Representing User Information Context with Ontologies

Ahu Sieg, Bamshad Mobasher, Robin Burke, Ganesh Prabu, Steve Lytinen

School of Computer Science, Telecommunication and Information Systems
DePaul University
243 South Wabash Avenue, Chicago, Illinois, 60604, USA
{asieg, mobasher, rburke, gprabu, lytinen}@cs.depaul.edu

Abstract

One of the key factors for accurate and effective information access is the user context. The critical elements that make up a user's information context include the semantic knowledge about the domain being investigated, the short-term information need as might be expressed in a query, and the user profiles that reveal long-term interests. In this paper, we propose a framework for contextualized information access that seamlessly combines these elements in order to effectively locate and provide the most appropriate result for users' information needs. In particular, we focus on integrating a user's query with semantic knowledge from an existing concept hierarchy to assist the user in information retrieval. In our framework, the user's "context" is captured via nodes in a concept lattice induced from the original ontology and is updated incrementally based on user's interactions with the concepts in the ontology. Our experimental results show that utilizing the user context improves the effectiveness of the search queries, especially in the typical case of Web users who tend to use very short queries.

1 Introduction

The scope and rapid expansion of the universe of Internet-accessible information and the penetration of the World Wide Web into all areas of human activity have made the problem of information access one of universal importance. Tasks from scientific discovery to national security all require extracting useful needles from information haystacks. One of the key factors for accurate information access is user context: a system that does not know who is asking for information and for what purpose will never be able to provide more than very general answers.

Despite their popularity, users' interactions with Web search engines can be characterized as one size fits all (Allan et al., 2003). The representation of user preferences, search context, or the task context is non-existent. Indeed, contextual retrieval has been identified as a long-term challenge in information retrieval. Allan et al. (Allan et al., 2003) define it as follows: *Contextual retrieval: Combine search technologies and knowledge about query and user context into a single framework in order to provide the most appropriate answer for a user's information needs*.

Though the above definition focuses on information retrieval, the notion of "user's information context" is also critical for a variety of other information access modalities, including browsing and filtering. In general, the critical elements that make up a user's information context include the semantic knowledge about the domain being investigated, the short-term information need as might be expressed in a query, and the user profiles that reveal long-term interests.

Various forms of query enhancement have long been a research topic in information retrieval, from relevance feedback (Buckley et al., 1994) to incremental query refinement (Allan, 1996; Eguchi, 2000) to query expansion based on lexical variants such as synonyms (Miller, 1997). Synonym-based query expansion could be considered a primitive form of the application of domain knowledge. However, without a mechanism to disambiguate the sense of the user's query, mere synonym expansion does not tend to improve precision. A number of researchers have explored intelligent Web agents that learn about user's interests in Web-based information access (Boley et al., 1999; Joachims et al., 1997; Lieberman, 1997). All of this work depends on elicitation of user interests and preference. To lessen the requirement for user interaction, lately the attention of the research community has been drawn to

explore how various implicit measures of user interests can be used in information retrieval and filtering applications (Dumais et al., 2003; Kelly & Teevan, 2003). In recent years, researchers have designed systems which can provide users with relevant resources in the context of work they are performing (Budzik et al. 2002). These approaches, while taking into account the user behavior, do not consider the domain knowledge that can be used as an additional source for disambiguating the context.

In this paper we develop a framework for contextualized information access that seamlessly integrates the essential elements of user's information context. In particular, we propose a unified model for representing a user's context which takes into account the user's short-term and long-term profiles, as well as relevant concepts from a pre-existing ontology.

We envision a system where the user interacts with a domain-specific ontology, represented as a concept hierarchy. The domain knowledge inherent in the ontology is utilized to disambiguate the user context. This "context" is represented as an extension of the concept hierarchy and is maintained and updated incrementally based on user's interactions with concepts in the ontology. The user's browsing behavior and other implicit measures of user interest are observed over time for user profiling. Once the user's information context is derived based on these sources of evidence, the underlying context representation can be translated into a form that can be utilized for different information access activities. We provide an example of how the derived context can be used to enhance a user's initial search query in information retrieval.

2 Representing and Maintaining User Context

Semantic knowledge is an essential part of the user context. When disambiguating the context, the domain knowledge inherent in the ontology is called upon as a source of key domain concepts. Thus, the central notion in the user context is a domain-specific ontology and the user's interaction with this ontology.

2.1 User Context

We use a domain ontology as the fundamental source of semantic knowledge in our framework. The ontology is a set of concepts with "is-a" relationships between them. Thus, the ontology is organized as a concept taxonomy. In our current implementation, we use the Yahoo concept hierarchy. The ontology is maintained in an aggregate form.

In our framework, we use a term-vector based representation for the concepts. A number of approaches, including various methods for ontology learning, can be utilized to collect the term vectors to represent these concepts. We use the sample documents indexed under the concepts in the Yahoo hierarchy. Each concept in the hierarchy is associated with a set of related Web pages. To generate the term vector representation for a concept, the content of the pages associated with the concept are combined to yield a single term vector.

To create the aggregate representation of our ontology we compute a weighted term vector $\vec{n}_i$ for each concept *i* in the Yahoo concept hierarchy. In Yahoo, each concept contains a collection of documents D_i, and a set of subconcepts S_i. To compute $\vec{n}_i$, first we compute a term vector $\vec{n}_d$ for each document $d \in D_i$. Each vector $\vec{n}_d$ is computed using standard information retrieval techniques to obtain term weights (Salton & McGill, 1983). Then, $\vec{n}_i$ is computed as $\vec{n}_i = \sum_{d \in D_i} \vec{n}_d + \sum_{s \in S_i} \vec{n}_s$ where each $\vec{n}_s$ is the aggregate term vector for each subconcept $s \in S_i$, computed recursively. Thus, $\vec{n}_i$ is an aggregation of the documents indexed under concept *i* along with the subconcepts of *i*.

As an example, Figure 1 displays a portion of the Yahoo hierarchy corresponding to the concept *Languages* and the partial term vector for this concept. The terms which are included in this term vector essentially provide context by integrating the domain knowledge from all descendants of *Programming Languages*. Notice that these terms are not simple lexical variants such as synonyms; rather they provide semantic information about a specific concept in the ontology. The term vector which represents the concept *Languages* is computed from a combination of the

documents indexed under this concept as well as the term vectors representing its subconcepts such as *C#*, *Python*, and *Java*.

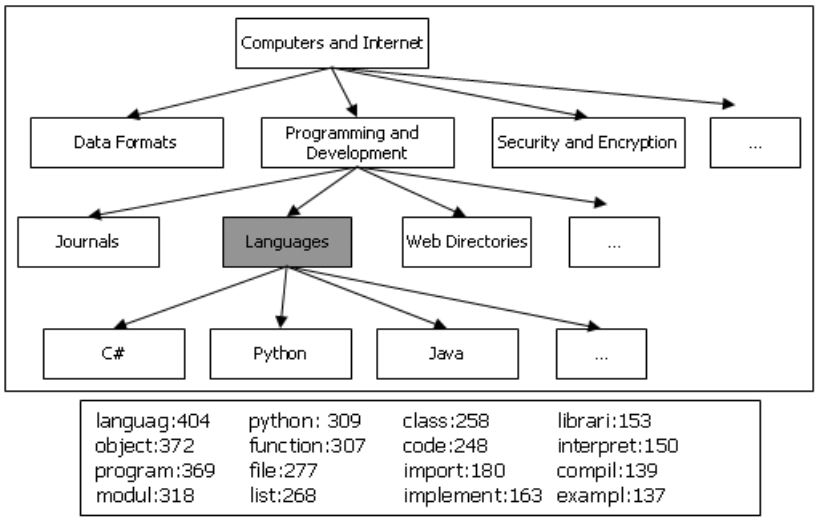

Figure 1: Partial Term Vector for a Sample Concept in the Yahoo Hierarchy

This aggregation method provides a natural partial order among concepts in the ontology, each represented by a term vector as described above: Let $\vec{n}_1 = \left\langle w_1^1, w_2^1, w_3^1 ..., w_k^1 \right\rangle$ and $\vec{n}_2 = \left\langle w_1^2, w_2^2, w_3^2 ..., w_k^2 \right\rangle$ be two nodes in the ontology. Then, $\vec{n}_1 \le \vec{n}_2$ if and only if $\forall_j w_j^i \le w_j^2$, where w_j^i is the weight of a term *j* in the term vector for $\vec{n}_i$. It should be clear from this definition that if a concept s' in the hierarchy is a descendant of s, then $\vec{n}_{s'} \le \vec{n}_s$.

Our system presents the relevant portions of the ontology (e.g., the Yahoo concept hierarchy) to the user. The relevant nodes in the ontology are selected automatically based on an initial user query or, alternatively, based on a stored profile for the user. The user interacts with the ontology by *selecting* concepts (nodes) that are relevant to their information needs and by *deselecting* concepts that are not relevant. In this case, selection of concepts is treated as positive evidence whereas deselection is treated as negative evidence. Our goal is to represent the user context by encapsulating the positive evidence and the negative evidence from the user's session.

Since the elements in the ontology are represented as term vectors, the selection and deselection of these elements must be represented using vector operations. Two natural operations that can effectively model the selection and deselection of concepts are the vector *min* and *max* operations, defined respectively as: $\min(\vec{n}_1, \vec{n}_2) = \left\langle \min(w_1^1, w_1^2), ..., \min(w_k^1, w_k^2) \right\rangle$ and $\max(\vec{n}_1, \vec{n}_2) = \left\langle \max(w_1^1, w_1^2), ..., \max(w_k^1, w_k^2) \right\rangle$. The *min* operation is essentially the extension of set intersection operation to vectors and is used to create an aggregate representation of the selected concepts. On the other hand, the *max* operation extends the set union operation to vectors and, in our model, is used to represent the collection of deselected concepts.

Intuitively, The *min* operation effectively isolates the knowledge that is common between two concepts. On the other hand, the *max* operation serves to combine the unique knowledge associated with two concepts. In our framework, selection of multiple concepts by the user is interpreted as a conjunction of concepts. Therefore, the *min* operation is an appropriate way of representing this conjunction. On the other hand, when multiple concepts are deselected, this is inferred as expanding the set of concepts to be excluded from consideration. Thus, the *max* operation is utilized to combine the deselected concepts.

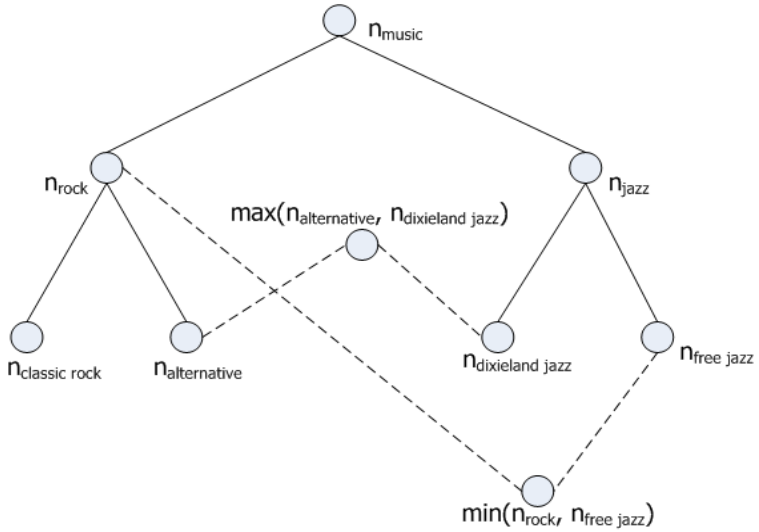

Figure 2: An Example on the Selection and Deselection of Nodes

Figure 2, displays a portion of an ontology corresponding to the node *Music*. The user has selected *Rock* and also *Free Jazz*. The user has deselected *Alternative* and *Dixieland Jazz*. The resulting term vector, which represents the positive evidence, includes only those terms that appear in both the term vector for *Rock* and the term vector for *Free Jazz*. The term vector for the negative evidence contains all terms related to *Alternative* and all terms related to *Dixieland Jazz*.

Thus, the user context is represented as a pair of elements: $c_i = \langle \vec{P}, \vec{N} \rangle$ where $\vec{P}$ is a term vector for an element which represents positive evidence and $\vec{N}$ is a term vector for an element which represents negative evidence. When multiple concept nodes are selected, the positive evidence is aggregated using the *min* operation: $\vec{P} = \min(\vec{n}_1, \vec{n}_2)$, where $\vec{n}_i$ represents the term vectors for user-selected concept nodes in the ontology. On the other hand, the negative evidence, based on multiple deselected concept nodes, is combined via the *max* operation: $\vec{N} = \max(\vec{n}_1, \vec{n}_2)$.

These operations together with the natural coordinatewise partial ordering on vectors allow us to induce a *concept lattice* whose elements represent various combinations of concepts in the original concept hierarchy. The *min* and *max* operations defined above represent the *meet* (Greatest Lower Bound) and the *join* (Least Upper Bound) operations in the induced lattice, respectively. Conceptually we can view the user context as pairs of elements in this lattice: $c_i = \langle \vec{P}, \vec{N} \rangle$. In particular, we use lattice operations to derive the user context by combining different sources of evidence. The positive evidence is aggregated using the *meet* operator of the induced lattice: $\vec{P} = \vec{n}_1 \wedge \vec{n}_2 \wedge ... \wedge \vec{n}_k$. On the other hand, the negative evidence is combined via the *join* operator of the induced lattice: $\vec{N} = \vec{n}_1 \vee \vec{n}_2 \vee ... \vee \vec{n}_k$.

It should be noted that the nodes that are involved in selection and deselection may or may not be in the original ontology (e.g., the Yahoo concept hierarchy). In essence, the concept lattice can be viewed as an extension of the original ontology. Elements of this lattice represent, possibly new, concepts that are formed as a result of selections and deselections made by the user during their interactions with the ontology. Using lattice operations, we can also maintain and update the user's information context in the long term.

2.2 Maintaining User Profiles

The profiling component of our framework attempts to learn a model of user behavior through the passive observation of user's information seeking activity including browsing and searching. Implicit measures of user

interest such as dwell time, click through, and user activities like annotation, printing, and purchasing can also be used to develop predictive models for a variety of purposes (Dumais et al., 2003).

Each time the user interacts with the ontology as might be in one information seeking session, the user's immediate short term interest is represented as a context, c_i, which is a pair of term vectors, as discussed above, representing positive and negative evidence. In order to represent the user's long-term context, we construct a user profile as a set of contexts: $pr = \{c_0, c_1, c_2, ..., c_n\}$. Based on user behavior, a specific context in the user profile can be updated or a new context can be added. The user profiles are utilized to provide the user with a domain ontology that is more consistent with their view of the world.

The important question is to determine at what point to update an existing context and when to add a new context to a user profile. In our framework, combining positive evidence relates to isolation of terms in contexts. In general, it is not beneficial to always update the same context based on the user's behavior. Since we use the *min* operation to combine positive evidence, if the same context is always updated, more terms would be eliminated from the term vector with each update. As a result, a user profile will end up with a context that contains no terms for positive evidence. The outcome of combining negative evidence using the *max* operation is an expanded set of terms in the resulting term vector. Thus, if we always update the same context, more terms would be added to the term vector for negative evidence with each update. Eventually, the negative evidence in the context will end up with a term vector that contains every single possible term. To avoid this problem, we must carefully determine whether to update an existing context or to add a new one.

One approach is to use a similarity measure to compare the pair of term vectors that make up the user's short-term context with the term vectors for each context in the long-term user profile. The term vector that represents positive evidence in the user's short-term interest can be compared to the term vectors that represent positive evidence in the user profile contexts. Similarly, the term vector that represents negative evidence can be compared with the term vectors that represent negative evidence in the contexts that make up the user profile. If the similarity between two term vectors exceeds a certain threshold, those term vectors can be aggregated into a single term vector using vector operations.

As explained above, the *min* and *max* operations together with the natural coordinatewise partial ordering on vectors allows us to preserve the concept lattice structure within the user profiles. The positive evidence is aggregated using the *meet* operator of the induced lattice: $\vec{P} = \vec{n}_1 \wedge \vec{n}_2$, where $\vec{n}_i$ represents the positive evidence in a given context. On the other hand, the negative evidence is combined via the *join* operator of the induced lattice: $\vec{N} = \vec{n}_1 \vee \vec{n}_2$. This allows us to update one or more contexts in the long-term user profile by updating the term vector for the positive evidence, the negative evidence, or both based on the user's short-term interest. If the similarity between the term vectors for the user's short-term context and the term vectors in the user profile is below the threshold, a new context is added to the user profile. Note that a new context is added only if the similarity comparison for both the vectors representing positive evidence and the vectors representing negative evidence results below the determined threshold.

Each context in the user profile can also be associated with a specific ontology. This allows the user to switch between representations of different domain ontologies while the system accurately maintains the contexts related to a particular ontology.

3 Utilizing User Context for Web Search

We now present a specific example of how the positive and negative evidence embedded in our model of the user context can be combined to enhance a user's initial query for Web search. For this purpose we use our client-side Web agent ARCH (Adaptive Retrieval based on Concept Hierarchies) (Sieg et al., 2003, 2004), which derives the user context by combining automatically learned user profiles and the semantic knowledge of the domain represented by the ontology. In order to effectively locate and provide the most appropriate result for users' information needs, the positive and negative evidence in the user's context can be combined to formulate an effective search query which can then be submitted to a search engine. Unlike traditional approaches to relevance feedback, ARCH assists users in the creation of an effective query prior to the initial search task. Let's assume a

scenario in which the user has started with a search query *Python*. The system, by matching this initial query to portions of the Yahoo concept hierarchy, presents the potentially relevant nodes to the user. Let us assume that the user has selected *Programming* and *Python* and deselected *Monty Python* under *Entertainment* and *Pythons* under *Snakes*.

We handle the above scenario by computing the term vectors for two separate elements in the induced concept lattice: one for positive evidence and one for negative evidence. Given a context $c = \langle \vec{P}, \vec{N} \rangle$, the term vector for the search query Q is computed as follows: $Q = \langle w_1, w_2, ..., w_k \rangle$ where $w_i = P_i - N_i$, if $P_i > N_i$, and $w_i = 0$, otherwise.

The resulting query effectively represents the user's context. Rather than performing the search with the user's poorly designed keyword query, our framework efficiently produces a richer and therefore less ambiguous query. The user interacts with the system in some immediate context, which may include an explicit query. Next, domain knowledge is called upon as a source of key domain concepts. In addition, a user profile, maintained over many interactions, provides clues as to the user's intentions. The query enhancement mechanism is displayed in Figure 3. Our experimental results show that our integrated approach does, indeed, result in substantial gains in retrieval precision, without sacrificing recall.

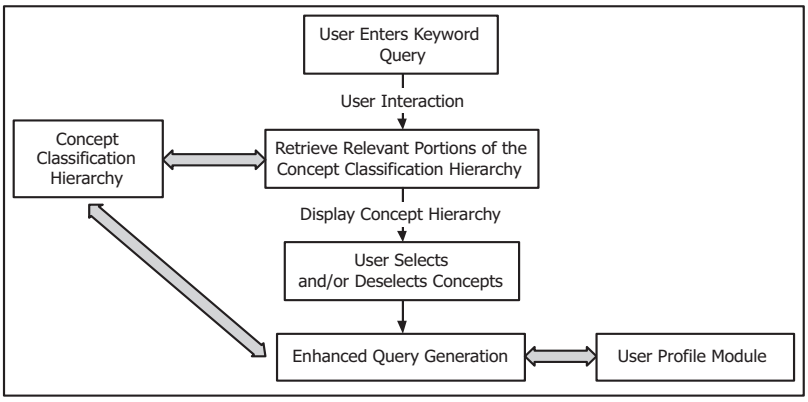

Figure 3: Query Enhancement Mechanism in ARCH

Since the queries of average Web users tend to be short and ambiguous (Spink et al., 2002), the search keywords we used in our experiments were intentionally chosen to be ambiguous. For our experiments, a one-time learning of a portion of the Yahoo concept hierarchy was necessary to build the domain ontology.

For evaluation purposes, 10 documents were collected for each word sense of our predetermined keywords which were intentionally chosen to be ambiguous. As an example, for the keyword query *python*, a total of 30 documents were collected where 10 documents related to the *snake* sense of the word *python*, 10 documents provided information about *Python* as a *programming language*, and the rest of the documents discussed the comedy group *Monty Python*. We created an index using these signal and noise documents.

We used the system to perform a simple query search and an enhanced query search for each of our keyword queries. In the case of simple query search, a term vector was built using the original keyword(s) in the query text. The search results were retrieved from the signal and noise document collection by using a cosine similarity measure for matching. The similarity scores between the queries and the documents are normalized so that the best matching document always has a score of a 100%. This allows us to apply certain thresholds to the similarity scores. Based on the user's intent for the query and the search results, we calculated the precision and recall metrics for our keyword searches at each 10 point interval between similarity thresholds of 0% to 100%. Table 1 displays a few examples of our keyword queries and search scenarios.

Table 1: Example Keyword Queries and Corresponding Search Scenarios

# of Term(s)	Query	Signal	Noise
1	bat	buying a baseball bat	information on bat mammal
1	bug	information on surveillance equipment	insects and software programming bugs
1	python	python as a snake	Monty Python and Python programming language
2	baseball bat	buying a baseball bat	information on bat mammal
2	bug spy	information on surveillance equipment	insects and software programming bugs
2	python snake	python as a snake	Monty Python and Python programming language
3	baseball equipment bat	buying a baseball bat	information on bat mammal
3	bug spy security	information on surveillance equipment	insects and software programming bugs
3	python snake reptile	python as a snake	Monty Python and Python programming language

In the case of enhanced query search, we used the query that was generated by ARCH. Based on our search scenarios, the user profiles could also be utilized to automatically select and/or deselect certain concepts in the hierarchy for the generation of the enhanced query. We measured the effectiveness of query enhancement in terms of precision and recall. Figure 4, shows the average precision and recall of the enhanced queries in comparison with simple queries. As indicated in Table 1, simple queries can contain a single keyword, two keywords, or three keywords.

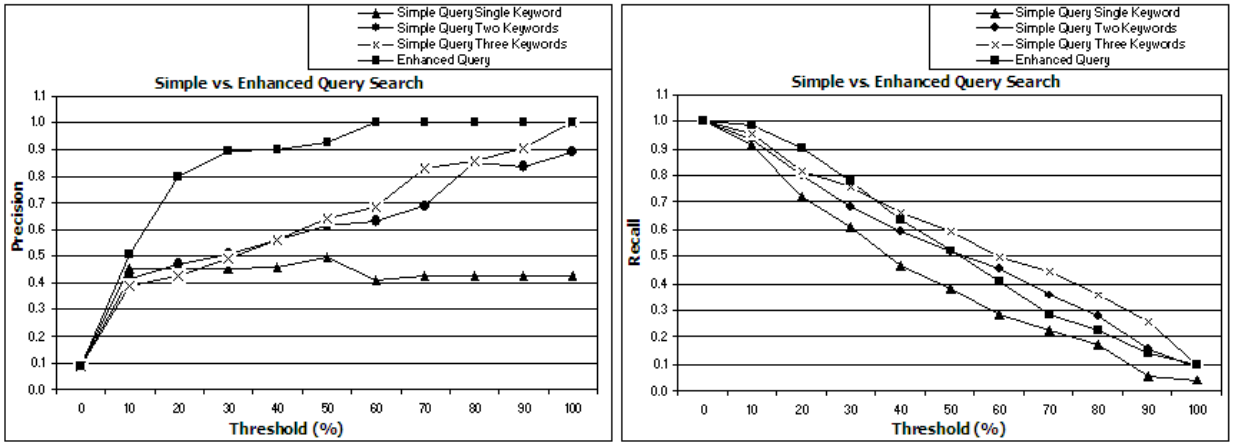

Figure 4: Avg. Precision and Recall: Enhanced Query vs. Simple Query Search

These results were obtained by averaging precision and recall values over all the search scenarios. Our results show that enhancing the query based on the user's search context leads to significantly higher precision for ambiguous queries without sacrificing recall. From the user's perspective, precision is improved since ambiguous query terms are disambiguated by the enhanced query. Utilizing the user context as described above improves the effectiveness of the search queries, especially in a typical case when relatively short queries are used.

4 Conclusions and Outlook

We have presented a framework for contextual information access using ontologies and demonstrated that the semantic knowledge embedded in an ontology combined with long-term user profiles can be used to effectively represent a user's information context. We have discussed our method for capturing the user's "context" via nodes in a concept lattice induced from the original ontology and updating the context incrementally based on user's interactions with the concepts in the ontology. Our experimental results show that utilizing the user context improves the effectiveness of the search queries, especially in the typical case of Web users who tend to use very short queries. In future work, we plan to explore how long term user profiles can be maintained in our framework. Since the user profiles are designed as a set of contexts, we must determine when to update an existing context and when to add a new context to a user profile. We also plan to extend the underlying representation of concepts in our framework from unstructured term-vector representation to an ontology-based object-oriented representation which takes into account relationships among objects based on their properties.

References

Allan, J. (1996). Incremental Relevance Feedback for Information Filtering. In *Proceedings of the ACM SIGIR 1996*.

Allan, J. et al. (2003). Challenges in Information Retrieval and Language Modeling: Report of a workshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, September 2002. *ACM SIGIR Forum*, 37(1):31-47.

Boley, D., Gini, M., Gross, R., Han, E.H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., & Moore, J. (1999) Document Categorization and Query Generation on the World Wide Web Using WebACE. *Artificial Intelligence Review*, 13(5-6):365-391.

Buckley, C., Salton, G., & Allan, J. (1994). The Effect of Adding Relevance Information in a Relevance Feedback Environment. In *Proceedings of the ACM SIGIR 1994*.

Budzik, J., Bradshaw, S., Fu, X., & Hammond, K. (2002). Supporting Online Resource Discovery in the Context of Ongoing Tasks with Proactive Software Assistants. *International Journal of Human-Computer Studies*, 56 (1):47-74.

Dumais, S., Joachims, T., Bharat, K. & Weigend, A. (2003). SIGIR 2003 Workshop Report: Implicit Measures of User Interests and Preferences. *ACM SIGIR Forum*, 37(2).

Eguchi, K. (2000). Incremental Query Expansion Using Local Information of Clusters. In *Proceedings of the 4th World Multiconference on Systemics Cybernetics and Informatics (SCI 2000)*, 2000.

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan.

Kelly, D. & Teevan, J. (2003). Implicit Feedback for Inferring User Preference: A Bibliography. *ACM SIGIR Forum*, 37(2).

Lieberman, H. (1997). Autonomous Interface Agents. In *Proceedings of the ACM Conference on Computers and Human Interface (CHI-97)*, Atlanta, GA.

Miller, G. (1997). WORDNET: An Online Lexical Database. *International Journal of Lexicography*, 3(4), 1997.

Salton , G. & McGill, M.J. (1983). Introduction to Modern Information Retrieval. New York: McGraw-Hill.

Sieg, A., Mobasher, B., Lytinen, S., & Burke, R. (2003). Concept Based Query Enhancement in the ARCH Search Agent. In *Proceedings of the 4th International Conference on Internet Computing*, Las Vegas, NV.

Sieg, A., Mobasher, B., Lytinen, S., & Burke, R. (2004). Using Concept Hierarchies to Enhance User Queries in Web-based Information Retrieval. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria.

Spink, A., Ozmutlu, H.C., & Jansen, B.J. (2002). U.S. Versus European Web Searching Trends. In *Proceedings of the ACM SIGIR Fall 2002*.