ELSEVIER

Knowledge-Based Systems 13 (2000) 307–314

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Short Communication

An agent-based approach for integrating user profile into a knowledge management process

S. Pierre[a,*], C. Kacan[b], W. Probst[c]

[a]*Mobile Computing and Networking Research Laboratory (LARIM), Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montreal, Quebec, Canada H3C 3A7*

[b]*Public Technologies Multimedia, 1001, rue Sherbrooke Est, Montreal, Quebec, Canada H2L 4L5*

[c]*Department of Computer Science, Université du Québec à Montréal, C.P. 8888, Succ. Centre-Ville, Montreal, Quebec, Canada H3C 3P8*

Received 11 January 2000; accepted 31 July 2000

Abstract

This paper presents an approach based on a user agent to permit a number of users connected to distant machines to access different information sources in order to satisfy their requests. This user agent permits the simplification of the information search from distributed sources by making them transparent to the users. The agent considers the specific needs of each user during the search and responds with reference to their profile. It also permits the processing of one or more information requests by one or more users, as well as concurrent responses to each of them. Moreover, the agent provides its users with a measure of interaction, in order to enhance the quality and quantity of the results obtained. As a result, the agent is endowed with the ability to filter and refine the search, thus improving its service to the users.

Keywords: Information retrieval; User profile; Multi-agent system

1. Introduction

The progress attained in the domain of information technologies over recent years has produced a great expansion in the sources of information distributed throughout the world. The enormous quantities of information render the task of information research increasingly complex. It has therefore become necessary to provide users with knowledge management mechanisms to facilitate this task.

On the Internet, information derives from heterogeneous information systems and are presented under a variety of file formats. Several search tools are being used to find information (Yahoo, Alta Vista, Excite, etc.) some of which are more efficient than others. However, some of them do not allow users to reuse the results obtained in order to improve them. Others do not take into account the interests or the profiles of the users at the time of the information search.

In order to be efficient, a research tool must be able to filter information and refine the search in accordance with the profile of the user. This tool may be conceived as an agent capable of taking into account the individual needs of each user and of responding to one or several requests for information originating from one or several users. In addition, the agent should allow the users a certain interaction, so as to improve the quality and the quantity of the results obtained. To this end, it must be capable of filtering and refining the search in order to respond adequately to different types of users.

In general, a *user profile* consists of a set of specifications and characteristics which describe a particular user. In the framework of information search, this profile can represent the needs and interests of the user and its role is to direct the search according to those needs. Indeed, if the interests of the user of a given request are taken into account, the quality of the results yielded can be greatly improved upon as a function of the needs of this user.

This paper deals with the integration of the user profile into the process of the search for information. In this context, we discuss an agent-based approach which enables a number of users connected to distant machines to access information sources in order to fulfill their requests. Section 2 outlines previous work in this domain by showing the representation model called "vector space". Section 3 presents the spherical model, proposed as a means of

* Corresponding author. Tel.: +1-514-340-4711, ext. 4685; fax: +1-514-340-3240.

E-mail address: samuel.pierre@polymtl.ca (S. Pierre).

0950-7051/00/$ - see front matter
PII: S0950-7051(00)00090-3

integrating the user profile into the information search process for the purpose of filtering. Section 4 focuses on the implementation and application of this model.

2. Background and related work

In order to integrate the user profile element into the search context and thus improve the quality of the results, we will then analyze three approaches derived from this model [1]. Myaeng [7] defines three methods for using the information of the user profile in the course of a search: (i) the modification of the request by the profile; (ii) the processing of the request and the profile as two entities directing the search; and (iii) the filtering of the results corresponding to the profile. A large number of models and approaches based on these methods have been proposed and tested. The definition of the majority of them has been based upon the representation or *vector space* model [2,6]. In this model, the documents, requests, and user profiles are considered as a set of *vectors* or *points* in a space of n dimensions, where n denotes the number of terms t_i in a given source of information [10].

Several schemes have been proposed for the definition or the parameterization of the vector space. Indeed, the vectors or the points in this space must be well defined in order to be used as a basis for comparison or measurement of the similarity among them. It should be noted that these vectors represent documents, requests, or user profiles.

In the *vector space* representation, a document vector may be obtained from the set of terms which constitute this document. To support the set of documents of an information source S, each document must represent all the terms $t_i (1 \leq i \leq n)$ which can exist in this source. This source S is also considered as a vector of terms $t_i (S = (t_1, t_2, t_3, \ldots, t_n))$ such that for every $i = 1, \ldots n$, the term t_i belongs to a certain document D of the source S. In addition, to define the document vectors in this source, its terms t_i must be classified in a certain sequence (e.g. alphabetically). In this manner, the representation of a document D in the source will be as follows:

$$D = (d_1, d_2, \ldots, d_n)$$

This document vector D is of length n (same length as S), where each component d_j of this vector corresponds to the j^{th} term (t_j) of the source S. More precisely, if the term t_j of the source S does not belong to the document D, then d_j is equal to 0. If the term t_j of S belongs to the document D, then d_j will be equal to 1. This is one of the ways that has been proposed to represent the document vectors in a given source of information; this representation of *vector space* is qualified by "vector 0–1" and seems to be the simplest. In this way, a document vector D may be represented as follows: $D = (1, 1, 0, 1, \ldots 1, \ldots, 0)$.

On the other hand, Salton and McGill [10] have proposed another scheme which is very popular and more effective to represent the document vectors in the information source S. It consists of TF $*$ IDF (Term Frequency $*$ Inverse Document Frequency), where the component document vectors may have values other than 0 and 1. This scheme is based on the frequency of occurrence of the terms t_j of S in both the documents and the source.

The request and profile vectors Q and P of the user are obtained from their terms along with the weights assigned to them, which are supplied by the user in this case. Indeed, when the user sets up the request or profile, a certain weight may be assigned to each term relative to its level of interest. Therefore, to construct the request vector Q for example, all the terms t_i of the source S are examined. If the term t_i of S belongs to the request Q, then the value of the component q_i of request vector Q is equal to the weight assigned to this term by the user. If the term t_i of the source S does not belong to the request Q, then q_i will be equal to 0. The profile vector P of the user is constructed in the same way.

In the vector space representation model, the profile of the user (characteristics and interests) is seen as a vector of terms. Moreover, each term belonging to the profile has a certain weight p_i representing the interest of the user in this term. Similarly, regarding the document vectors previously described, the profile vector P is constructed in the following way: $P = (p_1, p_2, p_3, \ldots, p_n)$, where p_i denotes the weight of term t_i of S in the user profile:

$$p_i = \begin{cases} \text{Weight}(t_i) & \text{if the term } t_i \text{ belongs to the profile} \\ 0 & \text{otherwise} \end{cases}$$

This profile vector P represents the profile of the user and is used during the information search process.

Similarly, the user formulates a request composed of a set of terms and weights associated with these terms. The request vector Q is obtained in the same way as the profile vector P of the user. Q is defined as follows: $Q = (q_1, q_2, q_3, \ldots, q_n)$, where q_i denotes the weight of term t_i of S in the user request:

$$q_i = \begin{cases} \text{Weight}(t_i) & \text{if the term } \mathrm{t_i} \text{ belongs to the request} \\ 0 & \text{otherwise} \end{cases}$$

The source vector S is formed by a set of terms which exist in the documents contained in the source. Given that the number of these terms may be very large, this may significantly affect the performance of the search system as well as the quality of the results. Indeed, the source vector S will be the first vector to be set up and all other vectors will be constructed from it: document vectors, request vectors and profile vectors. Furthermore, all the vectors so obtained must be of the same size as the source vector S, according to the vector space model. In order to construct the source vector S, while reducing the number of its terms, two techniques have been used: the "stop" lists and the "stemming" algorithms [2]. Despite the efficiency of this latter technique in most cases, Korfhage [2] mentions that this type of algorithm consumes considerable machine time, particularly

when the quantity of information is great. In addition, the size of the documents may be reduced up to a maximum of 5%. For this reason, a compromise has been proposed which consists of applying this type of algorithm only to requests and user profiles.

Other approaches based on the vector space representation model have been proposed in order to improve the quality of the results of a search according to the information in each user profile. These approaches are largely inspired by the research of Myaeng and Korfhage [2–6] and Pazzani et al. [8,9].

3. Proposed hybrid search model

The hybrid approach proposed in this section consists of combining the approaches described above in a way that enables a user agent (UA) to carry out the search task with a quality of service corresponding to the needs of the user. The concept of quality of service refers principally to the quality of the results and the response time required. The basis for the new approach is to allow the utilization of the profile information to refine the results supplied by the models described in the preceding section.

3.1. Principles of the model and the user agent

We propose a hybrid model called the Spherical Model. According to this model, we begin by setting up the initial information describing the profile P of a user, who must fill out a specific form. This form records a set of information representing the interests, preferences, and the knowledge of the user. The user must also express the request to the UA as a set of terms that represent the desired information (needs of the user). A weight may also be associated with each of the terms of the request as well as the terms of the profile. This weight represents the interest of the user concerning the term in question. As a general rule, the weights of the terms in the request must have greater values than those of the terms in the profile.

Once this information becomes available to the UA (that is, request and user profile), the latter creates a new request composed of the terms of the request supplied by the user, as well as the terms of the user profile. This first task of the agent UA consists therefore of merging the set of request terms and profile terms into one new vector. The agent then creates the composite request vector $Q = (q_1, q_2, q_3, \ldots, q_n)$. This vector corresponds to the weights of the terms t_i of S in the composite and newly created request; $S = (t_1, t_2, t_3, \ldots, t_n)$ represents the set of terms t_i of the agent's base.

To find the set of documents relevant to the request in question, it is sufficient to determine the set of points found inside a sphere having as center Q, our composite request, and radius k. The value of k may be predefined by the UA as a constant, or it may be selected and modified by the user. This value k determines the size of the sphere in which the documents are to be subsequently found, and it determines directly the number of documents to be returned.

After this search step is carried out by the agent, the documents obtained are presented to the user who can evaluate them and assign a relevance value e_i to each one. If the user is not satisfied, the agent may reconstruct and update the composite request Q (original user request and profile of same) according to the evaluations e_i given by the user to the documents D_i. Then, it applies the spherical model again in order to filter the results already obtained from the first search. The agent UA is able to repeat this process of search refinement and of information filtering until the user is satisfied with the results.

3.2. Search/filtering algorithm

The information search/filtering algorithm proposed consists of nine main steps outlined as follows:

Step 1:

Create **m** documents, selected in a particular domain, and called Doc$i$, $i = 1, \ldots, m$.

This set of documents represents the base of pertinent documents of the UA. The base of documents of the agent UA contains: (Doc1, Doc2, …, Doc$i$,…,Doc$m$), where each document Doc$i$ ($i = 1, \ldots, m$) contains the fields Author, Editor, Title, Date, Size, Summary. These fields were chosen to represent the documents in the agent base. Also, proceeding from this set of fields, the user will be able to readily evaluate the relevance of each document returned as the result of a request.

Step 2:

Create the **m** term vectors of the documents Doc$i$, called TermDoc$i$, $i = 1, \ldots, m$.

In this step, the work of the UA consists of extracting the terms of each document Doc$i$ from the three fields Author, Editor and Title of the document. In the process, non-significant English and French words are eliminated, e.g. the, of, and, to, on, a, in, de, le, la, et, au dans, les, des, en, etc. This set of words has been chosen to act as the "stop list" mentioned in Section 2; we will refer to it as StopList. The **m** term vectors TermDoc$i$ obtained will therefore be of the following form:

TermDoc1: (term_{11}, term_{12},…,$\text{term}_{1\ n(1)}$;
TermDoc2: (term_{21}, term_{22},…,$\text{term}_{2\ n(2)}$;
…
TermDoc$i$: (term_{i1}, term_{i2},…,$\text{term}_{i\ n(i)}$;
…
TermDoc$m$: (term_{m1}, term_{m2},…,$\text{term}_{m\ n(m)}$.where $n(i)$ denotes the size of term vector TermDoc$i$ created.

Step 3:

Create the source vector of the terms called VecSource. This vector will be formed from the **m** term vectors TermDoc$i$ obtained in Step 2. The source vector thus

created must also be sorted in alphabetical order while eliminating all redundant terms, that is:

$$\text{VecSource} = (\text{term}_1, \text{term}_2, \ldots, \text{term}_n)$$

where n corresponds to the size of VecSource, which, on the average, is equal to m times the mean size of the term vectors TermDoc$i$.

Step 4:

Create **m** weight vectors of the documents TermDoc$i$, called WeightDoc$i$, $i = 1, \ldots, m$. These weight vectors are obtained from the term vectors TermDoc$i$ and the source vector VecSource. These vectors are defined as:

$\text{WeightDoc1} = (d_{11}, d_{12}, \ldots, d_{1n})$;
$\text{WeightDoc2} = (d_{21}, d_{22}, \ldots, d_{2n})$;
…
$\text{WeightDoc}i = (d_{i1}, d_{i2}, \ldots, d_{in})$;
…
$\text{WeightDoc}m = (d_{m1}, d_{m2}, \ldots, d_{mn})$.

The weights of the document terms are defined as d_{ij} with:

$$d_{ij} = \begin{cases} 5 & \text{if term}_j \in \text{TermDoc}i \\ 0 & \text{otherwise} \end{cases}$$

where m denotes the number of documents in the agent's base, n the size of the vector VecSource, i an index varying from 1 to m, and j an index varying from 1 to n.

Step 5:

Once the weight vectors WeightDoc$i$ of the documents are available, the agent then allows for two types of search in this step:

(a) Search from a request already known to the agent. When the user has previously saved a request, the agent can present a list of request identifiers from which the user is able to make a choice.

(b) Search from a new request formulated by the user. The user specifies the request and may also assign weights to its terms.The agent now creates the weight vector of the request, constructed from the source vector $\text{VecSource} = (\text{term}_1, \text{term}_2, \ldots, \text{term}_n)$ and the terms assigned by the user to the request. This vector, called WeightRequest or simply Q, is defined as $Q = (q_1, q_2, \ldots, q_n)$, such that, for $j = 1, \ldots, n$, we have:

$$q_j = \begin{cases} \text{weight}(\text{term}_j) & \text{if term}_j \in \text{Request} \\ 0 & \text{otherwise} \end{cases}$$

Step 6:

Search for the documents relating to the request Q among the m documents represented by WeightDoc$i$ in the base of the agent.

The documents considered pertinent to the request Q are found in the interior of a sphere having as center the request point $Q = (q_1, q_2, \ldots, q_n)$, and k as radius. This value k corresponds to a given constant and Q corresponds to the request vector WeightRequest obtained in Step 5. The value of k may be defined by default in the agent or selected by the user (recall that this value determines the number of documents to be returned). We apply the spherical model defined in Section 3.1 to the request Q. Thus, we calculate the Euclidean distance between each of the m documents WeightDoc$i$ and the request point Q represented by WeightRequest. These distances called Dist_i are obtained as follows:

$$\text{Dist}_i = \|\text{WeightDoc}i,\ \text{WeightRequest}\| = \left[\sum_{j=1}^{n} (d_{ij} - q_j)^2\right]^{1/2}$$

where $i = 1, \ldots m$, and n denotes the number of terms of the source vector VecSource. Note that this distance, used to find the similarity between the vectors or points in vector space, may also be obtained by other metrics, e.g. the one based on the cosine of the angle between the vectors.

Step 7:

Present the documents found in Step 6.

For $i = 1, \ldots m$, if $\text{Dist}_i \leq k$, the agent presents the document Doc$i$ created in Step 1 (details of the document: author, title, etc.), thereby allowing the user to evaluate each document presented in this step. These evaluations will be taken from the field e_i of each document presented. The value e_i varies between -5 and $+5$; it is chosen from a list consisting of these values in order to allow the user to evaluate the returned documents positively or negatively. Firstly, these positive and negative evaluations will be of great help in the filtering and refinement of the search. Secondly, the profile of the user will be created and updated based on these evaluations.

Step 8:

In this step, the user has the choice either of recording the search session if the results are satisfactory, or of filtering them. Thus:

(a) the user may record the current search session or request, providing a unique name or identifier for this request. In this case, the agent saves the name of the request and the WeightRequest vector Q in the user file;

(b) the user may also filter and refine the search. In this situation, the UA reformulates the request by calculating a new value for the vector Q, denoted by Q' and derived by means of the following formula:

$$Q' = Q + \sum_{i=1}^{nb \text{ of documents judged}} e_i^* \text{WeightDoc}i$$

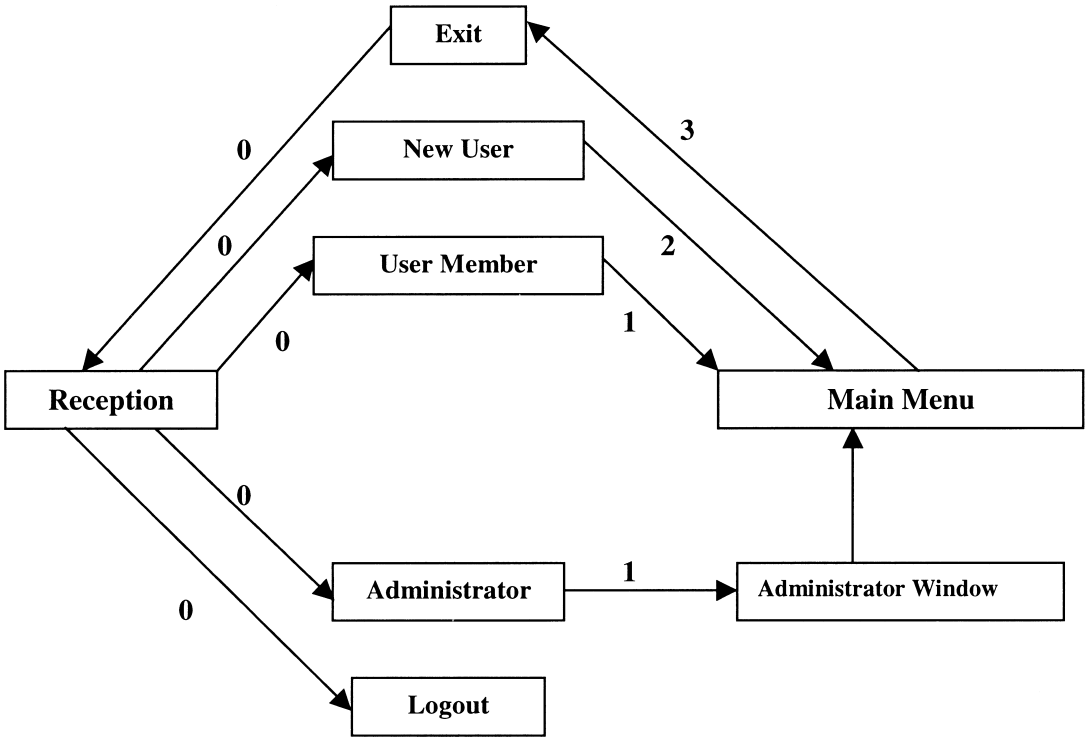

Fig. 1. Entry into the system and main menu of the interface of the UA.

This new request vector $Q' = (q'_1, q'_2, \ldots, q'_n)$ will be the weight vector of the user's new request. This vector will represent the user profile as well as the revised search request. This profile may be updated several times by the agent in order to satisfy the needs of the user.

Step 9:

In this last step, the user has the choice of ending the operation or of continuing the filtering process, that is:
(a) terminate the search; or
(b) refilter the results obtained up to this step, beginning again from Step 6 with a new request Q' created in Step 8.

4. Implementation and application

The UA integrates a client part and a server part. Each of these two parts is composed of a set of modules or classes which combine to carry out the work of the agent. The UA interface comprises a number of screens and windows implemented on client sites which allow users to interact directly with the agent.

Main Menu. Fig. 1 illustrates the way to enter the system as well as the main menu of the user interface. In this figure, a rectangle denotes the type of information supplied by the agent and presented to the user; the arrows between the rectangles indicate certain actions which can change the state of the agent. To initiate communication with the agent, the user must know the IP address or the name of the machine on which the agent is installed (server). In fact, the agent may be resident on any machine within the network and is capable of servicing a certain number of users who will eventually be distributed on different sites.

- *Reception.* Point of entry into the system.
- *Exit.* Point of exit from the system.
- *New User.* A user who wishes to register with the agent may gain access at this point. By selecting an access code and password, the agent performs the requisite validations and accepts or rejects the registration. In the case of refusal, an explanatory reason will be given.
- *User Member.* The users which are already members of the agent, may enter the system at this point by keying in their access code and password.
- *Administrator.* There is a single administrator for all administrative tasks. This is a user like all other registered members, but having access to a greater number of functions. Among other duties, the administrator controls the number of users allowed to register, in order to avoid overloading the agent with excessive requests and information maintenance. There may also be limitations set on the number of documents and registered user members may be eliminated at any given time.
- *Logout.* A user member who wishes to end the contact with the agent must supply access code and password to verify that these data correspond to said member. The

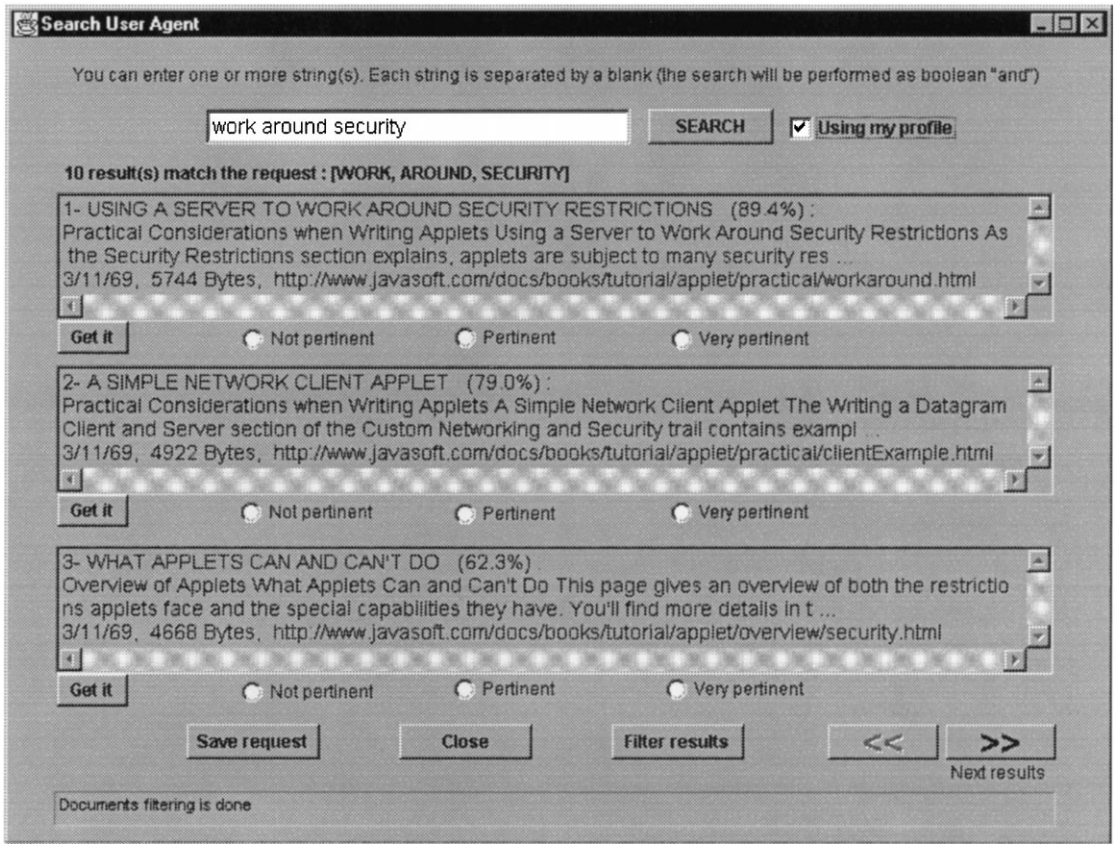

Fig. 2. Search window of the agent.

agent will then delete the member's account along with all associated information.

- *Main Menu.* When a member or a new user enters the system, the principal options of the main menu become accessible on the agent interface. We will first present the significance of each number shown on the arrows of Fig. 1, before providing the specifications of these different menu options.

The set of classes of the UA and its interfaces have been developed with the assistance of the following software tools: Operating System: Windows 95; Programming Language: Java; Development Tools: Symantec Visual Cafe 2.0 and Java Development Kit (JDK) 1.1.3 of Sun Microsystems.

The components implemented in the UA allow for various interaction scenarios: user registration and logout, administrative tasks done by the agent, simultaneous processing of one or more member requests, returning the most relevant results, integrating the user profile causing the same request to yield different results for different users, recording search sessions for subsequent usage, handling a request already known to the agent, and search filtering and refining based on user feedback.

We now consider a scenario in which the agent UA is active on the server machine; it is therefore ready to receive messages or information requests and to respond to them. In order to use the services offered by UA, that is, to conduct an information search, a new user referred to as U_x, must complete the following steps: Register with the agent; Send an information request to the agent; Filter the results returned; Record the search session; Submit a request already known to the agent; Refilter the results again and subsequently rerecord the search session.

Registering with the agent. In order to initiate an interaction with the agent, the user activates the *UserAgentClient* class, passing as a parameter on the command line the IP address of the machine where the agent is installed. In this way, the *UserAgentClient IP* command allows the client part of the agent to establish a connection with the server.

Sending an information request to the agent UA. Once registration has been successfully achieved, all agent functions become accessible to the user.

Fig. 2 shows a window reflecting an actual search performed by the agent and the results consequently obtained. A single iteration of filtering and refinement was done in this search, resulting from our evaluation of the documents obtained in the first pass. The desired document was obtained (in this first iteration), as the second result and with a 79% degree of pertinence. Fig. 3 shows the window that allows users to supply information relevant to their profile.

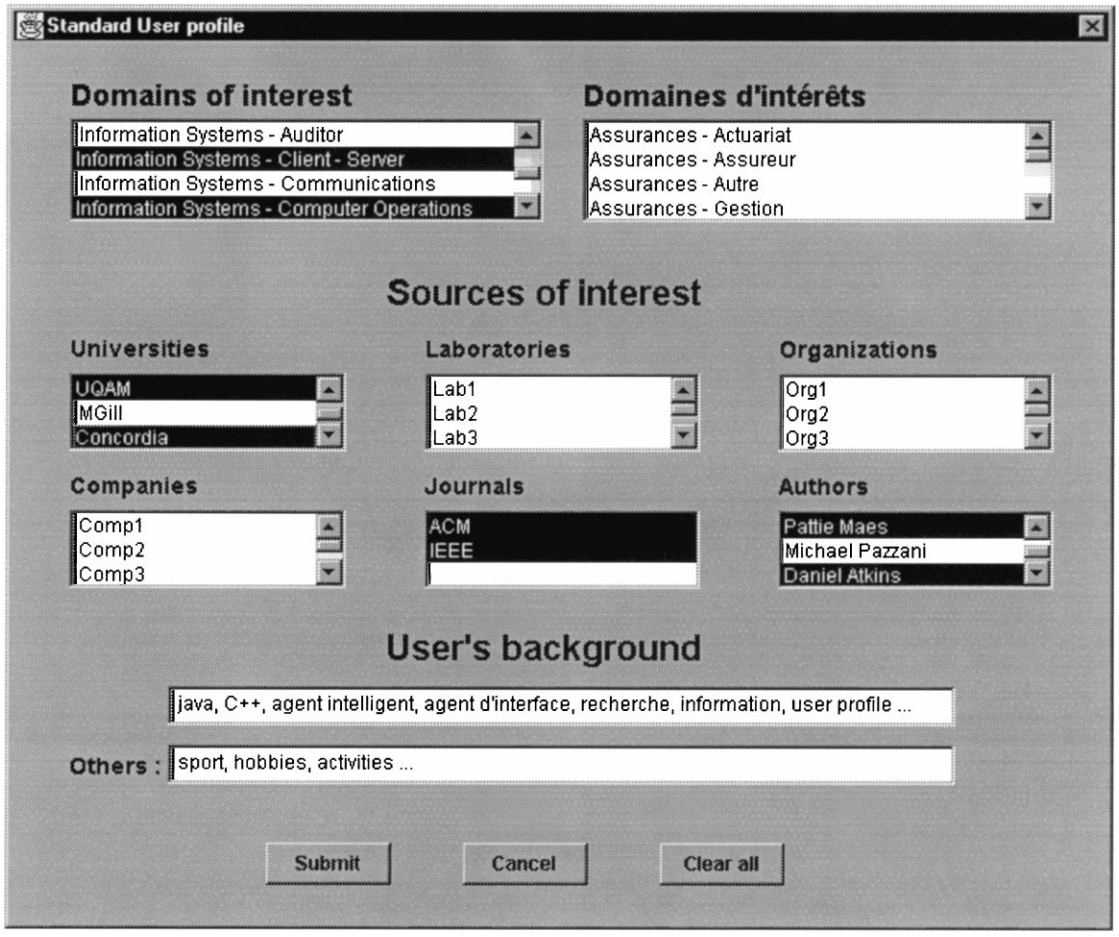

Fig. 3. User profile information window.

Filtering the results obtained. Should a user wish to refine the results further, evaluations must be assigned to the documents presented, followed by a prompt to the agent to refilter them. The agent will then apply the information-filtering algorithm described in Section 3 to the results of the request. More specifically, the agent will search for documents similar to those considered pertinent by the user, while eliminating those judged irrelevant. The new results will then be returned to the user, who may elect to repeat this operation as many times as necessary until the results are satisfactory. Our tests with the agent indicate a high degree of filtering efficiency and ability to return results which best correspond to user needs, even after only one or two iterations of search refinement.

Recording the search session. When satisfactory results have been obtained, the user may want to save this search for subsequent utilization. In this case, the user may request that the search session be recorded along with all related information. The agent can then attach the details of the search and its results to the user file.

Submission of a request already known to the agent. The user has the option to make a search by submitting a new request to the agent, or by using a request previously recorded in the agent.

Refiltering the results. When the results of a request are obtained, the user may have them refiltered and refined, and then have the search session recorded again.

5. Conclusion

This paper has described the elaboration and the implementation of an approach based on a UA which permits a number of users connected to distant machines to access different information sources in order to satisfy their requests. The UA as a developed permits the simplification of the information search from distributed sources by rendering them transparent to the users. The agent considers the specific needs of each user in the course of the search and responds with reference to their profile. The agent also permits the processing of one or more information requests by one or more users, as well as concurrent responses to each of them. Moreover, the agent provides its users with a measure of interaction, in order to enhance the quality and quantity of the results obtained. As well, the agent is endowed with the ability to filter and refine the search, thus improving its service to the users. Finally, all the administrative work related to this agent can be

accomplished remotely, that is, the administrator may be connected to the agent from any location in order to perform the required administrative tasks.

Certain elements currently implemented in the agent need to be reconsidered in future so as to improve the architecture and the services offered by the agent. First, the methods of exclusion and withdrawal of the members should be revised to avoid the loss of information collected for these users. It would also be desirable for the agent to integrate a more complex model of user profiles and to store them for a certain period after logout for recovery purposes, should re-registration take place within a reasonable delay. Furthermore, when testing will be carried out with a greater number of users as well as a larger information base, the indexation algorithm will undoubtedly require revision. By integrating concepts such as those currently used in existing search engines on the Internet, a greatly enhanced expression of information requests would result. Finally, the implementation of UAs might be reviewed in order to provide for their distribution onto different servers.

References

[1] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison Wesley/ACM, New York, 1999.

[2] R.R. Korfhage, Library of Congress Cataloguing-in-publication Data, Information Storage and Retrieval, Wiley Computer Publishing, New York, 1997.

[3] R.R. Korfhage, Information Retrieval in the Presence of Reference Points, Part 2, School of Library and Information Science, University of Pittsburgh, LIS034/IS91002, Pittsburgh PA 15260, March 1991.

[4] R.R. Korfhage, Information Retrieval in the Presence of Reference Points, Part 1, School of Library and Information Science, University of Pittsburgh, LIS001/IS88001, Pittsburgh PA 15260, January 1988.

[5] R.R. Korfhage, Query Enhancement of User Profiles. Research and Development in Information Retrieval, Cambridge University Press, Cambridge, MA, 1984, pp. 110–122.

[6] S. Myaeng, R.R. Korfhage, Integration of user profiles: models and experiments in information retrieval, Information Processing and Management 26 (6) (1990) 719–738.

[7] S.H. Myaeng, The roles of user profiles in information retrieval, PhD Dissertation, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas, 1987.

[8] M. Pazzani, D. Billsus, Learning and revising user profiles: the identification of interesting web sites, Machine Learning 27 (1997) 313–331.

[9] M. Pazzani, J. Muramatsu, D. Billsus, Syskill and Webert: Identifying Interesting Web Sites, Department of Information and Computer Science, University of California, Irvine, CA 92717, 1997.

[10] G. Salton, M. McGill, Introduction to Modern Information Retrieval, Computer Science Series, McGraw-Hill, New York, 1983.