USING CONCEPT HIERARCHIES TO ENHANCE USER QUERIES IN WEB-BASED INFORMATION RETRIEVAL

Ahu Sieg, Bamshad Mobasher, Steve Lytinen, Robin Burke
{asieg, mobasher, lytinen, rburke}@cs.depaul.edu
School of Computer Science, Telecommunication and Information Systems
DePaul University
243 South Wabash Avenue, Chicago, Illinois, 60604, USA

ABSTRACT
The effectiveness of Internet search engines is often hampered by the ambiguity of user queries and the reluctance or inability of users to build less ambiguous multi-word queries. Our system, ARCH, is a client-side Web agent, which incorporates domain-specific concept hierarchies together with interactive query formulation in order to automatically produce a richer and therefore less ambiguous query. Unlike traditional relevance feedback methods, ARCH assists users in query modification prior to the search task. ARCH uses the domain knowledge inherent in Web-based classification hierarchies such as Yahoo, combined with a user's profile information, to add just those terms likely to improve the match with the user's intent. The goal of the system is therefore to meet the user's information needs by closing the gap between the user's stated query and the actual intent of the search. We present a detailed evaluation of the query enhancement in ARCH, comparing enhanced and non-enhanced queries over a range of topics. Our results show that concept-based query enhancement in ARCH leads to significantly higher precision for ambiguous queries without sacrificing recall.

KEY WORDS
Intelligent agents, concept hierarchies, query enhancement, information retrieval

1 Introduction

Despite dramatic advances in Web information retrieval leading to sophisticated Web search engines, typical users continue to find it difficult to formulate effective queries which represent their true search intent. As a result, the user experience is often unsatisfactory. The divide between the users' intent and search queries emanates from a number of factors including the users' lack of knowledge about the problem domain, their reluctance or inability to formulate more sophisticated multi-word queries, and the inability of search engines to resolve query ambiguity based on the users' search context.

Research confirms that the queries submitted to search engines by Web users are relatively short and are usually limited to less than three keywords [17]. The key to improving search, therefore, may not be to tweak search engine technology, but rather to improve the input to such systems: to help users formulate queries that better reflect their search intent, and have a higher likelihood of returning good results. In this paper, we present our client-side Web agent ARCH (Adaptive Retrieval based on Concept Hierarchies), which uses domain specific concept hierarchies together with user profiles to assist users in formulating effective search queries. We provide a detailed evaluation of the query enhancement mechanism in ARCH and show that concept-based query enhancement leads to significant improvement in search results.

Our previous work has discussed the architecture of the overall system [12] and implementation of the semantic aspect of the system [16]. The system's query enhancement uses two mutually-supporting techniques: semantic and behavioral. The behavioral aspect requires observing the users' browsing behavior for user profiling and automatic query enhancement, while the semantic aspect supports the use of a modular concept hierarchy for interactive query enhancement.

Recent work in the area of information retrieval has involved the design of intelligent tools, such as intelligent Web agents [2, 3, 6, 9, 10]. Research has also been performed in designing mechanisms to incrementally refine user queries [1, 7] and query expansion based on lexical variants such as synonyms [11]. The rapidly growing amount of information on the Web has brought new challenges and has uncovered the need for standard information retrieval work to be extended to deal effectively with the Web. As an attempt to cope with these challenges, enhanced indexing methods such as the PageRank algorithm [4], which makes use of the linkage structure of the Web to calculate a quality ranking for each Web page, have been proven to be successful in prioritizing the results of Web keyword searches.

Neither query expansion techniques nor enhanced indexing mechanisms consider the user context or knowledge of the problem domain. Traditional approaches to query formulation expand the initial query based on relevance feedback from the search results [5]. In contrast, ARCH assists users in the creation of an effective query prior to the initial search task. ARCH modifies a user's initial query based on the user's interaction with a modular concept hierarchy. Since the concept hierarchy

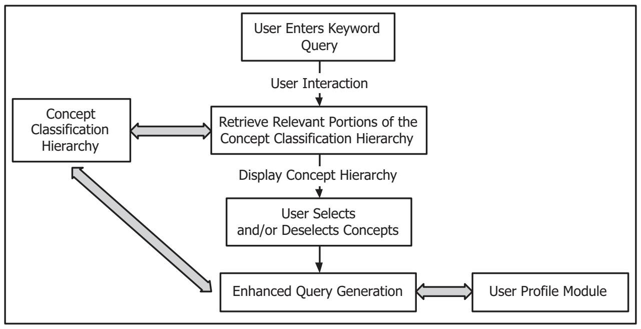

User Enters Keyword Query
User Interaction
Concept Classification Hierarchy
Retrieve Relevant Portions of the Concept Classification Hierarchy
Display Concept Hierarchy
User Selects and/or Deselects Concepts
Enhanced Query Generation
User Profile Module

Figure 1. Query Enhancement Mechanism in ARCH

is domain-specific, the query enhancement in ARCH integrates the apriori knowledge in the problem domain which may be unfamiliar to the user.

Furthermore, the behavioral aspect of ARCH aims at learning a user profile by using text mining techniques to capture behavioral patterns and profiles of users. Using these techniques enables the extraction of information implicitly contained in collections of user profiles. The learned user profiles can be utilized to provide additional context to the user's original keyword query. The goal of the system is to meet the user's information needs by closing the gap between the user's initial query and the actual intent and motivation for the search.

ARCH uses the domain knowledge inherent in a concept hierarchy, combined with a user's profile information, to add just those terms likely to improve the match with the user's intent.

The evaluation discussed in this paper focuses on the experimental results of query enhancement based on concept hierarchies. These results do not take into account the behavioral aspect which involves query enhancement based on user profiles. Our evaluation results show that concept-based query enhancement in ARCH leads to significantly higher precision for ambiguous queries without sacrificing recall.

2 Query Enhancement Mechanism

ARCH has an interactive approach for assisting the user in formulating an effective search query. The query enhancement mechanism is displayed in Figure 1. The user's initial query is modified based on the user's interaction with a concept hierarchy. In addition, ARCH passively learns a user profile by observing the user's past browsing behavior. The profile information is used to provide additional context to the user's initial query.

2.1 Query Enhancement Based on Concept Hierarchies

The most critical element of query enhancement in the system is the concept classification hierarchy. ARCH

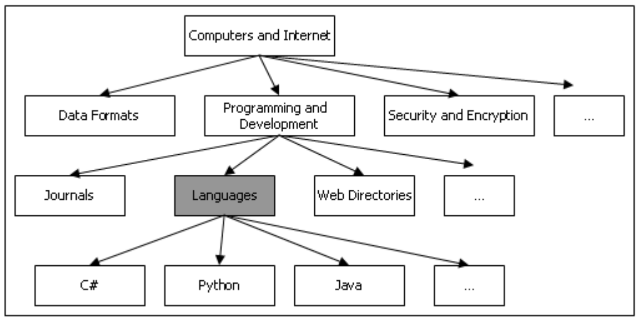


Figure 2. Portion of the Yahoo hierarchy corresponding to the node *Languages*

includes an offline component which allows the system to learn a concept classification hierarchy which is then maintained in the system in aggregate form. Since ARCH can learn various concept hierarchies, the user is allowed to switch among the representations of different domain-specific hierarchies depending on the goals of the search. The current implementation of the system uses the Yahoo concept hierarchy.

The system maintains an aggregate representation of the concept hierarchy by pre-computing a weighted term vector for each node in the hierarchy which represents the centroid of all documents and subcategories indexed under that node.

Let's consider a specific node n in the concept hierarchy and assume this node contains D_n, which is a collection of individual documents. This node also contains S_n, which is a set of subconcepts. The term vector for the node n is computed as follows:

$$T_n = \left[\left(\sum_{d \in D_n} T_d \right) / |D_n| + \sum_{s \in S_n} T_s \right] / (|S_n| + 1)$$

In the above formula, T_d is the weighted term vector which represents an individual document d indexed under node n and T_s is the term vector which represents the subconcept s of node n. Note that a term vector is calculated for each indexed document under a concept. The term vectors for the indexed documents are added to get a single term vector which represents the average. This term vector is added to the term vectors for the subconcepts to calculate the final average for the concept.

A global dictionary of terms is created by performing standard information retrieval text preprocessing methods. A stop list is used to remove high frequency, but semantically non-relevant terms from the content. Porter stemming [13] is utilized to reduce words to their stems. For computing the term weights extracted from content, the system employs a standard function of the term frequency and inverse document frequency (tf.idf) [15, 8]. As an example, Figure 2 displays a portion of the Yahoo hierarchy corresponding to the node *Languages*. The term vector which represents the node *Languages* is computed from a combination of the documents indexed under this node as well as the term vectors representing its subconcepts such as *C#*, *Python*, and *Java*. The partial term vector for this node is displayed in Figure 3.

languag:1	code:0.364	java:0.297	librari:0.256
python:0.452	interpret:0.316	compil:0.283	user:0.251
program:0.38	implement:0.315	document:0.266	
object:0.364	exampl:0.298	file:0.264	

Figure 3. Partial term vector which provides an aggregate representation of the node *Languages*

To initiate the query generation process, the user enters a keyword query. Based on the user's interaction with the system, the system responds by displaying the appropriate portions of the hierarchy. The user interface allows the user to select those categories which are relevant to the intended query and to deselect those categories which are not relevant. We employ a variant of Rocchio's method [14] for relevance feedback to generate the enhanced query. The pre-computed term vectors associated with each node in the hierarchy are used to enhance the original query as follows:

$$Q_2 = \alpha.Q_1 + \beta.\sum T_{sel} - \gamma.\sum T_{desel}.$$

In the above formula, T_{sel} is a term vector for one of the nodes selected by the user. On the other hand, T_{desel} is a term vector for one of the nodes which is deselected by the user. The factors α, β, and γ are respectively the relative weight associated with the original query, the relative weight associated with the selected concepts, and the relative weight associated with the deselected concepts.

3 Experimental Evaluation

The purpose of our system is to assist users in formulating an effective search query. Since the queries of average Web users tend to be short and ambiguous [17], the search keywords we use in our evaluation are intentionally chosen to be ambiguous. In the worst case scenario, the user will enter an extremely ambiguous search query which contains only one keyword. Our evaluation results demonstrate what an outstanding job ARCH does in solving the problem of query disambiguation even with the worst case scenario.

We measure the effectiveness of query enhancement in terms of precision and recall. Our goal is to show that concept-based query enhancement in ARCH leads to significantly higher precision for ambiguous queries without sacrificing recall.

3.1 Evaluation Methodology and Experimental Data Sets

For our experiments, we use categories from the Yahoo concept hierarchy, a real-world domain. A one-time learning of certain portions of the Yahoo hierarchy is necessary in order to represent the concept hierarchy in our our system. Also for evaluation purposes, 10 documents are collected for each word sense of our predetermined keywords which are intentionally chosen to be ambiguous. As an example, for the keyword query *python*, a total of 30 documents are collected where 10 documents relate to the *snake* sense of the word *python*, 10 documents provide information about *Python* as a *programming language*, and the rest of the documents discuss the comedy group *Monty Python*. For each of our keyword queries, several search scenarios are created with the intention of solving the problem of query disambiguation. Our keyword queries and search scenarios are displayed in Table 1. Depending on the search scenario, each document

# 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

Table 1. Example Keyword Queries and Corresponding Search Scenarios

in our collection can be treated as a signal or a noise document. The signal documents are those documents that should be ranked high in the search results. The noise documents are those documents that should be ranked low or excluded from the search results. In order to create an index for the signal and noise documents, a term frequency and inverse document frequency (tf.idf) weight is computed for each term in the document collection using the global dictionary of the concept hierarchy.

As depicted in Table 1, our keyword queries are used to run a number of search scenarios. The first set of keyword queries contain only one term and include the following: *bat*, *bug*, *hardware*, *mouse*, and *python*. For example, in order to evaluate the search results when the single keyword *bat* is typed by the user as the search query, one scenario assumes that the user is interested in *buying a baseball bat*. Therefore, in this scenario, the documents that are relevant to the *baseball* sense of the word *bat* are treated as signal documents whereas the documents that are related to the *bat mammal* are treated as noise documents.

The second set of queries contain two terms and include the following: *baseball bat*, *bat mammal*, *bug spy*, *hardware computer*, *hardware tools*, *hardware upgrade*, *mouse computer*, and *python snake*. For example, in the case of a user typing *bug spy* as the search query, our search scenario assumes the user's intent for the query is to find information about the *surveillance* sense of the word *bug* as opposed to the *software programming* or the *insect* senses.

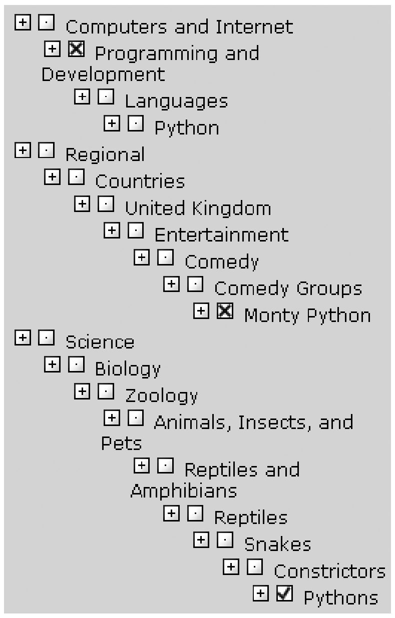
The third set of queries contain three terms and include the following: *baseball equipment bat*, *bat mammal fly*, *bug spy security*, *computer hardware upgrade*, *computer hardware mouse*, *home hardware tools*, and *python snake reptile*. As the number of keywords in a query increase, the search query becomes less ambiguous.

Our evaluation methodology is as follows. We use 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 is built using the original keyword(s) in the query text. Removal of stop words and stemming is utilized. Each term in the original query is assigned a weight of 1.0.

In the case of enhanced query search, we use the query that is generated by ARCH. Based on our search scenarios, we select and/or deselect certain concepts in the hierarchy for the generation of the enhanced query. The search results are retrieved from the signal and noise document collection by using a cosine similarity measure for matching. The similarity scores 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. For each of our search scenarios, we calculate the precision and recall at each 10 point interval between a similarity threshold of 0% and a similarity threshold of 100%.

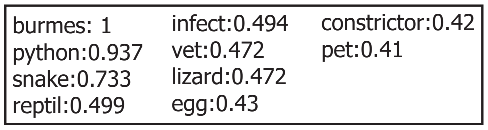
As an example for the generation of the enhanced query, consider the scenario in which we start with a single keyword query *python*. The system will respond to our query by displaying the relevant portions of the Yahoo hierarchy including the parents, children, and siblings of the nodes corresponding to the initial keyword query. We can now interact with the concept hierarchy by selecting and/or deselecting various nodes.

In this case, our ambiguous keyword causes the system to display several different portions of the hierarchy. Our specific search scenario aims at finding information about *python* as a *snake*. Therefore, we select the concept *Pythons* under *Reptiles and Amphibians* in the hierarchy. Figure 4 displays the portions of the Yahoo hierarchy that corresponds to this scenario. In this case, we are definitely not interested in gathering information about *programming languages*. Therefore, we deselect the node *Programming and Development*. We are also not interested in finding out about the comedy group *Monty Python*, thus deselect *Monty Python*. Using the selected and deselected nodes, the system generates an enhanced query as depicted in Figure 5.



```
⊞ ☐ Computers and Internet
    ⊞ ☒ Programming and Development
        ⊞ ☐ Languages
            ⊞ ☐ Python
⊞ ☐ Regional
    ⊞ ☐ Countries
        ⊞ ☐ United Kingdom
            ⊞ ☐ Entertainment
                ⊞ ☐ Comedy
                    ⊞ ☐ Comedy Groups
                        ⊞ ☒ Monty Python
⊞ ☐ Science
    ⊞ ☐ Biology
        ⊞ ☐ Zoology
            ⊞ ☐ Animals, Insects, and Pets
                ⊞ ☐ Reptiles and Amphibians
                    ⊞ ☐ Reptiles
                        ⊞ ☐ Snakes
                            ⊞ ☐ Constrictors
                                ⊞ ☑ Pythons
```

Figure 4. Portions of the Yahoo hierarchy corresponding to the query *python*. The user has selected the node *Pythons* and deselected the nodes *Programming and Development* and *Monty Python*



burmes: 1	infect:0.494	constrictor:0.42
python:0.937	vet:0.472	pet:0.41
snake:0.733	lizard:0.472	
reptil:0.499	egg:0.43	

Figure 5. Enhanced Query for the keyword *python* based on the selected and deselected nodes in the hierarchy

As mentioned above, each search scenario assumes that the user has a specific goal for the search. Based on the user's intent for the query and the search results, we calculate the precision and recall metrics for our keyword searches. For each of our search scenarios, the precision and recall metrics are calculated at each 10 point interval between a similarity threshold of 0% and a similarity threshold of 100%.

3.2 Evaluation Results

In order to compare the simple query search results with the enhanced query search results, we have created separate precision and recall graphs for each of our search scenarios. In the case of the simple query search, precision is expected to improve as more terms are added to the query. Our evaluation results verify that precision is higher for the simple query search when using multiple keywords than performing a simple query search using a single keyword.

As displayed in Figure 6, our evaluation results also show significant improvement in precision when using the enhanced query for searching. As we can see in Figure 7, recall has also improved. Figure 8 depicts a scatter

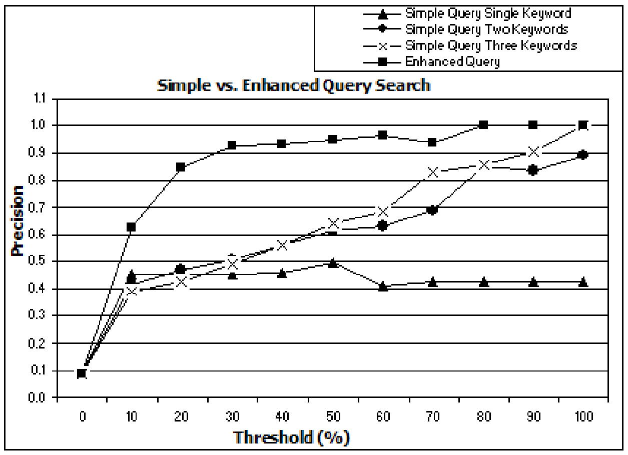


Figure 6. Average Precision for Enhanced Query versus Simple Query Search

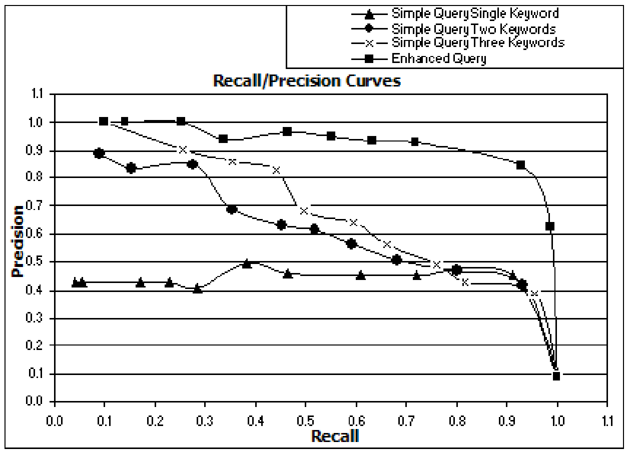


Figure 8. Average Recall/Precision Curve for Enhanced Query versus Simple Query Search

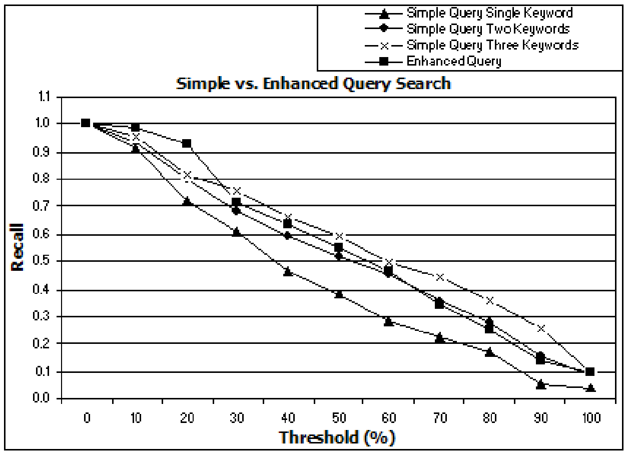


Figure 7. Average Recall for Enhanced Query versus Simple Query Search

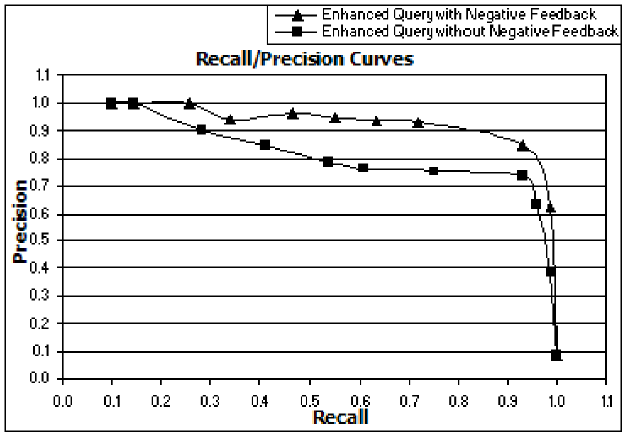


Figure 9. Average Recall/Precision Curve for Enhanced Query with Negative Feedback versus Enhanced Query without Negative Feedback

plot, namely Recall/Precision curve, which demonstrates the advantage of the enhanced query. Even when using three keywords, the simple query still performs fairly poor compared to our enhanced query.

These results show that the query enhancement in ARCH significantly improves the effectiveness of the search query. We see two types of improvement in the search results using the enhanced query. From the user's perspective, precision is improved since ambiguous query terms are disambiguated by the enhanced query. In addition, when comparing single keyword queries to the enhanced query, we see better recall in the search results since additional query terms retrieve documents that would not be retrieved by using only the original keyword query.

3.3 Impact of Negative Feedback on the Enhanced Query

We have experimented with variations of selection and/or deselection of concepts in the hierarchy in order to find trends that might contribute to improving the enhanced query. We have discovered that the deselection of the concepts that are not relevant to the search query has a significant impact on the quality of the enhanced query. In other words, as part of the interactive query formulation, providing negative feedback improves the quality of the query that is generated by ARCH. Our evaluation methodology is used to compare the search results that are generated by the enhanced query with negative feedback and the enhanced query results without negative feedback. As an example, consider the scenario in which we start with a single keyword query *hardware*. Again, our ambiguous keyword causes the system to display several different portions of the hierarchy including the parent nodes *Computers and Internet* and *Home Improvement*. Our specific search scenario aims at finding information about *computer hardware*. Therefore, we select the concept *Hardware* under *Computers and Internet* in the hierarchy. However, we intentionally do not deselect the concept *Hardware* under *Home Improvement*.

Since the negative feedback in our system is based on domain specific information, the enhanced query with negative feedback results in higher precision and lower recall. Figure 9 depicts the Recall/Precision curve, which clearly illustrates the advantage of the enhanced query with negative feedback.

As a result, lack of negative feedback in the enhanced query has a negative impact on search results. In other words, the queries that are generated without deselection of concepts seem to be less effective in retrieval than the enhanced queries that are generated after proper selection and/or deselection of concepts.

4 Conclusions and Outlook

We have presented a detailed evaluation of query enhancement in ARCH based on the user's interaction with a concept hierarchy. Our evaluation results have shown that concept-based query enhancement in ARCH leads to significantly higher precision for ambiguous queries without sacrificing recall. For future work, our immediate focus will be on the evaluation of query enhancement based on user profiles.

Our future work will also involve integrating the system with a specific search engine of the World Wide Web such as Google or AltaVista. This will require the translation of the enhanced query that is generated by ARCH into a Boolean query which can be submitted to the search engine.

We are also planning on expanding the idea of query generation based on concept hierarchies to query generation based on ontologies. This will allow our system to take advantage of other semantic relationships in a domain-specific ontology in addition to the hierarchical relationship in the concept hierarchy.

References

[1] J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of the ACM SIGIR 1996*, 1996.

[2] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using webace. *Artificial Intelligence Review*, 13(5-6):365–391, 1999.

[3] K. Bollacker, S. Lawrence, and C. Lee Giles. Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceeding of the 2nd International Conference on Autonomous Agents*, Minneapolis, MN, 1998.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the ACM SIGIR1994*, 1994.

[6] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1-2):69–113, 2000.

[7] K. Eguchi. Incremental query expansion using local information of clusters. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000)*, 2000.

[8] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[9] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

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

[11] G. Miller. Wordnet: An online lexical database. *International Journal of Lexicography*, 3(4), 1997.

[12] S. Parent, B. Mobasher, and S. Lytinen. An adaptive agent for web exploration based on concept hierarchies. In *Proceedings of the Ninth International Conference on Human Computer Interaction*, New Orleans, LA, 2001.

[13] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[14] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[15] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY, 1983.

[16] A. Sieg, B. Mobasher, S. Lytinen, and R. Burke. Concept based query enhancement in the arch search agent. In *Proceedings of the 4th International Conference on Internet Computing*, Las Vegas, NV, 2003.

[17] A. Spink, H.C. Ozmutlu, S. Ozmutlu, and B.J. Jansen. U.s. versus european web searching trends. In *Proceedings of the ACM SIGIR Fall 2002*, 2002.