-relationship graph is a weighted and directed graph. We model terms and relationships as a graph $G_T = \{V_T, E_T\}$ where nodes in $V_T = \{v_t\}$ represent terms inside query logs and edges in $E_T = \{e_t\}$ represent relationships between terms. A directed edge which appears between two term nodes, represent a kind of direct relationship, if the user input them in different queries, and clicked the same URL in their list of results. The weight of edge $\omega(e_{ij})$ is the number of co-occurrence represent associated degree between two terms i and j.

The term-relationship graph can be obtained by two transformations from the query-click graph.

3.2.1 Query Nodes Substitution

Although the query-click graph can reflect all queries, URLs and the connections between them completely, then we can discover knowledge at the level of queries. But in this paper, we propose to mine query logs for query expansion at the level of terms through analyzing the relations inside queries. Thus we must substitute term nodes for query nodes by following four steps:

1. For every query node v_q in the query-click graph, replace it with all term nodes inside the query of v_q.
2. If the node does not exist, build this node.
3. Copy all adjacency edges from every query node to all terms nodes include in it.
4. Delete all query nodes and their adjacency edges.

Then we obtain a graph composed by term nodes, URL nodes and directed edges between them.

3.2.2 URL Nodes Elimination

Query expansion technique needs the relations at the level of terms, so the URL nodes are unnecessary, and what's more, it will cause lots of extra cost. For this reason, we decide to eliminate them by following steps:

1. For every term node pair (v_{t1}, v_{t2}) and every URL node v_u, if there are two edges from v_{t1} to v_u and from v_{t2} to v_u concurrently, then add two edges in the graph, one is from v_{t1} to v_{t2} and the other is from v_{t1} to v_{t2}, that both have weight as $c(v_{t1}, v_{t2}, v_u)$, or add the weight of edge to $c(v_{t1}, v_{t2}, v_u)$.
2. Delete all URL nodes and their adjacency edges.

Besides, $c(v_{t1}, v_{t2}, v_u)$ represents the value of crossover frequency which can be compute by following formula:

$$c(v_{t1}, v_{t2}, v_u) = \min\left(\omega(v_{t1} \to v_u), \omega(v_{t2} \to v_u)\right) \tag{1}$$

To sum up, after these steps in Subsection 3.2.1 and Subsection 3.2.2, we can transform query-click graph into term-relationship graph, and the transformation has many practical significance:

1. The term-relationship graph is conducive to mine the association between terms in query logs.
2. The analysis is at the level of terms rather than at the level of queries, and it is useful to help to reformulation an effective query.
3. Compared to the number of queries and URLs, the number of terms is quite small, so the space which the term-relationship graph takes in the memory is smaller.
4. Compared to the bi-partite graph model, the process of SimRank and optimization based on one type node graph will be more convenient.

4 Computing SimRank

4.1 Naïve Method

SimRank [4] is a method for computing object similarities, applicable in any domain with object-to-object relationships, that measures similarity of the structural context in which objects occur, based on their relationships with other objects. The intuition behind it is that, in many domains, similar objects are related to similar objects. More precisely, objects a and b are similar if they are related to objects c and d, respectively, c and d are themselves similar. The base case is that objects are similar to themselves.

Moreover, SimRank is an iterative technique to compute the similarity score for each pair of objects. In our case, we can consider the terms as one type of objects and use SimRank to compute similarity scores for each term-term pair.

For a node v_t in the term-relationship graph, we denote by $I(v_t)$ the set of in-neighbors, and individual in-neighbors are denoted as $I_i(v_t)$, for $1 \le i \le I(v_t)$. Let $Sim(v_a, v_b)$ denote the similarity between term v_a and v_b. For $v_a \ne v_b$, we write the naive equation:

$$Sim(v_a, v_b) = \frac{C}{|I(v_a)||I(v_b)|} \sum_{i}^{|I(v_a)|} \sum_{j}^{|I(v_b)|} Sim\left(I_i(v_a), I_j(v_b)\right) \tag{2}$$

If $v_a = v_b$, then $Sim(v_a, v_b)$ is defined to be 1. Otherwise, where C is a constant between 0 and 1, gives the rate of decay as similarity flows across edges. In the SimRank paper [4], it is shown that a simultaneous solution $Sim(*,*) \in [0,1]$ to the above equations always exists and is unique. Also notice that the SimRank scores are symmetric, i.e. $Sim(v_a, v_b) = Sim(v_b, v_a)$.

According to the Equation (2), we can use the SimRank score as the association for query expansion instead of co-occurrence degree and probability of translating.

4.2 Normalized Weight SimRank

In the term-relationship graph, there may be many "unpopular" terms, i.e., term nodes with very few in-degree. Although the scarcity of contextual information makes them difficult to analyze, these terms are often the most useful, since they tend to be harder to find. Unlike the simple co-occurrence scheme, SimRank can effectively analyze terms with little contextual information in sparse graph.

Unfortunately, because of the weight on graph edges, the naive SimRank fails to properly identify term similarities in our application. In the naive form SimRank, the information of weights is neglected and we tried to derive similarity scores for term pairs by just using the term-relationship graph's structure.

In general, in order to utilize the edge weights in the computation of similarity scores. We first normalize the in-edge weights for every node v_t in the term-relationship graph by following the next equation:

$$\sum_{i}^{|I(v_t)|} \omega\left(I_i\left(v_t\right) \to v_t\right) = 1 \tag{3}$$

And now, we can incorporate the normalized weight into the SimRank equations. In the new form SimRank, named normalized weight SimRank (NWS), we modify the Equation (2) as follows:

$$NWS\left(v_a, v_b\right) = \begin{cases} C \sum_{i}^{|I(v_a)|} \sum_{j}^{|I(v_b)|} NWST\left(I_i\left(v_a\right), I_j\left(v_b\right)\right) & v_a \neq v_b \\ 1 & v_a = v_b \end{cases} \tag{4}$$

$$NWST\left(I_i\left(v_a\right), I_j\left(v_b\right)\right) = \omega\left(I_i\left(v_a\right) \to v_a\right)\omega\left(I_j\left(v_b\right) \to v_b\right) NWS\left(I_i\left(v_a\right), I_j\left(v_b\right)\right) \tag{5}$$

The value $\omega(v_a \to v_b)$ corresponds to the normalized weights of the edge that start from v_a to v_b.

4.3 Pruning

Let us analyze the time and space requirements for this method of computing SimRank. Let d be the average of $I(v_t)$, and K be the number of iterations of SimRank computing. The space required is simply $O(n^2)$ to store the results. The time required is $O(Kn^2d^2)$, since on each iteration, the score of every node-pair (n^2 of these) is updated with values from its in-neighbor pairs (d^2 of these on average). We mentioned that typically $K \approx 5$ [4], n^2 can be prohibitively large in some cases, such as the query logs. Thus, in this subsection we do briefly consider pruning techniques that reduce both the time and space requirements.

Our way to reduce the resource requirements is to prune the term-relationship graph. So far we have assumed that all n^2 node-pairs graph are considered, and a similarity score is computed for every node-pair. In the SimRank paper [4], it is shown that if n is significantly large, it is very likely that the neighborhood (say, nodes within a radius of 2 or 3) of a typical node will be a very small percentage of the entire domain. Nodes far from a node v, whose neighborhood has little overlap with that of v, they will tend to have lower similarity scores with v than nodes near it. Thus one pruning technique is to set the similarity between two nodes far apart to be 0, and consider node-pairs only for nodes which are near each other. So, in our works, we consider only node-pairs within a radius r of n from each other, and there are on average d_r such neighbors for a node, then the time and space complexities become $\mathrm{O}(Knd_r d^2)$ and $\mathrm{O}(nd_r)$ respectively.

Moreover, in most cases we also expect the average in-degree to be relatively small. According to the Equation (4) and (5), it is very likely that the in-edge with a very small weight will tend to have smaller contribution to the $NWS(v_t, *)$ score than edges with large weight. So another pruning technique is to limit the maximum in-degree to a fixed value of d_m, consider only the top d_m edges sorted by their weight, and then the time complexities become $\mathrm{O}(Knd_r d_m^{\ 2})$.

Of course, the quality of the approximation needs to be verified experimentally for the actual datasets. For the case of scientific papers, our empirical results suggest that this is a good approximation strategy, and allows the computation to be carried out entirely in several days for a query-log of $n = 413{,}013$ terms. More details can be found in Section 6.

5 Query Expansion

For any given query $q = w_1 \ldots w_{i-1} w_i w_{i+1} \ldots w_n$, our approach will iterate through all terms w_i and try to find similar terms in the query logs for each of them. For each w_i, the approach select candidate terms and do query expansion by following three steps:

1. If there is a node of t_i in the term-relationship graph, then consider every terms t_j with $NWS(v_i, v_j) \neq 0$ as a candidate term.
2. Remain only the top M expansion candidate terms t_j sorted by their $NWS(v_i, v_j)$.
3. Join all remaining candidate terms in the original query, and then retrieval using the new query.

In the follow sections, the method above is called NWSE (Normalized Weight SimRank based Expansion).

6 Experiments

6.1 Dataset

Our experiments are based on the query logs distributed by AOL, containing 36,389,567 records, 10,154,742 distinct queries and 657,426 users. Most of them are in English and sampled during three month and including an anonymous user-id, a query, a timestamp, and the results (for each result, the position on the result page is also provided).

All evaluation is done using three standard TREC collections: Robust04 and Gov-2, containing 528,155 documents and 25,205,179 documents respectively. It should be noted that the two collections are quite different: Robust04 is a newswire collection, while the Gov-2 is a web collections.

We used 150 queries associated with each collection for evaluation (No.301-450 queries for Robust04 and No.701-850 queries for GOV-2 respectively). In particular, we treat all topic titles as queries and neglect their description.

6.2 Design of Experiments

6.2.1 Baseline

In order to evaluating the effectiveness of our method, we use two methods as the baseline for comparison. One is the language model implemented by Indri toolkit and the Dirichlet smoothing prior μ was set to 1500 empirically. This method is denoted by LM-Dir. The other one, we implemented a Relevance-Based Language Model, one of the PRF expansion approaches, based on the method by Lavrenko and Croft [10], denoted by RM.

We used precision at 10 (P@10), precision at 20 (P@20) and MAP to measure retrieval effectiveness for all experiments.

6.2.2 Query Expansion with NWSE

In order to remove noisy records, we remove all index queries and meaningless queries from AOL query logs, i.e., queries such as “fidelity.com” and “wu 20v 20-----”. And stopword removal was done in this process. Moreover, there are also a large number of adult queries, and we did not use it in our experiments, though. For creating the term-relationship graph, we creating the query-click graph first, that contains approximately 4 million query nodes and 2 million clicked URL nodes. After several transformations have been described in Subsection 3.2, the term-relationship is obtained finally, containing 413,013 term nodes.

Before computing the similarity scores, let’s discuss the parameter for pruning. Table 1 shows that how the d_m (the maximum number of in-edges considered by the NWS) influences the NWS scores, When M (the number of candidate terms remained) is set to 5 empirically. We find that the NWS scores nearly no longer grow, if the value of d_m is bigger than 500, so the d_m is set to 500 in all experiments.

Table 1. Examples of NWS similarity when d_m takes different values

d_m :	100	300	500	700	900
NWS("history", "record")	0.0031	0.0067	0.0083	0.0086	0.0088
NWS("diamond", "gold")	0.0047	0.0069	0.0079	0.0081	0.0081
NWS("state", "country")	0.0078	0.0094	0.0105	0.0107	0.0107

Similarly, we set r (the radius of similarity transmission) and K (the number of iterations) to 2 and 5 respectively.

According to the description in section 5, we can expand the original query weight candidate terms. The candidate terms and the original query terms are combined using "#weight" operator and "#combine" operator implemented in Indri, and Equation 6 is implemented by creating an Indri query of the form:

$$\#weight(\ \lambda\ Q_{ori}\ (1.0-\lambda)\ Q_{exp}) \tag{6}$$

Where Q_{ori} is the original query and λ is free parameter determining the weight given to the original query. Specifically, we work out two different strategies for creating the final query. One is all candidate terms are considered equally good in the Q_{exp}, named NWSE-Unweighted. The other one is adding their NWS scores in the final query as weights, named NWSE-Weighted.

6.3 Results

Figure 1 shows the results of assigning different λ on two TREC collections, when the M is set to 30 empirically. As can be seen in Figure 1, the RM method perform well when $\lambda = 0.3$ and $\lambda = 0.6$. On the other hand, the MAP of the NWSE-Weighted method and NWSE-Unweighted method is higher than the RM method's. The NWSE-Weighted method and NWSE-Unweighted method performed the best when λ is set to 0.5 for Robust 04, 0.9 and 0.8 respectively for Gov-2. Finally, we set the parameter λ to optimal values for each of methods.

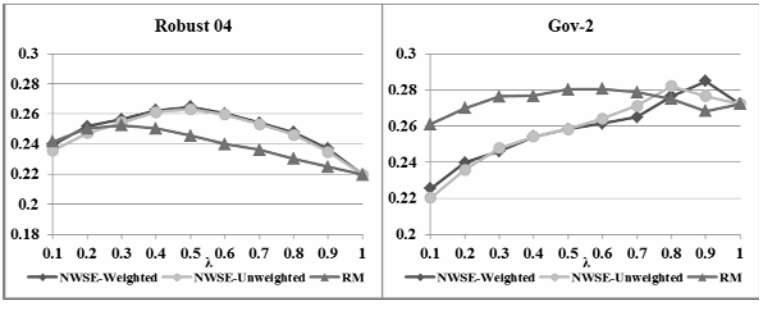

Fig. 1. MAP performance of all query expansion approach when M=30

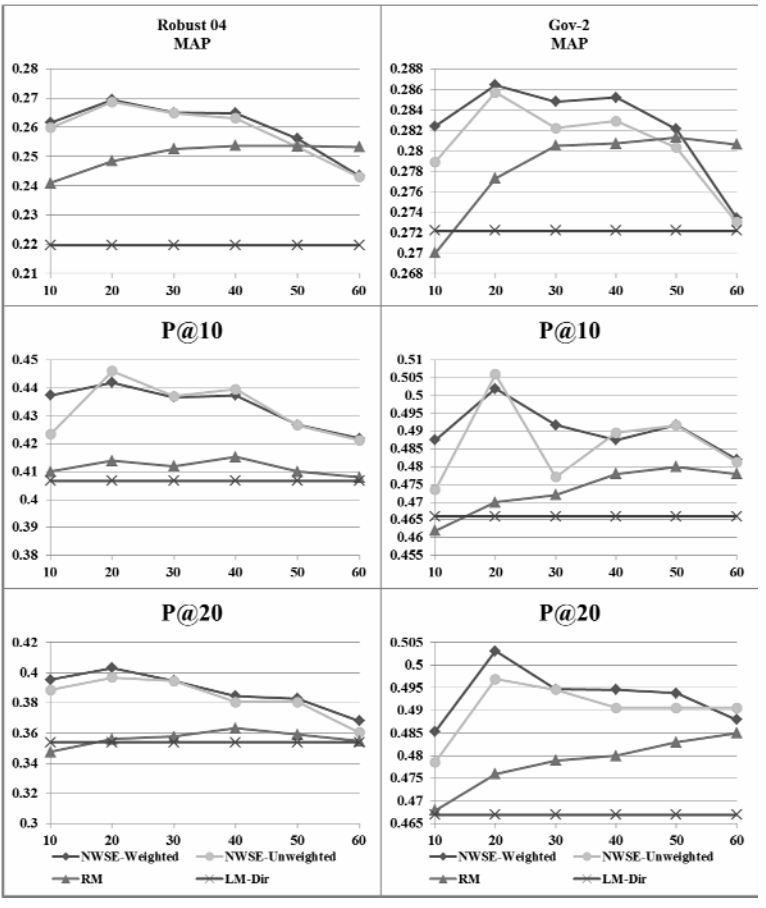

Fig. 2. MAP, @10 and P@20 score curves for all query expansion results

The Figure above show the results of assigning different value to M for all methods on three TREC collections. The MAP of our methods are 6.19% and 1.81% higher than the query expansion based on pseudo relevance feedback, 7.49% and 5.44% higher on P@10, and 10.96% and 3.73% higher on P@20 for Robust 04 and Gov-2 respectively.

According to Figure 2, we can see the NWSE-Weighted and NWSE-Unweighted methods preform as well as the RM model does in the MAP evaluation. We also can see that the methods we proposed outperform the RM method, it shows that the expansion terms selected from query logs by NWSE is more relevance than them extracted from pseudo-relevance documents.

Figure 2 shows the trend of MAP, P@10 and P@20 for all methods. Comparing to the LM-Dir method, we can see all of the three query expansion methods have the very great scope enhancement in the evaluating metrics. The NWSE-Weighted and NWSE-Unweighted methods perform well when $20 < M < 40$, but RM method does the same when the expansion scale is bigger. It means that the effective retrieval

process can be finish in a short time by using the NWSE methods, just because more terms in query cause more time and space cost. Specially, we notice that the NWSE-Weighted works better and more stable than NWSE-Unweighted, and it proves that the weights provide by NWS scores is quality.

7 Conclusion

In this paper, we have explored the AOL query logs as a new resource for query expansion. We generated a weighted bi-partite query-click graph, with queries on one side and URLs on the other, and then, obtained the term-relationship graph by several transformations from it. We also proposed new methods for mining expansion terms from AOL query logs. Our methods named normalized weight SimRank are based on a revised SimRank algorithm, and we measure the importance of expansion terms according to their NWS with the original query terms. Our experiments show that the expansion terms extracted from AOL query logs are better than those from the feedback documents, using a simple statistical method, on the different TREC collections.

There are a few limitations of our work. First, all the experiments are based on click-throughs instead of real relevance judgments, so a future work would be to test the proposed methods with real relevance data. Second, query logs have more meaningful information of sessions was neglected in our experiments. The last, because of the great capacity for query log, we do not have enough time and equipment for pruning radius optimization. So we will pay more attention on this in our future work.

Acknowledgements

This work was supported in part by grant from the Natural Science Foundation of China (No. 60673039 and 60973068), the National High Tech Research and Development Plan of China (2006AA01Z151), the Doctoral Fund of Ministry of Education of China 20090041110002.

References

1. Dang, V., Croft, W.B.: Query Reformulation Using Anchor Text. In: Proceedings of WSDM 2010, New York City, New York, USA, pp. 41–50 (2010)
2. Wang, X., Zhai, C.: Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In: Proceedings of CIKM 2008, Napa Valley, California, USA, pp. 479–488 (2008)
3. Boldi, P., Bonchi, F., Castillo, C.: Query Suggestions Using Query-Flow Graphs. In: Proceedings of WSCD 2009, Barcelona, Spain, pp. 51–58 (2009)
4. Jeh, G., Widom, J.: SimRank: A Measure of Structural-Context Similarity. In: Proceedings of SIGKDD 2002, Edmonton, Alberta, Canada, pp. 538–543 (2002)
5. Sparck, J.K.: Automatic Keyword Classification for Information Retrieval. Butterworth, London (1971)
6. Croft, W.B., Xu, J.X.: Query Expansion Using Local and Global Document Analysis. In: Proceedings of SIGIR 1996, Zurich, Switzerland, pp. 4–11 (1996)

7. Diaz, F., Metzler, D.: Improving the Estimation of Relevance Models Using Large External Corpora. In: Proceedings of SIGIR 2006, Seattle, Washington, USA, pp. 154–161 (2006)
8. Rocchio, J.: Relevance Feedback in Information Retrieval. J. The SMART Retrieval System: Experiments in Automatic Document Processing 21, 313–323 (1971)
9. Robertson, S., Sparck, J.K.: Relevance Weighting of Search Terms. J. The American Society for Information Science 27, 129–146 (1976)
10. Lavrenko, V., Croft, W.B.: Relevance Based Language Models. In: Proceedings of SIGIR 2001, New Orleans, Louisiana, United States, pp. 120–127 (2001)
11. Antonellis, I., Molina, H.G., Chang, C.C.: Simrank++: Query Rewriting through Link Analysis of the Click Graph. In: Proceedings of VLDB 2008, Auckland, New Zealand, pp. 408–421 (2008)
12. Collins-Thompson, K., Callan, J.: Query Expansion Using Random Walk Models. In: Proceedings of SIGIR 2005, Bremen, Germany, pp. 704–711 (2005)
13. Xu, Y., Jones, J.F., Wang, B.: Query Dependent Pseudo-Relevance Feedback Based on Wikipedia. In: Proceedings of SIGIR 2009, Boston, Massachusetts, USA, pp. 59–66 (2009)
14. Baeza-Yates, R.: Graphs from Search Engine Queries. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 1–8. Springer, Heidelberg (2007)
15. Baeza-Yates, R., Tiberi, A.: Extracting Semantic Relations from Query Logs. In: Proceedings of SIGKDD 2007, New York, NY, USA, pp. 76–85 (2007)
16. Beeferman, D., Berger, A.: Agglomerative Clustering of A Search Engine Query Log. In: Proceedings of SIGKDD 2000, Boston, Massachusetts, USA, pp. 407–416 (2000)
17. Silverstein, C., Marais, H., Henzinger, M.: Analysis of A Very Large Web Search Engine Query Log. ACM SIGIR Forum 33(1), 6–12 (1999)