Mining Search Engine Query Logs for Query Recommendation[*]

Zhiyong Zhang
Dept. of Computer Engr & Computer Science
Speed School of Engr, University of Louisville
Louisville,KY 40292, USA
z0zhan13@louisville.edu

Olfa Nasraoui
Dept. of Computer Engr & Computer Science
Speed School of Engr, University of Louisville
Louisville,KY 40292, USA
olfa.nasraoui@louisville.edu

ABSTRACT

This paper presents a simple and intuitive method for mining search engine query logs to get fast query recommendations on a large scale industrial-strength search engine. In order to get a more comprehensive solution, we combine two methods together. On the one hand, we study and model search engine users' *sequential* search behavior, and interpret this consecutive search behavior as client-side query refinement, that should form the basis for the search engine's own query refinement process. On the other hand, we combine this method with a traditional *content* based similarity method to compensate for the high sparsity of real query log data, and more specifically, the shortness of most query sessions. To evaluate our method, we use one hundred day worth query logs from SINA' search engine to do off-line mining. Then we analyze three independent editors evaluations on a query test set. Based on their judgement, our method was found to be effective for finding related queries, despite its simplicity. In addition to the subjective editors' rating, we also perform tests based on actual anonymous user search sessions.

Categories and Subject Descriptors: H.2.8 Information Systems:Database Applications-Data mining

General Terms: Algorithms, Performance, Design

Keywords: Recommendation, Session, Mining, Query Logs

1. INTRODUCTION AND RELATED WORK

Providing related queries for search engine users can help them quickly find the desired content. Recently, some search engines started showing related search keywords in the bottom of the result page. Their main purpose is to give search engine users a comprehensive recommendation when they search using a specific query. Recommending the most relevant search keywords set to users not only enhances the search engine's hit rate, but also helps the user to find the desired information more quickly. Also, for some users who are not very familiar with a certain domain, we can use the queries that are used by previous similar searchers who may have gradually refined their query, hence turning into *expert searchers*, to help guide these novices in their search. That is, we can get query recommendations by mining the search engine query logs, which contain abundant information on past queries. Search engine query log mining is a special type of web usage mining. In [2], a "content-ignorant" approach and a graph-based iterative clustering method was used to cluster both the URLs and queries. Later in [3], the authors presented a well-rounded solution for query log clustering by combining content-based clustering techniques and cross-reference-based clustering techniques. In [1], methods to get query recommendation by utilizing the click-through data were presented.

[*]part of this work was done while the first author was at Search Engine R&D Group, SINA Corporation, Beijing, China

Copyright is held by the author/owner(s).
WWW 2006, May 22–26, 2006, Edinburgh, Scotland.
ACM 1-59593-332-9/06/0005.

2. ONE DIMENSIONAL GRAPH MODEL BASED SIMILARITY FOR CONSECUTIVE QUERIES

As illustrated in Figure 1, suppose we have three consecutive queries after pruning identical ones in one query session, the vertices in the graph represent the query and the arcs between them represent their similarities. We will define the value of the arcs as a forgetting factor or damping factor d, in $(0, 1)$. For two consecutive queries (neighbors in the graph) in the same query session, their similarities are set to this damping factor d, and for queries that are not neighbors in the same session, their similarities are calculated by multiplying the values of the arcs that join them. For example, in Figure 1, the similarity between query1 and query3 would be $\sigma(1,3) = d \times d = d^2$. The longer the distance, the smaller the similarity. After getting the similarity

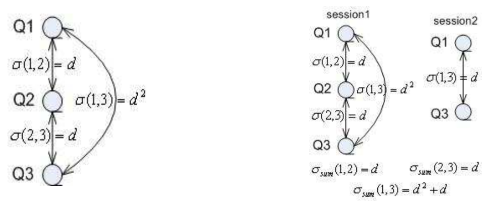

Figure 1: Similarity Graph for 1 session (left) and 2 sessions (right)

values of each pair in each session, we add them up to get the cumulative similarity between two queries. This process is shown in the right side of Figure 1.

Our model works by accumulating many query sessions and adding up the similarity values for many same query pairs, and by keeping a query's most similar queries in the final clusters. Our experiments confirm these expectations. For example, using our method, when we input "River Flows Like Blood", a novel written by a Chinese writer "Haiyan", we not only obtained the writer himself as its top-ranked related queries, but also obtained his other popular novels in the top-ranked clusters.

3. CONTENT BASED SIMILARITY AND COMBINATION OF TWO METHODS

For content-based similarity calculation, we use the cosine similarity given by

$$\sigma(p,q) = \frac{\sum_{i=1}^{k}(cw_i(p)) \times (cw_i(q))}{\sqrt{\sum_{i=1}^{m} w_i^2(p)} \times \sqrt{\sum_{i=1}^{n} w_i^2(q)}} \quad (1)$$

where $cw_i(p)$ and $cw_i(q)$ are the weights of the i-th common keyword in the query p, and q respectively and $w_i(p)$ and $w_i(q)$ are the weights of the i-th keywords in the query p and q respectively. For weighting the query terms, instead of using $TF*IDF$, we use $SF*IDF$, which is the search frequency multiplied by the inverse document frequency. The reason why we use SF instead of TF is, considering the sparsity of the query terms in one query, TF in one query makes almost no difference. While using SF in its position is a kind of voting mechanism that relies on most people's interests and judgements. SF is acquired from all search queries, while IDF is acquired from the whole document set.

Using the above two methods, we can get two different types of query clusters. The two methods have their own advantages and they are complementary to each other. When using the first consecutive query based method, we can get clusters that reflect all users' consecutive search behavior as in collaborative filtering. While using the second, which is content based method, we can group together queries that have similar composition. To quickly combine them together, for one specific query, we will get two clusters of related queries, which are sorted according to their cumulative similarity voting factor, when using the two methods. We then use the following similarity combination method to merge them together.

$$similarity = \alpha \times sim_{consec\_query} + \beta \times sim_{content} \quad (2)$$

where α and β are the coefficients or weights assigned to consecutive-query-based similarity measures and content-based similarity measures respectively. In our experiments, we give them the same value.

4. EVALUATION AND EXPERIMENTAL RESULTS

We use two methods for evaluating our methods. First, we randomly chose 200 testing sessions, each of which containing two queries or more. These sessions were extracted from one week's query logs from April 2005, so they have no overlap with the query logs used in the training set. The latter contained the query logs for September, October, and November. We did our test based on a maximum of 10 recommendations per query. We then counted the number of recommendations that matched the hidden queries of one session. Table 1 shows the testing results, where n1 is the number of remaining queries that fall into our recommendation set for the *first* query in the same session, while n2 is the number of remaining queries that fall into our recommendation set for the *second* query in the same session.

Table 1. Session Coverage Result

session length	num of sessions	n1	n2	average coverage
2-query session	147	35	29	21.8%
3-query session	37	23	19	28.4%
4-query session	14	5	9	16.7%
6-query session	2	2	0	10%
total average coverage(200 sessions/473 queries)				22.3%

Search engine users' subjective perceptions may be more likely to indicate the success or failure of a search engine. For this reason, in our second method, we used editors' ratings to evaluate our query recommendations. We tested one hundred queries. These queries were selected by our editors according to several criteria. These criteria include that they have high visit frequency and that they cover as many different fields as possible. For each query, we ask our editors to rank the recommended query results from 5 to 0, where 5 means very good and 0 means bad. For precision, we ask them to review whether the related queries are really related to the original query. For coverage, we ask them whether they would click on the query recommendations in a real search scenario.

Table 2. Editor Evaluation Result

Ranking	coverage(among 100)	precision(among 100)
very good(5)	35	29
good(4)	33	42
marginally good(3)	13	11
bad(2)	9	8
very bad(1)	7	7
nothing(0)	3	3
average rating	3.71	3.69
standard deviation	1.37	1.32

In Table 2, the numbers mark how many query recommendations are ranked in the specific rank among the one hundred queries. For example, in the second column (coverage), 35, 33, and 13 queries were rated as very good, good, and marginally good respectively.

5. CONCLUSIONS AND FUTURE WORK

It is presumable that K-means or a simple K-Nearest Neighbors method can be used to obtain the related query clusters. However, one problem in this case would be how to fix the K value. In comparison with these methods, our method is adaptive and autonomous since we don't predefine any fixed number of clusters. Instead, it would fluctuate with the users' search patterns or some events related to search trends in general. Also, more work can be done for mining the users' sequential search behavior for search engine query refinement, and to take into account the *evolving* nature of users' queries.

6. ACKNOWLEDGMENTS

This work is partially supported by a National Science Foundation CAREER Award IIS-0133948 to Olfa Nasraoui.

7. REFERENCES

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Creete, Greece, March (to apper in LNCS).*, 2004.

[2] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. *Proceedings of ACM SIGKDD International Conference, pages 407–415.*, 2000.

[3] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems, 20(1), pages 59 – 81.*, 2002.