Searching the World-Wide-Web with Learning Agents

Ioan Alfred Letia
Technical University of Cluj-Napoca
Baritiu 28, RO-400391 Cluj-Napoca
Romania
letia@cs.utcluj.ro

Raluca Vartic
Technical University of Cluj-Napoca
Baritiu 28, RO-400391 Cluj-Napoca
Romania
rvartic@cs-gw.utcluj.ro

Anca Mărginean
Technical University of Cluj-Napoca
Baritiu 28, RO-400391 Cluj-Napoca
Romania
anca@cs-gw.utcluj.ro

Mircea Tesa
Technical University of Cluj-Napoca
Baritiu 28, RO-400391 Cluj-Napoca
Romania
tesa@cs-gw.utcluj.ro

***Abstract*—Present day search engines are far from perfect because they retrieve a great number of pages from different sources (reliable or not), and because they sometimes return unexpected results. The search results would improve if the search engine knew a set of reliable sites where we are expecting to find good results. We propose a server-side multi agent system that addresses search problems like source reliability, time-constrains and reducing the complexity. The agents learn degrees of confidence for some sites of interest, starting from a limited number of domains known as trustworthy and updating them from future search results. These degrees of confidence are used to select where to search next, considering that we are expecting to find more promising results on the sites with a higher confidence degree. Using input from the user, these degrees are updated with every new search.**

Keywords: agents, web search, confidence, preferences

I. Introduction

The search process can be analyzed from different perspectives: what to look for, where to look, how to evaluate the likelihood that a page is a good result.

Searching the web for relevant and trustworthy information is not an easy task. The number of pages retrieved is very large, they are not always what the user was looking for, and they are often unreliable. Searching only by keywords may not always be suitable for the huge amount of information on the Web. Research in the area is trying to improve the search process so that a small number of good results from reliable sources is returned.

Because of the quantity of information available on the Internet, finding something specific can be quite a headache. This was the motivation behind the numerous search engines used today. However, searching through the entire web can be a difficult task. Most of the problems encountered are related to complexity and reliability. Therefore, the results returned by the search engines are sometimes unsatisfactory, due to both their big number and their unexpected content. Also, one rarely really wants to search through all the various topics debated on the web. These are the main reasons extra processing of the search results is necessary for better information retrieval performance. Possible approaches include: *Web Mining* - extracting useful information from the set containing all the pages on the Web; *Classification of Web Pages* - grouping similar pages is done either manually or automatically; *Building a Knowledge Base* - mirroring the contents of the World Wide Web; *Using Learning Algorithms*; *The Semantic Web* - an extension of the current Web, with support for databases in machine-readable form; *Relying on Trust and Reputation*; *Creating User Profiles.*

A significant amount of effort is being deployed to enable more effective retrieval of Web information, as well as new uses of the Web to support knowledge-based inference and problem solving [1].

The need to evaluate user reactions when using autonomous agents to assist with information centric tasks on the Semantic Web gave birth to *Nuin*, an agent platform that was designed for practical development of agents in Semantic Web applications, based around belief-desire-intention (BDI) principles [2].

One first approach to make the retrieval of Web information more efficient is to rank the pages based on their relevance to a specific user, using contextual information. The personalization of the search engine can be made using a short-time user profile constructed by a client-side application that captures current user activities. This approach has the advantage that the user is not directly asked for information, but the contextual information allows ranking of results. [3]

Another approach is to learn hypertext classifiers by combining a statistical text-learning method with a relational rule learner. The statistical component allows text characterization in terms of word frequencies, and the relational component describes how neighboring documents are related to each other by the hyper-

links that connect them [4].

Research in Web information retrieval is already exploiting the use of agent. An example to support such agent-based applications is *Nuin*, an agent platform that was designed for practical development of agents in Semantic Web applications, based around belief-desire-intention (BDI) principles [2].

Our multi-agent system is tackling the problem of source reliability by selecting a number of sites we know to be trustworthy. As new queries are being addressed, we are learning a confidence knowledge base, using the user's evaluation of the results.

II. Theoretical Approach

There are 3 stages in the execution of our application: *training, learning, exploring.* In the training stage, the user rewards the pages that meet his/her interests, and the system learns the common features of these pages. Then, in the learning stage, the system uses the Google search engine to search the main sites set for results.The resamblance of these results with the pages from training set is computed in a given limited time and used to rank the results. The user offers rewards for the results he/she considers good. Instead of the learning stage, the user can point the application to perform exploring. The sites with high confidence values will be searched first and the search results will be displayed in the order they are retrieved. The user can offer rewards for these results too. The rewards given by the user in the second or third stage are used to update the confidence values for sites. The algorithm used for these updates makes use of reinforcement learning and will be described in section *2).*

A. Hypertext Processing

The text of a Web page is an essential resource for retrieving information. A hypertext can be viewed from different perspectives: *the Gross Text; the Annotated Text; the Language; the Knowledge Contained in the Text; the Hypertext Graph Structure; the Displayed Text.* There are many ways to retrieve hypertext information. Work in the area includes: *formal methods* - exploiting the content and the graph structure, link analysis, *algorithms and heuristics for searching hypertext, extraction and exploitation of metadata.* We are tackling the task of information retrieval by parsing web pages and remembering information about the words and the links in the hypertext. The resulted bag of words contains unstructured information on the most relevant words and the number of times they occur in the text. The relevant words are selected by eliminating common words, words containing digits, words shorter than 3 letters or longer than 19. This information is stored on files, one for each page.

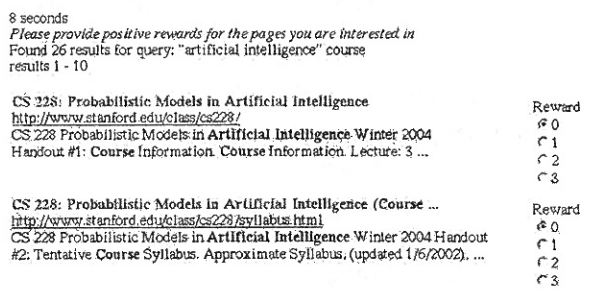

Fig. 1. Providing rewards

B. Learning Confidence Values for Sites

As one can see in figures 1 and 2, the user is the one providing rewards for the search results. The system uses this information to build an ordering of the selected trustworthy sites. We are considering that certain sites are more likely to contain good results on a global basis. The system assigns numerical

quantifiers to measure the likelihood that searching a specific site would yield good results. These results are updated accordingly with every new search.

For example, if the user expresses interest in the courses on a certain site, there is a high probability that the user will also find research topics and lab activities on the same site interesting.

1) *Action Selection Policy:* As we need to retrieve information in a given time, we must ensure the most promising results are processed first. We define an action as pointing the system to process the next result from a certain site. Thus, the number of actions is equal to the number of sites we can conduct our search on, namely the number of sites in our set of selected sites. We are using the confidence values maintained for these sites to select the next action. Applying a greedy policy, the site with the greatest confidence value is selected. After processing, the temporary confidence value for the given site will be decreased by a factor ϵ.

$$C(site) = \epsilon * C(site) \quad (1)$$

To be noted that these updates are performed on temporary variables, they have meaning on the current search only, and they have no effect on the overall confidence values.

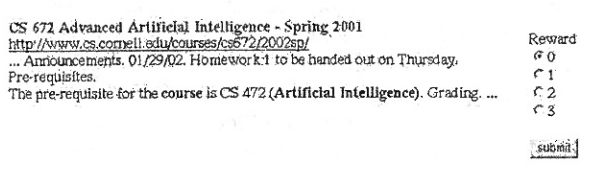

Fig. 2. Submitting rewards

2) *Applying Reinforcement Learning:* We have used a simplified reinforcement learning method. We are interested in the value we get from performing a certain action. Unlike the reinforcement learning model, we are considering that actions are not taking us from one state to another. We are using the following formula to update the value of performing a certain action:

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * (r + \gamma * max(Q(s,a'))) \quad (2)$$

The α parameter is the learning rate, which indicates how quickly the system learns, and γ controls the relative importance of future actions. r is the reinforcement signal, $max(Q(s,a'))$ is the maximum value that can be attained by performing an action after a has been executed.

The system can adapt to changes in the environment. Thus, if one site is temporarily unavailable, the function that computes the resemblance between that page and the positive results from the training set will return a low value, and hence the page will be displayed among the last results. This means there is a small probability that the user will reward the page, and consequently the action of searching that specific site will have a decreased value.

C. Agents

The application comprises many different tasks, therefore we designed it as a multi-agent system, with agents specialized for performing simple tasks. At this point, we needed to integrate our agents in a framework, to enable cooperation between them.

The Open Agent Architecture (OAA)[1] is a multi-agent framework that focuses on enabling flexible interactions among a dynamic community of heterogeneous software agents. The key idea in OAA is delegation: instead of each agent hard-coding its interactions (method calls or messages), explaining how and who it will interact with, OAA agents express interactions in terms of needs delegated to a Facilitator agent. A Facilitator agent will coordinate the agent community in achieving the task, providing services such as parallelism, failure handling, and conflict detection that each client agent does not have to worry about itself.

1) *The Multi-Agent System:* In figure 3, one can observe the main system interactions. It can be easily noticed that all requests go through the Facilitator, who than chooses the most appropriate agent to perform the task. The following interactions can be noticed: the Search Agent that queries the Google Agent on a string input, the Url Agent that asks the Parse Agent to process a search result and the Url Agent that asks the Disk Agent to store the results on disk.

In figure 4 we can see a classical sequence of execution events. First, the user is entering the query, and

[1] http://www.ai.sri.com/~oaa

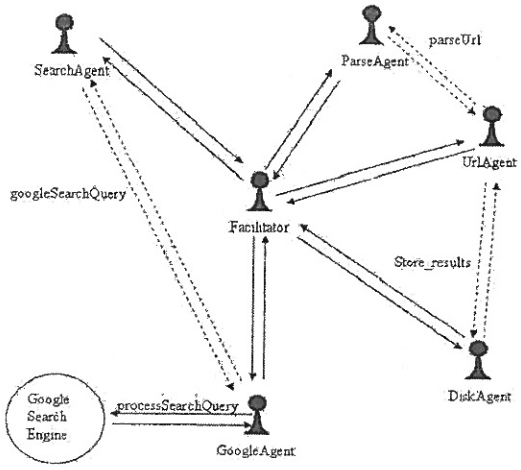

Fig. 3. Overview of system interactions

the search is performed on the sites in the training set. All results are displayed, and the user selects the results he finds satisfactory. The Parse Agent is then parsing these sesected results, and stores information referring to the words and links occurring in the page on a file on disk. Then, the user points the system to start searching on the main set of sites. These search results can be processed. The user specifies the number of minutes he/she is willing to wait, and points the system to start processing. Based on the reputation degrees, the Page Learner agent selects an order to process url-s. When time's up, the processed results are displayed, ordered by how much they resemble the good results found in the training phase. In the final stage, the user evaluates these results. The rewards inputed by the user for the results are used for updating the reputation values.

D. The System as a Web Application

Since the system deals with and uses the web, the decision to design the user interface as a web application came naturally. In the future, we are thinking about including all the necessary features to publish it on the Internet. At this time, we are deploying the application on Apache Tomcat web server. Searching is done simply by typing the search query in a text field and pushing the search button (or simply pushing the enter key).

We are using the following solvable to interact with Google:

```
google_search( +STRING: GoogleLicenceKey,
  +STRING: GoogleSearchString,
  +INTEGER: MaxNoResults,
  -STRING: ResultTitle, -STRING: ResultURL,
  -STRING: ResultSnippet)
```

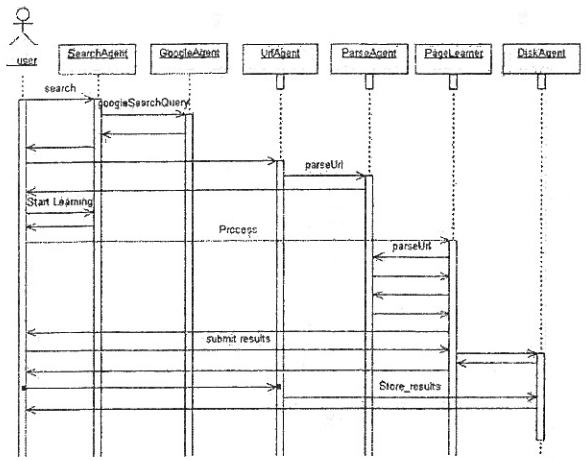

Fig. 4. Sequence diagram

E. The Search Agent

This agent addresses queries to the Google Agent and formats the result. It is used both in the training phase, to call the query on the test sites, as well as in the second phase, when we are searching through a bigger portion of the web. This agent is also responsible for splitting the results in groups of ten and formatting the text in order to be displayed in a friendly manner in the browser.

F. The URL Agent

This agent can be used to: point the Parse Agent to process a URL, ask the Parse Agent for the most frequent words occurring in the pages selected as promising from the training set. point the Disk Agent to store the results on disk and clean the *newLearn* directory, point the Disk Agent to clean the *newLearn* directory, making it ready for a new search

G. The Parse Agent

This agent is used to parse URL-s and store the resulting files, retrieve the set of words occurring in most of the rewarded pages in the training set, get the words from a file resulted from parsing a URL, compute the resemblance between the words in a newly processed URL and the words encountered in the training set: The formula for computing the resemblance between a Web page and the positive results from the training set is given below:

$$resemblance(page, training\_set) = no\_elements(most\_frequent\_words(training\_set) \cap get\_words(page))$$

where resemblance returns the number of words resulted from the intersection of two sets of words, most_frequent_words is applied on a number of sets of words returns a set of words representing the words occurring in most input sets, and get_words is applied on a Web page and returns the most relevant words in the page.

H. The Page Learner Agent

The Page Learner Agent performs the following tasks: it retrieves the confidence values associated to the set of sites selected for processing, points the Parse Agent to parse a URL, points the Parse agent to retrieve the words of a file resulted from processing a URL, point the parse agent to compare the words occurring in a file to the most frequent words occurring in the training set, to see how much they resemble.

I. The Disk Agent

This agent is used to store the search results, and store and retrieve the site confidence values.

III. Experimental Results

We set up an experiment for the query: *"artificial intelligence" course*. We are expecting as results home pages for artificial intelligence related courses at different universities. Notice that the word *course* may have different meanings: university course, course of action, soup as main course.

First, we ran the query directly on Google. There were 656,000 results returned in 0.35s. We analyzed the first 140 results, and found that 65 (46%) were indeed what we were looking for. The positive results came from more than 40 web sites.

If we use the Google agent for the same query, our search will be limited to just 10 results. These results will be the same as the first 10 results displayed by Google. We are in fact extending the Google agent's capabilities by performing one search for one site, and thus retrieving a maximum of 10 results per site, for a total of maximum $10 * NumberOfSites$ results.

Then, we tried the same query on our system. In the training phase, we got 21 good results out of a total of 26 (80%). We offered positive rewards for those 21 results that met our criteria, and started the search on the main set of sites. The search took 30 seconds, and returned 141 results, out of which 60 (42%) were found satisfactory. At this stage, we wanted to process these results, so that the most promising results were displayed first. This being the first learning stage, we considered all sites had equal chances of containing good results. The processing took 15 minutes and the search results were reordered. We found that our expectations were met, and most of the results considered good were repositioned among the first results, as it can be seen in figure 5.

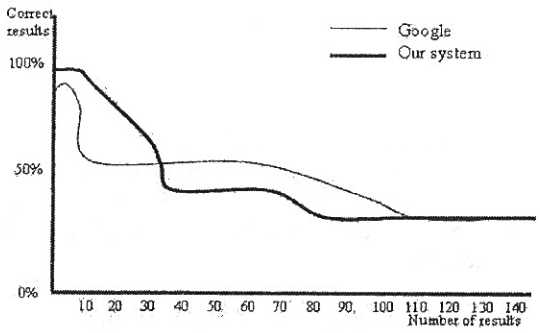

Fig. 5. Processing results

In the table I we have shown the distribution of the significant results in the set of the displayed results produced by our multi-agent system (MAS) and those returned by Google. As one can see, most of the significant results are displayed on the first pages.

Overall, there is no significant difference between the performance of our system and Google. But if we look closely to the first 30 results, we can see that our system has 80% accuracy, compared to Google's 60%.

IV. Conclusions and Future Work

We have outlined a system that proposes extra processing of the search results returned from a limited number of sites, in order to find the pages that best meet the user's interest. We have seen an example where the first results returned by the system are superior to the results returned if we simply ran a query on Google. In addition, we have the certainty that these results come from sites we considered worthy. The drawback that naturally follows is that this system as it is right now can not find new areas of interest. This is something we are hoping to amend in the future.

We are planning to improve these results by improving the comparison function between pages and by selecting optimal values for α and γ in the learning stage. The system will also be extended to include searches in different areas. We are thinking about creating user profiles, allowing different types of users searching for different things. A user would be able to configure the sites he wants searched and the system would be able to notify its users on changes in pages of interest.

Facing searching the WWW should not disregard situated agents [5]. Relevance feedback, which refines the retrieval by utilizing user's feedback history, has been successfully applied to image relevance [6] and we intend to apply it to various representations for concepts. The semantic dimension [7] constitutes another hope for improving significantly the user expectations from search. Hierarchical reinforcement learning using multi-agents is also a promising new line of research [8].

TABLE I
Performance of search

Index	Results of MAS	Results of Google
1..10	9	8
11..20	8	5
21..30	7	5
31..40	4	5
41..50	4	5
51..60	5	7
61..70	3	5
71..80	1	4
81..90	2	3
91..100	3	3
101..110	3	3
111..120	4	6
121..130	2	2
131..140	4	4

V. Acknowledgments

Part of this work has been supported by a grant from the National Research Center of the Romanian Ministry for Education and Science (CNCSIS contract 528/2002).

References

[1] 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*, vol. 118, no. 1-2, pp. 69–113, 2000.

[2] Ian Dickinson and Michael Wooldridge, "Towards practical reasoning agents for the semantic web," in *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, J. Rosenchein, M. Wooldridge, T. Sandholm, and M. Yokoo, Eds., Melbourne, Australia, 2003.

[3] Reddy Challam Vishnu Kanth, "Contextual information retrieval using ontology based user profile," M.S. thesis, Information and Telecomunication Technology Center, University of Kansas, 2004.

[4] M. Craven and S. Slattery, "Relational learning with statistical predicate invention: Better models for hypertext," *Machine Learning*, vol. 43, no. 1-2, pp. 97–119, 2001.

[5] George Dimitri Kandaris, "Behaviour-based reinforcement learning," MsC thesis, School of Informatics, University of Edinburgh, UK, 2003.

[6] Peng-Yeng Yin, Bir Bhanu, Huang-Cheng Chang, and Anlei Dong, "Reinforcement learning for combining relevance feedback techniques," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, 2003.

[7] John Domingue, Martin Dzbor, and Enricoi Motta, "Magpie: Supporting browsing and navigation on the semantic web," in *Proceedings of the Conference on Intelligent User Interfaces*, Madeira, Portugal, 2004.

[8] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh, "Hierarchical multi-agent reinforcement learning," in *Proceedings of the 5th International Conference on Autonomous Agents*, Montreal, Canada, 2001.