An Agent-based System for Modelling the Searching Process on the Web

C. Cândea
Intelligent Agents Lab.
AI Research Group
Reconstrucţiei, Sibiu, Romania
ciprinac@airg.verena.ro

M. Staicu
Intelligent Agents Lab.
AI Research Group
Reconstrucţiei, Sibiu, Romania
mariuss@airg.verena.ro

C. B. Zamfirescu
Department of Computer Science
"Lucian Blaga" University of Sibiu
Emil Cioran, Sibiu, Romania
zbc@acm.org

Abstract — Due to the vastness and dynamic nature of the World Wide Web, there is a tremendous need for flexible services-finding systems that can be easily customised to the personal interests of individuals. Following an agent-oriented approach, the research in this paper aim at addressing such circumstances in a more comprehensive framework, able to extend our results to other interrelated challenges. Usually settled in the information retrieval (IR) research field some relevant issues together with some effective methods to address them are identified and discussed.

I. INTRODUCTION

The highly distributed nature of the Web, and the fact that the content is constantly being updated, presents a serious challenge to those who want to be aware of all kinds of services made available daily. Various search engines and software agents providing various different services are already deployed on the Web. However, novice users of the Web may have no idea where to start their search, where to find what they really want, and what agents are available for doing their job. Even experienced users may not be aware of every change in the Web, e.g., relevant web pages might not exist or their content be valid anymore, and agents may appear and disappear over time. The user is simply overtaxed by manually searching in the Web for information or appropriate agents.

When, at the beginning of 1950's, Calvin Moers, one of the information science pioneers, coined the term IR he also defined the problems addressed by the activity: (1) how to represent and organise information intellectually? (2) how to specify a search intellectually? and (3) what systems and techniques to use for those processes?[1]. Most existing IR systems provide limited assistance to users in locating the relevant information that they need [2]. Much research has focused on designing entirely new IR systems. Given the development costs, organisational friction, and system transfer expenses, the total replacement approach will probably have limited impact. In this context, new trends were emerged, exploring the feasibility of combining software agents with effective IR techniques to improve the performance of current IR systems [3]. Caglayan and Harrison [4] outlines some of the agents' benefits in broad functional categories: automation, customisation, notification, learning, tutoring and messaging.

The remainder of this paper is organised as follows. Section 2 summarises the outstanding category of agent-based IR systems. The third section will present some user modelling aspects that was been taken into account in our implementation. Some architectural issues will be given in the section 4. Finally, some remarks and future path will establish the framework that will provide the basis for future improvements of our tool.

II. AGENT-BASED IR SYSTEMS

Following Croft's Top Ten List [2], of the most significant questions facing current IR systems, Finin, Nicholas and Mayfield [5] identified the agency features that are able to accomplish each particular issue (TABLE I).

TABLE I
IR and agent characteristics (adapted from [5])

	Autonomy	Cooperation	Adaptation
Relevance Feedback		✓	✓
Information Extraction		✓	
Multimedia Retrieval		✓	
Effective Retrieval		✓	✓
Routing & Filtering	✓	✓	✓
Interfaces & Browsing		✓	✓
Term Expansion			
Efficiency & Flexibility	✓	✓	✓
Distributed IR	✓	✓	
Integrated solutions	✓	✓	✓

Consequently, the multi-agent system paradigm represents one of the most promising approaches to build complex and flexible architectures, offering a new dimension for large-scale integration. Below will be briefly outlined some of the well-known types of agent-based IR systems.

Knowbot agent-based IR Systems. A 'knowbot' is an agent-based IR system that provides a single query language to access a variety of information sources. It serves as a representative for the user (which demands program autonomy). Some prototypical examples from this category are MetaCrawler, SavvySearch and NetbotJango.

Adaptive IR systems. The most cited and well-known example from this category is the Fab [6] multi-agent system developed at Stanford University. Fab recommends web pages using adaptive information retrieval techniques to learn an individual's profile. For that, it takes into account users' feedback on how much they liked recommended pages used to adapt the user's profile and assign credit or blame to the recommending collection agents. A "genetic algorithm" is used to evolve the population of collection agents. Collection of

agents will specialise over time to different topics, serving distinct groups of users and useless collection agents die, successful ones live and reproduce.

Collaborative IR systems. A collaborative filtration agent-based IR system makes recommendations to a person based on the preferences of similar users. Content-based recommendation retrieves other documents similar to those liked earlier, while collaborative recommendation retrieves documents liked by other people similar to a relevant one. Typically, they operate with a large vector space (in which each element represents one dimension) and a sparse vector of element ratings. Some classical examples for collaborative IR systems include Yenta, (recommend people), Firefly (recommend products), and Phoaks (recommend readings) [7].

Proactive IR systems. These are systems that only provide information "on request". The user has to know that there is knowledge to be had in a particular situation. The system therefore needs at least some ability to be proactive in its suggestions. However, unlike the calendar program that warns of upcoming meetings, it is impossible to create a back end able to reliably know when a document is useful for a user. In this context Remembrance Agent [8] indexes personal files and e-mails when you perform a task, automatically suggesting relevant documents providing continuous associative recall. In the same category, Letizia [9] is a user interface agent that assists a user in browsing the World Wide Web.

III. ASPECTS OF USER MODELLING

Accordingly to [10], approaches to user modelling in IR can be divided in two main categories: system-centred and human-centred. While the first put emphasis on *relevance feedback* (users are modelled through texts or clusters of texts) and *query expansion* (the initial or modified query is used as a basis for user modelling), the second one takes into account *question shape* (user modelling is accomplished through various interview and analysis techniques). Another method is to build into the system ways and means by which users can on their own model articulate their problem with the system's assistance.

On the user side we can model cognitive, affective and situational levels. Saracevic et al. [10] suggest that user modelling is an interactive process that proceeds in a dynamic way at different levels trying to capture user's *cognitive*, *situational*, *affective* and possible other elements that bear upon effectiveness of retrieval. In such a framework the request must be scrutinised through all its related steps: from request formulation to answer acceptance. So, in a searching process can be delineated three key stages: request formulation, selection of retrieved pages and locating the intended information.

A. *Profile enhancement*

Keywords occurring in a particular searching process (e.g. source description, contextual links) will be clustered using a similarity matrix for the keywords stored in the user profile, very similar with the approach followed by [11] in their Jasper implementation. Contrasting with them, we look at the search process as a whole, instead of the pages stored in the ultimate part of the user's request. We capture the user's choice, the rational behind each of them, the open questions related to the request, the assumption behind it and any related supporting information. The matrix used will give us a measure of the 'similarity' of keywords in the user's profile. For two keywords K_i and K_j, the Dice coefficient is given by the equation (1).

$$2 X \left| K_i \cap K_j \right| / \left| K_i \right| + \left| K_j \right| \quad (1)$$

Once the similarity matrix is calculated it is exploited in two ways: *profile enhancement* (adding those keywords most similar to the keywords explicitly represented in the user's profile in similar way of query reformulation techniques) and *proactive searching* (search proactively for new WWW pages relevant to user's interest).

Using complete-link clustering technique [11] the similarity between the least similar pair of items from two clusters is taken as the similarity between the clusters obtaining the cluster dendogram. A similarity threshold can be set to provide the similarity degree between the clusters.

B. *Adaptive recommendation*

Accordingly to [6] for the content-based approach, there are four essential requirements:

w – a representation of a Web page.

m – a representation of the user's interests.

p(w, m) – a function to determine the pertinence of a Web page given a user's interest

u(w, m, s) – a function returning an updated user profile given the user's feedback *s* on a page *w*.

The assumption underlying content-based systems is that the content of a page is what establishes the user's interest. Going on, the content of a page can be represented purely by considering the words contained in the text and also by its description. Considering the vector-space model of IR [12] as a suitable mechanism for documents based representation, documents and queries are represented as vectors. This model has been used and studied extensively, representing a competitive representation form with alternative IR methods [13]. This model assumes a dictionary vector *d*, where each element d_i is a word. Each document then has a vector *w*, where element w_i is the weight of a word d_i for that document. If the document does not contain d_i, then w_I=0. As in Fab implementation [6], in our formulation we reduce words to their steams using the Porter algorithm [14]. That will ignore words from standard stop-list-words, and calculate a TFIDF weight: the weight w_i of a word d_i in a document *W* is given by equation (2).

$$w_i = (0.5 + 0.5\, tf(i) / tf_{max}) (\log (n / df(i))) \quad (2)$$

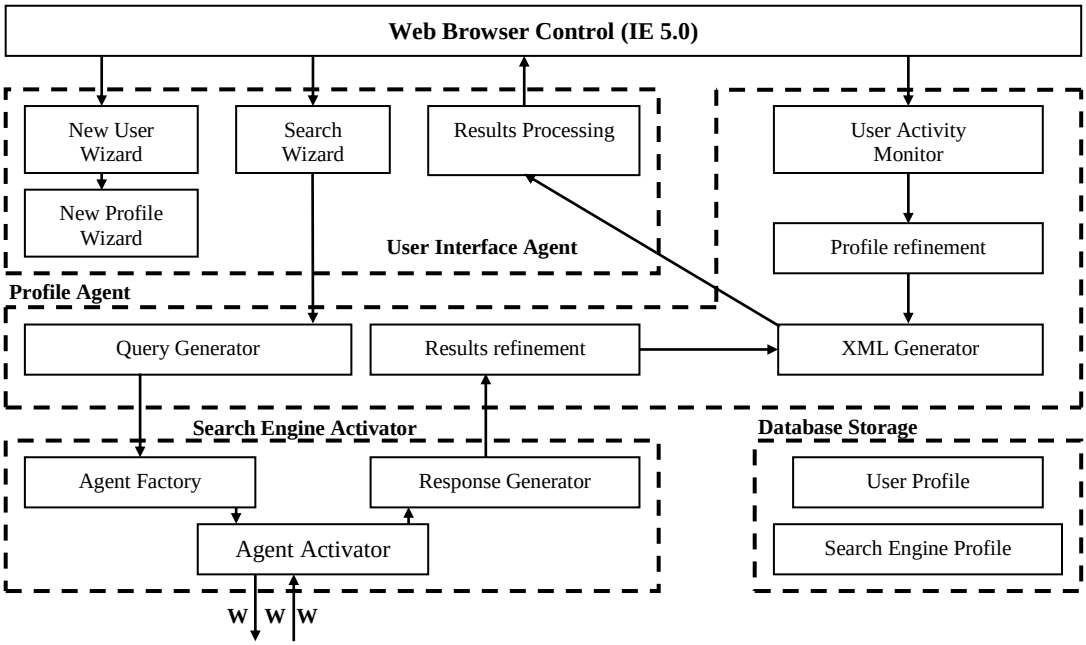

Fig.1: SEA overall architecture

where *tf(i)* is the number of times word d_i appears in document *W* (the term frequency), *df(i)* is the number of documents in the collection which contain d_i (the document frequency), *n* is the number of documents in the collection and tf_{max} is the maximum term frequency over all words *W*. To avoid over-frittering, accordingly to experiments described in [15], the optimum of used words is between 30 and 100. In our implementation, because the algorithm is used especially for recommendation based on the page's description and not for the content of the page itself, a range between 20 and 50 is sufficient. Once the top approximately 30 words have been picked we normalise *w* to be of a unit length to allow comparisons between documents of different lengths.

IV. ARCHITECTURE AND IMPLEMENTATION

At this time our implementation (SEA) is an example of a IR multi-agent system, helping the user manage the "information overload" problem often encountered when using a WWW. To undertake all kinds of above mentioned questions, our implementation [16] support the following issues: (1) assist the user in the diagnosis process and question reformulation; (2) select appropriate search engine for efficient searching accordingly to their profiles; (3) translate the question into one or more queries and search strategies acceptable to the given search engine; (4) manage searching strategy; (5) support the user in the results assessment; (6) support the user in resource description; (7) provide the user with the appropriate outputs in a suitable structure; and (8) advice he or she in the follow-up activity.

The entire system is composed by several modules (Fig. 1) which are shortly presented below.

The *Web Browser Control* has to present to the user the found results and the web pages.

The *User Interface Agent* deals with user input and shows the results. It consist in several sub-modules: *New User Wizard* assists the user in creating a new user for the system (it takes some basic information from the user which will be used later for deciding how many search process details will be hidden from him and which default parameters are to be used), *New Profile Wizard* that assists the user in the definition of a new interest profile (the collected information consist in the interest domains and relevant keywords - it is not necessary to provide the keywords but doing so will greatly improve the search process because the training period will be much shorter), the *Search Wizard* takes the request from the user and forwards it to the Profile Agent (some search parameters may be set in this wizard, also), and the *Result Processing* sub-module deals with various conversions needed in the presentation process.

The *Profile Agent* has to maintain the user profiles and use it in the search process. It's main tasks are to generate the query from the user request and to refine the user profile once the search process is completed. Thus, the *Query Generator* takes user requests and build generic queries used by the search engine agents (for this it use information from the user profile), the *Results Refinement* perform a classification of the founded links (this classification is done by taking the page ranking given by the search engines and the keywords from the query and user profile), the *XML Generator*, as its name suggest, generates a XML document with the sorted results and send it to the User Interface Agent, the *User Activity Monitor* collects information from the web browser (these consist in links followed by the user and where the search process had finish), the *Profile Refinement* updates the user profile (this is done by analysing the visited pages and extracting relevant keywords for the user).

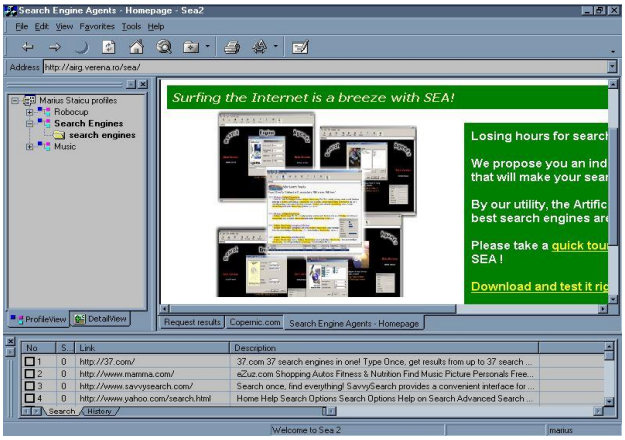

Fig. 2: SEA interface

Search Engine Activator will decide where to search the needed information. It selects between several search engines that are adapted for a specific domain. Here, the *Agent Factory* manage the available search agents (there is an agent who knows how to do an Internet search and it is cloned for each search engine to deal with it), the *Agent Activator* starts the agents, monitor their search progress and gather the results, and the *Response Generator* will filter the founded links (it deletes duplicate links and may check for death ones).

The *Database Storage* module handle the user profiles and the search engine profiles.

Our system is implemented using the Visual C++ 6.0 under Windows environment. As web browser we are using the Internet Explorer 5.0 (actually only the WebBrowser Control) because it has integrated parsers for HTML and XML (Fig. 2). The communication between the database storage and the other modules is done using XML documents.

V. CONCLUSIONS AND FUTURE WORK

Although the separated features have been treated separately by the current approaches, the current trends impose the need of an osmotic approach able to deal with heterogeneous resources. Compared with traditional search engines, SEA promotes a more anthropocentric orientation, improve data access capabilities and communication ability. In the future we will try to automate the degree of interest by measuring

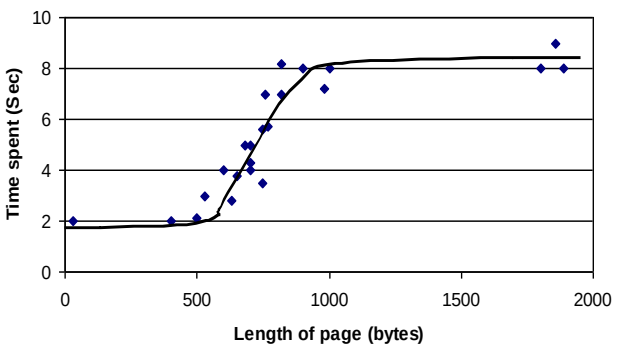

Fig.3: Reading time spent by the user (adapted from [17])

reading time spent by the user. Here, some promising research was been carried out for similar contexts (Fig. 3). At this time, we work at providing our tool with collaborative capabilities.

REFERENCES

[1] C. Moers, Zatocoding applied to mechanical organization of knowledge. *American Documentation*, 2: 20-32, 1951.

[2] W.B. Croft, What Do People Want from Information Retrieval?. *D-Lib Magazine*, November, 1995.

[3] C.B. Zamfirescu, Agent-based computing for information retrieval. *Technical Report IWAS-2000-02*, Klagenfurt University, Department of Industrial Informatics, Austria, 2000.

[4] A. Caglayan and C. Harrison. *Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents*, Wiley Computer Publishing, 1997.

[5] T. Finin, C. Nicholas, J. Mayfield. Software Agents for Information Retrieval. *Tutorial presented at ADL'98*, 1998.

[6] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation; *Commun. Of the ACM* 40(3): 66 – 72, 1997.

[7] L.N. Foner. Yenta: A Multi-Agent, Referral Based Matchmaking System. *Proceedings of the First Int. Conf. on Autonomous Agents*, Marina del Rey, California, ACM Press, 1997.

[8] B. Rhodes and T. Starner. The Remembrance Agent: a continuously running automated information retrieval system. *Proc. of The First Int. Conf. on The Practical App. of Intelligent Agents and Multi Agent Technology*, pages 487-495, London, 1996.

[9] H. Lieberman. Autonomous Interface Agents. *Proc. of the ACM Conf. on Computers and Human Interface*, Atlanta, Georgia, ACM Press, 1997.

[10] T. Saracevic, A. Spink, M. Wu. Users and Intermediaries in Information Retrieval: What Are They Talking About? *Proceedings of the Sixth Int. Conf. on User Modelling*, Vienna, 1997.

[11] N.J. Davis, R. Weeks, M.C. Revett. Information Agents for the World-Wide Web. In *Software Agents and Soft Computing. Towards Enhancing Machine Intelligence*, Springer Verlag, Berlin, pages 81-99, 1997.

[12] G. Salton and M.J. McGill. A*n Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[13] D. Harman. Overview of the third Text Retrieval Conference. In *Proceedings of the 3rd Text Retrieval Conference***,** Gaithersburg, 1994.

[14] M. Porter. An algorithm for suffix stripping. *Program*, 14(3): 130-137, 1980.

[15] M. Pazzani, J. Muramatsu, D. Billus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, 1996.

[16] C.B. Zamfirescu, M. Staicu, M. Luca. SEA: Search engine agents. In *Proceedings of the Int. Conf. Beyond 2000*, Sibiu, XXIX: 139-144, 1999.

[17] E. Horvitz. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the ACM Int. Conf. on Computer Human Interaction*, ACM Press, 1999.