IJCSN International Journal of Computer Science and Network, Volume 6, Issue 2, April 2017
ISSN (Online) : 2277-5420
www.IJCSN.org
Impact Factor: 1.5

Data Mining Techniques for Mining Query Logs in Web Search Engines

[1] Ahmed S. Al-Hegami; [2] Hussein H. Al-Omaisi

[1] University of Sana'a –Sana'a - Yemen

[2] Alandalus University for Sciences and Technology/Engineering & IT college, Sana'a, Yemen

Abstract - The Web is the biggest repository of documents humans have ever built. Even more, it is increasingly growing in size every day. Users rely on Web search engines (WSEs) for finding information on the Web. By submitting a textual query expressing their information need, WSE users obtain a list of documents that are highly relevant to the query. Moreover, WSEs store such huge amount of users activities in query logs. Query log mining is the set of techniques aiming at extracting valuable knowledge from query logs. This knowledge represents one of the most used ways of enhancing the users search experience. The primary focus of this work is on introducing the data mining techniques for mining query logs in web search engines and showing how search engines applications may benefit from this mining.

Keywords - *Mining Techniques, Query logs, Search Engines*

1. Introduction

The Web is a huge read-write information space where many items such as documents, images or other multimedia can be accessed. In this context, several information technologies have been developed to help users to satisfy their searching needs on the Web, and the most used are search engines. Search engines allow users to find Web resources formulating queries (a set of terms) and reviewing a list of answers.

Web search has its root in information retrieval (or IR for short), a field of study that helps the user find needed information from a large collection of text documents. Traditional IR assumes that the basic information unit is a document, and a large collection of documents is available to form the text database. On the Web, the documents are Web pages.

The resulting growth in on-line information combined with the almost unstructured web data necessitates the development of powerful yet computationally efficient web data mining tools. Web data mining can be defined as the discovery and analysis of useful information from the WWW data. Web involves three types of data; data on the WWW, the web log data regarding the users who browsed the web pages and the web structure data. Thus, the WWW data mining should focus on three issues; web structure mining, web content mining and web usage mining. At the end of the nineties, the estimated size of the Web to be around 200 million static pages[1]. Recently, pointed out that the number of indexable documents in the Web exceeds 11.5 billion. The large size of the Web has impulsed the continuous development of a class of systems that help the users search for documents on the Web. These systems, known as search engines (Google , Yahoo , MSN Search and among others), give the users a simple interface by means of which to enter a set of terms known as queries[1].

Web search engines (WSEs) are queried by users to satisfy their information need. WSEs have stored in their logs information about users since they started to operate. This information often serves many purposes.

The primary focus of this work is on introducing the data mining techniques for mining query logs in web search engines and showing how search engines applications may benefit from this mining.

2. Query Log Mining and Web Search Engines

2.1 Web Search Engines

Web Search Engines (WSEs) are the primary way users access the contents of the Web. WSEs are part of a broader class of software systems, namely Information

Retrieval (IR) Systems.An IR system is basically a software whose main purpose is to return a list of documents in response to a user query. This description makes IR systems similar to DB systems. Indeed, the most important difference between DB and IR systems is that DB systems return objects that exactly match the user query, whereas IR systems have to cope with natural language that makes it simply impossible for an IR system to return perfect matches. A Web search engine is thus an IR system on a very large scale.

The first systems similar to modern web search engines started to operate around 1994.World WideWebWorm (WWWW) [1] created by Oliver McBryan at the University of Colorado, and the AliWeb search engine [1] created by MartijnKoster in 1994, are the two most famous examples. Since then many examples of such systems have been around theWeb: AltaVista, Excite, Lycos, Yahoo!, Google, ASK, MSN. Nowadays, searching is considered one of the most useful application on the Web.

In a paper overviewing the challenges in modern Web search engines' design, [1] state: The main challenge is hence to design large-scale distributed systems that satisfy the user expectations, in which queries use resources efficiently, thereby reducing the cost per query.

Therefore, the two key performance indicators in this kind of applications, in order, are: (i) the quality of returned results (e.g. handle quality diversity and fight spam), and (ii) the speed with which results are returned. Figure 2.1 shows the way the three applications interact and how the main modules of a web search engine are connected.

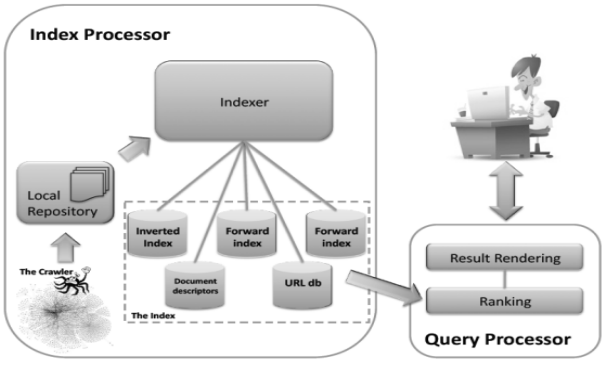


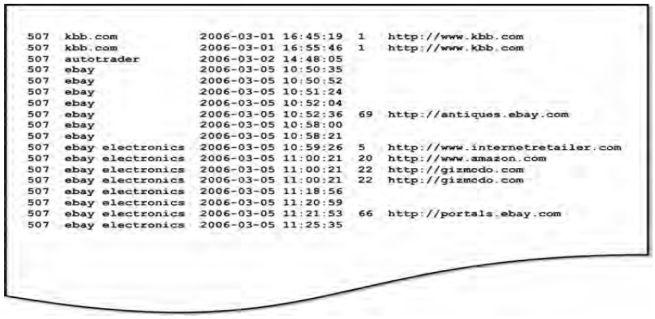
Figure 2.1The Typical structure of a web search engine [1].

Crawlers store the data into a repository of content (also known as Web document cache), and structure (the graph representing how Web pages are interconnected). In modern Web search engines, crawlers continuously run and download pages from the Web updating incrementally the content of the document cache.

2.2 Query Log

One of the most used ways of enhancing the users' search experience, in fact, is the exploitation of the knowledge contained within past queries. A query log, typically, contains information about users, issued queries, clicked results, etc. From this information knowledge can be extracted to improve the quality (both in terms of effectiveness and efficiency)of their system. Figure 2.2 shows a fragment of the AOL query log. The format of this query log represents a record using five features: user id, query, timestamp, rank of the clicked result, host string of the clicked URL [1].

Query logs keep track of information regarding interaction between users and the search engine. They record the queries issued to a search engine and also a lot of additional information such as the user submitting the query, the pages viewed and clicked in the result set, the ranking of each result, the exact time at which a particular action was done, etc. In general, a query log is comprised by a large number of records [q_i, u_i,t_i, V_i, C_i] where for each submitted query q_i, thefollowing information is recorded: i) the anonymized identifier of the user u_i,, ii) the timestamp t_i, iii) the set V_i of documents returned by the WSE, and iv) the set C_i of documents clicked by u_i.



```
507  kbb.com              2006-03-01 16:45:19  1   http://www.kbb.com
507  kbb.com              2006-03-01 16:55:46  1   http://www.kbb.com
507  autotrader           2006-03-02 14:48:05
507  ebay                 2006-03-05 10:50:35
507  ebay                 2006-03-05 10:50:52
507  ebay                 2006-03-05 10:51:24
507  ebay                 2006-03-05 10:52:04
507  ebay                 2006-03-05 10:52:36  69  http://antiques.ebay.com
507  ebay                 2006-03-05 10:58:00
507  ebay                 2006-03-05 10:58:21
507  ebay electronics     2006-03-05 10:59:26  5   http://www.internetretailer.com
507  ebay electronics     2006-03-05 11:00:21  20  http://www.amazon.com
507  ebay electronics     2006-03-05 11:00:21  22  http://gizmodo.com
507  ebay electronics     2006-03-05 11:00:21  22  http://gizmodo.com
507  ebay electronics     2006-03-05 11:18:56
507  ebay electronics     2006-03-05 11:20:59
507  ebay electronics     2006-03-05 11:21:53  66  http://portals.ebay.com
507  ebay electronics     2006-03-05 11:25:35
```

Figure. 2.2 A fragment of the AOL query log [1].

2.2.1 Search Session

From query log information it is possible to derive Search Sessions, sets of user actions recorded in a limited period of time. The concept can be further refined into: i) Physical Sessions, ii) Logical Sessions, and iii) Super sessions.

Physical Sessions: a physical session is defined as the sequence of queries issued by the same user before a predefined period of inactivity. A typical timeout threshold used in web log analysis is t_0 = 30 minutes. [1].

Logical Sessions: a logical session or chain [1] is a topically coherent sequence of queries. A logical session is not strictly related to timeout constraints but collects all the queries that are motivated by the same information need (i.e., planning an holiday in a foreign country, gathering information about a car to buy and so on). A physical session can contain one or more logical session.

Super sessions: we refer to the sequence of all queries of a user in the query log, ordered by timestamp, as a supersession. Thus, a supersession is a concatenation of sessions.

Sessions are, thus, sequences of queries submitted by the same user in the same period of time. This data can be used to devise typical query patterns, used to enable advanced query processing techniques. Click-through data (representing a sort of implicit relevance feedback information) is another piece of information that is generally mined by search engines. In particular, every single kind of user action (also, for instance, the action of not clicking on a query result) can be exploited to derive aggregate statistics which are very useful for the optimization of search engine effectiveness. How query logs interact with search engines has been studied in many papers [1].

2.2.2 The query log mining process

In order to observe a query session, it is necessary to pre-process the server log file that identifies the user's request and that belongs to a query session. To do this, it is necessary to identify IP numbers, browsers, operative systems and query instances to associate a given request to a query session.

When query session data is built, it is possible to explore several relationships between queries and documents. We call this process query log mining. The query log mining process is defined as the search of non-trivial patterns in the query log data. Of course it is defined as a data mining process but it focuses on the exploration of patterns in the query log data.

A query log mining process is going to be used in order to identify useful patterns in the query log data that allow us to improve the quality of the answer lists given by a search engine. In order to explore the data we can use standard data mining techniques such as association rules, clustering and classification.

[1] proposed a query log mining process. First of all, focus on identify associations among queries. Associations allow us to identify generalization/specialization relationships among queries, or similarity relations among them.

When relationships among queries are identified, it will be possible to propose reformulation schemes in order to expand the original query or to propose alternative queries that are more related to a specific concept.

Second, work on the construction of representations of query sessions using the terms that compound queries and documents. Several term weight schemes can be used at this point based on the patterns discovered in the query log mining process. By defining distance functions between queries and standard data mining techniques, such as clustering, it will be possible also to identify groups of similar queries. Finally, by using the query clusters it will be possible to identify recommendable queries or documents to the users who formulate queries that belong to any cluster.

Frequently, search engines provide Web directories to help users in their searches. Web directories are hierarchical structures of nodes organized as trees. The hierarchy represents a taxonomy and the relationships among nodes represent generalization/specialization relationships among the concepts behind the structure. Each node of the Web directory shows a list of documents related to a concept.

Following the query log mining process, we will build representations for the nodes in the directories using the terms of the documents that compound each document list. By defining distance functions between queries, documents and directory nodes, it will be possible to classify queries and documents in a Web directory. We will also show that the classification of queries into a Web directory is helpful to maintain the structure. The main activities in the query log mining process are described in figure 2.3.

In light of the above, the query log mining process will be based on the following ideas:
1. Query specialization and/or generalization. This will allow us to refine the initial query, guiding the user toward a search with more precise results.

2. Determination of similar query groups. It will allow us to group queries having a large number of preferences, with similar queries having a small number of preferences. Then, queries and documents may be recommended according to the popularity of each group.
3. Query classification into directories. Given a query, the nearest node in a given directory can be determined. Each directory node may show documents and/or queries which are relevant to the topic of the node. Finally, it will also be possible to specialize or generalize the query by navigating the directory.

At this point, it is relevant to specify which kind of applications will be considered as output of the query log mining process. Three kind of applications will be the output[1]:

1. Query recommendation: focusing on the reformulation of the original query, these kind of applications aim at identifying relationships between the original queries and alternative queries, such as generalization/specialization relationships.
2. Document recommendation: these kind of applications will identify relevant documents to the original query.
3. Query classification: these kind of applications will identify relevant queries and documents for each node in the directory, enriching their descriptions.

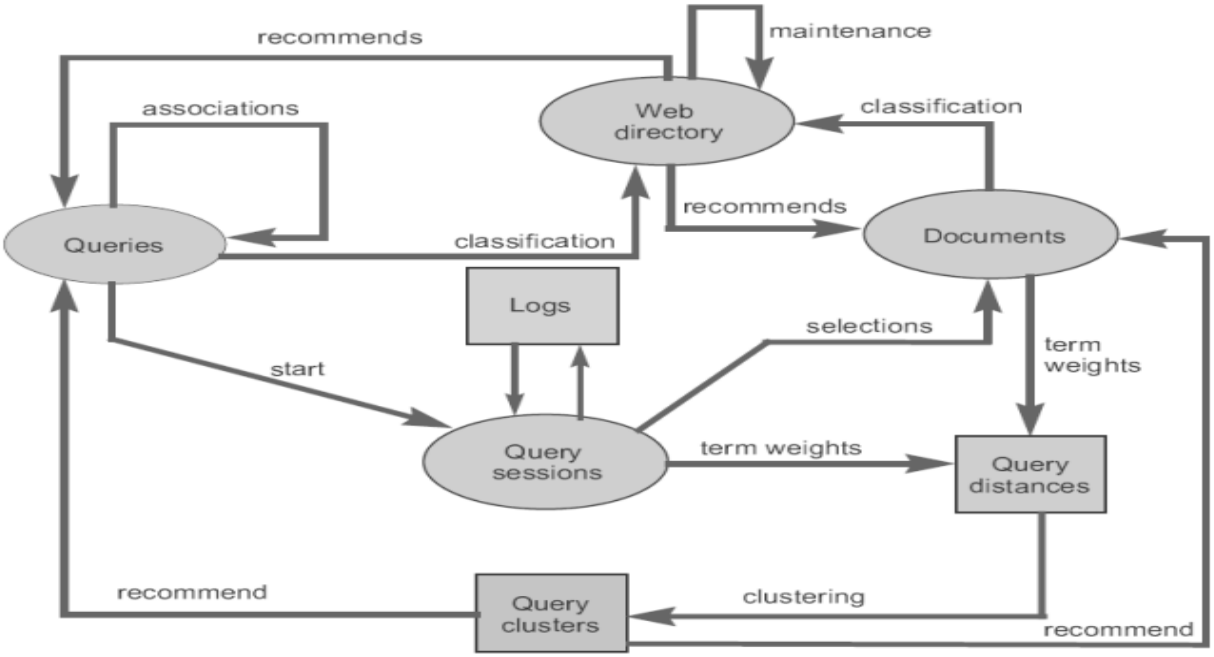


Figure 2.3 The query log mining process [1].

A global view of the relationships between data mining techniques used and the applications generated from the use of these techniques over the query log data is given in Figure 2.4.

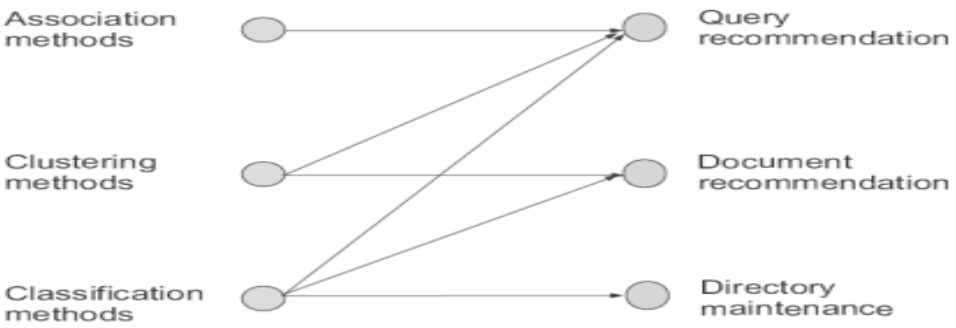


Figure 2.4 Relation between the data mining techniques used in this thesis and the applications generated [1].

3. Data Mining Techniques for Mining Query Logs in Web Search Engines

Previously submitted queries represent a very important mean for enhancing effectiveness of search systems. As already mentioned, query logs keep track of information regarding interaction between users and the search engine. Sessions are the sequences of queries submitted by the same user in the same period of time. This data can be used to devise typical query patterns, used to enable advanced query processing techniques[2]. Click-through data (representing a sort of implicit relevance feedback information) is another piece of information that is generally mined by search engines. All in all, every single kind of user action (also, for instance, the action of not clicking on a query result) can be exploited to derive aggregate statistics which are very useful for the optimization of search engine effectiveness.

Differently from what is presented in this section about enhancing efficiency, all the techniques presented in this part impact on the user experience starting from the interface presented by the search engine.

In this section, we report the possibilities offered by data mining techniques on query logs for the enhancement of search effectiveness in search engines.

Query log mining is useful for enhancing the search experience of web search engine users in term of effectiveness. Effectiveness in search systems refers to the quality of returned results. To this aim, five main application scan be highlighted: i) query expansion, ii) query recommendation, iii) personalized query results, iv) learning to rank, and v) query spelling correction. In the following we will investigate the major techniques that are mainly related to query expansion and query recommendation.

3.1 Query Expansion

The statistics showed that web queries are very likely to be short, poorly built, and sometimes mistyped. For these reasons, the recalling power of queries is too much strong resulting in overwhelming the user with a lot of (sometimes) useless results. One of the main means with which a search engine precision might be enhanced are query expansion and query suggestion.
To confirm that these two mechanisms are rather important also in the real-world, it is worth noticing that quite recently web search engines have started to insert within their results both suggestions, and refinements to queries submitted by users. For instance, ASK within its result page inserts two boxes: “Narrow your search” (query expansion), and “Expand your search” (query suggestion), both containing related, and potentially useful, queries.

The first question we want to answer is: can query expansion be of help? The main advantage of query expansion is that users tune interactively the expansions and that when based on query log analysis, expansion is more effective due to the fact that queries are expanded with terms that previous users have specified to improve the query. The main drawback is that the search (and also the interactive search process) can take slightly longer.

It has been highlighted by [3] that queries and documents are rather poorly correlated. Using a two-month query logs (about 22 GB) from the Encarta search engine (http://encarta.msn.com), and 41,942 documents from the Encarta website, they measure the gap between the document space (all the terms contained in documents) and the query space (all the terms contained in queries). For each document in the document space they construct a corresponding virtual document in the query space by collecting all queries for which the document has been clicked on. A virtual document is represented as a vector, where the weight of each term is defined by the $tf \times idf$ measure. The similarity between a query and the relative clicked documents is in turn measured by computing the cosine similarity between the two corresponding vectors. Actually, since many terms in document vectors appeared with zero weights in its corresponding query vector, to obtain a fairer measure, they only used the most important words in the document vectors as resulting from a simple $tf \times idf$ term ranking schema. Results confirm what expected: in most cases, the similarity values of term usages between user queries and documents are between 0.1 and 0.4 with only a small percentage of documents having similarity above 0.8. The average similarity value across the whole document collection is 0.28, which means the average internal angle between the query vector and the document vector is 73.68 degrees confirming that there is a quite large gap between the query space and the document space. Expanding a query might be of great help to bridge this gap as much as possible.

Query expansion is a technique dating back to seventies. Actually, one of the first works making explicit use of past queries to improve the effectiveness of query expansion techniques in traditional IR systems is [11]. It was proposed just a few years before the actual boom of

search engines. The method, called past-query feedback, tests its effectiveness against the TREC-5 dataset. It uses a query DB made upon a combination of 50 ad hoc TREC-5 queries, 50 ad hoc TREC-3 queries, and 10 ad hoc TREC-4 queries. Basically, they build off-line an affinity pool made up of documents retrieved by similar past queries. When a query is submitted it is firstly checked against the affinity pool (which represent a sort of "high-quality" document repository for the purpose of expansion) and from the resulting top scoring documents, a set of "important" terms is automatically extracted to enrich the past query. Term relevance (i.e., importance) is computed using a straightforward ***tf* × *idf*** scoring function. Past-queries feedback showed an improvement of 38.3% in mean average precision if compared to the non-query expansion case. We do not enter into the details of this method since it does not refer only to the case of web search engines. Besides, the method shows the in-embryo idea underlying most of the methods proposed thus far in the literature.

One of the first works exploiting past usage information to expand web queries is the one presented by [3]. Their method works by exploiting correlations among terms in clicked documents and user queries. Indeed, the exploitation of click-through data is due to the general assumption that usually is made in these kind of studies: clicked documents are relevant to a query. The method starts by extracting a set of query sessions from the query log. A query session, for a given user, consists of a query and a list of clicked results. For example the query session "Britney Spears"-4,982-2,212-8,412 represents a user submitting the query "Britney Spears" and successively clicking on documents 4,982, 2,212, and 8,412 respectively. For each one of the clicked documents, in each query session, a list of terms is extracted. The set of all terms contained within each clicked documents makes up the Document Terms set. The set of all the terms contained within all the queries, instead, forms the Query Terms set. Given a term in the Document Terms set ***td*** and a term in the Query Terms set ***tq***, a link is created between ***td***, and ***tq*** if and only if for at least one query containing ***tq*** there exists a licked document containing the term ***td***. Each link is then weighted by the degree of term correlation between its endpoints. The correlation is given by the conditional probability that term ***td*** appears given that term ***tq*** already appeared, i.e., ***P(td|tq)***.
By an algebra argument,1 this probability can be computed as:

$$P(t_{\mathrm{d}}|t_{\mathrm{q}}) = \sum_{D_i \in S_{\mathrm{q}}} P(t_{\mathrm{d}}|D_i) \frac{\mathrm{freq}(t_{\mathrm{q}}, D_i)}{\mathrm{freq}(t_{\mathrm{q}})}$$

Equation 3.1

where:

- ***Sq*** is the set of clicked documents for queries containing the term ***tq***.
- ***P(td|Di)*** is the normalized weight of the term td in the document ***Di*** divided by the maximum value of term weights in the same document ***Di***.
- ***freq(tq,Di)*** is the number of the query sessions in which the query word ***tq***and the document ***Di*** appear together.
- ***freq(tq)*** is the number of queries containing term ***tq***.

The term correlation measure is then used to devise a query expansion method relying on a function measuring the cohesion between a query ***Q*** and a document term ***td***. The formula of the cohesion weight (CoWeight) is given by:

$$\mathrm{CoWeight}(Q, t_{\mathrm{d}}) = \log\left(\prod_{t_{\mathrm{q}} \in Q} P(t_{\mathrm{d}}|t_{\mathrm{q}}) + 1\right)$$

Equation 3.2

It is worth remarking that we are assuming that terms within a query are independent (this justifies the productory in the formula). The cohesion weight is used to build a list of weighted candidate expansion terms. Finally, the top-k ranked terms (those with the highest weights) are actually selected as expansion terms for ***Q.***

Pseudo-Relevance Feedback [4] is one of the most famous techniques used to expand queries. Some techniques build upon Pseudo-Relevance Feedback to provide better expansions.

In the previously presented paper, the query log used is from the Encarta search engine (http://encarta.msn.com), and the relative 41,942 documents are from the Encarta website. To assess the improvement in effectiveness, a total of 30 queries are extracted randomly from a combination of the Encarta query logs, from the TREC query set, and from a small subset of hand-crafted queries. The query-log-based method was compared against the baseline of not using query expansion, and the local context analysis method (that does not make use of query log information) proposed by [4]. For the local context method the number of expanded terms for each query was set to 30, whereas for the log-based it was set to 40. The mean average precision for the baseline on the test collection was 17.46%, whereas the local context method

scored a mean average precision of 22.04%, an improvement of around the 26.24% on the baseline. The log-based method scored a mean average precision of 30.63% corresponding to an improvement of around 75.42% on the baseline. The number of terms used in the expansion has an impact on the results, actually another experiment showed that when more than 60 terms are considered in the expansion the mean average precision of the log-based method starts to decrease.

The technique of [4] is quite obsolete if compared to today's search engine technology. In particular, expanding a query to be longer than 30 terms is definitely not viable due to the overhead it causes in the corresponding query processing phase.

Indeed, Query/Documents terms co-occurrence is just a possible way of associating document terms to query terms.[5] use the concept of Query Association. User queries become associated with a document if they are "similar" that document. Queries, thus, enrich documents with an associated Surrogate Document. Surrogate documents are used as a source of terms for query expansion. Obviously not all of the queries concur to form query associations. First of all, whenever a query is submitted it is associated with the top ***K*** documents returned. Furthermore, to make the system more scalable, instead of keeping all of the queries associated with the documents only the M closest queries are kept. Similarity, then, is computed using a standard scoring function. In the experiments performed by the authors they used the Okapi BM25 [6] scoring function. In case a document reached its full capacity of associated queries, the least similar query is replaced by a newly submitted one only in case it has a similarity higher than the least similar associated query. Depending on two parameters, the method has to be fine tuned with respect to the different values of ***K*** and ***M***. In a recent paper, [7] empirically set ***K*** = 5, and ***M*** = 3. Summaries, thus, tend to be small but composed of high quality surrogate documents.
[5] use the Robertson and Walkers term selection value formula [179] (Equation (3.3)) to select the e expanding terms.

$$\text{weight}(t) = \frac{1}{3}\log\frac{\frac{\text{RetDocs}(t)+0.5}{f_t-\text{RetDocs}(t)+0.5}}{\frac{|T|-\text{RetDocs}(t)+0.5}{\text{NDocuments}-f_t-|T|+\text{RetDocs}(t)+0.5}}$$

Equation 3.3
where:

- ***RetDocs(t)*** is the number of returned documents containing term ***t.***
- ***ft***is the number of documents in the collection containing ***t***.
- ***NDocuments*** is the total number of documents in the collection.

How the above equations are used? First of all, we shall present a generalization of a query expansion algorithm:

(1) For a newly submitted query ***q***, an initial set ***T*** of top ranked documents is built.
(2) From ***T***, a list of e expanding terms is selected using a term selection value formula.
(3) A new query made up of the previous one with appended the top most scoring terms from the collection is submitted again to the search engine to obtain a new list of results.

In standard query expansion, steps (1) and (2) of the above generalized expansion algorithm are performed within the full-text of the entire collection. [5] refer to this as the FULL–FULL method. Considering a surrogate documents in either step (1), or (2) or both we may have the following three combinations:

- **FULL–ASSOC**, where the original query ranks documents of the full text collection and expansion terms are selected from the surrogate document. Therefore, step (1) of expansion is on the full text of documents in the collection, while step (2) is based on query associations.
- **ASSOC–FULL**, where surrogates have been built and retrieved instead of the full-text, but the query expansion is computed selecting highly scoring terms from the full-text.
- **ASSOC–ASSOC**, where both retrieval and expansion are computed on surrogate documents.

In addition they also experiment with an expansion schema called **QUERY–QUERY** where steps (1) and (2) are performed directly on previously submitted queries, instead of considering query associations. The **QUERY–QUERY** schema captures the idea of expanding queries with terms used by past users in queries considered better specified.

Experiments conducted on a TREC collection showed the superiority of the **ASSOC–ASSOC** schema that outperformed the classical **FULL-FULL** by 18–20%. In this setting, the values for K and M were set to those empirically found by [7] to perform better. It is worth noticing how the combination of both past queries and full-text is superior to using either one of the two independently.

Just to make the discussion more concrete, following an example of choice of terms for expansion. For the query

earthquake, the **FULL–FULL** method expanded the query with the terms: *"earthquakes tectonics earthquake geology geological"*, whereas the **ASSOC–ASSOC** expanded earthquake with a more descriptive "*earthquakes earthquake recent nevada seismograph tectonic faults perpetual 1812 kobe magnitude california volcanic activity plates past motion seismological"*.

The query expansion techniques shown so far have been rarely applied by search engines. As it is evident from the discussion so far, they suffer, mainly, of scalability issues. In many cases the analysis done is very heavy and cannot scale very well. Another possibility is represented by query suggestion. In the next part we present these techniques that have been proven to be effective and scalable in search engines.

3.2 Query Suggestion

As opposed to query expansion, query suggestion is the technique consisting of exploiting information on past users' queries to propose a particular user with a list of queries related to the one (or the ones, considering past queries in the same session) submitted. Furthermore, the number of suggestions can be kept as short as we want, usually it ranges from two to five queries suggested. With query expansion, in fact, users can select the best similar query to refine their search, instead of having the query automatically, and uncontrollably, stuffed with a lot of different terms. Obviously, this only partially overcomes the issue above, because if a topic is still under-represented it is very unlikely that it is suggested within a query suggestion list.

Roughly speaking, drawing suggestions from queries submitted in the past can be interpreted as a way of "exploiting queries submitted by experts to help non-expert users [8]." Therefore, the majority of query suggestion techniques detect related queries by selecting those that are mostly similar to the ones submitted in the past by other users.

A naive approach, as stated in [9], to find queries similar to another one consists of simply looking for those queries sharing terms regardless of every other possible feature. So, for instance, the query "George Bush" would be considered, to some extent, similar to the query "George Michaels" given that, both share the term George. Obviously, this simple example shows that the naive approach might result misleading for users.

In literature there have been presented quite a lot of works on query recommendation. Ranging from selecting queries to be suggested from those appearing frequently in query sessions [10], to use clustering to devise similar queries on the basis of cluster membership [12, 15, 16], to use click-through data information to devise query similarity [17, 18].

Query sessions can be a precious source of information for devising potentially related queries to be suggested to a user. The idea is that if a lot of previous users when issuing the query ***q1*** also issue query ***q2*** afterwards, query ***q2*** is suggested for query ***q1***. One choice for capturing this idea is association rule mining [11] and it is actually used to generate query suggestions by Fonseca et al. [10].

Basically, the setting for a typical associations-mining algorithm consists of a set ***D*** of itemsets (i.e., sets of items) ***A***, each of which is drawn from a set ***I*** of all possible items. Each ***A*** is a member of the power set ***2I*** . The database considered is made up of transactions, each transaction is a set of items. Given two non-overlapping itemsets ***A*** and ***B,*** an association rule is a formula $A \Rightarrow B$ stating that the presence of ***A*** in a transaction implies the presence of ***B***. The algorithm has two parameters: the minimum support value σ – i.e., the minimum number of transactions containing $A \cup B$, and the confidence γ – i.e., the minimum accepted probability of having ***B*** in a transaction given that ***A*** is contained within the same transaction.

Computing association rules in very large DBs can be computationally expensive. In fact the approach by[10] allows only rules of the form $qi \Rightarrow qj$. Therefore, the effort needed to compute the frequent itemset mining phase is considerably reduced.

For each query qi, all the rules $qi \Rightarrow q1, qi \Rightarrow q2, \ldots, qi \Rightarrow qm$ are extracted and sorted by confidence level. To experiment the technique it has been used a query log coming from a real Brazilian search engine and consisting of 2.312.586 queries. The mining process is carried out by setting the support $\sigma = 3$, and by extracting all the possible association rules. To assess the validity of their approach they conducted a survey among a group of five members of their laboratory by asking for the top five frequent queries whether the proposed suggestions were relevant or not. Results were encouraging, even if the assessment phase is not convincing enough. For example, for the top 95 frequently submitted queries the system is able to achieve a 90.5% precision measured as the number of suggestions retained relevant for the five laboratory members surveyed. Clearly, the population of assessors is biased (they are all members of the same lab), thus

potentially another group of people might have found those results less meaningful.

The number of queries suggested is also important. Obviously, in this case it is not true that the more the better. Actually, increasing the number of queries causes a drop in the suggestion quality. The 90.5% figure in precision was measured for the case of five query suggested. Precision drops to 89.5% when 10 queries are suggested, down to 81.4% when 20 queries are suggested.

Actually, there is a trade-off not highlighted by authors. Although 90.5% precision for five queries corresponds to more than four queries relevant, 89.5% precision for 10 queries, instead, consists of around nine queries out of 10. The list of potentially interesting queries is thus richer in the case of more suggestion shown. On the other hand, users presented with a long list of queries might experience a "*swamping effect*" resulting in users simply ignoring suggestions. Therefore, fine-tuning the number of suggestions is of utmost importance for the success of the method. Indeed, the need for fine tuning the number of suggestions is on a per-query basis: for frequently submitted queries a long number of suggestion would be better, for rarely submitted ones the number of suggestion should be, very likely, kept inherently small.

As opposed to association rule mining, [9] use a Query Memory to store past queries and retrieve them according to the one submitted. In a sense, this method computes associations on-the-fly at query resolution time.

A Query Memory is a set of queries represented by six different features: (i) *BagTerms*: the bag-of-words representation of the query, i.e., the unordered set of terms contained within the query string; (ii) *Count*: the frequency with which the query has been submitted; (iii) *Ldate*: the timestamp recording the last time the query was submitted; (iv) *Fdate*: the timestamp recording the first time the query was submitted; (v) *QResults*: the query result list stored as records made up of *Rurl* – the URL, *Rtitle* – the title, and *Rsnippet*– bag-of-words representation of the snippet, of each result page; (vi) *Rdate*: the timestamp recording the date at which results were obtained (note that this timestamp not necessarily relates to *Ldate* and *Fdate*).

Let ***Q*** be the submitted query, ***Δ*** be the Query Memory, ***Q.terms*** be the bag-of-words representing the query terms, and ***Q.results*** be the array storing the results. In particular, ***Q.results[i]*** is the ith entry of the result set for ***Q***.

By using this representation we have seven different methods for returning a set of queries similar to the submitted one.

Naive query-based method: returning queries having in common at least one term.

$$\{q \in \Delta \text{ s.t. } q.\text{terms} \cap Q.\text{terms} \neq \emptyset\}$$

Naive simplified URL-based method: returning queries having in common at least one URL in the result lists.

$$\{q \in \Delta \text{ s.t. } q.\text{QResults.Rurl} \cap Q.\text{QResults.Rurl} \neq \emptyset\}$$

Naive URL-based method: returning queries having in common a large fraction of URLs in the result list. In the formula below, ***θm***, and ***θM***are a minimum and maximum threshold (both real numbers ranging from 0 to 1) used to tune the intersection cardinality.

$$\{q \in \Delta \text{ s.t. } \theta_{\text{m}} \leq \frac{|q.\text{QResults.Rurl} \cap Q.\text{QResults.Rurl}|}{|Q.\text{QResults.Rurl}|} \leq \theta_{\text{M}}\}$$

Query-title-based method: returning queries where terms in their result titles are contained in the submitted query.

$$\{q \in \Delta \text{ s.t. } \exists i \; q.\text{QResults[i].RTitle} \cap Q.\text{terms} \neq \emptyset \text{ and } q.\text{QResults}[i] \notin Q.\text{QResults}\}$$

Query-content-based method: it is the same as 4 only considering snippet terms instead of title terms.

$$\{q \in \Delta \text{ s.t. } \exists i \; q.\text{QResults[i].RSnippet} \cap Q.\text{terms} \neq \emptyset \text{ and } q.\text{QResults}[i] \notin Q.\text{QResults}\}$$

Common query title method: returning queries whose results share title terms.

$$\{q \in \Delta \text{ s.t. } \exists i, j \; q.\text{QResults[i].RTitle} \cap Q.\text{QResults[j].RTitle} \neq \emptyset\}$$

Common query text method: it is the same as 6 only considering snippet terms instead of the title terms.

$$\{q \in \Delta \text{ s.t. } \exists i, j \; q.\text{QResults[i].RTitle} \cap Q.\text{QResults[j].RTitle} \neq \emptyset\}$$

Experiments in the paper were, as in the previous case, conducted on a user study. They collected more than half a million different queries from the Meta crawler search engine. From this set of queries they extracted and harvested the results of a subset of 70,000 queries (which constitutes the Query Memory). Submitting queries to the search engine and harvesting result is, indeed, very computationally and network intensive. More realistically, since the query recommender is usually on the search engine side, not only the query trace would be immediately available, but also the inverted indexes of the

search engine would also be available avoiding the submission of queries for results harvesting.

The most interesting result claimed is that "it was difficult to find a winner (i.e., the similarity measure with the best recommendation)" [9]. Indeed, from a general perspective, the main reason for this to hold, could be the fact that the validation process is a very subjective one.

One of the most interesting observation made by [9] is on the scalability of their approach to a real-world setting. Actually the query memory is a DB for queries on which depends the effectiveness of a recommendation method. The richer the query memory, the better the suggestions computed. Yet, the way query records are stored is a crucial point and the paper does not discuss it as deep as it should. Searching for queries containing a given keyword, or keywords, would require in real world systems an index per-se, making the methods paying a double index access for each submitted query. Obviously, these two accesses can be done in parallel using a distributed architecture.[8] use a clustering approach to query recommendation. The query recommender algorithm operates using a two tiered approach. An offline process clusters past queries using query text along with the text of clicked URLs. The online process follows a two-stage approach: (i) given an input query the most representative (i.e., similar)cluster is found; (ii) each query in the cluster is ranked according to two criteria: the similarity and the attractiveness of query answer, i.e., how much the answers of the query have attracted the attention of users (this is called support in their paper and should not be misinterpreted as support of association rules [11]). Interestingly this work has a lot of points in common to the one of [12] that has another (quite different) purpose. Enhancing the effectiveness of collection representation in collection selection functions.

The offline query clustering algorithm operates over queries enriched by a selection of terms extracted from the documents pointed by the user clicked URLs. Clusters are, thus, computed by using an implementation of the ***k-means*** algorithm [13] contained in the CLUTO software package. The similarity between queries is computed according to a vector-space approach. That is, each query ***q*** is represented as a vector ***_q*** whose ***ith*** component ***qi*** is computed as

$$q_i = \sum_{u \in URLs} \frac{\text{Clicks}(q,u) \times \text{Tf}(t_i,u)}{\max_t \text{Tf}(t,u)}$$

where:

• ***Clicks(q,u)*** is th e percentage of clicks URL u receives when answered in response to the query ***q***.

• ***Tf(ti,u)*** is the number of occurrences of the term t in the document pointed to URL ***u***.

The sum is computed by considering all the clicked URLs for the query ***q***. The distance between two queries is computed by the cosine similarity of their relative vectors.

The ***k-means*** algorithms is sensitive to the value of parameter ***k***, i.e., the number of computed clusters. To measure the optimal number of cluster an empirical evaluation measuring the compactness of clusters have been performed. A run of a k-means algorithm is executed with different values of ***k***. Each clustering result is assigned to a function computing the total sum of the similarities between the vectors and the centroids of the clusters that are assigned to. For the considered dataset the number of query cluster has been set to ***k*** = 600. The query log (and the relative collection) they use, comes from the TodoCL search engine and contains 6,042 unique queries along with associated click-through information. 22,190 clicks are registered in the log spread over 18,527 different URLs.

The experimental evaluation of the algorithm is performed on 10 different queries. Evaluation is done again by a user study. The evaluation assesses that presenting query suggestions ranked by support (that is the frequency with which the query occur in the log) yields to more precise and high quality suggestions.
[14] have proposed a model for generating queries to be suggested based the concept of query rewriting. A query is rewritten into a new one by means of query or phrase substitutions (e.g. the query "*leopard installation*" can be rewritten into "*mac os x 10.5 installation*"). The rewriting process is based on a log-likelihood ratio (LLR) measure to evaluate interdependencies between terms of queries. The metric tests the hypothesis that the probability of seeing a term ***q2*** in a query is the same whether or not the term ***q1*** has been seen. Basically, the hypothesis ***H1:P(q2|q1) = p = P(q2| ¬q1)*** is tested against the hypothesis ***H2:P(q2|q1) = p1 ≠ p2 = P(q2| ¬q1)***. The LLR is a statistical test for making a decision between two hypotheses based on the value of this ratio [15]. The LLR is computed as

$$\text{LLR} = -2 \log \frac{L(H_1)}{L(H_2)}$$

and large values of the ratio means that the likelihood of the two words of being dependent is higher. From observations, query pairs with high log likelihood ratio are identified as substitutables. Furthermore given that the likelihood ratio, computed as ***λ = (L(H1)/L(H2))***, is

distributed as a $\chi 2$, a score of 3.84 for log likelihood ratio gives a 95% confidence in rejecting the null hypothesis (therefore, the relation between the two terms (or phrases) is statistically significant). Indeed a 95% confidence means 1 in 20 not correctly detected substitutable. For this reason in [14] they set the log likelihood ratio threshold to a high level of 100 thus making the method highly selective and precise. Indeed, since they are dealing with logs of millions of queries even with such a high threshold level the number of candidates found is acceptable.

A noteworthy point of the paper is the systemization of possible suggestion into four classes going from the most precise, down to the less precise class: precise rewriting, approximate rewriting, possible rewriting, and clear mismatch. Precise rewriting (class 1) means that the query suggested has exactly the same semantics of the one to be replaced (e.g. automotive insurance is a precise rewriting of automobile insurance). In approximate rewriting (class 2) the scope of the initial queries is narrowed or broadened (e.g. ipod shuffle is a more narrow of, but still related to apple music player). Possible rewriting (class 3) is a still less precise query suggestion methodology: queries are either in some categorical relationship (e.g. eye-glasses and contact lenses), or describes a complementary object (e.g. orlando bloom vs. johnnydepp). The last class (4), clear mismatch, is the less precise and contains query pairs where no relationships can be found among them.

Using these four classes we can identify the two general tasks of query suggestions, namely Specific Rewriting (or 1+2), and Broad Rewriting (or 1+2+3). The former consists in generating suggestions of the first and second class, the latter consists in generating suggestions of the first, second, and third class. Given these tasks, the problem of generating query suggestion can be reformulated into a classification problem.

In the original paper many classifier have been tested. Among the others, a linear classifier is used. Training is performed on a manually annotated set of substitutions extracted from queries of a query log. The extraction is made on the basis of the log likelihood ratio defined above, and the features considered are of three different types: characteristics of original and substituted query in isolation (e.g. length, character encoding, etc.); syntactic substitution characteristics (edit distance, number of tokens in common, etc.); and substitution statistics (e.g. log likelihood ratio, $P(q2|q1)$, etc.)

We omit the details on how the classifier has been trained and tested and on how data has been prepared. We report, here, only the main results. Interested readers are encouraged to read the whole paper by [14] to get the full picture of their technique. The learned ranking function is given by

$$f(q1,q2) = 0.74 + 1.88\ editDist(q1,q2) + 0.71\ wordDist(q1,q2) + 0.36\ numSubst(q1,q2)$$

and it is computed for query pairs whose log likelihood ratio is above an empirical threshold set to 100 (i.e., LLR > 100).

One of the main advantage of the rewriting approach is that it is possible to experimentally assess the quality of suggestions without recurring to user surveys. For the log used in [14], the precision of the method has been measured to be 76% with a recall figure of 92%.

As in the case of query expansion, query suggestion might suffer of the same problem due to different tastes of users. Next paragraph addresses exactly this aspect and shows how information on past queries and more specifically on who submitted them can be used to create results tailored on a particular user category.

As a final note, it is worth mentioning a body of literature that recently has started to consider link recommendations instead of query suggestions to enhance the effectiveness of a web search engine. Basically, the idea is as follows: use click-through information to infer what users are looking for and instead of suggesting a possible query suitable for finding it, suggest immediately the site they were potentially looking for. Typical mechanisms for achieving this are quite similar to those shown above for query suggestion. The interested readers are encouraged to read [16,17,18,19] as a starting point.

A slightly different problem, yet related with query recommendation is the problem of finding "Query Shortcuts" [20,21,22]. Authors define formally the Search Shortcut Problem (SSP) as a problem related with the recommendation of queries in search engines and the potential reductions obtained in the users session length. This new problem formulation allows a precise goal for query suggestion to be devised: recommend queries that allowed in the past users that used a similar search process, to successfully find the information they were looking for. The approach take so far for the approximation of the solution of the SSP is based on the adoption of collaborative filtering techniques. Authors point out to some limitations of traditional algorithms, so

future work should develop new algorithms specially designed for this new problem.

4. Experiments and Results

There are many Query Log mining techniques for enhancing the search experience of web search engine users in term of effectiveness.

4.1 Experimental Evaluation for a Query Expansion Technique

One of the first works exploiting past usage information to expand web queries is " Probabilistic Query Expansion Using Query Logs" presented by [3]. In this section, we report the experimental results on the performance of the log-based probabilistic query expansion method[3].

4.1.1 Data

[3] used the same two-month query logs (about 22 GB) from the Encarta search engine (http://encarta.msn.com), as well as the 41,942 documents in the Encarta website, which contains 4,839,704 user query sessions. The documents collection is made up of 41,942 Encarta documents with various topics. The lengths of the documents also vary greatly, from dozens of words to several thousand words.

A total of 30 queries are used to conduct the experiments. Some queries are extracted randomly from the query logs. Some others come from the TREC query set. Yet another subset of queries are added manually by [3]. The queries in our experiments are very close to those employed by the real web users and the average length of all queries is 2.1 words. Figure 4.1 lists the 30 queries used in the experiments.

1 Java computer	2 nuclear submarine	
3 Apple computer	4 Windows	5 fossil fuel
6 cellular phone	7 search engine	
8 Six Day War	9 space shuttle	
10 economic impact of recycling tires		
11 China Mao Ze Dong	12 atomic bomb	
13 Manhattan project	14 Sun Microsystems	
15 Cuba missile crisis	16 motion pictures	
17 Steve Jobs	18 pyramids	19 what is Daoism
20 Chinese music	21 genome project	
22 Apollo program	23 desert storm	
24 table of elements	25 Toronto film awards	
26 Chevrolet truck	27 DNA testing	
28 Michael Jordan	29 Ford	30 ISDN

Figure 4.1 The experimental query set [3].

4.1.2 Quality of Expansion Terms

[3] examined the top 50 expansion terms for all the 30 queries to check the relevance of the expansion terms. The Table 4.1 shows the number of relevant expansion terms suggested by the local context analysis (LC Analysis) [23] and our log-based method (Log Based). As we can see, our method is 32.03% better. Figure 4.2 illustrates the top 50 expansion terms for the query "Steve Jobs" by our method. Some very good terms, such as "personal computer", "Apple Computer", "CEO" , "Macintosh", even "graphical user interface", "Microsoft" can be obtained by our techniques.

	LC Analysis (base)	Log Based	Improvement (%)
Relevant Terms (%)	23.27	30.73	+32.03

Table 4.1 Comparison on relevant percentage of expansion terms (Top 50 terms) [3].

4.1.3 Retrieval Effectiveness

[3] compare the retrieval performance of the log-based query expansion with the baseline (without query expansion) and the local context analysis method. Interpolated 11-point average precision is employed as the main metric of retrieval performance. Statistical paired t-test [24] is also used to determine the significance of differences. For the local context analysis, the default is to use 30 expansion terms, which include words and phrases, from 100 top-ranked documents for query expansion. The smoothing factor δ in local context analysis is set to 0.1 here, as proposed in [4].

For the log-based query expansion, [3] use 40 expansion terms. The expansion terms are extracted from top 100 relevant documents according to the query logs. Phrases that appear in the query logs are assigned a parameter S, which is 10 in our experiments.

We see that [3] log-based query expansion performs well on the experiments. It brings an improvement of 75.42% improvement over the baseline method, while the local context analysis achieves a 26.24% improvement over the baseline. Log-based query expansion also provides an average improvement of 38.95% compared to local context analysis, which is also statistically significant. Generally, log-based query expansion selects expansion terms from a relatively narrower but more concentrated area. In contrast, local context analysis searches expansion terms in the top-ranked retrieved documents and is more likely to add some irrelevant terms into the original query, thus introducing undesirable side effects on retrieval performance.

1. Apple	2. *personal computer*	3. *Computers*
4. *personal computers*	5. *Apple Computer*	
6. *operating system*	7. *Newton*	
8. *graphical user interface*	9. graphical user	
10. *Software*	11. user interface	
12. programming language	13. programming languages	
14. *computer*	15. wozniak	
16. CPU	17. operating systems	
18. mainframe computer	19. personal	
20. principia	21. jobs	
22. *CEO*	23. company	
24. computer systems	25. high-level	
26. assembly language	27. machine language	
28. computer system	29. *Gates*	
30. analog	31. circuit board	
32. vice president	33. opticks	
34. analytical engine	35. *Microsoft*	
36. jacquard	37. output devices	
38. Halley	39. woolsthorpe	
40. output device	41. Calculus	
42. input devices	43. *Lisa*	
44. Pixar	45. first computer	
46. Paul Allen	47. white light	
48. *Macintosh*	49. slide rule	50. markkula

Figure 4.2 Expansion terms for "Steve Jobs" [3].

Recall	Baseline	LC Exp	On Log Exp
10	40.67	40.33(-0.82)	62.00(+52.46)
20	26.83	33.33(+24.22)	44.67(+66.46)
30	21.56	27.00(+25.26)	37.00(+71.65)
40	17.75	23.08(+30.05)	31.50(+77.46)
50	15.07	20.40(+35.40)	27.67(+83.63)
60	13.00	17.89(+37.61)	24.56(+88.89)
70	11.43	16.29(+42.50)	22.24(+94.58)
80	10.17	15.08(+48.36)	20.42(+100.82)
90	9.44	13.96(+47.84)	18.89(+100.00)
100	8.70	13.07(+50.19)	17.37(+99.62)
Average	17.46	22.04(+26.24)	30.63(+75.42)

Table 4.2 A comparison of retrieval performance of Baseline, query expansion base on local context analysis (LC Exp), and log-based query expansion (On Log Exp). All experiments are done with phrases included [3].

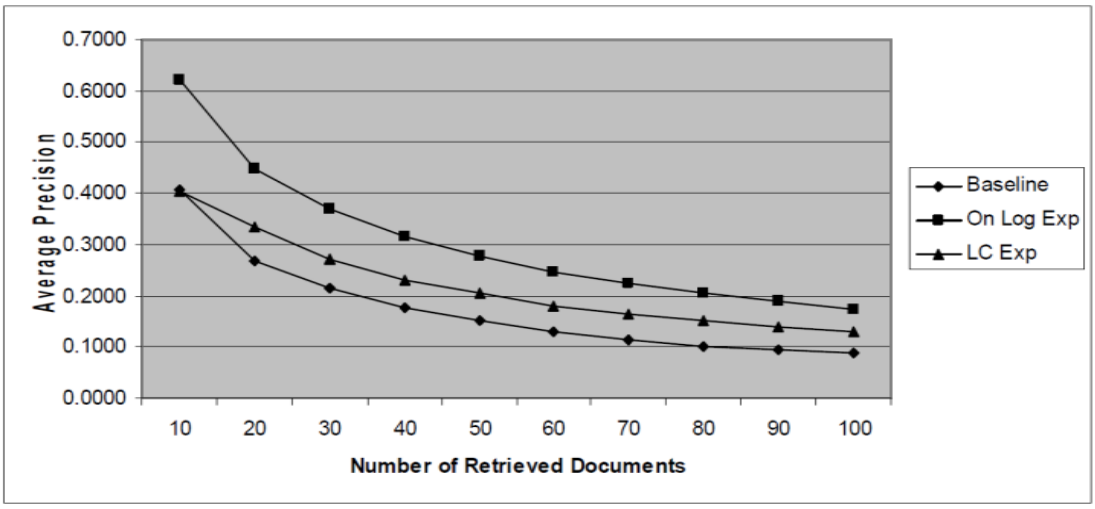


Figure 4.3 Average precision for Baseline, LC Exp and On Log Exp (with phrases) [3].

4.1.4 Impact of Number of Expansion Terms

In general, the number of expansion terms should be within a reasonable range in order to produce the best performance. Too many expansion terms not only consume more time in retrieval process, but also have side effects on retrieval performance. [3] examine performance by using 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 expansion terms for retrieval. Table 4.3 and Figure 4.4 show the impact on average precision by the number of expansion terms. The best performances are obtained within the range of 40 and 60 terms. The performance drops when the number of expansion terms is larger than 70, which indicates that the terms beyond 70 are less relevant to the original query.

Number of Expansion Terms	Precision
10	0.271
20	0.294
30	0.303
40	0.306
50	0.306
60	0.308
70	0.304
80	0.304
90	0.302
100	0.296

Table 4.3 Comparison of performance using various number of expansion terms [3].

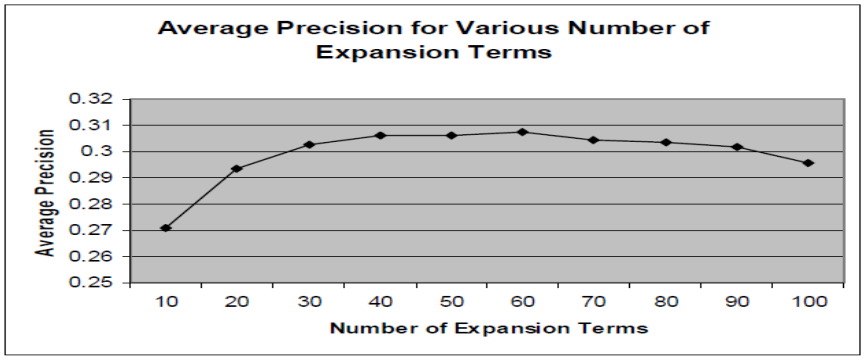


Figure 4.4 Impact of various number of expansion terms [3]

4.2 Experimental Evaluation for a Query Suggestion Technique

In this section, we report the experimental evaluation results of "Query Recommendation using Query Logs in Search Engines" presented by [8] that use a clustering approach to query recommendation. In [8] experiments ,they consider ten queries. 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 4.5 shows the clusters to which the queries belong. In addition, [8] 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%)

Figure 4.5 Clusters for the experiment [8].

Figure 4.6 shows the ranking suggested to Query 3 (Chile rentals).

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

Figure. 4.6 Ranking of queries recommended to the query Chile rentals [8].

In order to assess the quality of the results, [8] follow a similar approach to [10]. The relevance of each query to the input query were judged by members of [8] department. They analyzed the answers of the queries and determined the URL's in the answers that are of interest to the input query. [8] results are given in graphs showing precision vs. number of recommended queries.

5. Conclusions and Future Work

This work has covered some of data mining techniques for mining query logs in web search engines for enhancing the search experience of web search engine users in term of effectiveness. Effectiveness in search systems refers to the quality of returned results. We covered only the major techniques that are mainly related to query expansion and query recommendation of five main applications that can be enhanced : i) query expansion, ii) query recommendation, iii) personalized query results, iv) learning to rank, and v) query spelling correction. We show the experimental evaluation for some of the query log mining techniques in search engines under each discussed query log mining applications: query expansion and query suggestion respectively.

As future work we intend to expand this work to make it include all possible query logs mining techniques to enhanced the web search engines effectiveness in terms of query results, learning to rank, and query spelling correction. Furthermore, we want to study how usage patterns in web search engine logs can be exploited to design effective methods for enhancing efficiency of the web search engines. Finally, it would be interesting to develop a framework for mining query logs in web search engines.

References

[1] Marcelo Rocha, "Query Log Mining in Search Engines", Department of Computer Science , University of Chile, June 2007.

[2] FabrizioSilvestri. Mining query logs: Turning search usage data into knowledge. Foundations and Trends in Information Retrieval, 1(1-2):1{174, 2010.

[3] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, "Probabilistic query expansion using query logs," in WWW '02: Proceedings of the 11th International Conference on World Wide Web, pp. 325–332, New York, NY, USA: ACM, 2002.

[4] J. Xu and W. B. Croft, "Improving the effectiveness of information retrieval with local context analysis," ACM Transactions on Information Systems, vol. 18, no. 1, pp. 79–112, 2000.

[5] B. Billerbeck, F. Scholer, H. E. Williams, and J. Zobel, "Query expansion using associated queries," in Proceedings of the twelfth international conference on information and knowledge management, pp. 2–9, ACM Press, 2003.

[6] K. S. Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: Development and comparative experiments," Information Processing and Management, vol. 36, no. 6, pp. 779–808, 2000.

[7] F. Scholer, H. E. Williams, and A. Turpin, "Query association surrogates for web search: Research articles," Journal of the American Society for Information Science and Technology, vol. 55, no. 7, pp. 637–650, 2004.

[8] R. Baeza-Yates, C. Hurtado, and M. Mendoza, Query Recommendation Using Query Logs in Search Engines. pp. 588–596. Vol. 3268/2004 of Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, November 2004.

[9] O. R. Za¨ıane and A. Strilets, "Finding similar queries to satisfy searches based on query traces," in OOIS Workshops, pp. 207–216, 2002.

[10] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani, "Using association rules to discover search engines related queries," in LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress, p. 66,Washington, DC, USA: IEEE Computer Society, 2003.

[11] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26–28, 1993, (P. Buneman and S. Jajodia, eds.), pp. 207–216, ACM Press,1993.

[12] D. Puppin and F. Silvestri, "The query-vector document model," in CIKM '06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 880–881, New York, NY, USA: ACM, 2006.

[13] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," ACM Computing Surveys, vol. 31, no. 3, pp. 264–323, 1999.

[14] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in WWW '06: Proceedings of the 15th International Conference on World Wide Web, pp. 387–396, New York, NY, USA: ACM Press, 2006.

[15] C. D. Manning and H. Sch¨utze, Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press, 1999.

[16] M. Bilenko and R. W. White, "Mining the search trails of surfing crowds: Identifying relevant websites from user activity," in WWW '08: Proceeding of the 17th International Conference on World Wide Web, pp. 51–60, New York, NY, USA: ACM, 2008.

[17] R. W. White, M. Bilenko, and S. Cucerzan, "Studying the use of popular destinations to enhance web search interaction," in SIGIR '07: Proceedings of the 30th

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 159–166, New York, NY, USA: ACM, 2007.

[18] R. W. White, M. Bilenko, and S. Cucerzan, "Leveraging popular destinations to enhance web search interaction," ACM Transactions on the Web, vol. 2, no. 3, pp. 1–30, 2008.

[19] R. W. White and D. Morris, "Investigating the querying and browsing behavior of advanced search engine users," in SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 255–262, New York, NY, USA: ACM, 2007.

[20] R. Baraglia, F. Cacheda, V. Carneiro, F. Diego, V. Formoso, R. Perego, and F. Silvestri, "Search shortcuts: A new approach to the recommendation of queries," in RecSys '09: Proceedings of the 2009 ACM Conference on Recommender Systems, New York, NY, USA: ACM, 2009.

[21] R. Baraglia, F. Cacheda, V. Carneiro, V. Formoso, R. Perego, and F. Silvestri, "Search shortcuts: Driving users towards their goals," in WWW '09: Proceedings of the 18th International Conference on World Wide Web, pp. 1073–1074, New York, NY, USA: ACM, 2009.

[22] R. Baraglia, F. Cacheda, V. Carneiro, V. Formoso, R. Perego, and F. Silvestri, "Search shortcuts using click-through data," in WSCD '09: Proceedings of the 2009 Workshop on Web Search Click Data, pp. 48–55, New York, NY, USA: ACM, 2009.

[23] Attar, R. and Fraenkel, A.S. 1977. Local feedback in full-text retrieval systems. J. ACM 24, 3 (July), 97-417.

[24] Hull, D., 1993, Using statistical testing in the evaluation of retrieval experiments. In Proceedings of the ACM SIGIR, pages 329--338, Pittsburgh, PA, June 1993.