Volume: 05, Issue: 03, Pages:1917-1921 (2013) ISSN : 0975-0290

Query Recommendation employing Query Logs in Search Optimization

Neha Singh
Department of Computer Science, Shri Siddhi Vinayak Group of Institutions, Bareilly
Email: singh26.neha@gmail.com
Dr Manish Varshney
Department of Computer Science, Shri Siddhi Vinayak Group of Institutions, Bareilly
Email: itsmanishvarshney@gmail.com

----------**ABSTRACT**----------
In this paper we suggest a method that, given a query presented to a search engine, proposes a list of concerned queries. The concerned queries are founded in antecedently published queries, and can be published by the user to the search engine to tune or redirect the search process. The method proposed is based on a query clustering procedure in which groups of semantically like queries are named. The clustering procedure uses the content of historical preferences of users registered in the query log of the search engine. The method not only discloses the related queries, but also ranks them agreeing to a relevance criterion. Finally, we show with experiments over the query log of a search engine the potency of the method.

Keywords – **Clustering, search engine.**

Date of Submission: December 29, 2013 Date of Acceptance: January 6, 2014
---

1 Introduction

A major factor for the popularity of today's Web search engines is the friendly user inter-faces they provide and the ease with which information can be retrieved. Indeed, search engines allow users to specify queries simply as phrases constituting a list of keywords, following the traditional approach of information retrieval systems [2]. Keywords may refer to broad topics, to technical terminology, or even to proper nouns that play a major role in guiding the search process for the relevant collection of documents.

Despite the fact that this simple interaction mechanism has proved to be successful for searching the Web, a list of keywords is not always a good descriptor of the information needs of users. It is not always easy for users to retrieve relevant information as they cannot formulate effective queries to search engines. One reason for this is the ambiguity that arises in many terms of a language such as ambiguity aroused due to polysemous words. Queries having ambiguous terms may retrieve documents which are not relevant to user's search because they are retrieved on the basis of page rank. On the other hand, users typically submit very short queries to the search engine, and short queries are more likely to be ambiguous. From a study of the log of a popular search engine, Jansen *et al* [5], conclude that most queries are short (around 2 terms per query) and imprecise. When many users from different domain are searching for the same information may phrase their queries differently. Of-ten, users try all possible different queries until they are satisfied with the results. In order to formulate effective queries, users may need to be familiar with various specific terminologies in a knowledge domain which can help in refining results. However users may have little knowledge about the information they are searching, and worst, in might be the case that they could not even be certain about what to search for. As an example, a student for his science project searching for the word "*crane*" could result in information about mechanical heavy duty machine instead of living creature (a type of bird). In contrast, an ornithologist may have the expertise to submit queries with the term *crane*, when they are looking for some information regarding this bird. The idea is to use these expert queries generated by the domain specialist to help non-expert users.

In order to overcome these ambiguity and short queries problems, some search engines have implemented methods to suggest alternative queries to users. Their aim is to help the users to specify alternative related queries in their search process so as to yield more refine needed result. Typically, the list of suggested alternative queries is evaluated by processing the query log of the search engine. This query log stores the history of previously submitted queries and in relation the URL's selected as their answers. In this context, a problem is how to model the information needs associated to a query. In previous work, some proposed models (e.g., [3]) represent a query as the set of URL's clicked by users for the query. However there is a limitation with this approach when it comes to identify similar queries because two related queries may have different output URL's in the first places of their answers, thus inducing clicks in different URL's. In addition, according to an empirical study [1], the average number of pages clicked per answer is very low i.e. around 2 clicks

Volume: 05, Issue: 03, Pages:1917-1921 (2013) ISSN : 0975-0290

per query. Our data also shows the same.

In traditional document retrieval, the quality of service depends on the query recommendation which present the ordering in which the queries are returned to the user, even more important than the set of recommendations itself. As far as we know, a problem not yet addressed is the definition of a notion of *interest* for the suggested queries.

1.1 Contributions

In this paper, we provide an algorithm which will present a recommend related queries to a query submitted to a search engine. These groups of related queries are created by running a clustering process over the queries and their associated information in the logs.

The clustering process is analyzed on the basis of a term-weight vector representation of queries. This is obtained from the aggregation of the term-weight vectors of the clicked URL's for certain query. Queries which are similar in terms of semantics may not share query-terms but they do share some terms in the documents selected by users. This is the problem that was appeared in previous work on query clustering i.e. to find semantically similar queries for clustering. Thus the framework presented over here avoids the problems of comparing and clustering sparse collection of vectors. Further, our query vectors can be clustered and manipulated similarly to traditional document vectors.

Here we also provide a relevance criterion that is to be followed to rank the suggested queries. We rank the queries according to two criteria: (a) keyword matching, that is the similarity of the queries to the input query (query submitted to the search engine); and (b) the support, which measures how much the results of the query have attracted the attention of users. It is important to have a measure of *support* for the recommended queries, because it is worthy to recommend queries that are useful to many users (searching for related information) in our context. The combination of these two measures (a) and (b) predicts the *interest* of a recommended query.

Finally, we present an experimental evaluation of the algorithm, using logs from a popular search engine for the Google.

2 Discovering Related Queries

The algorithm here only considers queries that appear in the query-log. A different *query session* is generated for each submission of the query in spite the fact that a single query (list of terms) may have been submitted to the search engine several times. In this paper, we use a simple notion of query session similar to the notion introduced by Wen *et al.* [6] which consists of a query, along with the URLs clicked in its answer.

```
QuerySession := (query,(clickedURL)*)
```

In this paper for improved versions of the algorithm, among other data we also considered a more detailed notion of query session which may analyze the rank of each clicked URL and the answer page in which the URL appears.

This algorithm operates in the following steps:

1. In the first phase queries along with the text of their clicked URL's are extracted from the Web log and are clustered. This is a preprocessing phase of the algorithm that can be conducted at periodical and regular intervals.
2. When a query is submitted to the search engine then this *input query* is searched in logs to find the cluster to which the input query belongs. After that we compute a rank score for each query present in the cluster. The method applied for computing the rank score is presented in next section.
3. Finally, the response constitutes the related queries that are returned ordered according to their rank score.

Interest is measured on the basis of this rank score of a related query and is obtained by combining the following notions:

1. **Similarity of the query (keyword matching).** The similarity of the query to the input query. It is measured using the notion of similarity introduced in Section 3.1.
2. **Support of the query.** This measures the relevance of a query in cluster. We measure the support of the query as the fraction of the documents returned by the query that captured the attention of users (clicked documents). It is estimated from the query log as well.

On considering the number of times the query has been submitted as the support of a query, we analyzed the logs in our experiments in which we found that popular queries are those whose answers are of little interest to users. In order to avoid and deal with this problem we define the support of a query as the fraction of clicks in answers of the query. As an example, the query *rental offices* has a low popularity (2.52%) in its cluster, but users in the cluster found this query very effective, as its support in Fig. 3 shows.

The rank score of the query can be evaluated by normalizing the similarity and support of a query and then linearly combining them. Another approach may consider to output a list of suggestions showing the two measures to users, and to let them tune the weight of each measure for the final rank.

3 Query Clustering

3.1 Query Similarity

To compute the similarity of two queries, we first have to build a term-weight vector for each query. Our vocabulary constitutes a set of all different words except the Stopwords(frequent words) in the clicked URLs. Now each term is weighted according to the number of occurrences and the number of clicks of the documents in which the term appears.

Given a query q, and a URL u, let Pop (q, u) be the popularity of u (fraction of clicks) in the answers of q. Let

Volume: 05, Issue: 03, Pages:1917-1921 (2013) ISSN : 0975-0290

Tf(t, u) be the number of occurrences of term t in URL u. We define a vector representation for q, q, where q[i] is the i^{th} component of the vector associated to the i^{th} term of the vocabulary (all different words), as follows:

$$q[i] = \sum_{URL\ u} \frac{Pop(q,u) \times Tf(t_i,u)}{\max_t Tf(t,u)} \quad (1)$$

where the sum ranges over all clicked URLs. Note that our representation changes the inverse document frequency by click popularity in the classical tf-idf weighting scheme.

Different notions of vector similarity (e.g., cosine function or Pearson correlation) can be applied over the proposed vectorial representation of queries. In this paper we use the cosine function, which considers two documents similar if they have similar proportions of occurrences of words (but could have different length or word occurrence ordering).

3.2 Computing the Clusters

Here we analyzed queries extracted from a 15-day query-log of the Google search engine. The log constitutes 6042 queries having clicks in their answers. The total clicks registered in log are 22190 and these clicks are over 18527 different URL's. Thus if we take an average then we found that users clicked 3.67 URL's per query.

The clusters are computed by successive calls to a k-means algorithm, using the CLUTO software package. We choose k-means algorithm for implementation as this algorithm is known for its simplicity and low computational cost as compared with other clustering algorithms. In addition, it has shown good quality performance for document clustering. For further details we refer the reader to [9].

The criterion function presented by common implementations of a k-means algorithm is used to measure quality of the resulting clusters [10]. This function evaluates the total sum of the similarities between the vectors and the centroids of the cluster that are assigned to. In k-means algorithm the number of clusters k is fixed for a single run. So, to determine the final number of clusters we have to perform successive runs of the algorithm. Fig. 1 shows the quality of the clusters found for different values of k (function criterion FC). The curve below (DIFF) shows the incremental gain of the overall quality of the clusters. We selected $k = 600$, for which we obtain a 0.6 average distance of each point to its cluster centroid. We ran the clustering algorithm on a Pentium IV computer, with CPU clock rate of 2.4 GHz, 512MB RAM, and running Windows XP. The algorithm took 64 minutes to compute the 600 clusters.

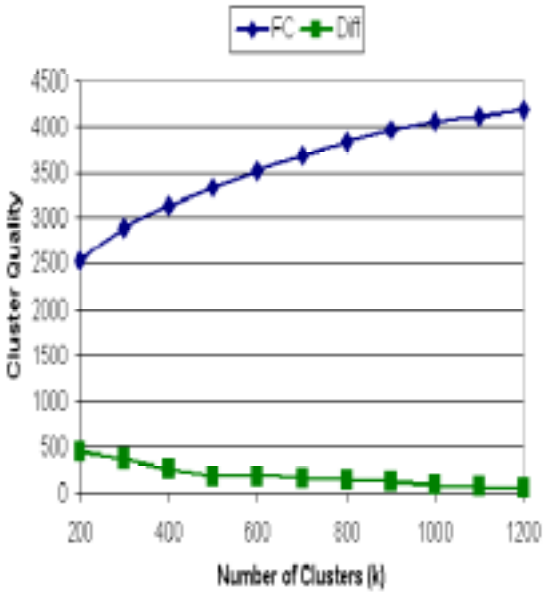

Fig.1. Cluster quality vs. number of clusters

4 Experimental Evaluation of the Algorithm

In our experiments we consider ten queries: (1) *theater (teatro);* (2) *rental apartments vi na del mar (arriendo de departamentos en vi˜na del mar);* (3) *chile rentals (arriendos chile);* (4) *recipes (recetas);* (5) *roads of chile (rutas de chile);* (6)*fiat;* (7) *maps of chile (mapas de chile);* (8) *resorts of chile (resorts de chile);* (9) *newspapers (diarios)* and (10) *tourism tenth region (turismo d´ecima regi´on).*

These ten queries were selected based on the probability distribution of the 6042 queries of the log. Here the original queries along with the results shown in this section are translated from Spanish to English. Fig. 2 shows the clusters to which the queries belong. Along with it, we show the cluster rank, the quality of each cluster (average internal similarity), the cluster size (number of queries that belongs to each cluster) and the set of feature terms that best describe each cluster. Right next to each feature term or we can say a keyword; there is a percentage that shows the cluster similarity that this keyword can explain.

Volume: 05, Issue: 03, Pages:1917-1921 (2013) ISSN : 0975-0290

Q	Cluster	ISim	Size	Query Selected	Descriptive Keywords Rank
q_1	15	0,98	8	theater	productions (18,4%)
					Campbell productions (7, 7%)
					dance (4,5%)
q2	81	0,709	15	rental apartments *Vi˜na del Mar*	real estate (21, 7%)
					property (17, 0%)
					used (11, 1%)
q3	124	0,618	9	*Chile* rentals	storehouse (5,3%)
					warehouses (4,6%)
					office (3, 0%)
q4	136	0,588	7	recipes	food (28,4%)
					soft drinks (9,4%)
					eggs (2, 2%)
q5	147	0,581	14	roads of *Chile*	maps (10,8%)
					springs (4,2%)
					ski (4, 0%)
q6	182	0,519	8	*Fiat*	spare parts (28, 2%)
					shock absorber (3,9%)
					mechanic (3, 1%)
q7	220	0,48 1	7	maps of *Chile*	maps (50,3%)
					geological (1, 1%)
					Mapcity (1,0%)
q8	306	0,420	11	resorts of *Chile*	hotels (69,2%)
					region (1, 4%)
					bay (0, 5%)
q9	421	0,347	7	newspapers	journal (25,6%)
					elmercurio (18,1%)
					estrategia (1, 9%)
q10	597	0,264	7	tourism tenth region	*Montt* (17,9%)
					Osorno (5, 5%)
					Chait'en (3, 7%)

Fig. 2. Clusters for the experiment.

Fig. 3 shows the ranking suggested to Query 3 *(chile rentals).* The second column shows the popularity of the queries in the cluster. The third column shows the support, and the last column illustrates the similarity of the queries. According to the similarity to the input query the queries are arranged. The figure shows that algorithm discovered semantically connected queries that are building upon different keyword. As an example, for a non-expert user the keyword *lehmann* may be unfamiliar for searching rental adds. However, this term refers to a rental agency having a significant presence in Web directories and ads of rentals in Chile. Notice that our algorithm found queries with related terms, some of which would be difficult to use for users.

Query	Pop. (%)	Support (%)	Similarity
rentals	23,74	0,24	0,998
real estate	1,44	0,1	0,9852
lehmann properties	0,72	0,1	0,963
properties	56,83	0,19	0,7203
parcel purchase	3,6	0,1	0,7089
rental offices	2,52	0,19	0,655
free advertisement	5,76	0,29	0,602
rental apartments	3,6	0,24	0,396

Fig. 3. Ranking of queries recommended to the query *Chile rentals*

In order to assess the quality of the results, we follow a similar approach to Fonseca *et al [4]*. The relevance of each query to the input query was judged by members of our department. They analyzed the answers of the queries and determined the URL's in the answers that are of interest to the input query. Our results are given in graphs showing precision vs. number of recommended queries. Fig. 4 shows the average precision for the queries considered in the experiments.

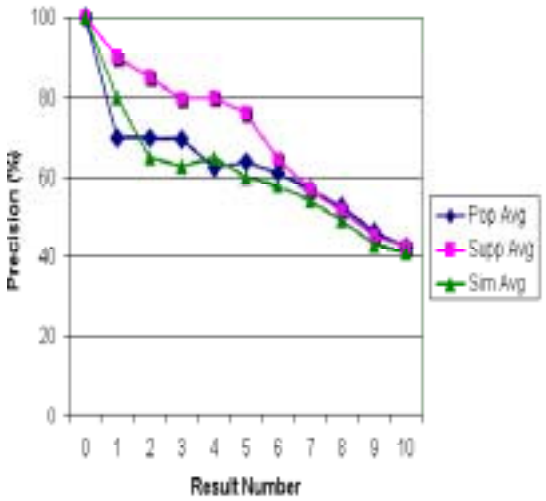

Fig.4. Average retrieval precision for the queries considered in the experiments.

The figure elaborates the precision of a ranking obtained using the similarity, support, and popularity of the queries. The graphs show that using the support measure, in average, we obtain a precision of 80% for the first 3 recommended queries. For both popularity and similarity, the precision decreases. Our results show that the similarity criterion for

Volume: 05, Issue: 03, Pages:1917-1921 (2013) ISSN : 0975-0290

ranking queries could be improved by considering how effective are queries in returning pages that are preferred by users, which is measured using our notion of support.

5 Conclusions

Here we have presented a technique for suggesting and recommending related queries based on a clustering process over data extracted from the query log. We are in process to perform more experiments with larger logs and considering more queries to improve the empirical evaluation of our approach. Along with it we are also trying to perform query expansion using the keywords associated to clusters. We also are working towards the improvement of the notion of support of a query. One way for doing so is to only consider clicks in the query answer to documents that are similar to the term-weight representation of the input query.

As future work we tend to improve the notion of interest of the recommended queries and to develop alternative notions of interest for the question recommender system. For an example, finding queries that share words however not clicked URL's. This may imply that the common words have totally different meanings if the text of the URL's is not shared. Thus we are able to sight polysemic words. On the other hand, if words don't seem to be shared and several terms within the URL's are shared, that will imply a semantic relation among words that can be stored in ontology.

References

[1] R. Baeza-Yates. Query usage mining in search engines. *Web Mining: Applications and Techniques, Anthony Scime, editor. Idea Group,* 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval (*Addison-Wesley, 1999) 75–79.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, Boston, MA USA, 2000.

[4] B. M. Fonseca, P. B Golgher, E. S. De Moura, and N. Ziviani. Using association rules to discovery search engines related queries. In *First Latin American Web Congress (LA-WEB'03)* ,November, 2003. Santiago, Chile.

[5] M. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *ACM SIGIR Forum, 32(1):5-17,* 1998.

[6] J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. In *Proc. at 10th International World Wide Web Conference,* pages 162–168. W3C, 2001.

[7] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with the local context analysis. *ACM Transaction of Information Systems,* 1(1 8):79–1 12, 2000.

[8] O. R. Zaiane and A. Strilets. Finding similar queries to satisfy searches based on query traces. In *Proceedings of the International Workshop on Efficient Web-Based Information Systems (EWIS),* Montpellier, France, September, 2002.

[9] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. In *SIAM Workshop on Clustering High-dimensional Data and its Applications*, 2002.

[10] Y. Zhao and G. Karypis. Criterion functions for document clustering. Technical report, University of Minnesota, MInneapolis, MN, 2002.