Query Recommendation using Query Logs in Search Engines

Ricardo Baeza-Yates[1], Carlos Hurtado[1], and Marcelo Mendoza[2]

[1] Center for Web Research
Department of Computer Science
Universidad de Chile
{rbaeza,churtado}@dcc.uchile.cl
[2] Department of Computer Science
Universidad de Valparaiso
marcelo.mendoza@uv.cl

Abstract. In this paper we propose a method that, given a query submitted to a search engine, suggests a list of related queries. The related queries are based in previously issued queries, and can be issued by the user to the search engine to tune or redirect the search process. The method proposed is based on a query clustering process in which groups of semantically similar queries are identified. The clustering process uses the content of historical preferences of users registered in the query log of the search engine. The method not only discovers the related queries, but also ranks them according to a relevance criterion. Finally, we show with experiments over the query log of a search engine the effectiveness of the method.

1 Introduction

A key factor for the popularity of today's Web search engines is the friendly user interfaces they provide. Indeed, search engines allow users to specify queries simply as lists of keywords, following the approach of traditional information retrieval systems [2]. Keywords may refer to broad topics, to technical terminology, or even to proper nouns that can be used to guide the search process to the relevant collection of documents.

Despite 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 formulate effective queries to search engines. One reason for this is the ambiguity that arise in many terms of a language. Queries having ambiguous terms may retrieve documents which are not what users are searching for. 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.

Users searching for the same information may phrase their queries differently. Often, users try different queries until they are satisfied with the results. In order to formulate effective queries, users may need to be familiar with specific terminology in a knowledge domain. This is not always the case: users may have little knowledge about

the information they are searching, and worst, they could not even be certain about what to search for. As an example, a tourist seeking for *summer rentals* ads in Chile may not know that the vast majority of such ads in the Web are for apartments in *Viña del Mar*, a popular beach in the central part of Chile. In contrast, local users may have the expertise to submit queries with the term *Viña del Mar*, when they are looking for a location to spend their vacations. The idea is to use these expert queries to help non-expert users.

In order to overcome these problems, some search engines have implemented methods to suggest alternative queries to users[3]. Their aim is to help the users to specify alternative related queries in their search process. Typically, the list of suggested queries is computed by processing the query log of the search engine, which stores the history of previously submitted queries and the URL's selected in their answers. A central problem that arises in this context is how to model the information needs associated to a query. Some models proposed in previous work (e.g., [3]) represent a query as the set of URL's clicked by users for the query. This approach have limitations when it comes to identifying similar queries, because two related queries may output different URL's in the first places of their answers, thus inducing clicks in different URL's. In addition, as an empirical study shows [1], the average number of pages clicked per answer is very low (around 2 clicks per query). Our data shows the same.

As in traditional document retrieval, in query recommendation one may expect that the ordering in which the queries are returned to the user plays a central role in the quality of the service, 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 present an algorithm to recommend related queries to a query submitted to a search engine. Groups of related queries are found by running a clustering process over the queries and their associated information in the logs.

The clustering process is based on a term-weight vector representation of queries, obtained from the aggregation of the term-weight vectors of the clicked URL's for the query. Semantically similar queries may not share query-terms but they do share terms in the documents selected by users. Thus our framework avoids the problems of comparing and clustering sparse collection of vectors, in which semantically similar queries are difficult to find, a problem that appears in previous work on query clustering. Further, our query vectors can be clustered and manipulated similarly to traditional document vectors.

We provide a relevance criterion to rank the suggested queries. We rank the queries according to two criteria: (a) the similarity of the queries to the input query (query submitted to the search engine); and (b) the support, which measures how much the answers of the query have attracted the attention of users. It is important to have a measure of *support* for the recommended queries, because queries that are useful to many users (searching for related information) are worth to be recommended in our

[3] Some search engines (e.g., Lycos, Altavista, AskJeeves) provide query recommendations to users, however there is not much public information on the methods they use to do so.

context. The combination of measures (a) and (b) defines the *interest* of a recommended query.

Finally, we present an experimental evaluation of the algorithm, using logs from a popular search engine for the Chilean Web (TodoCl.cl).

1.2 Related Work

Baeza-Yates [1] presents a survey on the use of Web logs to improve different aspects of search engines.

Wen et al. [6] propose to cluster similar queries to recommend URLs to frequently asked queries of a search engine. They use four notions of query distance: (1) based on keywords or phrases of the query; (2) based on string matching of keywords; (3) based on common clicked URLs; and (4) based on the distance of the clicked documents in some pre-defined hierarchy. Befferman and Berger [3] also propose a query clustering technique based on distance notion (3). Notions (1)-(3) are difficult to deal with in practice, because distance matrices between queries generated by them from real query logs are very sparse, and many queries with semantic connections appear as orthogonal objects in such matrices. Ad-hoc clustering algorithms are needed to deal with this problem. Notion (4) needs a concept taxonomy and requires the clicked documents to be classified into the taxonomy as well.

Fonseca *et al* [4] present a method to discover related queries based on association rules. Here queries represent items in traditional association rules. The query log is viewed as a set of transactions, where each transaction represent a *session* in which a single user submits a sequence of related queries in a time interval. Their notion of query session is different than the notion we use in this paper. The method shows good results, however two problems arise. First, it is difficult to determine sessions of successive queries that belong to the same search process; on the other hand, the most interesting related queries, those submitted by different users, cannot be discovered. This is because the support of a rule increases only if its queries appear in the same query session, and thus they must be submitted by the same user.

Zaiane and Strilets [8] present a method to recommend queries based on seven different notions of query similarity. Three of them are mild variations of notion (1) and (3). The remainder notions consider the content and title of the URL's in the result of a query. Their approach is intended for a meta-search engine and thus none of their similarity measures consider user preferences in form of clicks stored in query logs.

Another approach adopted by search engines to suggest related queries is *query expansion* [2, 7]. The idea here is to reformulate the query such that it gets closer to the term-weight vector space of the documents the user is looking for. Our approach is different since we study the problem of suggesting related queries issued by other users and query expansion methods construct artificial queries. In addition, our method may recommend queries that are related to the input query but may search for different issues, thus redirecting the search process to related information of interest to previous users.

Outline The remainder of this paper is organized as follows. In Section 2 we present the method proposed for computing and ranking related queries. Section 3 describes

and shows the results of the query clustering process. In Section 4 we present the experimental evaluation of the method. Finally, in Section 5 we conclude and outline some prospects for future work.

2 Discovering Related Queries

Our algorithm considers only queries that appear in the query-log. A single query (list of terms) may be submitted to the search engine several times, and each submission of the query induces a different *query session*. 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.

$$\texttt{QuerySession} := (\texttt{query}, (\texttt{clickedURL})^*)$$

A more detailed notion of query session may consider the rank of each clicked URL and the answer page in which the URL appears, among other data that can be considered for improved versions of the algorithm we present in this paper.

The query recommender algorithm operates in the following steps:

1. Queries along with the text of their clicked URL's extracted from the Web log are clustered. This is a preprocessing phase of the algorithm that can be conducted at periodical and regular intervals.
2. Given an *input query* (i.e., a query submitted to the search engine) we first find the cluster to which the input query belongs. Then we compute a rank score for each query in the cluster. The method for computing the rank score is presented next in this section.
3. Finally, the related queries are returned ordered according to their rank score.

The rank score of a related query measures its interest and is obtained by combining the following notions:

1. **Similarity of the query**. 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 is a measure of how relevant is the query in the 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.

One may consider the number of times the query has been submitted as the support of a query. However, by analyzing the logs in our experiments we found popular queries whose answers are of little interest to users. In order to avoid 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 Figure 3 shows.

The similarity and support of a query can be normalized, and then linearly combined, yielding the rank score of the query. 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

In order to compute the similarity of two queries, we first build a term-weight vector for each query. Our vocabulary is the set of all different words in the clicked URLs. *Stopwords* (frequent words) are eliminated from the vocabulary considered. 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 $\texttt{Pop}(q, u)$ be the popularity of u (fraction of clicks) in the answers of q. Let $\texttt{Tf}(t, u)$ be the number of occurrences of term t in URL u. We define a vector representation for q, $\boldsymbol{q}$, where $\boldsymbol{q}[i]$ is the i-th component of the vector associated to the i-th term of the vocabulary (all different words), as follows:

$$\boldsymbol{q}[i] = \sum_{URLu} \frac{\texttt{Pop}(q, u) \times \texttt{Tf}(t_i, u)}{\max_t \texttt{Tf}(t, u)} \tag{1}$$

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

We considered queries extracted from a 15-day query-log of the Todocl search engine. The log contains 6042 queries having clicks in their answers. There are 22190 clicks registered in the log, and these clicks are over 18527 different URL's. Thus in average users clicked 3.67 URL's per query.

We compute the clusters by successive calls to a k-means algorithm, using the CLUTO software package[4]. We chose an implementation of a k-means algorithm for the simplicity and low computational cost of this approach, compared with other clustering algorithms. In addition, the k-means implementation chosen has shown good quality performance for document clustering. We refer the reader to [9] for details.

The quality of the resulting clusters is measured using a criterion function, adopted by common implementations of a k-means algorithm [10]. The function measures the total sum of the similarities between the vectors and the centroids of the cluster that are assigned to. Since in a single run of a k-means algorithm the number of clusters k is fixed, we determine the final number of clusters by performing successive runs of the algorithm. Figure 1 shows the quality of the clusters found for different values

[4] CLUTO is a software package developed at the University of Minnesota that provides algorithms for clustering collections of documents in high-dimensional vectorial representations. For further information see http://www-users.cs.umn.edu/ karypis/cluto/.

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.

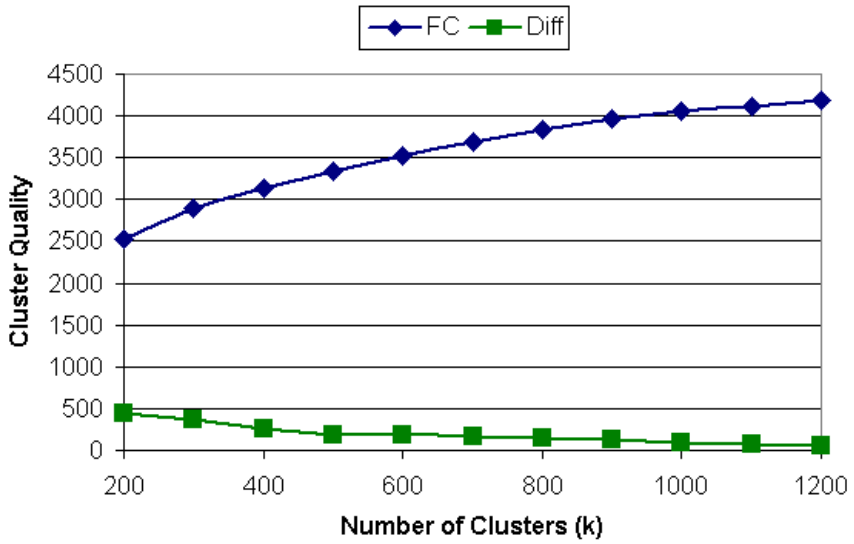

Fig. 1. Cluster quality vs. number of clusters.

4 Experimental Evaluation of the Algorithm

In our experiments we consider ten queries: (1) *theater* (*teatro*); (2) *rental apartments viña del mar* (*arriendo de departamentos en viña 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 décima región*).

The ten queries were selected following the probability distribution of the 6042 queries of the log. The original queries along with the results shown in this section are translated from Spanish to English. Figure 2 shows the clusters to which the queries belong. In addition, 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, there

is a number that is the percentage of the within cluster similarity that this particular feature term can explain.

Q	Cluster Rank	ISim	Size	Query Selected	Descriptive Keywords
q_1	15	0,98	8	theater	productions $(18, 4\%)$ *Campbell* productions $(7, 7\%)$ dance $(4, 5\%)$
q_2	81	0,709	15	rental apartments *Viña del Mar*	real estate $(21, 7\%)$ property $(17, 0\%)$ used $(11, 1\%)$
q_3	124	0,618	9	*Chile* rentals	storehouse $(5, 3\%)$ warehouses $(4, 6\%)$ office $(3, 0\%)$
q_4	136	0,588	7	recipes	food $(28, 4\%)$ soft drinks $(9, 4\%)$ eggs $(2, 2\%)$
q_5	147	0,581	14	roads of *Chile*	maps $(10, 8\%)$ springs $(4, 2\%)$ ski $(4, 0\%)$
q_6	182	0,519	8	*Fiat*	spare parts $(28, 2\%)$ shock absorber $(3, 9\%)$ mechanic $(3, 1\%)$
q_7	220	0,481	7	maps of *Chile*	maps $(50, 3\%)$ geological $(1, 1\%)$ *Mapcity* $(1, 0\%)$
q_8	306	0,420	11	resorts of *Chile*	hotels $(69, 2\%)$ region $(1, 4\%)$ bay $(0, 5\%)$
q_9	421	0,347	7	newspapers	journal $(25, 6\%)$ *el mercurio* $(18, 1\%)$ *estrategia* $(1, 9\%)$
q_{10}	597	0,264	7	tourism tenth region	*Montt* $(17, 9\%)$ *Osorno* $(5, 5\%)$ *Chaitén* $(3, 7\%)$

Fig. 2. Clusters for the experiment.

Figure 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 depicts the similarity of the queries. The queries are ordered according to their similarity to the input query. The figure shows that algorithm discovered semantically connected queries that are build 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 asses the quality of the results, we follow a similar approach to Fonseca *et al* [4]. The relevance of each query to the input query were 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. Figure 4 shows the average precision for the queries considered in the experiments.

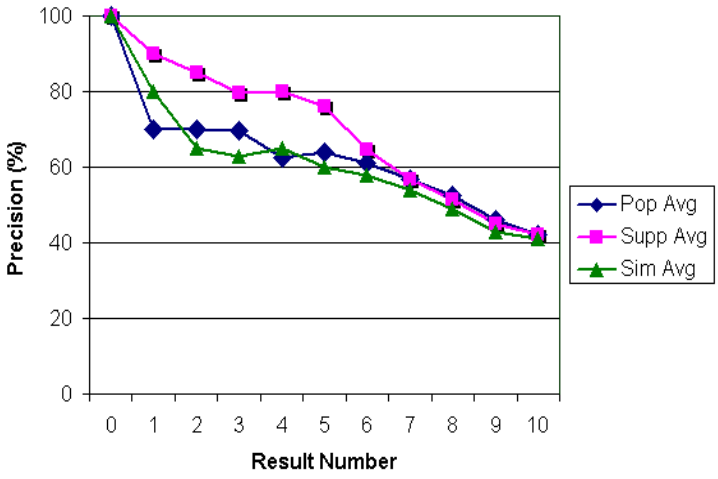

Fig. 4. Average retrieval precision for the queries considered in the experiments.

The figure shows 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 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 Conclusion

We have presented a method for suggesting related queries based on a clustering process over data extracted from the query log. We are currently preforming experiments with larger logs and considering more queries, to improve the empirical evaluation of our approach. In addition, we are trying to do query expansion (which can be seen as automatic query recommendation) using the keywords associated to clusters. We also consider to improve the notion of support of a query. One direction 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 consider to improve the notion of interest of the suggested queries and to develop other notions of interest for the query recommender system. For example, finding queries that share words but not clicked URL's. This might imply that the common words have different meanings if the text of the URL's also are not shared. Hence we can detect polysemic words. On the other hand, if words are not shared and the many terms in the URL's are shared, that may imply a semantic relation among words that can be stored in an ontology.

Acknowledgments This research was supported by Millennium Nucleus, Center for Web Research (P01-029-F), Mideplan, Chile. M.Mendoza was partially supported by the Universidad de Valparaíso DIPUV-03 project.

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*, chapter 3, pages 75–79. Addison-Wesley, 1999.
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(18):79–112, 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.