Semantic Search Guidance: Learn From History

M. Barouni-Ebrahimi, Reza Zafarani, Ebrahim Bagheri, and Ali A. Ghorbani
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada
{m.barouni, r.zafarani, e.bagheri, ghorbani}@unb.ca

Abstract

User queries submitted to web search engines are not always informative enough for retrieving the related pages to the user intention. The main problem is that users may not know the best query items they should enter to get the most related web pages to their intentions. They may not be familiar with the specific keywords in that domain knowledge. A user may remember only a part of the phrase that he/she wants to use in the query string. Sometimes the user does not know how to order the keywords (most web search engines are sensitive to the order of the keywords) or even does not know the correct spelling of a specific keyword in the query string. A novice user sometimes sends an imperfect query and scans the returned web pages (even reads a number of the returned documents) to prepare a more precise query by finding new related keywords in the documents. To assist the users in formulating their search queries, we propose two query recommendation algorithms which are based on frequent query phrase similarity, and latent semantic indexing, respectively. The proposed models have been introduced in this paper. They have been evaluated in order to analyze their performance based on the contribution and point-wise mutual information metrics. The algorithms show a promising performance and can serve as a nice starting point for further investigation.

1 Introduction

The incredible growth of the volume of accessible information on the web has brought about confusion for the proper access and navigation of users attempting to access their required information, leaving them lost in the hyperspace despite the use of powerful web search engines. Large number of recommended web pages by a search engine for a single query leaves a user wondering which pages may be suitable for his/her purpose. Although the exploitation of more powerful ranking schemes within a search engine results in a more precise outcome, the single-sided approach to this problem seems to be insufficient. Users in a typical setting are required to enter appropriate query phrases comprehensible to the search engine. Our observation in the AOL query log discussed in [2] shows that the average items in a query is 2.14 while it is worse in the University of New Brunswick (UNB) search engine query log which is 1.94, which is often insufficient for finding the most relevant web pages.

The lack of user knowledge or familiarity with the specific keywords and phrases in the target domain leaves him/her wondering what phrases would be the most relevant ones to his/her intentions. A significant amount of research focuses on mining search engine query logs in order to identify useful past query patterns that may assist novice searchers. These works can be classified into three high-level categories, namely: query expansion, query recommendation, and query completion.

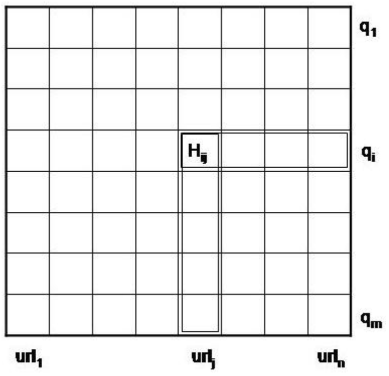

Figure 1: Hit matrix in I-SPY system.

1.1 Model Overview

In [11], a web search method is proposed called I-SPY that relies on the past users' selections to re-rank the query results. The algorithm sends the submitted query to a conventional web search engine. The results are re-ranked based on the past community selections. The number of users in a community selecting each page of the results is stored in a hit-matrix. As shown in Figure 1, q_i is a query submitted by users, url_j is a page of the results for q_i and H_{ij} is the number of users selecting url_j for q_i. For a new user query, the results are re-ordered based on the relevancy of the retrieved pages to the submitted query using the hit-matrix. The relevance value of the page url_j to the query q_i is calculated as follows:

$$Relevance(url_j, q_i) = \frac{H_{ij}}{\sum_{j=1}^{n} H_{ij}}, \tag{1}$$

The paper reports that only 15% of the general queries are duplicated. This causes two problems. First, the number of queries rapidly increase. This makes the hit-matrix very large that has effects on the performance of the system. Second, for most queries, the number of users' selections may not be large enough to be considered valid by the algorithm. Therefore, a similarity between queries calculated by Equation 2 is used to increase query duplication. Two queries are considered to be the same if they are within a given similarity threshold.

$$Sim(q, q') = \frac{|q \bigcap q'|}{|q \bigcup q'|}, \tag{2}$$

where q and q' are two queries, $|q \bigcap q'|$ is the number of the identical items in q and q' and $|q \bigcup q'|$ is the number of the distinct items in q and q' . This helps the system to have a smaller hit-matrix as well as more hits for a specific page of a submitted query. However, this has a negative effect on the precision.

We alter this approach by employing frequent phrases within the queries rather than considering the whole query as a single element. The queries q_1 to q_m in the hit-matrix are replaced with the frequent phrases within the query log. In this way, the number of queries in the hit-matrix is much smaller, while it yields more relevant query-URL relationships. We further propose two semantic inferencing algorithms to identify related query terms from the hit-matrix. These algorithms identify query similarity based on the selected URLs by the search engine users.

The paper is structured as follows: Section 2 gives an overview of some of the research that has been performed related to query suggestion, expansion, and completion. Section 3 will then introduce our proposed algorithms. The algorithms are evaluated using several criteria in Section 4, and the paper is finally concluded in Section 5.

2 Related Work

Query expansion is a way to resolve the problem of submitting very short queries [7]. A newly submitted user query is expanded through the concatenation of other related items to the initial query expression. Two methods for query expansion have been described by [8]. In one of the methods, the user query is sent to the search engine and the key terms extracted from the top returned documents are then added to the query. The new generated query is sent again to the search engine. In the second method, a list of past queries is provided. For a newly submitted query, three most similar queries are selected from the set of queries. An affinity pool of the documents is created based on the top document results of the similar queries. The user query is then applied to the affinity pool and the key terms extracted from the top resulted documents are added to the query. In another work, the algorithm proposed in [5] extracts the probabilistic correlation between the query terms and the document terms by analyzing query logs. The high quality expansion terms are then selected from document space for automatic query expansion.

Baeza-Yates et al. [1] propose a method, which suggests a rank-ordered query list for any submitted query. The suggested queries are ordered based on two relevancy criteria: 1) the similarity of the suggested queries to the submitted query and 2) the degree of user's interest to the results of the suggested queries. In this method, a dataset of query sessions is employed. Each query session consists of a query submitted to a web search engine along with the related URLs selected from the set of its results. The algorithm initially clusters the queries along with the texts of their clicked URLs. Given an input query, it finds the appropriate cluster of the query and computes the rank score for each query of that cluster. The queries of the cluster being suggested are ordered with respect to their rank scores. In [13], a different solution has been proposed for query clustering, where two methods have been combined together to reach a more comprehensive solution. The users' sequential search behavior is considered as a primary method of clustering, which is combined with a traditional content-based similarity method.

The algorithm proposed in [10] defines a query-base as a set of query vectors. For each persistent query, a query vector is created which points to a list of related retrieval outputs. The user query is then compared with the query-base to find a persistent query whose similarity to the query satisfies a minimum similarity threshold. The related retrieval output of the selected persistent query is then presented to the user. On the other hand, the persistent queries can be used to help the users formulate appropriate queries. Community Search Assistant [9] is another software agent program that suggests a list of related queries. It creates a query graph in which each node is a past user query. Two nodes are connected whenever the two queries have more than a specific number of common items in the *n* top document results. The queries of the query graph, related to a new submitted query are then suggested to the user.

Furthermore, a query completion algorithm has been proposed in [4] that suggests the frequent words containing the last incomplete word in the query. The suggested words are extracted from the documents that contain the previous words of the incomplete query. The structure of the query space is different from the structure of the document space. Google has launched the Google suggestion service that recommends relevant phrases in order to perform query completion. A deficiency of Google's method is that the suggestions are only provided for the first phrase in a query. If the user attempts to enter two different frequent phrases in the query, Google does not provide any suggestions for the second one. It seems that Google deals with the whole query as one phrase. On the other hand, the suggestions are only the words that can be appended to the end of the query. What if the user only knows the middle part of a query, or needs semantically related suggestions?

3 Proposed Query Suggestion Models

The research described here focuses on query suggestion techniques which are mainly based on inter-query similarities in frequent phrases. In this section we propose two different methods for query suggestion, namely: Semantic Phrase Adviser (SPA), and LSI-based Recommender (LSIr). The former elicitates inter-phrase proximity based on shared URLs between phrases whereas the latter is forming URL neighborhoods in order to find semantically similar phrases.

3.1 Semantic Phrase Adviser (SPA)

The hit-matrix sits at the focal point of SPA. The values in each cell of the hit-matrix are the occurrence frequency of each URL with regards to a given frequent phrase which has already been mined using the OFSD algorithm [3]. The rows of the hit-matrix in our model are frequent phrases within the queries while they are grouped queries in I-SPY. Furthermore, in our model, the frequency value is used instead of the occurrence number of the URL, since the contents of a URL may change over time, and therefore, the relevancy of the URL and the phrase may also increase or decrease. Hence, the occurrence frequency value used in the hit-matrix considers 'observation time' as one of its important factors in developing the occurrence frequency value. The occurrence frequency value used in SPA is defined as follows:

$$hit\_matrix(P, url) = \frac{Num(P, url)}{N_{t_P}} \tag{3}$$

where Num(P,url) defines the number of times that url has been visited for a given phrase P and N_{t_P} represents the number of times that a given phrase has been employed. Based on this definition, similar phrases can be mined from within the hit-matrix. For all phrases in the hit-matrix, their similarity values with P are calculated. The similarity of two phrases (P_1, P_2) is calculated using Equation 6.

$$urlList(P) = \{url_i | hit\_matrix(P, url_i) > F_{min}\}. \tag{4}$$

where F_{min} is a user-defined threshold that defines the acceptability boundary with regards to frequency of a URL for a given phrase P.

$$URLset = urlList(P_1) \cap urlList(P_2). \tag{5}$$

$$sim(P_1, P_2) = \frac{\sum_{i=1}^{|URLset|} (hit\_matrix(url_i, P_1) + hit\_matrix(url_i, P_2))}{\sum_{url \in urlList(P_1)|} hit\_matrix(url, P_1) + \sum_{url \in urlList(P_2)} hit\_matrix(url, P_2)} \tag{6}$$

In SPA, for each given phrase P, the top ten most similar phrases in the hit-matrix are suggested to the user. Time complexity is an important issue in query suggestion algorithms, since the users should be informed about relevant queries as soon as possible. As discussed in [3], the time complexity of the frequent phrase miner (OFSD) is of order $O(\log n)$ where n is the number phrases in the hit-matrix. Since the time complexity of extracting the similar phrases for a given phrase is of order $O(n)$, the extraction process for all phrases is done during the pruning process. Therefore, the average time complexity of the algorithm would be of order $O(n \log n)$, which is a suitable time complexity for our purpose.

3.2 LSI-based Recommender (LSIr)

The hit-matrix can be used to represent the contents of the URLs. It can be seen that for a given hit-matrix, related keywords to each URL can be extracted by gathering all phrases for which the specific URL has been clicked on. The intuition behind LSIr is that the URLs which have similar keywords (phrases) can be incorporated to extract similar recommendation phrases for the user query.

In the LSIr algorithm, a list of recommended queries are found based on the obscure relations between the queries in the query-URL space. Latent Semantic Indexing [6] has been used in this model due to its high precision in the area of Information Retrieval. Our proposed algorithm is presented in Figure 2.

The LSIr algorithm reads as follows. LSIr is supplied with the hit-matrix (as described in the previous section) and a query. It also maintains two candidate lists for words and URLs ($CandidateWords$ and $CandidateURLs$) throughout its process. In the first step (lines 1-5), the set of URLs related to the requested query are added to the $CandidateURLs$. From now on, a set of actions are repeated until both candidate sets do not expand any longer (lines 6-12).

```
0.Algorithm LSIr( hit-matrix, requested query )
1.  Begin Algorithm
2.      Set $p$ to the requested query
3.      Set $CandidateURLs$ and $CandidateWords$ to empty.
4.      Add all URLs that have been clicked on for $p$ to $CandidateURLs$.
5.      Add $p$ to $CandidateWords$
6.      While ($CandidateURLs$ and $CandidateWords$ are not changing)
7.      {
8.       For every word $w$ in $CandidateWords$
9.         Add all URLs that have been clicked on for $w$ to $CandidateURLs$.
10.      For every URL $u$ in $CandidateURLs$
11.        Add all words that $u$ has been clicked on for to $CandidateWords$.
12.     }
13.     Generate word-URL matrix $m$ from $CandidateWords$ and $CandidateURLs$
14.     Create a query vector $q$ from the requested query($p$).
15.     Retrieve most relevant URLs to $p$ using LSI.
16.     Return most frequent words in most relevant URLs
17.  End Algorithm
```

Figure 2: LSIr algorithm

First, for each word in $CandidateWords$, all URLs that are clicked on for that specific word are added to our $CandidateURLs$ (lines 8-9). In the second phase (lines 10-11), for every URL in $CandidateURLs$, a set of words for which that word has been clicked on are also added to $CandidateWords$. These steps are executed consecutively until no change is observed in the sizes of both $CandidateWords$ and $CandidateURLs$ sets.

In other words, starting from the given query, a subspace of phrase-URL which can be reached by traversing all connected phrase-URLs is selected. Furthermore, a query is created using the requested phrase. The created subspace is reduced using SVD. The related URLs to the constructed query are extracted using LSI. A sorted set of frequent phrases in the most relevant URLs are extracted and are returned as recommendations.

4 Performance Evaluation

The models are evaluated using two different methods. In the first method, the models are evaluated against a baseline system whereas in the second technique mutual information between suggestions is used to evaluate the results. In the first method, ten different phrases are given to each model (LSIr and SPA) and the best phrase is selected from the top ten recommended phrases. The phrase, and the recommended phrases are sent to Google individually and top five retrieved URLs were selected for each sent query. We used three human evaluators to select best phrases, and analyze the relevancy of the URLs. Relevant URLs for each query are assessed based on the evaluators manual understandings. Relevancy and percentage of contribution can be easily calculated by comparing relevant and irrelevant URLs for each model. Percentage of contribution can be calculated using the following formula:

$$Contribution(Model) = \frac{Number\ of\ relevant\ URLs\ found}{Number\ of\ retrieved\ URLs} \tag{7}$$

A performance diagram of the different models over ten random words along with the average value is shown in Figure 3. It can be observed from this figure that one of the models is usually contributing more than the average value. The figure also shows that at least in 8 out of 10 words, one of the models shows better or equal contribution in comparison with the baseline value.

We have also evaluated recommended phrases using the Pointwise Mutual Information (PMI) between the original phrase and the recommended phrases. PMI is defined as follows [12]:

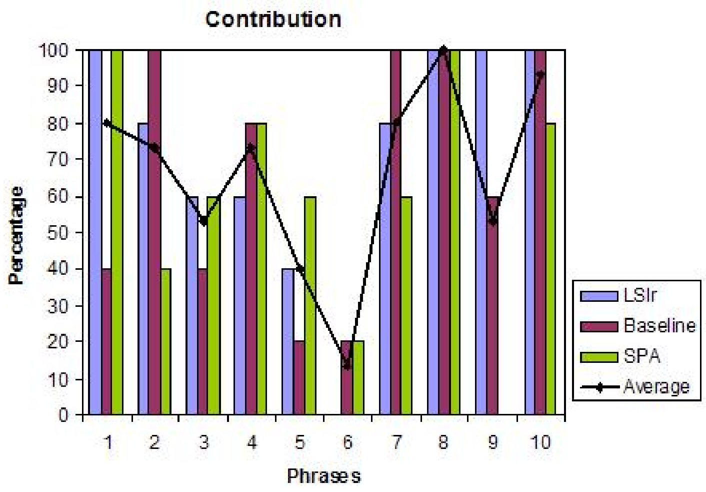

Figure 3: Contribution values for different models

$$PMI(word1,\ word2) = log_2(\frac{Hits(word1\ \&\ word2)}{Hits(word1) \times Hits(word2)}) \quad (8)$$

Here, $hits(phrase\&recommendedPhrase)$ is the number of co-occurrence of $phrase$ and $recommendedPhrase$ in Google search. If the phrases have statistical independence, then the likelihood that they co-occur is calculated by the product $hits(phrase) \times hits(recommendedPhrase)$. The ratio between $hits(phrase\ \&\ recommendedPhrase)$ and $hits(phrase) \times hits(recommendedPhrase)$ is thus a measure of the extent of statistical dependence between the phrases. The log function is used to compute the information value inferred about the presence of one of the words by analyzing the other. PMI values of the proposed models over the same ten random words along with the average value is shown in Figure 4. It can be observed that in most cases the PMI values are close which indicates that the semantic similarity between the given phrase and the phrases recommended by the different models is high.

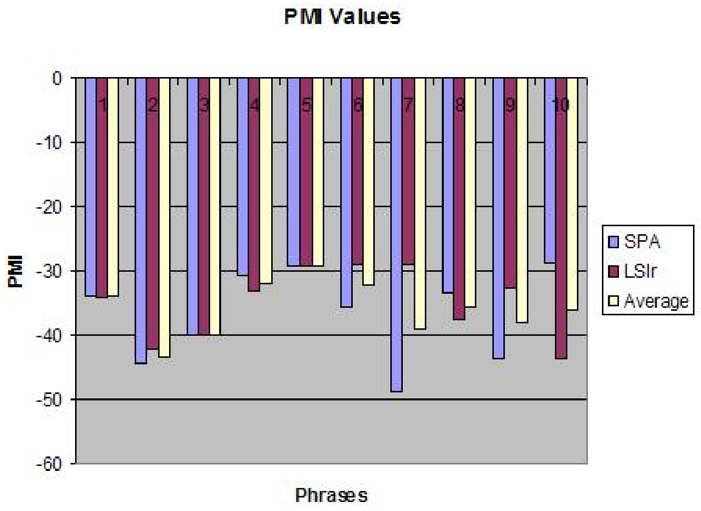

Figure 4: PMI values for proposed models

5 Concluding Remarks

The lack of users' knowledge or familiarity with the specific keywords and phrases in the target domain usually leaves him/her wondering what phrases would be the most relevant ones to use to search the Internet. Query recommendation is one of the ways to resolve this problem. In this approach, a newly submitted user query is refined using other related items to the initial query. In this paper, we have proposed two semantic phrase suggestion algorithms which are based on frequent query phrase similarity, and Latent Semantic Indexing. The algorithms have been evaluated using the PMI and the contribution metrics. The algorithms show a promising performance and can serve as a nice starting point for further investigation. It has been shown that at least in 8 out of 10 test words, one of the models show better or equal performance in comparison with the baseline value. Furthermore, phrases recommended by each of the proposed models were semantically related based on the PMI analysis of the results.

References

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. *Lecture Notes in Computer Science: Current Trends in Database Technology - EDBT 2004 Workshops*, pages 588–596, 2004.

[2] M. Barouni-Ebrahimi and A. A. Ghorbani. On query completion in web search engines based on query stream mining. In *International Conference on Web Intelligence (WI'07), Accepted*, 2-5 Nov. 2007.

[3] M. Barouni-Ebrahimi and A. A. Ghorbani. An online frequency rate based algorithm for mining frequent sequences in evolving data streams. In *international conference on information technology and management (ICITM'07)*, Hong Kong, 2007.

[4] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06)*, pages 364–371, New York, NY, USA, 2006.

[5] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web (WWW '02)*, pages 325–332, New York, NY, USA, 2002. ACM Press.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] F. Ensan, E. Bagheri, and M. Kahani. The application of users' collective experience for crafting suitable search engine query recommendations. In *CNSR*, pages 148–156, 2007.

[8] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '97)*, pages 306–313, New York, NY, USA, 1997. ACM Press.

[9] N. S. Glance. Community search assistant. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI'01)*, pages 91–96, New York, NY, USA, 2001. ACM Press.

[10] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '95)*, pages 344–350, New York, NY, USA, 1995.

[11] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell. Exploiting query repetition and regularity in an adaptive community-based web search engine. *User Modeling and User-Adapted Interaction*, 14(5):383–423, 2005.

[12] P. Turney et al. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, 2002.

[13] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web (WWW'06)*, pages 1039–1040, New York, NY, USA, 2006. ACM Press.