INTELLIGENT RETRIEVAL
Open Journal

Review

Retriever: An Agent for Intelligent Information Recovery

D. Fragoudis[1*]; S. D. Likothanassis[2]

[1]*University of Patras and ZEUS Consulting S.A., Greece*
[2]*University of Patras and Computer Technology Institute, Greece*

[*]***Corresponding author***
D. Fragoudis
University of Patras and ZEUS Consulting S.A., Greece

Article information
Received: August 2nd, 2023; **Revised:** November 1st, 2023; **Accepted:** November 23rd, 2023; **Published:** December 20th, 2023

Cite this article
Fragoudis D, Likothanassis SD. Retriever: An agent for intelligent information recovery. 2023; 1(1). doi: https://doi.org/10.70705/ppp.ir.2023.v01.i01.pp50-53

ABSTRACT

Since the Internet and the amount of information published on it have grown at an exponential rate, finding relevant information has become a laborious and time-consuming process. While search engines were created to assist individuals in managing this overwhelming amount of information, they do have some drawbacks. Here we detail our current efforts towards developing Retriever, a self-sufficient agent that can process user requests and provide them with accurate responses. In order to begin its independent investigation of the Web, Retriever makes use of preexisting search engines to get beginning points. Afterwards, it undergoes a self-training procedure to gain knowledge of the query domain and enhance its performance. Upon learning the query domain, the agent refines its search approach, broadens its initial inquiry, and sets out to find the user-requested content.

Keywords
Internet; Intelligent agents; Retriever.

INTRODUCTION

There has been an exponential increase in the quantity of content published on the Internet in recent years, directly proportional to the exponential expansion of the Internet itself. Without knowing where the information came from, it is almost difficult to search for relevant results. For this reason, search engines have emerged to aid users in locating specific data. There are three big problems with search engines. Not only do they provide an overwhelming amount of data, but they also overwhelm the user with unnecessary material and force them to stay at the desktop while "the next ten documents evaluate." Intelligent agents for Web searching and filtering have been created to make up for these deficiencies. The makers of these information-seeking helpers claim they may aid users in the "battle" against data overload. A personal newspaper may be created using the agent WebMate (Chen and Sycara 1998) by monitoring a collection of URLs or by querying search engines for interesting items. The agent has the capability to identify a maximum of a predetermined number of specific domains. A co-evolution model of information discovery and information filtering agents, Amalthea (Moukas, 1997) checks preexisting search engines and certain websites for relevant articles. The ecosystem learns the user's preferences via relevant feedback and adjusts its actions accordingly. As the user navigates the web, agents like Letizia (Lieberman, 1995) and WebWatcher (Armstrong et al., 1995) scan the links ahead of them in a broad-first search to deliver individualized suggestions. Autonomously searching the Web for interesting content to provide to its user, LIRA (Balabanovic and Shoham 1995) is an agent. In order for the system to adjust to the user's preferences, relevance feedback is used. An agent that can be taught new skills, WebACE (Han et al., 1998) uses clustering methods to group similar training papers together. After then, it uses search engines to find papers that are part of these groups. ARACHNID is a multi-agent system that searches the web autonomously for engaging papers. There may be zero need for human interaction as the system strives to self-adapt to its data surroundings. This study discusses a novel strategy called Retriever that uses the beginning points collected by prominent search engines to autonomously explore the Web. What follows is an outline of the rest of the document: Section 2 presents the system's architecture. The search phase is covered in Section 4, whereas Section 3 is devoted to the training phase. Section 5 concludes by summarizing the findings and reporting on the next 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.

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."

3.3 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 QQD, is formed

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 QQD 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:

Fragoudis and Likothanassis

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 tfi is the text frequency of the term i, dfi its document frequency based on the IR dictionary, and tfmax the maximum term frequency within the inspected document. The resulting vector is matched against the QR 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

No starting-points gathering step is required at this time since the link repository is full with intriguing links. But before you even think about doing a search, you should review the documents that make up the query domain. In the preceding paragraph, we saw the process of doing this. As the user expands their query, relevant documents within the query domain are shown to them. Retriever resumes its independent exploration after all of the papers have been processed. Just as during training, the search heuristic remains unchanged. Once again, the global best-first heuristic is used to ascertain the subsequent link to be followed, with the same link repository serving as the link selection source. Whenever the link repository is depleted, AltaVista is requeried using the enlarged query.

5. CONCLUSIONS AND FUTURE WORK

Here we introduce Retriever, a self-training autonomous agent that learns the query domain and improves its efficiency by using current search engines. During training, we evaluated the system's functionality. Even during training, Retriever has shown to be quite effective. In addition, the query domain, which includes a wide range of user inquiries, has very high-quality content. As part of our ongoing effort, we have tested the system throughout the search phase to continue with the review. After the assessment is complete, the intelligent information-gathering agent will provide its consumers high-quality search results, according to some early findings described in Fragoudis and Likothanassis (1999). We want to evaluate Retriever's performance in comparison to other search engines in upcoming work.

REFERENCES

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

"Learning Routing Queries in a Query Zonem," presented at the 1997 Twentyieth International ACM SIGIR Conference on Research and Development in Information Retrieval in Philadelphia, Pennsylvania, by Buckley, Mitra, and Singhal.

Presented at the 1995 American Association for Artificial Intelligence Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources in Stanford, California, "Learning Information Retrieval Agents: Experiments with Automated Web Browsing" was written by Balabanovic and Yusof Shoham.

Presented at the Autonomous Agents'98 Conference in Minneapolis, Minnesota in May 1998, "WebMate: A Personal Agent for Browsing and Searching" was written by Chen and Sycara.

In their 1999 paper "Retriever: A Self-Training Agent for Intelligent Information Discovery," Fragoudis and Likothanassis presented their work at the International Conference on Intelligent Information Systems in Washington, DC.

This work is edited by W. B. Frakes and R. Baeza-Yates. Data Structures and Algorithms for Information Retrieval, 2nd ed., Prentice Hall, Inc., 1992, Englewood Cliffs, NJ.

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

"Letizia: An Agent That Assists Web Browsing," given by Lieberman in August 1995 at the International Joint Conference on Artificial Intelligence in Montreal.

"Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem" by Moukas, A. was published in Applied Artificial Intelligence: An International Journal (11:5) in 1997 and can be found on pages 437-457.

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

According to Porter (1980), "An Algorithm for Suffix Stripping" was published in Program (14:3).

"Term Weighting Approaches in Automatic Text Retrieval"

(Technical Report 87-881), Cornell University Department of Computer Science, 1987. Salton, G., and Buckley, C.

In 1983, Salton and McGill published an introduction to modern information retrieval. The book was published in New York by McGraw-Hill, Inc.