
An Ontology-Based Method for User Model Acquisition

Célia da Costa Pereira¹ and Andrea G. B. Tettamanzi¹

Università Degli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
Via Bramante 65, I-26013 Crema (CR), Italy
E-mail: pereira@dti.unimi.it, andrea.tettamanzi@unimi.it

Summary. This chapter illustrates a novel approach to learning user interests from the way users interact with a document management system. The approach is based on a fuzzy conceptual representation of both documents and user interests, using information contained in an ontology. User models are constructed and updated by means of an on-line evolutionary algorithm. The approach has been successfully implemented in a module of a knowledge management system which helps the system improve the relevance of responses to user queries.

1 An introduction to User Model Acquisition

Research in User Model Acquisition (UMA) has received considerable attention in the last years. As the amount of information available online grows with astonishing speed, people feel overwhelmed navigating through today's information and media landscape. The problem of information overload leads to a demand for automated methods to locate and retrieve information with respect to users' individual interests. Unfortunately, methods developed within information retrieval leave two main problems still open.

The first problem is that most approaches assume users to have a well-defined idea of what they are looking for, which is not always the case. We solve this problem by letting fuzzy user models evolve on the basis of a rating induced by user behavior.

The second problem concerns the use of *keywords*, not *concepts*, to formulate queries. Considering words and not the concepts behind them often leads to a loss in terms of the quantity and quality of information retrieved. We solve this problem by adopting an ontology-based approach.

The approach described in this paper has been implemented and successfully tested in the framework of the Information and Knowledge Fusion (IKF) Eureka Project E!2235 [8]. The motivation of this work, was to propose an adaptive method for acquiring user models for the users of the IKF

(Information Knowledge Fusion) system. That system may be regarded as a knowledge-aware document management system offering advanced search capabilities to its users. However, a more familiar, but entirely appropriate, scenario for understanding the approach may be a document retrieval problem like the ones solved by WWW search engines.

The idea is to develop a system that knows its users. If a user has already interacted with the system, the system is capable of addressing her directly to the feature she is interested in. In order to make that possible, the first time a user logs in, the system constructs an initial model of the user. Then, for each of the user's subsequent sessions, the existing model is updated by creating or adding new information or deleting information which has become invalid. An accurate and reliable model would allow for quick and efficient answers to user demands.

In Section 2, we survey some of the existing works in the UMA field, those with common characteristics with the approach we propose here. In Section 3 we carry out a general presentation as regards our approach to the UMA; Section 4 describes a genetic algorithm for evolving user models, and Section 5 provides a discussion of the approach.

2 Related Work on User Model Acquisition

Techniques for User Model Acquisition are used in different fields. In this section we present some of the contexts in which techniques for UMA have been devised in recent years.

2.1 UMA for the Worldwide Web

In the Worldwide Web (WWW) framework, the main goal of UMA is to “learn” user interests from information obtained each time she/he logs in and chooses a page to read. In [27] the authors propose a novel method for learning models of the user's browsing behavior from a set of annotated web logs. A similar work is described in [5], in which user interest profiles are automatically constructed by monitoring user web and email habits. A clustering algorithm is employed to identify interests, which are then clustered together to form interest themes. In Jasper [7] the authors have constructed a distributed system of intelligent agents for performing information tasks over the WWW on behalf of a community of users. Jasper can summarize and extract keywords from WWW pages and can share information among users with similar interests automatically. Experimental work has been carried out to investigate user profiling within a framework for personal agents [21]. The approach is based around the electronic documents a user reads or writes. Typically, when a user marks a page as interesting (or uninteresting), or the user visits a particular page, an agent performs some analysis of that page in order to determine what feature(s) caused the user to be interested (or uninterested)

in it. More recently, in [27] a novel method is presented for learning models of the user's browsing behavior from a set of annotated web logs — i.e., web logs that are augmented with the user's assessment of whether each webpage is an *information content* (IC) page (i.e., it contains the information required to complete her task). The system uses this kind of information to learn what properties of a webpage, within a sequence, identify such IC-pages. According to the authors, as these methods deal with *properties* of webpages (or words), rather than specific URLs, they can be used anywhere throughout the web; i.e., they are not specific to a particular website, or a particular task.

2.2 UMA for Recommendation Systems

Another field in which UMA techniques are applied is recommendation systems. A recommendation system tracks past actions of a group of users to make recommendations to individual members of the group. MORSE (movie recommendation system) [10] makes personalized movie recommendations based on what is known about users' movie preferences. These informations are provided to the system by users ratings about movies they have seen on a numeric scale. MORSE is based on the principle of social filtering. The accuracy of its recommendations improves as more people use the system and as more movies are rated by individual users. In [19], a content-based recommendation component is developed for an information system named ELFI. This system learns individual interest profiles from observation considering positive evidence only. In the case in which standard classification methods are not appropriate, the authors use both a probabilistic and an instance-based approach to characterize the set of information objects a user observed to interact with in a positive way. Another approach based on collaborative information filtering [4] consists in predicting items a user would like on the basis of other user's ratings for these items. The best-performing algorithm proposed in this work is based on the singular value decomposition of an initial matrix of user ratings, exploiting latent structure that essentially eliminates the need for users to rate common items in order to become predictors for one another's preferences.

2.3 UMA for Information Retrieval

The idea behind UMA is also used in information retrieval (IR) approaches. In [11], the authors propose a system that carries out highly effective searches over collections of textual information, such as those found on the Internet. The system is comprised of two major parts. The first part consists of an agent, MUSAG, that learns to relate concepts that are semantically "similar" to one another. In other words, this agent dynamically builds a dictionary of expressions for a given concept. The second part consists of another agent, SAg, which is responsible for retrieving documents, given a set of keywords with relative weights. This retrieval makes use of the dictionary learned by

MUSAG, in the sense that the documents to be retrieved for a query are semantically related to the concept given. A related approach is proposed in [17]: an IR method is described in which what is considered is the semantic sense of a word, not the word itself. The authors have constructed an agent for a bilingual news web site — SiteF —, which learns user interests from the pages the users request. The approach uses a content-based document representation as a starting point to build a model of the user’s interests. As the user browses the documents, the system builds the user model as a semantic network whose nodes represent senses (as opposed to keywords) that identify the documents requested by the user. A filtering procedure dynamically predicts new documents on the basis of that semantic network.

2.4 UMA for Predictive Systems

UMA techniques are also used in predictive systems, as it is the case with the LISTEN Project [2]. That project comprises an “intelligent tutor” for reading, which constructs and uses a fine-grained model of students. Model construction uses a database describing student interactions with their tutor to train a classifier that predicts whether students will click on a particular word for help with 83% accuracy. The goal of the authors is to strengthen the Reading Tutor’s user model and determine which user characteristics are predictive of student behavior. Thus doing, if the tutor knows when the student will, for example, require assistance, it can provide help in a proactive way.

A recent overview on how to apply machine learning techniques to acquire and continuously adapt user models [24] describes a novel representation of user models as Bayesian networks (BNs). The generic framework considered by the autor is flexible with regard to several points such as:

- *off-line learning and on-line adaptation.* During the off-line phase, the general User Model is learned on the basis of data from other previous system users or data acquired by user studies. These models are in turn used as a starting point for the interaction with a particular new user. The initial general model is then adapted to the individual current user and can be saved after the interaction for future use when this particular user will interact the next time with the system.
- *Experimental data and usage data.* Experimental data, which mostly does not represent the “reality”, is collected in controlled environments just as done in psychological experiments. Usage data, which often includes more missing data and rare situations, are underrepresented in such data sets; such data are collected during the real interaction between users and the system.
- *Learning the BN conditional probabilities and structures.* The learning and adaptation tasks are two-dimensional: learning the conditional probabilities and learning the BN structures. To deal with sparse data the autor has introduced additional available domain knowledge into the learning procedures to improve the results, especially when the data is indeed sparse.

- *Degree of interpretability*: the author tries to ensure or at least improve the interpretability of the final learned models, e.g., by respecting whatever background information that is *a priori* available and that can be introduced into and exploited within the learning process.

3 An Ontology-Based Approach

We assume concepts and their relations to be organized in a formal ontology [13, 22]. It is convenient to logically divide an ontology into three levels¹: the *top-level* ontology, containing the most general concepts, valid for all applications; a *middle-level* ontology, comprising more specific concepts which are shared by several applications in the same broad area (e.g., Finance, Manufacturing); a *domain* ontology, comprising all the technical concepts relevant to a specific domain or sub-domain (e.g., Banking, Automotive Industry, Internet Services).

The key advantage an ontological approach offers is that one can work with *concepts* rather than with *keywords*: we will assume documents have already been processed by a suitable natural language processing technique (for instance, a semantic parser) and the appropriate sense (i.e., concept) has been associated to each word or phrase in the document².

For example, when a user is looking for a place to stay, the system should automatically consider all places in which it is possible to stay more than one day, i.e., houses, apartments, bed-and-breakfasts, hotels, etc., without requiring the user to enter all these (and possibly other) keywords. Precisely, the system will consider all the concepts subsumed by the concept “place to stay” according to the ontology. This consideration allows us to solve a common problem existing in many UMA approaches, concerning the loss of information due to the simple use of keywords instead of the concepts they stand for.

We have adopted a collaborative filtering [4, 10] approach for rating users preferences. Consequently, if a new user has a profile similar to a set of users already existing in the system, a set of documents interesting for these users is also proposed to the new user. The central idea of collaborative filtering is to base personalized recommendations for users on information obtained from other, ideally likemind, users.

¹ Of course, it is possible to consider the division on fewer or more levels depending on the problem one is solving

² The knowledge management system for which the UMA approach here described has been developed actually uses a state-of-the art semantic parser, which takes advantage of a lexical database derived from WordNet [9], and an array of pattern- and rule-based transformations to semantically annotate documents.

3.1 Document Representation

In this framework, a document is represented as a vector, whose components express the “importance” of a concept, instead of a keyword, as it will be explained in this section.

Passing from an unstructured set of (key)words seen as independent entities to an ontology of concepts, structured by the hierarchical *is-a* relation into a lattice, is not trivial. Suppose, for example, that we find in a document the three related concepts of *cat*, *dog*, and *animal*: now, *animal* is a super-class of both *cat* and *dog*. Therefore, mentioning the *animal* concept can implicitly refer both to *cat* and *dog*, although not so specifically as mentioning *cat* or *dog* directly would. Therefore, we need to devise a system to take this kind of interdependence among concepts into account when calculating levels of importance.

Choice of the Concepts to represent a Document

The main idea is to consider but the leaf concepts in the ontology, i.e., the most specific concepts only, as elements of the importance vector for a document. This choice can be justified by thinking that more general concepts (i.e., internal concepts) are implicitly taken account of through the leaf concepts they subsume. As a matter of fact, considering internal concepts too, along with their sub-classes, would create overlaps or dependencies in the importance vector, in that the component measuring the importance of an internal concept should be a function of all the components measuring the importance of all of its sub-classes, down to the leaf concepts subsumed by it.

Instead, by considering the leaf concepts only, the concepts in the document that are internal nodes of the ontology lattice are implicitly represented in the importance vector by “distributing” their importance to all of their sub-classes down to the leaf concepts in equal proportion. By doing so, we are sure that our document vector is expressed relative to concepts such that one does not subsume another. In other words, the components of the vector are independent, or orthogonal, with respect to each other.

Since an ontology (consisting of all the three layers mentioned above) can potentially contain many concepts (e.g., in the order of the hundreds of thousands), it is convenient to limit ourselves, for the purpose of calculating document importance vectors, only to the top-level ontology, whose size may be in the thousands of concepts. Anyway, it is important to remark that the approach described below is absolutely independent of which subset of the ontology one decides to restrict to, and that what we call “leaf concepts” can be defined in any way that is consistent with our premise, that is, that they form a set of mutually independent concepts (i.e., none of them subsumes, or is subsumed by, any other). The ontology lattice, cut this way, can be treated as a direct acyclic graph (*dag*), having \top (the most general concept) as its root.

Representing a Document as a Vector of Concepts

The quantity of information given by the presence of some concept c in a document depends on the depth of c in the ontology graph, on how many times it appears in the document, and how many times it occurs in the whole information repository. These two frequencies also depend on the number of concepts which subsume c . Let us consider a concept a which is a descendant of another concept b which has q children including a . Concept b is a descendant of a concept c which has k children including b . Concept a is a leaf of the graph representing our ontology. If we consider a document containing only “ ab ”, the occurrence of a in the document is $1 + \frac{1}{q}$. In the document “ abc ” the occurrence of a is $1 + \frac{1}{q}(1 + \frac{1}{k})$. As we can see, the number of occurrences of a leaf is proportional to the number of children which all of its ancestors have.

Explicit and implicit concepts are taken into account by using the following formulas:

$$N(c) = \text{occ}(c) + \sum_{c \in \text{Path}(c, \dots, \top)} \sum_{i=2}^{\text{length}(c)} \frac{\text{occ}(c_i)}{\prod_{j=2}^i \|\text{children}(c_j)\|}, \quad (1)$$

where $N(c)$ is the count of explicit and implicit occurrences of concept c , and $\text{occ}(c)$ is the number of occurrences of lexicalizations of c .

The quantity of information $\text{info}(c)$ given by the presence of some concept c in a document is given by

$$\text{info}(c) = \frac{N_{doc}(c)}{N_{rep}(c)}, \quad (2)$$

where $N_{doc}(c)$ is the number of times a lexicalization of c appears in the document, and $N_{rep}(c)$ is the total number of its (explicit as well as implicit) occurrences in the whole document repository.

Each document is then represented by a document vector I , which can be understood as a concept importance vector, whose j th component is

$$I(c_j) = \frac{\text{info}(c_j)}{\sum_{i \in \mathcal{L}} \text{info}(i)}, \quad (3)$$

where c_j is a leaf concept, and \mathcal{L} is the set of all leaf concepts.

We define similarity between documents on the basis of their vector representation, and we group them into clusters according with their similarity, in order to express user interests with respect to clusters instead of all individual documents.

3.2 Grouping Documents by Interest

The model of a user of the system contains her/his preferences about the documents of the repository. To optimize the space required to represent them, we decide to group all documents with a certain level of similarity and associate a label to that group.

Similarity between Two Documents

To compare two document vectors, we consider their shared and exclusive concepts with their respective importance degrees. We suppose that the presence of a very specific concept in a document and its absence in the other document increases the difference between them. Two functions to calculate the similarity between two documents d_i and d_j are proposed here. The first one is based on the Euclidean distance between two vectors. It is given by:

$$\text{sim}(d_i, d_j) = e^{-\lambda D(d_i, d_j)} \quad (4)$$

where $D(d_i, d_j)$ is the Euclidean distance between document vectors d_i and d_j and λ is a non negative coefficient which represents how quickly the distance between two document vectors influences their similarity.

Let I_i and I_j be the concept importance vectors corresponding to the documents d_i and d_j respectively. $D(d_i, d_j)$ is then computed as follows:

$$D(d_i, d_j) = \sqrt{\sum_k^K |I_i(c_k) - I_j(c_k)|^2} \quad (5)$$

where K is the number of leaf concepts in the ontology graph.

The second proposition is based on both the importance of concepts co-occurring in the documents and on the concepts which are present in a document and not in the other. It is given by:

$$\text{sim}(d_i, d_j) = \frac{|d_i \wedge d_j|}{|d_i \wedge d_j| + \alpha(d_i, d_j) + \alpha(d_j, d_i)} \quad (6)$$

where

$$|d_i \wedge d_j| = \sum_{c \models d_i \wedge d_j} \min(I_i(c), I_j(c)). \quad (7)$$

which quantifies the importance of the number of the concepts co-occurring in documents d_i and d_j .

In opposition, the function α allows us to quantify the importance of concepts which are exclusive to one or the other document. It is defined as:

$$\alpha(d_i, d_j) = \sum_{c \models d_i \wedge \bar{d}_j} I_i(c). \quad (8)$$

Grouping Similar Documents into Fuzzy Clusters

Due to the availability of ever larger numbers of text documents in digital form and to the ensuing need to organize them for easier use, the dominant approaches to classify documents is one of building text classifiers automatically. Approaches based on machine learning have been proposed [12, 25, 20].

Many other approaches are based on clustering algorithms [26, 23]. Recently an approach has been proposed [16], which is based on a newly developed algorithm learning very large tree-like Bayesian networks.

In this work we use a clustering-based method [14, 15] to classify documents. By adopting a clustering method for grouping documents, we aim at keeping together all documents which are similar. By doing so, we have the following advantages:

- the number of clusters is much less than the number of documents. This fact may allow us to reduce considerably the computational cost of our evolutionary algorithm and thus to accelerate the search time;
- because similar documents are grouped in the same cluster, if a document is interesting to a user, the documents in the same cluster have a big chance to be interesting too to the user. This fact presents a particular advantage in the sense that it dispenses us from making a number of comparisons which is proportional to the size of the cluster.

It is likely that a given document may fit, to some extent, into more than one cluster. Any fuzzy clustering algorithm, like fuzzy c means [3], can serve this purpose.

The fuzzy clustering algorithm we have chosen for grouping the documents is based on the similarity of the documents. The three main objectives of the algorithm are:

- maximize the similarity sim between the documents in the same cluster,
- maximize the distance between the centers of the clusters,
- minimize the number of clusters.

A document in a fuzzy cluster resulting from our algorithm may also represent more than one interest and thus belong to more than one cluster.

A fuzzy cluster i is then represented as:

$$i = \{(d_{i1}, \mu_i(d_{i1})), \dots, (d_{in}, \mu_i(d_{in}))\}, \quad (9)$$

where n is the number of documents in the cluster i , d_{ij} is the j th document in the cluster and $\mu_i(d_{ij})$ is the degree with which document d_{ij} belongs to the cluster i (its membership degree).

If K is the number of clusters, for all document d ,

$$\sum_{k=1}^K \mu_k(d) = 1, \quad (10)$$

which represents the fact that, while one document may belong to more than one cluster, the clusters form a fuzzy partition of the document space.

No cluster is empty and no cluster is the whole set of documents in the information repository. Thus, for all cluster i :

$$0 < \sum_{k=1}^n \mu_i(d_{ik}) < n. \quad (11)$$

Each cluster i has a *cluster center* v_i such that the importance of the j th coordinate $v_i(c_j)$ is calculated as:

$$v_i(c_j) = \frac{\sum_{k=1}^n \mu_i^m(d_{ik}) \cdot I_k(c_j)}{\sum_{k=1}^n \mu_i^m(d_{ik})} \quad (12)$$

where $m \geq 1$ is the weighting parameter controlling the amount of fuzziness of the clustering process.

The clustering algorithm is used during the initial phase of the acquisition process, and at any time it will be necessary. That's because, initially the information repository contains a certain number of documents, but it is possible that new documents are inserted (or existing documents are removed) during the acquisition process. For this reason, we decided to update the contents of the repository every T periods. The value of T depends on how many new documents have been inserted (or how many existing documents have been removed).

In our approach, we consider that all the documents in the same cluster must have a similarity degree greater than a threshold ϑ . The general principle of the fuzzy clustering algorithm is:

1. consider two documents d_i and d_j such that $\text{sim}(d_i, d_j) < \vartheta$. Construct then two different clusters $C_i = \{d_i\}$ and $C_j = \{d_j\}$.
2. for each other documents d_k out of any cluster compute the similarity with the center v_i of the existing clusters C_i : $\text{sim}(d_k, v_i)$. If all these similarities are less than ϑ then construct a new cluster $C_k = \{d_k\}$; else if there is at least one similarity which is greater than ϑ then d_k is inserted in the existing clusters with a membership degree in the i th cluster:

$$\mu_i(d_k) = \frac{\text{sim}(d_k, v_i)}{\sum_{z=1}^{nbclusters} \text{sim}(d_k, v_z)} \quad (13)$$

3. go back to Step 2.
4. associate to each cluster a set of the q most important key concepts occurring in the documents of the cluster.

Similarity between Document Clusters

We now extend the fuzzy relation sim introduced in Section 3.2, which gives the similarity between two document vectors, in order to compute the similarity between two clusters. Let us consider two clusters C_i and C_j and two vector documents x and y . Three methods could then be used for calculating the similarity sim between C_i and C_j :

- *optimal method* in which we use the maximum similarity of pairs. Thus,

$$\text{sim}(C_i, C_j) = \max_{x \in C_i, y \in C_j} \text{sim}(x, y). \quad (14)$$

After merging c_i and c_j , the similarity of the resulting cluster to any other cluster c_k , can be computed by:

$$\text{sim}(C_i \cup C_j, C_k) = \max(\text{sim}(C_i, C_k), \text{sim}(C_j, C_k)); \quad (15)$$

- *pessimistic method* in which we use the minimum similarity of pairs. Thus,

$$\text{sim}(C_i, C_j) = \min_{x \in C_i, y \in C_j} \text{sim}(x, y). \quad (16)$$

After merging C_i and C_j , the similarity of the resulting cluster to any other cluster C_k , can be computed by:

$$\text{sim}(C_i \cup C_j, C_k) = \min(\text{sim}(C_i, C_k), \text{sim}(C_j, C_k)); \quad (17)$$

- *cluster center method* in which we use the similarity between the two cluster centers. Thus, if v_i and v_j are the cluster centers of clusters C_i and C_j respectively,

$$\text{sim}(C_i, C_j) = \text{sim}(v_i, v_j). \quad (18)$$

3.3 Constructing the User Model

The principle of our approach to UMA is the following. When a user logs into the system, there are two possibilities: the user is either new or known. In the former case, the user will be asked to enter her/his personal data and a description of his/her interests (optional). In both cases, an evolutionary algorithm [1], described in Section 4, is automatically set to run in the background, its objective being to optimize a model for the current user, with minimal user input.

Information gathered directly from the users at the time of their registration is used to generate, for each user, an initial population of user models. From then on, it is the evolutionary algorithm which is responsible for tracking their interests as they interact with the system. Every action taken by a user in response to any proposition made by the system on the basis of a user model is interpreted as a rating of that user model.

Initializing User Models

The first step, applies to the case of a new user. Our approach consists then of using the set of cluster interests generated as described in Section 3.2 to initialize the list of cluster interests in the user profile. Summarizing, a user whose profile does not contain any group of interests (a new user), is asked

to describe, with key concepts, document title, phrases or others, his/her center of interest. In the case where this information does not consist of a list of concepts, we consider the list of the most important concepts within the information. These concepts are then compared with the descriptions of all the clusters generated as described in Section 3.2. The profile of the user is then set up by adding to the list of its interest groups the instances of clusters with features similar “enough” to those requested by the user. To each of these groups of interests it is associated a degree corresponding to the importance the cluster has to the user. Thus, if \mathcal{U} is the set of key concepts entered by the user and \mathcal{C} the set of key concepts associated to a cluster, the importance \mathcal{I} that this cluster has for the user is given by:

$$\mathcal{I}_c = \frac{|\mathcal{U} \cap \mathcal{C}|}{|\mathcal{U} \cap \mathcal{C}| + |\mathcal{U} \setminus \mathcal{C}| + |\mathcal{C} \setminus \mathcal{U}|} \quad (19)$$

The list of concepts associated to this cluster in the user profile contains only the concepts in $|\mathcal{U} \cap \mathcal{C}|$.

The second step applies to the case when the user is not new. Information about user interests is obtained automatically using a genetic algorithm which optimizes a model whose objective is to propose to the user clusters of documents which are interesting for her. The optimization procedure uses some heuristics described below and a principle of collaborative filtering which consists on updating the user profile by considering also the profiles of similar users.

Summarizing, a user profile contains:

- the personal data of the user,
- a list of fuzzy clusters which interest the user and their respective documents,
- the degree with which each cluster interests the user (its importance for the user).

Using Heuristics to update User Models

The user model is automatically updated during the entire acquisition period. Once the initial user profile has been generated, it must be updated. A user profile evolves by means of a genetic algorithm, whose details are given in Section 4, with the goal of representing as faithfully as possible the user interests. Below, we describe the methods used for updating the user profile.

The user model is updated during the entire acquisition period because we consider that the user preferences may evolve over time. The typical cases of user interest changes that need to be tracked happen when:

1. the initial clusters are changed, for example, new documents are inserted into some clusters,
2. feedback from the user is available which brings more information about his/her preferences or about changes thereof,

3. there is a user with a similar profile, but richer in terms of information content.

The first point applies to the case in which a new document is added to the repository. The document is then inserted into the cluster to which it is more similar. If the new formed cluster becomes sufficiently similar to an existing cluster in the user profile, it is automatically merged with it.

The feedback of the user allows the system to consider the changes of the user preferences during the acquisition process. Typically it occurs when a proposition based on the model is presented to the user by the system and the user does not agree with all the information reported in it. The feedback is then the result of the interaction between the system and the user.

The third point applies to the case in which there is another user in the system whose profile is sufficiently similar to the profile of the current user. In this case, all the information which is in the similar profile, but not in the profile of the current user, is also inserted into his/her profile. We can extend the similarity measure sim in order to compute the similarity between two users u_1 and u_2 as follows:

$$\text{sim}(u_1, u_2) = \frac{\sum_i \sum_j \min(\text{sim}(i, j), \text{eq}(I(i), I(j)))}{\sum_i \sum_j \text{sim}(i, j)}, \quad (20)$$

where i is the index of clusters in the profile of the user u_1 , j is the index of clusters in the profile of the user u_2 and $\text{eq}(I(i), I(j))$ returns the equality degree between the importance degree of clusters i and j . Function eq is defined as follows:

$$\text{eq}(x, y) = 1 - |x - y|. \quad (21)$$

4 A Genetic Algorithm for evolving User Models

Finding, at any point in time, a user model, i.e., the model which allows the system to make the best proposition to the relevant user, is clearly an optimization problem, which presents two main challenges:

1. as user interests may change in time, the problem is dynamic, i.e., its optimum can change and must be tracked;
2. within the scope of our work, a model of a given user can be regarded as a mapping which associates a degree of interest to each document: this general definition would make the search space really huge (the unit hypercube of dimensions equal to the number of documents in the repository), and a more specific definition has to be sought for.

Evolutionary computation [1] provides an ideal technique for approaching this difficult problem, given its ability to effectively search huge solution spaces and track moving optima in dynamical optimization problems.

A genetic algorithm has been implemented for evolving user models within the framework of this work. For each user, the algorithm maintains a distinct population of n models. The population is initialized when the user logs into the system for the first time, and updated each time the user interacts with the system, giving implicit feedback as to the quality of the relevant model. After an initial transition phase, during which the evolutionary algorithm “learns” user interests by trial and error, at steady state, the evolutionary algorithm will maintain a population made of models with a high fitness, and will effectively track any changes in the user interests by having the population evolve accordingly.

4.1 Representation

A user model is represented as a vector of membership degrees in $[0, 1]$, which describe the model’s guess at the extent to which the user is interested in each document cluster defined in the system. This choice dramatically reduces the number of degrees of freedom by exploiting the fact that a rational user must have a similar degree of interest for documents having similar contents. Of course, the dimension of such a vector may vary as new clusters are created or a number of existing clusters are aggregated into one bigger cluster; therefore, the evolutionary algorithm will have to take these possibilities into account.

4.2 Fitness Calculation

The evolutionary algorithm is on-line: each time the user interacts with the system, one of the models in the current population is randomly chosen to make its proposition to the user. If the user is satisfied (resp. unsatisfied), the fitness of the chosen model increases (resp. decreases), as well as the fitness of all other models with the same proposition. The fitness of the other models is not changed. The on-line property of our algorithm poses some interesting problems, in that a precise calculation of fitness is not available for every individual at any moment. We invite the reader to see [6] for more details.

Due to the way it is incrementally estimated, the fitness of a model is uncertain. We cope with this uncertainty by representing the (fuzzy) fitness as a possibility distribution over the interval $[0, 1]$, where 0 represents the worst possible rating for a model and 1 represents the rating of a “perfect” model, i.e., a model which always makes the right proposition to the user. The possibility distribution is represented by means of a trapezoid (a, b, c, d) , where (a, d) is the support interval and (b, c) is the core interval. When a model is created at random (e.g., when the population is initialized), its fitness is completely uncertain, and thus represented by the possibility distribution assigning a possibility of 1 to all fitness values, namely $(0, 0, 1, 1)$.

Each time a model is tried and the system receives feedback from the user, the possibility distribution (a, b, c, d) representing its fitness is updated to (a', b', c', d') according to rating $r \in [0, 1]$ ($r = 0$ means the user feedback

was completely negative, $r = 1$ completely positive) and confidence $w \in [0, 1]$ of the rating, as follows:

$$a' = (a + wr)/(1 + w); \quad (22)$$

$$b' = (b + wr)/(1 + w); \quad (23)$$

$$c' = (c + wr)/(1 + w); \quad (24)$$

$$d' = (d + wr)/(1 + w). \quad (25)$$

4.3 Genetic Operators

The algorithm is elitist, i.e., it always preserves a copy of the best individual unchanged.

The *mutation* operator consists in applying binomially distributed perturbations to each gene independently and with the same mutation probability p_{mut} .

When a model undergoes mutation, the possibility distribution (a, b, c, d) representing its fitness is updated to (a', b', c', d') according to the strength s of the mutation ($s = 0.05n/(n + 1)$, where n is an integer which measures the total perturbations caused by mutation) as follows:

$$a' = \max\{0, b' - (b - a)(1 + s)\}; \quad (26)$$

$$b' = b(1 - s); \quad (27)$$

$$c' = c(1 + s); \quad (28)$$

$$d' = \min\{1, c' + (d - c)(1 + s)\}. \quad (29)$$

The *recombination* operator is uniform crossover: each child has a 0.5 probability of inheriting each gene from either parent. We tried several probability crossover rates and 0.5 is the probability which assures a non homogeneous population after few generations.

When two models are recombined with crossover, the possibility distribution (a, b, c, d) of their offspring is obtained from the possibility distributions (a_1, b_1, c_1, d_1) and (a_2, b_2, c_2, d_2) of the two parents as follows:

$$a = b - (b_1 + b_2 - a_1 - a_2)/2; \quad (30)$$

$$b = \min\{0.95b_1, 0.95b_2\}; \quad (31)$$

$$c = \max\{1.05c_1, 1.05c_2\}; \quad (32)$$

$$d = c + (d_1 + d_2 - c_1 - c_2)/2. \quad (33)$$

Random tournament *selection* with a tournament size of 3 is responsible for selecting the pool of parents for the next generation. This tournament size value has been chosen after many experiments as it is the one which provides an acceptable selective pressure while avoiding premature convergence of the population.

The random tournament consists in extracting a random deviate from a probability distribution obtained from normalizing the possibility distribution of the fitness for each individual in the tournament; the winner is the individual for which the greatest deviate has been extracted.

4.4 Validation of the Approach

The approach described above has been implemented and experimentally validated as follows. A simulation was set up whereby user interests are randomly created and used to simulate user interaction with the system. Precisely, initially a reference model is created. While evolving, the algorithm makes propositions, which are rated according to the reference model. This allowed us to effectively verify that the evolutionary algorithm was capable of learning user interests from user feedback. Figure 1 shows a graph of the distance between the reference model and those used by the system to make its propositions during one of the tests.

In a subsequent phase, a drift effect was applied to the simulated user interests, and the ability of the algorithm to effectively track changing user interests was experimentally proved.

5 Conclusions

The evolutionary algorithm described above will not be always running. It will run only when a request is made by the user. In this case, once a feedback from the user is received, evolution will advance by one generation and then pause, waiting for the next user interaction.

This work is a significant step ahead toward the definition of a UMA approach suited for an ontology-based knowledge management framework, like the one developed in the IKF Project. This approach is novel, although inspired by several works in the literature.

We have first established the representation for the data of the problem. We have redefined the notion of document vector to consider concept, rather than keyword, occurrences in a document. Thus, a document is represented by a vector containing the importance of all the leaf concept obtained when considering the ontology only to a given depth. We have also defined a similarity function for two documents to use a fuzzy clustering algorithm.

A user model is essentially made up of personal data, a set of document clusters grouped according to their similarity and, for each of these document clusters, an associated degree corresponding to the interest of the user in that document cluster. The user models evolve in time tracking changes of user preferences. An evolutionary algorithm has been devised to obtain this behavior. The goal of that algorithm is to find, at each stage in evolution, the best models, i.e., the ones capable of making the best propositions to the user.

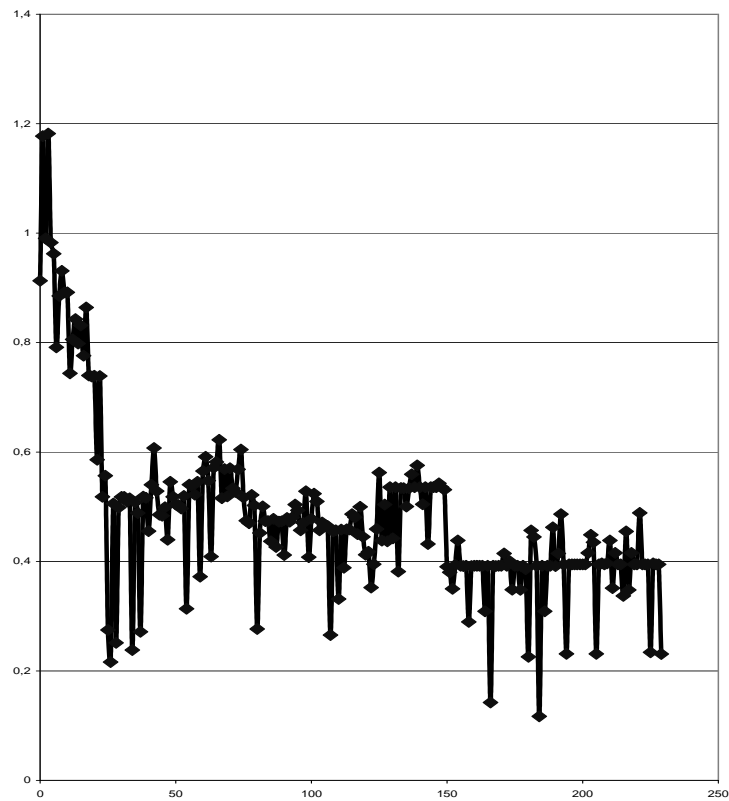


Fig. 1. Distance between the model to learn and the models obtained during the evolutionary process.

References

1. T. Bäck, D. B. Fogel, and T. Michalewicz, editors. *Evolutionary Computation (in two volumes)*. Institute of Physics, Bristol, UK, 2000.
2. Joseph E. Beck, P. Jia, J. Sison, and Jack Mostow. Predicting student help-request behavior in an intelligent tutor for reading. In *Proceedings of the 9th International Conference on User Modeling*, pages 303–312, Johnstown, PA, USA, June 2003.
3. J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

4. Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conf. on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.
5. B. Crabtree and S. Soltysiak. Identifying and tracking changing interests. *International Journal on Digital Libraries*, 2(1):38–53, 1998.
6. E. Damiani, A. Tettamanzi, and V. Liberali. On-line evolution of FPGA-based circuits: A case study on hash functions. In A. Stoica, D. Keymeulen, and J. Lohn, editors, *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 26–33, Pasadena, CA, July 19–21 1999. IEEE Computer Society.
7. N.J. Davies, R. Weeks, and M.C. Revett. Information agents for the world wide web. *BT Technology journal*, 14(4):105–114, 1996.
8. IKF Eureka Project El2235. URL: “<http://www.ikfproject.com/Project.htm>”.
9. Christiane Fellbaum (ed.). *WordNet. An Electronic Lexical Database*. The MIT Press, Cambridge, MA, 1998.
10. D. Fisk. An application of social filtering to movie recommendation. In *Software Agents and Soft Computing*, pages 116–131, 1997.
11. C. Goldman, A. Langer, and J. Rosenschein. Musag: an agent that learns what you mean. *Journal of Applied Artificial Intelligence*, 11(5):413–435, 1997.
12. Marko Grobelnik and Dunja Mladenić. Efficient text categorization. In *Text Mining Workshop of the 10th European Conference on Machine Learning ECML98*, Chemnitz, Germany, 21–24 April 1998.
13. Nicola Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal Ontology and Information Systems: Proceedings of the first international conference (FOIS’98)*, Amsterdam, 1998. IOS Press.
14. A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Dordrecht, 1988.
15. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
16. Mięczysław A. Kłopotek. Very large bayesian multinets for text classification. *Future Gener. Comput. Syst.*, 21(7):1068–1082, 2005.
17. Bernardo Magnini and Carlo Strapparava. Improving user modelling with content-based techniques. In *Proc. 8th International Conference on User Modelling*, pages 74–83. Springer-Verlag, 2001.
18. M. Rodriguez and M. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. on Knowledge and Data Eng.*, 15(2):442–456, 2003.
19. Ingo Schwab, Wolfgang Pohl, and Ivan Koychev. Learning to recommend from positive evidence. In *Intelligent User Interfaces*, pages 241–247, 2000.
20. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
21. S. Soltysiak and B. Crabtree. Automatic learning of user profiles — towards the personalisation of agent service. *BT Technology Journal*, 16(3):110–117, 1998.
22. John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Pacific Grove, CA, 2000.
23. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, Boston, MA, August 20–23 2000.
24. Frank Wittig. ML4UM for bayesian network user model acquisition. In *Proceedings of the second workshop on machine learning, information retrieval and user modeling*, June 2003.

25. Yiming Yang, Tom Ault, Thomas Pierce, and Charles W. Lattimer. Improving text categorization methods for event tracking. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 65–72, New York, NY, USA, 2000. ACM Press.
26. Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. *ACM Press*, pages 515–524, 2002.
27. Ting Shao Zhu, Russell Greiner, and Gerald Häubl. Learning a model of a web user's interests. In *Proceedings of the 9th International Conference on User Modeling*, pages 65–75, Johnstown, PA, USA, June 2003.