Available online at www.sciencedirect.com
SCIENCE DIRECT®
Cognitive Systems Research 5 (2004) 207–222
Cognitive Systems RESEARCH
www.elsevier.com/locate/cogsys

Interface agents personalizing Web-based tasks

Action editor: Ning Zhong

Daniela Godoy [a,b,*], Silvia Schiaffino [a,b], Analía Amandi [a]

[a] *ISISTAN Research Institute, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires, Campus Universitario, Paraje Arroyo Seco, B7001BB0, Tandil, Argentina*
[b] *CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina*

Received 11 March 2003; accepted 30 March 2004
Available online 24 June 2004

Abstract

The volume of information available on the Web is constantly growing. Due to this situation, users looking for documents relevant to their interests need to identify them among all the available ones. Intelligent agents have become a solution to assist users in this task since they can retrieve, filter and organize information on behalf of their users. In this paper we present two experiences in the development of interface agents assisting users in Web-based tasks: *PersonalSearcher*, a personalized Web searcher, and *NewsAgent*, a personalized digital newspaper generator. The main challenge we faced to personalize the tasks carried out by these agents was learning and modeling specific and dynamic user interests. Our proposed approach consists of incrementally building a hierarchy of users' relevant topics and adapting it as agents interact with users over time.

Keywords: Intelligent agents; User profiling; Interface agents; Personalization

1. Introduction

The explosive growth of the WWW has originated several problems to users trying to find relevant documents in the vast amount of information available. As an example, suppose a user who is trying to find interesting Web documents using a traditional search engine. Such a user will obtain a ranked list of documents containing some interesting ones, but also a lot of uninteresting documents. Then, he will have to browse them all to identify the relevant documents. The same happens to a user who is trying to find relevant news in on-line newspapers. He will have to read several articles within various newspapers, probably belonging to different sections due to the organization of the newspaper, until he finds what he wants.

In this context, the main problems these users have to face are the effort and the time they have to

* Corresponding author. Tel./fax: +54-2293-440363/62.
E-mail address: dgodoy@exa.unicen.edu.ar (D. Godoy).

1389-0417/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cogsys.2004.03.003

spend looking for relevant information. In spite of this effort, they sometimes cannot reach some relevant documents due to the search engines mechanisms and the organization of the information on the Web.

Interface agents have become a viable solution to these problems. These agents act as intelligent assistants to users with computer-based tasks since they can learn users' habits and preferences to personalize the execution of these tasks. However, learning users' preferences is not a straightforward issue since they can be hard to acquire and express and they can change over time. Besides, the assistance an agent can provide to a user depends on what and how much this agent can learn about the user.

We have carried out two experiences developing interface agents to assist users in performing Web-based tasks. The agents we have developed perform a number of tasks on behalf of their users, such as finding, filtering and organizing relevant Web documents according to users' preferences. *PersonalSearcher* is an interface agent that helps users who are searching the Web for relevant information by filtering a set of documents retrieved from several search engines according to users' interests. *NewsAgent* is an interface agent that generates personalized digital newspapers selecting those articles that are relevant to a user from several on-line newspapers.

To acquire information about users' interests, *PersonalSearcher* and *NewsAgent* observe users' behavior while they are reading Web documents, recording the main features characterizing these experiences. The goal of these agents is to detect not only general interests such as *sports* and *economy*, but also the sub-topics of these interests that are relevant to a given user. Our agents have to be able to detect, for example, that a certain user is interested in documents talking about a famous football player and not in *sports* or *football* in general. To tackle this problem, we propose an approach to model user interests in a user profile and adapt it over time considering specific and dynamic interests. We used this approach to develop *PersonalSearcher* and *NewsAgent*.

The work is organized as follows. Section 2 presents an overview of the agents we have mentioned. Section 3 describes what a user profile is according to our approach and its construction is explained in Section 4. Experimental results are summarized in Section 5. Section 6 describes some related works. Finally, Section 7 presents our conclusions.

2. Web information agents' overview

Two tasks in which agents can be helpful are searching the Web for relevant documents and looking for interesting articles in on-line newspapers. Both tasks demand time from the user and can become repetitive and tedious. *PersonalSearcher* and *NewsAgent* assist users in these tasks and they both share the same basic functionality that is illustrated in Fig. 1.

Both agents observe users' behavior while they are navigating the Web and record information about the read documents. The information recorded during this process includes the characteristics of the Web documents as well as data about the reading experience. Each agent uses the information obtained from observation to build a user profile, which is a representation of a user's topics of interest. For our information agents, a user profile consists of a set of weighted topics relevant to a user, which is obtained using the approach we will explain in Section 4. Our agents use the acquired profiles to provide personalized assistance to users. Profiles are refined over time according to the positive or negative feedback users provide regarding agents' suggestions.

2.1. PersonalSearcher

PersonalSearcher (Godoy & Amandi, 2000) is an intelligent agent that assists users in finding interesting documents on the Web. The agent carries out a parallel search in the most popular Web search engines and it filters the resultant list of Web pages according to a user's interests.

Search engines are the most widely spread tools for searching Web pages. However, the low levels of precision in the answers of these engines lead users to dedicate a considerable amount of both time and effort to browse a ranked list of

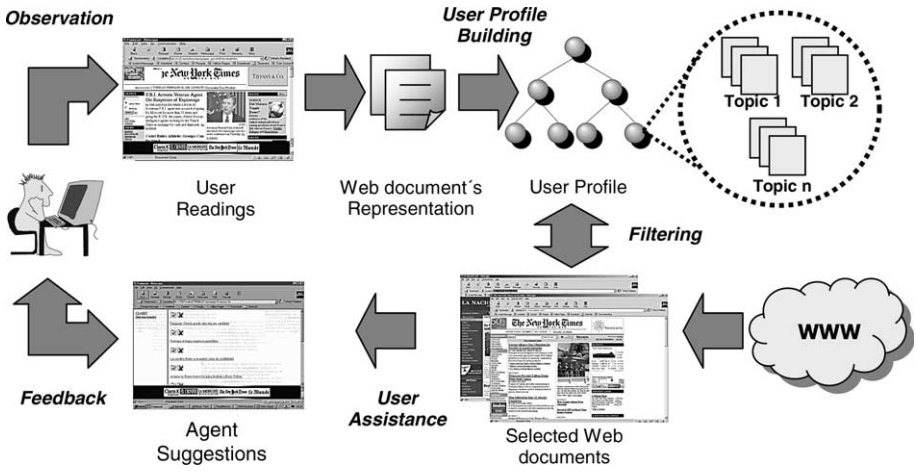

Fig. 1. Information agent's overview.

documents. Generally, this list contains a lot of uninteresting documents and just a few relevant ones.

PersonalSearcher helps users to identify a reduced number of documents with high probability of being relevant to them by contextualizing the search process according to users' information preferences. Each agent, instance of *PersonalSearcher*, learns about a user's interests from the observation of user browsing on the Web.

Each instance of this agent monitors the Web activity of its associated user in order to collect interesting documents. Expecting the user to manually give relevance judgments about each visited Web page is not a desirable characteristic for an agent, since it is a time-consuming task that increases the cognitive load on the user. Because of this reason, *PersonalSearcher* unobtrusively observes user browsing behavior in order to approximate the degree of user interest on each visited Web page. In order to accomplish this goal, for each reading in the standard browser the agent observes a set of implicit indicators in a process known as implicit feedback (Oard & Kim, 1998).

Implicit interest indicators used by *PersonalSearcher* include the time consumed in reading a Web page (considering its length), the amount of scrolling in a page, and whether it was added to the list of bookmarks or not. A number of studies has already proved that both reading time and scrolling have a strong correlation with explicit interest feedback, as opposed to the number of mouse clicks, which does not result in a good indicator (Morita & Shinoda, 1994; Claypool, Le, Wased, & Brown, 2001; Kelly & Belkin, 2001). The inclusion of a Web page to the list of bookmarks is also considered as a strong indicator of user interest as it supposes a user intention to return to the page in the future.

By a combination of implicit indicators, the agent estimates the user interest in each visited Web page. Based on the Web pages considered as interesting for a user, the agent incrementally builds and adapts a user interest hierarchy constituting the user profile.

To find interesting documents, users interact with their *PersonalSearcher* agent expressing their information needs by keywords as usual. In turn, the agent posts this query to a set of the most popular search engines, such as *Google* and *Yahoo!*, getting a set of documents that covers a wide portion of the Web.

PersonalSearcher determines the convenience of suggesting a document by computing its relevance degree regarding a user's interests. Only documents that surpass a given relevance threshold are sent back to this user as a result to his query. Users can customize the relevance threshold through the user interface provided by *PersonalSearcher* within an interval from 0 (all documents will be

recommended) to 1 (only documents with the maximum relevance will be recommended).

Once the agent has presented a set of documents as a result to a given query, the user is again observed. This new observation can produce adaptations in the user profile in terms of the user's approval or disapproval to the agent's suggestions. Fig. 2(a) shows two screenshots where we can observe a Web search and the list of Web pages suggested by *PersonalSearcher*.

2.2. NewsAgent

NewsAgent (Cordero, Roldán, Schiaffino, & Amandi, 1999) is an interface agent that assists users who periodically read on-line newspapers. These users generally concentrate their attention on those news reports and articles belonging to topics they like to read because of their hobbies and interests, and also on those they need to read because of their work. Although news in digital newspapers are organized into sections and links to the main news can be found in the main page, the user has to navigate through several pages until he reaches the page he wants to read. Besides, the organization of the paper can sometimes place a news report in a section where the user probably does not reach it because he is looking for it in a different section. For example, an article about a conflict between Uruguay and Brazil can be placed in the *international* section, but also in the *economy* or *politics* sections if there are consequences of this conflict affecting our country, Argentina.

NewsAgent has the capability of building a personalized newspaper containing only those news the user wants to read. To achieve this goal, the agent has to discover a user's interests in the news domain, which can be quite specific. For example, a user can be interested in *basketball* news, particularly in those about the *NBA*. More specifically, the user can be interested in reading news about *San Antonio Spurs* and *Emanuel Ginobili*. This user would prefer a newspaper containing only *San Antonio Spurs* related news, instead of one containing all *sports* or *basketball* news, as offered by some digital newspaper services.

In order to detect the topics each user is interested in, *NewsAgent* monitors users' behavior while they are reading newspapers on the Web and it records information about the different articles they read and some indicators about their relevance to the user. These data include the name of the newspaper, the section the article belongs to and an estimated user interest in the article. The user can optionally provide information about his

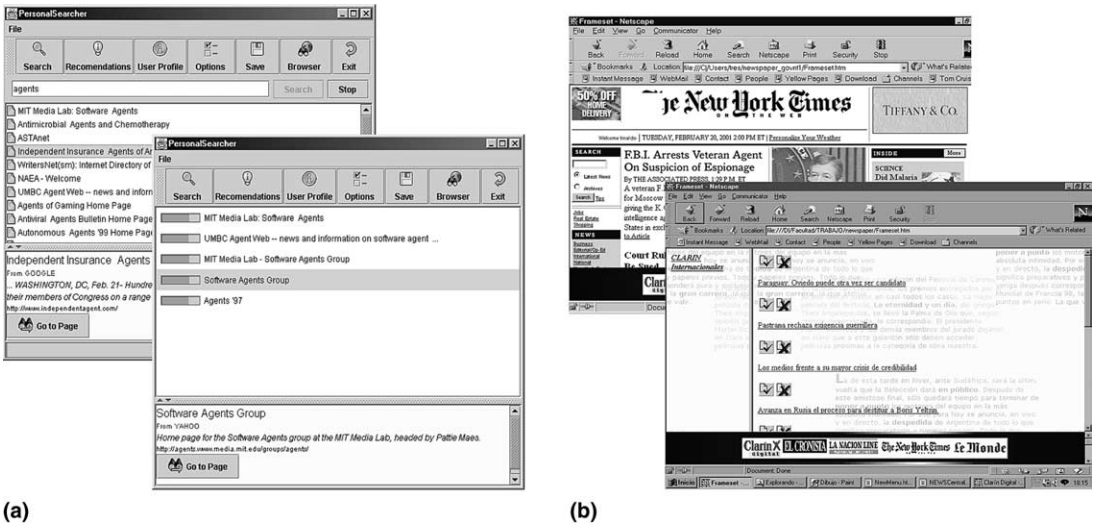

Fig. 2. Web information agents screenshots: (a) *PersonalSearcher* screenshots; (b) *NewsAgent* screenshots.

interests (such as sections in the paper or general topics such as *football*) through a user interface, and he can also rate pages as interesting or uninteresting while he is reading. Based on this explicit information, the agent can generate a newspaper in initial learning stages.

Once the agent has gathered information about the news the user is interested in, it builds a hierarchy containing the different topics these news talk about. This hierarchy is built incrementally as new information is obtained from observation. The top levels of the hierarchy contain information about general topics such as *sports* and *economy*. As we move down, we will find more specific topics such as *basketball* and even *San Antonio Spurs*.

NewsAgent searches several newspapers selecting news according to the user profile, filtering out those news that do not match the user's interests. The personalized newspaper is built by including those news having high probability of being relevant to the user. The relevance of each article is determined by considering the same parameters that *PersonalSearcher* takes into account. Once the agent has generated a personalized newspaper, the user can provide some explicit feedback regarding the relevance of the suggested news. The agent also observes the user's actions in order to capture some implicit feedback, such as the user reading or not reading the news the agent suggested. The user feedback is used to refine the knowledge the agent has about the user. Fig. 2(b) shows two screenshots, one belonging to an on-line newspaper a user is reading and the other belonging to a personalized newspaper generated by *NewsAgent*.

3. Representing users' interests

In order to assist a user with Web-based tasks, agents have to build a user profile, which is a representation of a user's interests. Agents can learn about these interests from different sources: asking users to provide them explicitly, inferring them from users' actions and feedback, or interacting with other agents (Maes, 1994). Since our agents act in isolation and they do not want to disturb users from their activities to ask about their interests, they learn about a user by watching his behavior while he is performing Web-based tasks.

During observation, our agents determine users' interests by identifying the different types of documents a user generally reads. In this regard, users may have diverse information interests related to, for example, their hobbies (e.g. *sports*) or their work (e.g. *programming*), which have to be modeled into separate categories. In addition to modeling multiple interests in several domains, profiles also have to model the different abstraction levels of such interests. Thus, a natural way of organizing such interests is through a hierarchy where categories at the top level represent broad interests and those at the bottom level represent particular aspects of them. A hierarchical organization of user interests enhances the semantic of user profiles as it is closer to the human conception of a set of interests.

A pre-defined hierarchy of topics is not suitable for personal agents dealing with unpredictable subject areas for several reasons. First, these static hierarchies contain general categories trying to cover a wide population of users, but they cannot guarantee the representation of the interests of every user. Second, they cannot reach high levels of detail regarding the topics they can contain because there are a potentially infinite number of aspects within the same topic. Finally, even when this kind of hierarchies may be constructed, it would be difficult to maintain them up to date. In the *NewsAgent* domain, for example, a static hierarchy containing the sections of newspapers as topics may be feasible. However, sub-topics within these sections cannot be expected to remain constant in time and, consequently, cannot be statically defined.

Our proposed solution to deal with the problems described above is to dynamically build a personalized hierarchy for every user interacting with an agent. This hierarchy is built starting from reading experiences recorded while agents observe users' behavior. Fig. 3 shows an example of a user profile showing a topic hierarchy with several levels that can be built with our technique.

Under the assumption that documents talking about the same topic have similar contents,

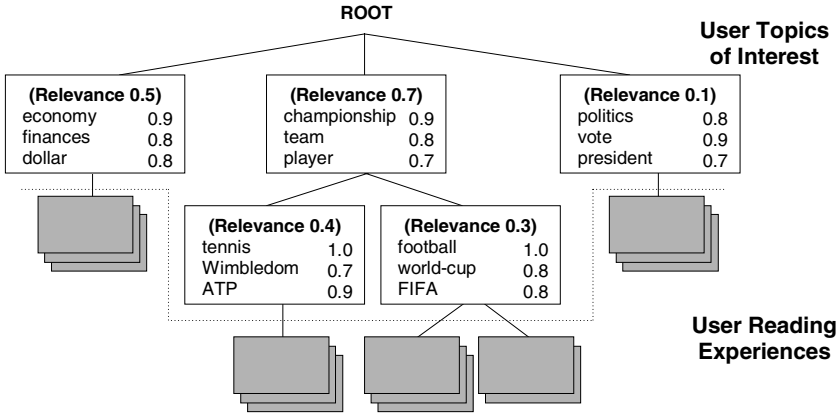

Fig. 3. An example of a topic hierarchy built by our agents.

reading experiences are grouped to identify different interests. Thus, the lowest levels in the hierarchy hold users' reading experiences grouped by content similarity. Each group represents a topic of interest within a more general one. An example of such a topic is the interest of a user in the current champion football team. The construction of the hierarchy starts from the reading experiences at the bottom levels and it continues upwards defining the topics contained in each node.

Each internal node in the hierarchy represents a topic of interest for a user, which is defined by a set of representative words. These topics are gradually discovered by the algorithm starting from the more general to the most specific ones. For instance, in the previous figure an agent first discovered the topic *sports*, represented by the words *championship*, *team* and *player*. Below *sports*, experiences were divided into two different groups, allowing the agent to identify two new sub-topics *football* and *tennis*. Below both sub-topics, reading experiences can be further divided into one or more groups, which in turn can become new sub-topics. In the case of *tennis* only one group holds all pages read by the user thus far, whereas in the case of *sports* two different types of readings can be identified and are expected to generate two new sub-topics in the future.

In a topic description, words are associated with a degree of importance. In the definition of *sports*, for example, the word *championship* is considered more representative than the word *player*. Child or sub-topic nodes in the hierarchy share the words included in their parent node and they contain more specific ones describing themselves. For example, the words *football*, *world-cup* and *FIFA* define the sub-topic *football*, whereas the words *tennis*, *Wimbledon* and *ATP* define the sub-topic *tennis*. Topics without those words live in or below the sibling nodes.

We have developed a technique that enables agents to incrementally build and adapt this kind of topic hierarchies. This technique is based on Textual Case-Based Reasoning (TCBR), a specialization of Case-Based Reasoning (CBR) for textual documents. CBR is a problem-solving paradigm that reuses solutions of previous experiences, which are named cases (Kolodner, 1993). In this context, users' reading experiences are represented as textual cases, which hang on leaf nodes in the hierarchy.

We consider that the most important issues when learning and modeling Web users' interests are obtaining them dynamically, representing them in a way that allows an easy manipulation and adapting them to reflect their changes over time. In the following sections we describe our approach to solve these issues.

4. User profile building

In this section we explain the user profiling technique we have developed, which is based on

TCBR. CBR is a technique that solves new problems by remembering previous similar experiences. It has been used in several areas, including information retrieval and filtering (Lenz, Hübner, & Kunze, 1998; Smyth & Cotter, 1999). Particularly, TCBR is applied to those situations where experiences are given by textual documents (Lenz, 1998).

A case-based reasoner represents problem-solving situations as cases. Thus, a new problem is solved by remembering a previous similar situation (previous cases matching the current problem) and by reusing information and knowledge of that situation. In an interpretative approach, CBR is applied to accomplish a classification task, i.e. find the correct class for an unclassified case. The class of the most similar past case becomes the solution to the classification problem.

In our technique, cases represent relevant documents read by a user on the Web. Each case records the main characteristics of a document, which enable the reasoner to determine its topic considering previous readings. The topics of documents represent user interests and they constitute cases' solutions. Therefore, a topic or user interest category is extensionally defined by a set of cases sharing the same solution.

Using this approach, previous read documents can help to categorize new ones into specific categories, assuming that similar documents share the same topics. In the next sections we explain how textual cases are obtained from Web documents, how the similarity between them is measured and how we can build a topic hierarchy starting from them.

4.1. Representing reading experiences

Cases represent specific and contextual knowledge that describes a particular situation (Kolodner, 1993). In our domain, an experience or case describes a particular situation denoting a user interest, such as the reading of a Web page that was considered relevant to the user. A case has three main parts: the description of the situation or problem, the solution, and the outcome or results of applying the solution to the problem.

In the chosen application domains, the description of the situation includes the URL (Uniform Resource Locator) of the document, the date and time when the user read this page, a general category the document belongs to within the topic hierarchy, a set of representative words, and the perceived user interest in the Web page (that was calculated by the agent based on implicit interest indicators). The solution is a code number that identifies a topic of interest for a user and, finally, the outcome describes the user's feedback.

For representing Web page contents, we use vector representations where pages are identified by feature vectors in a space in which each dimension corresponds to a distinct term associated with a weight indicating its importance. The following term-weighting function is applied to calculate these weights:

$$\text{weight}(w_j, d_i) = tf_{ij} + \Delta_{ij}, \tag{1}$$

where tf_{ij} is the frequency of a word w_j in the document d_i and Δ_{ij} is an additive factor defined in terms of several word characteristics in the Web page. The value of Δ_{ij} is calculated taking into account the word location inside the HTML document structure (e.g. words in the title are more important than words in the document body) and the word style (e.g. bold, italic or underlined). For example, for words between the tags <TITLE> and </TITLE>, $\Delta_{ij} = 4$; for words in the headings, $\Delta_{ij} = 2$; and for words in the document body, $\Delta_{ij} = 0$.

Previous to the document representation as cases, non-informative words are removed using standard stop-word lists and a stemming algorithm is applied to the remaining words (Porter, 1980).

A case also registers the user feedback to actions carried out by the agent based on the knowledge it provides about the user interests. Basically, cases hold the proportion of relevant and irrelevant suggestions an agent has made based on them. This information allows agents to increase or decrease the user interest in each collected case to decide about future actions or, eventually, determine when a case is no longer valid. A case is eliminated from a user profile if either it was not used within a configurable time

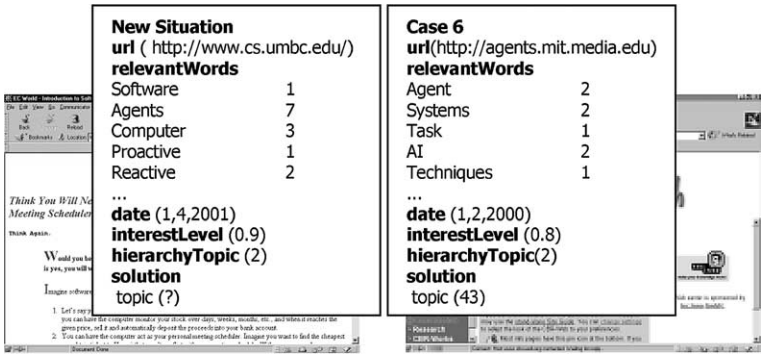

Fig. 4. Case representation of Web documents.

window or its level of interest falls (because of negative feedback) below a certain minimum value which allows the agent to have confidence in suggestions based on the case.

In Fig. 4 we can observe the resulting textual case representation of two Web pages. The page located on the right side in this figure has a defined topic as its solution, since it is a case already in the profile. Instead, the page on the left needs to be compared with others to determine the topic it belongs to.

4.2. Grouping similar Web documents

A group of cases with the same value as case solution extensionally defines a topic of interest within a user profile. As mentioned in the previous section, the solution of a textual case is assigned according to its similarity with other cases already classified. In other words, cases are clustered according to its similarity with past experiences defining user interest categories or topics.

The topic a new experience belongs to is determined by following a k-nearest neighbor (k-NN) approach. The fundamental assumption behind this approach is that cases which are close in the feature space according to an appropriate similarity measure belong to the same category. In this way, a new case is assigned to the cluster to which the closest case belongs to, established by comparison with every case in the category. An experience not close enough to any other case in the current clusters given a predefined similarity threshold, originates a new singleton cluster.

Comparison of cases is performed through a number of dimensions that describe them. A similarity function is defined for each of these dimensions, being the most important the one that measures the similarity between the relevant words lists. The dominant similarity measure in text classification is the cosine similarity that can be calculated as the normalized dot product (Salton & McGill, 1983):

$$\mathrm{sim}_{\text{word-lists}}(l_i, l_j) = \frac{\sum_{k=1}^{r} w_{ik} * w_{jk}}{\sqrt{\sum_{k=1}^{r} w_{ik}^2} * \sqrt{\sum_{k=1}^{r} w_{jk}^2}}, \qquad (2)$$

where l_i and l_j are the respective lists of words, w_{ik} and w_{jk} the weights of the word k in each list and r is the amount of different words in the lists.

Additional dimensions can be given by the application domain. For example, in *NewsAgent*, the degree of similarity across the *section* dimension is obtained by computing the similitude in a qualitative scale of the sections two pieces of news belong to. For news appearing in *The New York Times*, an article belonging to the *pro-football* section is quite similar to an article belonging to the *football* section, but it will be less similar to an article belonging to the *hockey* section, and it will be completely different from those news in the *space and cosmos* section.

A numerical evaluation function that combines the matching of each dimension with the importance value assigned to that dimension is used to obtain the global similarity between the entry case C_E and the retrieved one C_R. The function used in our technique is the following:

$$S(C_E, C_R) = \sum_{i=1}^{n} w_i * \mathrm{sim}_i(f_i^E, f_i^R), \quad (3)$$

where w_i is the importance of the dimension i given by the agent designer, sim_i is a similarity function for this dimension (when the dimension being compared is the list of words of both cases, sim is computed using Eq. (2)), and f_i^E, f_i^R are the values for the feature f_i in both cases (when Eq. (2) is being used $f_i^E = l_i$ and $f_i^R = l_j$). If the similarity value obtained from S is higher than a given threshold δ, the cases are considered similar, and then, we can conclude that both cases are in the same user interest topic.

The similarity threshold δ impacts on two aspects of the resulting clustering solutions, the total number of clusters and the cluster homogeneity. A value of δ near to 0 will produce a very few clusters grouping even highly dissimilar cases; while a value of δ near to 1 will produce a big number of clusters. In both situations, the technique accuracy to discover interest topics is affected. Because of this reason, we tunned the δ parameter by analyzing the quality of clustering solutions of k-NN under variations of δ based on the number of resultant clusters and a measure of cluster homogeneity such as Entropy (Shannon, 1948), using several text collections. [1] Labeled training data were used for entropy calculation but not during clustering, expecting clusters generated by our technique to be as homogeneous as possible with respect to the labeling of their containing cases. Through experimentation we determined that the optimal value for δ threshold is 0.3, where a compromise between homogeneity and number of clusters can be achieved.

4.3. Defining user interest topics

A user interest hierarchy needs to be built starting from scratch. In order to accomplish this goal, as soon as new cases appear describing user interests, they are clustered by similarity in the profile, identifying user interest topics. Afterward, descriptions for such topics are automatically gleaned by observing the characteristics cases in a topic have in common, as well as the ones a novel case should also have in order to belong to this topic or category.

The hierarchy of increasing specificity used to organize the topics a user is interested in, is composed of an arbitrary number of categories or topics, denoted by $T = \{t_1, t_2, \ldots, t_n\}$, which are gradually discovered as new cases become available. In order to automatically assign cases to categories, our technique associates to each of them a description given by a set of terms weighted according to their importance in the topic description. An interest hierarchy could be seen like a tree. Each internal node in the tree holds features shared by their child nodes and the cases below it, acting as a classifier to the category.

A classifier for a category is composed of a function $F_i : d_j \to [0, 1]$ that, given a document d_j, returns a number between 0 and 1 that represents the evidence for the fact that d_j should be classified under category t_i. This function also has a threshold τ_i such that $F_i(d_j) \geqslant \tau_i$ is interpreted as a decision of classifying d_j under t_i, while $F_i(d_j) < \tau_i$ is interpreted as a decision of not classifying d_j under t_i. A classifier for a category is automatically extracted by observing the characteristics of a set of cases that have been classified under t_i, i.e. cases whose solution is t_i.

Although there are several types of classifiers potentially applicable to this problem, linear classifiers exhibit a number of interesting properties. Linear classifiers are a family of text classification learning algorithms which examine training instances a finite number of times in order to construct a prototype instance which is later compared against the instances to be classified. A prototype represents a topic or category and consists of a weighted vector $t_i = \langle (e_1, w_1), \ldots, (e_m, w_m) \rangle$ where w_j is the weight associated to the term e_j in the category t_i. The F_i function associated with each classifier is the cosine similarity measure shown in Eq. (2). Linear classifiers are both efficient, since classification is linear on the number of terms, cases and topics, and easy to interpret, since it is assumed that terms with higher

[1] In these experiments we used Syskill&Webert and BankSearch datasets.

weights are better descriptors for a topic than those with lower weights.

Naturally, the value of threshold τ plays an important role in the classification process. A value of τ higher than expected would prevent instances from being classified into the correct category, whereas a small value of τ would cause instances to be classified into wrong categories. In order to determine a suitable value for the classification threshold τ we evaluated the classification effectiveness using precision and recall measures using a subset of the text collections previously mentioned to build a hierarchy, and the remaining ones to measure precision and recall values. In these experiments we determined that the optimal value for τ threshold is 0.7.

At any time, a user interest hierarchy could consist of just a single topic representing the root category or several topics in one or more hierarchical levels. In both learning or prediction time, cases have to be assigned to topics in the current interest hierarchy. Having a hierarchy containing at least the root category, a new case is sorted down by recursively assigning the case to the best child topic at each level. The classification proceeds in a top-down manner. Initially, a case belonging to the root category (i.e. all cases) is categorized by the first level classifier. Then, classifiers at the second level take the case that has been already classified by the first one and classify it into the best topic at this level. This procedure continues until the case has reached some leaf node in the hierarchy or it cannot be further classified.

The process of hierarchy formation involves the gradual creation of topics summarizing cases within clusters and it is mainly driven by an evaluation function. Every time a new case is inserted into a hierarchy, this function determines whether a new topic can be defined to summarize cases in the cluster. Given the algorithm goal, which is partitioning the whole set of cases into a set of crescent specificity topics, a suitable evaluation measure should recognize the opportunity to create a new topic each time the existence of cases from more than one topic is detected inside a cluster. In other words, a loss of cohesiveness could give some indication of the existence of potentially new sub-clusters within a more general one. In such a case, it can be assumed that subtracting the set of features shared by most cases in the cluster to describe a general topic, a new partitioning of cases could be obtained and the hierarchy could gain an additional level of specificity.

The method applied to compute cluster cohesiveness consists of using the average pairwise similarity of cases that comprises the cluster as follows:

$$\frac{1}{n_r^2} \sum_{C_I, C_J \in s_r} \mathrm{sim}(C_I, C_J), \tag{4}$$

where n_r is the size of cluster s_r and C_I and C_J are a pair of cases in the cluster. If the cohesiveness value is higher than a given threshold φ, a new concept is created; otherwise no updating in the hierarchy takes place. We have empirically determined a value of $\varphi = 0.25$ by comparing hierarchies of topics generated with different values of φ in several collections against the target or expected hierarchies.

In case the creation of a new topic is established, a feature selection method is applied over cases in the cluster, taking as input a set of features and resulting in a subset of features that is relevant to the target topic. A weight is individually computed for each feature and features are ordered according to their assigned weights. Then, a feature selection threshold is defined such that the weight required for a feature to be selected needs to be higher than this threshold. There are several methods to compute weights for features appearing in a set of Web pages. A simple and yet effective one is the document frequency, denoted by $\mathrm{DF}(e_k)$, according to which the weight of a term e_k is the number of cases in which this term occurs. The document frequency is computed for every unique term appearing in the cluster and those terms whose frequency is less than the threshold are removed from the feature space. As it was stated by Koller and Sahami (1997), an accurate classification can be based on only a few features, those with better discriminant power to a given category. Each time a case is incorporated in the hierarchy, the nodes in which it was classified are updated to reflect the new information available about the topic, i.e. words and weights describing a topic can change over time.

After feature selection a supervised learning algorithm is applied to learn a classifier for the new category. In particular, we used in this work a instantiation of the Rocchio algorithm (Rocchio, 1971), the same used in Dumais, Platt, and Heckerman (1998), with parameters fixed to $\alpha = 1$ and $\beta = 0$, yielding

$$p_{t_i} = \frac{1}{n_{t_i}} \sum_{C_I \in t_i} C_I$$

as a prototype for each category $t_i \in T$. Hence, a classifier for a topic t_i is the plain average or centroid of all cases belonging to this topic. Once a classifier is built for a topic, the same process can be applied to identify sub-topics and create new classifiers starting from them. This leads to a hierarchy of classifiers of increasing specificity that constitutes a user profile.

Part of the user profiling process for two users, *User A* and *User B*, is illustrated in Fig. 5. In time $t = 1$, both users have the same interests, *sports* and *economy*. However their interests and, correspondingly, their profiles start to diverge in $t = 2$. While *User A* starts reading about *basketball* and a topic about this subject is added to his profile, the agent assisting *User B* observes readings about *tennis* and thus creates a new topic accordingly. *User A*'s profile becomes more specific regarding *basketball*, including two new sub-topics *NBA* and the *Argentinian league*. Likewise, the same can be expected from *User B*'s profile which will specialize in *tennis*. In account of the achieved profiles, agents will personalize their assistance for each of these users.

5. Experimental results

PersonalSearcher and *NewsAgent* have been implemented using the Java programming language.

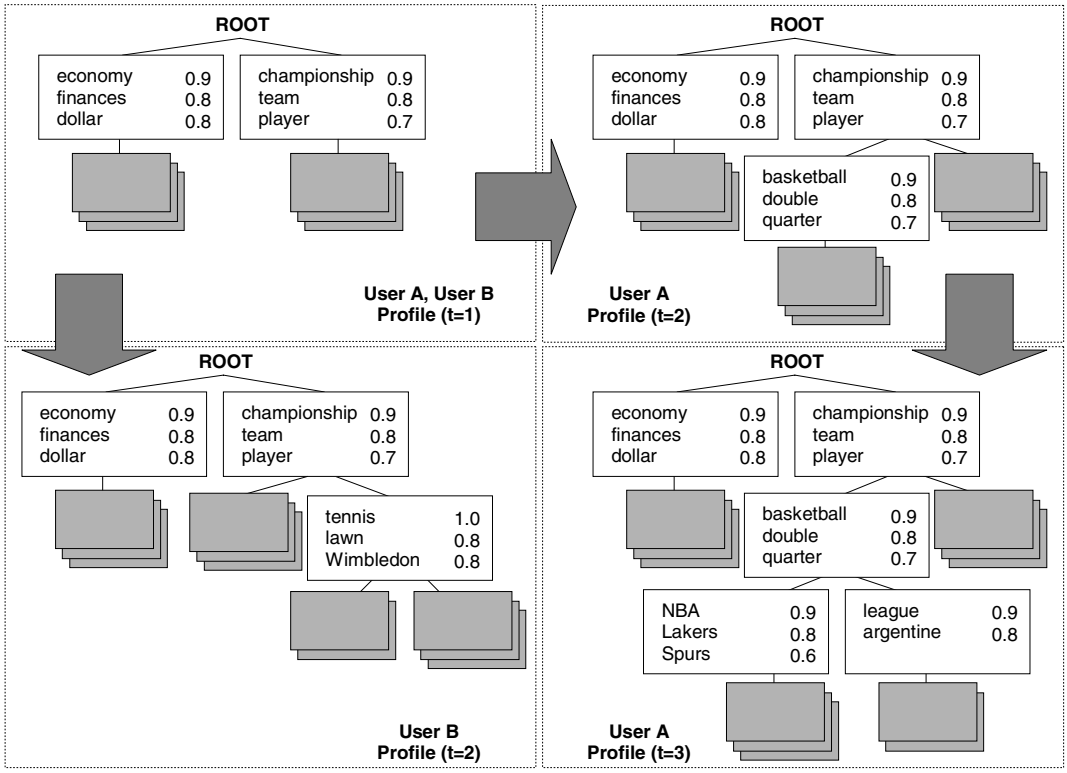

Fig. 5. Example of user profile evolution.

Our agents were evaluated by 35 users, which were Computer Science Engineering students. To test *NewsAgent*, 10 users navigated through three digital newspapers: *Clarin* [2], *La Nacion* [3] and *The New York Times*. [4] To test *PersonalSearcher*, the 35 users navigated freely through pages of their interest.

We have carried out three different kinds of experiments in order to evaluate the performance of our agents and the capability of our technique to determine users' interests. First, we studied the evolution of user profiles over time by analyzing user feedback. Second, we studied the performance of *PersonalSearcher* by analyzing its precision. Third, we analyzed the behavior of our agents when users' interests change over time. These experiments are explained in the following sections.

5.1. Analysis of agent performance through user feedback

To measure the effectiveness of our approach at inferring users' preferences, we evaluated agents' performance based on user feedback, as in Sheth (1994). In this way, we can view the evolution of user profiles over time. Since the goal of our agents is suggesting a user relevant Web documents or news, a measure of its effectiveness is the percentage of documents that were actually relevant to him among all the documents suggested by the agent.

Each user interacted with our agents to get relevant information, asking the agent for help from time to time. Each time the agent suggested the user some Web documents or news according to the user's inferred profile, we asked the user to provide feedback regarding the suggested documents. The graphic in Fig. 6(a) plots the performance of *PersonalSearcher* and *NewsAgent* over time. It shows the average proportion of documents that received positive feedback and the average proportion of documents that received negative feedback for each assistance session. These averages were calculated considering the feedback provided by all the users each time they were assisted by the agent. In order to make results comparable, each user asked an agent for assistance (i.e. Web page suggestions or a personalized newspaper) after reading 10 Web documents.

5.2. Analysis of agent performance through precision

Real user testing is an important component of the performance evaluation of agents. In order to carry out this experiment, a group of 25 users were given an initial explanation about how to use *PersonalSearcher* as well as access to the agent during a period of three weeks. Each user was asked to complete a form reporting a number of specific data about every browsing session as well as Web search performed in the course of this time.

In addition to providing feedback about the relevance of suggested Web pages, users were asked to report specific data about each Web search performed with *PersonalSearcher*. For a given search, users had to indicate the total number of search results, the number of pages suggested by the agent, the size of the user profile in terms of number of experiences at the moment the search took place and the number of interesting and uninteresting suggestions. In addition, they were also asked to go through the search results and find the number of relevant pages that the agent should have recommended but it did not.

Based on the reported data, effectiveness of filtering was measured using recall and precision. Recall is the ratio of the number of relevant documents returned to the user versus the total number of existing relevant documents, and precision is the percentage of the documents returned to the user that were actually relevant.

Each user reported the result of five searches at different stages of user profiling, with the amount of expected results set to 50 pages (the amount of pages to be retrieved is a configurable feature of *PersonalSearcher*) and the relevance threshold set to 0.5. In this experiment, precision fluctuates around an average of 0.82, while recall shows a

[2] http://www.clarin.com
[3] http://www.lanacion.com.ar
[4] http://www.nytimes.com

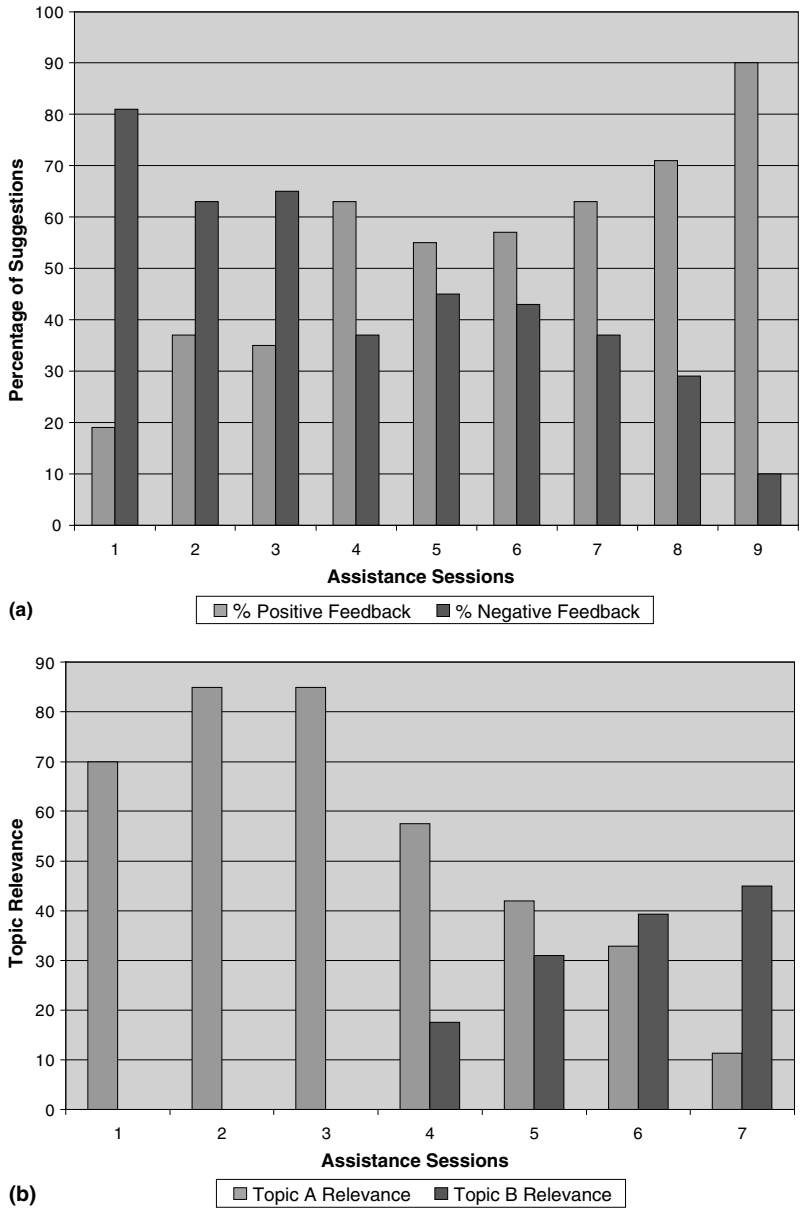

Fig. 6. Experimental results: (a) evolution of user profiles; (b) modeling dynamic interests.

clear tendency to increase, starting in zero when no pages can be recommended because the profile is empty. As a consequence, users have observed a definite pattern of consistent and improving performance as it is summarized by F-Measure (Larsen & Aone, 1999) in Fig. 7.

An empirical observation that can also be extracted from this experiment is the time required to attain good levels of assistance from *PersonalSearcher*. As it can be observed in the figure, around 25 experiences are enough to reach high values of F-Measure. On the other hand, agents

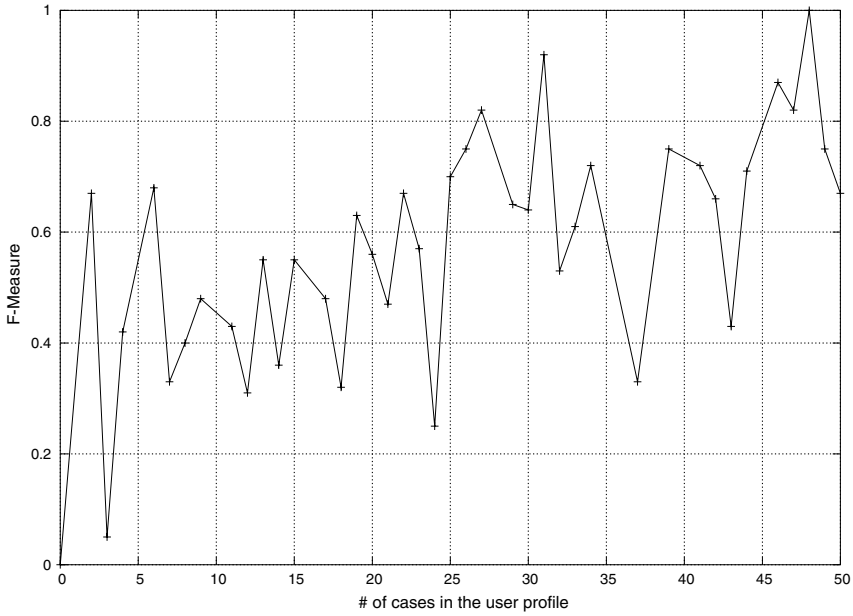

Fig. 7. F-Measure results from *PersonalSearcher*.

are highly interactive systems and their performance greatly depends on how the user uses them, which also explains some of the variations in the results.

5.3. Adapting to dynamic user interests

To perform this test we asked users to interact with our agents, acting according to a predefined behavior. First, they have to show interest in a given topic, reading documents about it and giving positive feedback to agents' suggestions related to this topic. Then, they have to abandon this topic and start reading documents about another topic. Besides, they have to provide negative feedback to suggestions related to the old interest and positive feedback to the ones related to the new interest. The graphic in Fig. 6(b) shows the variation of the relevance of the two topics involved in the experiments. The graph plots for each assistance session the average relevance value of the first topic against the average relevance value of the second topic.

We use the formula shown in Eq. (5) to compute the relevance value of a topic i. This formula is an adaptation of Rocchio's formula (Rocchio, 1971), which is commonly used in Information Retrieval to take relevance feedback into account

$$\mathrm{Rel}_i = \alpha * \mathrm{Rel}_i^{\mathrm{old}} + \beta \frac{\mathrm{PF}_i}{\mathrm{TotalF}} - \gamma \frac{\mathrm{NF}_i}{\mathrm{TotalF}}, \tag{5}$$

where $\alpha = 0.7$, $\beta = 0.15$ and $\gamma = 0.15$ are the weights of each term in the equation, $\mathrm{Rel}_i^{\mathrm{old}}$ is the previous relevance value (initially measured by considering the reading time, the amount of scrolling and bookmarks), PF_i is the positive feedback for topic i, NF_i is the negative feedback and TotalF is the total amount of provided feedback. The values of α, β and γ are chosen according to the importance each factor has for the agent designer. Since we consider that the relevance given by the reading time, scrolling and bookmarking is more informative regarding a user's interest than positive and negative feedback, we assigned α a greater value than β and γ.

Summarizing the results, the plots enable us to say that our agents' suggestions tend to users' interests, since the percentage of documents with positive feedback grows and the percentage of documents with negative feedback decreases. On

the other hand, the changes in the relevance of different topics reflect the changes in users' interests.

6. Related work

A number of personal assistants that help users in Web-based information tasks have been built in the last decade. Some of these developments include *Letizia* (Lieberman, 1995; Lieberman, Fry, & Weitzman, 2001), *Syskill&Webert* (Pazzani & Billsus, 1997), *WebMate* (Chen & Sycara, 1998), *Amalthaea* (Moukas & Maes, 1998), *Personal WebWatcher* (Mladenic, 1999) and *NewsDude* (Pazzani & Billsus, 1997).

Most of these assistants track user browsing behavior to acquire user profiles by employing text classification methods coming from either the machine learning or the information retrieval areas. *Letizia* is a browsing assistant that uses TF-IDF (term frequency/inverse document frequency) (Salton & McGill, 1983) vectors to model users' interests. In this technique the weight of each word is calculated by comparing the word frequency in a document against the word frequency in all the documents in a corpus. *WebMate* generates personal newspapers based on multiple TF-IDF vectors representing topics of interest. *Personal WebWatcher* and *Syskill&Webert* use naive Bayes classifiers for the same task. *Amalthaea* uses genetic algorithms to evolve a population of vectors representing the user interests and, finally, *NewsDude* obtains a short-term interest profile using the k-NN algorithm and a long-term interest profile using a naive Bayes classifier.

On the one hand, learning algorithms acquire user profiles by running over examples and producing results that are not explicitly represented but hidden in their own representational formalisms, such as decision trees or a naive Bayes classifier. Despite some of these approaches have proved their effectiveness for user profiling, they have still an important limitation. Resulting profiles are difficult to be either interpreted by users or other agents. On the other hand, information retrieval algorithms which have been adapted to the user profiling task, model user interests as either a single vector encompassing all user interests or, more frequently, multiple vectors representing individual user interests. However, profiles resulting from these approaches represent user interests at general levels of detail (e.g. a vector will group all readings about *basketball*, even when they are about *NBA* or *Lakers*).

The user profiling approach used to develop *PersonalSearcher* and *NewsAgent* aims at generating comprehensible user profiles that accurately capture user interests starting from observation of user behavior on the Web. This approach allows the hierarchical representation of very specific and changing topics of interest. Unlike most user profiling approaches, our technique offers comprehensible clustering solutions that can be easily interpreted and explored by either users or other agents for explanation/exploration as well as knowledge sharing.

7. Conclusions

In this paper we have described two experiences in the development of agents that assist users in Web-based information tasks. Both agents learn users' interests based on an approach for user profiling that models specific user interests in a topic hierarchy. The utilization of this kind of hierarchies as user profiles is a contribution both to user profiling and agent development areas.

Regarding other profiling algorithms, most of them produce results which are not explicitly represented but hidden in their own representational formalisms. Although these approaches are effective at user profiling, they have still two important limitations. First, resulting profiles are difficult to be either interpreted by users or other agents. Second, those information retrieval algorithms that have been adapted to the user profiling task represent user interests at general levels of detail.

We have evaluated our agents from different points of view. First, we studied the performance of the agents by analyzing user feedback. Second, we studied agents performance by studying precision and recall scores. Finally, we studied the ability of our agents to adapt to changing interests. Experiments made thus far with these agents have demonstrated the usefulness of our approach to detect

specific and dynamic topics of interest for different users. In addition, experiments carried out with *PersonalSearcher* proved that high levels of accuracy in suggesting Web pages can be reached.

However, we have discovered some limitations concerning the hierarchical organization of interests. We only classify documents under the best matching topic, but in some cases a document can belong to one or more topics. A solution could be placing this document under every matching topic in the hierarchy. As regards hierarchy construction, a hierarchy having multiple inheritance can be more appropriate when two or more topics share a given sub-topic. We are now working on these issues in order to improve the precision of user profiles and, consequently, the performance of our agents.

References

Chen, L., & Sycara, K. (1998). Webmate: A personal agent for browsing and searching. In K. P. Sycara & M. Wooldridge (Eds.), *Proceedings of the second international conference on autonomous agents* (pp. 132–139). St. Paul, MN, USA: ACM Press.

Claypool, M., Le, P., Wased, M., & Brown, D. (2001). Implicit interest indicators. *Intelligent User Interfaces*, 33–40.

Cordero, D., Roldán, P., Schiaffino, S., & Amandi, A. (1999). Intelligent agents generating personal newspapers. In *Proceedings of the 1st international conference on enterprise information systems (ICEIS'99)*, Setubal, Portugal (pp. 195–202).

Dumais, S. T., Platt, J., Heckerman, D., & Sahami M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM-98, 7th ACM international conference on information and knowledge management*, Bethesda, MD, USA (pp. 148–155).

Godoy, D., & Amandi, A. (2000). Personalsearcher: An intelligent agent for searching web pages. In M. C. Monard & J. S. Sichman (Eds.), *International joint conference IBERAMIA-SBIA'2000* (pp. 43–52). Atibaia, São Paulo, Brazil: Springer.

Kelly, D., & Belkin, N. J. (2001). Reading time, scrolling and interaction: Exploring implicit sources of user preferences for relevant feedback. *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 408–409). New Orleans, LA, USA: ACM Press.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In D. H. Fisher (Ed.), *Proceedings of the ICML-97, 14th international conference on machine learning* (pp. 170–178). Nashville, TN, USA: Morgan Kaufmann.

Kolodner, J. (1993). *Case-based reasoning*. San Mateo, CA, USA: Morgan Kaufmann.

Larsen, B., & Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 16–22).

Lenz, M. (1998). Textual cbr and information retrieval – a comparison. In L. Gierl & M. Lenz (Eds.), *Proceedings 6th German workshop on CBR*.

Lenz, M., Hübner, A., & Kunze, M. (1998). Question answering with textual cbr. In *Proceedings of the third international conference on flexible query answering systems* (pp. 236–247). Roskilde: Denmark.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. *Proceedings of the 14th international joint conference on artificial intelligence (IJCAI'95)* (pp. 924–929). Los Altos, CA, USA: Morgan Kaufmann.

Lieberman, H. L., Fry, C., & Weitzman, L. (2001). Exploring the web with reconnaissance agents. *Communications of the ACM, 44*(8), 69–75.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM, Special Issue on Intelligent Agents, 37*, 30–40.

Mladenic, D. (1999). Machine learning used by personal webwatcher. In *Proceedings of ACAI-99 workshop on machine learning and intelligent agents*, Chania, Crete.

Morita, M., & Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of SIGIR '94* (pp. 272–281). Dublin, Ireland: ACM Press.

Moukas, A., & Maes, P. (1998). Amalthaea: An evolving multi-agent information filtering and discovery system for the www. *Autonomous Agents and Multi-Agent Systems, 1*(1), 59–88.

Oard, D., & Kim, J. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, Madison, WI, USA (pp. 80–82).

Pazzani, M. J., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning, 27*(3), 313–331.

Porter, M. (1980). An algorithm for suffix stripping program. *Program, 14*(3), 130–137.

Rocchio, J. (1971). Relevance feedback in information retrieval. In *The SMART retrieval system* (pp. 313–323). Englewood Cliffs, NJ, USA: Prentice-Hall.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. New York: McGraw-Hill.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*.

Sheth, B. (1994). A learning approach to personalized information filtering. Master's Thesis, Massachusetts Institute of Technology.

Smyth, B., & Cotter, P. (1999). Surfing the digital wave: Generating personalised television guides using colaborative, case-based recommendation. In *Proceedings of the third international conference on case-based reasoning*, Munich, Germany (pp. 561–571).