Summarizing Local Context to Personalize Global Web Search

Paul - Alexandru Chirita
L3S / University of Hannover
Deutscher Pavillon, Expo Plaza 1
30539 Hannover, Germany
chirita@l3s.de

Claudiu S. Firan
L3S / University of Hannover
Deutscher Pavillon, Expo Plaza 1
30539 Hannover, Germany
firan@l3s.de

Wolfgang Nejdl
L3S / University of Hannover
Deutscher Pavillon, Expo Plaza 1
30539 Hannover, Germany
nejdl@l3s.de

ABSTRACT

The PC Desktop is a very rich repository of personal information, efficiently capturing user's interests. In this paper we propose a new approach towards an automatic personalization of web search in which the user specific information is extracted from such local desktops, thus allowing for an increased quality of user profiling, while sharing less private information with the search engine. More specifically, we investigate the opportunities to select personalized query expansion terms for web search using three different desktop oriented approaches: summarizing the entire desktop data, summarizing only the desktop documents relevant to each user query, and applying natural language processing techniques to extract dispersive lexical compounds from relevant desktop resources. Our experiments with the Google API showed at least the latter two techniques to produce a very strong improvement over current web search.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguisting processing*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

Algorithms, Experimentation

Keywords

Personalized Web Search, Desktop Summarization, Relevance Feedback, User Profile

1. INTRODUCTION

Keyword queries are inherently ambiguous. Take for example the query "canon book", which covers several different areas of interest: religion, digital photography, and music. Clearly, since the URLs most likely to be visited are those returned at the very top of the result list [16], search engine output should be filtered to better align the results with user's interests. Thus, the photographer should receive pages about digital cameras, the clergyman should get religious books, and the performing artist should obtain documents on music theory. In fact, a recent study presented by SearchEngineWatch [35] indicates that more than 80% of the users would prefer to receive such personalized search results.

One of the early Information Retrieval techniques used to enhance search quality is Relevance Feedback [31]. Even though it does not deal with user specific personalization per se, it has been shown to be quite effective in improving retrieval performance. It is based on collecting relevance information from a set of carefully selected documents, which is then used to modify the search query and perform an additional retrieval step. However, when selecting these relevant documents *automatically* (e.g., by considering all Top-K search engine output documents as relevant), several terms unrelated to the user query might still be added as expansion keywords if they are present in these automatically selected documents and have the suitable distribution in the document collection [25]. In this paper we address this problem by choosing query expansion terms from a different data source, the personal desktop, in which all documents are at least to a certain extent related to user's interests. By "personal desktop" or "PC desktop" we denote the set of personal documents residing on each user's personal computer. Thus, the personalization dimension is also automatically included in the search algorithm.

Several advantages arise when moving web search personalization down to the desktop level. First comes of course the quality of personalization: The local desktop is a very rich repository of information, accurately describing most, if not all interests of the user. More, as all the "profile" information is stored and exploited locally, on the personal desktop, another very important benefit can be drawn: Privacy. Search engines should not be able to know about a person's interests, i.e., they should not be able to connect some person with the queries she issued, or worse, with the output URLs she clicked within the search interface[1] (see Volokh [38] for a discussion on the privacy issues related to personalized web search). Almost all previous algorithms for personalizing web search need such input information though.

In this paper we propose a novel approach to selecting query ex-

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.

[1]Generally, search engines can map queries at least to IP addresses, for example by using cookies and mining the query logs. However, moving the user profile entirely down to the desktop level would at least ensure such information is not explicitly associated to a user ID and stored on the search engine side. More, if desired, the desktop level personalized search application could be developed in such a way as to conceal the user identity and search history from the search engine.

pansion terms for web search by adapting summarization and natural language processing techniques to extract these supplementary keywords from locally stored desktop documents. After having discussed related work in Section 2, we investigate three broad methods towards achieving this goal. First, in Section 3.1 we propose to summarize the entire desktop using term clustering techniques and then to choose suitable expansion terms from these clusters, exploring both context independent and query biased approaches. Afterwards, we move our focus towards single document summarization techniques. In Section 3.2 we propose to issue the original web user query on the desktop and to extract expansion keywords from the most significant sentences within the Top-K (desktop) hits. Similarly, in Section 3.3 we investigate the possibilities to select query expansion keywords from the most dispersive lexical compounds within the Top-K (desktop) hits returned to user's initial web query. Our experiments performed with the Google API[2] (Section 4) show an improvement in Mean Average Precision (MAP) [2] of up to 81.57% for single word queries and up to 21.11% for multi-word queries when comparing to regular Google web search.

2. PREVIOUS WORK

Within this work we find ourselves at the confluence of personalized search and summarization algorithms. There are only very few previous publications combining these areas and even fewer address both the PC Desktop and the World Wide Web environments. From the existing articles, a very important one is the work of Lam and Jones [19]. They improve pseudo relevance feedback by selecting query expansion terms only from the most significant sentences (as computed using query based summarization techniques) of the Top-K search engine output. We used their technique as a starting point for one of our algorithms, but we moved the set of relevant documents from the web down to the desktop, and we adapted the methods for selecting meaningful sentences appropriately.

The following two sections will now discuss some of the most important works in either one of our two main research areas, personalizing search and summarization.

2.1 Personalized Search

We distinguish two broad approaches to personalizing web search, based on the way user profiles are exploited to achieve personalization: (1) integrating the personalization aspect directly into PageRank [27], and (2) using the personalized search algorithm as an additional search engine measure of importance (together with PageRank, TFxIDF, etc.). Let us now inspect each of them in detail.

PageRank-based Methods. The most efficient personalized search algorithm will probably be the one having the personalization aspect already included in the initial rankings. Unfortunately, this seems very difficult to accomplish. Initial steps in this direction have been already described by Page and Brin [27], who proposed a slight modification of the PageRank algorithm to redirect the random surfer[3] towards some preferred pages. However, it is clearly impossible to compute one PageRank vector for each user profile, i.e., for each set of pages "privileged" by the random surfer.

Haveliwala [14] proposed to alleviate this problem by building a topic-oriented PageRank, in which a set of 16 PageRank vectors biased on each of the 16 main topics of the Open Directory is initially computed off-line, and then these vectors are combined at run-time based on the similarity between the user query and each of the 16 topics. This approach is clearly feasible, but also limited to the 16 pre-defined topics.

Finally, Jeh and Widom [15] identified the possibility to express each Personalized PageRank Vector (PPV) as a linear combination of a set of "special" vectors, called basis vectors. At query time, an approximation of the PPV is constructed from the precomputed basis vectors using dynamic programming. Nevertheless, the input data (a set of preferred URLs) can only be selected from within a small pre-defined group of pages[4] (common to all users) and the computation time is relatively high for large scale graphs.

Hybrid Ranking Methods. As search engines rely on various indicators when ordering query output (i.e., textual content, link structure, etc.), current research has focused more on a different approach: Building an independent and simpler personalized ranking algorithm, whose output is combined with that of PageRank, TFxIDF, and other metrics.

Sugiyama et al. [34] analyze user's surfing behavior and generate user profiles as features (terms) of the pages they visited. Then, upon issuing a new query, the results are ranked based on the similarity between each URL and the user profile. In a similar work, Gauch et al. [12] build profiles by exploiting the same surfing information (i.e., page content and length, time spent on each URL, etc.), as well as by spidering the URLs saved in the personal web cache and classifying them into topics of interest. Both solutions are orthogonal to ours, as they only inspect previous web browsing behavior, whereas we explore a much richer information repository, the entire PC Desktop.

Liu et al. [22] restrict searches to a set of categories defined in the Open Directory. Their main contribution consists in investigating various techniques to exploit users' browsing behavior for learning profiles as bags of words associated to each topical category. Similarly, Chirita et al. [6] proposed to use already existing large scale taxonomies to personalize search by reordering search results based on several graph distances between the topic associated to each output URL and the topics defined in the user profile. Even though both approaches showed good results when compared to non-personalized web search, their performance is very much dependent on the URLs classified within such taxonomies.

More recent, the work of Teevan et al. [36] is the only one also exploiting desktop data for web search. They modified the query term weights from the BM25 weighting scheme [17] to incorporate user interests as captured by her desktop index. However, they selected the query expansion terms from the Top-K documents returned by the web search engine, whereas we seek for these terms within the more relevant, personal PC desktop data, thus avoiding choosing unrelated expansion keywords which could be present in the Top-K web search output [25].

2.2 Summarization

Automated summarization usually deals with concatenating text-span excerpts (i.e., sentences, paragraphs, etc.) into a human understandable document summary and it dates back to the 1950's [23]. More, with the advent of the world wide web and large scale search engines, an increased attention has been focused towards this research area and quite several new approaches have been proposed. For example, the diversity of concepts covered by a document has been first explored by Carbonell and Goldstein [4] in 1998. They proposed to use Maximal Marginal Relevance (MMR), which selects summary sentences that are both relevant to the user query

[2]http://api.google.com

[3]PageRank can be viewed as the stationary distribution of an infinite random walk over the web graph, following an outgoing link from the current page with probability $(1 - c)$ (usually $c = 0.15$) and getting bored and selecting a random page with probability c.

[4]Some work has been done in the direction of improving the quality of this set of pages [7], but users are still restricted to select their preference set from a rather restricted corpus.

and least similar to the previously chosen ones. Later, Nomoto and Matsumoto [26] developed this into a generic single-document summarizer that first identifies the topics within the input text, and then outputs the most important sentence of each topic area.

Another quite different new approach was to generate the summary as the set of top ranked sentences from the original document according to their salience or likelihood of being part of a summary [13, 11]. Consequently, more search specific applications of summarization have been proposed. Zeng et al. [41] for example used extraction and ranking of salient phrases when clustering web search results. Others have used hierarchies to improve user access to search output by summarizing and categorizing the retrieved documents [20], or to organize the topic words extracted from textual documents [21, 33]. Finally, this work is in fact an application of summarization techniques into a new research area, that of personalizing web search.

Along with the fast growth of information amounts, a recent need for multi-document summarization has also been exerted and thus a few algorithms have been proposed. A very popular one is MEAD [28], which first uses an implementation of the "pile metaphor" [32] to create groups of similar documents, and then selects the most representative sentences from these clusters according to several salience measures. We used it as basis for one of our desktop summarization approaches.

There exists also a large amount of document clustering research and we believe this could be exploited to develop better means of summarizing personal information repositories (see Willett [39] for a relatively old, but very comprehensive review of the fundamental aspects related to clustering). However, it is outside the scope of this paper to review these techniques here, and thus we only mention some of the relevant ones for our scenario, namely Scatter/Gather [9, 8] for clustering based on the term vectors describing each document, Grouper [40] for clustering based on phrases rather than terms, and the work of Zeng et al. [41] for combining multiple evidences in document clustering.

Finally, this work is to some extent connected to the Just-In-Time Information Retrieval paradigm, in which each user's currently active desktop document is automatically analyzed to extract its keywords (i.e., using algorithms similar to the ones we present in this paper), and then to recommend other related documents which could be useful in performing the on-going activity. Several approaches exist, either aimed at finding such relevant documents among the personal (desktop) repository [29, 10], or within the World Wide Web [3, 5].

3. SUMMARIZING THE DESKTOP DATA

Summarizing desktop data is itself a challenge, since most of the current summarization research has not tackled such complex data sets. For example, it is quite common nowadays to have about 100,000 indexable desktop items (i.e., containing some amount of textual information), these documents being either HTML pages, Word documents, small textual notes and chat conversations, or even smaller metadata for mp3 files, etc. We therefore investigated several summarization paths towards choosing the right web query expansion keywords: (1) a multi-document summarizer which outputs centroids as bags of words with weights associated to them, (2) a single-document summarizer which ranks sentences according to their representativeness for the user query and for the document itself, and (3) a lexical compounds generator for the top ranked documents returned when issuing the user query on the desktop. We think these three paths cover most of the important approaches to selecting both query specific and query independent expansion terms from the desktop document collection.

Following the work of Lam et al. [19], we chose to index only documents with at least 7 indexable terms (i.e., not stopwords). Moreover, we defined several heuristics to exclude from the index some very common automatically generated file categories such as Java documentation, as their large granularity tended to negatively influence the desktop summaries. Finally, when choosing the terms to expand user's original query (e.g., after the centroids have been output by the multi-document summarizer), we decided to only use TF, rather than TFxIDF, as one very frequent local term (e.g., PageRank) might in fact be rather rare in the web[5]. A large stopword list was used to initially remove any possible misleading terms. Also, summarization was achieved employing a logged version of TF in order to avoid having some too frequent terms mislead the results. The variants of TF and IDF we used were as follows:

$$TF_{t_k,D_j} = \begin{cases} 0 & \text{, if } TF'_{t_k,D_j} = 0 \\ 1 + log(TF'_{t_k,D_j}) & \text{, otherwise} \end{cases}$$
$$IDF_{t_k} = log(1 + \frac{N}{DF_{t_k}})$$

where TF'_{t_k,D_j} is the actual frequency of term t_k in document D_j, N is the total number of documents in the collection and DF_{t_k} is the document frequency for term t_k.

Let us now present the details of each of our above mentioned three approaches to extract suitable web search query expansion terms from personal information repositories.

3.1 Centroid Based Summarization

The centroid based summarization was first proposed by Radev et al. in [28], who applied the "pile metaphor" document clustering approach of Rose et al. [32] for summarization purposes. Its main underlying principle is to represent all documents in a collection as in traditional IR, using term vectors, and then to group these vectors into representative clusters. Thus, a scan is performed over all documents within the collection; for every document, if its similarity with at least one cluster centroid is above a certain threshold, it will be associated to its most similar cluster. Otherwise, a new cluster is created having the current document term vector as centroid.

As the algorithm of Radev et al. was intended for much smaller data sets such as news articles, we had to incorporate in it several desktop specific aspects. First, we ordered terms within cluster centroids only by their TF values, rather than TFxIDF. The document similarity formula was the sole place where both TF and IDF had been considered. This is reasonable, since two documents both containing many infrequent words are most probably related within the desktop environment as well. Second, due to the large amount of data residing in personal information repositories, we had to limit the centroid cluster size to its top $\delta = 500$ terms, as otherwise the computation time would have grown too much. Third, we attempted to cluster either all desktop indexable documents, or only the documents manually accessed within the last three months a system call hook was employed here to log every user resource open / create access for this period.. As the former approach covered much more documents, the optimal similarity threshold was $\tau = 0.01$, whereas for the latter one we found $\tau = 0.1$ to perform best. Other 14 possible values were investigated, ranging from 0.001 to 0.1. Finally, we defined "PC Desktop" as the collection of all emails, web cache documents, and indexable files of a user. For the latter ones, we did not index the entire hard disks, but only the list of paths containing personal documents, as spec-

[5]Note that frequent local terms have a low TFxIDF score on the desktop. However, they might have a high TFxIDF score on the Web, thus being very discriminative when expanding user's query.

ified by each person[6]. The complete form of the algorithm is also depicted in Algorithm 3.1.

Algorithm 3.1. Centroid Based Desktop Summarization.

Similarity (Document D_i, Centroid C_j):

1: **Return** $\frac{\sum_{Term\ t_k \in D_i} TF_{t_k,D_i} \cdot CF_{t_k,C_j} \cdot IDF_{t_k}}{\sqrt{\sum_{Term\ t_k \in D_i} TF^2_{t_k,D_i}} \cdot \sqrt{\sum_{Term\ t_k \in C_j} CF^2_{t_k,C_j}}}$

where CF_{t_k,C_j} is the weight of term t_k within centroid C_j.

1: **For** each new document D_i
2: MaxSim = $\mathbf{Max}_{\forall j}\ Similarity(D_i, C_j)$
3: MaxCen = $\{j \mid Similarity(D_i, C_j) == MaxSim\}$
4: **If** ($MaxSim \leq \tau$ **or** $\nexists$ Centroids) **then**
5: **Create** new centroid with the top δ D_i terms with respect to TF_{t_k,D_i}.
6: **Else**
7: **Let** d be the number of documents covered by C_{MaxCen}.
8: **For** each $t_k \in D_i \cup C_{MaxCen}$
9: $CF_{t_k,C_{MaxCen}} = CF_{t_k,C_{MaxCen}} * d/(d+1) + TF_{t_k,D_i}/(d+1)$
10: **Reduce** C_{MaxCen} to its top δ terms with respect to $CF_{t_k,C_{MaxCen}}$.

Manual inspection showed these clusters to be quite representative for the data they represented. Yet how can they be used for query expansion? We have investigated several options:

1. Select the Top-C biggest clusters (with respect to the number of documents contained therein) and from each cluster choose the term with the highest BM25 value [17]. The BM25 probabilistic weighting scheme practically incorporates relevance feedback into ranking by modifying query term weights to bias search output on the documents the user selected as relevant. As we took the automatically generated desktop cluster centroids as implicit relevance judgments, we used the modified version of BM25 proposed by Teevan et al. [36] which also covers relevant documents from outside the document space (i.e., the web), as follows:

$$W_i = log\frac{(r_i + 0.5)(N - n_i + 0.5)}{(n_i + 0.5)(R - r_i + 0.5)}$$

where N represents the amount of documents in the search corpus (web), n_i is the number of documents in the corpus that contain term i, R is the amount of documents for which relevance feedback has been provided (desktop), and r_i is the number of these latter documents which also contain the query term. We approximated N with the number of web documents containing the term "the".

2. Use W_i to choose C expansion terms proportionally to the dimension of the desktop clusters. For example, if there are say three clusters of 15, 5, and 5 documents respectively, and $C = 5$, then choose three terms from the first cluster and one term from each of the subsequent two ones.
3. Select the clusters that contain all keywords from the original query. Then, use W_i to choose C expansion terms proportionally to the dimension of these chosen clusters.

In all cases we experimented with $C = 5$ and $C = 7$, but due to the space limitations we will only report the former minimally better parameter setting. Also, as in all three cases the selected query expansion terms may not always be related to the actual user query (since they represent a part of the entire desktop), significantly more importance (i.e., weight) should be given to user's initial query keywords. As Google API does not allow specification of term weights, we experimented with this feature by using the above mentioned BM25 model to re-rank the Top-50 URLs output to the user query. This way, the entire document candidate set was surely related to the original query.

[6]Although this definition was targeted at single-user PCs, one could easily extend it to multiple-user ones.

3.2 Sentence Selection

There exist quite several approaches to sentence based summarization. However, we chose to start from [19], as it had a similar end goal with us, i.e., to select terms for query expansion. Thus, for each user query, we first issue it on the PC desktop and retrieve the Top-30 documents using the Lucene[7] scoring function. Then, from each of these documents we extracted the most salient sentences with respect to the user query by evaluating the following formula:

$$SentenceScore = \frac{SW^2}{TW} + \frac{TQ^2}{NQ}$$

The first term is based on Luhn's cluster measure [23] and is the ratio between the square amount of significant words within the sentence and the total number of words therein. A word is significant in a document if its real frequency (i.e., not logged) is above a threshold as follows:

$$TF > ms = \begin{cases} 7 - 0.1 * [25 - NS] & , if\ NS < 25 \\ 7 & , if\ NS \in [25, 40] \\ 7 + 0.1 * [NS - 40] & , if\ NS > 40 \end{cases}$$

with NS being the total number of sentences in the document.

The second term comes from the work of Tombros and Sanderson [37] and is computed using the ratio between the square number of query terms present in the sentence and the total number of terms from the query. It is based on the belief that the more query terms contained in a sentence, the more likely will that sentence convey information highly related to the query.

Lam et al. [19] also investigated the use of several other sentence salience metrics, such as the document title and the location of each rated sentence within a document. We argue that such metrics are not suitable for PC desktop resources, first because many of them have no title, and second because unlike for news articles, where the first sentences are usually quite representative for the entire document, here there is no clear correlation between the location of sentences and their importance for a document.

Once these sentence scores were computed, we sought for (five) query expansion terms using two approaches: (1) using W_i (see BM25 in the previous section) over the top 9 sentences, as reported in [19], and (2) using W_i over the top 2% sentences. The new latter approach is motivated by previous findings that longer documents tend to contain more content words [18] and does indeed slightly improve over the former one (see Section 4 for more details).

As these query expansion terms had been selected from documents *relevant* to the user query, we experimented here with two different techniques: (1) simply expanding the query with the selected keywords having the same weight as the original ones (since the Google API did not allow us to specify different weights), and (2) re-ranking Google's Top-50 output URLs according to the modified BM25 scheme, as with the previous algorithm. The amount of five query expansion keywords has been selected based on two

[7]http://lucene.apache.org

premises: First, previous Okapi TREC submissions [30] also employed only a small number of expansion terms; second, at experimentation time, the Google API allowed at most ten terms per query, and thus using more than five expansion keywords would have incorrectly limited too much the maximum query length available to our testers. Even so, both query expansion approaches showed a highly significant improvement over the original Google results.

3.3 Lexical Compounds

The final algorithm is based on the natural language processing solutions proposed by Anick and Tipirneni [1]. They defined the *lexical dispersion hypothesis*, according to which an expression's lexical dispersion (the number of different compounds it appears in within a document set) can be used to automatically identify key concepts of that document set. As the local desktop resources are implicitly relevant for user's interests, we thus sought for such concepts within desktop files that are also relevant for the user query.

As with our previous algorithm, we start by issuing the user query down on the desktop search engine and selecting the Top-30 results. Then, we inspect these output documents for all their lexical compounds of the following form, as defined in [1]:

$$\{\ ?adjective\ noun+\ \}$$

Although we performed this step at run-time (using WordNet [24]), it could be easily run off-line, at indexing time, as the compound generation process is not influenced by the user query (i.e., all compounds from the selected documents are generated). The only query dependent aspect is the selection of the documents whose compounds are included in the dispersion calculation. Thus, once these lexical constructions have been identified, they are sorted depending on their dispersion within these Top-30 desktop documents and the terms of the most frequent three compounds are used as query expansion keywords.

The output of the lexical algorithm so far showed itself to be indeed very significant for the original query, and thus we chose to experiment both the regular query expansion approach and that of re-ranking Google's Top-50 original output URLs (see the end of the previous section for more details on each of these solutions).

4. EXPERIMENTS

4.1 Experimental Setup

We started our analysis by manually inspecting the output of each desktop summarization algorithm. In all cases, we found it to be quite representative for the original document or set of documents. However, as in other similar works (e.g., [19]), our main objective measure of summary quality was its overall effect on web search performance, and thus we will focus our presentation only towards this aspect.

To evaluate the precision of our personalization algorithms we interviewed 15 subjects (either architects or researchers in different computer science areas and education). In the first phase of the evaluation they installed our activity logging tool and used it continuously for about three months. Then, they installed our desktop indexer and chose six queries *related* to their everyday activities in a similar manner to the work of Chirita et al. [6], as follows:

- One single-word *specific* query, which they *knew to have one or maximum two meanings*[8].
- One single-word *relatively ambiguous* query, which they knew to have two or three meanings.
- One single-word *ambiguous* query, which they knew to have at least three meanings, preferably more.
- Three queries of the same types as above, but with *multiple keywords* (i.e., at least two, preferably more).

For each query, the Top-10 results generated by 15 versions of the algorithms we presented in Section 3 were shuffled into one set containing usually between 50 and 80 URLs. Thus, each subject had to assess about 400 URLs for all six queries, being neither aware of the algorithm, nor of the ranking of each assessed URL. Overall, 90 queries were issued and about 6,000 URLs were evaluated during the experiment. For each of these URLs, the testers had to give a rating ranging from 0 to 2, thus dividing the relevant results in two categories, (1) relevant and (2) highly relevant. Also, the output quality was evaluated in terms of Mean Average Precision (MAP) over the first 10 results, precision at the first 5 positions of the resulted ranking (P@5), as well as precision at the top 10 output rankings (P@10). Finally, all our results were tested for statistical significance using T-tests (i.e., we tested whether the improvement over the Google API output[9] is statistically significant).

In all the forthcoming tables, we will label the algorithms we evaluated as follows:

- **Google**: The actual Google query output, as returned by the Google API.
- **CATF**: Re-ranking Google's Top-50 URL's by selecting query expansion terms from the Top-5 biggest desktop clusters, as computed over the entire desktop.
- **CAP**: Same as above, but selecting the additional keywords proportional to the dimension of the clusters (see Section 3.1 for more details).
- **CAQ**: Same as previous, but considering only the clusters that contain all original query keywords. If no cluster satisfies this condition, then the clusters containing maximal subsets of the query are regarded.
- **CRTF, CRP, CRQ**: Same as the previous three approaches respectively, but building clusters only over the documents accessed by the user at least once during the past 3 months.
- **SentQEP**: Expanding the user query using the Sentence Selection method over the most significant 2% sentences of each relevant desktop document.
- **SentQEF**: Same as above, but applied over the most significant 9 sentences of each relevant desktop document.
- **SentRRP**: Re-ranking the Top-50 URLs returned by the Google API using a query expanded with the Sentence Selection method applied over the most significant 2% sentences of each relevant desktop document.
- **SentRRF**: Same as above, but applied over the most significant 9 sentences of each relevant desktop document.
- **LexQE**: Expanding the user query using the Lexical Compounds method over the 30 most relevant desktop documents with respect to the original search terms.
- **LexRR**: Re-ranking Google's Top-50 URLs using the query expansion terms extracted analyzing Lexical Compounds.

4.2 Results

Ambiguous queries. Our results for the various scenarios using ambiguous queries are presented in Tables 1, 2, 3, and 4. All our approaches performed very well on single-word ambiguous queries,

[8]Of course, that did not necessarily mean that the query had no other meaning.

[9]Whenever necessary, we also tested for significance the difference between pairs of the algorithms we proposed.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.280	0.226	0.279	-
CATF	0.360	0.313	0.356	*Minimal*, p = 0.0959
CAQ	0.346	0.320	0.346	*Minimal*, p = 0.0791
CAP	0.360	0.333	0.345	*Minimal*, p = 0.0877
CRTF	0.444	0.366	0.411	*Yes*, p = 0.0234
CRQ	0.422	0.388	0.426	*Highly*, p = 0.0077
CRP	0.422	0.366	0.409	*Highly*, p = 0.0093
LexQE	0.600	0.526	0.605	*Highly*, p = 0.0007
LexRR	0.400	0.353	0.381	*Yes*, p = 0.0248
SentQEP	0.573	0.573	0.591	*Highly*, p = 0.0032
SentRRP	0.413	0.353	0.378	*Yes*, p = 0.0136
SentQEF	**0.733**	**0.709**	**0.735**	***Highly*, p = 0.0014**
SentRRF	0.333	0.320	0.334	*Minimal*, p = 0.0885

Table 1: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for single word ambiguous queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.360	0.386	0.451	-
CATF	0.400	0.400	0.401	-
CAQ	0.440	0.380	0.445	-
CAP	0.386	0.393	0.409	-
CRTF	0.244	0.300	0.321	-
CRQ	0.333	0.333	0.354	-
CRP	0.355	0.344	0.368	-
LexQE	**0.600**	**0.526**	**0.596**	***Yes*, p = 0.0213**
LexRR	0.413	0.393	0.443	-
SentQEP	0.514	0.443	0.511	*Minimal*, p = 0.2474
SentRRP	0.440	0.420	0.467	*No*, p = 0.3732
SentQEF	0.546	0.477	0.544	*Minimal*, p = 0.1635
SentRRF	0.413	0.386	0.425	-

Table 2: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for multi-word ambiguous queries.

the improvement for this particular case reaching even 163.44% for SentQEF (Table 1). On the other hand, with multi-word queries the re-ranking methods performed all relatively poor. More, this phenomenon occurred relatively often within all multi-word scenarios, especially for the clustering methods. This is most probably because the query expansion terms were too specific for user's interests to help accurately re-order only the Google Top-50 results. However, inspecting more than the first 50 URLs is very time consuming without having a real search engine available. Finally, the fact that the high quality results were usually residing below the 50^{th} place is also probed by the very good results obtained with LexQE, SentQEF and SentQEP (always a significant improvement over Google), which simply expand the query and issue it again to the search engine.

Semi-ambiguous queries. The outcome was very similar for the semi-ambiguous queries (Tables 5, 6, 7, and 8) with the only difference that our improvements were slightly smaller, i.e., up to 44.73% for single-word queries with SentQEP (Table 7) and up to 35.67% for multi-word queries with LexQE (Table 6). This is correct, since Google has been shown to perform better as the query specificity increases [6]. We should also note that in all query type scenarios, restricting the analysis to only highly relevant results had a very small impact on the findings, minimally modifying them in both directions. For example, the improvement generated by LexQE for single-word semi-ambiguous queries was somewhat significant considering all relevance judgments and not significant considering only the highly relevant ones, but on the other hand the improvement of SentQEF was significant only when exclusively considering the highly relevant results.

Clear queries. As expected, it was more difficult to overcome

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.093	0.086	0.130	-
CATF	0.120	0.106	0.144	*No*, p = 0.3690
CAQ	0.133	0.106	0.138	*No*, p = 0.4132
CAP	0.146	0.126	0.121	*No*, p = 0.3696
CRTF	0.155	0.122	0.221	*Yes*, p = 0.0354
CRQ	0.133	0.111	0.164	*Minimal*, p = 0.2254
CRP	0.155	0.111	0.190	*Minimal*, p = 0.0928
LexQE	0.320	0.273	0.289	*Highly*, p = 0.0026
LexRR	0.133	0.120	0.127	-
SentQEP	0.280	0.266	0.269	*Yes*, p = 0.0219
SentRRP	0.106	0.100	0.100	-
SentQEF	**0.346**	**0.333**	**0.307**	***Highly*, p = 0.0040**
SentRRF	0.106	0.106	0.099	-

Table 3: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for single word ambiguous queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.186	0.180	0.235	-
CATF	0.146	0.160	0.162	-
CAQ	0.213	0.173	0.181	-
CAP	0.186	0.166	0.171	-
CRTF	0.066	0.111	0.151	-
CRQ	0.155	0.122	0.165	-
CRP	0.111	0.111	0.127	-
LexQE	0.306	0.266	**0.332**	*Highly*, p = 0.0031
LexRR	0.173	0.146	0.170	-
SentQEP	0.257	0.207	0.315	*Minimal*, p = 0.1029
SentRRP	0.200	0.180	0.201	-
SentQEF	**0.373**	**0.277**	0.295	*Minimal*, p = 0.1553
SentRRF	0.173	0.173	0.190	-

Table 4: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for multi-word ambiguous queries.

Google output for such queries, the best performances obtained being a 64.50% enhancement for single-word queries (LexQE, Table 9) and 9.30% for multi-word ones (SentQEP, Table 12), while the re-ranking approaches had only an average performance with small improvements for single-word queries and even worse results in the multi-word setting. The complete results are depicted in Tables 9, 10, 11, and 12.

Conclusions. Three broad conclusions could be drawn from our results. First, query expansion terms inferred from desktop data are quite specific, and thus useful only when re-ranking a large number of the top search engine results, in order to reach pages that are relevant, while also containing the expansion terms with a large frequency.

Second, when looking at smaller experiment sets (e.g., Tables 1-12), sometimes another phenomenon occurred: Small significance levels for quite high MAP differences, pointing out that although for most users our query expansion algorithms are very efficient, for another small number of subjects it performed relatively similar to Google. This was because their desktops contained almost no textual content (e.g., for junior students). Further research is needed to cope with such special cases, for example by indexing automatically generated metadata about local multimedia files, games, etc.

Finally, when these expansion terms were used for regular query expansion over the entire web, they did provide overall strong improvements compared to the initial search. For 13 of our subjects, including the junior students, the best algorithm overall was LexQE. The other two persons (both computer science researchers) ranked LexQE second, after SentQEP and Google respectively.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.520	0.426	0.500	-
CATF	0.520	0.500	0.540	*Minimal*, p = 0.1480
CAQ	0.506	0.480	0.515	*No*, p = 0.3537
CAP	0.533	0.480	0.520	*No*, p = 0.3348
CRTF	0.577	0.533	0.567	*Minimal*, p = 0.1065
CRQ	0.533	0.533	0.530	*No*, p = 0.2877
CRP	0.511	0.522	0.570	*Minimal*, p = 0.0821
LexQE	0.560	0.540	0.566	*Minimal*, p = 0.2413
LexRR	0.480	0.460	0.484	-
SentQEP	**0.600**	**0.551**	**0.614**	***Yes*, p = 0.0242**
SentRRP	0.493	0.500	0.515	*No*, p = 0.3531
SentQEF	0.533	0.471	0.545	*No*, p = 0.3714
SentRRF	0.520	0.460	0.472	-

Table 5: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for single word semi-ambiguous queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.506	0.426	0.485	-
CATF	0.440	0.400	0.440	-
CAQ	0.426	0.413	0.434	-
CAP	0.426	0.420	0.441	-
CRTF	0.466	0.400	0.505	*No*, p = 0.2573
CRQ	0.488	0.377	0.454	-
CRP	0.466	0.366	0.440	-
LexQE	**0.626**	**0.613**	**0.658**	***Yes*, p = 0.0227**
LexRR	0.480	0.420	0.454	-
SentQEP	0.471	0.446	0.494	*No*, p = 0.4576
SentRRP	0.440	0.400	0.427	-
SentQEF	0.560	0.533	0.526	*No*, p = 0.3481
SentRRF	0.466	0.440	0.437	-

Table 6: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for multi-word semi-ambiguous queries.

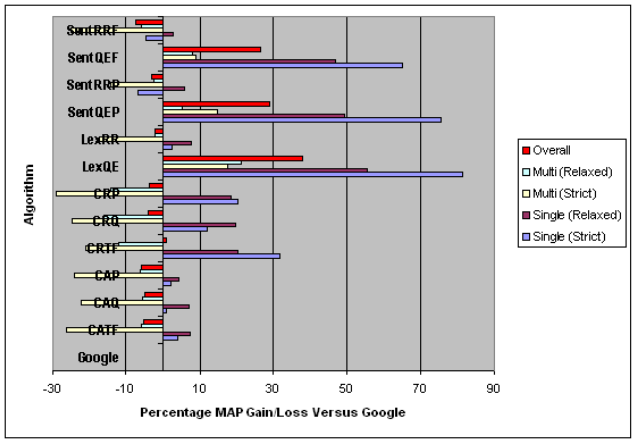

Figure 1: Relative MAP gain (in %) for each algorithm overall, as well as separated per query length category.

Also, over the entire experiment, LexQE improved over Google with 37.90% (significant with $p < 10^{-9}$), SentQEP improved with 29.00% ($p < 10^{-6}$), and SentQEF with 26.67% ($p \approx 10^{-4}$). Moreover, LexQE was also significantly better than SentQEP and SentQEF ($p = 0.04$), thus being our best approach by far. All results are depicted graphically in Figure 1.

Practical Issues. The response time is quite important for current search engines, and thus only those algorithms which can yield a quick valuable output are suitable for large scale topic independent applications. Therefore, even though the Sentence Selection

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.240	0.173	0.228	-
CATF	0.213	0.213	0.220	-
CAQ	0.200	0.186	0.201	-
CAP	0.226	0.213	0.221	-
CRTF	0.266	0.266	0.279	*Minimal*, p = 0.1626
CRQ	0.222	0.255	0.234	*No*, p = 0.4606
CRP	0.222	0.255	0.264	*Minimal*, p = 0.2374
LexQE	0.266	0.240	0.263	*No*, p = 0.3320
LexRR	0.213	0.206	0.219	-
SentQEP	**0.280**	**0.281**	**0.330**	***Minimal*, p = 0.0936**
SentRRP	0.200	0.206	0.214	-
SentQEF	0.266	0.235	0.256	*No*, p = 0.3847
SentRRF	0.213	0.206	0.210	-

Table 7: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for single word semi-ambiguous queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.213	0.193	0.217	-
CATF	0.160	0.160	0.167	-
CAQ	0.173	0.180	0.200	-
CAP	0.146	0.180	0.182	-
CRTF	0.155	0.155	0.208	-
CRQ	0.155	0.122	0.171	-
CRP	0.177	0.144	0.167	-
LexQE	**0.386**	**0.313**	**0.327**	***Yes*, p = 0.0232**
LexRR	0.186	0.180	0.177	-
SentQEP	0.240	0.206	0.225	*No*, p = 0.4575
SentRRP	0.186	0.153	0.171	-
SentQEF	0.306	0.300	0.299	*Minimal*, p = 0.1146
SentRRF	0.186	0.180	0.186	-

Table 8: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for multi-word semi-ambiguous queries.

approach did yield very good results when used with query expansion techniques, as it is delayed by the computation of query specific sentence scores, it only makes a good candidate for domain specific search engines (e.g., medical), where some additional time can be traded for a better output.

At the other end, both the Clustering methods and the Lexical Compounds ones provide very quick results, as their computation demanding step can be implemented off-line at indexing time, thus making them (especially the latter one) very suitable candidates for real world search applications.

5. CONCLUSIONS AND FURTHER WORK

In this paper we proposed to select query expansion terms for web search by adapting summarization and natural language processing techniques to extract these supplementary keywords from locally stored desktop documents. We investigated three possible approaches, based on (1) summarizing the entire desktop data, (2) summarizing only the desktop documents relevant to the current user query, and (3) extracting dispersive lexical compounds from relevant desktop resources. Our experiments showed an improvement in Mean Average Precision of up to 81.57% for single word queries and up to 21.11% for multi-word queries when compared to regular Google web search.

While some of our approaches did perform very well already, in future work we intend to investigate their performance using a study targeted towards persons with very limited textual data, as well as to analyze the possibilities to automatically generate and exploit additional metadata which would further increase web search

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.453	0.413	0.431	-
CATF	0.426	0.400	0.403	-
CAQ	0.453	0.413	0.433	*No*, p = 0.4914
CAP	0.413	0.420	0.399	-
CRTF	0.466	0.466	0.478	*No*, p = 0.2920
CRQ	0.511	0.477	0.492	*Minimal*, p = 0.1627
CRP	0.466	0.488	0.456	*No*, p = 0.3892
LexQE	**0.723**	**0.663**	**0.709**	***Highly*, p = 0.0012**
LexRR	0.413	0.433	0.439	*No*, p = 0.4602
SentQEP	0.560	0.506	0.603	*Yes*, p = 0.0188
SentRRP	0.426	0.393	0.389	-
SentQEF	0.440	0.460	0.496	*Minimal*, p = 0.2321
SentRRF	0.440	0.413	0.440	*No*, p = 0.4575

Table 9: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for single word clear queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.640	0.620	0.601	-
CATF	0.613	0.626	0.605	*No*, p = 0.4563
CAQ	0.613	0.620	0.574	-
CAP	0.600	0.613	0.592	-
CRTF	0.577	0.566	0.528	-
CRQ	0.488	0.544	0.486	-
CRP	0.511	0.555	0.511	-
LexQE	**0.626**	**0.613**	**0.609**	***No*, p = 0.4133**
LexRR	0.626	0.626	0.608	*No*, p = 0.4070
SentQEP	0.600	0.588	0.613	*No*, p = 0.4050
SentRRP	0.613	0.620	0.604	*No*, p = 0.4664
SentQEF	0.500	0.493	0.592	-
SentRRF	0.613	0.613	0.585	-

Table 10: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for multi-word clear queries.

performance not only for such users, but also for those already possessing vast textual resources.

6. ACKNOWLEDGEMENTS

This work was supported by the Nepomuk project funded by the European Commission under the 6th Framework Programme (IST Contract No. 027705).

7. REFERENCES

[1] P. G. Anick and S. Tipirneni. The paraphrase search assistant: Terminological feedback for iterative information seeking. In *Proc. of the 22nd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1999.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] J. Budzik and K. Hammond. Watson: Anticipating and contextualizing information needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science*, 1999.

[4] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of the 21st Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1998.

[5] P. A. Chirita, C. S. Firan, and W. Nejdl. Pushing task relevant web links down to the desktop. In *Proc. of the 8th ACM Intl. Workshop on Web Information and Data Management held at the 15th Intl. ACM CIKM Conference on Information and Knowledge Management*, 2006.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.200	0.180	0.181	-
CATF	0.200	0.193	0.197	*No*, p = 0.3973
CAQ	0.200	0.193	0.206	*No*, p = 0.3542
CAP	0.186	0.206	0.209	*No*, p = 0.3345
CRTF	0.177	0.233	0.212	*No*, p = 0.3899
CRQ	0.200	0.222	0.207	*No*, p = 0.3588
CRP	0.177	0.222	0.195	*No*, p = 0.3129
LexQE	**0.440**	**0.393**	**0.428**	***Highly*, p = 0.0029**
LexRR	0.213	0.200	0.206	*No*, p = 0.3519
SentQEP	0.306	0.253	0.350	*Highly*, p = 0.0031
SentRRP	0.226	0.186	0.189	*No*, p = 0.4530
SentQEF	0.320	0.300	0.327	*Highly*, p = 0.0045
SentRRF	0.200	0.186	0.206	*No*, p = 0.3603

Table 11: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for single word clear queries.

Algorithm	P@5	P@10	MAP	Signif. for MAP
Google	0.466	0.400	0.462	-
CATF	0.346	0.380	0.344	-
CAQ	0.346	0.346	0.328	-
CAP	0.346	0.360	0.341	-
CRTF	0.377	0.333	0.365	-
CRQ	0.311	0.311	0.352	-
CRP	0.333	0.322	0.353	-
LexQE	0.440	0.400	0.414	-
LexRR	0.373	0.373	0.410	-
SentQEP	**0.493**	**0.486**	**0.505**	***Minimal*, p = 0.2488**
SentRRP	0.400	0.380	0.412	-
SentQEF	0.386	0.414	0.401	-
SentRRF	0.293	0.326	0.307	-

Table 12: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* answers selected by our testers for multi-word clear queries.

[6] P.-A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using odp metadata to personalize search. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[7] P.-A. Chirita, D. Olmedilla, and W. Nejdl. Pros: A personalized ranking platform for web search. In *Proc. of the 3rd Intl. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2004.

[8] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *SIGIR*, 1993.

[9] D. R. Cutting, J. O. Pedersen, D. R. Karger, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *SIGIR*, 1992.

[10] S. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (iq) for contextualized search. In *Proc. of the 27th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2004.

[11] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

[12] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelli. and Agent Sys.*, 1(3-4):219–234, 2003.

[13] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *Proc. of the 22nd Intl. ACM SIGIR*

Conf. on Research and Development in Information Retrieval, 1999.

[14] T. Haveliwala. Topic-sensitive pagerank. In *In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii*, May 2002.

[15] G. Jeh and J. Widom. Scaling personalized web search. In *Proc. of the 12th Intl. World Wide Web Conference*, 2003.

[16] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[17] K. S. Jones, S. Walker, and S. Robertson. Probabilistic model of information retrieval: Development and status. Technical report, Cambridge University, 1998.

[18] S. Katz. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–59, 1996.

[19] A. M. Lam-Adesina and G. J. F. Jones. Applying summarization techniques for term selection in relevance feedback. In *Proc. of the 24th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[20] D. Lawrie and W. Croft. Generating hierarchical summaries for web searches. In *Proc. of the 26th Intl. ACM SIGIR Conf. on Research and Development in Information Retr.*, 2003.

[21] D. Lawrie, W. B. Croft, and A. L. Rosenberg. Finding topic words for hierarchical summarization. In *Proc. of the 24th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[22] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. on Knowledge and Data Eng.*, 16(1):28–40, 2004.

[23] H. Luhn. Automatic creation of literature abstracts. *IBM Journ. of Research and Development*, 2(2):159–165, 1958.

[24] G. Miller. Wordnet: An electronic lexical database. *Communications of the ACM*, 38(11):39–41, 1995.

[25] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proc. of the 21st Intl. ACM SIGIR Conf. on Research and Development in Information Retr.*, 1998.

[26] T. Nomoto and Y. Matsumoto. A new approach to unsupervised text summarization. In *Proc. of the 24th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[28] D. R. Radev, H. Jing, M. Stys, and D. Tam. Centroid-based summarization of multiple documents. *Inf. Process. and Management*, 40(6):919–938, 2004.

[29] B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Syst. J.*, 39(3-4):685–704, 2000.

[30] S. E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *TREC*, 1999.

[31] J. Rocchio. Relevance feedback in information retrieval. *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 313–323, 1971.

[32] D. Rose, R. Mander, T. Oren, D. Ponceleon, G. Salomon, and Y. Wong. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *Proc. of the 16th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1993.

[33] M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Proc. of the 22nd Intl. ACM SIGIR Conf. on Research and Development in Information Retr.*, 1999.

[34] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proc. of the 13th Intl. WWW Conf.*, 2004.

[35] D. Sullivan. The older you are, the more you want personalized search, 2004. http://searchenginewatch.com/searchday/article.php/3385131.

[36] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[37] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proc. of the 21st Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1998.

[38] E. Volokh. Personalization and privacy. *Commun. ACM*, 43(8), 2000.

[39] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Inf. Process. and Management*, 24(5), 1988.

[40] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. *Comput. Networks*, 31(11-16), 1999.

[41] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proc. of the 27th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2004.