The New Era in Intelligent Agent Design Using Genetic Algorithms Approach

Gholamali Montazer[1], and Nassrien Deldadeh[2]

ABSTRACT

The intelligent agent is a program that can operate autonomously to accomplish unique task without human counterparts. This concept has gradually grown up since the world wide web protocols (esp. http) have been the most popular media in the world .

In this article , describing the main categories of intelligent agents and their functionalities ,the essential features of these softbots have been discussed , and then a novel approach based on the notions of Genetic Algorithms has been proposed to design the new generation of agents.

Keywords :*Internet, Intelligent Agent, Search Engine, Genetic Algorithms.*

1. Introduction

The world wide web(www) was developed initially to support physicists and engineers at CERN, the European Particle Physics Laboratory in Geneva, Switzerland (1:189-199). In 1993, when several browser programs became available for distributed mul-

1. Assis. Prof. of Elec. Eng.
e-mail:montazer@modares.ac.ir
2. MSc. Student of Elec. Eng.

timedia and hypertext-like information fetching, Internet became the preview for the rich and colourful information cyberspace. However, as the Internet services based on www have become more popular, information overload also has become a pressing research problem. The user interactions paradigm on internet has been shifted from simple hypertext-like *browsing* (human-guided activity exploring the organization and contents of an information space) to content-based *searching* (a process in which the user describes a query and a system locates information that matches the description).

Many researchers and practitioners have considered Internet searching to be one of the most challenging and rewarding research areas for national information infrastructure applications. Internet searching has been one of the hottest topic at recent www conferences (2:225-268). To solve this problem, two major approaches have been developed and tested:

Client-based searching agent

Online database indexing and searching

Client-based searching agent , broadly defined as "agent", is a program that can operate autonomously to accomplish unique task without direct human counterparts such as real estate agents, travel agents, etc. The basic idea of the agent research is to develop software systems which engage and help all types of end users. Such agents might act as "spiders" on the Internet and look for relevant information, schedule meeting on behalf on executives based on their constrains (3:178-188). Many researchers have focused on developing scriptings of interfacing languages for designers and users so that they can create mobile agents for their own (4:604-618). Some researchers attempt to address the question: "how should agents interact with each other to form digital teamwork?". Other researchers are more concerned about designing agents which are "intelligent" (5; 6:98-107) .

Several software programs based on the concept of spiders, agents, or softbots (software robots) have been developed. TueMosaic and the webcrawler are two prominent examples. Both of them are using the best-first search techniques (7:72-79). Using TueMosaic which was developed at the Eindhoven University of Technology, users can enter keywords, specify the depth and width of search for links contained in the current homepage, and request the spider agent to fetch homepages connected to the current homepage. The fish search algorithm is a modified best-first search. Each URL corresponds to a fish. After the document is retrieved, the fish spawns a number of children (URLs). These URLs are produced depending on whether they are relevant and how many URLs are embedded. The URLs will be removed if no relevant documents are found after following several links. The searches are conducted by keywords, regular expression, or by relevancy ranking with external filters. However, potentially relevant homepages that don't connect with the current homepage can not be retrieved and the search space becomes enormous when the depth and breadth of search become large (an exponential search). The inefficiency and local search characteristics of the spiders and the communication bandwidth bottleneck on Internet severely constrained the usefulness of such an agent approach (8:30-40).

The webcrawler extended the TueMosaic's concept to initiate the search using its index and to follow links in an intelligent order. It first appeared in April of 1994 and was purchased by America Online in January of 1995 (9). However The webcrawler evaluates the relevance of the link based on the similarly of the anchor text to the user's query. The anchor texts are the words that describe a link to anchor document. These anchor texts are usually short and do not provide relevance information as much as the full document texts.

Moreover, problems with the local search and communication bottleneck persist. A more efficient and global Internet search algorithm is needed to improve client-based searching agents.

The ***Tkwww*** robot was developed by Scott Specta and was funded by the Air force Rome Laboratory (10). The ***Tkwww*** robots are dispatched from the ***Tkwww*** browser. The robot is designed to search web neighbourhoods to find logically related homepages and returns a list of links that look like a hot list. However, the search is limited to one or two hops, or links, from the original homepages. The ***Tkwww*** robots can also be run in the background to build HTML indexes, compile www statistics, collect a portfolio of pictures, or perform any other function that can be described by the Tkwww Tcl extension. The major advantage of Tkwww robots over other existing spiders is its flexibility in adapting to virtually a set/ number of criteria to guide its search path and control selection of data for retrieval.

The ***Webants***, developed by Leavitt at Carnegie Mellon University, investigates the distribution of information collection task to a number of cooperating processors (11). The goal of ***Webants*** is to create cooperating explorers (ants) that share the searching results and the indexing load without repeating each other's effort. When an ant finds a document that satisfies the search criteria, it shares it with other ants so that they will not examine the same document. For indexing, cooperation among ants allows each indexer to conserve resources by distribution the indexing load between different explorers. During query, user could restrict the query to the local ant or allow it to propagate to the entire colony.

The RBSE ***(Repository Based Software Engineering)*** spider was developed by David Eichmann and was funded by NASA (12:18-21). RBSE spider was the first spider to index documents by content. It uses the Mite program to fetch documents and uses four searching mechanisms, (i) Breadth First Search from a given URL, (ii) Limited Depth First Search from a given URL, (iii) Breadth First Search from unvisited URLs in the database, and (iv) Limited Depth First Search from unvisited URLs in the database. The database of RBSE spider consists of three tables, ***Spider Table, Leaf Table***, and ***Avoid Table***. The Spider ***Table*** keeps track of URLs that have been visited, the ***Leaf Table*** contains node documents, which do not have links, and the Avoid Table contains a permanent list of patterns to match documents against for visit suppression.

An alternative approach to Internet resource discovery is based on the database concept of indexing and keyword searching. They retrieve entire web documents or parts of these documents and store them on the host server. This information is then indexed on the host server to provide a server-based replica of all the information on the web. This index is used to search for web documents that contain information relevant to a user's query and point the user to those documents.

The ***World Wide Web Worm (WWWW),***

developed by McBryan, uses crawlers to collect Internet homepages and was made available in March of 1994 (13). It was an early ancestor to the newer species of spiders on the web today. After homepages are collected, the system indexes their titles, subject headings and anchor texts and provides a UNIX grep-like routine for database keyword searching. However, it does not index the connect of the documents. The title, subject headings and anchor texts do not always provide accurate description of the document's information. Moreover, *WWWW* does not create a large collection of relevant documents in the database due to the limitation of its crawlers and its grep-like keyword searching capability is very limited.

Aliweb developed by Koster, adopts an owner-registration approach to collecting Internet homepages (11). Homepage owners write descriptions of their services in a standard format to register with ***Aliweb. Aliweb*** regularly retrieves all homepages according to its registration database. ***Aliweb*** alleviates the spider traversal overhead. However, the manual registration approach places a special burder on homepage owners and thus is unpopular. The Aliweb database is small and incomplete for realistic searches.

The ***Harvest*** information discovery and access system developed by Bowman, et al..., presents a big leap forward in Internet searching technology (14). The Harvest architecture consists of five subsystems, ***Gather, Broker, Index/search*** subsystem, ***Object Cache***, and ***Replicator***. Gather collects indexing information and it is designed to run at the service provider's site, which saves a great deal of server load and network traffics. ***Gatherer*** also feeds information to many ***Brokers,*** which saves repeated gathering cost.

2. Essential features of intelligent agent

Before designing agent architecture, we should know that there is too much information for the user to architecture. Since the purpose of an intelligent internet searching agent is to assist people retrieve and manage information on the web, it should have the following features (15:269-277):

I. Intelligent/interactive search;

An effective and efficient search of information from a database is a major issue on the research of information retrieval. When people start to search for information from the web, they often become frustrated when the search result contains too little useful information (or too much garbage). An intelligent agent should give the user an interactive environment so that the user's information need can be pinpointed exactly.

II. Navigation guide; When surfing through the web, it is easy to "go stray" in cyberspace. A good Intelligent Information Retrieval Agent (IIRA) should provide guides or roadmaps so that users can get assistance when stuck in their navigation on the web. For instance, the agent may analyze pages recently read by the user in order to suggest related subject areas and pages. Another kind of navigational guide is to highlight potentially interesting hyperlinks.

III. Information auto-notification; It is tedious for people to check whether a page has been updated. After the user specifies the kind of information she needs, an IIRA should be able to detect updated information or even download them automatically. Massages may be sent to the user to notify that new data have become available. Furthermore, it is worthwhile for an agent to analyze the user's reading preferences, so that it may prompt interesting pages to the user automatically. On-line subscribed news notification is also useful.

IV. Personal information management; Categories, directories, or folders are familiar ways for people to manage tree-structured hierarchical data. It is useful for an IIRA to manage personal categories in an intelligent/interactive way. For instance, the agent may provide suggestions to build a personal category tree for each user. This category information can later be used to help user search or navigate on the web. Another example of information management is to automatically organize bookmarks so that the user may handle bookmarks more easily.

V. Dynamic personalized web pages; If possible, an IIR agent should interact with connect providers to dynamically produce web page appropriate to the user. For instance, the agent may give content providers the user preferences on interface setting (such as user's background, preferred languages, font size, background color, etc.). The content providers may then produce web pages likely to be more interesting to the user.

VI. Tools as reading-aid; A good

IIR agent may also provide tools to help the user with reading retrieved papers. Such tools may include an on-line directory, an on-line encyclopaedia, and some translation programs. These programs are usually stand along agents themselves. The IIR agent should allow easy incorporation of such tools.

3. Architecture of intelligent agent

The architecture of the intelligent agent is divided into five components:

I. Requests and control parameters

a search begins whenever a task is submitted. Users submit queries with information such as: the starting URLs, the keywords, the number of URLs expected to return and the category of the searching space. The resource in the server provides several categories of searching space when a task is submitted, the appropriate searching space in the available database will be invoked.

II. Graphical user interface and client-server

Common Gateway Interface (CGI) and JAVA have been developed for the graphical User Interface (GUI). GUI provides a link between the submitted task and the searching engine. It provides graphical interfaces, such as : forms, images, scrollers , and radio buttons, for the user to fill up the input and control parameters engine in the formats of table, graphics, or others.

In The CGI interface, server is completely event driven. It can only responds to requests from the clients, but cannot initiate requests of its own, and also responds to one event. These limitations lead to relatively static user interface. The lack of capability in dynamic interaction is undesirable to our spider application. The searching process of the spider usually takes from 5 minutes to 20 minuets to complete. Users are easily frustrated when they are relegated to passive roles during searches and unable to view intermediate results or change parameters in responses to reaching events. As a result, a truly dynamic user interface, based on a client- server design and TCP/IP, is desired. CGI use www protocol, HTTP but because of the two way data flow cannot be handled by this protocol and in order to develop a dynamic interface, the client-server architecture based on the UNIX sockets is created in both C++ and JAVA.

III. Search engine

For this research, several searching

algorithms have been investigated. They include ge netic algorithm, best-first search, and simulated annealing. The goals of these searching engines are to visit the URLs in the neighbourhood and the URLs in the selected searching space and to find the most related URLs by comparing their links and keywords. The details of these algorithms will be discussed later.

IV. Homepage fetching

Currently, there are several fetching machines to retrieve HTML documents in the internet such as |Lynex and HtmlGobble. Lynex is a fully featured www client for users running cursor addressable, character cell display devices such as vt100 terminals, vt 100 emulators running on PCs or Macs. It will display HTML hypertext document containing links to files residing on the local system, as well as files residing on remote system running Gopher, HTTP, FTP, WAIS, and NNTP servers. Similarly, HtmlGobble grasps HTML pages from remote web sites. In order to make the spider more portable and light weighted, instead of using Lynex or HtmlGobble, a genetic fetching function is developed to fetch URLs. Using this genetic fetching function, it speeds up the fetching time by a significantly. Moreover, the spider code is now portable and light weighted.

V. Indexing Score

The goal of indexing is to identify the content of the document (homepage). The indexing method employed in the usual case goes troagh the following procedures: word identification, stop wording, term-phrase formation.

Word identification

Words are identified by ignoring punctuation and case.

- *Stop wording*

A "stop word' list, which includes about 1000 common function words and "pure" verbs, is developed. The common function words are non semantic bearing words, such as on, in, at, this, there, etc. The "pure" verbs are words which are verbs only, such as calculate, articulate, teach, listen, etc. High frequency words that are too general to be useful in representing document content are deleted.

- *Term- phrase formation*

Adjacent words then used to form phrases. Phrases are limited to tree words. The resulting phrases and words are referred as the keywords of the documents (homepages).

After retrieving the keywords of the homepages, the concurrence pattern of indexes which appears in all homepages are identified. Jaccard's score is used to measure the similarity of homepages. The score is computed in terms of the homepages' common links and term frequency and homepage frequency.

4. Genetic algorithms

Genetic algorithms are problem solving systems based on principles of evolution and brevity. Genetic algorithms perform evolution process toward global optimization through the use of crossover and mutation operator. The search space of the problem is represented as a collection of individuals referred to as chromosomes. The genetic algorithms find the chromosomes with the best "genetic material". The quality of a chromosome is measured with an evolution function such as Jaccard's score, (see Appendix). In the algorithm, the initial population is chosen and the quality is determined. In every iteration, parents are selected and children are produced by crossover and mutation operations. Each iteration is referred as a generation.

A sketch of the genetic algorithm for internet client-based searching is presented below:

I. initialize the search space: the spider will attempt to find the most relevant homepages in the search space, using the user-supplied starting homepages. Save all the input homepage, input1,input2, ..., into a set of Current Generation, CG={ Cg1,Cg2,...}.

Homepages in 14 categories will be used as the search space for mutation, the categorization is usually done by multi-layered Kohonon self-organizing feature map. These categories include arts, business, computer, education, entertainment, health, news, recreation, reference, regional, science, social-culture, and social-science.

II. crossover and mutation

Crossover

Fetch the homepages that are linked by the homepages in Current Generation . Compare the linked homepages of these fetched homepages and the linked homepages of the homepages in current generation , cgi. The fetched homepages which have the highest number of overlapping linked homepages with the homepages in current generation will be saved in the set of Crossover Homepages, C ={c1,c2,...}.

Mutation

Homepages are obtained from the ranked homepages using SWISH in the user selected category and saved in the set of Mutation homepages, M ={M1,M2,...}.

The sizes of crossover homepages and mutation homepages are both 50% of the population size. The population size, N, is double of the number of output homepages requested.

III. stochastic selection scheme based on fitness:

Compute the Jaccard's score for each home in crossover homepages (M). The homepages in crossover homepages and mutation homepage are compared with the homepages in Elitepool, E={e1,e2,..eN}. The best N homepages are selected and updated. If the Elitepool is empty, comparison is not processed. Instead, the homepages in crossover homepages and mutation homepages are saved to the Elitepool and automatically become the best N Homepages. Selection of the population for the next generation is based on the fitness scores. Fitter homepages in the Elitepool have better chances of getting selected. Create a "roulette wheel" with solts (F) sized according to the total fitness of the population. F is defined as follows:

$$F=\sum JS(e_i)\ ;\ i=1,2,...,N \qquad (1)$$

Each homepage in these best homepage has a certain number of slots proportional to its Jaccard's score. A homepage is selected by spinning the wheel and saved to current generation. The total number of spinning is N. some homepages will be selected more than once. This is in accordance with the genetic inheritance: The best chromosomes get more copies, the average stay even and the worst die off.

IV. Converge

Repeat steps 2 and 3 until the improvement in total fitness between two generations is less than a threshold. The final converged set of homepages is presented as the output homepages.

Best first search

Best first search is a state space search algorithm. We explore the best homepage at each iteration. The best homepage is the homepage with the highest Jaccard's score. The iteration is terminated when the number of homepages required by the user is obtained. The algorithm is summarized as follows and illustrated in fig1.

- ***initialization:*** initialize the counter k to 0. obtain a set of anchor homepages, A=(A1,A2,...), and the number of desired homepages from the users, D. fetch the anchor homepages and save the links of the anchor homepages to the unexplored set of home pages, H=(H1,H2,H3,...).

- ***Find the best homepage:*** fetch the unexplored homepages and compute their Jaccard's score. The homepage that has the highest Jaccard's score is the best homepage. The best homepage is then saved in the output set of homepages.

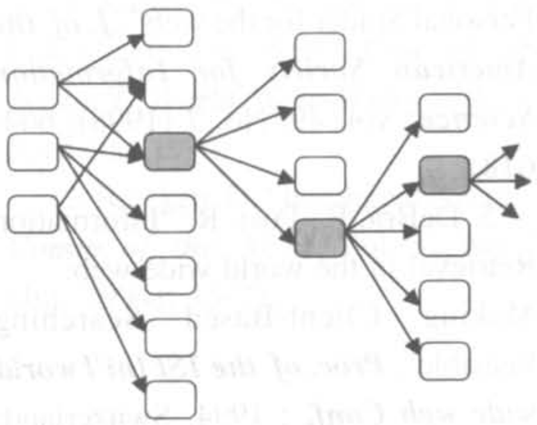

Fig 1. best first search

Explore the best homepage: fetch the best homepage that is determined in he last step and insert its links to the unexplored set of homepages. Increment the counter by 1.

● Repeat step 2 and three until k=D.

5. Conclusion

The Intelligent Agents deal with Internet data discovery automatically, so the use of new concepts of Artificial Intelligence, such as Genetic Algorithms, can develop and extend their abilities. In this paper defining the main aspects of Intelligent Agents, a genetic based algorithm has been applied to the softbot to guarantee the optimal data retrieval of Internet.

Appendix

Jaccard's score

We use Jaccard's score to measure the similarity of Internet documents. The score is computed in terms of the documents common links and indexing. A document with a higher Jaccard's score has a higher fitness with the input document.

The Jaccard's score based on links by the number of total links of two documents. Given two documents, x and y, and their links, $X=x_1,x_2,...,x_m$ and, $Y=y_1,y_2,...,y_m$, the Jaccard's score between x and y based on links is:

$$JS_{link}=\frac{\#(X\cap Y)}{\#(X\cap Y)} \qquad (2)$$

The Jaccard's score based on indexing is computed in terms of the term frequency and document frequency. The term frequency, tfxj , is the number of occurrence of the term j in document x. The document frequency, dfj , is the number of document in a collection of N documents in which term j occurs. The combined weight of term j in document x, dxj , is:

$$d_{xj} = tf_{xj} \times \log(\frac{N}{dfj} \times w_j) \qquad (3)$$

where wj is the number of words in term.

The Jaccard's score between document x and y based on indexing is then computed as:

$$JS_{link}(x,y) = \frac{\sum_{j=1}^{L} d_{xj}\, d_{yj}}{\sum_{j=1}^{L} d_{xj} \sum_{j=1}^{L} d_{y} \sum_{j=1}^{L} d_{yj}+} \qquad (4)$$

Where L is the total number of terms.

References

1. Lee, J. K.W. et al. "Intelligent Agents for Matching Information Providers and Consumers on the World Wide Web " ***Proc. Of the 30th IEEE Conf. on System Science*** ;1997.

2. Tu, H.C.; Hsiang. J." An Architecture and category Knowledge for Intelligent Information Retrieval Agents". ***Decision Support Systems;*** No. 28, (Apr. 2000) : 255-268.

3. Chen, H. et al. "Intelligent Spider for Internet Searching". ***Proc. of the 30 th IEEE Conf. on System Science,*** 1997, 178-188.

4. Chen, H. et al. "An Intelligent Personal Spider for the web". ***J. of the American Society for Information Science***; vol. 49, No. 7 (1998): 604-618.

5. DeBra, P.; Post. R. "Information Retrieval in the world wide web: Making Client-Based Searching Feasible". ***Proc. of the 1St Int'l world wide web Conf.*** ; 1994, Switzerland; Geneva .

6. Bowman, C. M. and et al. "Scalable Internet Resource Discovery: Research Problems and Approaches"; ***Comm. of the ACM***; Vol. 37, No. 8 (Aug. 1994): 98-107.

7. Etzioni, O; D. Weld. "A softbot-Based Interface to the Internet" Comm. of the ACM, Vol. 37, No. 7 (Jul. 1994): 72-79.

8. Maes, P. "Agents that Reduce

work and Information Overload". ***Comm. of the ACM***, Vol. 37 , No. 7 (Jul. 1994): 30-40.

9. Koller, D.; M. Sahami. "Hierarchically Classifying Documents Using Very Few Words". ***Proc. of the 14th Int'l Conf. on Machine Learning***; Jul. 1997.

10. Spetka, S. "The Tkwww robot : Beyond Browsing"; ***Proc. of the 2nd World Wide Web Conf***. '94 ;Mosaic and the web; 1994.

11. Koster, M. "Aliweb: Archie-Like Indexing In The Web". ***Proc. of the 1st Int'l World Wide Web Conf***. '94 ;Geneva; 1994.

12. Riecken, D. "Intelligent Agents"; ***Comm. of the ACM***, Vol. 37, No. 7; (Jul. 1994): 18-21.

13. Armstrong, R. et al ;"Web Watcher: A Learning Apprentice for the World Wide Web" ; ***AAAI Spring Symp.*** on Information Gathering from Heterogeneous. 1995.

14. Goldberg, D.E. "Genetic Algorithms In Search, Optimization and Machine Learning". Addison-Wesley; MA,. 1989.

15. Yang. C.C. and et al. "Intelligent Internet Searching Agent Based on Hybrid Simulated Annealing". ***Decision Support Systems;*** No. 28 (Apr. 2000): 269-277.