User Profiling for Web Page Filtering

To help address pressing problems with information overload, researchers have developed personal agents to provide assistance to users in navigating the Web. To provide suggestions, such agents rely on user profiles representing interests and preferences, which makes acquiring and modeling interest categories a critical component in their design. Existing profiling approaches have only partially tackled the characteristics that distinguish user profiling from related tasks. The authors' technique generates readable user profiles that accurately capture interests, starting from observations of user behavior on the Web.

Daniela Godoy and Analía Amandi
ISISTAN Research Institute, UNICEN University

In the past decade, personal agents have emerged as an alternative for helping users cope with the increasing amount of information available on the Web. To provide personalized advice, these agents rely on knowledge about users contained in user profiles, such as models of user preferences and interests that agents employ to assist user activities. Profiling approaches have traditionally been concerned with obtaining predictive models that let agents decide the relevance of unseen pieces of information, but they've paid little attention to the assessment of explicit, readable descriptions of interests that users and other agents can interpret.

Researchers have frequently viewed acquiring and modeling user interests as a text classification problem.[1,2] In this context, users engage in a supervised learning process that generates user profiles that depend on the representational formalisms provided by learning algorithms (for example, a decision tree). Even though such predictive models directly support agent decisions and offer clear and unambiguous semantics of their output formats, user profiles act as *black boxes*. That is, the profile's internal structure is obscure to nonexpert users because it's too complex to be understood without explanation.

Recently, some have proposed ontology-based user profiling approaches that use ontologies to represent user interests.[3,4] Such approaches promise to close the semantic gap between those that have used low-level features extracted from documents (words) and the more abstract, conceptual view users might have of their interests. In spite of this advantage, ontological commitment for user profiling has several pitfalls. First, dealing with the high number of concepts most general-purpose ontologies

 Published by the IEEE Computer Society 1089-7801/05/$20.00 © 2005 IEEE

embrace could become rather expensive for modeling a single user profile. Second, although ontologies often mirror the shared knowledge of either a particular community or a mass of users, they fail to capture specific concepts that individual users might find interesting and thus don't capture the kind of documents users like to read.

To be truly useful, profiles should not only precisely describe users' specific interests but also provide a readable description of such interests so that users can explore their profiles and verify their correctness. Indeed, some have argued that the lack of transparency in recommender systems could result in a loss of trustworthiness.[5] Moreover, because user profiles are a starting point for creating user communities based on shared interests, meaningful profiles help users to more easily search the profiles of other users with common interests.

We propose a user profiling technique that's designed to support incremental learning and adaptation of user profiles in personal agents assisting users on the Web. The technique is built on the Web Document Conceptual Clustering algorithm that lets agents acquire profiles without an a priori knowledge of user interest categories,[6] so that the learning process is completely unsupervised. Furthermore, this algorithm belongs to the conceptual clustering paradigm, which includes clustering and characterization — that is, the formation of intentional concept descriptions for extensionally defined clusters. The use of conceptual clustering results in both fine-grained and readable descriptions of interests for user profiling.

Learning User Profiles

The first step in specifying a user-profiling technique for Web agents is to determine what knowledge profiles should model about users. Users could have diverse information interests related to, for example, their hobbies (sports), or their work (programming), which would be modeled into separate categories. In addition to modeling multiple interests in several domains, profiles must model the different abstraction levels of such interests. Thus, a natural way to organize interests is through a hierarchy, in which 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 not only enhances user profile semantics (as it is much closer to the human conception of a set of interests), but also lets agents have a temporal view of such interests. That is, even when some interests are expected to change over time, users frequently show a certain persistence of others. Therefore, interests at the top levels are seen as long-term interests, whereas those at the bottom levels are short-term.

We consider agents that determine user interests by identifying the different types of Web pages users generally visit. To build a hierarchy, the algorithm must gather, identify, and organize user experiences.

Representing User Interest Experiences

A *user experience* encapsulates both specific and contextual knowledge that describes a particular situation denoting a user's interest in a Web page.

A *user experience* encapsulates both specific and contextual knowledge denoting a user's interest in a Web page.

Experiences can be divided into three main parts: the description of the Web page content, the description of the situation during which it was captured, and the outcome of applying this experience to personalization.

To represent page content, the algorithm transforms pages into feature vectors in a space in which each dimension corresponds to a distinct term associated with a weight indicating its importance. The resulting representation is, therefore, equivalent to an n-dimensional vector

$$d_j = \langle (t_1, w_1), \ldots, (t_n, w_n,) \rangle,$$

in which w_i represents the weight of the term t_i in the document d_j. Before obtaining a document representation, the algorithm removes noninformative words using a standard stop-word list; the remaining terms are stemmed using the Porter stemming algorithm.[7] This is a process of linguistic normalization in which the variant forms of a word are reduced to one common form.

An experience also describes the contextual information of the situation in which it was cap-

tured, including the URL, date and time the experience was registered, and level of interest the user showed in the page, according to an agent criterion. The interest level is estimated using either explicit or implicit feedback. In the first case, the user gives an explicit evaluation of each visited Web page, whereas in the second case, the agent observes several implicit interest indicators (such as the time consumed in reading and the amount of scrolling) without disturbing users' normal behavior. This interest level serves as an initial degree of confidence in the experience that agents gather through feedback mechanisms. For example, if the user spent time reading an article about politics, it is considered a valuable experience or an example of the user's interests.

To determine whether to assign an experience to a given concept in the hierarchy, concept descriptions act as classifiers for categories.

Finally, an experience also registers user feedback about actions taken based on the knowledge it provides. Basically, experiences register the number of successes and failures an agent had when making decisions using them. This information lets agents increase or decrease their confidence in collected experiences to decide about future actions or, eventually, determine when an experience is no longer valid and can be forgotten.

Identifying User Interest Categories

Agents obtain experiences by extracting feature vectors from Web pages and incrementally presenting them to the clustering algorithm, which forms hierarchies of concepts or categories representing user interests. These hierarchies are classification trees, in which internal nodes represent concepts and leaf nodes represent experience clusters. A hierarchy root corresponds to the most general category within the user interests, which includes all experiences the algorithm has seen, whereas inner concepts become increasingly specific as they're placed lower in the hierarchy, covering only subsets of experiences by themselves. In turn, terminal concepts are those with no further child concepts.

As mentioned earlier, the clustering algorithm performs conceptual clustering; given a sequential presentation of experiences and their associated descriptions, the algorithm's task is to find clusters that group these experiences into concepts or categories, describe each concept, and organize them hierarchically.[8] A hierarchy of user interests will thus be constituted by an arbitrary number of concepts, denoted by $C = \{c_1, c_2, ..., c_m\}$, that the algorithm gradually discovers as new experiences become available. The algorithm automatically assigns a text-like description to concepts given by a set of terms, $c_i = \langle (t_1, w_1), ..., (t_p, w_p,) \rangle$, which are weighted according to their importance in the concept summarization. Thus, a category is any set of experiences, whereas a concept is a category's internal representation.

Leaves in the hierarchy correspond to clusters of experiences belonging to all ancestor concepts. Intuitively, clusters correspond to groups of experiences whose members are more similar to each other than to members of other clusters — that is, clusters group highly similar experiences seen by the algorithm. In general terms, a set of n_i experiences, belonging to a concept c_i and denoted by $D_i = \{d_1, d_2, ..., d_{ni}\}$, is organized into a collection of k clusters, $S_{ji} = \{s_{1i}, s_{2i}, ..., s_{ki}\}$, containing elements of D_i such that $s_{li} \cap s_{pi} = \theta$, $\forall l \neq p$.

User interest hierarchies must be built from scratch. As soon as new experiences appear, the algorithm clusters them by similarities in the profile to identify categories. Then, it automatically gleans a category description by observing the features that experiences in the category have in common, as well as the ones a novel Web page should have to belong to this category.

At any given moment, a hierarchy could consist of just a single concept representing the root category or several concepts in one or more hierarchical levels. Given a hierarchy containing at least the root category, the algorithm sorts new experiences by recursively assigning them to the best child category at each level. To determine whether to assign an experience to a given concept in the hierarchy, concept descriptions act as classifiers for categories.

A classifier is a function $F_i : d_j \rightarrow [0, 1]$ that, given an experience d_j, returns a number that represents the evidence for the fact that d_j should be classified under the category c_i. This function also has a classification threshold τ such that $F_i(d_j) \geq \tau$ is interpreted as a decision to classify d_j

under c_i, whereas $F_i(d_j) < \tau$ is interpreted as a decision not to classify d_j under c_i. Although several types of classifiers are potentially applicable to this problem, linear classifiers show many interesting properties. This family of text-learning algorithms examines training examples a finite number of times to construct a prototype example, which the algorithm later compares with the examples that will be classified in the category. These classifiers are efficient because classification is linear on the number of terms and categories, and easy to interpret because terms with higher weights are better descriptors than those with lower weights. The clustering algorithm obtains a Web page's weight for a given category by computing the page vector's closeness to the weighted vector constituting the classifier by using the cosine similarity measure:

$$sim(v_i, v_j) = \frac{v_i \bullet v_j}{\|v_i\|\|v_j\|} = \frac{\sum_{k=1}^{r} w_{ik} * w_{jk}}{\sqrt{\sum_{k=1}^{r} w_{ik}^2} * \sqrt{\sum_{k=1}^{r} w_{jk}^2}}, \quad (1)$$

where v_i and v_j are the respective vectors, w_{ik} and w_{jk} the weights of the word k in each vector, and r the number of different words.

Once an experience has reached a given concept in the hierarchy, either because it's a terminal concept or because it can't be distilled into a smaller category, the algorithm clusters it according to its similarity with past experiences belonging to the category. Thus, the experience is placed in the most similar cluster in the concept it belongs to. To predict in which cluster the experience will be placed, the algorithm determines the closest centroid, using Equation 1 to compare the new instance with all existing cluster centroids. If an experience isn't close enough to any cluster given a predefined similarity threshold, the algorithm creates a new singleton cluster to contain it.

Inferring a User Interest Hierarchy

The hierarchical concept formation process involves the gradual creation of concepts summarizing experiences within clusters and is driven by the notion of cohesiveness. A loss of cohesiveness indicates that potentially new subcategories exist within more general ones. The algorithm creates a new concept each time it presumes the existence of multiple categories inside a cluster. In such cases, we can assume that by subtracting the set of features that most experiences in the cluster share to describe a general category, we can obtain a new partitioning of experiences, and the hierarchy gains an additional level of specificity.

Every time a new experience is inserted, causing the cluster to update, the algorithm evaluates the experience's cohesiveness to determine whether a new concept can be defined in the hierarchy to summarize these experiences. A cluster's cohesiveness refers to how well individual instances match the prototypical description given by the centroid of the cluster to which the instances are assigned. The algorithm computes this using the average pairwise similarity of experiences in the cluster, as follows:

The algorithm creates a new concept each time it presumes the existence of multiple categories inside a cluster.

$$\frac{1}{|s_r|^2} \sum_{d_i, d_j \in s_r} sim(d_i, d_j) = \frac{1}{|s_r|} \sum_{d_i \in s_r} d_i \bullet \frac{1}{|s_r|} \sum_{d_j \in s_r} d_j = \|p_{s_r}\|^2, \quad (2)$$

where p_{sr} is the centroid of the cluster s_r. Mathematically, the average pairwise similarity between all instances in the cluster, including self-similarity, is equivalent to the length of the centroid vector. This equivalence lets the clustering algorithm evaluate cohesiveness without reprocessing all instances. If the cohesiveness value is higher than a given threshold, the algorithm creates a new concept; otherwise no updating in the hierarchy occurs.

When the creation of a new concept occurs, the algorithm applies a feature-selection method to experiences in the cluster to identify the features that best describe the experiences. The feature-selection method individually computes a weight for each feature, and then orders them according to their assigned weights. Next, this method uses a predefined feature-selection

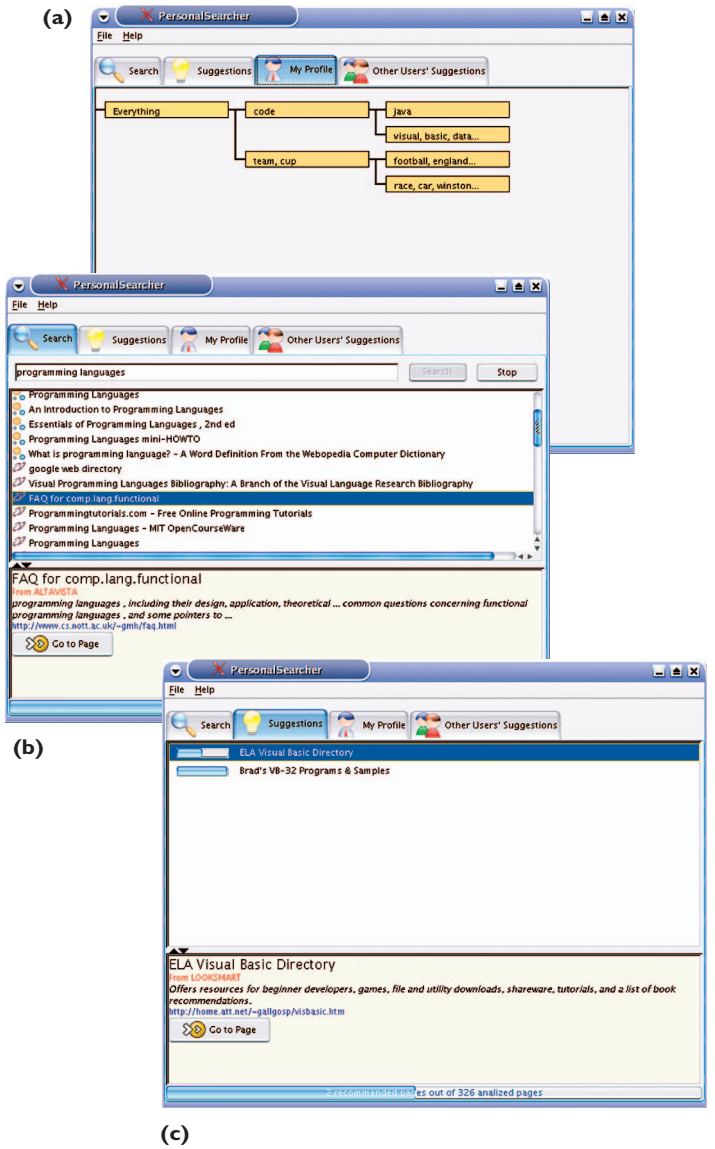


Figure 1. PersonalSearcher assistance. (a) The PersonalSearcher uses a user interest hierarchy to (b) filter search results for user queries and (c) suggest Web pages matching the user interests.

threshold that defines the weight required for a feature to be selected. A simple, effective way to weigh features is using the average frequency of occurrence that can be directly extracted from the centroid vectors. Because we can base an accurate classification on a few features,[9] such as those with better discriminant power in a given category, we perform an aggressive selection of terms to obtain those most representative for each category.

After feature selection, we use a supervised learning algorithm to learn a classifier or concept describing the new category. For this purpose, we applied an instantiation of the Rocchio algorithm, which is frequently used in information-retrieval approaches for user profiling (see the Related Work in Profiling Approaches sidebar on p. 62) with the parameters fixed to $\beta = 1$ and $\gamma = 0$, yielding

$$p_{c_i} = \frac{1}{|c_i|} \sum_{d \in c_i} d$$

as a prototype for each concept $c_i \in C$. Hence, a classifier for a new concept c_i is the average or centroid of all experiences belonging to this category as preceded by local feature selection. Finally, the algorithm added the new concept to the hierarchy and applies several operators to refine the hierarchical structure, including the merging, splitting, and promotion of concepts.

PersonalSearcher

We applied the user profiling technique we've been describing in developing the PersonalSearcher agent,[10] which assists users in finding interesting documents on the Web. This agent performs a parallel search in the most popular search engines and filters the resultant list of pages according to profiles it builds based on its observations of the users' Web browsing habits.

For each article read in a standard browser, PersonalSearcher observes a set of indicators to estimate how much interest a user has in a given Web page (time spent reading the page, how much they scroll, and whether they bookmarked the page). Using these indicators, PersonalSearcher agents obtain pages relevant to users without distracting them from regular activities. The clustering algorithm takes as input pages considered interesting to the user. The algorithm's output corresponds to a user interest hierarchy.

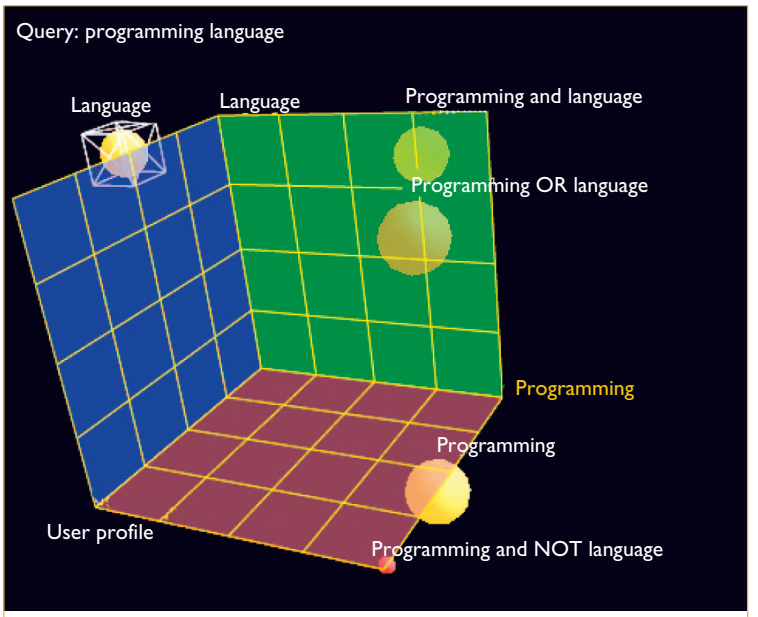
Users interact with their PersonalSearchers by expressing their information needs with keywords. The agent then posts these queries to the most popular search engines, receiving documents that cover a wide portion of the Web. PersonalSearcher determines the convenience of suggesting a Web page to the user by computing its relevance in relation to the user interest hierarchy. PersonalSearcher lets users customize how much assistance they want by adjusting the level of relevance of the expected suggestions from the graphical user interface. Once PersonalSearcher

has presented some suggestions, it again observes the user's behavior to adapt the profile according to approval to its suggestions.

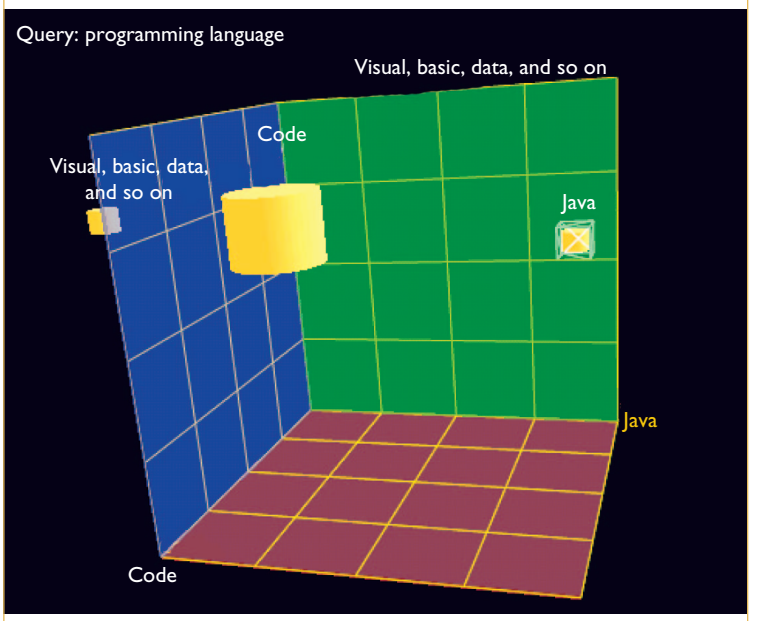
To illustrate PersonalSearcher behavior (Figure 1), let us suppose a user possesses several reading experiences in Visual Basic and Java languages, along with some about soccer and motor sports. Figure 1a shows the profile. The user profile reveals that PersonalSearcher can discover meaningful topics starting in pages, distinguishing the different subtopics of user interests and describing them accordingly.

If this user performs a Web search using the keywords "programming languages," the list of resulting pages could include pages about several aspects of programming, ranging from theoretical studies to diverse languages (Figure 1b). However, based on the knowledge of user interests, PersonalSearcher suggests pages mostly related to Visual Basic. Figure 1c shows some suggestions, along with an indication of their expected relevance. The agent discarded pages about other languages because, according to the user profile, it considered them uninteresting.

In addition to enhancing Web search by accounting for user interests, PersonalSearcher's 3D visualizations can help interactively filter the suggested pages, letting users find more relevant pages quicker than with traditional scrolled lists. In a 3D space, users can utilize different colors and shapes to support the visual cross-referencing of conceptual or topical information with other data or search properties (Figure 2). According to the 3D axis assignments and the composition of 3D objects, two kinds of visualizations are possible. The first, illustrated in Figure 2a, associates x and y axes with query keywords and the z axis with the whole profile. This view aims to help users understand how query keywords relate to their interests for further query refinements. The 3D objects group suggested pages containing both or one of the keywords in x and y axes; their volume is given by the amount of suggestions. The shift of objects over the z axis indicates its closeness to the user profile. The second visualization associates axes with categories in the profile and 3D objects with suggested pages in such categories (Figure 2b). The objects' positions in the space is based on their relationship with the categories in the axes. As a result, users can easily understand the semantic relationships of categories in the axes with categories represented by 3D objects.



(a)



(b)

Figure 2. PersonalSearcher results. (a) An example of visualizing suggestions by query keywords, in this case, programming language. (b) An example of visualizing suggestions by interest categories. Users can choose between several shapes (such as the circles and cylinders shown) and colors to quickly recognize objects and inspect their content (a transparent box encloses an object when it has been selected for examining its information).

Experimental results

To evaluate PersonalSearcher's performance and our technique's ability to determine user interests,

Related Work in Profiling Approaches

The user-profiling approaches on which personal information agents are based are typically drawn from the information-retrieval (IR) and machine learning (ML) fields. Both communities have explored the potential of established algorithms for user modeling purposes.[1,2] However, current user profiling approaches using these algorithms have several shortcomings.

Information Retrieval

In profiling approaches based on IR techniques, both documents and user interests are represented as vectors of weighted terms according to the vector-space model.[3] User interests are, therefore, represented by either a single vector embracing all interests or, more frequently, multiple vectors representing interests in several domains. The effectiveness of profiles in this approach depends on vectors' degree of generalization. We can achieve fine-grained profiles by increasing the number of vectors, whereas we can obtain coarse-grained profiles by representing interests with a small number of vectors. In the first case, profiles' effectiveness can be high, as can be their complexity, whereas in the second case, their effectiveness is limited because several interests coexist in the same vector. No attempt is made in these approaches to generalize the information available in a vector space.

Machine Learning

In profiling approaches based on ML techniques, learning algorithms acquire user profiles by running through examples and producing results that are not explicitly represented but rather hidden in their own formalisms. Several ML algorithms have been applied to user profiling in agents, such as Syskill&Webert.[4] Besides acting as black boxes for users, further weaknesses of these approaches include the computational complexity arising from the number of features and the problem of coping with dynamic interests. In most agents, adaptation has been almost exclusively limited to incorporating new information. Furthermore, most learning algorithms need several examples to build an accurate model, but generally profiles have to yield results from a few examples, and are sometimes based on positive evidence only.

Ontology-based Profiling

In ontology-based approaches, a profile is essentially the reference ontology whose concepts have weights that indicate the perceived user interest in each of them. Examples of this approach are *Ontology Based Informing Web Agent Navigation* (OBIWAN)[5] and *Quickstep*.[6] In heterogeneous domains such as the Web, the extent to which an ontology (such as *Yahoo!*) can represent a given user's interests is limited because an ontology can hardly embrace the specific interests of a large user population. Moreover, the number of concepts in the reference ontology an agent has to deal with increases the complexity of user profiling.

References

1. N. Belkin, J. Kay, and C. Tasso, special issue on user modeling and information filtering, *User Modeling and User Adapted Interaction*, vol. 7, no. 3, 1997.
2. G. Webb, M. Pazzani, and D. Billsus, "Machine Learning for User Modeling," *User Modeling and User-Adapted Interaction*, vol. 11, nos. 1–2, 2001, pp. 19–29.
3. G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," *Comm. ACM*, vol. 18, no. 11, 1975, pp. 613–620.
4. M. Pazzani and D. Billsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites," *Machine Learning*, vol. 27, no. 3, 1997, pp. 313–331.
5. J. Chaffee and S. Gauch, "Personal Ontologies for Web Navigation," *Proc. Int'l Conf. Information and Knowledge Management*, ACM Press, 2000, pp. 227–234.
6. S. Middleton, *Capturing Knowledge of User Preferences with Recommender Systems*, doctoral thesis, Univ. of Southampton, 2003.

we performed two different experiments. First, we analyzed how the profiles' content becomes refined over time, improving agent precision. Second, we analyzed the PersonalSearcher's behavior as user interests change over time.

Profile Evolution

We tested PersonalSearcher with 10 users and evaluated its performance based on relevance feedback. Because the agent's goal is to suggest relevant Web pages, a measure of its effectiveness is the proportion of pages relevant to a user out of all suggested ones. For this experiment, each user interacted with an instance of PersonalSearcher to get relevant information, asking the agent for advice at regular periods. Figure 3 shows the experiment's results. Each time the agent suggested Web pages according to the inferred profiles, users provided feedback regarding the suggested pages. Figure 3a plots PersonalSearcher's performance over time, showing the average proportion of Web pages that received positive and negative feedback for each assistance session. We calculated these averages with the user feedback provided each time agents assisted them. To make results comparable, each user asked an agent for assistance after reading 10 pages. The results of this experiment show that PersonalSearcher suggestions and user interests tend to be alike, because the percentage of pages with positive feedback grows, while the percentage of pages with negative feedback decreases.

Adaptation to Changing User Interests

To evaluate profile adaptation to changes in user interests over time, we asked the 10 users in the experiment to interact with their instances of PersonalSearcher according to a predefined behavior. First, they had to show interest in a given category, reading Web pages in this category and giving positive feedback to agent suggestions belonging to it. They then had to abandon this category and start reading about a second one. They also had to provide negative feedback to suggestions related to the first category and begin to give positive feedback to pages related to the new one.

We obtained the relevance of a category c_i in the profile as follows:

$$\mathrm{Re}\,l_{c_i} = \alpha\,\mathrm{Re}\,l_{c_i}^{old} + \beta\frac{PF_{c_i}}{TotalF} - \gamma\frac{NF_{c_i}}{TotalF},$$

where $\alpha = 0.7$, $\beta = 0.15$, and $\gamma = 0.15$, Rel_{ci}^{old} is the previous relevance value, PF_{ci} is the positive feedback for the category c_i, NF_{ci} is the negative feedback, and $TotalF$ is the total amount of received feedback. Figure 3b shows the variation in the relevance of the two categories, which we calculated based on the feedback received by their experiences. In these results, the changes in the relevance of different categories reflect the changes in user interests.

The experiment results and the example of using PersonalSearcher show the advantages of the profiling technique. First, as an incremental approach, it lets agents that interact with users acquire and maintain interest hierarchies as well as deal with unpredictable subject areas. Second, unlike most profiling approaches, our technique offers comprehensible clustering solutions that can be easily interpreted and explored by either users or agents. In preliminary experiments,[11] we've also achieved promising results from profile comparison.

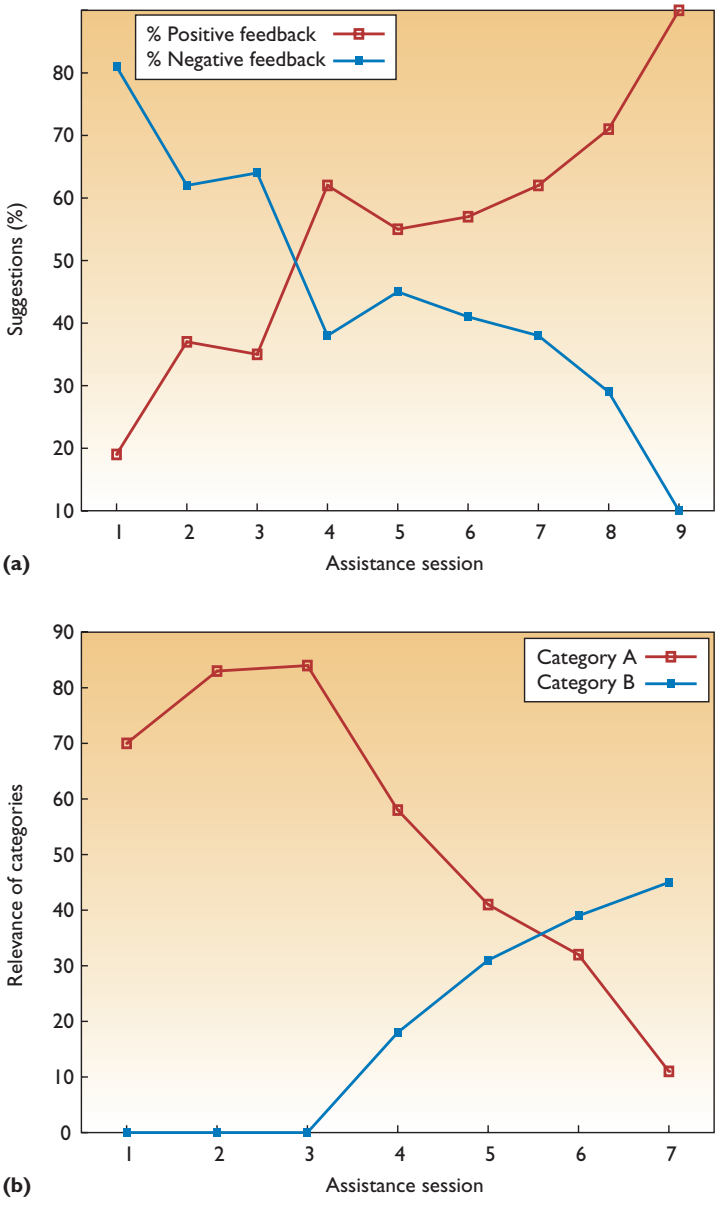


Figure 3. Evaluation of user profile effectiveness and adaptation. (a) We tested how PersonalSearcher's performance evolved over time, and (b) how it adapted profiles to users' changing interests.

The user-profiling technique we've presented provides a step toward the assessment of more comprehensible, semantically enhanced user profiles, the application of which can lead to more powerful personal agents, like PersonalSearcher, that can accurately identify user interests and adapt their behavior to interest changes. In addition, this technique opens new possibilities regarding users' interaction with their profiles as well as collaboration with other agents at a conceptual level.

Acknowledgments

Fundación Antorchas partially supported the research reported in this article.

References

1. G. Webb, M. Pazzani, and D. Billsus, "Machine Learning for User Modeling," *User Modeling and User-Adapted Interac-*

tion, vol. 11, nos. 1–2, 2001, pp. 19–29.

2. I. Zukerman and D. Albrecht, "Predictive Statistical Models for User Modeling," *User Modeling and User-Adapted Interaction*, vol. 11, nos. 1–2, 2001, pp. 5–18.
3. S. Gauch, J. Chaffee, and A. Pretschner, "Ontology-based Personalized Search and Browsing," *J. Web Intelligence and Agent Systems*, vol. 1, nos. 3–4, 2003, pp. 219–234.
4. S. Middleton, *Capturing Knowledge of User Preferences with Recommender Systems*, doctoral thesis, Univ. Southampton, 2003.
5. J. Herlocker, J. Konstan, and J. Riedl, "Explaining Collaborative Filtering Recommendations," *Proc. ACM Conf. Computer Supported Cooperative Work*, ACM Press, 2000, pp. 241–250.
6. D. Godoy and A. Amandi, "Modeling User Interests by Conceptual Clustering," to appear in *Information Systems*, Special Issue on Semantic Web and Web Services, 2005.
7. M. Porter, "An Algorithm for Suffix Stripping Program," *Program*, vol. 14, no. 3, 1980, pp. 130–137.
8. K. Thompson and P. Langley, "Concept Formation in Structured Domains," *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufmann, 1991.
9. D. Koller and M. Sahami, "Hierarchically Classifying Documents Using Very Few Words," *Proc. 14th Int'l Conf. Machine Learning*, Morgan Kaufmann, 1997, pp. 170–178.
10. D. Godoy and A. Amandi, "PersonalSearcher: An Intelligent Agent for Searching Web Pages," *Advances in Artificial Intelligence*, LNAI 1952, Springer-Verlag, 2000, pp. 43–52.
11. G. Giménez-Lugo et al., "Enriching Information Agents' Knowledge by Ontology Comparison: A Case Study," *Advances in Artificial Intelligence*, LNAI 2527, Springer-Verlag, 2002, pp. 546–555.

Daniela Godoy is a professor in the computer science department and the ISISTAN (Instituto de Sistemas Tandil) Research Institute of UNICEN (Universidad Nacional del Centro de la Provincia de Bs. As.) in Tandil, Argentina. Her research interests include intelligent agents, user profiling, and machine learning. She received her PhD in computer science from the UNICEN University, Tandil, Argentina. Contact her at dgodoy@exa.unicen.edu.ar.

Analía Amandi is a professor in the computer science department and head of the Intelligent Agents Group at ISISTAN Research Institute at UNICEN University in Tandil, Argentina. Her research interests include intelligent agents and knowledge management. She received a PhD in computer science in the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil. Contact her at amandi@exa.unicen.edu.ar.