Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 1999 Proceedings

International Conference on Information Systems (ICIS)

December 1999

Retriever: An Agent for Intelligent Information Recovery

D. Fragoudis
University of Patras and ZEUS Consulting

S. Likothanassis
University of Patras and Computer Technology Institute

RETRIEVER: AN AGENT FOR INTELLIGENT INFORMATION RECOVERY[1]

D. Fragoudis
University of Patras and
ZEUS Consulting S.A.
Greece

S. D. Likothanassis
University of Patras and
Computer Technology Institute
Greece

Abstract

With the exponential growth of the Internet and the volume of information published over it, searching for information of interest has become a very difficult and time-consuming task. Search engines, although they have been developed to help people cope with all of this information, have many shortcomings. In this paper, we present our ongoing work on **Retriever**, an autonomous agent that executes user-queries and returns high quality results to the user. **Retriever** utilizes existing search engines to obtain the starting points for its subsequent autonomous exploration of the Web. It then conducts a self-training process, in order to learn the query domain and increase its efficiency. When the query domain is learned, the agent expands the original query, reforms its search strategy, and goes out looking for the documents requested by the user.

1. INTRODUCTION

The exponential growth of the Internet in the last few years has resulted in a corresponding exponential growth of the amount of information published over it. Searching for information of interest is in practice impossible if its source is not known in advance. Thus, search engines have been developed to help the user locate desired information. Unfortunately, search engines suffer from three major shortcomings. They return too much data, they return too much irrelevant data, and they require the user to remain at the desktop "waiting for the next ten documents to evaluate." To compensate for these shortcomings, intelligent agents have been developed for searching and filtering information on the Web. These information seeking assistants promise to help people in their "battle" against information overload. WebMate (Chen and Sycara 1998) is an agent that compiles a personal newspaper from a set of monitored URLs or queries search engines for interesting documents. The agent may recognize up to a preset number of distinct domains of interest. Amalthea (Moukas 1997) is a co-evolution model of information filtering and information discovery agents that queries existing search engines and monitors specified Web sites for interesting documents. The ecosystem receives relevance feedback from the user and adapts its behavior to his/her changing interests. Letizia (Lieberman 1995) and WebWatcher (Armstrong et al. 1995) are two agents designed to assist and provide personalized recommendations to the user while browsing the Web by performing a breadth-first search on the links ahead. LIRA (Balabanovic and Shoham 1995) is an agent that autonomously searches the Web, looking for interesting documents to present to its user. Relevance feedback is used to help the system adapt to the user's interests. WebACE (Han et al. 1998) is a trainable agent that incorporates clustering techniques to construct clusters of the training documents. It then queries search engines for documents that belong to one or more of these clusters. ARACHNID (Menczer and Belew 1998) is a multi-agent system that autonomously searches the Web for

[1]This work was supported by the Greek GSRT and ZEUS Consulting SA under Grant 97 UP 188.

interesting documents. The system tries to self-adapt to the information environment and may operate without any human intervention at all. A new approach, **Retriever**, that performs autonomous exploration of the Web, with the starting points obtained by popular search engines, is discussed in this work. The remainder of this document is organized as follows: In section 2, the architecture of the system is presented. Section 3 deals with the training phase, while section 4 deals with the search phase. Finally, section 5 summarizes the conclusions and reports the future work.

2. ARCHITECTURE

In order to design **Retriever**, we posed three specifications: fast information retrieval, precision, and effectiveness. When a user needs some information, usually there is no time and no material to train an agent. Furthermore, the overwhelming majority of these searches do not express a persistent interest. When sufficient information is collected, the search process is terminated and no further search is performed unless a new need arises. All of the architectures described above consider a user with persistent interest in one or more domains. From this perspective, **Retriever** is novel agent architecture since it radically differs from all of the previous paradigms. **Retriever** is an agent that tries to maximize its efficiency in one single search. It does not depend on the user to assess the relevance of the returned documents in order to make better future searches. Document assessment is usually a time-consuming and frustrating task and presupposes an extended period of search. User input is restricted to the query itself and the agent operates in two phases, as shown in Figure 1.

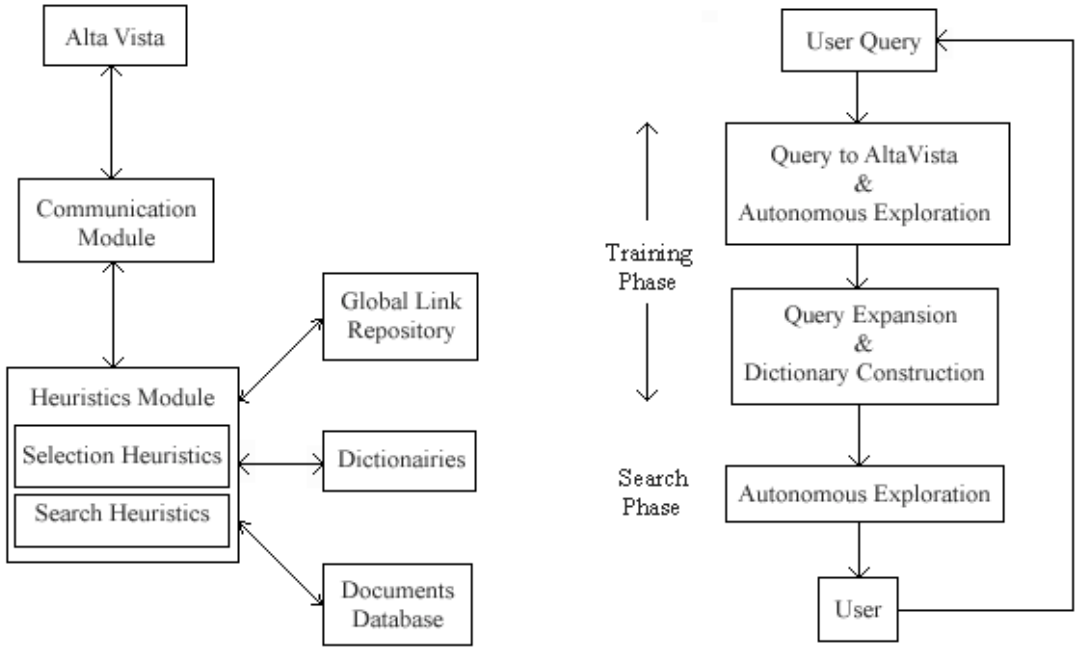

Figure 1. Retriever's Architecture and Flow Chart

The first phase is a self-training phase that will help **Retriever** to learn the query domain. By query domain, we mean the documents that are relevant to the query in terms of some document similarity measure. Research in the field of information retrieval has shown that learning queries in the query domain leads to increased efficiency in document assessment (Buckley, Mitra and Singhal 1997). In this phase, **Retriever** collects documents that are similar to the user query. When a sufficient number of relevant documents is retrieved, this process is terminated and the retrieved documents are analyzed. Document analysis is performed to allow the agent learn the query domain and increase its efficiency in the search phase. The search phase is the phase where the actual search is conducted. Documents retrieved as relevant in this phase are presented to the user. In the following sections, we will examine in further detail the two operation phases: the training phase and the search phase. We will see how a search is conducted (search heuristics) and what criteria are used for measuring document relevance (selection heuristics).

3. THE TRAINING PHASE

The training phase is the phase where **Retriever** learns the query domain. As described above, in this phase the agent collects documents relevant to the original user query and when a sufficient number of relevant documents are collected a document analysis is performed. This number is currently user defined, but it is within the scope of our research to have the agent determine it. Next, we analyze the search and selection heuristics for this phase as well as the document analysis that is performed at the end of the phase.

3.1 Search Heuristic

The term search heuristic is used to describe how the agent chooses, at any time, the next link to follow. At the beginning, after a new query is submitted, the agent needs some starting points for its autonomous exploration. These starting points are obtained as a result of a query to popular search engines. In the current implementation, **Retriever** is programmed to query only AltaVista. The query results are evaluated for relevance to the query and, after a sufficient number of relevant links is collected, the agent starts its autonomous exploration. In any case, if a link is found to be promising, it is added to the global link repository. The global link repository is a warehouse that contains all the relevant links that have not already been visited. When **Retriever** has to choose which link to follow, it always chooses the most promising one within this link repository. The link is then removed from the repository. After the new document is retrieved, it is processed, the links within are evaluated, relevant ones are added to the repository and so on. It could be observed that in this way the search is not continuous, in the sense that two subsequently inspected documents are usually not interlinked. However, the system always makes the best possible choice for continuing its journey and fulfilling its goal. It is also very difficult to be trapped at dead-end links since there are always alternatives. We call this search heuristic "global best-first," in conjunction with the pure best-first heuristic (http://www.cee.hw.ac.uk/~alison/ai3notes/subsection2_6_2_3_2.htlm) where all of the encountered links are stored in the link-repository. Also, the agent cannot avoid following dead and/or outdated links, since there is no way of determining their status. However, since a link is followed only once, such links will not be revisited.

3.2 Selection Heuristic

By the term selection heuristic, we mean how documents and links are represented and how similarity among them is measured. In **Retriever**, both documents and links are represented as keyword vectors. When the agent decides to follow a link, the communication module undertakes the task of fetching the HTML page. The retrieved document is then subjected to a series of steps that will construct its keyword vector. These steps are:

1. HTML tags and words that belong to a stop-word list are removed. The stop-word list contains frequent words that have proved to contribute very little in information retrieval (Salton and McGill 1983).

2. The remaining words are stemmed. We utilize Porter's (1980) stemming algorithm as implemented in Frakes and Baeza-Yates (1992) with a slight modification since AltaVista cannot accept stems shorter than three letters in length. Thus the stemming algorithm stops whenever a suffix replacement results a stem shorter than three letters. This way, the word "agents" is not reduced to "ag," but instead to "agent."

3. Stems that result from step 2 and belong to the training word dictionary are weighted in order to form the document weighted keyword vector. The formula is: $w_i = \frac{tf_i}{\sqrt{\sum_{j \in D} (tf_j)^2}}$. This is a simple formula that weights keywords (w_i) based only on their text frequency (tf_i). However, since the training word dictionary contains only the query terms and human queries almost always contain very discriminative keywords, this scheme is very efficient in capturing documents within the query domain. The same formula is used to construct the link keyword vectors, since links are treated exactly like documents.

The measure that is used to calculate similarity between documents or links and the user query is the cosine similarity measure: $Sim(V_j, V_k) = \frac{V_j \cdot V_k}{|V_j| \times |V_k|}$. V_j and V_k are the vectors being matched. A document or link is considered as relevant if its estimated relevance is above a given threshold.

3.3 Document Analysis

When a sufficient number of such relevant documents is retrieved, the search stops and the agent enters a document analysis phase that will help it learn the query domain. Document analysis is performed by constructing two distinct dictionaries, by expanding the original user query and by constructing a new separate query vector for matching documents within the query domain. The first dictionary is a standard IR dictionary that contains the document frequencies of the encountered words and it will be used to calculate the keyword weight in the search weighting formula. This formula will weight documents in the main search phase and it will be described later. In our first experiments, the IR dictionary contained about 3,500 terms. The most frequent terms of this dictionary along with the original query terms will form the query domain dictionary. This dictionary will replace the original "query-terms" dictionary that has been used to weight documents and links in the training phase. The original user query is now expanded with terms from the query domain dictionary and the new query vector, the query domain vector $Q_{QD,}$ is formed as the normalized vector of the document frequencies of the dictionary terms: $w_i = \frac{df_i}{\sqrt{\sum_{j \in D} (df_j)^2}}$ (w_i is the weight of the term i and df_i its document frequency within the dictionary D).

A new query vector is also constructed. The new vector, the relevance vector Q_R, will be the query vector for determining whether a document that belongs to the query domain is indeed relevant to the user query. The relevance vector is constructed as the normalized mean vector of the top ranked documents of the training phase: $w_i = \frac{\overline{w_j}}{\sqrt{\sum_{j \in D} (\overline{w_j})^2}}$. These vectors have first been re-weighted under the search weighting formula.

4. THE SEARCH PHASE

When the training phase is over and the document analysis is done, **Retriever** begins searching for interesting documents. In the following paragraphs, we will present how search and selection is conducted in this phase.

4.1 Selection Heuristic

Selection in the search phase is almost the same as in the training phase. Inspected documents are again removed from HTML tags and frequent words stemmed and weighted. Weighting is based on the training weighting formula and the expanded query domain dictionary. The resulting weighted-keyword vectors are matched against the Q_{QD} query vector and if relevance for a document is estimated above a user-defined threshold, the document belongs to the query domain. Documents in the query domain are re-reduced to keyword vectors. The weighting scheme is given by the formula:

$$u_i = \frac{\left(0.5 + 0.5\frac{tfi}{tf\ \max}\right)\left(\log\frac{1}{dfi}\right)}{\sqrt{\sum_{j \in D}\left(\left(0.5 + 0.5\frac{tfi}{tf\ \max}\right)^2\left(\log\frac{2}{dfj}\right)_2\right)}}$$

The search weighting formula is a sophisticated TF.IDF scheme that normalizes for unit length (Salton and Buckley 1987; Salton and McGill 1983). In this formula tf_i is the text frequency of the term i, df_i its document frequency based on the IR dictionary, and tf_{max} the maximum term frequency within the inspected document. The resulting vector is matched against the Q_R query vector and if again similarity is found above a given threshold, the document is presented to the user. Links are not processed in such degree. If a link is found to belong in the query domain, it is also considered relevant and added to the link repository.

4.2 Search Heuristic

With a link repository full of promising links there is no need to conduct a starting-points collection phase this time. However, the first step in the search phase is to re-examine the documents that constitute the query domain. We saw in the previous paragraph how this is accomplished. Documents in the query domain that are also found relevant to the expanded query are presented to the user. When all of the collected documents are processed, **Retriever** again starts the autonomous exploration. The search heuristic is exactly the same as in the training phase. The same link repository is used as link selection source and the global best-first heuristic again determines the next link to be followed. If at any time the link repository becomes empty, AltaVista is requeried, this time with the expanded query.

5. CONCLUSIONS AND FUTURE WORK

In this work, we have presented **Retriever**, an autonomous agent that utilizes existing search engines to conduct a self-training process, in order to learn the query domain and increase its efficiency. We have tested the system's performance in the training phase. Retriever has been proven to perform very well even in the training phase. Furthermore, the quality of the documents that constitute the query domain, for various user queries, is very high. In our work-in-progress, we have continued with the evaluation by testing the system in the search phase. Some preliminary results, reported in Fragoudis and Likothanassis (1999) showed that when the evaluation is over, the result will be an intelligent information-gathering agent that will provide its users with a high quality of search results. In our future work, we will deal with the performance evaluation of **Retriever**, compared to other search machines.

6. REFERENCES

Armstrong R.; Freitag D.; Joachims T.; and Mitchell T. "WebWatcher: A Learning Apprentice for the World Wide Web," *Proceedings of American Association for Artificial Intelligence Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources*, Stanford, California, 1995

Buckley, C.; Mitra, M.; and Singhal, A. "Learning Routing Queries in a Query Zonem," *Proceedings of the Twentieth International ACM SIGIR Conference on Tesearch and Development in Information Retrieval*, Philadelphia, Pennsylvania, 1997.

Balabanovic, M., and Shoham, Y. "Learning Information Retrieval Agents: Experiments with Automated Web Browsing," *Proceedings of American Association for Artificial Intelligence Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources*, Stanford, California, 1995

Chen, L., and Sycara, K. "WebMate: A Personal Agent for Browsing and Searching," *Autonomous Agents'98 Conference*, Minneapolis, Minnesota, May 1998.

Fragoudis, D., and Likothanassis, S. "Retriever: A Self-Training Agent for Intelligent Information Discovery,"*International Conference on Intelligent Information Systems '99*, Washington, DC, November 1999.

Frakes, W. B., and Baeza-Yates, R. (eds.). *Information Retrieval Data Structures and Algorithms*, Englewood Cliffs, NJ: Prentice Hall, Inc., 1992.

Han, S.; Boley, D.; Gini, M.; Gross, R.; Hastings, K.; Karypis, G.; Kumar, V.; Mobasher, B.; and Moore. J. "WebACE: A Web Agent for Document Categorization and Exploration," *Autonomous Agents'98 Conference*, Minneapolis, Minnesota, May 1998.

Lieberman, H. "Letizia: An Agent That Assists Web Browsing," *International Joint Conference on Artificial Intelligence*, Montreal, August 1995.

Moukas, A. "Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem", *Applied Artificial Intelligence: An International Journal* (11:5), 1997, pp. 437-457.

Menczer, F., and Belew, R. K. "Adaptive Information Agents in Distributed Textual Environments," *Autonomous Agents'98 Conference*, Minneapolis, Minnesota, May 1998.

Porter, M. "An Algorithm for Suffix Stripping," *Program* (14:3), 1980.

Salton, G., and Buckley, C. "Term Weighting Approaches in Automatic Text Retrieval," Technical Report 87-881, Department of Computer Science, Cornell University, 1987.

Salton, G., and McGill, M. J. *Introduction to Modern Information Retrieval*, New York: McGraw-Hill, Inc., 1983