Concept Based Query Enhancement in the ARCH Search Agent

Ahu Sieg, Bamshad Mobasher, Steve Lytinen, Robin Burke
School of Computer Science, Telecommunication and Information Systems
DePaul University
243 South Wabash Avenue, Chicago, Illinois, 60604, USA

Abstract *The quality of the search experience has been an enduring problem for the World Wide Web. One of the well-known difficulties is the tendency of users to use short, under-specified and ambiguous queries, which tend to retrieve large amounts of irrelevant material. Traditional query expansion and relevance feedback approaches have been used to address this problem, but without as much success as these techniques have garnered in more restricted areas such as traditional IR collections. This paper presents ARCH, an interactive query formulation aid that is based on conceptual categories. The user's query is reformulated to include categories that the user recognizes as important and exclude those that are not important. Unlike query expansion techniques, which might add lists of synonyms to increase recall, ARCH uses the domain knowledge inherent in Web-based classification hierarchies such as Yahoo to add just those terms likely to improve the match with the user's intent. The goal of the system therefore is to meet the user's information needs by closing the gap between the user's stated query and the actual intent of the search. In this paper, we discuss the implementation of ARCH with a particular focus on the query enhancement mechanism. We also discuss our evaluation of the system.*

Keywords: Concept hierarchies, intelligent Web agents, query enhancement, query reformulation

1 Introduction

The quality of the search experience has been an enduring problem for the World Wide Web. Although effective search engines are available on the Web, users do not find it easy to formulate effective queries to use with these engines, and as a result, the user experience is often unsatisfactory. The average Web user types less than three keywords when using a search engine [16], a statistic that has changed little in the history of the Web. 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 describe the implementation of the query enhancement component for ARCH (Adaptive Retrieval based on Concept Hierarchies), that uses domain-specific concept hierarchies to assist users in formulating an effective search query. Previous publications have introduced ARCH as an adaptive agent for retrieval [12]. 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 a concept hierarchy for interactive query enhancement. Our previous work has discussed the architecture of the overall system and the integration of the semantic and behavioral components. This paper discusses the implementation of ARCH with a particular focus on the query enhancement mechanism based on the semantic aspect of the system. We also discuss our evaluation of the system.

Recent work in the area of information retrieval involves 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]. Earlier work has focused on query expansion based on lexical variants such as synonyms [11]. This research attempts to find a solution that will generate richer queries than traditional simple query expansion methods.

In ARCH, the initial query is modified based on the user's interaction with a modular concept hierarchy. Since the concept hierarchy is domain-specific, the query enhancement in ARCH is based on domain knowledge. Therefore, the queries that are generated by ARCH have the potential to be far superior to those queries created by traditional query expansion mechanisms which use domain independent heuristics such as synonyms. In addition, earlier work has focused on query reformulation based on relevance feedback from the search results [4].

In contrast, this research focuses on the creation of an effective query prior to the initial search task. 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.

Because the concept hierarchy itself is simply another input to the system, ARCH can switch among the representations of different domain-specific hierarchies. For example, a user interested in performing a general search on the Web can use the Yahoo concept hierarchy whereas a user interested in medical information can use a medical concept classification hierarchy such as Medical Subject Headings (http://www.nlm.nih.gov/mesh/).

2 System Overview

The system consists of an offline and an online component. The offline component handles the learning of the concept classification hierarchy. The online component involves displaying the concept hierarchy to the user, allowing the user

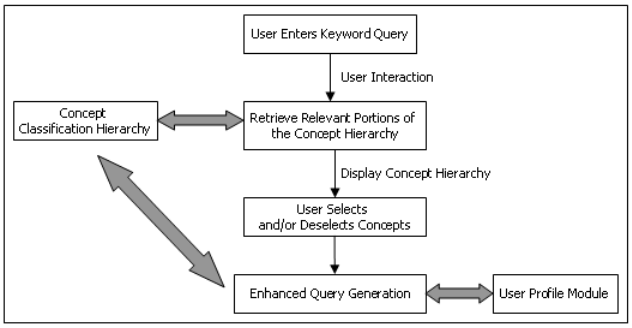

Figure 1: Query Enhancement Mechanism in ARCH

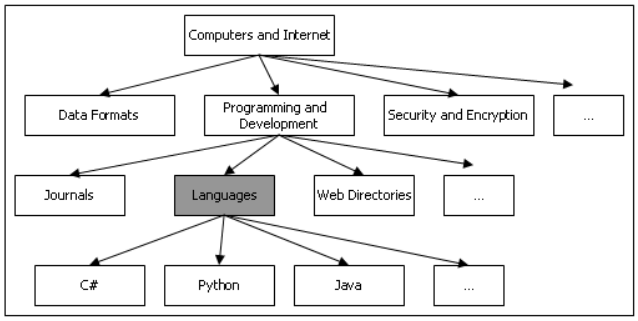

Figure 2: Portion of the Yahoo hierarchy corresponding to the node *Languages*

to select and/or deselect concepts, and generating the enhanced query. The query enhancement mechanism is displayed in Figure 1. The system is implemented using Microsoft Visual C# .NET. The system uses a SQL Server 2000 database for the initial formulation of the concept hierarchy. The aggregate representation of the concept hierarchy is represented in XML.

3 Offline Component

The offline component handles the learning of the concept classification hierarchy by building an aggregate representation of the concept hierarchy. This involves a spidering agent and a stand-alone application called the *Concept Hierarchy Application*. The system maintains an aggregate representation of the concept hierarchy by pre-computing the term vectors for each node in the hierarchy. The current implementation of the system uses the Yahoo hierarchy.

As an example, Figure 2 displays a portion of the Yahoo hierarchy corresponding to the node *Languages*. The partial term vector for this

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*

node is displayed in Figure 3. 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*. Let's consider a specific node n in the concept hierarchy and assume this node contains a collection of D_n of individual documents and a set of S_n 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.

3.1 Spidering Agent

The agent starts with a given URL (i.e. http://dir.yahoo.com). Then it identifies the links for concepts, which are referred to as categories in Yahoo, and spiders each concept (category) while keeping track of subconcepts. Yahoo has a section called *Site Listings* under specific concepts. If the agent identifies a *Site Listings* section under a concept, this set of documents becomes the collection for that specific category. In other words, these documents make up the collection of indexed documents for a given node in the concept hierarchy. A maximum depth level can be provided for the spidering agent if the intent is to represent only a portion of the Yahoo hierarchy.

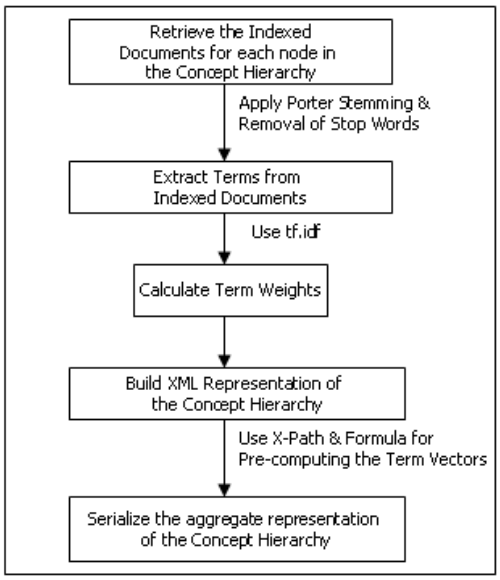

Figure 4: Concept Hierarchy Application work flow

As the agent is performing the spidering task, the information that is collected is added to a SQL Server database. A parent-child relationship is maintained by using document IDs in the table in order to keep track of subconcepts and documents that belong to a specific concept. The title and the content of documents are extracted from the HTML.

3.2 Concept Hierarchy Application

The work flow for the *Concept Hierarchy Application* can be seen in Figure 4. The first step we can perform in this application is to populate terms from each document into a table in our database. The entire content of the document is used to extract the words. A stop list is used to remove high frequency words from the content. Porter stemming [13] is utilized to reduce words to their stems. The next step is to calculate the term weights. For computing term weights extracted from content, a standard function of the term frequency and inverse document frequency (tf.idf) is employed [15; 8].

Once the term weights are calculated, an XML document is created. This XML docu-

ment represents the entire concept hierarchy. It contains each node in the hierarchy, the documents that are indexed under each node, and the terms that belong to these documents along with the term weights. Since the entire concept hierarchy is represented in the XML document, we no longer have any dependencies on the SQL database. The XML document acts as an in memory database. This significantly improves performance for data retrieval. In addition, the XML document allows us to store the information in a hierarchical format.

The final step is to compute the term vectors for the aggregate representation of the concept hierarchy. A term vector is calculated for each indexed document under a concept. The term vectors for the 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.

Once all the computations are completed, the aggregate form of the concept hierarchy is stored as a serialized object. Serialization facilitates the persistence of an object by transforming its data members into a single stream of XML. Figure 5 displays the format of the XML document for the aggregate representation of the concept hierarchy. Only a few sample terms have been included in the figure. Essentially, this XML file is the only external source our system accesses to retrieve the concept hierarchy information. This allows us to conveniently switch to different concept hierarchies when needed. Switching to a new concept hierarchy is a matter of replacing the XML file which contains the serialized concept hierarchy.

4 Online Component

The online component handles query enhancement. ARCH is a Web-based application. 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 either displaying an appropriate portion of the hierarchy or the entire concept hierarchy. The user interface is displayed in Figure 6. If the user enters a list of keywords in the search box and clicks on the arrow, the system matches the term vectors representing each node in the 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. Once the relevant portions of the hierarchy are displayed, 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.

```
<Concept>
  <Name>Languages</Name>
   <ConceptTermList>
     <Term>
        <Weight>1</Weight>
           <Name>languag</Name>
     </Term>
     <Term>
        <Weight>0.452</Weight>
           <Name>python</Name>
     </Term>
     <Term>
       <Weight>0.38</Weight>
           <Name>program</Name>
     </Term>
   </ConceptTermList >
     <SubConceptList>
       <Concept>
         <Name>Python</Name>
         <ConceptTermList>
           <Term>
             <Weight>1</Weight>
                <Name>python</Name>
           </Term>
         </ConceptTermList >
         <SubConceptList/>
       </Concept>
     </SubConceptList>
</Concept>
```

Figure 5: The aggregate representation of the concept hierarchy in XML format

4.1 Enhanced Query Generation

Although other effective query expansion methods such as multiplicative adaptive query expansion [5] exist, in our system we employ Rocchio's method [14] for relevance feedback for the generation of the enhanced query. The pre-computed term vectors associated with each node in the hierarchy are used to enhance the original query. The formula for the refined query is 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

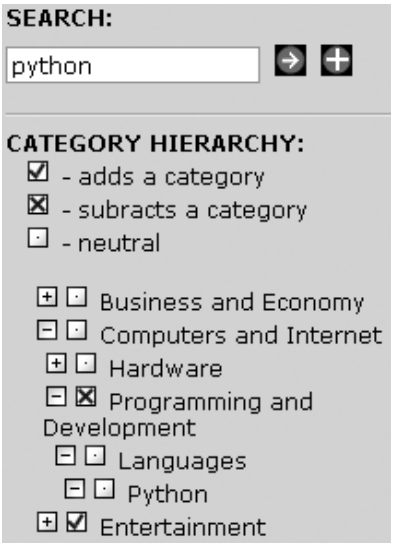

Figure 6: Concept Hierarchy User Interface

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. The condition for these tuning parameters is $\alpha + \beta + \gamma = 1$.

The goal of the system is to retrieve more valuable search results by using the enhanced query rather than the original keyword query.

5 Experiments with ARCH

In order to evaluate our system, we compiled a list of keyword queries. The first set of keyword queries contained only one term such as *python*, *hardware*, and *bat*. The second set of queries contained two terms such as *python snake*, *computer hardware*, and *bat mammal*. We collected a number of signal documents and a number of noise documents to construct our document collection for testing the system. For example, if the intent for the search is to find documents about a *python snake*, the signal documents would contain the *snake* sense of the word *python*, while noise documents would contain the *programming language* or *entertainment* senses of *python*.

We created an index for the signal and noise documents. A term frequency and inverse document frequency (tf.idf) weight was computed for each term in the document collection using the global list of terms. Note that the global term list was created using the documents that were indexed under the various nodes in the concept hierarchy.

We had the system 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. In the case of enhanced query search, we used the query that was generated by ARCH. The search results were retrieved from the signal and noise document collection by using a cosine similarity measure for matching.

As an example, consider the scenario in which the user starts with a single keyword query *python*, using the Yahoo hierarchy. The system will respond to the user's query by displaying the relevant portions of the hierarchy including the parents, children, and siblings of the nodes corresponding to the initial keyword query. The user can now interact with the concept hierarchy by selecting and/or deselecting various nodes.

In this case, the user's ambiguous keyword causes the system to display several different portions of the hierarchy. Let's assume the user is interested in finding information about python as a snake. Therefore, the user selects the concept *Pythons* under *Reptiles and Amphibians* in the hierarchy. Figure 7 displays the portions of the Yahoo hierarchy that corresponds to this scenario. In this case, the user is definitely not interested in gathering information about programming languages. Therefore, the user deselects the node *Programming and Development*.

The term vector which represents the *Programming and Development* node is an average of the term vectors for the documents that are indexed under this node and the term vector representing the node *Languages*, which is a subconcept of *Programming and Development*. Similarly, the term vector, which represents the node *Languages* is computed from a combination of the documents indexed under this node

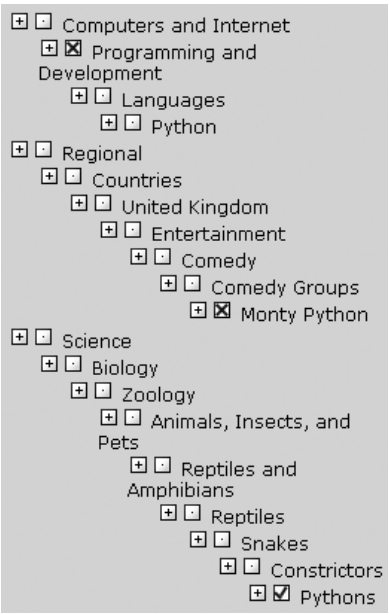

Figure 7: Portions of the Yahoo hierarchy corresponding to the query *python*. The user has selected the node *Pythons* and deselected the nodes *Programming and Development* and *Monty Python*

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 8: Enhanced Query for the keyword *python* based on the selected and deselected nodes in the hierarchy

as well as the term vector representing its subconcept *Python*. The user is also not interested in finding out about the comedy group *Monty Python*, thus deselects *Monty Python*.

Using the selected and deselected nodes, the system generates an enhanced query using the relevance feedback formula which was described before. The partial term vector for the enhanced query for this scenario is displayed in Figure 8. Based on the user's interaction with the concept hierarchy, the enhanced query that is generated accurately represents the user's intent for the initial query.

In order to compare the simple query search results with the enhanced query search results, we created precision and recall graphs for each of our keyword queries. As displayed in Figure 9, we have seen significant improvement in precision when using the enhanced query for searching. As we can see in Figure 10, recall has also improved. More information on our evaluation results is available online at http://www.ahusieg.com/arch.

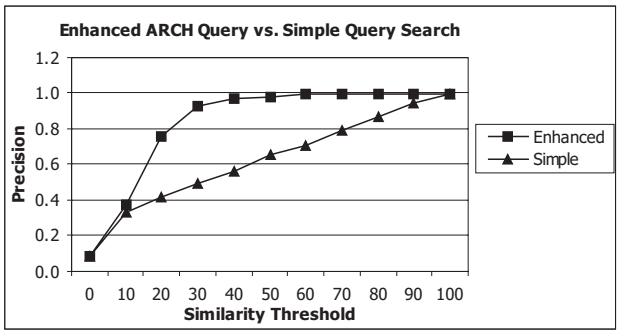

Figure 9: Average Precision for Enhanced Query versus Simple Query Search using two keywords

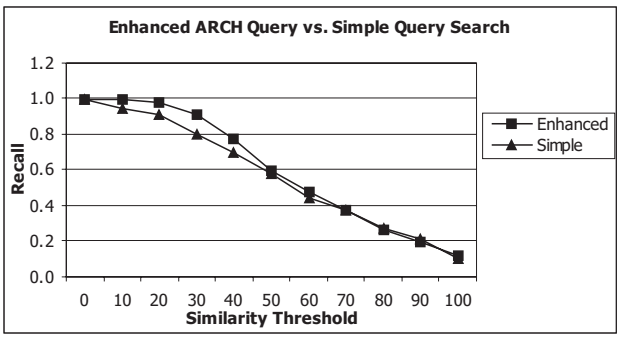

Figure 10: Average Recall for Enhanced Query versus Simple Query Search using two keywords

6 Conclusions and Outlook

We have presented the implementation of query enhancement in ARCH based on the user's interaction with a concept hierarchy. Preliminary experiments have shown that the system 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, we have better recall in the search results since additional query terms retrieve documents that would not be

retrieved by using only the original keyword query.

Our future work will 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 the 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] 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] Z. Chen. Multiplicative adaptive algorithms for user preference retrieval. *Lecture Notes in Computer Science*, 2108, 2001.

[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. 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.