Improving Search Engines by Query Clustering

Ricardo Baeza-Yates* and Carlos Hurtado
Center for Web Research, Department of Computer Science, Universidad de Chile, Chile.
E-mail: {rbaeza, churtado}@dcc.uchile.cl

Marcelo Mendoza
Department of Computer Science, Universidad de Valparaíso, Chile. E-mail: marcelo.mendoza@uv.cl

In this paper, we present a framework for clustering Web search engine queries whose aim is to identify groups of queries used to search for similar information on the Web. The framework is based on a novel term vector model of queries that integrates user selections and the content of selected documents extracted from the logs of a search engine. The query representation obtained allows us to treat query clustering similarly to standard document clustering. We study the application of the clustering framework to two problems: relevance ranking boosting and query recommendation. Finally, we evaluate with experiments the effectiveness of our approach.

Introduction

Nowadays, search engines are indispensable tools for finding information on the Web. However, the amount of information available to us in the Web is literally exploding, and thus search engines technology is continuously being pushed to the limit. Several new techniques have emerged to improve search engines, some of which have focused on the analysis of query logs. Query logs register the history of queries submitted to the search engine and the pages selected after a search, among other data. Analyzing query logs has a broad impact in different applications for Web searching such as Web findability, document and index caching, and Web crawling (Correia-Saravia et al., 2001).

The search for certain groups of queries capturing common sets of preferences and information needs has been a recent trend in query log analysis (Beeferman & Berger, 2000; Zhang & Dong, 2002; Wen, Nie, & Zhang, 2001). Groups of related queries can be discovered by clustering queries using their related data in query logs. The clusters can then be used to improve search engines in several aspects. For example, search engines such as Yahoo! and Ask recommend related queries to the query submitted by a user. The related queries are computed by running query clustering processes. However, there is not much public information on the methods they use to do so. Little formal research has been done on the different problems that arise when doing query clustering.

The central problem that arises is how to represent the information needs represented by a query. Queries themselves, as lists of keywords, are not always good descriptors of the information needs of users. One reason for this is the ambiguity carried by polysemic words. 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 and associates (Jansen, Spink, Bateman, & Saracevic, 1998), concluded that most queries are short (around two terms per query) and imprecise. 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 at worst, they may not even be certain about what to search for.

The definition of an adequate way to model semantics of queries and similarity for queries is still an open problem. Query logs can heavily help in doing so. Previous works have proposed models to represent information needs related to queries using distance functions over metric spaces (Paredes & Chávez, 2005) or using data based on logs. We use this term to refer to the successive submissions of a query (usually by different users) in a period of time, along with the sets of uniform resource locators (URLs) selected for them. For example, Beeferman and coworkers (Beeferman & Berger, 2000) represent the semantic of a query as the set of URLs selected in its answer list. This approach has limitations when it comes to identifying similar queries, because two related queries may output different

*Now at Yahoo! Research

Accepted January 4, 2007

 ● Published online 6 September 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.20627

URLs in the first places of their answers, thus inducing clicks in different URLs. In addition, as an empirical study shows (Silverstein, Henzinger, Marais, & Moricz, 1999), the average number of pages clicked per answer is very low (around two clicks per query).

Another inherent problem of the use of query logs is that the clicked URLs are biased to the ranking algorithm of the search engine. The number of preferences on a particular answer depends on the position of the page in the answer list. Each successive page in the list is less likely to be selected by users. In fact, several studies show that the average number of pages of 10 answers seen is less than 2 (Jansen et al., 1998). An adequate modeling for the preferences of users and for the semantics of queries should incorporate a method for reducing the bias caused by the current ranking produced by the search engine.

Contributions

In this paper we present a new framework for clustering queries. The clustering process is based on a term-weight vector representation of queries, obtained by aggregating the term-weight vectors of the selected URLs for the query. The construction of the vectors includes a technique to reduce and adjust the bias of the preferences to URLs in answers. The vectorial representation leads to a similarity measure in which the degree of similarity of two queries is given by the fraction of common terms in the URLs clicked in the answers of the queries. This notion allows one to capture semantics connection between queries having different query terms.

Because the vectorial representation of a query is obtained by aggregating term-weight vectors of documents, our framework avoids the problems of comparing and clustering sparse collection of vectors, a problem that appears in previous work on query clustering (see the section Related Work). The central idea in our vectorial representation of queries is to allow manipulating and processing queries in the same fashion as document are handled in traditional information retrieval (IR) systems, therefore allowing a fully symmetric treatment of queries and documents. In this form, query clustering turns into a similar problem to document clustering.

We present two applications of our clustering framework: query recommendation and answer ranking. The use of query clustering for query recommendation has been suggested by Beeferman and Berger (Beeferman & Berger, 2000), however as far as we know, there is no formal study of the problem. Regarding answer ranking, we are not aware of formal research on the application of query clustering to this problem. For both applications we provide a criterion to rank the suggested URLs and queries that combines the *similarity* of the query (respectively, URL) to the input query and the *support* of the query (respectively, URL) in the cluster. The support measures how much the recommended query (resp., URL) has attracted the attention of users. The rank estimates the *interest* of the query (resp., URL) to the user who submitted the input query. It is important to include a measure of *support* for the recommended queries (resp., URLs), because queries (URLs) that are useful to many users are worth being recommended in our context.

Finally, we present an experimental evaluation of the clustering framework algorithms, using the logs from a popular vertical search engine.

This article is a substantially improved version of earlier work (Baeza-Yates et al., 2004a, 2004b). Most of the sections have been reviewed and completed. In addition, this version presents a method to reduce the position bias in the selection of documents inside the clustering process and we perform experiments in the section Experimental Results with a 6-month log of the search engine (in the first versions of this work the experiments were done with a 15-days log). We present an evaluation of 30 queries (instead of 10) in the sections Query Recommendation and Answer Ranking. Further we compare the query similarity function we propose with two standard query similarity functions (mentioned in the next section), namely, similarity based on query terms (cosine of query term vectors) and similarity based on clicked URLs (cosine of vectors of clicked URLs). We present a study of the correlation between these measures and evaluate them with a list of pairs of quasi-synonymous queries from the log.

Related Work

Page ranking algorithms based on links (e.g., Most-Cited [Yuwono & Lee, 1996], PageRank [Brin & Page, 1998], and HITS [Kleinberg, 1998]) have been described and classified by Lawrence and Giles (1999). These algorithms take advantage of the fact that links in the Web usually represent human annotations about the quality of pages. However, Web links have some drawbacks. First, they are in general made by the experts who design Web pages and place links in the Web. Not always are the recommendations of experts as good as one may think. Frei and Schäuble (1991) argue that opinions of common users are more consistent than the ones of experts. On the basis of empirical evidence, Lesk and Salton (1968) conclude that when the opinions of common users are context independent, they do not have the bias inherent in a group of experts. Second, links do not necessarily reflect the dynamism of human preferences in the Web. It is natural for older sites to have more links to them than new sites (Baeza-Yates, Saint-Jean, & Castillo, 2002). Web sites that become obsolete for users may keep links and high ranks in search engines for long periods. Third, it is not easy to capture semantically meaningful connections between pages from links. Web resources may not be oriented to a single subject, in which case links represent ambiguous connections whose semantic value may not be clear.

Some ranking algorithms have included the analysis of query logs. Click-through data is also used by search engines to evaluate the quality of changes to the ranking

DOI: 10.1002/asi

algorithm by tracking them. DirectHit[1] used previous session logs of a given query to compute ranks based on the popularity (number of clicks) of each URL that appears in the answer of the query. This approach only works for queries that are frequently formulated by users, because less common queries do not have enough clicks to allow significant ranking scores to be calculated. For less common queries, the direct hit rating provides a small benefit.

Zhang and Dong (2002) propose the Matrix Analysis on Search Engine Log (MASEL) algorithm, which uses search engine logs to improve ranking. Clicks are considered positive recommendations on pages. The basic idea is to extract from the logs relationships of users, queries, and pages. These relationships are used to estimate the quality of answers, based on the quality of related users and queries. The approach relies on the identification of users of a search engine, a task difficult to achieve in practice. Our approach instead focuses on queries by aggregating user preference into queries, and then into clusters of queries.

There is also recent related work on query clustering, some approaches also consider data in query logs. Wen and associates (2001) propose clustering 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 predefined hierarchy. Befferman and Berger (2000) also propose a query clustering technique based on common clicked URLs. From a study of the log of a popular search engine, Jensen and colleagues (1998) concluded that most queries are short (around 2 terms per query) and imprecise, and the average number of pages clicked per answer is very low (around 2 clicks per query). Thus, notions (1)–(3) are difficult to deal with in practice, because distance matrices between queries generated by them are very sparse. Notion (4) needs a concept taxonomy and requires the clicked documents to be classified into the taxonomy as well.

Fonseca, Golgher, DeMoura, & Ziviani (2003) present a method to discover related queries on the basis of 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. 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 (2002) present a method to recommend queries on the basis of seven different notions of query similarity. Three of them are mild variations of notions (1) and (3). The other notions consider the content and title of the URLs in the result of a query. Their approach is intended for a metasearch engine and thus none of their similarity measures considers user preferences in the form of clicks stored in query logs. Another approach adopted by search engines to suggest related queries is *query expansion* (Baeza-Yates & Ribeiro-Neto, 1999; Xu & Croft, 2000). The basic 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. The second section is Query Clustering Framework. In Applications of Query Clustering we present the algorithms for discovering related queries and ranking answers. Experimental Results shows the experimental evaluation of the algorithms. Finally, the section Conclusion outlines some prospects for future work.

Query Clustering Framework

The available data are a set of user logs from which we extract query sessions. Following Wen and Coworkers (2001) a *query session* consists of one query, a list of URLs, and the URLs the user clicked on.

```
querySession := query rank (clickedURL)*
```

Figure 1 shows the relationships between the different entities that participate in the process induced by the use of a

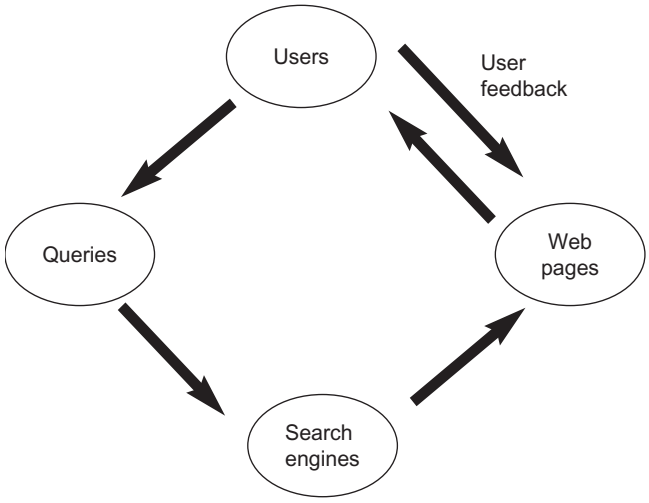

FIG. 1. Relationships of entities in query logs.

[1] Now part of Ask Jeeves http://www.askjeeves.com.

DOI: 10.1002/asi

search engine. In this paper, we focus on the relationship between queries, which is defined using the preferences/feedback of the user about Web pages in answers of the query. The relationship between queries is obtained by using the content of selected Web pages, which is indicated by the arrow from Web pages to queries.

Vectorial Representation of Queries

We next present the term-weight vector model for a query. 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 $\text{Pop}(u, q)$ be the number of clicks to u in the sessions related to query q. Let $\text{Tf}(t, u)$ be, as usual, the number of occurrences of term t in the excerpt of the URL u. *Stopwords* are eliminated from the vocabulary considered. We define the *vectorial representation of a query*, $\vec{q}$, as follows:

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

That is, Pop plays the role of *Idf* in the well-known tf-idf weighting scheme for the vector model.

We measure the similarity of two queries as the similarity of their vectors. Different notions of similarity (e.g., cosine function or Pearson correlation) can be applied over the vectorial representation of queries. In this paper we use the standard cosine function between two term vectors.

Our notion of query similarity has several advantages. First, it is simple and easy to compute. On the other hand, it allows one to relate queries that happen to be worded differently but stem from the same information need. Therefore, semantic relationships of queries are captured. Another important advantage is that it avoids sparse similarity matrices, which are usually generated by using previous notions of query similarity (see the section Related Work). As in our vectorial representation of queries we may view queries simply as documents, we can compare queries and documents using standard similarity functions for documents, as is the case in many IR systems. This is important for the applications we present in Applications of Query Clustering.

Vectorial Representation With Unbiased Popularity

Assume that users submit queries and have plenty of time to choose, among the whole set of URLs in the Web, the URLs that they like as answers to a query. In such a process we can easily estimate the relevance of a page. However, we need to estimate relevance from the query log. The main problem in doing so lies in the fact that URLs are returned by search engines in some ordering, and their preferences, as registered in the log, are biased against that ordering. In the remainder of the section we present a method to estimate the relevance of a page to a query from the clicks of the page obtained from the logs.

First we define the following random process. When a user submits a query q, the URL u is returned at a position r in the answer ranking. Then the user with some probability denoted $P_{visit}(X \geq r)$ skips the process at some URL greater then r. If the user reaches the position r, then he/she selects the URL u with probability $P_{likes}(u|q)$; otherwise he/she does not select the URL u.

Given a query q and URL u, we denote by $\text{Rank}(u, q)$ the current rank of u in the answers of q. Let us denote by $P_{rank}(u|q)$ the probability a user selects u, given that she/he has submitted the query q, considering that u appeared in position $\text{Rank}(u, q)$ in the answer ranking. We have

$$P_{logs}(u|q) = P_{visit}(X \geq \text{Rank}(u, q))P_{likes}(u|q).$$

Thus, in order to estimate $P_{likes}(u|q)$, we only need to estimate $P_{visit}(X > r)$, for each r. We posit that each successive position in the rank is less likely to be reached by users, with a power-law decay. We use a power law because this is the distribution that appears in incoming links (Broder et al. 2000) as well as in term frequency. In addition, the distribution of ranking of clicks in our data follows a power law, although there are discontinuities due to the 10-answer-per-page output. More precisely, since $P_{visit}(x)$ gives the probability a user visit is greater than or equal to x, it can be modeled using the following Pareto distribution:

$$P_{visit}(X \geq x) = \frac{1}{x^b} \quad (2)$$

Thus, we have

$$P_{likes}(u|q) = P_{rank}(u|q)(\text{Rank}(u|q))^b.$$

The cummulative density function (CDF) of the Pareto distribution is $P_{visit}(X < r) = 1 - (1/r)^b$, and its probability density function (PDF) is $p_{visit}(X = r) = br^{-(b+1)}$, which is a power-law distribution. The latter represents the probability that user exits the process exactly at the position x. It remains to estimate b. We do so as follows. Given a query session we assume that the user exits at the largest position in the answer list of a clicked URL. Then we fit a power-law distribution to the histogram for the frequencies of estimated exit positions. Figure 2 shows a plot of the logarithm for the rank of last URLs clicked in query sessions versus the logarithm for the number of users. We fitted a power-law distribution over these data, which leads to $b = 1.725$.

Finally, the adjusted popularity is as follows:

$$\text{AdjPop}(u, q) = \text{Pop}(u, q)(\text{Rank}(u, q))^b \quad (3)$$

In the unbiased similarity function, we replace $\text{Pop}(u, q)$ by $\text{AdjPop}(u, q)$ in the vectorial representation given in formula 1.

DOI: 10.1002/asi

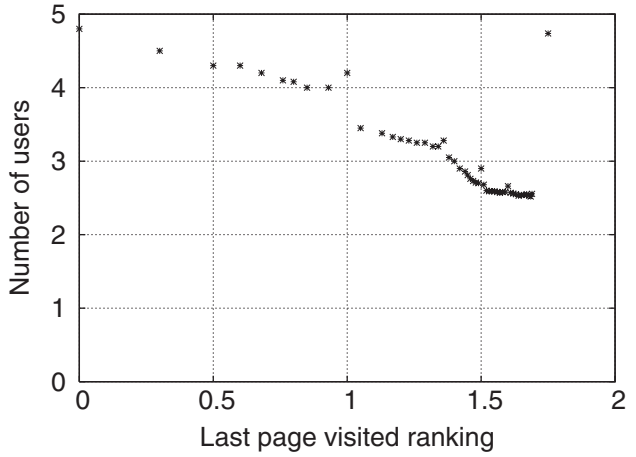

FIG. 2. Plot of the logarithm for the rank of last URLs clicked in query sessions versus the logarithm for the number of users.

Clustering Algorithm

Our vectorial representation of queries allows us to view the problem of query clustering as a document clustering problem. Plenty of research has been conducted in document clustering. We implement the clustering process using k-means algorithm provided by CLUTO,[2] a tool designed for document clustering. The CLUTO tool has proved to be very effective for clustering collections of documents (Zhao & Karypis, 2004); thus it also applies to clustering queries in our framework.

Although the clustering is performed offline, scalability of the algorithm is an issue. Among clustering approaches, centroid-based algorithms such as k-means achieve better performance. Therefore, we chose a k-means algorithm for the simplicity and low computational cost of this approach. The k-means algorithm operates as follows: initially, it randomly selects a set of objects (in our case, queries) that act as seed centroids of initial clusters. During each iteration the objects are visited in random order; each object is moved to the cluster whose centroid is closest to the object, whenever this operation leads to an improvement of the overall clusters quality. This process is repeated until no further improvement is achieved. Each run of the algorithm computes k clusters. Thus in order to determine an adequate value of k we run the algorithm several times.

For the applications of the clustering process (see Applications of Query clustering), it is of central importance to obtain strongly similar queries inside clusters. Intracluster similarity is more important for page ranking than query recommendation, since all the queries in the clusters are regarded as a single query and assumed to be used to search for the same set of documents. In contrast to other clustering approaches, intercluster dissimilarity is not a goal in our setting. Therefore, we measure the quality of the clusters using the total sum of the similarities between the vectors and the centroids (Zhao & Karypis, 2004).

Let C_r be a cluster found in a k-way clustering process ($r \in 1..k$), and let c_r be the centroid of C_r. The criterion function I is defined as

$$I = \sum_{r=1}^{k} \sum_{v_i \in C_r} sim(v_i, c_r), \tag{4}$$

where centroid c_r of a cluster C_r is defined as $\Sigma_{v_i \in C_r} v_i / |C_r|$.

The number of clusters is not predefined and is determined by the required quality of the clustering solution. That is, the number of clusters is such that the quality of the clustering solution is greater than a given quality threshold.

Applications of Query Clustering

We view query clustering as a fundamental technique underlying applications for query recommendation and document ranking based on a technique for building recommender systems called *collaborative filtering* (Breese, Heckerman, & Kadie, 1998).

The task of collaborative filtering is to predict the utility of items to a particular user, called the *active user*, on the basis of a database of votes from a sample or population of other users. Given a user searching for information, the idea is first to find similar users (via a k-neighborhood search or clustering process) and then suggest items preferred by users similar to the active user.

Since users cannot be identified in search engines, we aggregate them in queries, i.e., sets of users searching for similar information, by means of a clustering process. Therefore, the active user in our context is the input query. The items are respectively, Web pages and queries in the answer ranking and query recommendation algorithms.

We next sketch the two types of applications.

Answer Ranking

Our ranking algorithm considers only queries that appear in the query log. The algorithm operates in the following two phases:

1. Preprocessing phase at periodical and regular intervals:
   - Queries and clicked URLs are extracted from the Web log and clustered by using the text of all the clicked URLs.
   - For each cluster C_i, compute and store the following; a list Q_i containing queries in the cluster and a list U_i containing the k-most popular URLs in C_i, along with their popularity (number of clicks in the query log).

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

DOI: 10.1002/asi

2. Online searching phase: Given a query q, if that query appears in the stored clusters, we find the cluster C_i to which the query belongs. Then the answer to q are the URLs that have been selected for the queries, ordered according to a rank score that considers two criteria: similarity and support.

The similarity and support are defined as follows:

- **Document similarity.** The similarity of the page to the input query, measured by using the notion of similarity introduced in Vectorial Representation of Queries.
- **Document support.** It measures the relevance or weight of the URL in the cluster. This is estimated as the adjusted popularity of the URL in the cluster.

The rank score of a URL u is

$$\text{Rank}(u) = \text{Sim}(q, u) \times \sum_{q_i \in C} \text{AdjPop}(u, q_i) \qquad (5)$$

This ranking can then be used to boost the original ranking algorithm, using a linear combination of the two ranks.

$$\text{NewRank}(u) = \beta\ \text{OrigRank}(u) + (1 - \beta)\text{Rank}(u) \qquad (6)$$

Although it is true that not all clicks are relevant to a query, during the clustering process those URLs in most cases will be outliers or fall into bad clusters. Then, those *wrong* clicks will not be used.

Query Recommendation

The query recommender algorithm operates in the following steps:

1. Preprocessing phase similar to the answer ranking algorithm.
2. Online searching phase: Given an input query (i.e., a query submitted to the search engine) we first find the cluster to which the query belongs. Then we compute a rank score for each query in the cluster based on notions of similarity and support. 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:

- **Query similarity.** The similarity of the query to the input query, measured using the notion of similarity introduced in Vectorial Representation of Queries.
- **Query support.** This is a measure of the relevance of the query in the cluster. In order to measure support, one may consider the number of times the query has been submitted as 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 prevent this problem 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).

Finally, the rank score of a query is obtained by aggregating the adjusted popularity and similarity (to the input query) of the URLs in the answer of the query. Let q be the input query, let q_i be a query in the cluster C, and let U be the set of URLs in C, then the rank of q_i is

$$\text{Rank}(q_i) = \sum_{u \in U} \text{Sim}(q, u) \times \text{AdjPop}(u, q_i) \qquad (7)$$

Experimental Results

In order to evaluate our method, we performed experiments over the logs of a vertical search engine called TodoCl (www.todocl.cl). TodoCl mainly covers the .cl domain and some pages included in the .com domain and .net domains that are hosted by Chilean Internet service providers (ISPs). Currently the search engine collects more than 3 million Web pages and receives approximately 50,000 visits per day.

We ran experiments over a single dataset. The dataset is a 6-month log that contains 127,642 queries over 245,170 sessions. We selected the 30,363 queries with more the one session, which yield 147,891 sessions. These sessions have 431,432 clicks in total, which are more than 131,924 documents.

The following section shows experiments for the query clustering method proposed. The section Comparison of Query Distance Functions shows experiments that compare the distance functions we propose with other distance functions proposed in related work. In Evaluation of Applications, we select a clustering solution and use it to evaluate the answer ranking and query recommendation applications using 30 queries selected from the dataset.

Clustering Process

In this section, we evaluate the clustering process for the vectorial representations of queries given in Vectorial Representation Queries. After removing stopwords we reach 98,370 terms for the vectorial representation of the queries. We consider the cases with and without bias reduction. Figure 3 shows the quality of the clusters found for different values of k (recall that k is the number of clusters found). The figure displays the clustering quality for the similarity functions given in the sections Vectorial Representation of Queries (labeled excerpt term) and Vectorial Representation With Unbiased Popularity (labeled unbiased excerpt term). The curves of Figure 3 are not comparable since they are based on different similarity functions. However, the curves show a similar behavior, where a plateau is reached around $k = 600$ (number of clusters). Table 1 shows statistics for the clusters found for $k = 600$ for the 6-month log. The table shows an overall cluster quality (equation 4) of 14,900, and an average internal similarity between queries in each cluster of 0.2596, for the clustering based on excerpt terms. For the clustering based on unbiased excerpt terms, the figure shows an overall cluster quality of 14,900 and average internal similarity between queries in each cluster of 0.2614.

DOI: 10.1002/asi

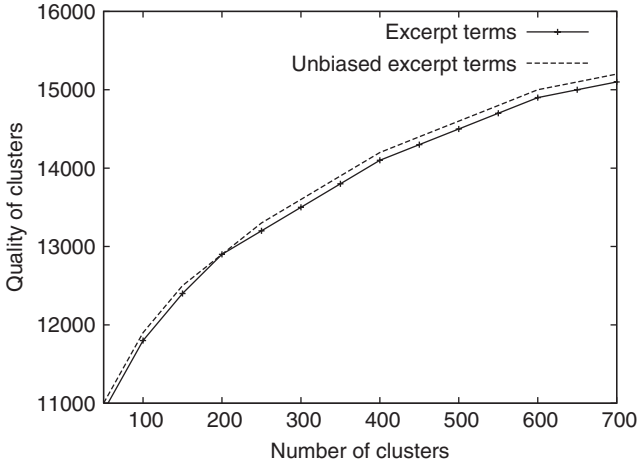

FIG. 3. Quality of clusters versus number of clusters.

TABLE 1. Statistics for the clusters found for $k = 600$.

	Excerpt terms	Unbiased excerpt terms
Overall cluster quality	14,900	14,900
Internal similarity (avg.)	0.2596	0.2614
Internal similarity (St.dev.)	0.0678	0.0678
External similarity (avg.)	0.0512	0.0514
External similarity (St.dev.)	0.0186	0.0188

Figures 4(a) and (b) show histograms for the number of queries per cluster when $k = 600$ for clustering solutions obtained without and with unbiasing, respectively. For both solutions, the average number of queries per cluster is 51, with standard deviations of 34 and 35, respectively. Over the 1200 clusters included in both solutions, there are only 121 clusters with fewer than 20 queries. Moreover, all the clusters have more than 10 queries.

We ran the algorithms on a Pentium IV computer, with central processing unit (CPU) clock rate of 2.4 GHz, 1024 MB RAM, running Fedora 4. Figure 5 shows the running times of the clustering processes.

Comparison of Query Distance Functions

In this section, we empirically evaluate the two distance functions we presented in the sections Vectorial Representation Queries and Vectorial Representation with Unbiased Popularity, which we refer to as *excerpt terms* and *unbiased excerpt terms* distances, respectively. We also consider in the experiments the following two distance functions introduced by Beeferman and Berger (2000): (a) *query terms*: standard cosine distance of a term vectorial representation of the query keywords using a TF-IDF weighting scheme, and (b) *cocitation*: degree of cocitation, calculated over the collection of documents selected by users for each queries.

Firstly, we selected the 500 most popular queries in the log and calculated a distance matrix for each of the four aforementioned distances. Then, we performed a *Mantel's test* to compare the distance matrices. Let A and B be two distance matrices of $N \times N$ elements. In a Mantel's test, the following statistic is computed:

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{i,j} B_{i,j} \tag{8}$$

A large value of Z would indicate a strong link between the distance functions. To test the significance of such a link, a permutation test was carried out, where the entries of one of the distance matrices are permuted using a Monte Carlo method. The value of Z is recalculated for each permutation. Finally, a P value is obtained by comparing the original Z

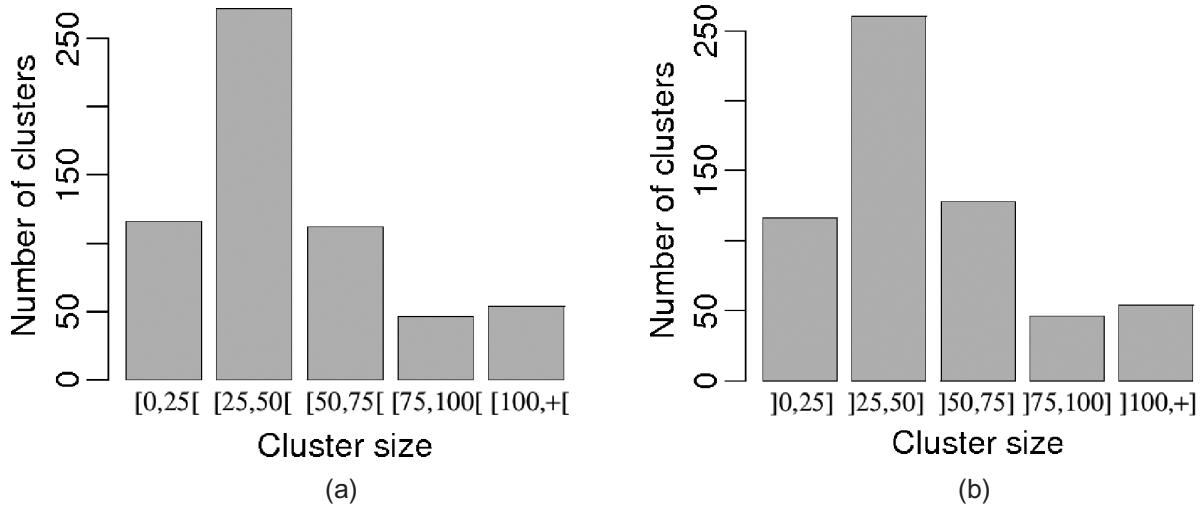

FIG. 4. Histogram of the number of queries per cluster for the (a) excerpt terms based method and the (b) unbiased excerpt terms based method.

DOI: 10.1002/asi

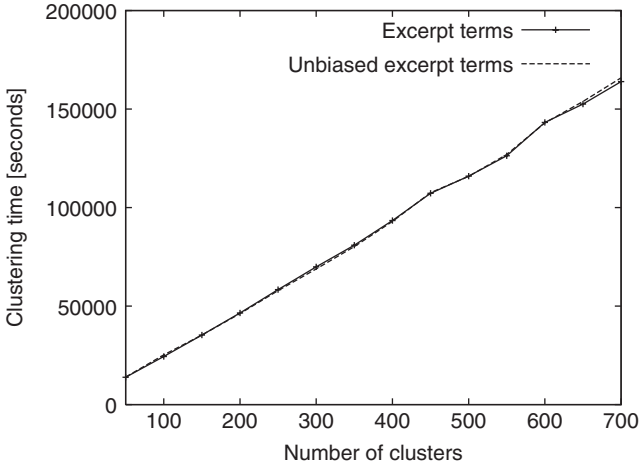

FIG. 5. Running times for the clustering processes.

statistic with each recalculated value of Z. In Table 2, we show the Z and P values obtained after 1000 permutations. In the third column, we show the Pearson correlation coefficient. Our results show that there exists a significant correlation between the query terms and the cocitation distances. The excerpt terms and the unbiased excerpt term distances are moderately correlated. We found a low correlations for the remaining pairs of distances.

In Figure 6, we show histograms of the distances calculated for the 500 queries selected. Figure 6c shows that 63% of the values for the excerpt term distance are less than 0.95. Figure 6d shows that more than 70% of the values for the unbiased excerpt term distance are less than 0.95. On the other hand, Figures 6a and 6b show that for the query terms and cocitation distances, more than 95% of the values are very close to the maximal value of 1.

We also evaluated the effectiveness of the proposed distance functions using a set of 600 pairs of quasi-synonym queries. Since each pair contains queries that are semantically similar, we expect to obtain small values for the excerpt term and unbiased excerpt term distances. Table 3 shows the distribution (percentage) of the values per decile, for each distance function, and Table 4 shows the median per decile and distance function. The results show that the distance functions we propose perform better than the cocitation and query term distance functions.

Table 5 shows the percentile of each distance value for 20 pairs of queries selected randomly among the 600 pairs of quasi-synonym queries. Rows are sorted first by query term distance, then by cocitation distance, and finally by excerpt and unbiased excerpt term distances. The values in bold represent the smallest percentiles obtained for each pair of queries. As the table shows, the distance functions we propose produce a best result in 9 of the 20 pairs of queries. In particular, in the first 7 pairs of queries the query terms and cocitation distances perform poorly. In addition, for the cases where the best results are obtained by the query terms or cocitation distances, the distance functions we propose perform well.

TABLE 2. Comparison of query distance functions based on a Mantel's test.

	Z value	P value	Pearson correlation
Query terms vs. cocitation	4914	0.000998	0.905293
Query terms vs. excerpt terms	4558	0.000996	0.365039
Query terms vs. unbiased excerpt terms	4501	0.000996	0.365535
Cocitation vs. excerpt terms	4570	0.000999	0.391686
Cocitation vs. unbiased excerpt terms	4512	0.000999	0.390907
Excerpt terms vs. unbiased excerpt terms	4506	0.000998	0.650991

Evaluation of Applications

The evaluation of the applications proposed in the two sections that follow considers the study of a set of 30 randomly selected queries. The 30 queries were selected following the probability distribution of the 30,363 queries of the 6-month log. In Table 6, we show the selected queries for the experiments and some descriptive features of their clusters calculated using the unbiased excerpt term distance and $k = 600$. We used the clustering solution obtained for $k = 600$, because it represents a good trade-off between the quality and the size of the clusters. The results are sorted by the cluster rank, which indicates the quality of the cluster. Proper nouns are written in italics.

The first column of Table 6 shows the selected queries. The second column gives the rank of the clusters, according to their quality. The third column shows the cluster quality. The fourth column shows the cluster size.

The last column depicts the set of feature terms that best describe each one of the clusters. We have translated the terms in the original language of the search engine. Beside each feature term, there is a number that is the percentage of the within-cluster similarity that this particular feature can explain. For example, for the cluster of the query *used notebooks* the feature *Compaq* explains 41% of the average similarity of the queries in the cluster. Intuitively, these terms represent a subset of dimensions (in the vectorial representation of the query) for which a large fraction of queries agree. The queries in the cluster form a dense subspace for these dimensions.

Our results show that many clusters represent clearly defined information needs of search engine users and reflect semantic connections between queries that do not share query words. As an example, the feature term *brakes* of the cluster of query *tires* reveals that users who search for Web sites about tires also search for information about brakes. Probably, some sites containing information about tires contain references to brakes. Another example is the term *restaurants* related to query *food home delivery*, which shows that users who submitted this query are mainly interested in information about restaurants. These examples, and many

DOI: 10.1002/asi

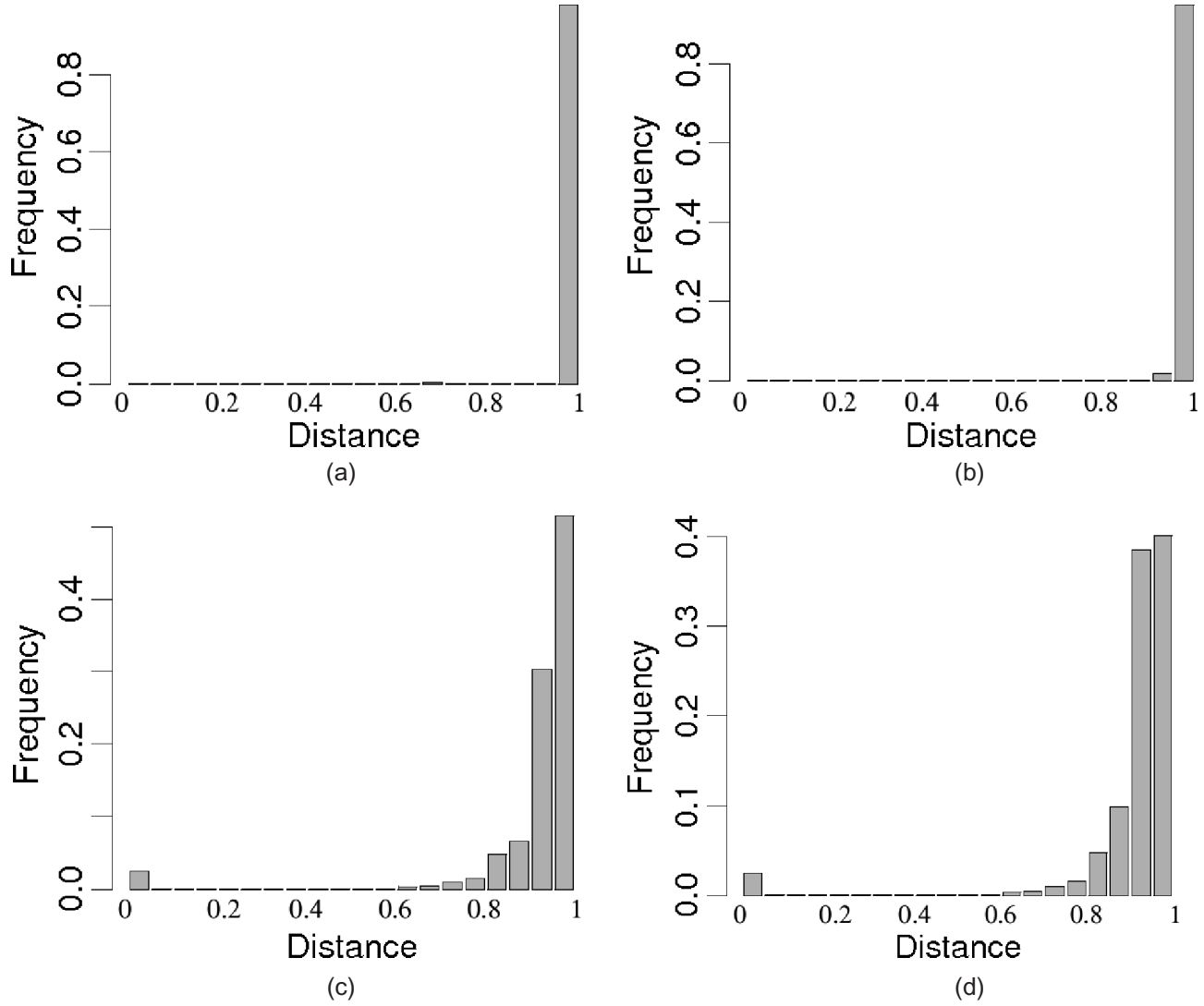

FIG. 6. Histogram of distance for the query collection considered in the experiments. (a) Query terms, (b) cocitation, (c) excerpt terms, (d) unbiased excerpt terms.

TABLE 3. Distance distributions (percentages) for a set of 600 pairs of quasi-synonym queries.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
Query terms	20.7	0	0	0	0	0	0	0	0	79.3
Cocitation	53	0	0	0	0	0	0	0	0	47
Excerpt terms	68.5	9.5	4.7	0.2	9.5	2.8	4.7	0.1	0	0
Unbiased excerpt terms	69	9.7	8.8	4.5	3.5	4.2	0.3	0	0	0

TABLE 4. Medians for each decile of the distance distributions shown in Table 3.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
Query terms	0.95	1	1	1	1	1	1	1	1	1
Cocitation	0.95	1	1	1	1	1	1	1	1	1
Excerpt terms	0.575	0.85	0.875	0.9	0.911	0.925	0.95	0.975	1	1
Unbiased excerpt terms	0.575	0.8	0.825	0.85	0.875	0.9	0.925	0.95	1	1

DOI: 10.1002/asi

TABLE 5. Distance values for 20 quasi-synonym queries.

Queries	Query terms	*P%*	Cocitation	*P%*	Excerpt terms	*P%*	Unbiased excerpt terms	*P%*
Houses—properties	1	100	1	100	0.899	42	0.874	25
Chilean music—*cueca*	1	100	1	100	0.869	19	0.852	23
Hospitals—clinics	1	100	1	100	0.913	47	0.882	36
Crashed cars—dismantlement	1	100	1	100	0.958	61	0.916	59
Games—toys	1	100	1	100	0.791	8	0.735	6
Chilean food recipes—typical food	1	100	1	100	0.861	19	0.825	12
Movies—films	1	100	1	100	0.741	5	0.642	4
Tarot—horoscope	1	100	0.981	12	0.872	20	0.771	7
Automobile dealers—car sales	1	100	0.971	9	0.597	4	0.549	3
Real estate—construction companies	1	100	0.967	8	0.449	3	0.411	3
Work—job	1	100	0.958	4	0.822	9	0.811	11
Jokes—humor	1	100	0.956	4	0.766	8	0.749	7
Cars—automobiles	1	100	0.944	1	0.77	8	0.71	5
Pubs—bars	1	100	0.925	1	0.701	5	0.604	4
Tires—wheel rims	1	100	0.916	1	0.556	3	0.584	4
Plans—maps	1	100	0.874	1	0.44	3	0.51	3
Work offers—job offers	0.591	1	0.966	8	0.388	3	0.323	3
Chords—songs with chords	0.422	1	0.761	1	0.318	3	0.372	3
Travel agencies—tourism agencies	0.333	1	0.869	1	0.408	3	0.406	3
Mp3—free mp3	0.292	1	0.894	1	0.231	3	0.357	3

Note. The columns labeled with *P%* indicate the percentile for each value.

TABLE 6. Queries selected for the experiments along with the cluster to which they belong.

Query	Cluster rank	Cluster quality	Size	Descriptive terms
Bikinis	1	0.863	11	Swimsuits (2%)
Chilean typical food	30	0.315	45	Typical (9%) south (4%)
Kino	46	0.278	33	*Polla* (24%) *Iman* (19%) *Loteria* (5%)
Embassy of *Canada*	67	0.261	27	Embassies (42%) consulates (18%)
Divorce law	68	0.260	49	Marriage (40%) law (12%) married couple (5%)
Chilean chats	73	0.248	43	Nick (2%) chat rooms (2%)
Swimming pools	76	0.244	45	Swimming pool (60%) Jacuzzi (2%)
Plots	94	0.241	74	Sales (21%) rentals (17%) properties (12%)
Second hand laptops	95	0.228	34	*Compaq* (41%) *Toshiba* (7%)
Used cars Chile	100	0.236	93	Cars (41%) automobiles (9%) vehicles (4%)
Lemon pie	117	0.221	57	Catering service (4%) cakes (6%)
Rental	119	0.219	56	Properties (47%) apartments (10%)
Rentals in *Iquique*	119	0.219	56	Properties (47%) apartments (6%)
Prefabricated homes	126	0.217	30	Intermodal (16%) houses (11%)
Yellowpages	135	0.209	50	*Publiguias* (38%) pages (17%)
Applications	155	0.2	55	Judicial (19%) appeal (18%)
Ministries	164	0.206	166	Law (35%) incise (11%) ministry (3%)
Harry Potter	188	0.190	86	*Papelucho* (38%) literature (2%)
V Region cabins	205	0.180	37	*Quisco* (32%) *Algarrobo* (12%)
Viña del Mar	226	0.177	76	*Valparaiso* (75%) guest house (2%)
Tires	228	0.173	55	Brakes (41%) batteries (6%)
Spare pieces	247	0.170	74	Accesories (48%) auto body shop (2%)
INE	269	0.164	28	*IPC* (25%) statistics (11%)
Florist shop	304	0.153	34	Flower (21%) roses (17%) lower bouquets (14%)
Emotional intelligence	351	0.148	51	*Reiki* (13%) *Yoga* (8%)
Bars	390	0.142	81	Panoramas (12%) gastronomy (5%)
Steam cycle	443	0.019	70	Weather (12%) mountain chain (7%)
Food home delivery	481	0.120	92	Food (41%) dishes (12%) restaurants (8%)
Software	543	0.102	82	Software (42%) *Windows* (18%)
Chile post office	578	0.095	130	Companies (7%) market (5%)

DOI: 10.1002/asi

others found in our results, showed the utility of our framework for discovering information needs related to queries.

Query Recommendation

In order to assess the quality of the query recommendation algorithm for the 30 queries given in Table 6, we follow a similar approach to Fonseca and coworkers (Fonseca et al., 2003). The relevance of each query to the input query was judged by 20 members of our Computer Science Departments. They analyzed whether the answers of the queries are of interest to the input query. Our results are given in graphs showing precision vs. numbers of recommended queries.

Figure 7 shows the average precision for the queries considered in the experiments. The ranking is obtained using the score function of equation 7. For the methods based on the cocitation and the query term distances the scores are calculated using the popularity of the queries. In average, with the proposed methods we obtain a precision of 75% for the first 10 recommended queries. Therefore, the suggested queries are relevant to users who submitted the original queries. Our results also show that the rank schemes proposed are better than the scores obtained by considering only the popularity of the queries in the cluster that are recommended using cocitation or query terms. Finally, the unbiased excerpt term distance achieves the best results among the distances considered in the experiments.

Answer Ranking

We compared our ranking algorithm with the algorithm provided by the search engine for the 30 queries given in Table 6. The proposed ranking algorithm was computing using $\beta = 0$ for the coefficient of equation 6. The ranking algorithm of the search engine is based on a belief network that is trained with links and content of Web pages and does not consider logs. The top-10 answers of the queries studied were considered in the experiment. The judgments of the relevance of the answers to each query were made by people from our Computer Science Departments.

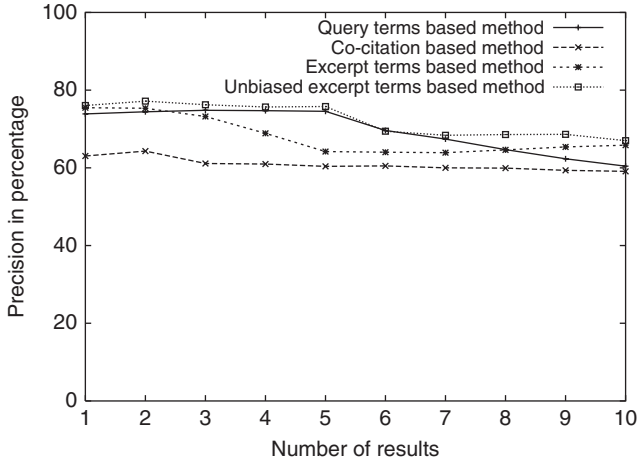

FIG. 7. Average retrieval precision of the query recommendation application.

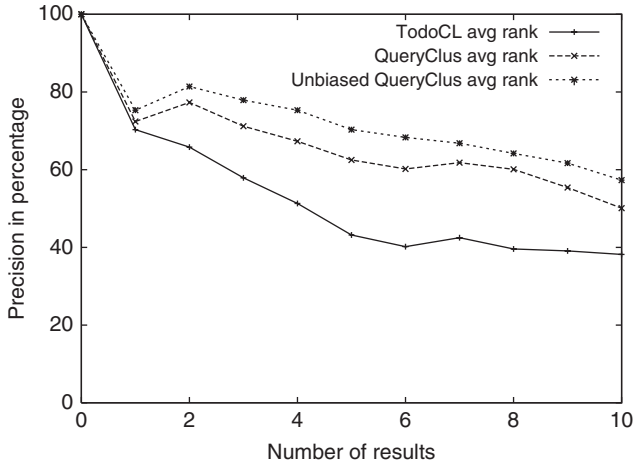

FIG 8. Average retrieval precision of the proposed ranking algorithm.

Figure 8 shows the average retrieval precision of the search engine and the proposed algorithms for answer ranking using both methods. The graph shows that our algorithm can significantly boost the average precision of the search engine. For all the queries studied in the experiment our algorithm outperforms the ranking of the search engine. The average precision of the proposed method is approximately 65% for the top-10 results. The original rank has only an average precision of 50%. The difference is more significant for the top-5 results. Our methods are close to a precision value of 75% while the original rank is close to the 60%. The figure shows that the method based on the unbiased excerpt term distance is better than the other two methods.

Figure 9 shows a scatterplot of the ranking based on unbiased excerpt terms and the original rank, for the 300

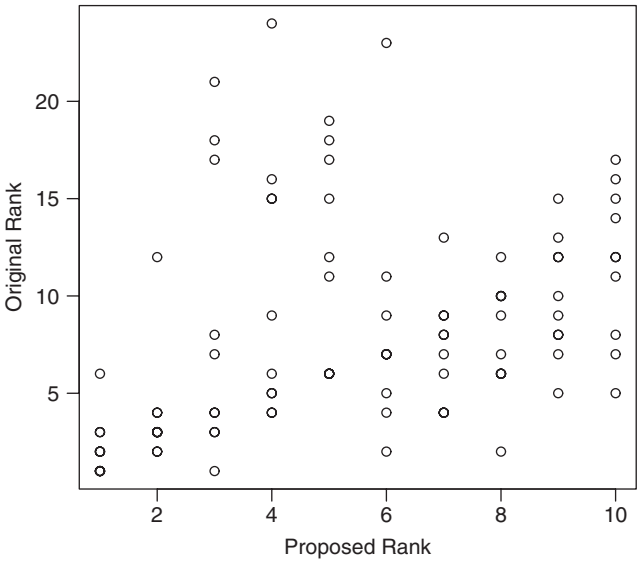

FIG. 9. Scatterplot of the original ranking versus the proposed ranking for the 300 documents considered in the experiments.

DOI: 10.1002/asi

documents considered in the experiments. As the plot shows, the rankings have low correlation. The Pearson coefficient for the graph is 0.3639. Observe that some of the top documents in our ranking appear in the last 20 or 30 places of the original ranking.

Conclusion

We have proposed a clustering framework that allows one to find groups of semantically related queries. Our experiments show that the bias reduction technique proposed improves the quality of the clusters found. The results also provide evidence that our ranking algorithm improves the retrieval precision of the search engine, and that our query recommender algorithm has good precision in the sense that it returns relevant queries to the input query.

The notion of query similarity we propose has several advantages: it is simple and easy to compute; it allows one to relate queries that happen to be worded differently but stem from the same information need; it leads to similarity matrices that are much less sparse than matrices based on previous notions of query similarity (see Related Work); the vectorial representation of queries we propose yields intuitive and useful feature characterizations of clusters.

Traditional techniques for document retrieval can be used to handle queries in our framework. As an example, we could implement an inverted index scheme for terms in queries to retrieve related queries efficiently.

Other measures for the interest of the queries in query recommendation are possible, for example, finding queries that share words but not selected documents. This might imply that the common words have different meanings if the text of the documents is also not shared. Hence we can detect polysemic words. On the other hand, if words are not shared and many terms in the documents are shared, that may imply a semantic relation among words that can be stored in an ontology.

Acknowledgments

Ricardo Baeza-Yates and Carlos Hurtado were supported by Millenium Nucleous Grant P04-067-F from Mideplan (Planning Ministry), Chile. Marcelo Mendoza was supported by FONDECYT project 1061201 from CONICYT, Chile.

References

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004a, May). Query clustering for boosting Web page ranking. In AWIC 2004, Lecture Notes in Artificial Intelligence (Vol. 3034, pp. 164–175). Incan, Mexico.

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004b). Query recommendation using query logs in search engines. In EDBT 2004 Workshops, Lecture Notes in Computer Science (Vol. 3268, pp. 588–596).

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval. Addison-Wesley.

Baeza-Yates, R., Saint-Jean, C., & Castillo, C. (2002, September). Web structure, dynamics and page quality. In SPIRE 2002, Lecture Notes in Computer Science (Vol. 2476, pp. 117–130). Libson, Portugal: Springer.

Beeferman, D., & Berger, A. (2000, August). Agglomerative clustering of a search engine query log. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 407–416). Boston: ACM Press.

Breese, J., Heckerman, D., & Kadie, C. (1998, July). Empirical analysis of predictive algorithms for collaborative filtering. In UAI '98: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. Madison, Wisconsin: Morgan Kaufmann.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. Computer Networks, 30(1–7), 107–117.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., et al. (2000). Graph structure in the Web. Computer Networks, 33(1–6), 309–320.

Correia-Saravia, P., Silva de Moura, E., Ziviani, N., Meira, W., Fonseca, R., & Ribeiro-Neto, B. (2001, September). Rank-preserving two-level caching for scalable search engines. In proceedings of the 24th International ACM Conference on Research and Development in Information Retrieval (pp. 51–58). New Orleans, LA: ACM Press.

Fonseca, B.M., Golgher, P.B., De Moura, E.S., & Ziviani, N. (2003, November). Using association rules to discover search engine related queries. In First Latin American Web Congress (LA-WEB '03; pp. 66–71.), Santiago, Chile: IEEE Computer Society Press.

Frei, H., & Schuble, P. (1991). Determining the effectiveness of retrieval algorithms. Information Processing and Management, 27(2), 153–164.

Jansen, M., Spink, A., Bateman, J., & Saracevic, T. (1998). Real life information retrieval: A study of user queries on the Web, ACM SIGIR Forum, 32(1), 5–17.

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. ACM-SIAM Symposium on Discrete Algorithms (SODA), 46(5), 604–632.

Lawrence, S., & Giles, C. (1999). Searching the Web: General and scientific information access. IEEE Communications, 37(1), 116–122.

Lesk, M., & Salton, G. (1968). Relevance assessments and retrieval system evaluation. Information Storage and Retrieval, 4(3), 343–359.

Paredes, R., & Chavez, E. (2005, November). Using the k-nearest neighbor graph for proximity searching in metric space. In SPIRE, Lecture Notes in Computer Science (Vol. 3772, pp. 127–138). Buenos Aires, Argentina: Springer.

Silverstein, C., Henzinger, M., Marais, H., & Moricz, M. (1999). Analysis of a very large Web search engine query log. SIGIR Forum, 33(1), 6–12.

Wen, J., Nie, J., & Zhang, H. (2001, May). Clustering user queries of a search engine. In Proceedings of the International Conference on World Wide Web (WWW) (pp. 162–168). Hong Kong, China: ACM Press.

Xu, J., & Croft, W.B. (2000). Improving the effectiveness of information retrieval with the local context analysis. ACM Transaction of Information Systems, 1(18), 79–112.

Yuwono, B., & Lee, L. (1996). Search and ranking algorithms for locating resources on World Wide Web. In Proceedings of the 12th International Conference on Data Engineering (pp. 164–171). IEEE Computer Society.

Zaiane, O.R., & Strilets, A. (2002, September). Finding similar queries to satisfy searches based on query traces. In Advances in Object-Oriented Information Systems, OOIS 2002 Workshops Lecture Notes in Computer Science (Vol. 2426). Montpellier, France: Springer.

Zhang, D., & Dong, Y. (2002). A novel Web usage mining approach for search engines. Computer Networks, 39(3), 303–310.

Zhao, Y., & Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. Machine Learning, 55(3), 311–331.

DOI: 10.1002/asi