A Web Searching Agent that Uses Intelligent Techniques

Carla Salazar-Serrudo and Javier García-Villalba

Department of Computer Systems and Programming
Complutense University of Madrid (UCM)
Avda. Complutense s/n, 28040 Madrid, Spain
E-mail: javiergv@sip.ucm.es

Abstract. This paper shows an agent that searches information in the Web using the search strategy A* belonging to Artificial Intelligence. This agent searches information in the "best" web pages, i.e., in the pages that we think are very good options. We can conclude three facts: this agent has similar behaviour to the humans, this solution produces better response times than users and a software that applies an old and fundamental concept in a current problem.

1 Introduction

The information amount saved in the Web is growthing in exponential and chaotic ways. Some authors compares Web with "The library of Babel" [9]: the data span over millions of computers and different platforms, the data are highly dynamic (40% of the Web information changes every month) [1][10] and the data are unstructured (don't have a conceptual model), redundant and of low quality (the data are not reliable) [1][14]. Therefore, tools to search, retrieve and filter the Web information are indispensable for an effective usage of the Web.

Previous solutions have included automated searching programs such as Gopher, WAIS or, ultimately, search engines that index a portion of the Web documents as text database such as Altavista, Lycos, Yahoo!, etc. These solutions are based just in information retrieval and database methods [1][5][14]. Instead, this work applies intelligent techniques to search information.

Mainly this paper contains the agent development that searches information in the Web pages space using the Artificial Intelligence algorithm A*. In the next sections we will explain the theoretical grounding, the architecture of the agent, how the agent works and other similar papers.

2 Theoretical Grounding

The main theoretical grounding of this work are the Artificial Intelligence problem solving by searching techniques and the software agents. Also, this work approaches the area of information retrieval.

2.1 Problem solving

Artificial Intelligence supplies problem solving by searching techniques. A problem is a collection of information to solve. The problem formulation is defined as a search in a state space. Each state represents a possible situation or configuration of the problem; therefore, the state space includes all the possible situations of the problem. It can specify one or more initial states where begin to solve the problem. Also, it can distinguish one or more goal states that will be the possible problem solutions. Of course, there are operators or rules that allow to reach other states by carrying out an action in a particular state [13][16][18].

Thus, the problem's solution is transformed into a search in a state space to find any path between initial state and goal state. This search is similar to building up a search tree [16][18]. The root of the search tree is constituted by the initial state, the middle-nodes corresponding to the sub trees and the leaf nodes that are the goal states or nodes that do not have successors in the tree.

There are many search strategies, some are blind searchs and others heuristic[1] searchs. The blind search are uninformed search, i.e. they have no information about how to arrive to the goal nor the number of steps or the path cost from the current state to the goal, etc. In the blind searchs, the nodes are expanded according to deepth level in the tree. Some blind searchs are: breadth-first search, depth-first search, depth-limited search, etc. The heuristic search is also named informed search because disposes of useful information to guide the search (i.e. rules of thumb, estimated costs, previous knowledge, etc.). Of course, informed search are more efficient that the uninformed strategies. Some heuristic searches are: best–first search, greedy search, A*, etc.[13][16][18].

The A* algorithm is the most important informed search and was created by Hart et al in 1968 [7]. The A* algorithm makes the search through the state tree expanding the nodes with the best heuristic function and the minor cost possible. Thus, the A* algorithm guarantees to find the optimal solution, if this exists, as long as the heuristic function never overestimates the cost to reach the goal [18].

2.2 Agents

An software agent (or agent), in this context, is a software that "inhabiting computers and networks assisting users with computer based tasks" [12]. At present, the software agents are

[1] Heuristic: Rules of thumb that domain experts could use to generate good solutions without exhaustive search [18, page 94].

very needed because every day tasks are computer based (e.g.: email, acquisition of information, e-shopping, etc.). Users can delegate to agents their tasks and the agents can act on user's behalf in tasks such as information retrieval, mail management, eshopping, memory aids and so on. The metaphor applied is that of "a personal assistant who is collaborating with the user in the same work environment" [11] (see figure 1).

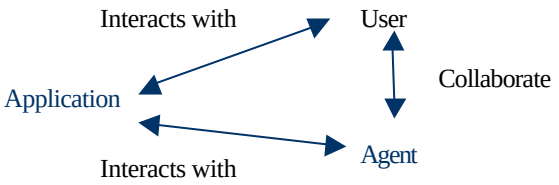

Fig. 1. Relationship between agent, user and application [11, page 33].

An agent principally has the next characteristics: a) Autonomy: it is able to formulate its own goals and to act in order to satisfy them. b) Reactive: it is situated in and reacts to its environment. c) Cooperation: it collaborates with other agents or humans to accomplish its tasks. d) Adaptive: it learns from their experience and the user's behaviour. e) Mobile: it migrates from machine to machine in a heterogeneous network under its own control [2][6][8][20].

People often feel lost or disoriented when navigating through the Web searching the information besides that findind useful information is a excessive time consuming process. The agents may help to alleviate this feeling by obtaining information for the users through the Web, accessing to multiple information sources and returning with responses in a minimal time and lowest cost.

3 Problem definition

The information search was made in the Bolivian Universities domain. Therefore, the problem state space is the set of web pages of this domain. The search tree is built up of the next way: the root of the tree is the web page in which the user starts the search (any one home page of Bolivian Universities). The nodes' tree will be the web pages that it can reach from initial page and in the leaf nodes will be the searched information or will be nodes that have not links (see figure 2). The search ends when searched words are found or when the search time is greater than a given limit time. It disposes an URL's database with home pages of the Bolivian Universities.

The most important implemented operators are: go-page, back-page, search-word, and go-link. The go-page operator allows to open a web page; back-page allows return to previous web page; search-word is a parser that examines a web page looking for words request by the user and go-link is the operator that expands a web page by theirs links to other pages. The go-link operator will just expand the links whose titles have similar words to the user query (another heuristic used).

The search-word operator is a parser that examines the HTML pages in search of the words request by the user so: it ignores the stopwords (or non-informative words such as: “a”, “the”, “is”, “in”, etc), stemming the plural noun to its single form and inflexed verb to its original form, following the information retrieval methods [1][17]. Also, its was necessary to have some tables with stem words, synonymous and stopwords of the domain.

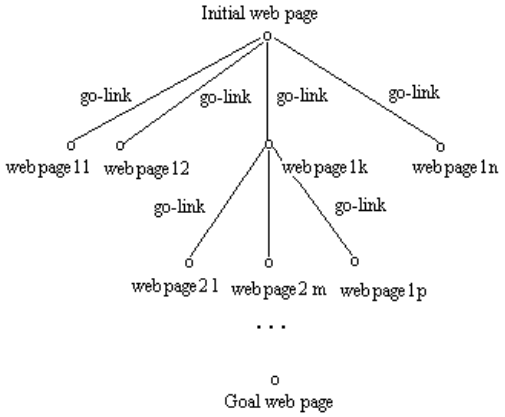

Fig. 2. Example of search tree for the Web.

The evaluation heuristic function (1) must be included in the A* algorithm to arrange the nodes to expand.

$$f(n) = g(n) + h(n) . \quad (1)$$

This function measures node goodness using numerical values and is composed by g(n) that is the path cost from initial page to page n. In this case, g(n) is the connection time from initial page to web page n. The h(n) term is the heuristic that estimates the path cost more cheapest from page n to goal page. In this problem, h(n) tries to get the web pages that includes the greatest amount of searched words in the minor time possible.

3 Agent Developing

In this section is described the main aspects of the development and working of the search agent.

3.1 Agent arquitecture

Agent architecture (see figure 3) reveals its main components. As we can see, the agent assumes a personal assistant role [9][11] of the user for the information search. The user delegates its work to the agent and this makes the search by him. It can note that the agent also has an intelligent

behaviour to select the best pages where browses thanks to its heuristic function implemented. Following the agent main components are described.

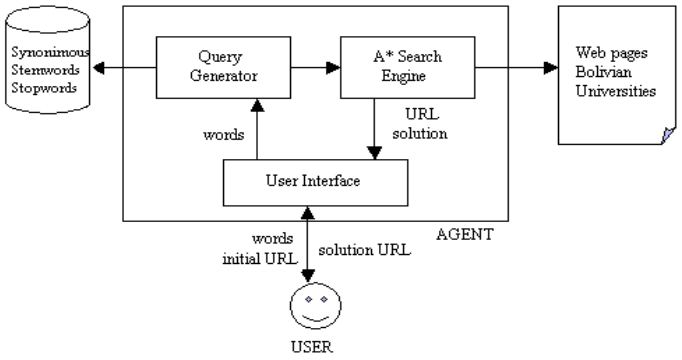

Fig. 3. Search agent architecture.

The user enters the words to search in the user interface module. Then, its generates the user query (i.e.: deletes stop words, stemming and synonymous replaces processing). Later, the A* agent engine searches the words in the Web pages of the application domain in the next way: the tree root is the first web page of the URL's table (also, the user can choose the root page), then the search engine tests if this is a goal page (i.e. if it includes the searched words) . If it is not, the engine search applies the operator go-link to the current page, thereby generating a new set of pages (as much links has the page). The search engine must make a choice about which one to consider further. The choice of which web page to expand first is determined by the heuristic function. The search algorithm is described in figure 4. Finally, the agent returns with the best page solution to the user query.

3.2 Search algorithm

The algorithm of the figure 4 was adapted from the [18] to be applied in the Web. A node is a data estructure used to represent state tree, parent, tree depth, path cost, operator applied, etc. [13][16][18]. The algorithm uses a queue structure named `nodes` to collect nodes that are waiting to be expanded. In the search algorithm `nodes` queue assume the value of the URL initial web page. In the loop, the first step is to test if `nodes` queue is empty. If it is empty the function returns failure, else the algorithm orders the queue using the heuristic function in this way: the best node is the first and so on. After, `node` assume the value of the URL web page of the best heuristc value node. The next step consists in to check if the `node` is the goal (i.e. its node

contains the searched words). If it is goal, then return `node`, else its expands the `node` (i.e. applying the go-link operator to the current web page).

```
Function SearchWebA* (String words, URL initial_page)
// Return web pages with searched words by the user
nodes ← initial_page
Loop do
   If Empty(nodes) then Return failure
   nodes ← Order_queue_heuristic_function(nodes)
   node ← Remove_front_queue(nodes)
   if Goal(node) then Return node
   nodes ← Insert_queue(Expand(node))
End Loop
End Function
```

Fig. 4. The algorithm that searching the Web using A* (adapted of [18, page 73]) .

3.3 Heuristics functions

The search A* algorithm uses the evaluation heuristic function (1) in the next way:

- g(n): connection time from initial node to node n.
- h(n): connection time gets by the ping[2] command from node n to possible goal node – number equal words in the web page (node n).

Since this h(n) never overestimate the real cost from node n to goal node, the implemented algorithm A* is complete (if any word exists, the algorithm will find it), optimal (the algorithm will find the cheapest solution) and uses the minor amount of time, cost and space [13][18].

It must underlines that a word of the title link is considered equal to searched word if is equal lexicographically, is synonimous or has the same root. Of course, the stopwords are omited in this comparation and search process. If there aren't equal words, this component of the function is assumed to zero.

3.4 Repeated nodes

In this problem is very important to avoid repeated nodes because the whole part of pages includes links to other pages and these referenced pages have links to the first page and so. This produces infinite searches and time and cost increment. The agent uses a hash table that stores all the URL's pages expanded. Thus, in every time it is generated a new node, it is verified if the new node

[2] Ping: Operating system command to get the connection time from source to target electronic address (IP).

already exists in a reasonably efficient way. If the node no exists, then simply it is added to tree; else it is deleted.

Another alternative will be: in every time it is generated a new node, it is verified if the new node is better than the old node, i.e., if the new path in the tree (start node to new node) is better than the old path between the start node and the old node. If it is better, it should update the heuristic function to new node; else, does not anything [16].

4. Agent working

The agent works in the next form: the user enters the word or words to search. Also, if user wants, he/she can specify the URL address where begins the search (one of the Bolivian Universities URL).

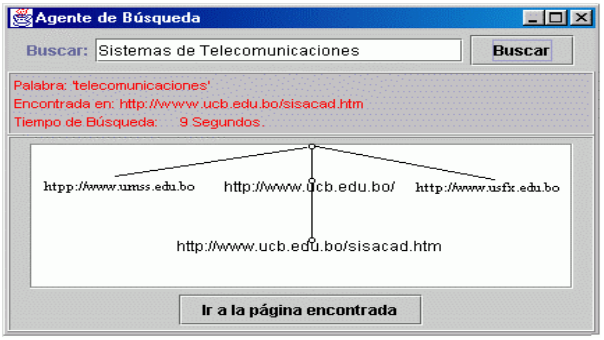

Fig. 5. Search agent interface.

Then, the agent generates the query. The agent begin the search building the search tree whose root is the URL indicated by the user. Subsequently, it is going to expand the links with the best heuristic function to find the web page that includes the searched words. Finally, the search agent returns a URL address with the searched words, if this exists (see figure 5).

5 Similars works

In researched bibliography we did not find similar approaches, i.e., agents that search the web using conventional techniques of Artificial Intelligence. Moreover, the works are focalized on in the information retrieval area, such as the crawlers (spiders, wanderers, walkers or knowbots), or in the information filtering or in the machine learning areas.

Web crawlers have neither the Artificial Intelligence nor the machine learning approach. They just use conventional search and indexing techniques of the information retrieval. The most populars agents on the Web are indexing agents such as Lycos, the WebCrawler and InfoSeek. Indexing agents carry out a massive and autonomous search of the Web (scanning over a million documents) and store an index of words in documents titles and documents texts. After a query of the user, indexing agents can deliver quick responses for documents containing certain keywords. But the indexing agents are very limited because keywords queries can be rather awkward and the indexes are not personalized (i.e.: most queries get many false hits) and of course, indexing agents cannot index all the Web [1][5][10].

WebMate [3] is a personal agent that helps users to browse and search the Web. It extends the information retrieval algorithms to personalize the browsing. It can learn the user profile using only word (it is not intelligent) and aids the user in alias, prefects and monitor bookmarks or web pages for changes.

Letizia [9] is an agent that assists a user browsing the Web. Letizia tracks user's browsing behavior and tries to anticipate what items may be of interest to the user. Letizia adopts a midway strategy between information retrieval and information filtering. PowerScout [10] is very similar to Letizia, but the first is more general because uses global reconnaissance and Letizia just local reconnaissance. Both are assistants personalized.

ShopBot [4][14] and ILA [14] are two agents that automatically learn to use the Web resources. ShopBot learns how to extract information from online vendors with the purpose of learning how to shop at those vendors. ILA learns to translate information from Internet sources into its own internal categories. Both using machine learning techniques.

6 Conclusions and future work

This paper exposes the developing of an agent that search information in the Web using the A* intelligent search strategy. In general, the search agent has better performance than the human users. Thought not so good as the search engines in speed, they are not static as them and can be more personalized.

We consider necessary to enhance the heuristic function to have any and more functions that satisfy the user's preferences and criterion (e.g: heuristic functions based on factors such as cost of access, coverage and speed, quality of the web page returned, etc.). Moreover, it has been planned to add machine learning techniques and information retrieval methods to get more powerful solutions.

Acknowledgements

This work has been partially supported by CYTED (Red VII.J RITOS 2). The authors want to thank the help of the Dra. Nieves Rodríguez from the University of La Coruña (Spain).

References

1. Baeza-Yates, R., Ribeiro-Nieto, B., *Modern Information Retrieval*, Addison-Wesley, 1999, USA.
2. Berney, B., "Software agents: A review", Technical Document, Manchester Metropolitan University, 2000.
3. Chen, L., Sycara, K., "WebMate: A personal agent for browsing and searching", The robotics Institute Carnegie Mellon University, Pittsburgh, USA, 1997.
4. Etzioni, O., Weld, D., "A softbot-based interface to the Internet", Communications of the ACM, July, 1996.
5. Etzione, O., Weld, D., "Intelligent agents on the internet: Fact, Fiction and Forecast", Department of Computer Science and Engineering, University of Washington, 1995.
6. Finin, T., Nicholas, C., Mayfield, J., "Software Agents for Information Retrieval", Tutorial, University of Maryland Baltimore County, URL: http://www.cs.umbc.edu/abir, April, 2002.
7. Hart, P.E., Nilsson, N., Raphael, B., "A Formal Basis for the Heuristic Determination of Minimun Cost Paths", *IEEE Transaccions on SSC*, Vol. 4, 1968.
8. Knapik, M., Johnson, J., *Developing Intelligent Agents for Distributed Systems*, Ed. McGraw-Hill, 1998, USA.
9. Lieberman, H., "Letizia: An Agent That Assists Web Browsing", Media Laboratory, Massachusetts Institute of Technology, USA, 2000.
10. Lieberman, H., Fry, C. , Weitzman, L., "Exploring the Web with Reconnaissance Agents", Communications of the ACM, August 2001, Vol.44, No. 8, pp. 69-75.
11. Maes, P., "Agents that reduce work and information overload", *Communications of the ACM*, July 1994.
12. Maes, P. "General tutorial on software agents", URL: http://pattie.www.media.mit.edu/people/pattie/CHI97/, Dic, 2001.
13. Nilson, N. J., *Principles of Artificial Intelligence*, Tioga, Palo Alto, California, 1980.
14. Perkowitz, M., Doorenbos, R., Etzioni, O., Weld, D., "Learning to Understand information on the Internet: an example-based approach", Kluwer Academic Publishers, Boston, 1999, pp. 1-24.
15. Omicini, A., Zambonelli, M., Klusch, M., Tolksdorf, R., *Coordination of Internet Agents: Models, Technologies and Applications*, Chapter 13, Eds: Springer, 2000.
16. Rich, E., Knight, K., *Inteligencia Artificial*, Ediciones McGraw-Hill Interamericana de España, España, 1994.
17. Rodhes, B., *Just-In-Time Information Retrieval*, Doctoral Thesis of Philosophy, Massachusetts Institute of Technology, USA, Junio 2000.

18. Russell, S. and Norvig, P., *Inteligencia Artificial: Un enfoque moderno*, Prentice Hall HispanoAmerica, S. A., México, 1996.
19. Welsh, M., *Linux: Instalación y Primeros Pasos*, Ed. Proyecto LuCAS, USA, 1998.
20. Wooldridge, M., Jennings, N., "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review 10, vol. 2, 1995, pp. 115-152.