Agent-Based Web Search Using Evolutionary Strategies

Ioan Alfred Letia
Department of Computer Science
Technical University of Cluj-Napoca
Baritiu 28, RO-400027 Cluj-Napoca
Romania
letia@cs.utcluj.ro

Stefan Mathe
Department of Computer Science
Technical University of Cluj-Napoca
Baritiu 28, RO-400027 Cluj-Napoca
Romania
mathestefan@freemail.utcluj.ro

Radu Razvan Slavescu
Department of Computer Science
Technical University of Cluj-Napoca
Baritiu 28, RO-400027 Cluj-Napoca
Romania
srazvan@cs-gw.utcluj.ro

Abstract – **The query results returned by the classical search engines are not always relevant for a user who wants to find a specific concept. In order to overcome this, we suggest an interactive approach of the process of posing queries. We have developed a multi-agent system which uses evolutionary strategies in order to improve the queries, according to the user's feedback. The system builds a query profile describing his preferences from the point of view of the concept he or she actually expects to find; this profile is used to improve the query accuracy. Then, the profile is updated to reflect the users preferences, allowing him or her to focus on relevant links. The results we have obtained proved to be encouraging from the perspective of the concept-based semantic search.**

I. INTRODUCTION

The explosive development of the World Wide Web offered the user the possibility to access virtually any information resource they might need. However, in order to access the information he or she actually needs, a certain user has to browse a huge number of web sites; among them, only a tiny fraction might be of real help for the topic he or she is interested in. The search engines which have been developed, and among them Google, Yahoo or Altavista, aim to solve this problem by indexing the web pages and providing the user with a set of links which might be of interest, based on a query issued by him or her. However, their performance is not 100% satisfactory due to some inherent limitations of the search strategies they employ.

First, the vast majority of today's search engines are based on the vector-space model described in [8]. The methods in this class regard a document as a vector of words and assess the similarity between documents as the dot product of the corresponding word vectors. This is a syntax-oriented approach rather than a concept-driven one; thus, it does not prove effective for revealing semantically similar documents, forcing the user to browse many irrelevant web sites.

Then, query-oriented surveys [3,9] showed that many users usually issue queries consisting of a small number of words. This makes the search engines retrieve many irrelevant documents which contain the submitted keywords, whilst missing some relevant documents which do not contain them.

Finally, the order in which the query results are ranked has a high impact on the search effectiveness as it may save an important amount of time if the proper results get higher ranks and are presented among the top links in the answer list. However, this target is difficult to achieve based solely on the web page structure. For instance, the Page Rank algorithm [6], a version of which is reportedly used by Google, computes the ranking of a page based on the links pointing to it. Because of this approach, again many irrelevant documents might get higher ranks, therefore increasing the time needed until the user gets the page he or she actually was after.

One of the solutions suggested for coping with these drawbacks is to build a system based on an interactive approach. The user offers a feedback to the system, by evaluating the query results according to his or her actual needs. This feedback comprises the user's preferences concerning the already provided results of the search; based on this feedback, the system will build a query profile which will guide the further search process accordingly. By combining the document space exploration with the query profile exploitation, the quality of the subsequent queries becomes higher than that of the original one. We can say the system tries to capture the concept the user had in mind when assessing the query relevance rather than guiding the search based solely on keywords' syntactical similarities.

This paper presents the multi-agent system we have built pursuing this approach. It follows the ideas presented in [4], with a set of improvements we will emphasize later. The system consists of four agents: the discovery agent, which poses queries to a search engine, namely Google; the information agent, which builds the query profile, based on evolutionary strategies; the filtering agent, which re-orders the query results according to the information in the query profile; the interface agent, which does the interaction with the user. The system still makes use of an existing search engine, i.e. Google, to answer the queries; however, it keeps on refining the issued queries till the actually needed results are found.

This refining process relies on the query profiles the system builds and maintains. A query profile is the list of the most frequent stems in the page already indicated by the user as being of interest. After a new page is stemmed and analyzed, the corresponding query profile is updated accordingly.

The rest of this paper is organized in the following manner. Section II gives a brief description of the evolutionary strategies and how they could be applied to the problem of web search. In Section III, we describe the multi-agent system in detail and explain how it is used for query refining and profile updating. Section IV presents the query results, together with the average fitness of individuals in the population; this is done for a couple of query refining iterations. Section V concludes and sketches the possible improvements of the system.

II. EVOLUTIONARY ALGORITHMS IN WEB SEARCH

Evolutionary algorithms try to use the idea of natural selection in guiding the search for a solution to a given problem [5]. Their general mechanism consists of a set of iterations executed over the whole set of individuals, called population. One individual represents a point in the space of all possible solutions of the problem. One iteration comprises the steps of selection, crossover and mutation. Selection means probabilistically choosing a set of individuals in the population to be replaced. The survival chance of given individual depends on how fit that individual is for the problem intended to be solved; a fitness function should be provided in order to asses the distance between the current individual and the ideal solution. Crossover means combining segments obtained from two individuals in order to obtain two new ones, while mutation is slightly altering one individual in order get a new one. The whole process is repeated till an acceptable solution is found or a given number of iterations is reached.

The Evolutionary Strategies (ES) approach [1] follows the same idea, but gives a higher importance to the mutation operator. An individual is seen as a fixed-length string, whose components are real values which have to be optimized. Mutation is the most important operator for creating the new generation and consists of adding normally distributed random numbers to all components of an individual in the same time. Each individual has assigned a standard deviation σ, called the step size, which is inherited from its parents and then modified by the logarithm of the normal random numbers. If μ is the number of individuals in the population and λ is the number of generated offspring individuals, then, for the next generation, μ individuals among the total number of $\mu + \lambda$ will be selected.

The ES approach is preferred if some dependencies among the components exist, due to their higher speed compared to the genetic algorithms. Because word frequencies are not mutually independent in case of document search, we picked up the ES for the problem we addressed. To be more specific, the implementation uses the algorithm described in [4]. Each individual consists of a word frequency vector $(f_1,f_2,..,f_n,\sigma_1,\sigma_2,..,\sigma_n)$ where f_i $(1 \le i \le n)$ are frequencies corresponding to the words in the query profile (all individuals are derived from the query profile by varying word frequencies) and σ_i are the corresponding standard deviations.

The process of mutating an individual is done in the following manner:

$$\sigma_i' = \sigma_i e^{\tau' N(0,1)+\tau N_i(0,1)} \quad (1)$$

$$f_i' = f_i + \sigma_i' N_i(0,1) \quad (2)$$

where:

$N(0,1)$ is a normally distributed random having mean 0 and variance 1

$N_i(0,1)$ is a normally distributed random number for component i, with mean 0 and variance 1

$$\tau = 1/\sqrt{2\sqrt{n}}$$

$$\tau' = 1/\sqrt{2n}$$

The evolutionary strategy takes place as follows:

1. Generate the initial population of μ individuals
2. Generate λ off-springs by applying the formulas above
3. Evaluate each offspring
4. Select the best μ fittest individuals from the original population together with the off-springs
5. Continue the process above a given number of times or until a timeout signal is received
6. Select the fittest individual from the final population and supply the resulting 1 words having the highest frequency to the user as the final query.

III. SYSTEM ARCHITECTURE

The multi-agent system we have developed is written in Java and makes use of Google as the primary search engine. This section describes the system components and roles and explains how the queries are successively modified, using ES, in order to obtain semantically significant search results.

The system comprises four types of agents:

1. Discovery Agent, which routes the queries to the search engine;
2. Information Agent, which updates the query profile and improve the queries based on this
3. Filtering Agent, which re-ranks the resulting links
4. Interface Agent, which does the communication with the user.

From a functional perspective, the search consists of two steps. In step one, the user formulates a query, which will eventually get answered by the search engine; the user will be presented a set of links which are supposed to be relevant for him or her. Step two starts when the user requires a query improvement. In this step, a set of new queries is sent to the search engine; the key terms of these new queries are built using the query profile in order to narrow down the bunch of links to the interesting ones.

To be more specific: in step one, the query issued by the user is taken by the Interface Agent and sent to the Discovery Agent. This latter agent sends the query further to a classical search engine. Right now, the search engine used by the system is Google. The answer of the search engine is sent to the Discovery Agent. The Discovery Agent then routes it to the Filtering Agent for re-ranking and then the result is displayed. The entire flow involved in step one is summarized in Fig. 1.

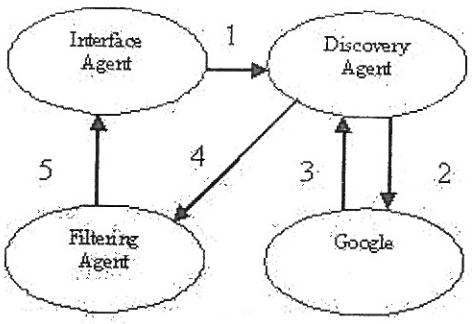

Fig. 1. Normal query

When the user chooses to improve the search quality, a different flow is employed. Based on the query profile, the Information Agent generates new queries and sends them to the Discovery Agent for the dialog with the search engine as above. In this case, the results are not filtered any more by the Filtering Agent.

The flow corresponding to this second step of the process is presented in Fig. 2.

In the rest of this section, we will describe each agent's functionality in more detail.

A. The Discovery Agent

The task of the discovery agent is to perform queries on the internet at the request of the interface or of the information agent. A classical search engine is used to perform these queries. Our implementation uses the free Google API java library, which is provided by Google for application developers. For performance reasons, query results are cached, in order to avoid repeated queries for the same keywords. Due to the dynamism of web, cache results are assigned an expiration period, beyond which the cache entry is considered stale.

B. The Information Agent

The information agent is the most complex agent in the system. The information agent plays two important roles in the system:

- Build and update the query profile
- Improve queries

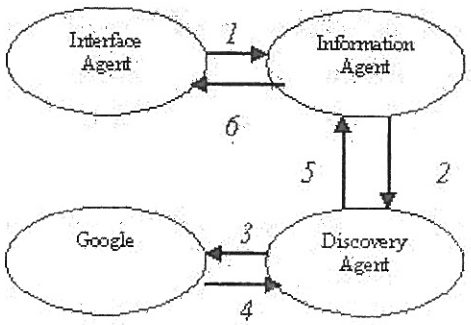

Fig. 2. Improved query

B1. Managing the query profile

The primary purpose of the information agent is to maintain a query profile which describes the pages that have been selected by the user so far.

The process of building the query profiles begins with an analysis of the URLs selected by the user. The analysis of a page consists of several steps:

1. Parse the page using an HTML stemmer, ignoring format and control information (HTML tags, scripts, etc.) and transforming each word into its stem. The process of extracting the stem from each word is achieved by using the Porter algorithm. A description of this algorithm, and a free implementation of it are available at [2]. Each word is counted, together with its stem.
2. Parse the list of word-stem pairs, by merging those pairs having the same stem and by keeping the most frequent form of the word that generated that stem.
3. Sort the list of stems in descending order and keep the top k most frequent stems.

Thus, at the end of this process, each page is transformed into a list of stems, each tagged with its number of occurrences in the page. The total word count of the page is also kept. Hence the frequency of each word is known, but, unlike in the approach in [4] we have decided to also keep the total word count of the page, which will prove useful when updating the query profile.

Once a page has been analyzed, the results of the analysis are stored in a cache, in order to avoid further useless computations.

The query profile is the list of the top 'm' most frequent word stems in the pages selected by the user so far. After a page is analyzed, the query profile is updated with the results of the analysis. The following formula is used for each word stem appearing in the profile, the analyzed pages or both:

$$f_{new} = \frac{f_{profile} count_{profile} + f_{page} count_{page}}{count_{profile} + count_{page}} \quad (3)$$

and the word count of the profile has to be maintained also in order to be able to infer the frequency of each stem:

$$count_{new} = count_{profile} + count_{page} \quad (4)$$

where:

$f_{profile}$ is the number of occurrences of the word stem in the profile

f_{page} is the number of occurrences of the word stem in the selected page

$count_{profile}$ is the total number of words in the profile

$count_{page}$ is the total number of words in the page

f_{new} is the number of occurrences of the word in the new profile

$count_{new}$ is the number of word encounters in the profile

When applying the above formulas in case a word does not appear in the page or profile, we assume a 0 corresponding occurrence count.

After applying this process, the stems are reordered based on their new frequencies and the top 'm' most

frequent stems are retained. Therefore, one can say that the profile of the query is actually an efficient approximation of the result that would be obtained by analyzing the concatenation of all the HTML pages selected so far. The word counts of the pages play the roles of weighting coefficients in (3). By doing this, the impact of a very frequent word in a page containing very few words is reduced compared to the impact of having a high frequency word in a large page. Thus longer pages will have higher influence on the query profile than shorter ones.

B2. Improving queries

Once a query profile has been built from the pages selected by the user, the information agent will try to improve the query. This is done by an evolutionary strategy described in Section II.

The fitness computation is done in the following manner:

1. Choose l words with highest frequency from the individual
2. Submit a query containing these words to the search engine. If the number of results returned is below a given threshold, relax the query by dropping the least frequent word from it and re-submit the query. Repeat this process until the number of results is above the threshold.
3. Analyze each result of the query.
4. For each result, compute a similarity value by making the cosine product between its word frequency vector and the original query profile.
5. Compute the fitness of the offspring by being the a weighted mean of the similarities obtained in the previous step. Here, we take a slightly different approach from that presented in [4]. If a result has already been selected, it will be assigned a smaller weight than a result which has not been selected yet. In this way we promote individuals that give many new relevant pages, rather than individuals giving selected pages over and over again. The final formula to be used is:

$$fitness = \frac{\sum_{i=1}^{k} w_i Sim_i}{\sum_{i=1}^{k} w_i} \qquad (5)$$

C. The Filtering Agent

The purpose of the filtering agent is to re-rank the pages based on their cosine similarity to the query profile, such that the user is presented with the pages most relevant to the query. The reason for this is that, once a query profile is building up, a discrepancy will come about between the ranking done by the classical search engine based on the similarity to the keywords and the one done by the filtering agent based on the similarity to the appropriate query profile.

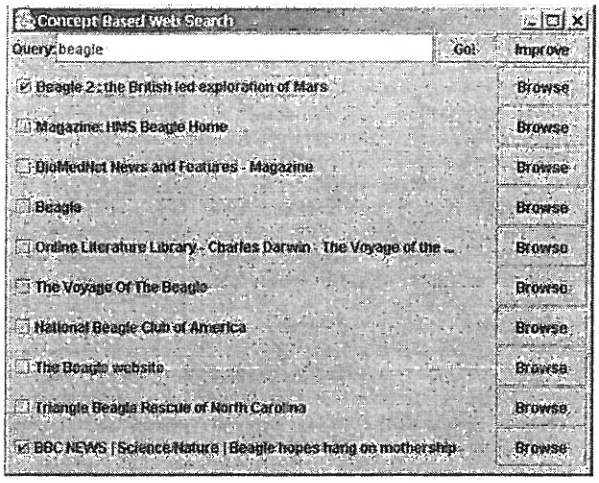

Fig. 3. The Interface Agent

D. The Interface Agent

The interface agent is responsible for all interactions with the user. By entering a query and clicking the Go! button, the user can submit his query in exactly the same way it would use a normal search engine. In this case, the interface agent will submit the query to the discovery agent, which in turn will pass it to the search engine. The results are then passed to the filtering agent for re-ranking.

A snapshot of the interface agent is presented in Fig, 3.

The user can browse each page by clicking on each hyper-link, very much like when using a classical search engine. If a page is considered to be relevant to his or her interests (i.e. it is related to the concept of his or her search), the user can select the page by checking the corresponding checkbox.

By clicking the Improve button, the user can ask the system for an improved query based on the pages he or she has selected as relevant. As a result, the interface agent will ask the information agent to update the query profile based on the pages that have been selected. It will then try to improve the query and return the final results to the user.

The user interface also provides the user with the possibility to save the current query profile to disk for later use. What he or she actually does is that he is saving information regarding the concept of his or her search such that, at a later time, the same concept can be loaded and used to search for newly available pages. In this way, after a couple of initial iterations aimed to "train" the system, the user can, at any time and by a single mouse click, query the web for news regarding the concept he or she is searching for.

IV. EXPERIMENTAL RESULTS

The system has been tested for more queries. In order to illustrate its capabilities, we will show the results we got when we searched for the concept "Beagle" - the European probe lost on Mars. Because of the semantic ambiguity of this word, many other links, not connected to Mars exploration, are discovered; e.g. Darwin's "Voyage of the beagle" or different sites concerned with breeding beagles

(i.e. birds). In order to test the query improvement capability of the system, we performed a two-step process as described earlier in this paper.

Fig. 3 presents the interface agent displaying the query results obtained after step one, which means no query refinement has been drawn yet. As we can see, only the first and the last link are actually related to the topic of Mars exploration which we were interested in from the very beginning.

Then we selected the links we considered of interest for us by checking the corresponding checkboxes next to them (see Fig. 3) and asked the system to improve the query. The result was that, after this query improvement, *all* query results are relevant for the area the user has focused on. The results displayed by the interface agent in this latter case are depicted in Fig. 4.

It can easily be noticed that each of the displayed links is concerned with the Mars exploring robot, while the previous irrelevant links are avoided altogether. The conclusion is the query has been significantly refined and could offer the user a much higher accuracy when guiding his or her search on a specific field.

The convergence of the evolutionary strategy, in terms of the average fitness value per generation versus the number of generations is presented in Fig. 5. The results have been obtained for 8 generations, using a population of 64 individuals, 64 off-springs and an initial step size of 0.5.

As one can easily see, the average fitness value increases from generation to generation. The same ascending trend has been obtained for different values of step size, number of individuals and/or number of off-springs, but the curve slope was different with each experiment. However, the higher the values of these three parameters, the longer the time needed for refining the query; therefore, a trade-off between the search accuracy and the responding time has to be achieved. Acceptable values for time and accuracy have been obtained for a step size of 0.03, a population of 16 individuals and an offspring of 16 individuals as well.

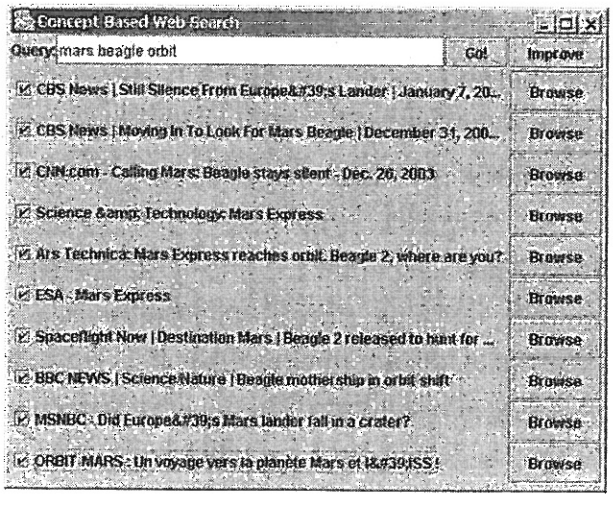

Fig. 4. Second try: all links are relevant

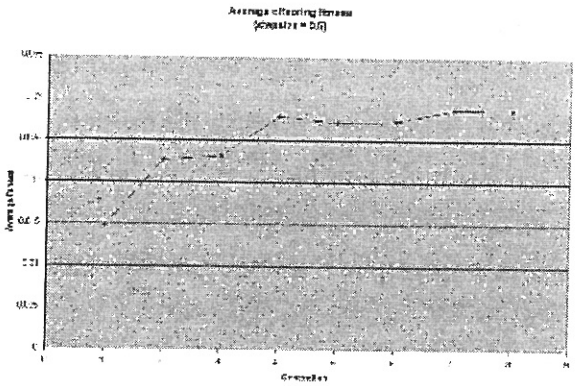

Fig. 5. Algorithm Convergence

V. CONCLUSIONS AND FUTURE WORK

The results we obtained show the ES approach of the web search could offer a significant improvement of the search accuracy when the user is focused on a specific concept. The suggested refinement offers the possibility to get relevant results even if the semantic concept is not specified in an explicit manner or the query term is ambiguous.

In order to prove this idea, we have built a multi-agent system which implements an improved version of the evolutionary strategies for web search and tested it on a set of queries, with good results. The convergence of the evolutionary process ensures a reasonable response time for the whole process, without sacrificing the search accuracy.

Investigations on evolving multi-agent systems [7] have already shown possible improvement in their overall behavior. Given the huge task such agents have to face, the decomposition of tasks and coordination among agents, specialized in various sub-areas, should further improve performance.

VI. ACKNOWLEDGEMENT

The work for this article has been supported in part by the National University Research Council in Romania, within the framework of the research project number 528 / 2002.

VII. REFERENCES

[1] T. Back. *Evolutionary Algorithms in Theory and Practice,* Oxford University Press, NY: 1996.

[2] http://www.tartarus.org/martin/PorterStemmer.

[3] B. Jansen, A. Spink, J. Baterman and J. Saracevic, "Real Life Information Retrieval: a study of user queries on the web" *SIGIR Forum* vol. 32(1), 1998, pp. 5-17.

[4] W.-P. Lee and T.-C. Tsai, "An Interactive Agent-Based System for Concept-Based Web Search" *Expert Systems and Applications* vol. 401, 2003, pp. 365-373.
[5] T. Mitchell. *Machine Learning,* McGraw Hill: 1997.
[6] L. Page and S. Brin , "The anatomy of a large scale hypertextual Search Engine," in *Proceedings of the 7th International World Wide Web Conference*, 1998.
[7] M. A. Qureshi. "The evolution of Agents," PhD Thesis, department of Computer Science, University of London, UK, 2001.
[8] G. Salton. *Automatic Text Processing* Reading, MA: 1989.
[9] C. Silverstein, M. Heizinger, M. Hannes and M. Moricz. "Analysis of a Very Large Web Search Engine Log" *SIGIR Forum* vol. 33(3), 1999, pp. 6-22.