Inferring User's Information Context: Integrating User Profiles and Concept Hierarchies

Ahu Sieg, Bamshad Mobasher, and Robin Burke

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

Abstract *The critical elements that make up a user's information context include the user profiles that reveal long-term interests and trends, the short-term information need as might be expressed in a query, and the semantic knowledge about the domain being investigated. The next generation of intelligent information agents, that can seamlessly integrate these elements into a single framework, are enabled to effectively locate and provide the most appropriate results for users' information needs. In this paper we present one such framework for contextualized information access. We model the problem in the context of our client-side Web agent ARCH (Adaptive Retrieval based on Concept Hierarchies). In ARCH, the user profiles are generated using an unsupervised document clustering technique. These profiles, in turn, are used to automatically learn the semantic context of user's information need from a domain-specific concept hierarchy. Our experimental results show that implicit measures of user interests, combined with the semantic knowledge embedded in a concept hierarchy, can be used effectively to infer the user context and improve the results of information retrieval.*

Keywords: User profiles, concept hierarchies, intelligent agents, information retrieval

1 Introduction

The heterogeneity and the lack of structure of the information sources on the World Wide Web, such as hypertext documents, make automated discovery, organization, and management of Web-based information difficult. Despite great advances in Web-based information retrieval technologies, the user interactions with Web search engines can generally be characterized as one size

fits all [2]. This is mainly due to the fact that current systems provide no mechanisms for the representation of user preferences and the semantic context of the domain or the task at hand. Indeed, contextual retrieval has been identified as a long-term challenge in information retrieval. Allan et al. [2] 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.*

In most information retrieval applications, it is difficult to get explicit feedback from users about the relevance of the results and the appropriateness of the presentation [7]. Furthermore, the burden is placed on the users to formulate effective queries which represent their true search intent. Typical users find this task quite difficult: research confirms that the queries submitted to search engines by Web users are relatively short and are usually limited to less than three keywords [21]. As a result, the user's search experience is often unsatisfactory.

In previous work we have presented our client-side Web agent ARCH (Adaptive Retrieval based on Concept Hierarchies) [15, 19]. ARCH allows the user to interact with a concept classification hierarchy and uses the results of this interaction to derive an explicit user context for each query session. This user context is represented in the form of an enhance query which can be used for Web search.

Various forms of query enhancement have long been a research topic in information retrieval, from relevance feedback [4, 6] to incremental query refinement [1, 8] to query expansion based on lexical variants such as synonyms [14]. 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. Unlike traditional approaches, ARCH assists users in the creation of an effective query prior to the initial search task. The initial query is modified based on the user's interaction with a domain-specific concept hierarchy. Portions of the hierarchy are used to provide the semantic context for the initial query. The goal of the system is, thus, to close the gap between the initial representation of the user's information need and the actual intent for the search.

A number of researchers have explored intelligent Web agents that learn about user's interests in Web-based information access [3, 10, 13]. 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 [7, 12]. An example of this trend is found in [5] in which the user's work context in office applications is used as a source of information access context. 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 extend ARCH to provide a more integrated framework for contextualized information access which combines automatically learned user profiles and the semantic knowledge of the domain represented by the concept hierarchy. Our specific goal is to reduce or eliminate the need for explicit user interaction by utilizing the user profiles, while still maintaining the explicit representation of the semantic knowledge as an integral part of the framework. To this end, we use unsupervised document clustering and other text mining techniques for the derivation of user profiles based in past actions. We, then, use these profiles, in lieu of explicit user interaction, to derive the user context from the concept hierarchy. Our experimental results show that our integrated approach does, indeed, result in substantial gains in retrieval precision, without sacrificing recall.

2 Inferring User Context for Information Access

Our main goal is to create a unified model of user information goals through the seamless integration of semantic knowledge with automatically learned user profiles. In this section we discuss our approaches for the derivation of user profiles from document clusters and for learning an aggregate representation of the domain ontology in the forma of a concept taxonomy. We then discuss how these two sources of information can be used for contextualized information access and retrieval.

2.1 Derivation of User Profiles from Document Clusters

The profiling component of ARCH attempts to learn a model of user behavior through the passive observation of user's information access activity (including browsing and searching) over time. Other 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 [7]. Our research attempts to automatically determine the topics of interest to the user. Through this observation, ARCH collects a set of documents in which the user has shown interest. Several factors, including the frequency of visits to a page, the amount of time spent on the page, and other user actions such as bookmarking a page are used as bases for heuristics that are employed by the system to automatically collect these documents without user intervention.

Once enough documents have been collected as part of the profile generation, a clustering algorithm is used to cluster the documents into semantically related categories based on a vector similarity measure. Each document is represented as a term vector. Term weights are computed by using the term frequency and inverse document frequency (tf.idf) measure. Our method for the generation of topical profiles is similar to the concept indexing method [11]. Individual profiles are obtained by computing the centroid vector of each of the

document clusters produced by the clustering algorithm. Specifically, given a document collection D and a document cluster C $\subseteq$ D, we construct a profile pr_c as a set of term-weight pairs:

$$pr_c = \{\langle t, weight(t, pr_c)\rangle \,|weight(t, pr_c) \geq \mu\,\}$$

In the above equation, the significance weight, *weight(t,pr_c)*, of the term t within the profile pr_c is computed as follows:

$$weight(t, pr_c) = \frac{1}{|C|}. \sum_{d \in C} w(t, d)$$

In the above formula, *w(t, d)* is the weight of term t in the document vector d. The threshold μ is used to filter out terms which are not significant within each profile. Each profile is represented as a vector in the original n-dimensional space of terms, where n is the number of unique terms in the global dictionary.

2.2 Representation of Domain Knowledge

To support the users with the task of searching on the Web, ARCH exploits a concept classification hierarchy. The Web is home to Web concept hierarchies, collections of documents labeled with conceptual labels and related to each other in a hierarchical arrangement. The most well known of these is Yahoo (http://www.yahoo.com), which has a multi-level hierarchy covering the full breadth of the topics found on the Web. ARCH uses the domain knowledge inherent in Yahoo and allows the users to interact with the concept hierarchy to explicitly provide user context for each of their query sessions.

ARCH includes an offline component which allows the system to learn a concept classification hierarchy which is then maintained in the system in aggregate form. The system maintains this aggregate representation 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, a collection of individual documents, and, S_n, a set of subconcepts. The term vector for the node n is computed as follows:

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

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's stemming algorithm [16] 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) [18, 9].

2.3 Utilizing User Context for Web Search

As noted earlier, once enough documents have been collected for profiling, these documents go through a process which clusters them into semantically related categories. The documents are preprocessed and converted into n-dimensional term vectors by computing the tf.idf weights for each term, where n is the total number of unique terms in the global dictionary. It should be noted that the global dictionary contains all unique terms in both the profile documents as well as those used in the aggregate representation of the concepts in the hierarchy. This allows us to use a clustering application, which performs k-means clustering, to partition the document set into groups of similar documents based on a measure of vector similarity. Individual profiles, each representing a topic category, are computed based on the centroid of the document clusters. As a result, each profile is represented as a term vector and the results of our computations are maintained in an XML document. Figure 1 depicts the format of the XML document which contains each profile and the terms that belong to these profiles along with the term weights. Only a few sample terms have been included in the figure.

Our previous work on ARCH focused on an interactive approach for assisting the user in formulating an effective search query. To initiate the query generation process, the user enters a keyword query. The system matches the term vectors representing each node in the concept hierarchy with the list of keywords typed by the user. Those nodes which exceed a similarity threshold are displayed to the user, along with other adjacent nodes. These nodes are considered to be the matching nodes in the concept hierarchy. Once the relevant portions of the hierarchy are displayed, the user interface allows the user to provide explicit feedback by selecting those categories which are relevant to the intended query and to deselect those categories which are not relevant. The query enhancement mechanism is displayed in Figure 2.

We employ a variant of Rocchio's method [17] 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, Q_1 as follows:

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

```
<PROFILES>
    <PROFILE>
        <TermList>
                <Term>
                        <Weight>12.5</Weight>
                        <Name>languag</Name>
                </Term>
                <Term>
                        <Weight>9.25</Weight>
                        <Name>python</Name>
                </Term>
                <Term>
                        <Weight>2.245</Weight>
                        <Name>program</Name>
                </Term>
        </TermList>
    </PROFILE>
    <PROFILE>
        <TermList>
                <Term>
                        <Weight>20</Weight>
                        <Name>python</Name>
                </Term>
                <Term>
                        <Weight>14.5</Weight>
                        <Name>snake</Name>
                </Term>
                <Term>
                        <Weight>8.7</Weight>
                        <Name>reptil</Name>
                </Term>
        </TermList>
    </PROFILE>
</PROFILES>
```

Fig. 1. The profile information in XML format

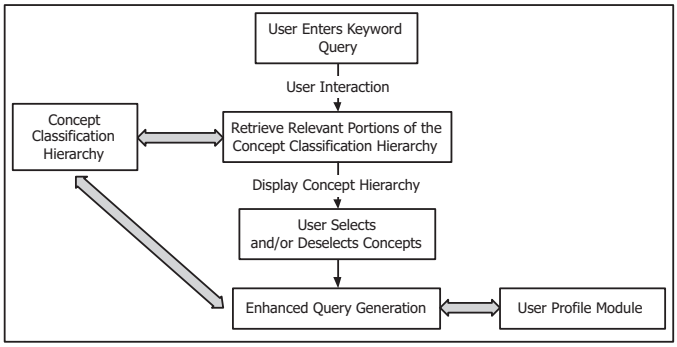

Fig. 2. Query Enhancement Mechanism in ARCH

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.

In order to eliminate the need for the explicit user feedback, we alternatively utilize the user profiles for the selection and deselection of concepts. Each individual profile is compared to the original user query for similarity. Those profiles which satisfy a similarity threshold are then compared to the matching nodes in the concept hierarchy. Note that the matching nodes include those nodes which originally exceeded a similarity threshold when compared to the user's original keyword query. While comparing these nodes

to the appropriate user profiles, each node is assigned a similarity score. Once all of the appropriate nodes are processed, the node with the highest similarity score is used for automatic selection. The nodes with relatively low similarity scores are used for automatic deselection.

3 Experiments with User Profiles

Our previous work provides a detailed summary of our evaluation results for query enhancement in ARCH based on concept hierarchies [20]. Our previous evaluation methodology involved the manual selection and deselection of concepts. Our current approach allows us to utilize the information stored in the user profiles for the automatic selection and deselection of concepts in the concept hierarchy in order to automatically determine user's context for the original keyword search query. Thus, we are able to utilize implicit measures of user interests to automatically provide explicit user feedback.

Since the queries of average Web users tend to be short and ambiguous [21], the search keywords we use in our experiments are intentionally chosen to be ambiguous. For each of our keyword queries, several search scenarios are created with the intention of solving the problem of effective discovery of user context. Our keyword queries and search scenarios are displayed in Table 1. 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 Experimental Setup

For our experiments, a one-time learning of the concept hierarchy is necessary to build the aggregate representation of the concept hierarchy in our system. In addition, two sets of user profiles are created with the intention of having each set represent specifically different user interests. The contents of the user profiles are depicted in Table 2.

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*.

Depending on the search scenario, each document 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.

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	hardware	home hardware and tools	upgrading computer hardware
1	python	python as a snake	Monty Python and Python programming language
2	baseball bat	buying a baseball bat	information on bat mammal
2	home hardware	home hardware and tools	upgrading computer hardware
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	computer hardware upgrade	upgrading computer hardware	home hardware and tools
3	python snake reptile	python as a snake	Monty Python and Python programming language

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.

Table 2. User Interests and Corresponding User Profiles

User Profiles	User Interest
Set 1	buying a baseball bat
Set 1	home hardware and tools
Set 1	information about a pet python
Set 2	information on bat as a mammal
Set 2	upgrading computer hardware
Set 2	Python programming language

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*. In this case, our user is represented by the first set of user profiles. 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, the user profiles are utilized to automatically 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 mentioned above, each search scenario assumes that the user has a specific goal for the search. The user's context is derived from the information stored in the user profiles. 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 3, our evaluation results show significant improvement in precision when using the automatically enhanced query for searching. As we can see in Figure 4, recall has also improved.

These results show that enhancing the query based on the user's search context improves the effectiveness of the search query. We see two types of improvement in the search results using the enhanced query in ARCH. 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

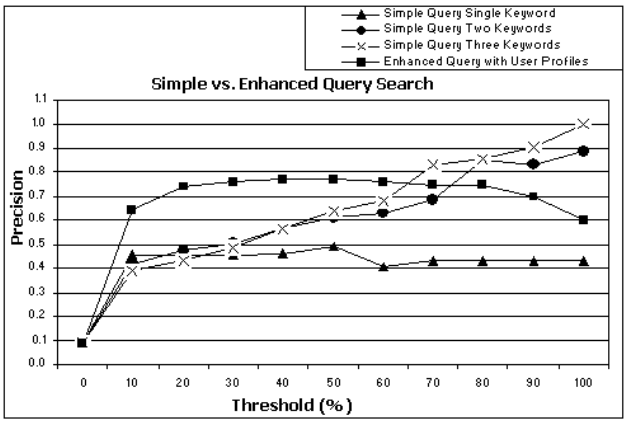

Fig. 3. Average Precision for Enhanced Query versus Simple Query Search

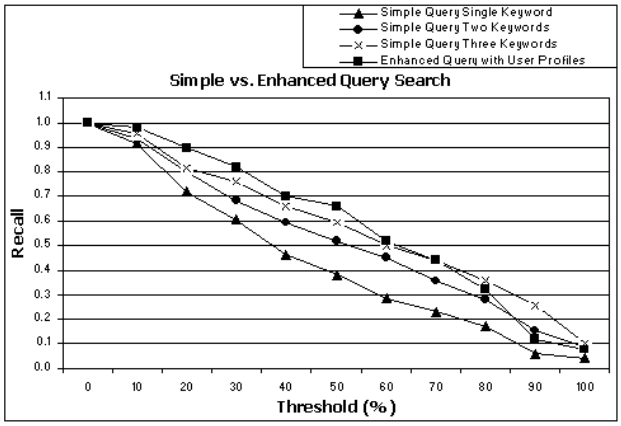

Fig. 4. Average Recall for Enhanced Query versus Simple Query Search

results since additional query terms retrieve documents that would not be retrieved by using only the original keyword query.

4 Conclusions and Outlook

We have presented a framework for the integration of user profiles in ARCH with semantic knowledge represented by a concept hierarchy. We have shown that the information from the user profiles can be successfully utilized to replace the selection and/or deselection of concepts in a domain-specific concept hierarchy. Our experiments also show that we able to create effective user profiles by using clustering techniques. The Web documents that are of

interest to the user are divided into clusters for user profiling. The user profiles that are derived from these document clusters accurately represent user interests and user context. This allows for the automatic generation of an enhanced query which retrieves much better search results than a user's poorly designed keyword query.

For future work, our immediate focus will be on performing a formal evaluation of query enhancement based on user profiles. 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 in which information extracted from a page is organized into one of more objects with associated attributes and relations. This will enable information access and discovery based on a more structured representation of Web information resources, as well as the integration of information access with semantic inference mechanisms.

References

1. J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of the ACM SIGIR 1996*, 1996.
2. J. Allan, J. Aslam, N. Belkin, C. Buckley, J. Callan, B. Croft, S. Dumais, N. Fuhr, D. Harman, D. J. Harper, D. Hiemstra, T. Hofmann, E. Hovy, W. Kraaij, J. Lafferty, V. Lavrenko, D. Lewis, L. Liddy, R. Manmatha, A. McCallum, J. Ponte, J. Prager, D. Radev, P. Resnik, S. Robertson, R. Rosenfeld, S. Roukos, M. Sanderson, R. Schwartz, A. Singhal, A. Smeaton, H. Turtle, E. Voorhees, R. Weischedel, J. Xu, and C. Zhai. 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, 2003.
3. 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.
4. 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.
5. J. Budzik, S. Bradshaw, X. Fu, and K. Hammond. Supporting online resource discovery in the context of ongoing tasks with proactive software assistants. *International Journal of Human-Computer Studies*, 56(1):47–74, 2002.
6. Z. Chen. Multiplicative adaptive algorithms for user preference retrieval. *Lecture Notes in Computer Science*, 2108, 2001.
7. S. Dumais, T. Joachims, K. Bharat, and A. Weigend. Sigir 2003 workshop report: Implicit measures of user interests and preferences. *ACM SIGIR Forum*, 37(2):–, Fall 2003.
8. 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.

9. W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.
10. 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.
11. G. Karypis and E.H. Han. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management (CIKM'2000)*, McLean, VA, 2000.
12. D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *ACM SIGIR Forum*, 37(2):–, Fall 2003.
13. H. Lieberman. Autonomous interface agents. In *Proceedings of the ACM Conference on Computers and Human Interface (CHI-97)*, Atlanta, GA, 1997.
14. G. Miller. Wordnet: An online lexical database. *International Journal of Lexicography*, 3(4), 1997.
15. 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.
16. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
17. 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.
18. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.
19. 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.
20. A. Sieg, B. Mobasher, S. Lytinen, and R. Burke. 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, 2004.
21. 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.