An Integrated Architecture for Personalized Query Expansion in Web Search

Alexander Salamanca and **Elizabeth León**
Data Mining Research Group - MIDAS
School of Engineering
National University of Colombia
Bogotá D.C., Colombia
{pasalamancar, eleonguz}@unal.edu.co

Abstract

In this paper we present an integrated architecture to perform personalized interactive query expansion in Web search. Our approach is to extract expansion terms in a three stage cycle: 1) keyword extraction with local analysis on search results, 2) keyword extraction with a recommender system on a community of users and 3) an algorithm to personalize the final list of suggestions. Three methods for local analysis and recommender models are presented and evaluated. The user profile is built with latent semantic analysis on search sessions. We prepared a set of experimental scenarios with user studies in order to assess the system. The results obtained shows good performance on the perceived quality by users on suggested terms.

Introduction

Personalization had been seen as one of the most promising trends in the near future for improving significantly the enjoyment of the search experience on the Web. The main idea is deliver quality results prepared uniquely for different users, which do not share necessarily the same interests on the long term with another people.

The approach described in this paper exploits relations between keywords of the user's search history and a more general set of keywords that expand the user's search scope.

Through a three stages cycle, which consists of extracting key terms by local analysis, extraction of other key terms through a system of automatic recommendation and an algorithm to personalize the final list of terms suggested.

Thus, we can produce high quality and relevant query suggestions reformulations of the user intent –verbalized as a Web query– that increases the chances of retrieving better results.

This paper is organized as follows: section 2 introduces related work. Section 3, describes the proposed approach; section 4, presents system validation; and finally, in section 5 conclusions are presented and we give guidelines for future research.

Related Work

Query expansion is the process of adding additional terms to a user's original query, with the purpose of improving retrieval performance (Efthimiadis 1995). Although query expansion can be conducted manually by the searcher, or automatically by the information retrieval system, the focus here is on interactive query expansion which provides computer support for users to make choices which result in the expansion of their queries.

A common method for query expansion is the relevance feedback technique (Salton 1979), in which the users judge relevance from results of a search. This information then is used to modify the vector-model query with increased weights on the terms found in the relevant documents. In (Salton and Buckley 1990), these techniques were proved to be effective. In (Efthimiadis 1995) is presented a comprehensive literature review to extract keywords based on term frequency, document frequency, etc.

Terms highly co-occurring with the issued keywords have been shown to increase precision when appended to the query (Kim and Choi 1999). Several techniques had been used to measure the text-term relationship level, either analyzing entire documents (Qiu and Frei. 1993; Kraft et al. 2006), lexical affinity relationships (Carmel et al. 2002), or formal concept analysis (Ducrou and Eklund 2007; Carpineto and Romano 2004) among others.

There are many other attempts to extract expansion terms from logs as a source data useful for this strategy (Ruthven, Lalmas, and van Rijsbergen 2003). For example, in (Billerbeck et al. 2003), large query logs are used to construct a surrogate for each document consisting of the queries that were a close match to that document. Baeza-Yates et al. (Baeza-Yates, Hurtado, and Mendoza 2004) recommend queries building a term-weight vector for each query. Each term is weighted according to the number of document occurrences and the number of selection of the documents in which the term appears. Then they use clustering techniques to identify related queries.

Instead, we investigate methods for analyzing and processing the initial search results, and allowing the specialization or generalization of the original query. This use of the data present in the top search results is called local analysis(Manning, Raghavan, and Schütze 2007). Also, we combine that information with a more flexible strategy, fo-

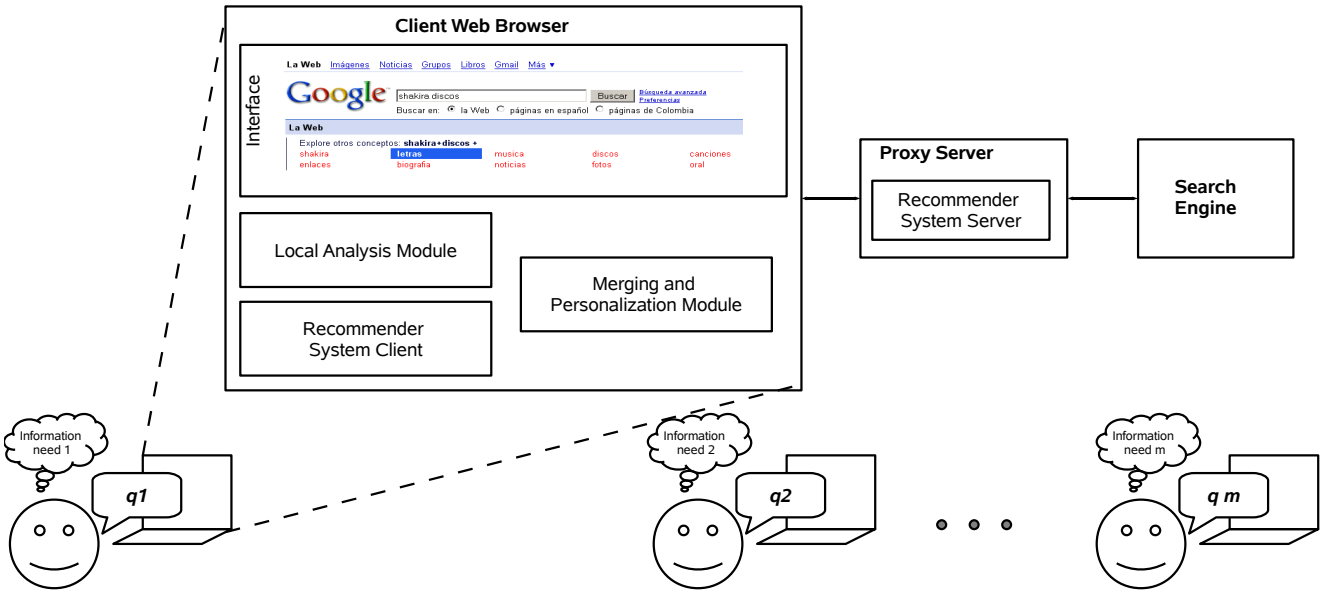

Figure 1: System architecture

cused on the identification of alternate queries used by another people in a similar information context. Finally, recognizing that people have unique preferences in the long run, we provide a method for merging results of local analysis and recommendation approaches in a personalized manner based on the query terms previously used by the user.

Approach

The characteristic of the Web queries as established in previous work is that they are generally short and ambiguous. The Web browsing and searching behavior observed in many studies complements this nature of search emphasizing the recurrent and limited vocabulary employed to represent the information need to be solved.

Previous studies in (Teevan, Dumais, and Horvitz 2005) had shown that the results lists of commercial search engines are very optimized for many common queries, but also had been identified some kind of search sessions where the user is clearly not satisfied with the original results provided by the engine for the first query, hence, the user may be forced to modify the original query adding or removing or changing the keywords that shape the information need behind his intent until the result set is satisfactory. Therefore, we observe that is missing some way to reinterpret the user information need in order to put it in terms of the vocabulary used for that concept beyond the language limits of the user.

The goal of the system is embody a technique to generate a handful of terms to re-express or to wide or to narrow the initial scope of the Web query. Using the intrinsic relations within the keywords extracted from the user behavior, in some meaningful way, and matching them with the keywords obtained from the local analysis of the Web search results to derive the desired terms is possible to get the desired result.

Aiming exploratory search scenarios, the system make use of the advantages of these ideas: with local analysis, the key terms are good to detect information descriptors of the results allowing fast rewriting of the original query, directing the search towards the desired conceptual space; with the recommender system, the key terms are from a different nature and they can make different functions: they can enrich the user vocabulary as well as they can show novel descriptors of seemingly disconnected concepts but related in the ground; the personalization is performed at the query refinement stage and not in the presentation result because we do want to preserve the original result list (besides relevance built with diversity, freshness, among other criteria) and we want to give to the user a sense of control with an interactive personalization scheme provided by the list of suggestions. Also, taking in account regular and previous behavior we provide a succesful way to tackle the problem of refinding (Teevan 2007).

Architecture

The prototype system architecture is depicted in the figure 1, where the component modules are presented: a module for extraction of terms with local analysis over the search engine result set (run from the web browser), a module for extraction of suggestions with a recommender system over a community of searchers (run from a proxy server or the search engine itself) and finally a module for personalized posprocessing and merging that use the user profile to reposition, if needed, the list of terms, to user preference.

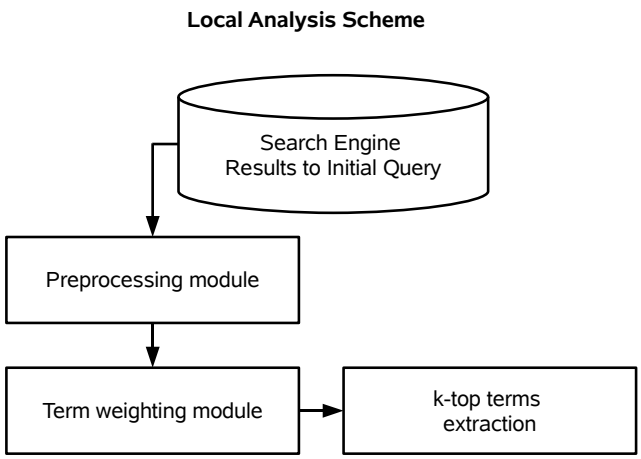

Figure 2: Local Analysis Module

Local Analysis

The local analysis module is outlined in the figure 2, where the weighting module can be changed according to the methods presented here. All they are based on vector model representation of the search engine results for a given query and calculate matrices term document. Henceforth, a document is another name for the surrogate that the search engine makes for a web result (title + summary). In this way, at best, we may consider up to 100 documents for processing. Stop words are discarded. Here are the specifics for each representation.

In order to detect key terms in a collection of documents with many topics, several techniques had been used with the goal of quantify the relative contribution of terms. An intuitive manner to measure relevance of terms, is calculating frequencies of appearance. However, it has been determined that prevalent relationships for frequency and usefulness in this method of measuring importance are: too much frequent terms are not valuable; mid frequent terms can be really useful; very infrequent terms are useful insofar as it can discriminate relevant documents (Efthimiadis 1995).

TFIDF A classic model of term frequencies, where relations between term frequency and document inverse are calculated as shown in equation 1.

With this model, for every term, a weight is calculated trying to moderate the contribution of too much frequent terms taking in account the number of documents that contain the term.

$$w_i^{TFIDF} = \sum_j \frac{f_{i,j}}{\max_z f_{z,j}} \times \log \frac{N}{n_i} \tag{1}$$

LSA A vector model based on spectral decomposition of the TFIDF matrix and recomposition to k-latent dimensions. Let $A = a_{i,j} = TF_{i,j} \times IDF_i$, we project to a space $k = \kappa$, trying to estimate hidden structure behind the concepts. Where κ is at most up to 30% of the contribution of singular values (proportion determined empirically) of A. With the latent semantic analysis we get the matrix B. The latent semantic analysis(Deerwester, Dumais, and Harshman 1990) uses singular value decomposition (SVD), $A_{(t\times d)} = U \times \Sigma \times V^T$, to discovery co-occurrence implicit associations. This method can be considered like a soft clustering method if we interpret each dimension of the reduced space like a group, and the value of the document like a fractional membership level to that group. The matrix Σ consists of entries are the singular values of A. Then, if the matrices U and V are "cut" to the first k columns, and Σ is left with range k, we can reconstruct an approximation of A. Let, $A \approx B = b_{i,j} = U_{(t\times k)} \times \Sigma_{(k\times k)} \times V^T_{(d\times k)}$. Where the $k = \kappa$ value is chosen according to $\frac{\sum_{i=1}^{\kappa} \sigma_i}{\sum_i \sigma_i} < 30\%$. Finally, the weighting vector, equation 2, is calculated in a similar way to TFIDF:

$$w_i^{LSA} = \sum_j b_{i,j} \tag{2}$$

A prevalent difficulty with LSA is the complexity of SVD, because is cubic in the size of the matrix. Thats why we consider only the first search summaries to be feasible to calculate from the client side nearly in real-time.

DFR A language generative model (Amati and Rijsbergen 2002) known as divergence from randomness, based on this simple idea: The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word w in the document d. To be more specific, is a combination of the following information measures:

$$C = c_{ij} = \text{Inf}^1_{ij} \times \text{Inf}^2_{ij} = -\log_2 \left[\text{Prob}^1_{ij}\right] \times \left(1 - \text{Prob}^2_{ij}\right)$$

Where each term of the above equation are calculated as:

$\text{Prob}^1_{ij} = \left(\frac{df_i + 0.5}{n+1}\right)^{tfn_{ij}}$, the probability of finding by pure chance, tf_{ij} occurrences of term i in the document j; and the probability of finding a new occurrence of term i in the document j, given that tf_{ij} occurrences have been found, $\text{Prob}^2_{ij} = 1 - \left(\frac{tc_i+1}{df_i \times (tfn_{ij}+1)}\right)$ and the term tfn_{ij}, calculated as $tfn_{ij} = tf_{ij} \times \log_2 \left(1 + \frac{c \cdot \text{mean dl}}{l_j}\right)$, where c is a constant established in 1.5, the factor l_j is the document length and mean dl is the average document length. At the end, we also use the same model for the extraction of a column vector with the relative importance of terms in the collection, see equation 3.

$$w_i^{DFR} = \sum_j c_{i,j} \tag{3}$$

The DFR model, assign a vector representation different with respect to the induced calculating frequencies only. In this, the more informative words are considered that they belong to an elite set of documents, where they distribute relatively in a greater extent than in the rest of the documents. On the other hand, there are words, which do not possess elite documents, and thus their frequency follows a random distribution, that is the single Poisson model (Amati and Rijsbergen 2002).

In the algorithm 1 is described the specific steps for the local analysis. The main idea for considering the sum as proposed in all three methods is that weighting scores represent a notion of importance in the collection.

Algorithm 1 Term extraction with local analysis

The search result summaries for a query are processed
Calculate the vector-space representation following any method between
Extract k-terms with most weight

Recommendation approach

For obtaining the terms suggested by a community of users, we use the approach described below. We compared the following approaches. First at all, a baseline which output terms based on their frequency through sessions. Then, we compared it against ideas from the model proposed by Dupret in (Dupret and Mendoza 2006), and two models proposed in this document. One model based on measures of co-occurrence using of query terms through sessions from (Matsuo and Ishizuka 2004) (originally designed for keyword extraction in a single document), and finally, a model based on measures of similarity. Here we explain more in detail the three models, where the rationale is, how to find suggestion terms, used by another people in a similar context and where they give them successful results.

It's noted that this component acts at a point where it could access the web behavior information activity from a relatively large community. For each one of the models presented here, we used the dataset from (Salamanca, González, and León 2007). It is composed of data collected from the proxy server at the National University of Colombia, since August 2006 to January 2008 in a spanning through 52 weeks (non consecutive), and it records Web behavior of about 52,000 users. With this dataset, were performed some basic preprocessing steps, such as cleaning, filtering and calculation of browsing sessions. They are defined as sets of activities within an inactivity time threshold (defined empirically).

On this sessions, searching sessions were extracted and we discarded entries with values in attributes like domains and query terms outside of frequency limits. This was performed in order to leave out domains and query terms too much (in)frequents. The approaches explained here are indifferent to whether they are executed directly in the search engine itself or if they are executed in an intermediate server (like a proxy) between many clients and the search engine.

Dupret The thought behind this method is how to find better queries on alternate queries with the same selected documents and get only the ones with improved ranking. We modified the original approach in the sense that we are only interested in query terms, not in complete queries.

Co-occurrence through sessions Calculating a co-occurrence matrix of query terms through sessions, and then, on the most frequent query terms co-occurring with terms in the initial query, a chi-square statistic, modified as can be seen the equation 4 is used to select the query terms with highest $\chi^{'2}$ value.

$$\chi^{'2}(w) = \sum_{c \in G} \left\{ \frac{(freq(w,c) - n_w p_c)^2}{n_w p_c} \right\} - \max_{c \in G} \left\{ \frac{(freq(w,c) - n_w p_c)^2}{n_w p_c} \right\} \quad (4)$$

where G refers to set of most frequent terms, p_g is the sum of the total number of query terms in sessions where g appears, divided by he total number of query terms in the whole log, and n_w is the total number of terms in sessions where w appears.

k-NN VPTree In the figure 3 the involved components in the recommendation method are shown. First at all, a preprocessing stage is needed in order to detect sessions of related objects of interest for users. With a binary representation of domains visited by the people we can use the Jaccard dissimilarity coefficient as a distance function, because this is suitable with asymmetric attributes, and in the distribution of frequency shows a very skewed shape. The problem is equivalent to find the k-nearest neighbors in a metric space defined by 5. For the implementation we use python and the metric spaces library by Jochen Schulz[1] in a space of 65434 terms with 1442 domains as dimensions. For efficiency reasons, we used a special data structure for the calculations of similarity in a metric spaces, the VPTree(Yianilos 1993).

A "Vantage Point Tree" is a static, balanced, binary tree which can be used to index metric spaces. Construction is done by picking one object (the "vantage point") from the set of objects to index and then determining the distance of all remaining objects to the vantage point. The median distance is determined and the set of remaining objects is split in two parts: objects whose distance to the vantage point is smaller than the median distance and the rest (objects whose distance to the vantage point is larger than or equal to the median). For both of the resulting sets new child nodes, called "left" and "right", are created recursively. The vantage point and the previously calculated median distance are stored in the current node. When searching a node, the distance between the current node's value and the search object is calculated. If it is $<= k$ (the given maximum distance), this node's value is added to the list of search results. Searching proceeds recursively into the left subtree if $distance - k < median$ and into the right subtree if $distance + k >= median$. Since the construction phase takes advantage of the median of all distances, VPTrees do a fairly good job of balancing their subtrees. Runtime complexity for the construction phase is $O(nlog(n))$, searching is $O(log(n))$ and space requirement is $O(n)$.

$$J_\delta(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \quad (5)$$

[1] svn://svn.well-adjusted.de/mspace/branches/nn-query

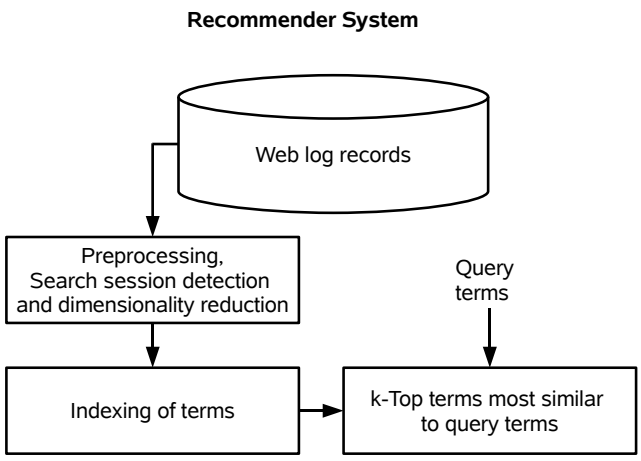

Figure 3: Recommendation system module

Personalization of term list

As in every personalization effort is required the evidence of past behavior, therefore we present our approach to capture the user profile.

This is how the search works from the user's perspective: first, the user raises their need for information and as verbalization let it be, q. Then, the search engine returns a set S of summaries of the relevant documents found.Each portion of text generated (title and summary) s_i is an approximate descriptor generated by the search engine on the content of the page. These summaries usually are built to put the terms of search in context within the page content, and this discussion deliberately ignore the algorithm that uses the search engine for deriving such abstracts, aware of the bias that can imply.

Often, the user scans very quickly the whole 10 snippets and decides to select a few results (usually 1 or 2) to satisfy the information need it has. Usually, scanning on the snippets that is more than enough to determine the likelihood of relevance (i.e. there is correlation between what the user deems as important and the fact that it is really important for that user in his context).

Therefore in this study we decided to make a representation bag of words collecting terms that the user employs for querying $w_i \in q$, and the words coming out in summaries of the search engine for pages selected as relevant by the user, excluding repetition of terms used in the query. This vector terms of what we to consider as a "document".

So, sets of "documents"are associated under a logical structure, it was decided to adopt as a criterion of selection taking those who are in a session, which corresponds to a series of events defined by a threshold of inactivity in an amount of given time and given the caveat that although there were sessions with multiple objectives here we suppose that they only deal with related topics.

With this group of "document vectors", in the next is explained of to calculate a summary of overall preferences for the user:

Once all the keywords most prominent have been extracted, these are moderated by the weights assigned to the profile vector P, and thus provides the user with relevant suggestions terms of expansion. The personalization is performed when we weight the set of main keywords using the respective weights from the vector P. If the word does not exist in the profile P, we weight it with a weight of $\inf$. The ranked list of suggestion terms is presented to the user ranked by their weights. In the algorithm 3 is summarized the complete process.

Algorithm 2 Preferences calculation algorithm

If the set of session matrices B are preprocessed by LSA as explained before (with a κ with up to 10% of the contribution from the singular values) aligned by rows (terms) and we sum all the weights through the columns and then normalize it we can derive a column vector P, which provides a level of importance of each term, then that the user has seen summaries of search engines to meet their information needs over time (equally weighted between short and long term). $P = p_i = \sum_j b_{ij}$

Now, if we repeat the above process through multiple sessions and update the vector P aligned with the terms, it will create a single vector with the tastes over time and it can be assumed what is a vector around the convergent terms of affinity for the interests of the user

Algorithm 3 Algorithm for adaptive term suggestions

Derive a vector with affinities for terms in the long term, based on observed behavior
Derive a set of key terms based on the results of the consultation issued by the user.
Derive a set of key terms based on the model of recommendation
Using the profile moderate or boost key terms found in the steps two and three
Show terms

System Validation

The evaluation focused on the user perspective to measure the quality of the suggestions made by the system. Evaluations were conducted offline through surveys that did generate terms at random. We asked volunteers to rate on a scale qualitative degree of relationship that they associated with each terms in the context of the original query.

The studies involved on a voluntary basis 69 students (52 men and 17 women) with an average age of 24.3 years in various undergraduate and postgraduate programs in the Faculty of Engineering. In the graphs depicted in 4 is summarized characteristics of the population in terms of the ability to search the Web and their knowledge of the subject of the proposed queries.

Local Analysis Offline Evaluation

To evaluate the system, we downloaded and preprocessed up to 100 snippets using the Google API, and the vector-space

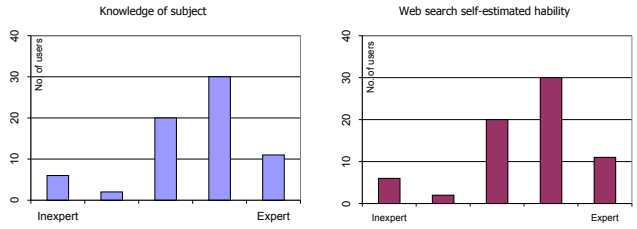

Figure 4: Some characteristics of participants

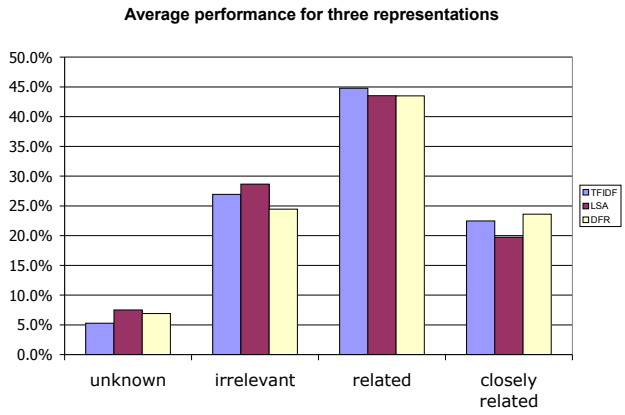

Figure 5: Terms quality as assessed by users with local analysis methods

model was calculated in Matlab with the help of Text Matrix Generator toolkit[2]. We set experiments varying number of documents (snippets) used for obtaining the terms in the analysis methods based on locally. They were calculated considering up to 10 documents, 20 documents, 50 documents and 100 documents. In total, it were assessed a group of 64 queries of broad themes and it were constructed a survey system which generated the terms and organized randomly, without replacement, so as to reduce the bias. It produced 10 terms and we asked to people to estimate, on a qualitative scale, the relationship level of every term regarding their knowledge of query subject in question. In the figure 5, it is shown the average results of all methods.

We also has evaluated the average runtime of the experiments in Matlab after 100 executions. The scripts for term generation were run in a Windows Turion of 1.8 GHz with 2 GB of RAM. The results are shown in figure 6, and they were obtained with the profiler built-in in Matlab.

Recommender System Offline Evaluation

For the evaluation process of query terms suggested by the methods of recommendation, we have implemented all methods in python scripts and we conducted surveys on the same group of participants that the local analysis method, but only in 34 queries from the original set. Figure 7 presents results on the same type of evaluation than before.

[2]http://scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/

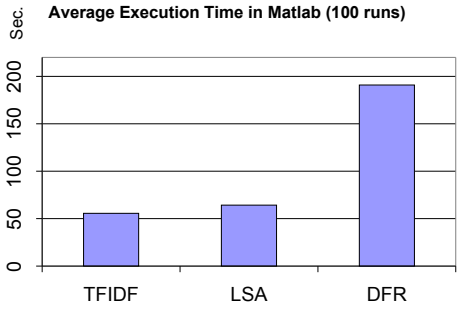

Figure 6: Average execution time for 64 queries (100 runs) with local analysis

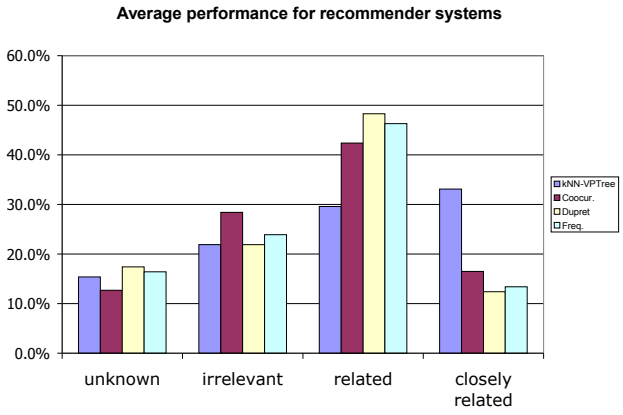

Figure 7: Terms quality as assessed by users with recommender systems

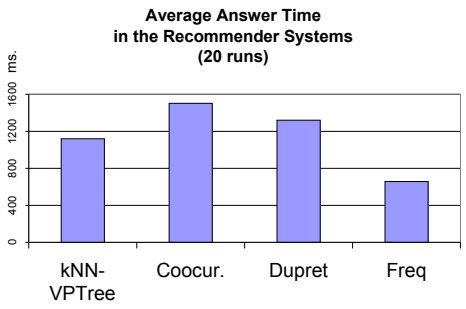

Figure 8: Average answer time for 34 queries (20 runs) with recommender systems

	baseline	proposed system
perceived time	5 and 10 min.	5 and 10 min.
effective time	5 and 10 min.	5 and 10 min.
query terms	3.2	3.1
reformulations	2.5	2.3
average ranking	2.5	2.8
avg. suggested terms used	-	1.4

Table 1: Assessments for concrete search tasks

In order to take a measure of the execution time of the recommendation systems, they were implemented in a python web server and their response times were calculated for the queries in the study. The results are shown in figure 8.

Interactive System User Study

To measure the effectiveness of the complete system, it was asked a group of 20 volunteers participate in a study where they searched for information, with exploratory tasks and with tasks with specific answers. They used a toolbar that implemented the client component of the architecture[3], using DFR as local analysis model rewriting the Google search results page with the recommendations and disabling the Google personalization at the client side. A control group only used the search engine Google as is (also with no personalization of Google). In both cases the URLs visited were tracked for further analysis. Participants included 15 men and 5 women with an average age of 23.5, a mastery of the subject average of 1.25 out of 5 and an average of search experience self-estimated at 3.3 (on the same scale that was used for the evaluation of local analysis) out of 5. The results obtained by the baseline and by the proposed system are shown in Tables 1, 2 and 3.

Also, to measure the personalization system quality, a group of 4 participants, who voluntarily agreed to leave their personal information available, were provided with an personalized exploratory task. This was chosen at random from a group of tasks designed based on observed behavior in the log files of proxy servers at campus. So, the pre-calculated profiles were preloaded in the browser component to the process of personalization. The results are shown in the table 4.

[3]developed as add-on of the browser mozilla firefox

	baseline	proposed system
perceived time	5 and 10 min.	5 and 10 min.
effective time	5 and 10 min.	5 and 10 min.
query terms	3.4	4.1
reformulations	2.7	2.4
average ranking	3.7	2.1
avg. suggested terms used	-	1.7

Table 2: Assessments for exploratory search tasks

easy use	4.6
perceived utility	2.7
enjoyment	3.8

Table 3: Overall user assessments of the proposed system interface

Results

According to the assessment of users, is a reasonable performance for the TFIDF model, while for LSA model quality was not as good, and in fact worsened this regard to the other. The model of divergence on the other hand, was the one who obtained the best results for the proportion of terms related documents while providing the lowest proportion of irrelevant terms on average.

We believe that the performance was acceptable compared against a system that randomly select terms, and that execution times for processing the four data sets 4, 10, 20, 50 and 100 snippets for 64 queries test shows that in all cases it is feasible to perform his calculations from the client. It is curious to note that the runtime thought that would be greater, LSA, is in fact the lowest of all. This may be due to Matlab provides libraries for highly efficient operations and matrix calculation of SVD, while the other deployments are not optimized in the same development environment.

About models of recommendation of terms, is that the best performance is given by the recommendation system based on measures of similarity with a binary representation of the terms on the search domains. The use of a structure as VPTree as a means to make metric indexing obviously accelerates response times. It also to be noted that a much simpler model based only in frequency of query terms, offers a reasonable performance and response time much smaller, but more conducive to recommend moderately related terms. All these models are sensitive to an accurate method of detecting sessions, therefore it is possible that the performance of the co-occurrence and Dupret models has not been as good as could be expected.

	baseline	proposed system
perceived time	5 and 10 min.	5 and 10 min.
effective time	5 and 10 min.	5 and 10 min.
query terms	2.7	3.3
reformulations	4.2	3.1
average ranking	2.8	2.6
avg. suggested terms used	-	2

Table 4: Assessments for personalized tasks

As for evaluations of the prototype in real user tasks. The system presents a minimal interruption in the normal flow of users, and receives an overall rating very positive in terms of ease of use and enjoyment. The listing of query terms suggested, as an addendum to the results page of the search engine Google, was received positively in general.

In all test scenarios, tasks were complex enough as they take between 5 and 10 minutes in both systems. The number of query terms used in all cases, are slightly greater than average, and this may indicate a level of recognition that more terms in Web search, the better the quality of the results. However, studies on larger populations has shown that this is not the case and that the most frequent users prefer to use queries with fewer words. In all cases, it was also noted that there were reformulations on the original query, and although the number of terms used at the suggestion of our system was not greater than 2, this indicates that there is scope to satisfy those needs better information.

Conclusions and Future Work

In this paper, we have presented our work on the development of an adaptive Web search system that allows the users to interactively refine their queries. According to user evaluation, we have found a reasonable model to provide adaptive expansion terms to reformulate user's information needs in case that the first results were no satisfactory, with a vector model built over the search engine results for a given query. Of the three methods evaluated, we have found the divergence from randomness and the term frequency-inverse document being almost equivalent, with the former being the slowest in our tests.

Also we did evaluate the recommender system and have found a similar performance in finding related terms based on the "wisdom of crowds", with semantic links granted by the aggregated behavior of a community of searchers.

In our evaluation of the whole system, participants showed a trend to use an amount of terms, superior to expected. Therefore we believe that our system can be most useful in situations in which users do not possess that level of sophistication. In all cases, the system was well received in terms of usability.

An important characteristic of the proposed system is that its performance improves with its continuous use, i.e., the more information in the system, the better the recommendations results.

In general, our architecture was able to find semantically related terms as effective support for the process of searching for users with customized interests.

Acknowledgments.

We are very grateful with all the participants of the studies presented here. Also, we thanks the very valuable help of Jochen Schulz in the implementation of the metric space for nearest neighbors search. Also, we thanks to the anonymous reviewers for helpful comments.

References

Amati, G., and Rijsbergen, C. J. V. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.* 20(4):357–389.

Baeza-Yates, R.; Hurtado, C.; and Mendoza, M. 2004. Query recommendation using query logs in search engines. In *Current trends in database technology - EDBT 2004 workshop*.

Billerbeck, B.; Scholer, F.; Williams, H. E.; and Zobel, J. 2003. Query expansion using associated queries. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, 2–9. New York, NY, USA: ACM Press.

Carmel, D.; Farchi, E.; Petruschka, Y.; and Soffer, A. 2002. Automatic query refinement using lexical affinities with maximal information gain. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 283–290. New York, NY, USA: ACM Press.

Carpineto, C., and Romano, G. 2004. Exploiting the potential of concept lattices for information retrieval with credo. *J. UCS* 10(8):985–1013.

Deerwester, S.; Dumais, S. T.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41:391–407.

Ducrou, J., and Eklund, P. 2007. Searchsleuth: The conceptual neighbourhood of an web query. In *Proc. of Fifth International Conference on Concept Lattices and Their Applications CLA 2007*.

Dupret, G., and Mendoza, M. 2006. Automatic query recommendation using click-through data. In *Symposium on Profesional Practice in Artificial Intelligence, 19th IFIP World Computer Congress, WCC 2006,*.

Efthimiadis, E. N. 1995. User choices: A new yardstick for the evaluation of ranking algorithms for interactive query expansion. *Information Processing & Management* 31:605–620.

Kim, M.-C., and Choi, K.-S. 1999. A comparison of collocation-based similarity measures in query expansion. *Inf. Proc. and Mgmt.* 35(1):19–30.

Kraft, R.; Chang, C. C.; Maghoul, F.; and Kumar, R. 2006. Searching with context. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 477–486. New York, NY, USA: ACM Press.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2007. *An Introduction to Information Retrieval*. Cambridge U.P.

Matsuo, Y., and Ishizuka, M. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*.

Qiu, Y., and Frei., H. 1993. Concept based query expansion. In *Proc. of the 16th Intl. ACM SIGIR Conf. on Research and Development in Inf. Retrs.*

Ruthven, I.; Lalmas, M.; and van Rijsbergen, C. J. 2003. Incorporating user search behavior into relevance feedback. In *JASIST 54(6):529-549.*

Salamanca, A.; González, F.; and León, E. 2007. Global analysis of web behavior at the national university of colombia. In *Proceedings of XXXIII Latinoamerican Conference in Informatics*.

Salton, G., and Buckley, C. 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41:288–297.

Salton, G., ed. 1979. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall. chapter Relevance feedback in information retrieval, 313–323.

Teevan, J.; Dumais, S. T.; and Horvitz, E. 2005. Beyond the commons: Investigating the value of personalizing websearch. In *Proceedings of Workshop on New Technologies for Personalized Information Access (PIA '05)*.

Teevan, J. 2007. *Supporting Finding and Re-Finding Through Personalization*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Yianilos, P. 1993. Data structures and algorithms for nearest neighbor search in metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*.