The Knowledge Engineering Review, Vol. 00:0, 1–33. © 2005, Cambridge University Press
doi:10.1017/S0269888905000391 Printed in the United Kingdom

User profiling in personal information agents: a survey

DANIELA GODOY and ANALIA AMANDI

ISISTAN Research Institute, UNICEN University, Campus Universitario, Paraje Arroyo Seco, Tandil (7000), Bs. As., Argentina;
e-mail: {*dgodoy,amandi*} @ exa.unicen.edu.ar

Abstract

Personal information agents have emerged in the last decade to help users to cope with the increasing amount of information available on the Internet. These agents are intelligent assistants that perform several information-related tasks such as finding, filtering and monitoring relevant information on behalf of users or communities of users. In order to provide personalized assistance, personal agents rely on representations of user information interests and preferences contained in user profiles. In this paper, we present a summary of the state-of-the-art in user profiling in the context of intelligent information agents. Existing approaches and lines of research in the main dimensions of user profiling, such as acquisition, learning, adaptation and evaluation, are discussed.

1 Introduction

Information agents personalizing information-related tasks based on the content of documents and the knowledge about a user interests offer users an alternative to cope with the vast amount of information available in on-line sources such as the Web. For this reason, they have been largely applied in recent years to accomplish several tasks, such as searching, filtering and summarizing information on behalf of a user or a community of users.

From their beginnings, information agents evolved from simple agents responding to either scripts or to user-defined rules, to agents exhibiting a complex behavior such as learning user preferences and interests in order to autonomously perform tasks on behalf of users. In the last instance, agents rely on having some knowledge about users contained into user profiles, i.e. models of user interests, preferences and habits.

The effectiveness of agents depends mainly on profile completeness and accuracy and, consequently, user profiling has become a key area in the development of information agents. In spite of this fact, there is no real agreement on what knowledge profiles consist of, how they are acquired, represented and adapted over time and how they are used by agent applications. In existing information agents, user profiles range from a simply set of keywords weighted according to their importance in describing the user interests or the output format of a particular learning algorithm such as a decision tree or a probabilistic network, to more sophisticated models keeping track of long-term and short-term interests.

In this paper, we review user-profiling approaches that have been used in building personal information agents. Even though modeling preferences is frequently applied toward personalization of a variety of contents besides the content of documents, including multimedia (Angelides, 2003) and digital television (Ardissono & Maybury, 2004), this review is focused exclusively on profiles representing user information needs.

The remainder of the paper is organized as follows. Section 2 describes the user-profiling task and its main dimensions. Acquisition, learning and evaluation of user profiles are thoroughly analyzed in Sections 3, 4 and 5, respectively. Methods to evaluate the accuracy and effectiveness of user profiles are summarized in Section 6. Section 7 discusses the scope, benefits and limitations of current approaches for user profiling. Concluding remarks and future lines of research are stated in Section 8.

2 Intelligent agents and user profiling

Information agent technology emerged as a major part of intelligent software agent technology as a response to the challenges introduced by the vast amount of heterogeneous information sources available on the Internet. The main idea of this technology is the use of autonomous computational software entities, called intelligent information agents, to access multiple, heterogeneous and geographically distributed information sources on the Internet (or corporate Intranets) with the goal of acquiring, mediating and maintaining relevant information on behalf of users or other agents (Klusch, 2001).

In particular, information agents assisting users to cope with the increasing volume of information available in on-line sources are called personal information agents. These agents are computational assistants that perform different tasks such as finding, filtering and accessing large amounts of information on behalf of users, and present a reduced and potentially relevant part of this information to them. Personal information agents act like human assistants, collaborating with a user and becoming more efficient as they learn about their interests, habits and preferences.

The agents developed in this area have addressed many tasks, such as assisting users in searching and browsing the Web, organizing e-mail messages or filtering electronic news. To provide effective assistance with these tasks, personal agents act based on the knowledge about users contained in their user profiles, i.e. models of interests and preferences agents use to assist a user's activities. As agent effectiveness is highly dependent on profile precision, user profiling has become a key area in the development of information agents.

The user-profiling or user-modeling task involves inferring unobservable information about users from observable information about them, for example their actions or utterances (Zukerman & Albrecht, 2001). In order to perform this task, agents must deal with the uncertainty of making inferences about the user in the absence of complete information. Like other knowledge-based approaches, user modeling is concerned with two main issues: acquisition and representation.

The acquisition of user profiles is related to the mechanisms an agent has to formulate assumptions about users. In this regard, users provide valuable information about themselves during interaction with computers. Both user behavior and user judgments about agent actions are the most trustworthy sources of information for acquiring profiles and, from their successful interpretation, information agents will be able to tailor actions to user idiosyncrasies. In addition, user profiles can be refined, boot-strapped or even exclusively built based on a number of additional sources, such as the characteristics and shared beliefs of groups of users with similar tastes, domain knowledge or level of expertise. Methods for profile acquisition are discussed in Section 3.

The most usual approach to profile acquisition, however, is the application of learning mechanisms. Learning of user profiles based on the observation of user behavior leads to explicit representations of user interests that enable agents to make decisions about future actions. As profiles in this approach are usually expressed using the representational formalisms of specific learning algorithms (e.g. decision trees, rules, etc.), the learning process and the content of the resulting profiles are, consequently, interlinked issues.

Ultimately, a profile contains those features which characterize user information interests enabling agents to categorize documents based on the features they exhibit, even when it might not be possible for these agents to assess the full meaning of the documents. Although profiles normally include information about relevant topics to the user, good results have also been achieved by

including information about irrelevant topics too (Hoashi *et al.*, 2000; Widyantoro *et al.*, 2001). In such cases, agents use both kinds of resources to identify relevant pieces of information and discard non-relevant ones simultaneously.

A user profile allows agents to make decisions about actions to be carried out with individual, previously unseen pieces of information. If user profiles are acquired using a learning algorithm, decision making is directly supported by the learning method. In other cases, several methods can be used in order to compare user interests and information items. Besides profile-item matching, agents acting collaboratively with other agents toward a common goal also need to be endowed with a method to match user profiles in order to find users with similar interests. Issues related to learning and representation of user profiles are examined in Section 4.

The adaptation of user profiles is also an important factor in user profiling. Assuming that certain user interests persist for a long time (e.g. the interest in *painting* in an adult), an agent should become increasingly better at satisfying user needs, i.e. an agent should be able to gradually converge to the part of user needs that is predictable and consistent over time. However, since the interaction may extend over a long period of time, the user interests cannot be assumed to remain constant during such a time. A change in user interests can be anything from a slight shift in relative priorities to a complete loss of interest in some domains or an increase of interest in others. An agent must be able to detect such changes and respond to them by adapting user profiles in order to refine future agent behavior. In Section 5 current approaches for profile adaptation are discussed.

Finally, a relevant issue in user profiling is the assessment of user model performance and accuracy, which is treated in Section 6. Methods for user-profile evaluation in the context of personal information agents can be broadly classified into two categories: those which evaluate the profiles by themselves and those which evaluate the performance of agents using these profiles. In the first category, a user profile is evaluated as a learning algorithm from the machine-learning point of view, whereas in the second the user profile is embedded into an information agent and the efficiency of the agent to retrieve relevant information is assessed.

3 User profile acquisition

In order to provide personalized assistance, agents must have certain knowledge about the user and the application domain. Indeed, an agent's success depends mainly on the available information about users and its ability to represent user interests. Three main approaches have been developed to provide agents with this knowledge: the user-programming, the knowledge-engineering and the machine-learning approaches.

In the user-programming approach agents are explicitly programmed by users from scratch by means of scripts or rules for processing information related to a particular task. An example of this approach is *Information Lens* (Malone *et al.*, 1986), where a user can create an electronic mail sorting agent by defining a number of rules that process incoming mail messages and sort them into different folders. The main problem of this approach is that it requires too much insight, understanding and effort from the user to recognize the opportunity to employ an agent, take the initiative to create it, supply the agent with explicit knowledge and maintain the underlying rules or scripts over time.

In contrast, the knowledge-engineering approach consists of constructing an agent with domain-specific knowledge of both the application and the user. In this case, the task of programming the agent lays on the knowledge engineer instead of the end user. However, it requires substantial efforts in the eliciting task and the result is highly domain-specific agents. Moreover, agent knowledge is relatively fixed and, consequently, hardly adaptable to different application domains or to perform personalized tasks. *Ugo* (Chin, 1991), an agent designed to help users to solve problems in the UNIX operating system by means of a large knowledge database, exemplifies this approach.

The third and prevalent alternative is to endow agents with machine-learning mechanisms in order to allow them to acquire the knowledge they need to assist users. In such cases, the agent is given limited background knowledge, and it learns appropriate behavior from the user and from other agents. The learning approach has two main advantages over the previous ones. First, less effort from both users and knowledge engineers is required in agent construction as profiles are automatically acquired through learning. Second, the agent can easily adapt profiles to changes in user interests over time and becomes customized to individual preferences and habits. Finally, the knowledge gained can be also transferred to other agents in a community.

Before applying machine-learning techniques to personal agents the following conditions have to be fulfilled (Maes, 1994): (1) the use of the application has to involve a substantial amount of repetitive behavior considering the actions of one user or among users; and (2) this repetitive behavior has to be potentially different for each user. If these conditions are satisfied, agents acquire their competence starting from four different sources.

- Observing and imitating the user: agents monitor user activities and record their actions over long periods in order to find regularities and recurrent patterns of behavior feasible of being automated, as explained in Section 3.1.
- Receiving positive and negative feedback from the user: either explicit or implicit feedback is given registering the course of actions taken by the user according to the agent suggestions (Rocchio, 1971), as explained in Section 3.2.
- Receiving explicit instructions from the user: users can train agents by giving them hypothetical examples of events and situations and telling them what to do in those cases; an approximation to this source of knowledge is given by the programming by example (PBE) paradigm described in Section 3.3.
- Sharing knowledge with other agents: agents can ask agents assisting other users with the same task (who may have built up more experience) for advice, or take advantage of the assessed generalizations about a user community. In this direction, the stereotype and collaborative approaches for agent construction are described in Section 3.4.

3.1 Observation of user behavior

The knowledge about users that has to be modeled into their profiles can be either implicitly or explicitly acquired from users, leading to a division between explicit and implicit user profiles (Rich, 1983). An explicit user profile is elicited from a series of questions designed to acquire user interests and preferences precisely. The main advantage of this method is the transparency of agent behavior as decisions can be easily deduced from the data provided. However, it requires a great deal of effort from users and, additionally, users are not always able to express their interests because they are sometimes still unknown. Instead, a shallow model of user preferences and interaction patterns can be obtained based upon a relatively short-term interaction with an agent monitoring user behavior by continuously 'looking over the user shoulder' as the user is performing actions. A more accurate model capturing user interests and goals can be in turn attained after a long-term interaction.

Implicit knowledge acquisition is often the preferred mechanism since it has little or no impact on the user regular activities. Unobtrusive monitoring of users allows agents to discover behavioral patterns that can be used to infer user interests, preferences and habits. In order to achieve this goal, a number of heuristics are commonly employed to infer facts from existing data. In the course of their activities, users leave behind a trail of information that can be used to model their interests. Some sources of information left by a user after browsing include: (1) the history of the user requests for current and past browsing sessions that is maintained by most browsers; (2) bookmarks giving a quick means for accessing a set of documents exemplifying user interests; (3) access logs where entries correspond to HTTP requests typically containing the client IP address, time-stamp, access method, URL, protocol, status and file size; (4) personal homepages and material as well as their outgoing links.

Implicitly acquired knowledge requires some degree of interpretation to understand the user's real goals (e.g. if the user does not read a Web page suggested by an agent it can be considered uninteresting). Hence, it is an unreliable, inherently error prone process, which reduces the overall confidence in the resulting profiles. In contrast, explicit knowledge is generally information that provides high confidence, since it is provided by the users themselves and not acquired from indirect sources. Explicit acquisition requires the agent to interrupt the user to either provide feedback or instruct the agent in some way. This is intrusive and provides no guarantee that the questions asked are answered truthfully, or even that the questions asked are the right ones to obtain the desired information.

3.2 Relevance feedback

A fundamental source of information about users is the relevance feedback they explicitly or implicitly provide about examined documents or agent actions. A user gives explicit feedback by using one or more ordinal or qualitative scales, whereas implicit feedback is estimated by agents according to observation of a group of interest indicators.

A central issue of explicit feedback is that the user has to examine items to assign them a value on a rating scale or write comments about items. Rating scales are typically numeric or symbolic with mapping to a numeric scale. In *Syskill & Webert* (Pazzani & Billsus, 1997) users have to rate Web pages as *hot*, *lukewarm* or *cold*; in *Amalthaea* (Moukas & Maes, 1998) and *NewsDude* (Billsus & Pazzani, 1999) users rate items on a five-point scale. In systems involving communities of users, a user is encouraged to provide comments about items in order to facilitate other users' decision-making. Although textual comments can be helpful, they require a careful analysis of a user to understand their positive or negative connotations.

Even though more reliable and easy to implement, explicit feedback burdens the user with an additional cognitive load caused by the necessity of evaluating each information item. In addition, this method has several shortcomings (Claypool *et al.*, 2001): (1) having to stop to enter explicit ratings can alter normal patterns of browsing and reading; (2) unless users perceive an immediate benefit from providing ratings, they may stop providing them; hence, users continue using the system to read but not to rate documents resulting in an unappropriated use of the system; (3) users can provide inconsistent ratings over time.

Implicit feedback, on the other hand, is calculated on the base of one or more implicit interest indicators, which act as surrogate measures for predicting user interest on a given information item. Each of these interest or approval indicators concentrates on a single behavioral sign or a pattern of behavior and they can be classified as belonging to three broad categories: examination, retention and reference (Oard & Kim, 2001). The first group is composed of those indicators related to the selection of individual items for further examination (e.g. time spent reading a Web page or amount of scrolling). In this group, a number of studies has been carried out to establish indicator accuracy to predict user interest level in information items. For example, it has been found that the time spent on a page, the amount of scrolling and the combination of time and scrolling have a strong correlation with explicit interest feedback, as opposed to the number of mouse clicks, which have been demonstrated not to be a good indicator (Claypool *et al.*, 2001; Kelly & Belkin, 2004). Likewise, some studies have revealed a positive correlation between reading time and explicit ratings in USENET news applications (Morita & Shinoda, 1994; Konstan *et al.*, 1997). The retention category is intended to group those behaviors that suggest some degree of intention from the user to make future use of an item (e.g. bookmarking a Web page is a simple example of such behavior). Finally, the reference group contains activities that have the effect of establishing some form of link between two items (e.g. forwarding a message establishes a link between the new message and the original one).

Implicit feedback can be combined with explicit feedback in order to decrease the user effort, while maintaining a certain accuracy in inferring user preferences. In *Anatagonomy* (Kamba *et al.*,

1997), an agent generating personal newspapers for users, explicit feedback is optional and is only given by users to show explicit interest in an article. Also feedback can be shared among agents assisting similar users for the purpose of exchanging recommendations. In this context, the feedback of another user on unseen items can be used as a basis to generate recommendations for a particular user. However, it has been established that users read more documents than the one they actually rate, while to establish a collaborative behavior, agents require many ratings for every item in order to produce accurate predictions and avoid the sparsity problem (Sarwar *et al.*, 2000). Research on the creation of interfaces for eliciting user preferences in this kind of system has demonstrated that users preferred to be asked which items they have to rate instead of choosing the items by themselves, which also increases their loyalty to the system (McNee *et al.*, 2003).

Further than explicit or implicit, feedback can be positive and/or negative and with different levels of intensity. On the one hand, strong positive or negative feedback should result in a significant change in user profiles. If an information item was completely uninteresting to the user, similar items should not be recommended by the agent in the future. Instead, potentially recommendable documents should be those that are similar to documents that were interesting to the user in the past and that, consequently, received strong positive feedback. Although it is not the only element to judge the relevance of documents, similarity provides a strong indication. In an examination of the similarity assumption it was found that similarity is only a fair predictor of utility, which can be improved with task-specific knowledge (Budzik *et al.*, 2000). On the other hand, moderate changes in users interests can be detected through less intensive feedback, allowing gradual adaptation of profiles. Then, a weak negative feedback effect is to decrease confidence in the recommendation of similar items. Inversely, the positive effect of a moderate positive feedback should be to increase the confidence in recommending documents that are similar to previously recommended documents.

3.3 Programming agents by example

Agents can also receive instruction from their users on how to behave in certain events or situations. PBE or programming by demonstration (PBD) (Lieberman, 2001), a powerful end-user-programming paradigm that enables users without formal training in programming to create sophisticated programs, is potentially applicable to accomplish this goal. In a PBD approach the interactions between the user and a direct manipulation interface are recorded in order to write a program that corresponds to the user actions and system behavior. Thus, it is possible to generalize a program to work in similar situations, not necessarily identical, to the received examples.

Even when there are no examples of PBD applications specific for user profiling, this paradigm has been used to solve common problems that personal information agents have to face, such as text recognition and script programming to extract information from the Web. A PBD application closely related to the task performed by personal information agents is *Grammex* (Grammars by Example) (Lieberman *et al.*, 1999). This is a direct manipulation interface designed to allow non-expert users to define grammars interactively in order to recognize texts. The user presents concrete examples of the kind of texts they want the agent to recognize, *Grammex* heuristically parses the examples and displays a set of hypotheses expecting the user to criticize them and, in this way, a number of rules are constructed in an iterative process. Likewise, the actions an agent has to take upon text recognition are also demonstrated by example.

Another example is a PBD system that processes Web sites in order to allow agents to autonomously gather information from them, called *Scriptor* (Nuri, 2000). The extraction of information from Web sites is generally based on scripts that are binded to the underlying HTML tag structure of the site. Thus, changes in their look-and-feel and structure need to be followed by the corresponding script updating. To address this problem *Scriptor* uses the site new pages as examples to create and update scripts that identify relevant information within the site. A similar approach is taken by the *trainable information assistants* (TrIAs) (Bauer *et al.*, 2000) to help in the

maintenance of database wrappers or patterns used to specify the format of Web pages. In order to guide the training of these agents, documents are opened in the Web browser and the user has to mark the position of the text or pictures to be extracted and, thus, give assistants hints to identify a particular piece of information within them. As a result, a new information extraction procedure is synthesized and inserted into the database for future use.

3.4 Knowledge sharing among agents

A personal information agent typically learns about individual users by observing their behavior over time. However, it may take a significant amount of time and observations to construct a reliable model of user interests, preferences and other characteristics. To reduce this time, agents can take advantage of the behavior of similar users accessible through the knowledge of other agents. A number of methods have been developed for *collaborative* or *social filtering* of information (Resnick *et al*., 1994). However, in an earlier stage to collaborative approaches, the user modeling community provided a different answer, namely the *stereotype approach*. Both alternatives are described in the following subsections.

3.4.1 Stereotypes

The acquisition of user profiles in a stereotype approach is assisted by the assessing of generalizations about communities of users (Rich, 1979). First, user subgroups are identified in order to determine the typical characteristics of their members. Afterward, users are assigned to one or more of these predefined user groups and the group characteristics are attributed to them. The application of stereotypes for user-profile acquisition has been shown to be useful in areas where a fast, but not necessarily precise, assessment of user interests is required. In such situations, stereotypes are a basic information source that is used for initial default information about the user when nothing else is available.

A stereotype contains the typical characteristics of a group of users in a particular application domain along with a set of activation conditions, which make it feasible to identify users belonging to this group. In this way, a stereotype is applicable to a given user if either it is manually assigned to it or the stereotype activation conditions match the available information about the user in an automatic classification process. As a consequence of assigning a user to a stereotype, all the stereotype characteristics are attributed to the user.

Stereotypes serve as substitutes of user profiles or as a starting point for their acquisition, in which case they are used to provide an initial definition in order to reduce an agent learning curve. As a user may be close to one or more stereotypes, an initial profile can be formed as a combination of several stereotypes. In any case, the profile that is first inherited from the stereotypes is later revised according to the user actions and responses to the recommended items. A comparative study of personal and stereotypical information filtering rules revealed that stereotypical rules are as effective as personal rules and, consequently, they can be safely used to provide a good initial user profile (Kuflik *et al*., 2003).

An obvious disadvantage of this approach is the necessity for a pre-definition of stereotypes, whose construction is almost exclusively manual as this is a process that involves the classification of users by an expert and the analysis of individual interests of users. Recently, the automatic construction of stereotypes by user clustering has also been proposed, i.e. groups of users are obtained based on a set of individual user profiles and a stereotype is constructed based on the common characteristics of each cluster (Orwant, 1995; Paliouras *et al*., 1999). An example of automatic construction of stereotypes by clustering is presented by Paliouras *et al*. (1999). In this work, an unsupervised learning algorithm (*COBWEB*) is used to create user communities in the context of a news filtering system based on the user interests in news categories, extracted from a set of questionnaires. Then, a supervised learning algorithm (*C4.5*) is applied over both personal data and interests on specific news categories in order to build stereotypes. The news categories that

are most representative of each community are identified by the system in order to improve its services. A further application of stereotyping is *Lifestyle Finder* (Krulwich, 1997), which creates user profiles according to demographic data, dividing a population into clusters according to their purchasing history, lifestyle characteristics, etc. Users are then classified in terms of their demographic data, and the classifications are used as general characterizations of the users and their interests.

3.4.2 Collaborative filtering (CF)

In the development of information agents two broad approaches can be identified: *content-based* and collaborative filtering (CF). Initially, systems developed under the second approach were also known as *recommender systems*, but the scope of this term has been extended over the past few years to embrace both approaches. The former is based on the intuition that each user exhibits a particular behavior under a given set of circumstances, and that this behavior is repeated under similar circumstances. The latter is based on the intuition that people within a particular group tend to behave alike under similar circumstances. Thus, in the content-based approach the behavior of a user is predicted from his past behavior, while in the collaborative approach, the behavior of a user is predicted from the behavior of other like-minded people (Zukerman & Albrecht, 2001). According to the previous definitions, it can be observed that the content-based approach is based on the comparison of documents to user profiles, while the collaborative approach is based on the comparison of user profiles.

CF is the most successful recommendation technology developed to date, and is used in many recommender systems on the Web, including those at Amazon.com and CDnow.com. It has been used mainly in applications not involving the analysis of textual information, for example in the recommendation of music, movies, etc. An example is *Ringo* (Shardanand & Maes, 1995), a music-recommendation system that recommends albums and artists that were highly scored by users with similar musical tastes. In this system, users have to describe their interests by rating some music; these ratings constitute their initial user profiles, which change over time as the user rates more artists. *Ringo* uses profiles to generate advice to individual users as well as to compare with other profiles and ascertain users with similar tastes (e.g. they like the same albums and dislike the same albums). *Ringo* applies methods such as the mean squared difference (MSD) and the Pearson correlation coefficient to compare users based on their previous actions.

A few collaborative systems used this approach for document recommendation. *GroupLens* (Konstan *et al*., 1997) and *PHOAKS* (Terveen *et al*., 1997) are two examples of collaborative recommendations of messages from USENET news. *GroupLens* computes correlation between readers of USENET newsgroups by comparing their ratings of news articles. The ratings given by an individual user are employed to find related users with similar ratings and to predict the user interest in news articles. *PHOAKS* searches messages from USENET news for Web links and counts each occurrence of a link in a valid context as a recommendation. Thus, when a user queries the system for Web addresses in certain topics, it searches news articles in that topic, retrieves the implicitly recommended Web addresses and presents them as a result for the user query.

Both collaborative and individual recommendation approaches have some limitations. For the content-based approach, the main problem is the shallow analysis of document contents that can be performed. A user can always be a better judge of the document contents than agents and, at the same time, other contents except for text, such as video or sound, are difficult to capture. Also content-based agents tend to over-specialize so that the user is limited to receiving recommendations of items similar to those already seen. CF approaches offer the possibility of solving this problem by providing a diversification of recommendations. However, a purely collaborative approach ignores the content of items and bases recommendations exclusively on the judgments or ratings given by users, frequently explicitly, on each item. If the number of users is reduced in relation to the information space to be explored, the distribution of items that have already received

some evaluation is sparse, which is known as the sparsity problem (Sarwar *et al.*, 2001). A further weakness of collaborative approaches is the cold-start problem, which is the lack of initial evaluations for items and the problem related to users with unusual preferences, which receive poor recommendation from their agents. An additional problem of collaborative filtering is the lack of transparency since agents generally act as black boxes—a user is given no indication whether to trust agent recommendations (Herlocker *et al.*, 2000).

Hybrid recommendation approaches combine the advantages of the mentioned approaches, helping to overcome their limitations. An example of a hybrid approach for news recommendation is the one taken in *FAB* (Balabanovic & Shoham, 1997). *FAB* features two types of agents: collection agents, which retrieve news articles and send them to a central router for distribution; and selection agents, which select documents for a particular user based on the user preferences and the preferences of other users with similar profiles. In order to give a recommendation, selection agents calculate the similarity between documents and the user profile and the best evaluated documents are sent to similar users. *PTV* (Cotter & Smyth, 2001) also uses a hybrid approach to recommend TV programs based on user profiles composed of TV channels, keywords, programs, etc. that are updated through relevance feedback. On the one hand, programs are compared to profiles in order to be recommended. On the other hand, the similarity among profiles is assessed to make recommendations for a target user based on the viewing preferences of a group of similar users using the system.

The aforementioned problems of collaborative approaches have been addressed from several perspectives. For example, two machine-learning algorithms combining collaborative and content information were evaluated based on a novel performance metric, called the CROC curve, on cold-start prediction (Schein *et al.*, 2002). The sparsity problem in recommender systems has been tackled using dimensionality reduction techniques, associative retrieval in the graph of items and users, item-based similarity instead of user-based similarity and content-boosted collaborative filtering. Dimensionality reduction is used to condense the user-item matrix applied in recommendation by removing unrepresentative or insignificant users or items (Sarwar *et al.*, 2000). The associative retrieval framework and related spreading activation algorithms were also used to explore transitive associations among consumers through their past transactions and feedback (Huang *et al.*, 2004). Item-based techniques first analyze the user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users (Sarwar *et al.*, 2001). Both the item-based and the content-boosted approaches require additional information regarding items as well as metrics to compute meaningful similarities among them.

Issues related to trust, reputation and reliability are becoming increasingly important in recommender systems. In order to collaborate, recommender agents exchange information by means of CF methods, which can be augmented with trust models. For example, the epinions.com service builds a trust model directly from trust data provided by users as part of the system Massa & Bhattacharjee (2004). Users can assign a trust rating to reviewers based on the degree of helpfulness and reliability. In Montaner *et al.* (2002) agents are considered as personal entities that are more or less reliable or trustworthy. Trust values are computed by pairs of agents on the basis of a conversational exchange in which one agent asks the opinions of another agent with respect to a set of items. Each agent then infers a trust value based on the similarity between its own opinions and the opinions of the other agent. Two different trust models, one that operates at the level of profiles and one at the level of items within a profile, had a positive impact on overall prediction error rates as reported by O'Donovan & Smyth (2005).

4 User-profile learning and representation

In order to adapt their assistance to individual users, agents have to learn about user preferences and attitudes and model them into user profiles. The main distinction that can be made in the

learning field is between *supervised* and *unsupervised* learning methods. The former requires a pre-classified set of training examples, in which each training example or instance is assigned a unique label indicating the class it belongs to among a finite set of possible classes. Therefore, the goal of a supervised algorithm is to induce a classifier, model or hypothesis to correctly classify novel unlabeled examples. In contrast, unsupervised learning methods do not require pre-classification of the training examples; these algorithms form clusters of examples which share common characteristics.

A supervised learning method can be directly implemented to user profiling since observations of user behavior can provide training examples an algorithm can use to form a model or user profile designed to predict actions. In casting a user-profiling problem to a supervised learning task, at least two classes can be anticipated interesting and uninteresting pieces of information (Web pages, news, e-mail messages, etc.). Indeed, most tasks personal agents have to face respond to a binary classification problem, for example, a Web page can be classified as interesting or uninteresting, an e-mail message can be classified as spam or not spam, and so forth. In order to build a user profile modeling these two classes, a set of positive and negative examples representing user interests needs to be collected by the agent through observation. Hence, the user provides feedback about the accuracy of the derived profile, which is used to guide further learning. User profiling can also be mapped to classification problems in which more than two classes are available, such as classifying e-mail messages into personal folders (e.g. *work*, *finances*, etc.) or news into pre-defined newsgroups (e.g. *rec.autos*, *rec.sport.baseball*, *talk.politics.mideast*, etc.).

In spite of the fact that user profiling seems to be a suitable task to straightforward applications of supervised learning algorithms, in most user-profiling problems user interests can not be defined beforehand as agents have to model user interests in unpredictable subject areas, which are also potentially different for every user. Unsupervised methods allow agents to discover user interest categories in several domains based on clustering of experiences. User interest categories are expected to group similar examples corresponding to the same user interests (e.g. *basketball* or *soccer*). A novel example is considered interesting to the user if it belongs to at least one of the already identified categories.

Recently, ontology-based user-profiling approaches have been proposed to take advantage of the knowledge contained in ontologies instead of attempting user-profile acquisition (Gauch *et al.*, 2003; Middleton *et al.*, 2004). An ontology is a conceptualization of a domain consisting of entities, attributes, relationships and axioms that can exist for an agent or a community of agents, which is established for the purpose of enabling knowledge sharing and reuse. In this context, an ontology is a specification used for making ontological commitments, i.e. an agreement to use a determined vocabulary in a way that is consistent with the theory specified by the ontology.

In this approach, user interests are mapped to domain concepts instead of generating a model by analyzing examples. A user profile is then represented in terms of which concepts from an ontology a user is interested in, irrespective of the specific examples of such interests. Examples of user interests are then classified into the concepts from the reference ontology and the user interest in such concepts is registered.

Ontology Based Informing Web Agent Navigation (*OBIWAN*) (Gauch *et al.*, 2003) is an example of an ontology-based user-profiling approach in which a user profile is created automatically and implicitly by analyzing the user browsing behavior. A profile in this system is essentially the reference ontology whose concepts have weights indicating the perceived user interest in each of them. The Web pages the user visits are automatically classified into the concepts contained in the reference ontology (reference ontologies used in this work are based on subject hierarchies and associated Web pages from *Yahoo*!, *Magellan*, *Lycos* and the *Open Directory Project*) and the results of the classification are accumulated. As a result, the concepts in the reference ontology receive weights based on the amount of related information the user has browsed. In order to classify the user-visited Web pages into concepts in the reference ontology, sample documents are gathered for each concept in the hierarchy so that they can be used as training data to build

classifiers. On the basis of these profiles the system presents to the user Web sites organized according to the user ontology.

A similar approach is taken by *Quickstep* (Middleton, 2003), a hybrid recommender system addressing the problem of recommending on-line research papers to researchers, which bases user interest profiles on an ontology of research paper topics. An initial profile of a new user is formed from a correlation between his/her historical publications and any similar user profiles. Afterward, the profiling algorithm performs a correlation between paper topics and user browsing logs. A classifier based on a training set of labeled example papers is in charge of assigning browsed URLs to topics and storing each new paper in a central database. Whenever a research paper that has been classified as belonging to a topic is browsed or receives explicit feedback, this topic accumulates an interest value for the user. Recommendations are then formulated from a correlation between the users current topics of interest and papers classified as belonging to those topics.

In vast and dynamic domains such as the Web, ontology-based profiling has a number of pitfalls. Although individual profiles have to manage a considerably high number of concepts (e.g. concepts embraced by *Yahoo!* ontology), these concepts can hardly embrace the potentially infinite number of specific interests of each user (e.g. *Yahoo!* ontology can represent the concept *baseball* inside *sports*,but not going further to represent an interest in a given non-famous *baseball team* or *player*). Besides failing to capture specific user interests, ontologies impose their organization of concepts to user profiles that are not necessarily in correspondence with user views of such concepts. Moreover, users can have different perceptions of what the same ontological concept means, leading to inaccurate profile representations.

In the same line of research as ontology-based user-profiling approaches, Semantic Web (Berners-Lee *et al*., 2001) technology that has emerged in the past few years permits the construction of semantic-enriched user profiles starting from Web pages marked up with semantic meta-data. A study of the impact of the Semantic Web on personalization and user profiling was carried out in Grimnes (2003). The hypothesis behind this work was that by providing structured information, reducing ambiguity and given useful references to background information in the form of ontologies, the Semantic Web could help to solve the fundamental problems that make machine learning over the Web difficult to apply and, consequently, outperform learning from unstructured documents. However, the results of two learning algorithms provided empirical evidence that machine learning over semantic annotated texts can not be expected to outperform conventional machine learning applied to plain texts, with regards to the accuracy of the resultant model. In spite of this fact, some meaningful and potentially reusable knowledge was discovered in the same work by using a more knowledge-intensive approach like Inductive Logic Programming (ILP) (Shapiro, 1981, Section 4.2.2). As the author of this work remarks, the Semantic Web technology is very dependent on the extend to which Web pages are annotated, which requires a degree of selflessness in authors, since the annotations provided will only help others searching their pages. Because of the huge number of Web pages that require annotation, in the foreseeable future it is likely that most Web pages will remain unannotated. The Semantic Web will be, thus, of partial benefit to the problem of formulating explicit user profiles.

The following subsections concentrate on supervised and unsupervised methods to build models of user interests. As unstructured documents are generally not suitable as input for machine-learning algorithms, preprocessing steps are needed to transform text into more treatable representations. Sections 4.1 reviews techniques for representing textual documents. Provided with a suitable representation for documents, in Sections 4.2 and 4.3 learning algorithms used in user profiling are discussed.

4.1 Document representation

The most common method in the information retrieval community for achieving effective representations of documents is the Vector Space Model (VSM) (Salton *et al*., 1975). In this model,

each document is identified by a feature vector in an n-dimensional space in which each dimension corresponds to a distinct term. A given document vector has, in each component, a numerical value or weight indicating its importance determined as a function of how often the corresponding term appears in the particular document and, sometimes, how often it appears in the total document collection. Different weighting approaches are achieved by just varying this function. Hence, the representation of a document d_j according to this model is given by the document vector

$$d_j=(w_{1j}, w_{2j}, \ldots, w_{nj})$$

where w_{kj} is the weight of the kth term in the document d_j. The vector space representation ignores the sequence in which terms appear in a document; because of this fact it is also referred to as *bag-of-words* approach of document representation. The resulting representations are equivalent to the attribute-value representations commonly used in machine learning.

The following subsections describe the steps required to transform documents into vector representations, on which dominant representational approaches in personal information agents are based.

4.1.1 Feature extraction and weighting

A number of variations of the vector space model accounts for different interpretations of what a term is and how to weight terms. A typical choice for the first issue is to identify terms with all the words occurring in the document. This method is not only language independent, but also computationally efficient. However, words alone do not always represent true atomic units of meaning. For example, the phrase *machine learning* has more specific meaning than the words *machine* and *learning* separately.

Two common alternatives to take this aspect into account are the use of phrases and n-grams. The use of phrases is justified by the observation that, especially in English, many expressions are in fact multi-word terms. Automatic detection of phrases consists of identifying frequently co-occurring sequences of words or the application of natural language processing techniques. However, to avoid a drastic increase in the number of possible terms, phrases are usually domain dependent and manually provided. N-grams, on the other hand, are defined as a sequence of contiguous n letters in words, where n is a positive integer. The term n-gram is also used to refer to sequences of words of length n. In n-gram analysis, trigrams or quadgrams, i.e. $n=3$ or $n=4$, are the most frequently found sizes, as higher values of n lead to a huge number of possible terms, whereas n-grams with $n<3$ do not provide sufficient syntactical information. The advantage of using n-grams is that the set of possible terms is fixed and known in advance (e.g. the 26 letters of the English alphabet lead to 26^3 trigrams). Furthermore, n-grams are language independent and are quite robust to both morphological and spelling variations as well as mistakes. A disadvantage is that the use of n-grams results in representations that are difficult to analyze as well as in an increase in the dimensionality of the feature space.

PSUN (Sorensen & McElligott, 1995), a system for filtering USENET news articles, uses n-grams to represent profiles that the system has initially built starting from some articles a user finds interesting. Recurring words in these articles are stored by means of n-grams (*n* words found to occur after each other a significantly high number of times), and the n-grams are stored in a network of mutually attracting or repelling words, the degree of attraction being determined by the degree of co-occurrences. Then, user profiles are stored in a similar fashion to Minsky's K-lines (Minsky, 1987), connecting n-grams of different weights. Each user has multiple profiles that compete via a genetic algorithm.

Although it seems reasonable to assume that more complex text representations using phrases or n-grams instead of simply words would lead to more effective text classifiers, a number of experiments in the text categorization area has found that more sophisticated representations yield worse categorization effectiveness, thereby confirming similar results from the information retrieval area (Sebastiani, 2002). It has been argued that a probable reason for these results is that, although

indexing languages based on phrases have superior semantic qualities, they have inferior statistical qualities with respect to indexing based on single words (Lewis, 1992). That is, the frequency of occurrence in a document of individual words is usually higher than the frequency of phrases or *n*-grams.

To sum up, the choice of the text analysis level on which to base the feature extraction is always a trade-off between semantic expressiveness and representational complexity; therefore, it heavily depends on the task at hand. As most information agents perform classification checks rather than understanding or interpreting data, methods of document representation in current information agents have been mostly based on single words.

Alternatively, document representations can be enriched considering the use of a thesaurus. *Syskill & Webert*, for example, use *WordNet*, a thesaurus that contains information about synonymy and hypernymy (*is-a* relationship) (Miller, 1995) for this purpose. This agent includes as informative words for a given topic those words that have some kind of relationship with the topic name (e.g. their synonyms). From this experience it could be concluded that lexical information causes a substantial increase in precision, mainly in the first stages of learning. Nonetheless, the use of a thesaurus is limited to languages and domains where dictionaries are available and, additionally, look-up in these dictionaries could be computationally expensive for agents acting in an on-line fashion.

As regard to the second issue, how to weight terms, there are several weighting functions to determine the weight w_{kj} of the term t_k in the document d_j. Three main components can be identified as useful in determining term importance: term frequency, collection frequency and normalization factors (Salton & Buckley, 1988).

The term frequency factor reflects the importance of term t_k within a particular document d_j and is typically obtained by applying a transformation function to the term frequency tf_{kj}, which is the frequency of the term t_k in the document d_j. A straightforward weighting function is the identity function, which does not change the given term frequencies, i.e. $w_{kj}=tf_{kj}$. In Boolean weights, a weighting scheme that is commonly used in conjunction with probabilistic classifiers, 1 denotes the presence and 0 the absence of a term in the document.

The global weighting factor takes into account the importance of a term t_k within the entire collection of documents, whereas a local weighting factor refers to the term weight in the given document exclusively. A widely applied global weighting factor is the inverse document frequency defined as $idf(t_k)=\log(N/n_k)$, where N is the total number of documents in the collection and n_k is the number of documents in the collection in which the term t_k appears at least once. A well-known function for term weighting such as *tf-idf* (term frequency×inverse document frequency) (Salton & Buckley, 1988) is obtained by using the product of the term frequency and inverse document frequency:

$$tf\text{-}odf(t_k, d_j) = tf_{kj} \cdot \log\left(\frac{N}{n_k}\right). \tag{1}$$

This measure formalizes two empirical observations: (1) the more times a term occurs in a document, the more relevant it would be to determine the topic this document is about; (2) the more times the term occurs along the collection of documents, the less powerful it is to discriminate among documents.

A third factor in term weighting that appears to be useful in systems with widely varying vector lengths is the normalization factor, which equalizes the length of the document vectors. For example, according to the cosine normalization a term weight is divided by a factor representing the Euclidean vector length.

4.1.2 Dimensionality reduction

After feature extraction, the number of resultant features (terms, phrases, etc.) is usually high, so that a dimensionality reduction step is required to reduce the vector space. Dimensionality

reduction is essential for two reasons. First, the complexity of many algorithms depends not only on the number of training documents, but also on the number of features to be considered. Therefore, a space reduction can cause an algorithm to be computationally treatable. Second, even when it is generally assumed that a higher number of features results in a higher precision of a learning algorithm, the existence of many irrelevant features would cause this assumption to be incorrect.

The problem of having too many features is often referred to as the *curse of dimensionality* (Lewis, 1992). Based on a fixed number of training examples, theoretical and empirical results in machine learning have shown that there is often a maximum number of features beyond which effectiveness of an induced classifier begins to decline. Hence, removing less informative features may actually increase classification performance. The following subsections describe some widely accepted methods of performing dimensionality reduction.

Stop-word elimination

A list of *stop-words* or a *negative dictionary* is composed of a set of words that either because of their frequency or their semantic do not count with discriminative power. These words are prepositions, conjunctions, pronouns, articles and very common verbs that exhibit a similar frequency in almost every document regardless of the topic, so they have little or no inherent topical content.

The presence of stop-words has two negative connotations that justify their elimination from the set of possible terms and, as a result, the reduction on the dimensionality of the associated vector space. First, their high frequency causes any weighting function to tend to decrease the impact of the remaining words. Second, as they have little meaning by themselves, they lead to unproductive processing time.

There are both manual and automatic approaches to obtain a list of stop-words. The most common approach is to manually establish a stop-word list from the ones available on the Internet or the literature. In some applications, it may be useful to provide domain-dependent stop-word lists in addition to or even instead of a general stop-word list. This is common on information agents acting on the Web where a domain-dependent list could include words such as *www*, *link*, *click*, etc.

Stemming

In most languages, words have many morphological variants with similar semantic interpretations that can be treated as equivalents for information retrieval, as opposed to linguistic applications. For example, the words *computer*, *computers*, *compute*, *computes*, *computed*, *computational*, *computationally* and *computable* would be reduced to the single word stem *comput*. In this way, the dimensionality of a feature space can be reduced by mapping morphologically similar words onto their word stem. This task is performed by stemming or conflation algorithms that are defined as processes of linguistic normalization in which morphological variants of words are reduced to their root form, called a *stem* (Porter, 1980).

The problem of conflation has been approached with different methods, such as dictionary look up, successors elimination and suffix removal. The most common form of stemming is the elimination of suffixes and/or prefixes. In this method, stemmers range from simply removing plural endings (Harman, 1989) to handling a variety of suffixes. Algorithms such as Paice (Paice, 1990), Porter (Porter, 1980) and Lovins (Lovins, 1968) belong to this category.

In spite of the benefits stemming algorithms can provide, the stemming process also has some disadvantages. Most stemmers operate without a lexicon and then ignore word meaning, leading to a number of stemming errors. In addition to stemming errors, stems produced by suffix removal are often not words, which makes it difficult to use them for any purpose other than information retrieval.

Feature selection

Feature selection techniques are concerned with finding a subset of features, among all possible feature sets, that allows a particular learning algorithm to induce the best hypothesis according to an effective measure. Two general alternatives can be distinguished with respect to feature subset selection, filter and wrapper approaches (John *et al.*, 1994). Feature selection methods in the filter approach are independent of the learning algorithm, whereas methods from the wrapper approach employ the learning algorithm as part of the feature subset evaluation.

As classification effectiveness crucially depends on the bias of the learning algorithm, wrapper approaches are generally preferred to filter approaches from the effectiveness point of view. In contrast, it is computationally expensive to apply the learning algorithm once or even more times on the training documents for each subset being considered. In text classification, this disadvantage is particularly serious because of the large number of features.

In the filter approach, the most commonly used in text domains, a score is assigned to each feature independently by using a given scoring measure. Then features are sorted according to the assigned score and either a predefined number of the best features or those features whose score is above or below a predetermined threshold are selected to conform the vocabulary. The scoring of individual features can be performed by using several measures. A simple and yet effective global function for feature selection is the document frequency (DF), the number of documents in which a term occurs, which makes it possible to exclude all terms whose document frequency is below some predetermined threshold from the vocabulary. This measure is based on the assumption that terms that occur in only a few documents are unlikely to have any discrimination power. More sophisticated information-theoretic functions in the literature include χ^2 (chi-squared), correlation coefficient, information gain, mutual information, odds ratio, relevancy score and simplified χ^2 (Yang & Pedersen, 1997).

Mutual information, information gain and odds ratio are the most common alternatives in personal information agents. *Syskill & Webert* selects the most relevant words based on information gain and *WebWatcher* (Joachims *et al.*, 1997) uses mutual information for the same task. Odds ratio was the best at performing feature selection in the context of a browsing assistant that highlights links in Web pages with a high probability of being relevant for a user (Mladenic, 2001).

4.1.3. Structural-enriched representations

Representation schemes based on term frequency often neglect the structured or semi-structured information that is inherent to some sorts of document. It has been shown that the use of this information can enhance the performance of text classification algorithms (Fürnkranz, 1999; Hodgson, 2001) and, transitively, of personal agents applying these algorithms.

For instance, the underlying Web pages structure provided by HTML tags has been shown to be useful for indicating semantic content as well as supporting page rendering. Elements that have been taken into consideration by agents to enrich Web page representations include anchors (words that occurred in the anchor text of a link), headings (words that occurred in headings of different levels), paragraphs (words that occurred in a window in which the link occurs) and meta-tags (words included by page authors to describe the page contents).

Also e-mail messages can be considered in a broad sense as semi-structured documents that possess a set of structured fields with predefined semantics and a number of variable-length free-text fields. Some structured fields are contained in the header of an e-mail message including the sender domain (such as.com or .edu), the sender address, the recipient, the date, the e-mail type (replied, forwarded or sent directly) and the content type (having an attachment or not). In addition, unstructured text sections can be found in an e-mail message body. A careful feature selection from structured fields and the free-text body has been demonstrated to enhance performance in e-mail classification (Diao *et al.*, 2000).

4.2 Supervised learning approaches

The problem of learning classifiers that can predict the class labels of new, previously unseen, examples based on a set of training examples with assigned class labels is known as supervised learning. The supervised learning approach in text domains is usually referred to as *text learning*, *text classification* or *text categorization*. It involves assigning a set of documents $D=\{d_1, \ldots, d_n\}$ to one or more pre-defined categories $C=\{c_1, \ldots, c_m\}$, where each semantic category or class is supposed to contain documents with certain common properties.

To formalize this notion, the classification task consists of approximating the unknown total function $f:D\times C\rightarrow\{0, 1\}$, which describes how documents ought to be classified, by means of a total function $f':D\times C\rightarrow\{0, 1\}$, called the *classifier*, *model* or *hypothesis*, such that f and f' coincide as much as possible (Sebastiani, 2002). According to this definition the classification of documents has two phases. In the learning phase a supervised learning algorithm is applied to build a classifier; in turn, this classifier is used to predict the category of new documents in the classification phase. The former phase supports the acquisition and representation of user profiles, while the latter is used in the decision-making process to be performed by agents operating over these profiles.

A variety of text learning algorithms has been studied and compared in relation to user profiling. Advantages and disadvantages of a number of supervised learning algorithms applied for user profiling in personal information agents are analyzed in this section. First, decision trees and decision rules are presented as they precede chronologically to other approaches. Second, more widely spread approaches such as instance-based learning and prototype-based classifiers are described. Third, also very common and commercially successful alternatives, such as probabilistic classifiers, are examined. Finally, a brief overview of further approaches is given.

4.2.1 Decision trees

Decision trees have been extensively studied and applied in a great variety of learning tasks, including user profiling. A decision tree classifier is a recursive structure where each node is either a leaf node indicating a class or a decision node that specifies some test to be carried out on a single attribute-value, with one branch and one sub-tree for each possible test outcome. Thus, a decision tree is used to classify an instance by starting at the root of the tree and moving through it until a leaf node is reached, which provides a classification for the instance. The basic algorithm for decision tree learning, corresponding approximately to the *ID3* algorithm and its successors *C4.5* and *See5*/*C5.0* (Quinlan, 1993), recursively partitions the training examples based on an attribute that maximizes some purity criterion. *C4.5*, for instance, uses information gain for such a purpose.

Agents such as *Syskill & Webert* and *Law* (Payne *et al*., 1997) experiment with decision trees to construct user profiles, particularly employing *ID3* and *C4.5* algorithms respectively. However, results have shown that decision trees achieved less precision than other techniques that add the evidence of many features instead of isolating only some relevant ones. In *Syskill & Webert*, *ID3* was outperformed by *k*-NN and naïve Bayes techniques, which are explained later in this section. Likewise, *C4.5* was compared with instance based prototypical learning (IBPL) (see Section 4.2.3) in *Law*, demonstrating that while *C4.5* consistently produced grater coverage, IBPL was more accurate. The greater coverage of *C4.5* was caused by a default rule this algorithm employs to classify situations not covered by the rules it induces from training data. The authors of this work concluded that IBPL, an instance-based learning approach, seems to be a more suitable algorithm for user profiling than *C4.5*.

4.2.2 Rule-based classifiers

In addition to being extracted from decision trees, rules to predict the class of an example from its attributes can also be induced directly from the data. A possible method is to build individual rules by incrementally adding rule conditions until some type of class purity criteria is met. Therefore, a set of these rules can be found by repeatedly inducing a rule, removing all positive instances covered

by that rule and continuing this process until all examples are covered. Both the *AQ* family of learning methods (Michalski, 1969) and *CN2* (Clark & Niblett, 1989) use variants of this method.

CN2 algorithm was thoroughly analyzed in the interface agent *Magi*, which predicts possible user actions for e-mail messages (Payne *et al*., 1997). Experimental results show that even when *CN2* reaches similar classification results to IBPL, it has limited practical use as the time taken to induce a user profile in all experiments results in significantly longer computation time (over 12 hours in some cases). Like decision trees, decision rules tend to isolate features, which leads to reduced precision compared with the use of approaches that combine features information.

A different method for rule learning is ILP (Shapiro, 1981). The goal of ILP methods is to learn first-order rules that can contain variables and, thus, represent knowledge about relationships. The resultant rules can often be interpreted by a logic programming language such as *Prolog*. An agent using an ILP algorithm is *Poirot* (Ramírez *et al*., 2000), a Web search agent which builds rules to determine what is relevant for a given user. This agent takes keywords from bookmarks and uses them to augment queries and re-rank search engine results. It can also help in locating domain specific search engines.

RIPPER (Cohen, 1995), an algorithm belonging to the ILP family, has also been used to improve a pure collaborative approach for movie recommendation (Basu *et al*., 1998). In this case, collaborative and content information is encoded as a part of the problem representation in order to learn movie preferences of a group of users. In the meta-search engine *OySTER*, users' models induced with an ILP approach are used to describe user interests in Web pages (Müller, 2001). Rather than learn over document vectors, *OySTER* based user modeling on an ontology of concepts describing document contents, so that user interests are expressed in terms of ontological concepts.

In another application of ILP in combination with *Semantic Web* technology, some pieces of general knowledge, which seem to be reusable to classify research papers, were found starting from semantic annotated Web pages. As the authors of this work pointed out, the Semantic Web can be expected to be only partially useful to the problem of formulating explicit user profiles depending on the degree of annotation of Web pages.

RIPPER was also used for e-mail filtering showing competitive results with a prototype-based classifier (see Section 4.2.4) in terms of accuracy, both obtaining acceptable results with a relatively small number of examples (Cohen, 1996). However, when adapted to text classification *RIPPER* may not be efficient enough for the sort of on-line classification that mail filtering requires given the dynamic of this domain. Furthermore, *RIPPER* needs a post-processing of the set of constructed rules. Such a process is effective when training is performed in large batches, but personal e-mail filtering is an iterative process where the classifier must be constantly updated and, consequently, training and classification are highly intertwined.

The same drawbacks of ILP for personal e-mail filtering constrain its application for user profiling in a broader sense. An important characteristic of ILP, however, is the comprehensibility of the models that it produces. As opposed to many learning algorithms, ILP generates representations of user models that are, to some extent, transparent and understandable by users.

4.2.3 Instance-based learning

Instance-based learning (IBL) algorithms (Aha *et al*., 1991) are learning methods that simply store the presented training examples in contrast to methods that construct a general, explicit description of a hypothesis. In IBL, generalization starting from examples is postponed until a new instance needs to be classified, in which case its relationship with the stored instances is calculated to obtain the class this instance belongs to.

k-Nearest Neighbor (*k*-NN) is one of the most basic forms of IBL. According to *k*-NN, examples are represented as points in an *n*-dimensional space of terms. Therefore, the nearest neighbors are defined as those that are closer in the space according to a given distance measure such as the Euclidean distance. The fundamental assumption of this algorithm is that a document belongs to the same category as its nearest neighbors.

A slightly modified version of *k*-NN that considers a single class of positive or interesting examples for user profiling was proposed by Schwab *et al.* (2000). In this case, a user profile is learned by considering only the interesting class under the assumption that supplying an appropriate set of negative examples (i.e. examples of a non-interesting class) is problematic since the central source of information about the user is the sequence of selected documents (unselected documents may have just been overlooked or will perhaps be visited later but still might be interesting). As a standard *k*-NN procedure would always classify a new example as positive in this situation, it was modified to consider a document interesting only if its distance to at least one previously selected document is less than a given radius. This method outperformed a probabilistic one applied to positive examples.

IBPL, which is also a variant of IBL, was used in *Law* and *Magi* agents and contrasted with *CN2* and *C4.5* algorithms respectively. For e-mail classification, these agents use features from the *From* and *Subject* fields as well as the top 10 most frequent words in the e-mail body. Conclusions drawn by the comparison of the algorithms allowed authors to conclude that IBPL seems to be more suitable for this task than the fastest algorithm and accepts multivalued fields.

Parallel Exemplar-Based Learning System (PEBLS) (Cost & Salzberg, 1993), an algorithm belonging to the *k*-NN family based on a variation of the Value Difference Metric (VDM) (Stanfill & Waltz, 1986) to calculate the distance between two examples, was also evaluated in *Syskill & Webert*. Experimental results reported in this agent established that PEBLS behaves more poorly than other algorithms such as naïve Bayes classifiers (see Section 4.2.4).

Memory-based reasoning (MBR) (Stanfill & Waltz, 1986) has been employed within the *Maxims* mail filtering agent (Maes, 1994). MBR is a member of the IBL family of algorithms that stores sets of exemplars or instances in order to compare them to new unclassified situations. This agent suggests actions to be followed with e-mail messages based on actions carried out by the user under similar conditions. A situation is described as the message characteristics *(subject*, *sender*, etc.), while the situation solution is the experimented reaction of the user who receives the message (forwarding, reply, eliminate, etc.). A modified version of the MBR algorithm was used in this agent incorporating priority weightings which enabled the agent to rate elements (such as different keywords) within exemplars by importance (Kozierok & Maes, 1993). When the user acts upon a message, keywords from the message and the user action are stored as an exemplar or instance and, consequently, a user profile is composed of a set of pairs situation–action. The agent then attempts to predict the user's probable response to the arrival of a new message, by comparing this message with the stored exemplars. If a similar match is found, a confidence rating is generated with the prediction determining how the agent has to react. If the agent makes an incorrect prediction, the user is given an explanation and also an opportunity to indicate if the mistake was caused by incorrect weightings. For instance, the agent may ask: '. . . *was the message less important than I thought*? . . .', and then adjust the priority weights for the keywords found in the message accordingly.

The main disadvantage of the IBL algorithm family to be applied in user profiling is the computational complexity classifying a new document, which is linear in the number of stored training documents n and in the number of index terms t, i.e. it is bounded by $O(nt)$. For a large number of stored training documents, applying the nearest-neighbor rule can be very inefficient computationally, particularly when compared to instance-averaging methods. In contrast most agents require a rapid response from classification; for example, to classify an e-mail massage into the user folders. However, the process of finding the nearest neighbors can be sped up drastically by using efficient indexing techniques.

4.2.4 Prototype-based classifiers

Prototype-based classifiers represent each class in terms of a prototype vector in the same dimensional space as documents, making it feasible to estimate the similarity between documents and prototypes of classes. These classifiers and their variations (with different weighting functions,

similarity measures, etc.), are known as *tf-idf* classifiers because of the *tf-idf* weighting scheme typically used for document and classifier representations.

A materialization of prototype-based classifier approach is the Rocchio algorithm which was originally developed as a method for relevance feedback in information retrieval (Rocchio, 1971). In that context, it is applied to automatically optimize queries initially formulated by a user on the basis of relevance judgments of the retrieved documents. Rocchio provides a method to incorporate these judgments to the original query to obtain a refined query in an iterative information retrieval process.

Relevance feedback in information retrieval is similar to the problem of learning changes in user interests in two ways. First, the relevance feedback process of the Rocchio algorithm is identical to the process in a typical personalized recommender system. Second, the refined query in the Rocchio algorithm can be seen as a representation of long-term user interests. Because of these reasons, *tf-idf* classifiers are one of the most popular methods applied to agent construction. *Fab* (Balabanovic & Shoham, 1997), *WAIR* (Seo & Zhang, 2000) and *WebMate* (Chen & Sycara, 1998) are some agents using *tf-idf* classifiers.

Some of the agents using *tf-idf* classifiers for user profiling use a single vector for representing user interests. For example, *Butterfly* (Dyke *et al*., 1999) is an agent focused on the domain of internet relay chat (IRC) real-time text messaging. This agent samples available conversational groups and recommends interesting groups in order to help a user to find channels. User models in *Butterfly* are based on a simple term vector with positive and negative weights, which are gathered from users when they send messages containing keywords for expressing some of their interests. IRC channel contents are also represented as term vectors with weights corresponding to frequency of occurrence. The relevance of a channel to a user profile is determined by the dot product of the two vectors. *Lira* (Balabanovic & Shoham, 1995), an agent that autonomously searches the Web for interesting pages and its collaborative successor *Fab*, both used a variation of this approach to represent user interests by means of a single *tf-idf* vector. *Letizia* (Lieberman, 1995), an agent assisting Web browsing, also considers a single *tf-idf* vector as user profile. Letizia does not determine how interesting the document is, but gives a preference ordering of interest among a set of links in a Web page to guide users during browsing.

A single user profile vector is not enough to effectively model multiple user interest in diverse areas, as its precision is supposed to decrease the more documents (and consequently more categories and features) it has to model. Rather than a single vector representing interesting documents, other agents have to consider multiple vectors identifying several interest categories. For example, *SwiftFile* (Segal & Kephart, 2000) uses a *tf-idf* approach that computes prototype vectors for each folder in a user e-mail client (add-on to Lotus Notes) and a distance measure is used to estimate the similarity a new message has with each folder. This task is simplified by the fact that folders are known in advance. *WebMate* (Chen & Sycara, 1998), an agent that helps users to effectively browse and search the Web, constructs user profiles consisting of a fixed number N of vectors ($N=10$ vectors in the described implementation), each of them representing a different user interest area. When the user marks a document as interesting, if the number of different domains of interest is fewer than N it is added to the profile as a new interest domain, otherwise it is combined with the most similar one. This is a very restrictive approach in which the number of interests to be modeled into user profiles is pre-established. A more flexible approach where user profiles are represented as a set of prototype vectors whose number, size and elements change adaptively is presented in Çetintemel *et al*. (2000). In this approach, each relevant document can either create a new prototype vector or be incorporated into an existing one according to its similarity. The incorporation of a vector into a given profile is accomplished by moving this profile linearly according to a predefined parameter: if this parameter is positive, the prototype is moved toward the new vector; otherwise, the prototype is moved away from the vector. Eventually, if two profile vectors come close enough to each other in the vector space after this displacement, it is considered that they represent similar (if not the same) concepts and they are merged in a single

vector. In addition to the merge operation a deletion operation is also used to reduce the profile size. In order to implement this operation, simple statistics about the feedbacks for the documents that were incorporated into each profile vector are taken into account.

A more refined approach using multiple vectors for user profiling is presented by Widyantoro *et al.* (2001). This approach generates user profiles consisting of multiple categories of interests, each of them formed by a three descriptor structure: one descriptor to model the long-term interests and two descriptors to model short-term interests for each user interest category. In addition each description has a weight according to its importance. The positive and negative descriptors maintain a feature vector learned from documents with positive and negative feedback separately, whereas the long-term descriptor maintains the feature vector for all documents regardless of the type of feedback.

Initially, a new document is incorporated to the user model following a k-NN approach. A new interest category is created to store a new document if the content of the document is different enough from all existing categories. A decision threshold is used to determine when the highest similarity to an existing interest category is low enough to justify the creation of a new interest category. The similarity of a document to an interest category is defined as the maximum similarity between the document and either the long-term descriptor, the positive descriptor or the negative descriptor of the category. If the similarity of a document to existing categories exceeds the decision threshold for several interest category models, the categories that should be modified can be either the one with the greatest similarity or all of them. Updating of the descriptor vectors is performed by a Rocchio algorithm with different parameters in each case, while learning of short-term descriptors is performed by modifying the positive and negative descriptor weights.

The existence of many applications of prototype-based classifiers in user profiling can be explained by the simplicity of calculating the prototype vectors, which can be implemented efficiently with a time complexity linear in the number of training examples n and the number of terms t, i.e. learning time is bounded by $O(nt)$. Furthermore, classifying a new document is linear in the number of classes m and the number of terms t, i.e. classification time is bounded by $O(km)$.

In spite of their classification and learning efficiency, a main concern related to prototype-based classifiers is the degree of granularity they can achieve in the representation of user interests. In this regard, a user profile can model the user interests by a few, vague categories embracing documents about several aspects in the category (e.g. a profile showing the interest of a user in *politics* and *sports*) or a large number of over-specified categories (e.g. a profile showing the interest of a user in *presidential elections*, *Bush affairs*, *democracy issues*, *Lakers games*, *Jordan story* and *NBA drafts*).

4.2.5 Naïve Bayes

Naïve Bayes (Duda & Hart, 1973) classifiers are among the most successful algorithms for learning to classify text documents. The Bayesian approach to classify a new instance consist in assigning the instance to the most likely class within a set of classes (e.g. the class of articles a user finds interesting) given a set of attributes that describe the instance.

In particular, the naïve Bayes classifier assumes that the attributes of the items are conditionally independent given a class. In spite of this assumption, which is generally not true in texts (e.g. the probability of observing the word *learning* is greater if it is preceded by the word *machine*), the Bayesian classifier is one of the most frequently applied to agents because of its precision. It has been used by many agents for user profiling as well as compared with several approaches. As a result, not only has it been shown to be one of the best performing and efficient algorithms, but also one of the few algorithms actually applied to real-word personalization applications.

The general purpose e-mail filter *iFile* (Rennie, 2000) includes a naïve Bayes classifier to help users in organizing their e-mail in folders. The *iFile* filter uses folders as classes and messages are assumed to be correctly classified unless the user indicates otherwise by moving the message to a different folder. For this agent, adding a document to a trained model requires the recording of word occurrence statistics for that document, while classifying a document involves calculating a

log sum for each class where the size of the sum equals the number of words in the training data. Naïve Bayes classifiers were compared with *tf-idf* classifiers, *RIPPER* and others algorithms to model user e-mail reading preferences on wireless platforms in order to predict whether to forward e-mail to a user wireless e-mail device (Macskassy *et al.*, 1999). The result of this comparison was that no algorithm produced significantly superior results, with all algorithms reaching a modest standard.

In addition to e-mail filtering applications, naïve Bayes classifiers were used in agents such as *Syskill & Webert* and *NewsDude* for filtering of Web pages and news articles. *Syskill & Webert* is an agent that recommends Web pages to browse starting from the current one. It requires users to indicate which pages are interesting (on a three point scale; *hot*, *medium* and *cold*) while they are browsing. Based on this information the agent estimates a link probability of being interesting before the user visits the linked page.

NewsDude, an agent designed to compile a daily news program for users, uses a multi-strategy machine-learning approach for the induction of user models that consists of separate models for long-term and short-term interests. The short-term model is learned from the most recent observations, using the *k*-NN algorithm, under the assumption that it can adjust faster to the user changing interests. In this model news stories are converted to *tf -idf* vectors and the cosine similarity measure is used to quantify the similarity of two vectors. If a story cannot be classified by the short-term model, it is passed on to the long-term model, which models a user's general preferences for news stories. The long-term model computes predictions for stories based on a naïve Bayes classifier. In this second model, stories are represented by Boolean vectors of domain-specific features manually selected, i.e. words that are likely to be indicators for commonly recurring themes in daily news stories (approximately 200 words were selected for diverse themes).

Naïve Bayes classifiers can build accurate models based on the training examples. However, if the new examples are significantly different from any other example seen by the algorithm before, the learned model will not be able to make an accurate prediction. For this reason, Bayesian classifiers are known to be more suitable for situations where the user interests remain constant in certain domains for a long period. This is the basic assumption behind the use of a naïve Bayes classifier to model long-term user interests in *NewsDude* and a learning agent for wireless news access (Billsus & Pazzani, 2000), in conjunction with a *k*-NN algorithm to allow the agent to gain precision in modeling short-term interests. In user profiling, however, agents need to yield results from a small number of examples, sometimes based on positive evidence only.

4.2.6 Genetic algorithms

Genetic algorithms (GAs) provide a learning method motivated by the analogy with evolution of biological systems. In the same way that biological systems contain the genetic information regarding themselves in minute structures called chromosomes, GAs encode solutions to a problem by means of strings, usually strings of bits although other representations are possible. Each chromosome is given a fitness value that is a measure of its effectiveness in solving the stated problem, which is determined by an objective function.

GAs begin with a set of solutions represented by chromosomes called population. Based on an initial population, solutions are taken and used to form a new population under the assumption that the new population will be better than the old one to solve the problem. New candidate solutions are introduced into the population by applying artificial genetic operators such as crossover and mutation. The solutions used to form new solutions (offspring) are selected according to their fitness, i.e. the more suitable they are the more chances they have to reproduce. As a consequence, guided by the fitness of each chromosome, the solution space is searched systematically.

NewT (Sheth, 1994) is a news filtering agent which models user interests in newsgroups as a population of individual profiles, each one searching for articles in a reduced domain. A profile stands for a user interest while the population as a whole is expected to converge to the complete

user interests and adapt to them. In turn, a user counts with a number of agents in specific subjects, each one modeling a population of profiles. An agent is initialized by receiving some examples of some accepted and rejected articles to retrieve in a PBD approach.

The representation of a profile (chromosome) consists of a number of fields, such as newsgroup, author, location and keywords, each one weighed in proportion to its importance. The filtering process consists of finding documents that are similar to the profiles and selecting the top-scoring articles for presentation to the user. For each of these articles the user can provide positive or negative feedback having two effects. First, the profile that retrieved the article is modified based on the relevance feedback for the article. Second, the fitness associated with the profile increases or decreases for positive and negative feedback respectively.

Periodically, a population evolves from one generation to the next (e.g. every three sessions). In this case, the profiles with high fitness are retained in the next generation, while the unfit ones are eliminated. The vacancies created in the population are filled in by genetic variants of the fit chromosomes, introducing newer members into the population that might potentially match user interests better than the profiles they replace. *NewT* uses a two-point crossover, where the two points are randomly selected in the list of fields, taking two parents and producing two offspring that inherit some attributes from one parent and the rest from the other. All the fields lying between the two points are exchanged between the two parents to create two new offspring. The mutation operator modifies the newsgroup field of the profile by randomly selecting a newsgroup from amongst the most similar newsgroups. Since this field controls the search domain of the profile, its change contributes to the exploratory behavior of the population of profiles.

Amalthaea (Moukas & Maes, 1998) is an evolving, multi-agent ecosystem for personalized filtering, discovering and monitoring of information to assist its users in finding interesting information. Two different categories of agents are introduced in this system: filtering agents that model and monitor the user interests by means of augmented weighted keyword vectors; and discovery agents that model the information sources, finding and fetching the actual information the user is interested in. The evolution of the agents is controlled by their individual fitness and the overall fitness of the system. The agents in the system are ranked according to their fitness and only the top ranked, the best performers of the whole population, are allowed to produce offspring.

Each information filtering agent is based on a keyword vector that is used to assess the match between an agent and a particular document, using the cosine similarity measure. In addition to a keyword vector, these agents also contain other information such as whether it was explicitly created by the user. If it was, then the long-term interest field (a Boolean parameter) is activated indicating that the documents proposed by this agent are treated more favorably.

A new population of agents is created by copy, crossover or mutation. All these operators are applied to the evolving part of the agents, the genotype, while the phenotype contains non-evolving information. The copy operator takes an agent and creates multiple copies of it in the new population. The crossover operator randomly selects two points in the keyword vector and exchanges all fields of the two parents that lie between these points creating two new agents. The mutation operator takes the genotype of an agent as argument and creates a new agent that is a randomly modified version of its parent. The weights of the mutated keywords are modified while the new mutated keyword is a randomly selected keyword from another agent or from a recently retrieved, highly rated document. In the evolution of information filtering agents the population does not evolve together as a whole; instead, agents compete with each other inside a cluster where similar vectors are grouped.

In *Amalthaea*, as in *NewT*, the fitness is related to the user feedback. The user feedback on the relevance of an item causes the system to relate this feedback to the filtering agent that proposed the item and the discovery agent that retrieved it. These agents are penalized or awarded an amount of credit directly proportional to their proposal confidence level. These credits serve as the fitness function in both populations, which evolve separately. The higher the fitness of an agent, the more chances it gets to survive and produce offspring.

Other GA approach to extract user interests from Web documents is presented by Masayasu (1997). In this work, user interests are expressed by a set of weighed terms, with positive or negative weights in proportion to their presumed importance. Given a set of documents and user relevance judgment over these documents, the GA searches for a profile that minimizes the difference between the similarity of a document with the user profile and the relevance judgment received from the user. Both terms and weights can simultaneously evolve in a profile chromosome by using a crossover and two mutation operators. The fitness of a profile is also a function of relevance feedback as in previous agents. A completely different approach to user profiling based on GAs was proposed in the development of *Webnaut* (Masayasu, 1997). This agent implementation considers a keyword dictionary to be the search space for the problem of suggesting new information. Therefore, the GA goal is to discover which of these keywords describe the user interests. It uses two genetic algorithms, a primary GA that creates a population of sets of unique keywords randomly selected from the dictionary, and a secondary GA, that creates a population of sets of logical operators for each of the primary GA members. Afterward, each keyword set is combined with the sets of logical operators, creating a number of different queries that serve to perform a Web search. The fitness of each member in both primary and secondary GA populations is calculated as a function of the obtained search results. After evaluation, genetic operators are applied to both populations to form two new ones. Also term weights in the dictionary vary according to the explicit feedback.

As it can be concluded, many approaches have explored the similarity between the problem of modeling changing user interests over time and interest populations that evolve based on relevance feedback. A problem these approaches have to face is the necessity of having an initial population of solutions, as user interests are unknown in advance. However, as explained in Section 3, diverse methods can be useful in this matter, such as inferring an initial profile from available information about the user (bookmarks, browsing history, etc.) or from stereotypes.

In spite of the resemblance between evolution in GAs and adaptation in user profiling, user profiles in this approach are still a set of vectors. Therefore, the similarity between new documents and user profiles has to be measured by comparing vector representations as in prototype-based classifiers. Finally, a major drawback of GAs applied to the user-profiling task is due to the nature of the algorithm itself as it requires a large number of generations after a change to a user profile to meet user interests again.

4.2.7 Other approaches

Artificial neural networks (ANNs) are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. A neural network is composed of a large number of highly interconnected processing elements or neurons tied by means of weighted connections that are analogous to synapses.

ANNs were used to represent a user preferences for news articles under the content-based approach in Jennings & Higuchi (1992). For each user, a neural network is learnt with nodes representing words that appear in several articles chosen by the user and edges representing the strength of association between words that appear in the same articles. The more representative nodes are expected to predominate in the network, leading to a more accurate representation of user interests. Initially, the user retrieves articles and marks articles that are not relevant in order to construct a network that is a user model over a personal thesaurus. Based on a set of candidate nodes and links, nodes can be eventually removed according to an energy value and, similarly, links can be removed according to the connection strength. After each session with the news system, the network is modified on the basis of the articles read and rejected by the user. The read articles contribute positive evidence or energy to the network, whereas rejected ones contribute negative energy. In *Re:Agent* (Boone, 1998), an agent that filters e-mails through a neural network previously trained with feature vectors of past messages, a neural network and a k-NN classifier gave comparable results when classifying e-mail.

ANNs present a feasible and potentially suitable approach for personalization as they provide compact representations that respond to queries quickly. In contrast, ANNs can be slow to train, requiring heavy use of off-line computation to develop extensive user models. Furthermore, the network representing the user model is invisible to the user while it is worth exploring ways of explaining to users the information encoded in networks (Jennings & Higuchi, 1992).

Bayesian networks (BNs) are directed acyclic graphs in which nodes represent propositional variables and arcs represent dependencies. BNs and their extensions have been used for a variety of user modeling tasks, particularly in student modeling (Jameson, 1995). An important property of BNs is that they support the combination of the collaborative and the content-based approach (Zukerman & Albrecht, 2001). The collaborative approach may be used to obtain the conditional probability tables and the initial beliefs of a BN, which can in turn be updated in a content-based manner when the network is accessed by a user.

Bayesian and Markov networks have also been proposed to build user profiles for information retrieval (Wong & Butz, 2000), on the grounds that queries involving any subset of terms may be posed to the network, which provides a formal foundation for probabilistic inference. In this case, user profiles are represented as probabilistic networks that are induced by applying an algorithm for learning Bayesian and Markov networks. In an off-line fashion, the algorithm takes as input a sample of documents represented by a fixed set of terms that the user has marked as either relevant or non-relevant to build a network. This network is later refined when the user reads new documents. Like neural networks, the response of a BN is faster but its construction is a computationally expensive process.

4.3 Unsupervised learning approaches

Unsupervised learning or clustering is a division of data into groups of similar objects. Each group, called a cluster, consists of objects that are similar to one another and dissimilar to objects in other groups. The main difference with supervised learning is that categories are formed by the learner itself. In the absence of *a priori* knowledge, unsupervised or clustering methods seem to be ideally suited to the categorization of the user information interests.

An agent using a greedy, incremental clustering algorithm to maintain a fixed number of clusters is *WebMate*. The algorithm was analyzed and compared with a second algorithm, named a doubling algorithm, for maintenance of user profiles (Somlo & Howe, 2001). A user-driven categorization where the user explicitly provides the topics associated with new documents was used as a benchmark for the best performance. Both clustering algorithms maintain a fixed number, k, of clusters and assign each of the first k documents to its own cluster. Once k clusters have been formed, the greedy algorithm merges the closest two clusters to incorporate a new document and still maintain k clusters. In the doubling algorithm, merging allows multiple clusters to be merged, so that the total number of clusters never exceeds k, but it needs not to be equal either. The doubling algorithm results in better recall, while the greedy algorithm results in better precision. Both algorithms were outperformed by the user-driven categorization in overall performance.

Doppelganger (Orwant, 1995) introduces the idea of using clustering techniques to find groups of users dynamically, based on all available individual user models. The descriptions of the clusters created starting from explicitly represented user models can be used to create stereotypes in a community of users. Predictions for individual users are based on cluster membership and are obtained by averaging the opinions of other users in the same clusters. At cluster level, recommendations are less personalized than those computed by taking the weighted average of the opinions of a set of nearest neighbors (Herlocker *et al*., 1999). Locating neighborhoods, even when computationally expensive, facilitates the fast incorporation of up-to-date information.

One major limitation of many classical clustering algorithms to be applied in user profiling is that they assume to know the number of clusters beforehand. However, in practice, the number of clusters or user interest categories are not only unknown and unpredictable for an agent, but also

variable. Beyond user profiling, unsupervised algorithms have been used in the context of information agents for both Web usage mining as well as exploration and visualization of documents. *ACR News* (Mobasher *et al.*, 2002) illustrates the first case and *WebACE* (Boley *et al.*, 1999) the second. In *ACR News*, user browsing sessions are mapped into multi-dimensional space as vectors of URL references. Usage clusters are identified by partitioning the space in groups of transactions that are similar, based in co-occurrence patterns of URL references. An active user section is, therefore, matched against clusters to recommend interesting URLs. *WebACE* is an agent that clusters the result of a Web search into a set of categories that are presented to the user in order to continue the search process. The resultant clusters or categories are used in the automatic generation of queries, the search for similar documents and the organization of favorite pages. Unlike the agents this work focuses on, *WebACE* does not maintain user profiles, but it uses partial clustering results to assist users.

5 User profile adaptation

An implicit assumption in user profiling is the existence of long-term and persistent information needs, which are also likely to change over time (Lieberman, 1995). Adaptation of user profiles is, therefore, an essential requirement for information agents that need to be capable of adjusting to changes quickly.

Early systems, such as *Sift Netnews* (Yan & Garcia-Molina, 1999), delegate user profile adaption to users, asking them to manually modify their profiles to include or exclude interests. From a genetic algorithm point of view, on the other hand, profile adaption is tied to the evolution of solution populations. New solutions representing user interests are produced by genetic operators such as crossover and mutation, while those solutions that receive negative feedback become unfit and tend to be eliminated. Even though evolution is a more realistic approach to adapting profiles than expecting users to modify them manually, it can take a population several generations to meet the user interests again after a change.

In most agents adaptation is restricted to the incorporation of new information acquired through user feedback. The main disadvantage of this method of adaptation is that old interests are not forgotten, causing not only an exponential growth of the user profile, but also a decrease in precision since agents continue recommending information matching the old interests.

Imitating the gradual process of natural forgetting, several forgetting mechanisms have been proposed in the literature to adapt user profiles (Webb & Kuzmycz, 1996). A simple approach is to consider a time window of fixed or adaptive size and learn the description of the user interests from only the latest observations. In *CAP* (Mitchell *et al.*, 1994), an agent that learns user scheduling preferences from experience, it was empirically found that 180 examples work well for the domain and learning method applied (rules extracted from decision trees). An improvement over fixed size approaches is the proposal of adaptive window sizes, which adapt themselves based on the predictive performance of a classifier and the properties of its hypothesis (Widmer & Kubat, 1996) or changes in term distributions (Klinkenberg & Renz, 1998). Regardless of the method used to establish the size of windows, examples are either abruptly forgotten or considered equally important for the learning algorithm.

A better approximation to gradual forgetting can be achieved by defining a function that assigns weights to examples according to their appearance over time. A linear function achieves this goal in a content-based recommender component of ELFI, a Web information system (Koychev, 2000). Instead of simply forgetting examples outside a window, other approaches retain specially selected, representative examples from the past experiences (Maloof & Michalski, 1995) or useful examples (Nakhaeizadeh *et al.*, 1998).

The idea of using a dual user profile to model separately long-term and short-term interests has also been explored in information agents. *NewsDude* proposes a hybrid user-profiling approach based on a *k*-NN algorithm to build a short-term model and a naive Bayes classifier to build a

long-term model of user interests. The short-term model is built starting from recent observations to keep track of the highly dynamic interests of users in daily news stories. *NewsDude* takes advantage of *k*-NN, which can be fairly accurate to recommend news stories that follow up the stories the user has previously read with a few examples, if the new examples are very similar to the training ones. In those cases in which *k*-NN is not confident enough to make a prediction, the prediction task is delegated to the long-term model. Another approach to separately learning long-term and short-term interests is the one proposed by (Widyantoro *et al.*, 2001), where each user interest category is described by three vectors. Two of them hold positive and negative short-term descriptions of the category and the third one a long-term description. This kind of dual model attempts to provide agents with the capability of rapidly adapting user profiles to changes in user interests, without sacrificing the potential benefits of data collection over longer periods (Webb *et al.*, 2001).

6 User profile evaluation

The application of particular user modeling techniques has been almost entirely justified by their success in empirical evaluation either in isolation or compared with other approaches. Empirical methods to evaluate user profiles can be divided into those evaluating the profiles themselves, which are borrowed from the machine-learning field, and those evaluating the performance of agents acting according to these profiles, which mainly belong to the information retrieval field. It is assumed that an agent that performs well under a large number of experimental conditions is likely to perform well in an operational situation where the relevance of documents for a particular user is unknown.

As mentioned, user profiling is frequently casted to a text categorization problem and, consistently, it is evaluated from the categorization effectiveness point of view. Typically, such evaluations consist in dividing a data set into a training set and a testing set, using the former to learn a classifier and the latter to evaluate the classifier effectiveness, i.e. its capability of taking right categorization decisions (e.g. to place e-mail messages in their correct folders). This methodology has also been applied to evaluate predictive user models, where classification effectiveness is assessed through several measures such as precision, recall, accuracy, error rate and utility.

Recall and precision are notions coming from the IR field, which were later adapted to the case of document categorization. Precision is the proportion of documents placed in a category that belongs to this category, and recall is the proportion of documents belonging to a category that are actually placed in the category. Other common performance measures in text categorization are accuracy, error rate and utility. Accuracy estimates the probability of a document of being classified correctly regardless of the class label assigned. Also the error rate estimates the probability of misclassification. Utility assigns different rewards or penalties for each combination of human judgments and agent decisions (e.g. the penalty for classifying an e-mail as spam when is not could be higher than the inverse situation). Utility-based measures are closer to user evaluations than the previous measures, since they take into account some user requirements (Zukerman & Albrecht, 2001).

The problems of evaluating user profiles with text categorization effectiveness measures reside in the dynamic aspects of user profiling. First, standard evaluation methodologies generally used in the text categorization or machine-learning literature, for example *n*-fold cross-validation, are not applicable to this scenario. This is mainly due to the chronological order of the training examples, which cannot be presented to the learning algorithm in random order without bias results. Second, in trying to approximate a profile, the user interests can be assumed to be neither static nor consistent, a user might assign different labels to the same documents at different times.

In order to evaluate profiles once embedded in personal information agents, recall and precision measures are particularly suitable. In this context, recall measures the proportion of interesting items recommended by an agent out of all the interesting ones, and precision measures the proportion of interesting items among the items recommended by an agent. Also fallout is an

estimate of the probability that a document be presented to the user in spite of being irrelevant. Thus, most of the predictive models that use these evaluation measures require users to provide ratings for all the recommended items or surrogate these measures by implicit feedback. Ideally, a predictive model should have both high recall and high precision. Precision and recall are quantified into a single measure by F-measure, a weighed combination of precision and recall that produces scores ranging from 0 to 1. All the effectiveness measures described above require the knowledge about the true class labels, i.e. the user judgments of all the documents involved in the evaluation. Besides effectiveness, efficiency is also a desirable characteristic of agents. This is a measure of the performance in relation to the resources that are consumed to produce an output such as computation time and, more notably in the context of supervised learning, the user effort for providing labeled training data (e.g. explicit feedback).

The extraction of these measures from the logs obtained in a real or evaluation environment with real users is a common technique used to evaluate recommender systems. Thus, evaluation consists in predicting the relevance of examples recorded in the logs and comparing them with the real evaluations. The simulation of the recommendation process has also been proposed in a methodology for evaluating recommender systems (Montaner *et al*., 2004). Based on a list of item evaluations previously provided by a real user, this methodology simulates the recommendation process of a recommender system over time. This simulation can be seen as a progressive discovery of user profiles. At the end of the simulations, the desired evaluation measures (precision, recall, etc.) are obtained. An extension of this methodology allows one to simulate the collaboration among agents.

There is an emerging understanding that good recommendation accuracy alone does not give users of recommender systems an effective and satisfying experience (Herlocker *et al*., 2004). The usefulness of recommendations given to users can be assessed through several measures: coverage, which measures the percentage of a dataset that the recommender system is able to provide predictions for; learning rate, which measures how quickly an algorithm can produce good recommendations; novelty and serendipity, which measure the extend to which recommendations are not obvious to users; and confidence metrics, which evaluate the effect of a recommendation confidence (how sure is the recommender system that its recommendation is accurate).

7 Discussion

Mostly, the user-profiling task in information agents has been approached through the use of standard machine-learning techniques, since it can be easily seen as supervised or unsupervised learning problem in which observations of user behavior provide training examples that a machine-learning algorithm can use to build a model of user interests. Nonetheless, several authors pointed out a number of challenges for machine learning that may hinder its application in user profiling (Webb *et al*., 2001; Zukerman & Albrecht, 2001).

One difficulty of the application of supervised learning algorithms in user profiling is related to the necessity of explicitly labeled examples since the correct labels of information items may not be readily apparent from simple observation of user behavior. For some agents, the identification of labels is directly supported by the task at hand. As an example, in organizing e-mail messages labels can be assumed to be folders and an example label is the folder it belongs to, because it was either classified there by an agent or manually placed by the user. Except for the agents that ask for explicit feedback, agents have to infer example labels from observation of user behavior. Although implicit feedback methods allow agents to collect data unobtrusively based on a number of heuristics, the confidence of agents on such labels is necessarily restricted. In labeling documents as interesting or uninteresting for users, for example, it has been stated that user behavior only leads to positive evidence (Schwab *et al*., 2000). Indeed, the non-selection of an information item does not necessarily mean disinterest, which makes it difficult to capture examples even to be used in unsupervised learning. As a consequence of misclassified examples, noisy data might be introduced

in the learning process during agent operation. The problem of learning from positive and unlabeled data has been addressed based on the adaptation of traditional algorithms (Letouzey *et al.*, 2000; Li & Liu, 2003; Yu *et al.*, 2004). Furthermore, learning algorithms treat all examples equally despite the agents levels of confidence in the collected examples varying according to the observed user behavior (e.g. different values of confidence can be achieved from the numerical combination of several implicit interest indicators). A user-profiling approach should, instead, prioritize high-confidence examples to obtain more accurate user profiles.

A second issue related to the nature of examples required in user profiling is the number of examples needed to obtain accurate user models. In several agents using conventional machine-learning techniques it has been observed that the learning algorithm does not build a model with acceptable accuracy until it has seen a relatively large number of examples (e.g. in *Syskill & Webert* it takes 50 examples). Refining an initial profile acquired through other techniques, such as the stereotypes or collaborative approaches, can help to mitigate this problem.

To handle changes in user interests, learning algorithms have to face the problem of adapting user profiling over time. Adaptation in current information agents is almost exclusively related to the incorporation of new information into the user profile, not with the suppression of no-longer valid assumptions. However, if the user's interests change, the underlying learning assumption that more training data leads to improved predictive performance is not always valid. A classifier built from a large number of training documents, which accurately reflect the user's past interests, is of limited practical use and might perform substantially worse than a classifier limited to recent data that reflects the user's current interests (Webb *et al.*, 2001). Profile learning using examples within time windows of fixed or adaptive sizes as well as keeping track of long-term and short-term interests separately are possible solutions to this problem.

A further essential issue to consider in learning user profiles is the computational complexity of many learning algorithms, which prevent them from being applied to personal information agents. It took the *CN2* algorithm, for example, more than 12 hours to generate a user profile in *Magi*. In spite of the constraints imposed by the necessity of efficient user-profiling algorithms, which rule out many computationally expensive learning algorithms, these algorithms can still be applied in scenarios where the dynamic of user profiles is moderate so that models can be learned off-line. The distinctive goals of machine-learning and user profiling are clear from this case; the former is concerned with improving predictive accuracy whereas subtle accuracy improvements do not make a crucial difference to user profiling.

Learning algorithms acquire user profiles by running over examples of user interests and retain the structure of the classifier as a profile. As a result, user interests are not explicitly represented but hidden in formalisms that are specific to each learning algorithm (e.g. a decision tree, a probabilistic network, a *tf-idf* vector, etc.). Even though these classifiers support agent decision straightforwardly, the reuse of results for other purposes than those for which they were originally learned is limited. It is not easy and sometimes nearly impossible for several different decision processes to take advantage of learning results.

Finally, little attention has been paid in user profiling to the assessment of explicit, comprehensible models of user interests feasible to be interpreted by users and other agents. It has been argued that the lack of transparency in recommender systems may result in a loss of trustworthiness, because humans need to feel they are in charge and this requires the system to be understandable by them (Herlocker *et al.*, 2000). However, personal agents generate predictive models that act as 'black boxes' to most users. Even though learning algorithms, for example, provide a clear and unambiguous semantics of their output formats, these implicit representations of interests can be difficult to understand by non-expert users. In order to be truly useful, a user profile needs to be readable for users so that they can explore their profiles and verify their correctness. Moreover, since user profiles are a starting point for the creation of user communities based on shared interests, meaningful user profiles can be easily compared in the search for common interests in groups of users.

8 Conclusions

The success of personal agents in satisfying user information needs intensely relies on the learning approach taken to acquire user profiles as well as the adaptation strategy used to cope with changes in user interests. To better understand user profiling, we surveyed the literature regarding the main dimensions involved in the construction of user profiles: acquisition, learning, adaptation and evaluation.

Most user-profiling approaches in the surveyed agents had only partially addressed the characteristics that distinguish user profiling of related tasks such as text categorization or supervised learning in general. In particular, essential issues regarding user profiles such as temporal and contextual aspects of user interests have been frequently neglected. Future user-profiling approaches for successful information agents should focus not only on the above aspects but also on the assessment of comprehensible, semantically enriched user profiles, which will take information agents to the next level.

For the main dimensions of user profiling we have explained the approaches proposed and developed in current personal agents. We hope this overview will encourage the development of novel techniques covering the key aspects of user profiling as well as constituting a starting point for further research in the area.

Acknowledgement

This work has been partially supported by Fundación Antorchas.

References

Aha, DW, Kibler, D and Albert, MK, 1991, Instance-based learning algorithms. *Machine Learning* **6**(1), 37–66.

Angelides, M. C, 2003, Multimedia content modeling and personalization. *IEEE Multimedia* **10**(4), 12–15.

Ardissono, L and Maybury, MT, 2004, Preface: special issue on user modeling and personalization for television. *User Modeling and User-adapted Interaction* **14**(1), 1–3.

Balabanovic, M and Shoham, Y, 1995, Learning information retrieval agents: experiments with automated Web browsing. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, pp. 13–18.

Balabanovic, M and Shoham, Y, 1997, FAB: combining content-based and collaborative recommendation. *Communications of the ACM* **40**(3), 66–72.

Basu, C, Hirsh, H and Cohen, W, 1998, Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pp. 714–720.

Bauer, M, Dengler, D, Paul, G and Meyer, M, 2000, Programming by demonstration for information agents. *Communications of the ACM* **43**(3), 98–103.

Berners-Lee, T, Hendler, J and Lassila, O, 2001, The Semantic Web. *Scientific American*.

Billsus, D and Pazzani, MJ, 1999, A hybrid user model for News story classification. In *Proceedings of the 7th International Conference on User Modeling (UM'99)*. Springer, pp. 99–108.

Billsus, D. & Pazzani, M, 2000, User modeling for adaptive news access. *User Modeling and User-adapted Interaction, Special Issue on Deployed User Modeling Systems* **10**(2–3), 147–180.

Boley, D, Gini, M, Gross, R, Han, E-H, Hastings, K, Karypis, G, Kumar, V, Mobasher, B and Moore, J, 1999, Document categorization and query generation on the World Wide Web using WebACE. *Artificial Intelligence Review* **13**(5–6), 365–391.

Boone, G, 1998, Concept feature in Re:Agent, an intelligent e-mail agent. In Sycara, KP and Wooldridge, M (eds.), *Proceedings of the 2nd International Conference on Autonomous Agents*, pp. 141–148.

Budzik, J, Hammond, KJ, Birnbaum, L and Krema, M, 2000, Beyond similarity. In *Working Notes of the AAAI 2000, Workshop on AI for Web Search*.

Çetintemel, U, Franklin, MJ and Giles, CL, 2000, Self-adaptive user profiles for large-scale data delivery. In *Proceedings of the 16th International Conference on Data Engineering*, pp. 622–633.

Chen, L and Sycara, K, 1998, WebMate: a personal agent for browsing and searching. In Sycara, KP and Wooldridge, M (eds.), *Proceedings of the 2nd International Conference on Autonomous Agents*, pp. 132–139.

Chin, DN, 1991, Intelligent interfaces as agents. *Intelligent User Interfaces*, 177–206.

Clark, P and Niblett, T, 1989, The CN2 induction algorithm. *Machine Learning* **3**(4), 261–283.

Claypool, M, Le, P, Wased, M and Brown, D, 2001, Implicit interest indicators. In *Proceedings of the 2001, International Conference on Intelligent User Interfaces*, pp. 33–40.

Cohen, WW, 1995, Fast effective rule induction. In Prieditis, A and Russell, S (eds.), *Proceeding of the 12th International Conference on Machine Learning*, pp. 115–123.

Cohen, WW, 1996, Learning trees and rules with set-valued features. In *Proceedings of the 13th National Conference on Artificial Intelligence*, vol. 1, pp. 709–716.

Cost, S and Salzberg, S, 1993, A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* **10**(1), 57–78.

Cotter, P and Smyth, B, 2001, PTV: intelligent personalised TV guides. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pp. 957–964.

Diao, Y, Lu, H and Wu, D, 2000, A comparative study of classification based personal e-mail filtering. In Terano, T, Liu, H and Chen, ALP (eds.), *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PADKK'2000) (Lecture Notes in Computer Science, 2702)*. Springer, pp. 408–419.

Duda, RO and Hart, PE, 1973, *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Dyke, NWV, Lieberman, H and Maes, P, 1999, Butterfly: a conversation-finding agent for internet relay chat. In *Proceedings of the 1999 International Conference on Intelligent User Interfaces*, pp. 39–41.

Fürnkranz, J, 1999, Exploiting structural information for text classification on the WWW. In Hand, DJ, Kok, JN and Berthold, MR (eds.), *Proceedings of the 3rd International Symposium on Advances in Intelligent Data Analysis (Lecture Notes in Computer Science, 1642)*. Berlin: Springer, pp. 487–498.

Gauch, S, Chaffee, J and Pretschner, A, 2003, Ontology-based personalized search and browsing. *Journal of Web Intelligence and Agent Systems* **1**(3–4), 219–234.

Grimnes, GA, 2003, Learning knowledge rich user models from the Semantic Web. In Brusilovsky, P, Corbett, AT and de Rosis, F (eds.), *Proceedings of the 9th International Conference on User Modeling (Lecture Notes in Computer Science, 2702)*. Berlin: Springer, pp. 414–416.

Harman, D, 1989, How effective is suffixing. *Journal of the American Society for Information Science* **42**(1), 7–15.

Herlocker, JL, Konstan, JA, Borchers, A and Riedl, J, 1999, An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230–237.

Herlocker, JL, Konstan, JA. and Riedl, J, 2000, Explaining collaborative filtering recommendations. In *Proceedings of ACM 2000, Conference on Computer Supported Cooperative Work*, pp. 241–250.

Herlocker, JL, Konstan, JA, Terveen, LG and Riedl, JT, 2004, Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* **22**(1), 5–53.

Hoashi, K, Matsumoto, K, Inoue, N and Hashimoto, K, 2000, Document filtering method using non-relevant information profile. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 176–183.

Hodgson, J, 2001, Do HTML tags flag semantic content? *IEEE Internet Computing* **5**(1), 20–25.

Huang, Z, Chen, H. and Zeng, D, 2004, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems* **22**(1), 116–142.

Jameson, A, 1995, Numerical uncertainty management in user and student modeling: an overview of systems and issues. *User Modeling and User-adapted Interaction* **5**(3–4), 193–251.

Jennings, A and Higuchi, H, 1992, A personal news service based on a user model neural network. *IEICE Transactions on Information and Systems* **75**(2), 198–209.

Joachims, T, Freitag, D and Mitchell, TM, 1997, Web Watcher: a tour guide for the World Wide Web. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)* vol. 1, pp. 770–777.

John, G, Kohavi, R and Pfleger, K, 1994, Irrelevant features and the subset selection problem. In Cohen, W and Hirsh, H (eds.), *Proceedings of the 11th International Conference on Machine Learning*, pp. 121–129.

Kamba, T, Sakagami, H and Koseki, Y, 1997, ANATAGONOMY: a personalized newspaper on the World Wide Web. *International Journal of Human-computer Studies* **46**(6), 789–803.

Kelly, D and Belkin, NJ, 2004, Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*, pp. 377–384.

Klinkenberg, R and Renz, I, 1998, Adaptive information filtering: learning in the presence of concept drifts. *Learning for Text Categorization*, pp. 33–40.

Klusch, M, 2001, Information agent technology for the internet: A survey. *Data & Knowledge Engineering* **36**(3), 337–372.

Konstan, J, Miller, B, Maltz, D, Herlocker, J, Gordon, L and Riedl, J, 1997, GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM* **40**(3), 77–87.

Koychev, I, 2000, Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI'2000 Workshop Current Issues in Spatio-temporal Reasoning*, pp. 101–106.

Kozierok, R and Maes, P, 1993, A learning interface agent for scheduling meetings. In *Proceedings of the International Workshop on Intelligent User Interfaces*, pp. 81–88.

Krulwich, B, 1997, Lifestyle Finder: intelligent user profiling using large-scale demographic data. *AI Magazine* **18**(2), 37–46.

Kuflik, T, Shapira, B and Shoval, P, 2003, Stereotype-based versus personal-based filtering rules in information filtering systems. *Journal of the American Society for Information Science and Technology* **54**(3), 243–250.

Letouzey, F, Denis, F and Gilleron, R, 2000, Learning from positive and unlabeled examples. In *Proceedings of the 11th International Conference on Algorithmic Learning Theory (Lecture Notes in Computer Science, 1968)*. Berlin: Springer, pp. 71–85.

Lewis, DD, 1992, Representation and learning in information retrieval. PhD thesis, University of Massachusetts.

Li, X and Liu, B, 2003, Learning to classify texts using positive and unlabeled data. In Gottlob, G and Walsh, T (eds.), *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 587–594.

Lieberman, H, 1995, Letizia: an agent that assists web browsing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pp. 924–929.

Lieberman, H, 2001, *Your Wish is my Command: Programming by Example*. Morgan Kaufmann.

Lieberman, H, Nardi, BA and Wright, D, 1999, Training agents to recognize text by example. In Etzioni, O and Muller, J (eds.), *Proceedings of the 3rd Annual Conference on Autonomous Agents (Agents'99)*, pp. 116–122.

Lovins, JB, 1968, Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* **11**, 22–31.

Macskassy, SA, Dayanik, AA and Hirsh, H, 1999, EmailValet: learning user preferences for wireless email. In *IJCAI 99 Workshop: Learning About Users and Machine Learning for Information Filtering*.

Maes, P, 1994, Agents that reduce work and information overload. *Communications of the ACM* **37**(7), 30–40.

Malone, T, Grant, K and Turbak, F, 1986, The Information Lens: an intelligent system for information sharing in organizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1–8.

Maloof, MA and Michalski, RS, 1995, Learning evolving concepts using a partial memory approach. In *Working Notes of the AAAI Fall Symposium on Active Learning*, pp. 70–73.

Masayasu, A, 1997, Extraction of user's interests from Web pages based on genetic algorithm. *IPSJ SIGNotes Intelligence and Complex Systems* **97**(51), 13–18.

Massa, P and Bhattacharjee, B, 2004, Using trust in recommender systems: an experimental analysis. In Jensen, CD, Poslad, S and Dimitrakos, T (eds.), *Proceedings of the 2nd International Conference on Trust Management (iTrust 2004) (Lecture Notes in Computer Science, 2995)*. Berlin: Springer, pp. 221–235.

McNee, SM, Lam, SK, Konstan, JA and Riedl, J, 2003, Interfaces for eliciting new user preferences in recommender systems. In Brusilovsky, P, Corbett, AT and de Rosis, F (eds.), *Proceedings of the 9th International Conference on User Modeling (UM'2003) (Lecture Notes in Computer Science, 2702)*. Berlin: Springer.

Michalski, RS, 1969, On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP'69)*, vol. A3, pp. 125–128.

Middleton, SE, 2003, Capturing knowledge of user preferences with recommender systems. PhD thesis, University of Southampton.

Middleton, SE, Shadbolt, NR and Roure, DCD, 2004, Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)***22**(1), 54–88.

Miller, GA, 1995, WordNet: A lexical database for English. *Communications of the ACM* **38**(1), 39–41.

Minsky, M, 1987, *The Society of Mind*. New York: Simon & Schuster Inc.

Mitchell, TM, Caruana, R, Freitag, D, McDermott, J and Zabowski, D, 1994, Experience with a learning personal assistant. *Communications of the ACM* **37**(7), 80–91.

Mladenic, D, 2001, Using text learning to help Web browsing. In Smith, M, Salvendy, G, Harris, D and Koubek, RJ (eds.), *Proceedings of the 9th International Conference on Human-Computer Interaction, HCI International'2001, Usability Evaluation and Interface Design*, vol. 1, pp. 893–897.

Mobasher, B, Dai, H, Luo, T. & Nakagawa, M, 2002, Discovery and evaluation of aggregate usage profiles for Web personalization. *Data Mining and Knowledge Discovery* **6**(1), 61–82.

Montaner, M, López, B and de la Rosa, JL, 2002, Developing trust in recommender agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 304–305.

Montaner, M, López, B and de la Rosa, JL, 2004, Evaluation of recommender systems through simulated users. In *Proceedings of the 6th International Conference on Enterprise Information Systems*, pp. 303–308.

Morita, M and Shinoda, Y, 1994, Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 272–281.

Moukas, A and 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.

Müller, ME, 2001, Inducing rules as user models. In *Proceedings of the 8th International Conference on User Modeling, Workshop Machine Learning for User Modeling*.

Nakhaeizadeh, G, Taylor, C and Lanquillon, C, 1998, Evaluating usefulness for dynamic classification. In Agrawal, R, Stolorz, P and Piatetsky-Shapiro, G (eds.), *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pp. 87–93.

Nuri, M, 2000, Scriptor: A programming by demonstration (PBD) system for robust information extraction from Web sites. Master's thesis, Carnegie Mellon University.

Oard, D and Kim, J, 2001, Modeling information content using observable behavior. In *Proceedings of ASIST 2001 Annual Meeting*, 38–45.

O'Donovan, J and Smyth, B, 2005, Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pp. 167–174.

Orwant, J, 1995, Heterogeneous learning in the Doppelgänger user modeling system. *User Modeling and User-Adapted Interaction* **4**(2), 107–130.

Paice, CD, 1990, Another stemmer. *SIGIR Forum* **24**(3), 56–61.

Paliouras, G, Karkaletsis, V, Papatheodorou, C and Spyropoulos, CD, 1999, Exploiting learning techniques for the acquisition of user stereotypes and communities. In *Proceedings of the 7th International Conference on User Modelling (UM'99)*, pp. 169–178.

Payne, T, Edwards, P and Green, C, 1997, Experience with rule induction and k-nearest neighbour methods for interface agents that learn. *IEEE Transactions on Knowledge and Data Engineering* **9**(2), 329–335.

Pazzani, M and 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.

Quinlan, JR, 1993, *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Ramírez, JM, Donadeu, J and Neves, FJ, 2000, Poirot: a relevance-based Web search agent. In *Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search*, pp. 52–59.

Rennie, JDM, 2000, iFile: an application of machine learning to E-mail filtering. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Text Mining (KDD'2000)*.

Resnick, P, Iacovou, N, Suchak, M, Bergstrom, P and Riedl, J, 1994, GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175–186.

Rich, E, 1979, User modeling via stereotypes. *Readings in Intelligent User Interfaces* **3**(4), 329–354.

Rich, E, 1983, Users are individuals: Individualizing user models. *International Journal of Man-machine Studies* **18**(3), 199–214.

Rocchio, J, 1971, Relevance feedback in information retrieval. In Salton, G (ed.), *The SMART Retrieval System*, pp. 313–323.

Salton, G and Buckley, C, 1988, Term weighting approaches in automatic text retrieval. *Information Processing and Management* **24**(5), 513–523.

Salton, G, Wong, A and Yang, CS, 1975, A vector space model for automatic indexing. *Communications of the ACM* **18**, 613–620.

Sarwar, BM, Karypis, G, Konstan, JA and Riedl, J, 2000, Application of dimensionality reduction in recommender system-a case study. In *Proceedings of the WebKDD Workshop at the ACM SIGKKD*.

Sarwar, BM, Karypis, G, Konstan, JA and Reidl, J, 2001, Item-based collaborative filtering recommendation algorithm. In *Proceedings of the 10th International World Wide Web Conference*, pp. 285–295.

Schein, AI, Popescul, A and Ungar, LH, 2002, Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 253–260.

Schwab, I, Pohl, W and Koychev, I, 2000, Learning to recommend from positive evidence. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pp. 241–247.

Sebastiani, F, 2002, Machine learning in automated text categorization. *ACM Computing Surveys* **34**(1), 1–47.

Segal, RB and Kephart, JO, 2000, Incremental learning in SwiftFile. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 863–870.

Seo, Y-W and Zhang, B-T, 2000, A reinforcement learning agent for personalized information filtering. In Lieberman, H (ed.), *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pp. 248–251.

Shapiro, EY, 1981, An algorithm that infers theories from facts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 446–452.

Shardanand, U and Maes, P, 1995, Social information filtering: algorithms for automating 'Word of Mouth'. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'95)*, vol.1, pp. 210–217.

Sheth, BD, 1994, A learning approach to personalized information filtering. Master's thesis, Massachusetts Institute of Technology.

Somlo, G and Howe, AE, 2001, Incremental clustering for profile maintenance in information gathering Web agents. In *Proceedings of the 5th International Conference on Autonomous Agents*, pp. 262–269.

Sorensen, H and McElligott, M, 1995, PSUN: a proling system for usenet news. In *Proceedings of the CIKM'95 Workshop on Intelligent Information Agents*.

Stanfill, C and Waltz, D, 1986, Toward memory-based reasoning. *Communications of the ACM* **29**(12), 1213–1228.

Terveen, L, Hill, W, Amento, B, McDonald, D and Creter, J, 1997, PHOAKS: a system for sharing recommendations. *Communications of the ACM* **40**(3), 59–62.

Webb, GI and Kuzmycz, M, 1996, Feature based modelling: a methodology for producing coherent, consistent, dynamically changing models of agent's competencies. *User Modeling and User-adapted Interaction* **5**(2), 117–150.

Webb, GI, Pazzani, MJ and Billsus, D, 2001, Machine learning for user modeling. *User Modeling and User-adapted Interaction* **11**(1–2), 19–29.

Widmer, G. and Kubat, M, 1996, Learning in the presence of concept drift and hidden contexts. *Machine Learning* **23**(1), 69–101.

Widyantoro, DH, Ioerger, TR and Yen, J, 2001, Learning user interest dynamics with a three-descriptor representation. *Journal of the American Society of Information Science* **52**(3), 212–225.

Wong, SKM and Butz, CJ, 2000, A Bayesian approach to user profiling in information retrieval. *Technology Letters* **4**(1), 50–56.

Yan, TW. and Garcia-Molina, H, 1999, The SIFT information dissemination system. *ACM Transactions on Database Systems* **24**(4), 529–565.

Yang, Y and Pedersen, JO, 1997, A comparative study on feature selection in text categorization. In Fisher, DH (ed.), *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pp. 412–420.

Yu, H, Han, J and Chang, KC-C, 2004, Pebl: web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering* **16**(1), 70–81.

Zukerman, I and Albrecht, DW, 2001, Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction* **11**(1–2), 5–18.