Intelligent Methods for Information Access in Context: The Role of Topic Descriptors and Discriminators

Carlos M. Lorenzetti[†] **Rocío L. Cecchini**[‡] **Ana G. Maguitman**[†]

e-mail: {cml,rlc,agm}@cs.uns.edu.ar
[†]Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
[‡]Laboratorio de Investigación y Desarrollo en Computación Científica
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, Argentina.*

Abstract

Successful access to information sources on the Web depends on effective methods for identifying the needs of a user and making relevant information resources available when needed. This paper formulates a theoretical framework for the study of context-driven Web search and proposes new methods for learning query terms based on the user task. These methods use an incrementally-retrieved, topic-dependent selection of Web documents for term-weight reinforcement reflecting the aptness of the terms in describing and discriminating the topic of the user context. Based on this framework, we propose an incremental search algorithm for information retrieval agents that has the potential to improve significantly over the traditional IR techniques. The new algorithm learns new descriptors by searching for terms that tend to occur *often* in relevant documents, and learns good discriminators by identifying terms that tend to occur *only* in the context of the given topic. We discuss the technical challenges posed by this new framework, outline our agent system architecture, and present an evaluation of the proposed techniques.

Keywords: context modeling, information retrieval, descriptors, discriminators

1 INTRODUCTION

The World Wide Web is an ever-expanding source of information about a huge diversity of topics. As a consequence, it is becoming increasingly important to know how to find out about a topic of special interest, focusing the search on material that is relevant to the current task. This search activity could be done more effectively if intelligent agents for information access and delivery were included as part of the system search tools. In order to reduce the user cognitive overload, task-specific Web search agents need to be adapted to deliver few but highly relevant resources.

An important requirement for these agents is to provide relevant material, doing it at the right time, and without causing undue or excessive distraction. Two elements that can be exploited by an agent to enhance Web search are *user context* and *user preferences*. User context reflects the task in which the user is immersed (e.g., [8, 14]). The context may consist of an electronic document the user is editing, Web pages the user has recently visited, etc. User preferences reflect the way in which a

*Partially supported by Agencia Nacional de Promoción Científica y Tecnológica (PICT 2005, Nro. 32373), Universidad Nacional del Sur (24/ZN13) and CONICET.

user would prioritize search results. User preferences could be entered explicitly by the user or could be inferred by the agent system (e.g., by monitoring the user's behavior).

As part of our research work we are studying how to build intelligent agents that can provide context-based and preference-based support by retrieving useful information from the Web. These agents monitor the user and search the Web for material related to the user current task and preferences. A general discussion of the proposed architecture and project goals can be found in [9].

This paper presents general techniques for incrementally learning important terms in the context of a user task. Specifically, we are studying three questions: (1) can the user context be usefully exploited by information retrieval agents to access relevant material on the Web?, (2) can a set of context-specific terms be incrementally refined, based on the analysis of search results?, and (3) are the context-specific terms learned by incremental methods better query terms than those identified by classical IR techniques? To address these questions section 2 presents a theoretical framework for the study of contextual search on the Web. Section 3 describes our agent system architecture for context-based Web retrieval. Section 4 shows the results of our evaluations. Section 5 discusses related work, and finally, section 6 presents our conclusions.

2 A FRAMEWORK FOR LEARNING CONTEXT-SPECIFIC TERMS

Search interfaces provide access to a vast repository of information on the World Wide Web. However, finding relevant information remains challenging, because of the need to select useful resources from an enormous range of possibilities. For many computer-mediated tasks, the user context provides a rich set of terms that can be exploited by intelligent information agents to enhance Web search. Such agents can be equipped with special monitoring capabilities, designed to generate a model of the user task. The agents will be in charge of observing how the user interacts with different kinds of computer utilities (such as email systems, browsers and text editors) to characterize the user's information needs as a collection of weighted terms. This requires a framework for learning context-specific terms.

Classical IR weighting schemes, such as *term frequency inverse document frequency* (TF-IDF) [23], are reasonable measures of term importance but are insufficient for the task domain for our research. As has been discussed by a number of sources, issues arise when attempting to apply conventional IR schemes for measuring term importance to systems for searching Web data [13, 5]. One difficulty is that methods for Web search do not have access to a fully predefined collection of documents, raising questions about the suitability of classical IR schemes for measuring term importance when searching the Web.

A central question addressed in our work is how to learn context-specific terms based on the user current task and an open collection of incrementally retrieved Web documents. In what follows, we will assume that a user task is represented as a set of cohesive terms summarizing the topic of the user context. Consider for example a topic involving the *Java Virtual Machine*, described by the following set of terms:

java	virtual	machine	programming	language
computers	netbeans	applets	ruby	code
sun	technology	source	jvm	jdk

Context-specific terms may play different roles. For example, the term *java* is a good descriptor of the topic for a general audience. On the other hand, terms such as *jvm* and *jdk*—which stand for "Java Virtual Machine" and "Java Development Kit"—may not be good descriptors of the topic for that audience, but are effective in bringing information similar to the topic when presented in a query. Therefore, *jvm* and *jdk* are good discriminators of that topic.

In previous work [18] we have tested the following two hypotheses:

- Good topic descriptors can be found by looking for terms that occur often in documents similar to the given topic.
- Good topic discriminators can be found by looking for terms that occur only in documents similar to the given topic.

Thus a possible strategy that an information agent can follow for finding good topic descriptors is to (1) find documents that are similar to other documents already known to include that topic, and (2) select from those documents the terms that occur often. On the other hand, a term is a good discriminator for a topic if most documents that contain that term are topically related. Therefore, finding good topic discriminators requires finding terms that tend to occur only in the context of the given topic. Both topic descriptors and discriminators are important as query terms. Because topic descriptors occur often in relevant pages, using them as query terms may improve recall. Similarly, good topic discriminators occur primarily in relevant pages, and therefore using them as query terms may improve precision.

As a first approximation to compute descriptive and discriminating power, we begin with a collection of m documents and n terms. As a starting point we build an $m \times n$ matrix $\mathbf{H}$, such that $\mathbf{H}[i,j] = k$, where k is the number of occurrences of term t_j in document d_i. In particular we can assume that one of the documents (e.g., d_0) corresponds to the initial user context. The following example illustrates this situation:

$$\mathbf{H} = \begin{array}{r} \\ \texttt{java} \\ \texttt{machine} \\ \texttt{virtual} \\ \texttt{language} \\ \texttt{programming} \\ \texttt{coffee} \\ \texttt{island} \\ \texttt{province} \\ \texttt{jvm} \\ \texttt{jdk} \end{array} \begin{array}{c} \begin{array}{ccccc} d_0 & d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{pmatrix} 4 & 2 & 5 & 5 & 2 \\ 2 & 6 & 3 & 2 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 1 & 1 \\ 3 & 0 & 2 & 2 & 0 \\ 0 & 3 & 0 & 0 & 3 \\ 0 & 4 & 0 & 0 & 2 \\ 0 & 4 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 3 & 3 & 0 \end{pmatrix} \end{array}$$

Documents:
d_0: user context
d_1: espressotec.com
d_2: netbeans.org
d_3: sun.com
d_4: wikitravel.org

The matrix $\mathbf{H}$ allows us to formalize the notions of good descriptors and good discriminators. We define *descriptive power of a term in a document* as a function $\lambda : \{d_0, \ldots, d_{m-1}\} \times \{t_0, \ldots, t_{n-1}\} \rightarrow [0,1]$:

$$\lambda(d_i, t_j) = \frac{\mathbf{H}[i,j]}{\sqrt{\sum_{k=0}^{n-1} (\mathbf{H}[i,k])^2}}.$$

If we adopt $\mathrm{s}(k) = 1$ whenever $k > 0$ and $\mathrm{s}(k) = 0$ otherwise, we can define the *discriminating power of a term in a document* as a function $\delta : \{t_0, \ldots, t_{n-1}\} \times \{d_0, \ldots, d_{m-1}\} \rightarrow [0,1]$:

$$\delta(t_i, d_j) = \frac{\mathrm{s}(\mathbf{H}[j,i])}{\sqrt{\sum_{k=0}^{m-1} \mathrm{s}(\mathbf{H}[k,i])}}.$$

Note that λ and δ satisfy the conditions

$$\sum_j (\lambda(d_i, t_j))^2 = 1 \quad \text{and} \quad \sum_j (\delta(t_i, d_j))^2 = 1.$$

Given a term t_i in a document d_j, the term t_i will have a high descriptive power in d_j if it occurs often in d_j, while it will have a high discriminating power if it tends to occur only in d_j (i.e., it seldom

occurs in other documents). The descriptive power and discriminating power values for the terms in the example given above are as follows:

$$\begin{array}{r} \\ \texttt{java} \\ \texttt{machine} \\ \texttt{virtual} \\ \texttt{language} \\ \texttt{programming} \\ \texttt{coffee} \\ \texttt{island} \\ \texttt{province} \\ \texttt{jvm} \\ \texttt{jdk} \end{array} \begin{array}{c} \lambda(d_0,t_j)^T \\ \left(\begin{array}{c} 0.718 \\ 0.359 \\ 0.180 \\ 0.180 \\ 0.539 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{array}\right. \end{array} \begin{array}{c} \delta(t_i,d_0) \\ \left.\begin{array}{c} 0.447 \\ 0.500 \\ 0.577 \\ 0.500 \\ 0.577 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{array}\right) \end{array}$$

The above weights reflect some of the limitations of this first approach. For instance, the weights associated with the terms *jvm* and *jdk* do not reflect their importance as discriminators of the topic under analysis. In the same way as the well-known TF and IDF measures [23], the functions λ and δ allow to discover terms that are good descriptors and good discriminators of a document, as opposed to good descriptors and good discriminators of the *topic* of a document.

Our current goal is to formulate notions of topic descriptors and discriminators suitable for the Web scenario. Rather than extracting descriptors and discriminators directly from the user context, we want to extract them from *the topic* of the user context. This requires an incremental method to characterize the topic of the user context, which is done by identifying documents that are similar to the user current context. Assume the user context and the retrieved Web documents are represented as document vectors in term space. To determine how similar two documents d_i and d_j are we adopt the IR cosine similarity [2]. This measure is defined as follows:

$$\sigma(d_i,d_j) = \sum_{k=0}^{n-1}[\lambda(d_i,t_k)\cdot\lambda(d_j,t_k)].$$

The similarity values between the user context (d_0) and the other documents in our example are as follows:

$$\sigma(d_0,d_j) = \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ \big(0.399 & 0.840 & 0.857 & 0.371\big) \end{array}$$

The notion of topic descriptors was informally defined earlier "as terms that occur *often* in the context of a topic." We define the *term descriptive power in the topic of a document* as a function $\Lambda : \{d_0,\ldots,d_{m-1}\} \times \{t_0,\ldots,t_{n-1}\} \rightarrow [0,1]$. If $\sum_{\substack{k=0\\k\neq i}}^{m-1}\sigma(d_i,d_k) = 0$ then we set $\Lambda(d_i,t_j) = 0$. Otherwise we define $\Lambda(d_i,t_j)$ as follows:

$$\Lambda(d_i,t_j) = \frac{\sum_{\substack{k=0\\k\neq i}}^{m-1}[\sigma(d_i,d_k)\cdot[\lambda(d_k,t_j)]^2]}{\sum_{\substack{k=0\\k\neq i}}^{m-1}\sigma(d_i,d_k)}$$

Thus, the descriptive power of a term t_j in the topic of a document d_i is a measure of the quality of t_j as a descriptor of documents similar to d_i. As we informally formulated earlier, a term is a good discriminator of a topic if it "tends to occur *only* in documents associated with that topic." We define

the *discriminating power of a term in the topic of a document* as a function $\Delta : \{t_0, \ldots, t_{n-1}\} \times \{d_0, \ldots, d_{m-1}\} \rightarrow [0, 1]$ calculated as follows:

$$\Delta(t_i, d_j) = \sum_{\substack{k=0 \\ k \neq j}}^{m-1} [[\delta(t_i, d_k)]^2 \cdot \sigma(d_k, d_j)].$$

Thus the discriminating power of term t_i in the topic of document d_j is an average of the similarity of d_j to other documents discriminated by t_i. The following are the topic descriptive and discriminating power for the terms in our example:

	$\Lambda(d_0, t_j)^T$	$\Delta(t_i, d_0)$
java	0.385	0.493
machine	0.158	0.524
virtual	0.014	0.566
language	0.040	0.517
programming	0.055	0.566
coffee	0.089	0.385
island	0.064	0.385
province	0.040	0.385
jvm	0.032	0.848
jdk	0.124	0.848

Guided by the notions of topic descriptors and discriminators, it is possible to learn novel context-specific terms and reinforce the weights of existing ones. This results in a better representation of the user search context, facilitating query refinement and context-based filtering. The next section shows how the proposed approach is applied in the design of an agent system for context-based Web retrieval.

3 AGENT SYSTEM ARCHITECTURE FOR CONTEXT-BASED WEB RETRIEVAL

An incremental approach to identify context-specific terms allows to go beyond the known user desires, to automatically generate a richer context representation through the use of topic descriptors and discriminators, and find what might be useful for the user. This kind of incremental mechanism can reveal similarities that were not previously apparent and present a "big picture" that can give the user a broader understanding of the current task.

Direct manipulation search interfaces provide fast access to information available on the Web, taking advantage of information indexed by major search engines (e.g., Google) or other searchable databases (PubMed, Amazon, etc.). However, the direct manipulation approach has a number of limitations including a large search space, actions in response to immediate user interaction only, and inability to learn from repetitive actions. An information agent-oriented approach can overcome some of these limitations, providing search and filtering capabilities, proactivity, task orientation and adaptivity [7].

The major challenge that an information agent that operates on top of a search interface needs to address is the generation of suitable queries. Our methods focus on how to incrementally generate queries based on context. Because search engines restrict queries to a small number of terms (e.g., the 32-term limit for Google) a single query cannot reflect extensive contextual information. In an incremental method, the first query terms generated for guiding a Web search may not provide the definitive results. However, comparing the set of search results to the user context can help to automatically refine subsequent queries.

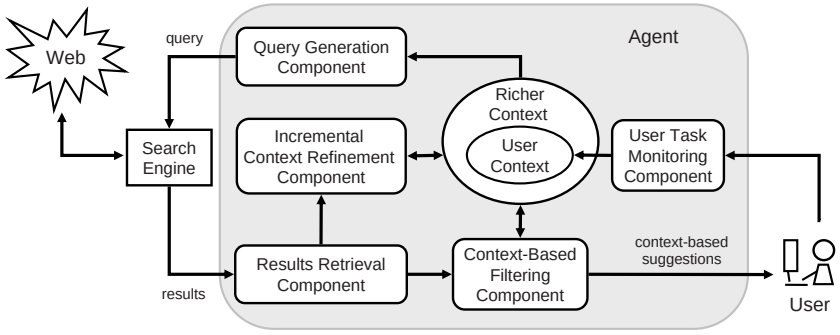

Figure 1: Agent system architecture for incremental context-based Web retrieval

Figure 1 depicts our agent system architecture for incremental context-based Web retrieval. In our prototype system, the agent handles partially observable environments such as the Web (through a search interface) and the user information needs (by observing the user behavior). It will maintain an internal state containing a representation of the user context, which is incrementally refined. The components that enable the agent to generate context-based suggestions are the following:

- **User Task Monitoring Component**. This component is in charge of observing how the user interacts with different kinds of computer utilities such as email systems, browsers and text editors, and generates a representation of the user context.
- **Query Generation Component**. This component selects terms from the user context and forms suitable queries, which are submitted to a standard search engine (e.g, Google) or entered into individual search forms (e.g., Amazon or PubMed). Initial queries will be entered by the user, or automatically formed with terms that occur frequently in the user context. Subsequent queries are refined as topic descriptors and discriminators are identified by the incremental search method.
- **Results Retrieval Component**. This component is in charge of retrieving the search results generated by the search interface, so that they can be locally analyzed.
- **Context-Based Filtering Component**. This component is in charge of estimating the relevancy of the documents collected by the results retrieval component. This is done by computing the similarity between the collected material and the user context. Both relevant and irrelevant material will be kept for use by the incremental context-refinement module (described below). However, only the material that is relevant to the current context will be presented as suggestions to the user.
- **Incremental Context-Refinement Component**. This component uses the content returned by the results retrieval component in combination with the relevancy information provided by the context-based filtering module to incrementally refine the context representation. It does so by adjusting the weights of the context-specific terms according to their descriptive and discriminating power.

4 EVALUATION

Because the goal of an information agent is to present useful suggestions, the ideal method for evaluating result quality would be an end-to-end user study, in which subjects directly assess the usefulness

of agent suggestions. However, to guide the bottom-up development of the methods, it is crucial to be able to assess incremental steps for which human-subjects evaluations would be impractical.

In previous work we showed that topic descriptors can help achieve good recall, while topic discriminators improve precision [18, 17]. Here, we present new empirical evidence that points out to the usefulness of combining topic descriptors and discriminators so that an intelligent agent performing context-based Web Search can maximize the contribution of both to form suitable queries.

In our evaluations we selected 15 pages from the ODP[1] directory (http://dmoz.org) to use as a base text representing the user context. These pages were obtained from the topics *Recreation*, *Society* and *Business* (we used 5 pages for each of the 3 topics). For each of the selected pages we generated a TF vector representation C. Then we applied an Intelligent Incremental Method (IIM) to generate queries. A schematic illustration of the IIM method is shown in figure 2 and summarized in the following steps:

1. Generate *k* queries using *l* terms in C, Queries[1..k], randomly;
2. i $\leftarrow$ 0;
3. For each *Query* in Queries, send it to a search engine;
4. Obtain the answers and convert the results to a vector representation;
5. Generate a sorted list L_Λ of topic descriptors;
6. Generate a sorted list L_Δ of topic discriminators;
7. Update L_Λ and L_Δ incrementally with some learning rate α;
8. Generate *k* queries using some combination of *m* terms from L_Λ and *n* terms from L_Δ, using a roulette selection method;
9. For each *Query* in Queries, send it to the search engine;
10. Obtain the answers and convert the results to a vector representation;
11. For each query result, test if it is a "good query";
12. For each bad query, reformulate it, obtain their results and convert them to a vector representation;
13. i $\leftarrow$ i + 1;
14. go to 5

In the IIM method, a good query is one that complies with at least one of the following conditions, depending on the case under analysis:

1. It retrieves at least one result whose similarity with C is higher than the highest similarity obtained in the previous iteration.
2. The average similarity of its' results is higher than the average similarity of the results from the previous iteration.

We have used Google Web API to collect search results and only the "snippets" returned by Google are used by our methods. The snippet is a text excerpt from the page summarizing the context in which the search terms occur. The input parameter l in the above algorithm determines the initial query size. The parameters m and n specify the number of descriptors and discriminators, respectively, used to form each of the subsequent queries. Note that this method generalizes those in which only descriptors or only discriminators are used (since either m or n could be set to zero). In our tests we tried different settings for the parameters l, m and n. The results reported here are restricted to the cases in which $l = 6$ and $m + n = 6$.

[1] Open Directory Project.

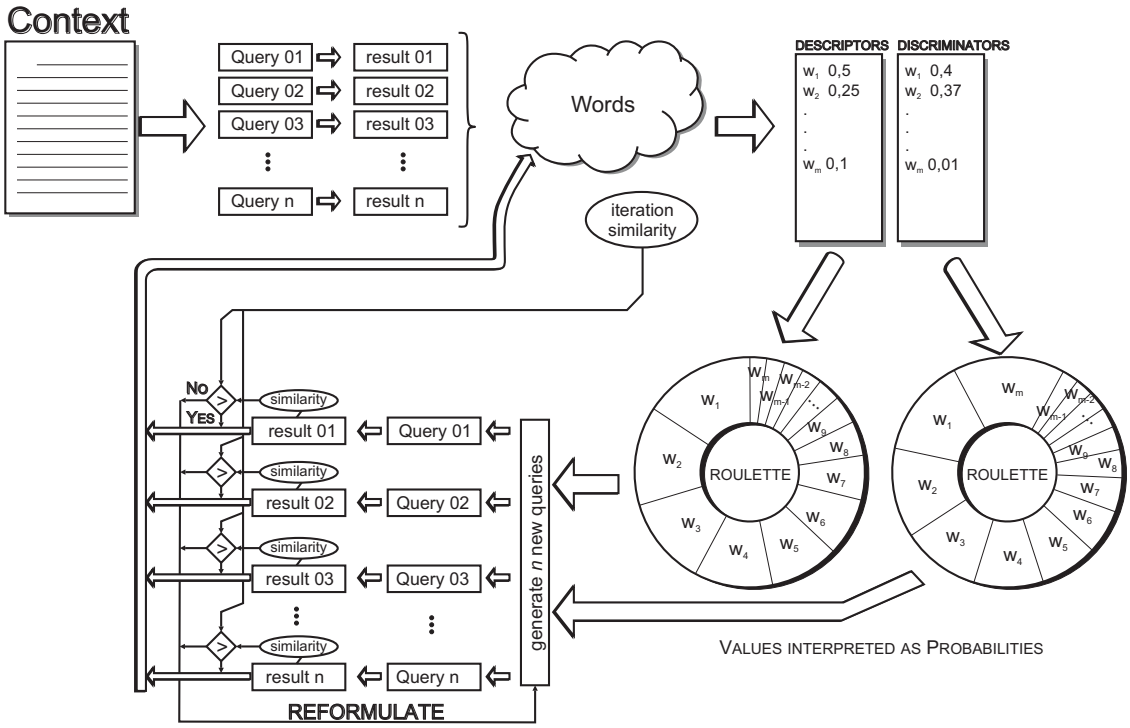

Figure 2: A schematic illustration of the IIM method.

As a baseline for comparison we used a Naïve Method (NM) that extends the baseline mechanism presented in [16] by supporting multiple simultaneous queries. In the same way as the IIM method, the NM method generates the initial queries randomly, and for the next iterations it generates the queries using a roulette-wheel method. We decided to use the roulette-wheel method due to its' simplicity, allowing us to favor the selection of the most promising terms. To set the probability of each term we used the term's weight as a descriptor or as a discriminator, depending on the case.

In our evaluation we take the minimum, average, and maximum similarity between the context (base document) and the retrieved results (snippets) as indicators of the methods' performance. The similarity measure is computed as the traditional cosine similarity in term space [2], except that the terms that occur in the query are disregarded. This prevents biasing the results in favor of those that select query terms directly from the base document.

Figure 3 shows the performance of the IIM and NM methods based on the minimum and average similarity between the search results and the initial context as an estimation of query quality (for the maximum similarity, the highest values achieved were approximately 0.6, starting from the initial iterations, and are not shown in the figure).

In our evaluations we observe that (1) NM always found the best results during the few first iterations, (2) IIM shows statistically significant improvement over the NM method, (3) IIM shows statistically significant improvements when the last iterations are compared to the first one, (4) IIM reduces its variance values at each iteration.

We performed additional evaluations, using different settings for the number of descriptors and discriminators in each query. Table 1 shows the performance based on average similarity for the three tested combinations of descriptors and discriminators (values in bold denote the maximum in that iteration):

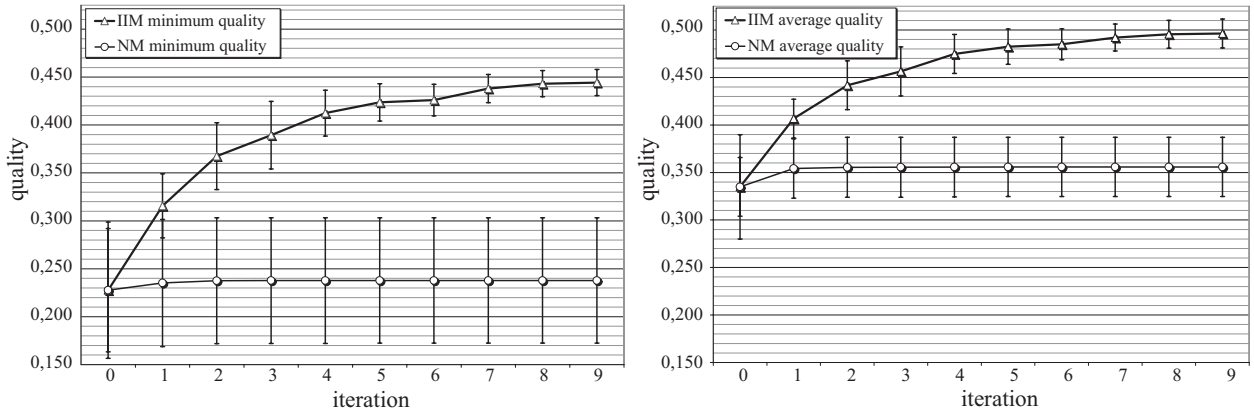

Figure 3: Performance Evaluation based on minimum similarity (left) and average similarity (right)

$\Lambda - \Delta$	(2–4)	(4–2)	(5–1)
iteration 0	0,328	**0,367**	0,335
iteration 1	0,351	**0,432**	0,407
iteration 2	0,367	0,441	**0,442**
iteration 3	0,374	0,449	**0,456**
iteration 4	0,388	0,454	**0,475**
iteration 5	0,400	0,467	**0,482**
iteration 6	0,408	0,479	**0,485**
iteration 7	0,413	0,482	**0,492**
iteration 8	0,415	0,488	**0,495**
iteration 9	0,419	0,495	**0,496**

Table 1: An analysis of the quality obtained using different combinations of descriptors and discriminators.

As stated in step 7 of our algorithm, the descriptor and discriminator weights were incrementally updated at each iteration based on some learning rate α as follows:

$$TermWeight(t,i) = TermWeight(t,i-1) * (1-\alpha) + NewTermWeight(t) * \alpha \quad (1)$$

We performed a number of tests to adjust the parameter α (not shown here for space reasons). For the cases analyzed, we found out that $\alpha = 0.4$ results in the best performance.

Another parameter tested is the number of queries generated during each iteration. We decided to use 20 queries for each generation but other settings could be tested as well.

5 RELATED WORK

Several existing systems exploit user interaction with computer applications to determine the user's current task and contextualize information needs. WebWatcher [1] is an early attempt to assist users locating information on the Web by highlighting hyperlinks in a page based on the declared preferences and browsing history of a user as well as information gathered from other users with similar interests. Personal WebWatcher [19] is a successor of WebWatcher that learns individual users' interests by observing their browsing behavior. Letizia [15] is a user interface agent that unobtrusively assists Web browsing. As the user navigates Web pages, Letizia performs a breadth-first search augmented by several heuristics to anticipate what items may be of interest to the user. Syskill & Webert

[20] uses information retrieval techniques to process the content of a page rated by a user, and machine learning to acquire a model, that is utilized to predict which links on a Web page a user will find useful. SenseMaker [4] is an interface that facilitates the navigation of information spaces by providing task specific support for consulting heterogeneous search services. The system helps users to examine their present context, move to new contexts or return to previous ones. Fab [3] is a hybrid content-based, collaborative Web page recommender system that learns to browse the Web on behalf of a user. Fab generates recommendations by the use of a set of collection agents (that find pages for a particular topic) and selection agents (that find pages for a particular user). Users' explicit ratings of the recommended pages combined with several heuristics are used to update personal-agents' profiles, remove unsuccessful agents, and duplicate successful ones. Broadway [12] is a case-based reasoning system that monitors a user's browsing activity and provides advice by reusing navigational cases extracted from past browsing experiences of a group of users. Another Web navigation assistant is SiteSeer [22], which recommends pages collaboratively by looking at users' bookmarks. The Watson system [8] is a context-aware search tool that attempts to find relevant online resources. Watson is part of a family of programs known as *Information Management Assistants* (IMAs) developed at the InfoLab of Northwestern University (Chicago, USA). The purpose of the IMAs is to anticipate the user's needs and to provide proactive and on demand support for the user's current activity. All these systems are similar to our proposal in attempting to provide users with context-relevant information, but differ in not attempting to learn context-specific terms by performing incremental search.

Extensions to basic IR approaches have examined some of the issues raised in this paper. For instance, some automatic relevance feedback techniques, such as the Rocchio's method [21], make use of the full search context for query refinement. In these approaches the original query is expanded by adding a weighted sum of terms corresponding to relevant documents, and subtracting a weighted sum of terms from irrelevant documents. As a consequence the terms that occur often in documents similar to the input topic will be assigned the highest rank, as in our descriptors. However, our technique also gives priority to terms that *occur only in relevant documents* and not just to those that *occur often*. In other words, we prioritize terms for both discriminating and descriptive power. The techniques for query term selection proposed in this paper share insights and motivations with other methods for query expansion and refinement [24, 6]. However, systems applying these methods differ from our framework in that they support this process through a query or browsing interface requiring explicit user intervention, rather than formulating queries automatically.

Our techniques rely on the notions of document similarity to discover higher-order relationships in collections of documents. This relates to the use of LSA [11] to uncover the latent relationships between words in a collection. However, LSA's goal is to compute a matrix representing semantic distance between terms and documents, without identifying topic descriptors and discriminators.

6 CONCLUSIONS

In this paper we have presented a novel approach for learning context-specific terms on the Web. Based on this approach, an intelligent agent can take advantage of the information available in the user context to perform incremental Web search. We have shown that the user context can be usefully exploited to access relevant material. However, those terms that occur more frequently in the user context are not necessarily the most useful ones. In light of this we proposed an incremental method for context refinement based on the analysis of search results. We have also proposed to distinguish two natural notions, namely topic descriptors and topic discriminators.

Our evaluations show the effectiveness of incremental methods for query generation and refinement. We are currently working on integrating the proposed method with qualitative approaches such

as the ones discussed in [9, 10] for ranking results based on user preferences. We are also analyzing different strategies for helping the system keep its focus on the initial context after several incremental steps have taken place. As part of our future work we expect to use standard large-scale collections (such as the TREC Web collection) to further evaluate our techniques.

Information agents have become essential components of today's Internet infrastructure. In light of the difficulties in accessing Web resources through traditional IR techniques, it is important to propose methods that allow the dynamic identification of useful context-specific material. We hope the methods proposed in this work provide new insights for further studies into this area.

REFERENCES

[1] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A learning apprentice for the World Wide Web. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[4] Michelle Q. Wang Baldonado and Terry Winograd. SenseMaker: an information-exploration interface supporting the contextual evolution of a user's interests. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18. ACM Press, 1997.

[5] Nicolas J. Belkin. Helping people find what they don't know. *Commun. ACM*, 43(8):58–61, 2000.

[6] Bodo Billerbeck, Falk Scholer, Hugh E. Williams, and Justin Zobel. Query expansion using associated queries. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 2–9. ACM Press, 2003.

[7] Jeffrey M. Bradshaw. An introduction to software agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, pages 3–46. AAAI Press / The MIT Press, 1997.

[8] Jay Budzik, Kristian J. Hammond, and Larry Birnbaum. Information access in context. *Knowledge based systems*, 14(1–2):37–53, 2001.

[9] Carlos Chesñevar, Carlos Lorenzetti, Ana Maguitman, Fernando Sagui, and Guillermo Simari. Exploiting user context and preferences for intelligent web search. In *Proceedings of the Workshop de Investigadores en Ciencias de la Computación (WICC 2006)*, May 2006.

[10] Carlos Chesñevar and Ana Maguitman. ArgueNet: An argument-based recommender system for solving Web search queries. In *Proceedings of the International IEEE Conference on Intelligent Systems (IS 2004)*, pages 282–287. IEEE, June 2004.

[11] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[12] Michel Jaczynski and Brigitte Trousse. BROADWAY: A World Wide Web browsing advisor reusing past navigations from a group of users. In *Proceedings of the Third UK Case-Based Reasoning Workshop (UKCBR3)*, Manchester, UK, September 1997.

[13] Mei Kobayashi and Koichi Takeda. Information retrieval on the Web. *ACM Comput. Surv.*, 32(2):144–173, 2000.

[14] David B. Leake, Travis Bauer, Ana Maguitman, and David C. Wilson. Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *Proceedings of the AAAI-00 Workshop on Intelligent Lessons Learned Systems. Austin, Texas*, pages 33–37. AAAI Press, 2000.

[15] Henry Lieberman. Letizia: An agent that assists Web browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. IJCAI-95*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[16] Carlos Lorenzetti, Fernando Sagui, Ana Maguitman, Carlos Chesñevar and Guillermo Simari. Incremental Methods for Context-Based Web Retrieval. In *Proceedings of the Congreso Argentino de Ciencias de la Computación (CACIC 2006)*, October 2006.

[17] Ana Maguitman, David Leake, and Thomas Reichherzer. Suggesting novel but related topics: towards context-based support for knowledge model extension. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 207–214, New York, NY, USA, 2005. ACM Press.

[18] Ana Maguitman, David Leake, Thomas Reichherzer, and Filippo Menczer. Dynamic extraction of topic descriptors and discriminators: Towards automatic context-based topic search. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM)*, Washington, DC, November 2004. ACM Press.

[19] D. Mladenic. Personal WebWatcher: Design and implementation. Technical report ijs-dp-7472, School of Computer Science, Carnegie-Mellon University, Pittsburgh, USA, 1996.

[20] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying interesting Web sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

[21] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[22] James Rucker and Marcos J. Polanco. Siteseer: personalized navigation for the Web. *Communications of the ACM*, 40(3):73–76, 1997.

[23] G. Salton and C. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29:351–372, 1973.

[24] Falk Scholer and Hugh E. Williams. Query association for effective retrieval. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 324–331. ACM Press, 2002.