

# Pushing Task Relevant Web Links down to the Desktop

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

Searching the web has become a task in many people's work, without which subsequent tasks would be hard to carry out or even impossible. But as people tend to have less time for querying the web or even for searching their personal computer for information they need, it becomes common to skip information gathering activities like trying to find useful resources on the web because of the "effort" it takes to query a web search engine. In this paper we propose to use software agents that collect useful web specific related information which would otherwise not be viewed at all. More specifically, we present two new algorithms to automatically search the web and recommend URLs relevant to user's current work, defined through his or her active personal desktop documents. Our experiments show our proposed algorithms, Sentence Selection and Lexical Compounds, to yield significant improvement over simple Term Frequency based web query generation, which we used as a baseline.

## 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—*Linguistic processing*; H.3.5 [Information Storage and Retrieval]: Online Information Services—*Web-based services*

## General Terms

Algorithms, Experimentation, Theory

## Keywords

Personalized Web Search, Just-in-Time Information Retrieval, Document Summarization, User Profile

## 1. INTRODUCTION

Information is a necessity for many of our tasks, at work or in our personal life, and we simply find no easier way than searching the web for answers to problems. The World Wide Web has grown

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIDM'06, November 10, 2006, Arlington, Virginia, USA  
Copyright 2006 ACM 1-59593-524-X/06/0011 ...\$5.00.

to be an extraordinary vast, but mostly unstructured and hard to precisely access information source. We therefore have to provide means to extract desired information with least effort. In many situations, when a user works on a text file (either reading a web page or a document, or writing an e-mail or a document) he needs other relevant information sources. Characterizing such a file with a few words in order to query a web search engine manually is less intuitive and demands additional effort from the user, so that he would have a strong tendency to sacrifice the results he would have obtained in favor of less work. On the other hand, it would be effortless for the user to just take a look at some recommended web links presented to him or her by a software agent running in the background.

Users need additional information related to their work in order to increase the quality of edited content, as well as significantly decrease task working time [23]. In order to minimize user effort locating suitable data sources, the PC environment should support the user as much as possible. To accomplish this, we have to understand the tasks the user carries out, then from this user/task-profile extract appropriate information used to query different information sources. Results from a software agent - which tries to understand the user's task and then extracts a list of keywords from a document or a collection of documents - may not be as effective or relevant as results manually retrieved from a search engine - used directly and with a well-known intention by humans. Nevertheless, there are cases where people would not start a web search process by themselves, or not bother to use a search engine in specific stages of their work. As already discussed in previous work, we would like to provide Information Management Assistants (IMA) [2] or Just-in-Time Information Retrieval (JITIR) Agents [23] to support the user in this context.

In this paper, we present two new algorithms to automatically search the web and recommend URLs relevant to the current task of a user. The algorithms extract information from the currently active user document and from this document's context (i.e., similar documents residing on the user's PC). Information extracted by different algorithms from these documents will then be synthesized into a list of keywords used to query a web search engine. The experiments performed in Section 5 show that our two keyword extraction algorithms, Sentence Selection and Lexical Compounds, show an average improvement of 18% and 41% respectively over the Term Frequency algorithm used as baseline. Using desktop documents yields better results than selecting emails as the input file because of the restricted amount of information usually found in emails. When selecting a web page as input file, it is more difficult for the software agent to extract user task or user interests as web pages are not user dependent.

The paper is structured as follows: The next section discusses

previous work in this area regarding JITIRs as well as keyword extraction from text documents. Section 3 presents three methods (one baseline and two enhanced algorithms) used to extract relevant search keywords from text documents in order to form web search queries. Section 4 describes how these keywords are used to query web search engines. Section 5 presents and discusses experiments and their results. Section 6 concludes the paper and discusses some future research issues.

## 2. PREVIOUS WORK

Relevant work includes summarization algorithms, just-in-time information retrieval agents (JITIRs), and personalized search. Only very few previous publications combine these areas and even fewer address both the PC Desktop and the World Wide Web. The work of Teevan et al. [26] is the only one exploiting desktop data for web search. They modified the query term weights from the BM25 weighting scheme [10] to incorporate user interests as captured by the desktop index, which is related to our approach. However, they select their web search query based on explicitly user entered keywords which they refine using expansion terms from the Top-K documents returned by the web search engine, whereas we use an *automatically* generated query from user's currently active document.

The following sections discuss important results from the two research areas JITIRs and Summarization.

### 2.1 Just-in-Time Information Retrieval

Rhodes and Maes [23] describe a new class of software agents, that of Just-in-Time Information Retrieval Agents (JITIRs), which are software agents that proactively present potentially valuable information based on a person's local context in an easily accessible yet non-intrusive manner. JITIRs provide useful or supporting information that is relevant to the current task, research results demonstrate that such systems encourage use of information that would not otherwise be looked at. Rhodes presents the kinds of agents: (1) The Remembrance Agent [20], an agent incorporated in Emacs, which continually presents a list of documents, from the PC desktop or from various databases, that are related to the current document being written or read. (2) Margin Notes [22] is a JITIR agent that automatically rewrites web pages as they are loaded into the browser adding hyperlinks to personal files, each HTML section receiving its own annotation in addition to a general entire page annotation. (3) Jimminy [21] is a third type of JITIR agent that provides information based on a person's physical environment. By using a shoulder-mounted wearable computer containing different environment-aware sensors, suggestions are presented to the user on an head-mounted display. All three JITIR agents presented by Rhodes use the same back-end system, called Savant, which computes the relevance score for each annotation based on co-occurrence of words using a term frequency / inverse-document frequency (*TFIDF*) method [24] and the Okapi weighting scheme [28]. The power of Savant comes from a strong template matching system that recognizes document structures and parses different fields. As necessary features of a JITIR agent, Rhodes [23] lists proactivity, the presentation of information in an accessible yet non-intrusive manner, and awareness of the user's local context.

Budzik and Hammond [2] introduced the concept of Information Management Assistants (IMAs). IMAs automatically discover related material on behalf of the user by serving as an intelligent intermediary between the user and information retrieval systems. Budzik et al.'s Watson system runs in the background on a user's computer and when possible retrieves web links simi-

lar to the active web page in the browser (Microsoft Internet Explorer or Mozilla FireFox), Microsoft Word, or Microsoft Outlook. These links are retrieved using different information sources like AltaVista Web Search and other user definable repositories, and are then presented in a separate window after a simple URL and page title based clustering is applied. For Watson, weighting terms in order to form a search query is highly dependent on a document's internal layout and word highlighting. In addition to document specific heuristics, Watson uses a standard information retrieval *TFIDF* weighting scheme after removing stop words, combined with word position information in the given document. In addition, Watson allows users to enter explicit queries, which are then refined by means of context related information extracted from the active document. In subsequent work [3], Budzik et al. argue about the usefulness of the retrieved results, stating that an IMA should focus on retrieving not only similar documents but documents that are relevant and useful in purposeful and interesting ways. In their experiments they assess that the similarity of a result accounts for about a quarter of the variance in the utility of a result.

Other JITIRs include Letizia [15], an agent which creates a short term user profile by compiling keywords contained in visited web pages, and highlights outgoing links from the current web page that match the profile. WebWatcher [9] is a system similar to Letizia, highlighting hyperlinks that match a user's stated interest. Maglio et al.'s SUTOR [17] uses multiple agents to watch several applications in parallel and provide results for the overall activity. RADAR [5] is a different front end for the Remembrance Agent [20] described earlier that uses Microsoft Word instead of Emacs and displays suggestions in a separate window, with Savant as information-retrieval engine. Finally, there are domain-specific JITIR agents like The Peace, Love, and Understanding Machine (PLUM) system [6] which adds hyperlinks to disaster news stories.

### 2.2 Summarization

Automated summarization deals with concatenating text-span excerpts (i.e., sentences, paragraphs, etc.) into a human understandable document summary, it dates back to the 1950's [16]. With the advent of the World Wide Web and large scale search engines, increased attention has been focused towards this research area and several new approaches have been proposed. The diversity of concepts covered by a document was first explored by Carbonell and Goldstein [4]. They proposed using Maximal Marginal Relevance (MMR), which selects summary sentences that are both relevant to the user query and least similar to the previously chosen ones. Later, Nomoto and Matsumoto [19] 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 approach is 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 [8, 7]. Consequently, more search specific applications of summarization have been proposed. Zeng et al. [29] 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 retrieved documents [13], or to organize topic words extracted from textual documents [14, 25].

## 3. EXTRACTING RELEVANT QUERY KEYWORDS

We will consider one current document - an email, a web page or another document containing text - as an input file. From this

input file the software agent has to deduct the task the user is currently working on. The current task will be represented as series of keywords. These keywords have to cover the topics present in the analyzed document, and represent each topic accurately. We present three algorithms to accomplish this tasks, to be described in the following subsections: Term Frequency (*TF*), Sentence Selection (*SS*) and Lexical Compounds (*LC*). The number of keywords is limited to 20 or the number of sentences the input file contains multiplied by two, whichever number is smaller, to ensure that only relevant keywords are extracted. If we use a larger number of keywords, then the probability that some of these keywords do not represent the current user task grows exponentially.

### 3.1 Term Frequency

The simplest approach is to extract terms based on their Term Frequency. As a simple, statistical term weighting scheme, this approach is used as a guideline in Information Retrieval (IR). In contrast to IR, we do not use also IDF weighting, this being unnecessary for such a restricted collection as the personal computer. In our approach a score is associated with each term, this score being composed of a linear descending from 1 to 0.5 position score multiplied by its term frequency within the given document, such that the first term's score is its term frequency and the last term's score within the document is only half of its term frequency, like shown in the following formula:

$$TermScore = \left[ \frac{1}{2} + \frac{1}{2} \cdot \frac{nrWords - pos}{nrWords} \right] \cdot TF$$

where *nrWords* is the total number of terms in the document and *pos* is the current term position. *TF* represents the previously described term frequency.

For terms appearing more than once throughout a document, the score for the first appearance was used and only the top 20 keywords are kept, as done previously in [2]. The terms were sorted descending according to their computed score for ranking purposes. Although the extra ranking possibility offered by file specific information could present itself to be a powerful topic detection and accurate keyword extraction mechanism, in this work we focused on finding algorithms that should not be constrained by the file type they analyze.

This algorithm is similar to the work of Budzik and Hammond [2]; thus, the Term Frequency algorithm will be used throughout this paper as a baseline to compare our other two algorithms with.

### 3.2 Sentence Selection

There exist quite several approaches to sentence based summarization. However, we chose to start from [12], which had actually the goal to select terms for query expansion. From the input document we extracted the most salient sentences with respect to the user query by evaluating the following formula in the case of keyword extraction from the current document:

$$SentenceScore_{single} = \frac{SW^2}{TW}$$

This formula is based on Luhn's cluster measure [16] 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 the log of the frequency) is above a threshold as follows:

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

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

When performing Sentence Selection over similar desktop documents we use the following extended formula:

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

The second term of this formula comes from the work of Tombros and Sanderson [27] and is computed using the ratio between the square number of query terms present in the sentence (*NQ*) and the total number of terms (*TQ*) 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. Note that in our case the query is formed of keywords extracted from the initial input file.

Lam et al. [12] 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 when using the sentence selection approach, first because many of them have no title other than the file name which can be arbitrary chosen or can contain user specific language, and second because for the sentence selection approach we try to focus on finding all the topic describing sentences based solely on the words contained and regardless of their position, as some documents, may cover more than one topic with no importance loss throughout the document.

Once these sentence scores were computed, we sought for query expansion terms combining two approaches: (1) extracting the terms with the highest term frequency in the documents the sentences originate from, over the top 9 sentences, as reported in [12], and (2) using the same approach but over the top 2% sentences. The new latter approach is motivated by previous findings that longer documents tend to contain more topics, and thus more content words [11] and does indeed slightly improve over the former one. We combined these two approaches as to use 2% sentences per document but still a minimum of 9 sentences.

### 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.

We inspect a given document for all its lexical compounds of the following form, as defined in [1]:

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

Although we performed this step at run-time (using WordNet [18]), it could be easily run off-line, at indexing time to generate all compounds for all the documents in the index. Thus, once these lexical constructions have been identified, they are sorted depending on their dispersion within the document and the terms of the most frequent ten compounds (as most of the compounds consist of two words) are used as query expansion keywords.

## 4. RECOMMENDING RELATED WEB PAGES

We now need to efficiently use these keywords in order to locate web pages relevant to user's current task. After extracting task descriptive keywords from the active document, one of two methods can be applied for each of the described algorithms.

**Exploiting only the Currently Active Document.** The first method we used is regarding the currently active document as sufficient task context, as to create a complete current user task image. Since the output from the algorithms is a sorted by importance list of terms, the selected keywords can be used directly to form a query, which is then used in conjunction with Google API<sup>1</sup> to retrieve the relevant URLs to be presented to the user. The only restriction applied is limiting the number of keywords used to form the query to a maximum of 20 terms or twice the number of sentences in the analyzed document. Thus, the web search query is formed by just concatenating the list of sorted space delimited keywords, since the Google API does not allow us to specify different weights for each query term.

**Exploiting the Full Context of the Currently Active Document.** There are cases when the currently active document does not provide sufficient information for the software agent to fully understand the user task. Therefore we need to supply additional information by using also other context related documents from the user's PC. For the software agent this translates into retrieving similar documents from the desktop by using the previously computed keywords (i.e., the output from each algorithm). The document retrieval is being achieved using the Lucene search engine. The number of keywords for finding similar desktop documents may be diminished until at least two documents are retrieved that contain all the searched keywords. For time saving reasons and due to the just-in-time nature of the software agent, only the top 10 desktop documents from the hits list are considered. The same keyword extraction algorithm is then applied to these retrieved desktop documents and another list of keywords is extracted. The two keyword lists, the one resulting from the initial document and the one extracted from the related desktop documents, are then combined in such a way that the number of keywords from each source is equal, but the weight of the keywords extracted from the initial document the user is working on is higher. When used with Google API, this translates into creating a web search query consisting of the first (maximum) 10 keywords from the initial document, followed by keywords from the desktop, to the maximum sum of 20 keywords.

The final query issued to Google API consisted only of English words listed in WordNet [18] as we noticed that without this restraint, some very user specific terms that are not found within documents on the internet could also be included. We also excluded keywords consisting of less than three characters, as they are usually abbreviations, and the abbreviations coming from the user desktop can be in many cases created by that user and have different meanings even throughout the same interest community.

## 5. EXPERIMENTS

### 5.1 Experimental Setup

Prior to running the main software agent, the entire desktop content i.e., all the files on the user's computer have to be indexed for future faster retrieval. The Lucene<sup>2</sup> information retrieval system suits our interests best, given its rapid search algorithms, its flexibility and adaptivity and last but not least its cross-platform portability. We have used Lucene to index the user's desktop contents for subsequent faster access to those files. 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

specified by each person<sup>3</sup>

Following the work of Lam et al. [12], 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, in all cases but one, we only used  $TF$ , rather than  $TFIDF$ , as one very frequent local term (e.g., PageRank) might in fact be rather rare in the web. A large stopword list was used to initially remove any possible misleading terms. Also, summarization was achieved employing  $\log TF$  rather than  $TF$  in order to avoid having some too frequent terms mislead the results. The variants of  $TF$  and  $IDF$  we used were as follows:

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

$$IDF_{t_k} = \log\left(1 + \frac{N}{DF_{t_k}}\right)$$

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  $t_k$ .

We started our analysis by manually inspecting the output of each keyword extraction algorithm. In most cases, we found it to be quite representative for the original document or set of documents, i.e. extracting the main topic keywords from the given text. However, as in other similar works (e.g., [12]), our main objective measure of quality was its overall precision in recommending related web page, and thus we will focus our presentation only towards this aspect.

To evaluate the precision of our personalization algorithms we interviewed 13 researchers in different computer science areas and education. In the first phase of the evaluation they installed our desktop indexer, then they chose three English language documents related to their everyday activities as follows:

- One *email* consisting of at least some sentences of written text, other than greetings;
- One *text document*, preferably text the user has written himself<sup>4</sup>;
- One *web page* that would otherwise also be watched, saved directly from the internet.

The email and the text document were used in three forms, using only the beginning (*min*), an introductory part (*med*) or the whole document (*max*) as to simulate a work in progress and to be able to test the algorithm for his initial purpose, that of helping the user in completing his work. As the web page does not represent the work actually done by the user, this subdivision was not necessary, so only the full HTML body was used. This subdivision of the files would result in actually using seven different input file types - 3 email sizes (the first 5, 15 and all the sentences), 3 document sizes (the first 10, 25 and all the sentences) and 1 web page. The number of sentences used for simulating the user's work in progress is also presented in Table 1.

Each one of the three described algorithms was (1) first used only on the input file itself and (2) afterwards on the given file and other similar documents from the user's PC desktop. This results in a total of six used algorithms.

For each input file selected by the user the Top-5 results from Google API (as a software agent should not present the user with

<sup>1</sup><http://api.google.com>

<sup>2</sup><http://lucene.apache.org>

<sup>3</sup>Although this definition was targeted at single-user PCs, one could easily extend it to multiple-user ones.

<sup>4</sup>Usually, the researchers used articles for this task.

Input type	min	med	max
<b>Email</b>	5	15	all
<b>Document</b>	10	25	all
<b>Web page</b>	-	-	all

**Table 1: Number of sentences used for the simulation of a work in progress**

overwhelmingly too much information) generated by the 2 versions of the 3 algorithms we presented in Section 3, applied to 3 different file sizes for the email and the document and 1 for the web page (in total, 6 “algorithms” over 7 inputs, thus 42 result sets), were shuffled into one set so that the user was neither aware of the algorithm, nor of the ranking of each assessed URL. Thus, each subject had to assess about 170 URLs for three document types. Overall, 39 input files were selected and over 2,000 URLs were evaluated during the experiment. For each of these URLs, the testers had to give two marks ranging from 0 to 2 thus rating first the relevancy to the input file type as (0) not relevant, (1) relevant and (2) highly relevant; secondly they also rated the personal overall usefulness of the recommended web pages as (0) not useful at all, (1) useful and (2) very useful. The output quality was evaluated in terms of Mean Average Precision over the first 5 results (MAP@5) as well as precision at the first 1 through 5 positions of the resulted ranking (P@1 .. P@5). Finally, all our results were tested for statistical significance using T-tests (i.e., we tested whether the improvement over the simple intuitive Term Frequency algorithm output<sup>5</sup> is statistically significant).

Due to space limitations, for presenting the following result values we used a relaxed evaluation schema with regard to the marks given by the test subjects. Thus all the web links found to be relevant or highly relevant (marks 1 or 2) are treated as unitary. In fact, when using a strict evaluation, i.e. considering only highly relevant web links as relevant, the drawn conclusions are similar. More, when judging the usefulness of the web links with respect to the current user task, the relative performance of the algorithms with respect to each other remains mostly unchanged.

In all the forthcoming tables, we will use the following labeling:

- **TF**: The Term Frequency algorithm run on the input file only;
- **SS**: The Sentence Selection algorithm run on the input file only;
- **LC**: The Lexical Compounds algorithm run on the input file only;
- **TFD, SSD, LCD**: The previous three algorithms run on the input file and at most 10 similar files from the user’s PC desktop;
- **Mail-5, Mail-15, Mail**: The first 5, 15, or all the sentences from the selected email;
- **Doc-10, Doc-25, Doc**: The first 10, 25, or all the sentences from the selected text document;
- **Web**: The entire selected web page.

## 5.2 Results

**Email.** The recall precision when searching for relevant web links to the current user task using an email as the input file type, with the three sizes the email was divided into, is presented graphically in Figures 1, 2, and 3. The three figures depict the precision, scaled from 0 to 1, as judged by the test subjects for all the five top links retrieved. We can see that although the average precision

<sup>5</sup>Whenever necessary, we also tested for significance the difference between pairs of the algorithms we proposed.

relies in the same interval for all the three email sizes, there are significant changes in the algorithms’ performances. As the email grows bigger, the desktop compensation importance declines, because the user task can be more easily understood from the email itself. When considering only the first 5 sentences of an email, the algorithms show better results when applied over the desktop content also. For the first 15 sentences used, performances of the algorithms are similar, regardless of the extension over the desktop. As for the whole email, extending the user task with some general user profile keyword extracted from similar documents, usually yields poorer results.

From the algorithmic perspective, for the beginning of the email, the *SSD* and *SS* approaches show the best results, but as the email information increases, the precision for these algorithms is diminished, the lexical compound algorithms taking over. Although the precision is generally low, the *SS* and *LC* approaches show better results than *TF* without desktop, with *LC* at a significant level (p-value 0.02). Emails are usually very close related to the user personal interests, i.e. the user’s desktop content; as a result, using the desktop usually brings an improvement, depending on the email size, especially with algorithms based on simpler statistics, such as *TF*.

Finally, we noticed that the recommended web results strongly depend on the location of the main topic words inside the email. When writing emails, users can get to the main point right after the greeting, or use some more sentences to describe a general situation or reply to some previous matter first. There are also often cases when replies also include the previous mail but have slight topic changes, where the users feel the main topic is the newer one, but the algorithms are not aware of the different topics contained, what results in extracting keywords of the same importance from throughout the whole email body. Usually the main topic of the email resides within the first 5 or 15 sentences, and therefore the precision slightly decreases on average as the email size increases. However, this cannot be generalized for every email. The best email description was extracted from the first 15 sentences, as only the first 5 include a lot of noise because of the introductory greeting typical to emails and except for very long emails, the email signature, usually present at the end of the email also adds significant noise. We are currently investigating more complex Topic Detection and Tracking (TDT) algorithms, which will allow us to better focus our keyword extraction algorithms towards the typically more relevant parts of each email.

**Text Document.** Results regarding the usage of a text document as the input file (Figures 4, 5, and 6) reveal that both Lexical Compounds methods (*LC* and *LCD*) yield the best results. Another important aspect of note is that document size matters. In general, the bigger the document, the more relevant information should be available. However, for those subjects who used articles (may them be scientific, or news), utilizing the full-document contents resulted in slightly worse results than when considering only the first 10 or 25 sentences. This is because in such cases, the first sentences abstract the entire text, and thus contain more topic-descriptive keywords than the rest of the document. For overall significance (the differences between all pairs of algorithms), p-value is 0.076, lowest for all input data. This is explainable, since manually edited documents contain both a personal language and enough interest-related words.

If only the currently active document is considered, then again the Lexical Compounds approach yields the best results, at a p-value of 0.097 versus *TF*. When this information is expanded with keywords extracted from other (similar) desktop documents, we observe a minimal loss of quality in the ratings, indicating that the

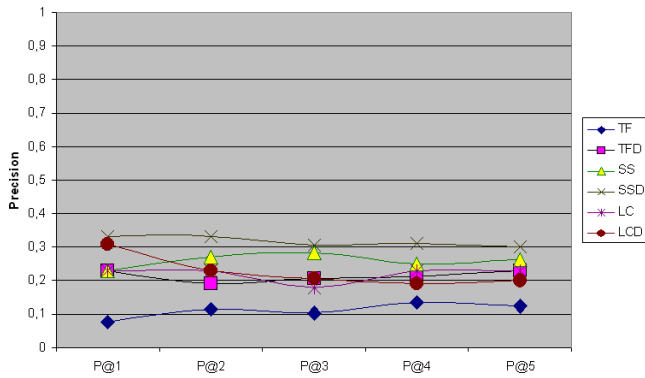


Figure 1: Precision at 1..5 considering only the first 5 sentences of an email

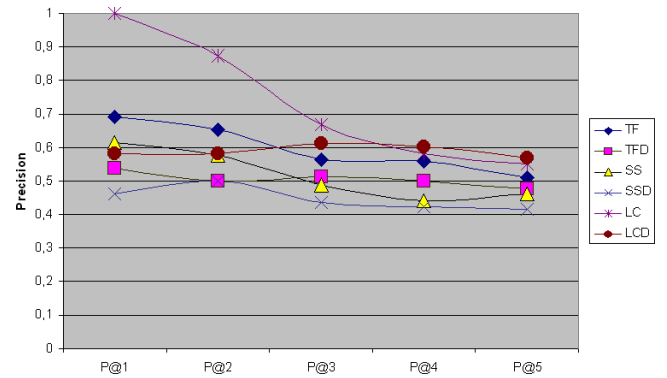


Figure 4: Precision at 1..5 considering only the first 10 sentences of a text document

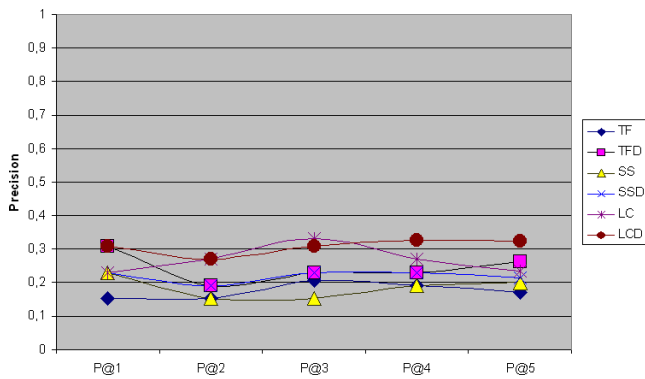


Figure 2: Precision at 1..5 considering only the first 15 sentences of an email

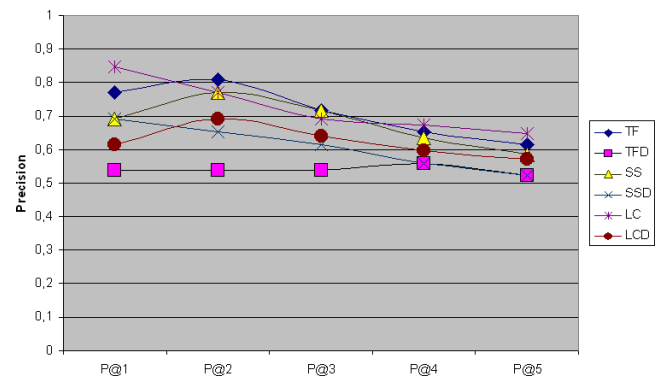


Figure 5: Precision at 1..5 considering only the first 25 sentences of a text document

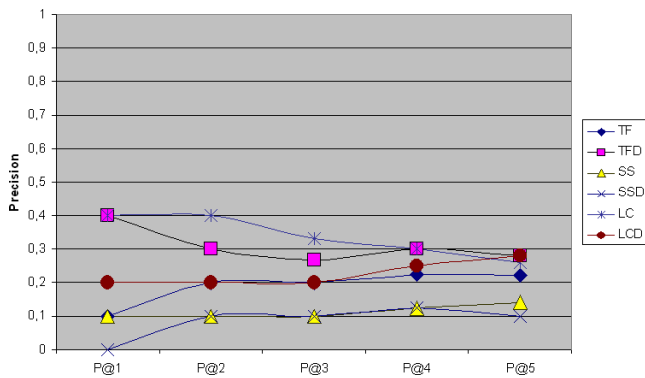


Figure 3: Precision at 1..5 considering the entire text of an email

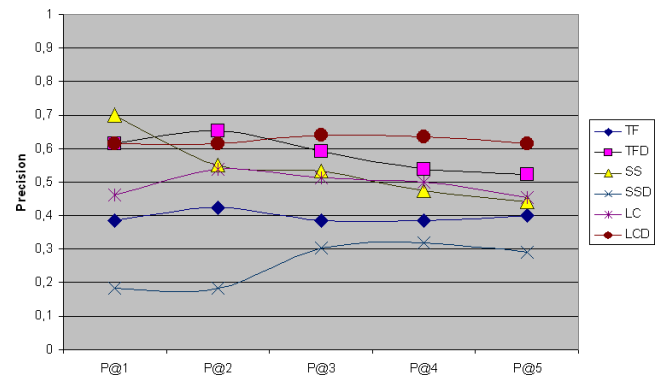


Figure 6: Precision at 1..5 considering the entire text document

active document provides a much clearer description of the current user task.

**Web page.** The results for the experiments with a web page as the input file type are depicted in Figure 7. Only the full text of the web page is considered as we do not simulate the user writing the web page step by step, but the user surfing the internet in search for knowledge.

The results show that the desktop supported keyword extraction variants usually produce results less precise than their single-document equivalents. This could be mainly because of differences in language use between local resources and global web pages. Although the simple term-frequency-based algorithm augmented by

using similar desktop documents shows in this case the best performances overall, the statistical significance is not satisfied. Moreover, it is important to note that the desktop enhanced variants of our algorithms also require additional computation time, in order to analyze the desktop related documents. Thus though interesting from a theoretical perspective, they are less suitable for a real-life application, which should provide the web recommendations preferably within less than 1 second. When considering only the current document, the *SS* approach performs best, followed by *LC*.

**Conclusions.** Like shown in Figure 8 and in Table 2, we can judge the usefulness of also including keywords from the desktop

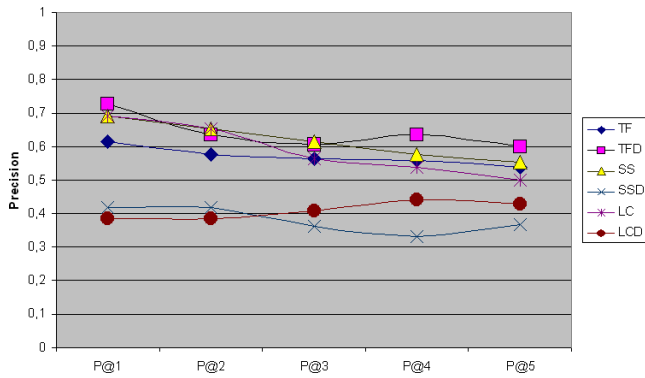


Figure 7: Precision at 1.5 considering the *entire* text of a *web page*

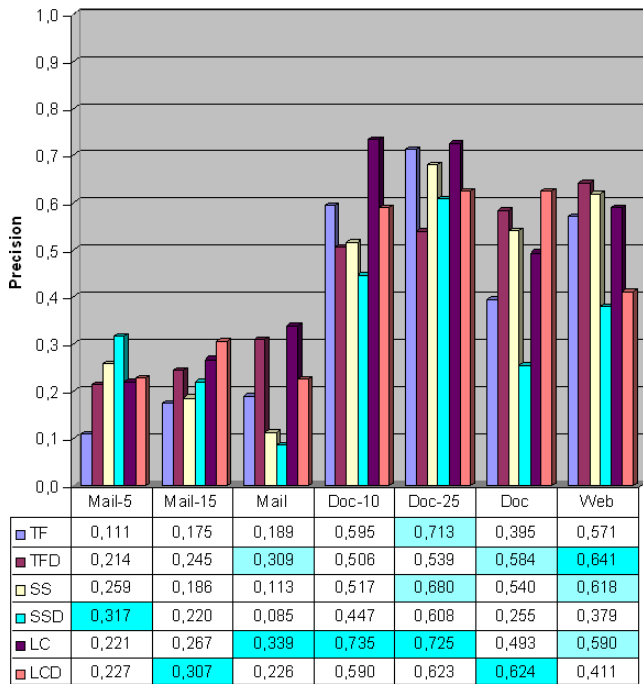


Figure 8: Mean average precision per input file type

context in the query used to retrieve the relevant web links as very important. Using similar desktop documents improves precision, except for the cases when the whole document text or the web page is used as input. This is mainly due to desktop documents being too similar to the input file, in case of using the text document, or the desktop language differing from the web language in such a way that for a web page, the same syntactical structuring cannot be found in both the input file and the PC desktop. We also found the confirmation of our starting idea, that *LC* and *SS* perform better than the *TF* baseline, except for *Doc-10* and *Doc-25* where only *LC* performs better than *TF*. The explanation for this exception is that most of the used test input files were articles, where the first sentences are the paper abstract, and hence there is no need for further summarization.

**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

Input type	TF	TFD	SS	SSD	LC	LCD
<b>Email</b>	0.158	<b>0.256</b>	0.186	0.207	<b>0.276</b>	<b>0.253</b>
<b>Document</b>	0.568	0.543	0.579	0.437	<b>0.651</b>	<b>0.612</b>
<b>Web page</b>	0.571	<b>0.641</b>	<b>0.618</b>	0.379	<b>0.590</b>	0.411

Table 2: Average precision for each algorithm per input file type

approach did yield good results when used with context 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 Term Frequency and the Lexical Compounds methods 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. As the Lexical Compounds method resulted in the best precision scores, we think it is the best suited for recommending web pages in such a scenario.

## 6. CONCLUSIONS AND FURTHER WORK

In this paper we proposed two new algorithms used to extract keywords from the user's documents in order to describe the current user task. The approaches, Sentence Selection and Lexical Compounds, adapt summarization and natural language processing techniques to extract these keywords from the active document or additionally from locally stored desktop documents. We investigated the performance of these with respect to three types of input files, namely emails, text documents and web pages. Our experiments showed an improvement over the baseline in Mean Average Precision of 18% for the *SS* approach and 41% for the *LC* approach, with a maximum improvement of 187% of the *SSD* algorithm over the *TF* method when using only the first 5 sentences of an email as input. The performance of the *LC* algorithm improves (in terms of Mean Average Precision) the performance of the *TF* approach for all different input file types. When using additional context information from the desktop, already the simple *TFD* approach yields an improvement in MAP of 31% over the *TF* approach, thus leaving only space for 1% improvement for the *LCD* algorithm over *TFD*.

In future work we want to investigate means to distinguish between different input file types and even file types categories, like personal descriptive emails, specialized concise emails, articles, news, stories, novels, etc. because of differences in the nature of the language used and the position of important keywords throughout these documents. When analyzing emails, we are investigating the means to extract only the topic related part of the email body, disregarding additional noise like introduction, signature or off-topic personal additions. Finally, we want to analyze other summarization and natural language processing techniques to extract keywords from documents.

## 7. ACKNOWLEDGEMENTS

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

## 8. 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] 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.
- [3] J. Budzik, K. J. Hammond, L. Birnbaum, and M. Krema. Beyond similarity. In *Working notes of the AAAI 2000 Workshop on AI for Web Search*. AAAI Press, 2000.
- [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] I. B. Crabtree, S. Soltysiak, and M. Thint. Adaptive personal agents. *Personal Technologies*, 2(3):141–151, 1998.
- [6] S. Elo. Plum: Contextualizing news for communities through augmentation, 1995.
- [7] 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.
- [8] 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.
- [9] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI97*, 1997.
- [10] K. S. Jones, S. Walker, and S. Robertson. Probabilistic model of information retrieval: Development and status. Technical report, Cambridge University, 1998.
- [11] S. Katz. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–59, 1996.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] H. Lieberman. Letizia: An agent that assists web browsing. In *In Proceedings of IJCAI 95*. AAAI Press, 1995.
- [16] H. Luhn. Automatic creation of literature abstracts. *IBM Journ. of Research and Development*, 2(2):159–165, 1958.
- [17] P. P. Maglio, R. Barrett, C. S. Campbell, and T. Selker. Suitor: an attentive information system. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 169–176, New York, NY, USA, 2000. ACM Press.
- [18] G. Miller. Wordnet: An electronic lexical database. *Communications of the ACM*, 38(11):39–41, 1995.
- [19] 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.
- [20] B. Rhodes and T. Starner. The remembrance agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*, pages 487–495, 1996.
- [21] B. J. Rhodes. The wearable remembrance agent: A system for augmented memory. *Personal Technologies*, pages 218–224, 1997.
- [22] B. J. Rhodes. Margin notes: building a contextually aware associative memory. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 219–224, New York, NY, USA, 2000. ACM Press.
- [23] B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Syst. J.*, 39(3-4):685–704, 2000.
- [24] G. Salton, editor. *Automatic text processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] S. Walker, S. Robertson, M. Boughanem, G. Jones, and K. S. Jones. Okapi at trec-6. *NIST Special Publication*, 1998.
- [29] 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.