UNSUPERVISED ADAPTATION OF THE USER INTERESTS

Lucas Marin, David Isern and Antonio Moreno
Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) Research Group
Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Tarragona, Catalonia, Spain

Keywords: Recommender system, Fuzzy systems.

Abstract: One of the main problems in recommender systems is to ensure the quality of the user profile. This issue is particularly challenging if the user preferences may vary in time. This paper proposes a novel unsupervised algorithm to adapt dynamically the user profile, taking into account the interaction of the user with the system. The paper discusses the influence of the basic parameters of the adaptation algorithm and presents some promising preliminary results.

1 INTRODUCTION

Nowadays, the knowledge society allows the access to unlimited sources of information that help us in our daily activities. In this context, when a user has to select one alternative from a set of possibilities, an interesting facility is the inclusion of an *user profile* in order to sort all the possibilities according to his interests. A *user profile* includes the user interests (preferences) about the criteria that define those alternatives. The maintenance of this profile is made through the collection of relevant information about the user's daily work. Successful interpretation of these inputs is required to tailor systems to each individual's behaviour, habits and knowledge (Montaner et al., 2003). In addition, users interests about an issue may evolve in time due to changes in their personal conditions (*e.g.*, change of location, having a child).

Learning and adapting profiles are widely studied research areas. The adaptation requires the collection of information about how the user is interacting with the system, called *relevance feedback*. This information can be given *explicitly* by the user and *implicitly* by observing the user behaviour. Some *hybrid* systems combine implicit and explicit approaches (Nichols, 1997). On the one hand, explicit feedback is obtained when users are required to explicitly evaluate items. On the other hand, implicit feedback is obtained by monitoring the user actions and automatically inferring the user preferences. Explicit feedback has some serious limitations, being the main one that users are reluctant to spend time giving explicit information. In this paper we present an algorithm that adapts the personal profile of the users in an implicit way. We assume that the system is used regularly, so that it can learn how the preferences of the decision maker evolve. The main goal is to obtain the appropriate recommendation over time by adapting the preference functions.

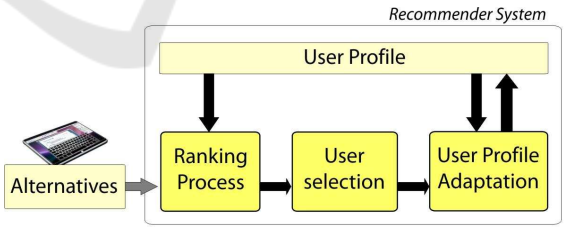


Figure 1: Architecture of the recommender framework.

As depicted in Fig. 1, the recommender system receives a set of alternatives composed by several criteria. The first step of the recommender is to rate all the alternatives taking into account the user profile in order to prioritize them according to his particular circumstances (this step is out of the focus of this paper but the reader can find more information in (Isern et al., 2010)). Then, the user selects the most appropriate alternative from this sorted list. Afterwards, the adapting algorithm collects the information provided by this selection, the set of non-selected alternatives, and all past selections, to infer which changes can be made to the current profile. As it will be described later, we have implemented a Web-based platform that permits to simulate all those stages and extract some conclusions.

Marin L., Isern D. and Moreno A..
UNSUPERVISED ADAPTATION OF THE USER INTERESTS.
DOI: 10.5220/0003180803370342
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 337-342
ISBN: 978-989-8425-40-9

2 RELATED WORK

As stated in (Porcel et al., 2009; Resnick and Varian, 1997), there are four main approaches to recommendation: content-based systems, collaborative filtering, demographic systems and knowledge-based systems. The main focus of this paper is to present a knowledge-based system as depicted in Fig. 1. The whole recommender framework includes three main parts: representing the user interests through criteria, rating and ranking the alternatives, and adapting the user profile through the experience.

Concerning the representation of the user interests, several works attempt to achieve this goal including history-based models (Montaner et al., 2003), semantic networks (Minio and Tasso, 1996), and classifiers (Boone, 1998). These models deal with a fixed number of attributes, and the aggregation of information included in the alternatives is also difficult to model.

The rating and ranking of alternatives is called aggregation of evaluations. As summarised in (Resnick and Varian, 1997), this stage generally concerns the evaluation of a resource, that can be done with the collaboration of other users (voting), or using combinations of evaluations with content-analysis. The use of the fuzzy linguistic approach has provided successful results modelling uncertain, vague and imprecise information.

The online adaptation of the user profile by learning from the interaction made by the user with the system is tackled in (Castellano et al., 2010). The user profile describes the preferences of the user on a set of criteria with fuzzy sets. A matching mechanism that uses the membership functions associated with the user profile and the resource, permits to evaluate the similarity of both elements, recommending the most similar. In addition, an adaptation mechanism permits to adapt the user profile taking into account the features of the selected resource. One of the drawbacks of this framework is the tagged information attached to each resource. This implies a subjective annotation of all the resources taking into account a set of available criteria.

3 REPRESENTATION OF THE USER PREFERENCES

The user profile stores the user preferences about some criteria using a linguistic domain (S).

Let $C = \{c_1, \ldots, c_\alpha\}$ be the set of criteria contained in the user profile.

Let $V_i = \{v_{i,1}, \ldots, v_{i,card(c_i)}\}$ be the set of values for the criterion c_i. V is the set of all values $\bigcup_{i \in c} V_i$. The number of elements depends on the cardinality of each criterion $card(c_i)$.

The profile P associates a level of preference within the linguistic domain S to each of those values ($c_i \in C$, $v_{i,j} \in V_i$ and $s_l \in S$). Then, $P(c_i, v_{i,j}) = s_l$.

This representation assumes a common linguistic domain S, whose selection supposes an important decision that influences the accuracy of results. For this reason different scenarios with both balanced and unbalanced sets of terms have been tested.

Most of the fuzzy applications use balanced sets. However, in some cases it is necessary to represent fuzzy terms that have different shapes and membership functions, as the unbalanced fuzzy set depicted in Fig. 2. This fuzzy term set S with nine unbalanced labels and triangular membership functions is used in the examples shown in this paper. In this case, the ordered set $S = \{VL, L, M, VM, AH, H, VH, AP, P\}$ permits to distinguish with precision positive examples (*e.g.*, *almost perfect* -*AP*- is slightly better than *very high* -*VH*-).

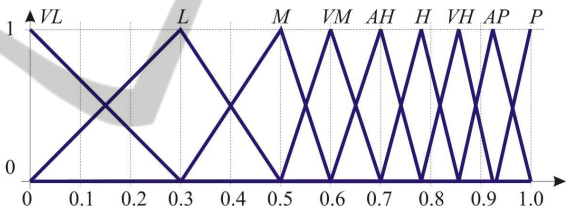


Figure 2: Fuzzy term set S with 9 unbalanced labels.

Table 1: Example set of rated and sorted alternatives.

Rank	*Distance*	*Evaluation*	*Day*	*Time*	*Parking*	*Selection*
1 (*VH*)	Far	Excellent	Sat	Aft	Blue zone	☐
2 (*H*)	Far	Bad	Sat	Night	Blue zone	☐
3 (*H*)	Very-Far	Good	Fri	Aft	No	☐
4 (*AH*)	Very-Far	Good	Fri	Aft	Free	☐
5 (*AH*)	Near	Very-Bad	Thu	Morn	Free	☐
6 (*M*)	Very-Near	Excellent	Mon	Morn	Free	■
7 (*L*)	Far	Notable	Wed	Aft	Free	☐

An example of user profile can be the interests of a patient about some healthcare centres around his town (see Table 1). In this case, the recommender system is used to prioritize sets of proposals of medical visits with specialists that a user (patient) can receive when his practitioner requires a medical test. In this case of study, each alternative contains the day of the week proposed for the appointment, the period of the day (morning, afternoon, night), the existence or not of a parking area (blue zone -payment area-, no parking, free parking), the distance from the patient's home to the medical centre (very near, near, far, very far), and

a global evaluation of the doctors of the medical centre (excellent, notable, good, normal, bad, very bad).

4 UNSUPERVISED ADAPTATION ALGORITHM

The profile adaptation approach is conducted by two processes. The first one, called *on-line* adaptation, is executed every time the user asks the system for a recommendation, and just evaluates the information that can be extracted from the current ranked alternatives. The second one, called *off-line* adaptation, is triggered after some user recommendations have been served, and uses the information of the previous ranked alternatives and selections.

On-line Adaptation Process. The on-line profile adaptation process maintains the user profile updated by evaluating each recommendation and the choice the user makes individually. The main goals of this stage are to *decrease* the preference of the attribute values that are causing non-desired alternatives to be ranked with high scores, and also to *increase* the preference of the attribute values which are important for the user but are not well judged on the current user profile. For each recommendation the system makes, two sources of information are evaluated: the selected alternative, which is the choice of the user, and the alternatives that were ranked above the selected one. Many conclusions can be extracted by evaluating this information.

We extract a set of *over ranked characteristics* by analysing the set of alternatives that were ranked above the user final selection. The process observes criterion-by-criterion the repetitions of the values in this set, but only when this set has enough elements. These two conditions introduce two parameters named *t* and *mo*. The first parameter identifies relevant characteristics, when a value that does not appear in the user selection is repeated on the over ranked alternatives in a percentage over *t*. When the over ranked set of alternatives contains few elements, the analysis of this set can produce erroneous evaluations. The parameter *mo* fixes a minimum number of elements to perform this stage (by default, 5). Those characteristics are used to *decrease* the level of preference of those attribute values pointed out by the characteristics. The intensity of that decrease is regulated by the number of repetitions: if a characteristic is repeated many times, we have more evidence to decrease the preference on that attribute value.

For instance, Table 1 shows a case in which the user selects the sixth alternative. Taking a threshold *t* of 40%, we obtain the tuple $\langle$*Day of Week, Sat, 2*$\rangle$ as an example of over ranked characteristic, being *Day of Week* the attribute name, *Saturday* a possible attribute value, and *2* the times it appears among the set of over ranked alternatives. Note that $\langle$*Day of Week, Thursday, 1*$\rangle$ can not be considered as an over ranked characteristic since it only appears once, thus it just represents a 20% of the over ranked alternatives.

We call *selection characteristics* the features extracted from the user final selection that do not appear on the set of over ranked alternatives more than a given number of times. In this case, the use of the repetition threshold *t* is opposite as when extracting over ranked characteristics. The adaptation process will only consider as selection characteristics the ones which appear on the over ranked alternatives with a percentage lower than said threshold. Selection characteristics are used to *increase* the level of preference of the attribute values pointed out by the characteristics. The intensity of that increase is greater the fewer times that characteristic appears among the over ranked characteristics. Using the previous example, we have more evidence to increase the preference of the value *Monday* of the attribute *Day of week* than the one of the value *Morning* of the attribute *Time*, since the first does not appear among the over ranked alternatives whereas the second appears once.

Off-line Adaptation Process. The previous process gives an immediate response to a single user interaction with the recommender system. This stage analyses past elements (non analysed over ranked alternatives and past selections) that permits to complement the on-line stage.

When the user selects the best ranked alternatives, there are not enough over ranked alternatives to analyse. A buffer stores all alternatives non analysed in the previous section, which are analysed a posteriori as a whole. When the number of saved over ranked alternatives is adequate, characteristics are extracted. The parameter *mo* is also used here to describe the minimum number of over ranked alternatives required to start extracting characteristics. Those characteristics are used to *decrease* the preferences in the same way as it is done in the on-line process: characteristics with a repetition value over the defined threshold are decreased. After the stored over ranked alternatives have been treated, they are erased from the temporal buffer.

On the other hand, user selections are stored and, after a certain number of choices have been made, they are evaluated. The number of selections needed for an evaluation is described by the parameter *h*. By extracting characteristics from that set of stored selec-

tions, the most repeated attribute values, which are the ones of most interest to the user, can have its preference *increased*. After evaluating them, they are erased from the buffer.

Adaptation Mechanism. After the on- and the off-line stages, many characteristics can be found and the system can deduce a large amount of changes to do in the profile. Many of those changes, also named *adaptations*, can be incorrect. We have addressed this problem by restricting the number of adaptations that can be made to the profile for each execution of the adaptation process. The parameter *pc* limits the number of increases and decreases which can be made to a profile per adaptation step (a value of *2* allows a total of 4 changes: 2 increases and 2 decreases).

When a large number of adaptations is being considered, only the more *evident* ones are performed. The ones with more repetitions will be decreased. For the increases, the evidence is measured by counting the number of over ranked alternatives which do not have those characteristics, so the ones with a greater counter will be increased. A *signed counter* regulates the final increase/decrease of the preference of the profile. When the adaptation process detects an evidence for a possible adaptation, it increases/reduces a counter, initially set to 0, associated to that attribute value. When this counter reaches the parameter *k*, the increase/reduction is finally performed over the preference in the profile and the counter is reset to 0.

5 EVALUATION

Web-based Platform. In order to test and evaluate the recommender framework presented in this paper, a Web platform has been implemented [1]. It permits the definition of a problem (set of user criteria, aggregation and adaptation parameters), the definition of an initial profile, the definition of an ideal profile that we aim to learn, the random generation of a corpus of alternatives for the problem, the simulation of tests of user interaction with the system, and the visualisation of the evolution of the user profile.

The accesses to the platform are regulated by a user name and a password, allowing the maintenance of multiple users with different profiles for each recommendation domain. This approach permits to analyse the variability of results under different circumstances in the aggregation and adaptation stages.

Users can upload data files with information about the alternatives that must be analysed in order to make a decision. The platform stores that information in a database, letting users define a decision making problem on those alternatives.

[1]Website: http://itaka2-deim.urv.cat/recommender/

General Performance. In order to evaluate the framework, we propose to iteratively measure the distance between the current user profile and the ideal profile that we want to learn. Iteratively and automatically, the system selects the alternative that best fits the ideal profile, which is the one that the user which we want to have its profile learnt would choose. Then, the selection and the over ranked alternatives are used by the adaptation processes to evaluate which changes should be performed. To compare the results of different tests, several simulations have been performed taking into account different initial profiles, but maintaining the ideal user profile. As it can be seen, distances between both profiles decrease over time.

The distance between the current (P) and the ideal (I) profiles is calculated as follows:

$$dist(I,P) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{m_j}\sum_{k=1}^{m_j}\frac{|COG_x(P(j,k)) - COG_x(I(j,k))|}{|COG_x(s_0) - COG_x(s_T)|} \quad (1)$$

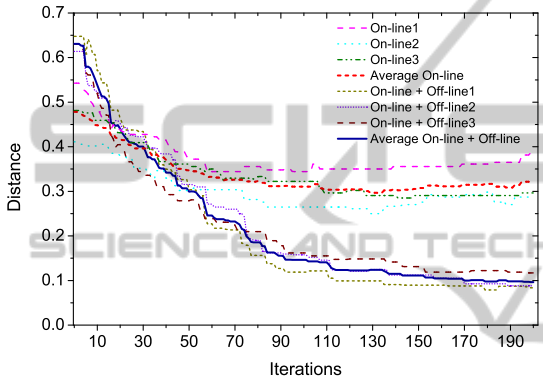
Here, n is the number of attributes, m_j is the cardinality of the attribute j, COG_x evaluates the coordinate x of the centre of gravity of a linguistic label (Isern et al., 2010), $P(j,k)$ and $I(j,k)$ return the linguistic preference value of the attribute j for a given value k in profiles P and I respectively. With this function we obtain an average of the distances between all pairs of labels. The distance is normalised by using the extreme values of the fuzzy set $S = \{s_i\}, i \in \{0,\ldots,T\}$. The distance is 0 when both profiles are identical. The maximum distance is 1 when all values contained in the user and the ideal profiles have just the opposite labels (s_0 and s_T). Moreover, this function is commutative ($dist(P,I) = dist(I,P)$).

The conditions of all tests were the following: 3000 alternatives initialised randomly, blocks of 15 alternatives per iteration (200 iterations), profiles (ideal and initial) initialised manually, and the term set of Fig. 2. Although we will analyse the influence of the main parameters later, these tests have been performed using the extraction characteristics threshold t of 30%, with the level of evidence k +/-5 to consider a change in a value, allowing only 2 changes (pc) in each sense in one iteration, storing up to 5 user selections (h) and needing at least 5 over ranked alternatives (mo) to extract characteristics. In all cases, three simulations with three initial profiles and sharing the ideal profile have been conducted.

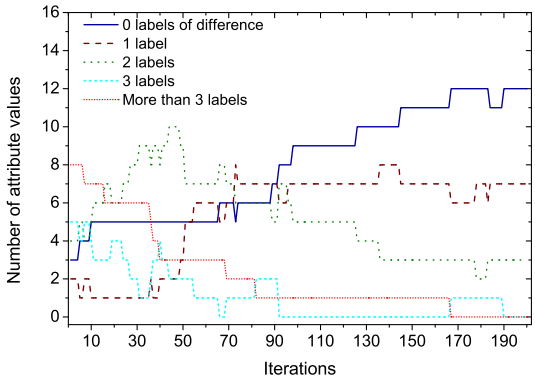
Fig. 3(a) compares the general performance of the adaptation algorithm using the on-line process, and

using the on-line combined with the off-line stage. The general performance accomplished when both processes are working together to adapt the profile is much better than when just using the on-line process.

Moreover, Fig. 3(b) shows the labels of distance between the values of the current and ideal profiles (*e.g.*, the distance between *H* (*high*) and *P* (*perfect*) is 3). The figure compares how many attribute values there are with difference 0 (correctly classified), 1, 2, 3 and more than 3 labels. It can be seen how the number of the most misclassified labels (distance 3 and more) decreases drastically, while the set of well classified attribute values (distances 0 and 1) increases.

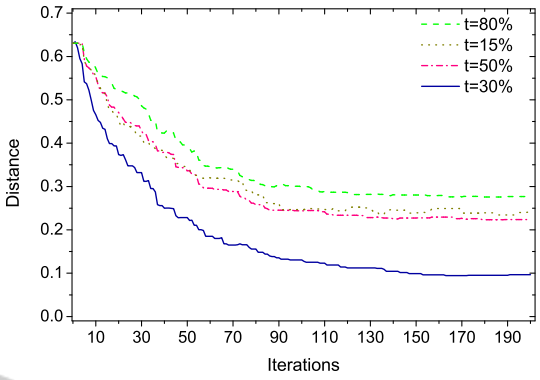


(a) Distances between user and ideal profile.

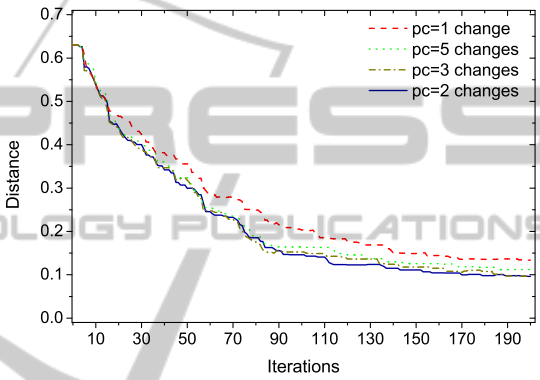


(b) Quantitate study of the labels misclassified in the comparison of the current profile with the ideal profile.

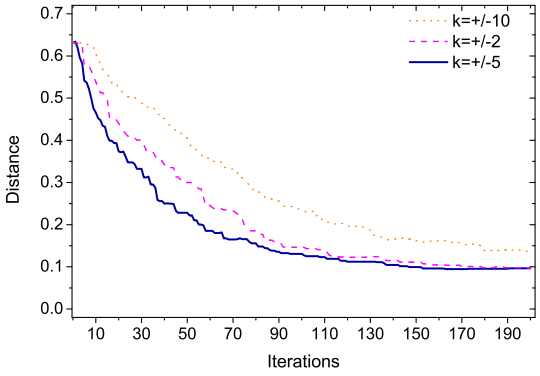
Figure 3: Results of the case study.



(a) Influence of the characteristics extraction threshold.



(b) Influence of the maximum number of preference changes at each iteration.



(c) Influence of the level of evidence during tests.

Figure 4: Parameters of the adaptation algorithm.

Discussion of Results. The proposed adaptation algorithm introduces several parameters that should be properly customised. This section explains the influence of some of these parameters in the final result, although it should be noted that exact parameter values may vary depending on the context in which the adaptation is taking place (number of attributes, number of values per attribute, etc.).

Percentage of Over Ranked Alternatives. Over ranked characteristics are extracted by considering a threshold defined as a percentage over the number of over ranked alternatives. Fig. 4(a) shows that after several tests (using the threshold values of 15%, 30%, 50%, and 80%), a threshold of 30% was the one that got better results. Having a greater threshold implies that the extracted characteristics need to have a greater repetition value, thus fewer characteristics are extracted, reducing the number of possible

changes in the profile. On the contrary, the extraction of valuable information is compromised as the threshold decreases, because incorrect characteristics (which usually have a low number of repetitions) are extracted more easily.

Number of Changes of Preferences at each Iteration (pc). On the adaptation mechanism section, a parameter for controlling how many increases/decreases of preference can be done at each adaptation iteration was introduced. Fig. 4(b) compares the performance of the system with different values. Setting that value to *2* proved to give better results than using a lower or greater number, letting the system make up to two increases and two decreases on any attribute value preference of the profile. This number can vary on the domain that we apply the recommender framework but, intuitively, a lower value permits to make the preference changes for which we have a greater supporting evidence.

Number of Evidences to Change a Preference (k). This parameter varies the necessary number of evidences to increase/decrease a preference on the profile. In this case, with lower values, the number of required evidences is lower, and the quality of these changes is also compromised. On the other hand, higher values for this parameter hinder making changes on preferences, so it requires more iterations (selections) to reach a near-ideal profile. Those differences, however, tend to decrease as the number of iterations increases. Fig. 4(c) shows these behaviours and how an intermediate value such as +/-5 exhibits a good performance.

6 CONCLUSIONS AND FUTURE WORK

The Web-based recommender system proposed in this paper has been designed as a knowledge-based framework that permits to solve decision problems using MCDA techniques. The separation between the domain-related (*e.g.*, adaptation thresholds) and the domain-independent data (*e.g.*, rating, ranking and adaptation processes) permits to understand the evolution of values, as well as to reuse the same methods in a wide range of problems.

The system has been prepared to make automatic simulations, which permit to study the influence of several parameters of the proposed adaptation algorithm.

The creation of this platform is the first step in a project that aims to develop techniques to adapt the user profile according to the selections made in the recommendation process. We consider that having automatic algorithms to update the preferences is required in many application domains, where the preferences may change dynamically over time.

ACKNOWLEDGEMENTS

This work has been supported by the Universitat Rovira i Virgili (predoctoral grant of L. Marin, post-doctoral contract of D. Isern, and the project 2009AIRE-04), the Spanish Government (PlanE) and the DAMASK project (TIN2009-11005).

REFERENCES

Boone, G. (1998). Concept features in Re:Agent, an intelligent Email agent. In *Proc. of AGENTS 98*, pages 141–148, Minneapolis, US. ACM Press.

Castellano, G., Castiello, C., Dell'Agnello, D., Fanelli, A. M., Mencar, C., and Torsello, M. A. (2010). Learning Fuzzy User Profiles for Resource Recommendation. *Int. J. of Uncert., Fuzziness and Know.-Based Syst.*, 18(4):389–410.

Isern, D., Marin, L., Valls, A., and Moreno, A. (2010). The Unbalanced Linguistic Ordered Weighted Averaging Operator. In *Proc. of the FUZZ-IEEE 2010*, pages 3063–3070, Barcelona, Catalonia. IEEE Computer Society.

Minio, M. and Tasso, C. (1996). User Modeling for Information Filtering on Internet Services: Exploiting an Extended Version of the UMT Shell. In *Proc. of UM 96*, Kailua-Kowa, US.

Montaner, M., López, B., and de la Rosa, J. L. (2003). A Taxonomy of Recommender Agents on the Internet. *Art. Intell. Rev.*, 19(3):285–330.

Nichols, D. M. (1997). Implicit Rating and Filtering. In *Proc. of the 5th DELOS Workshop of Filtering and Collaborative Filtering*, pages 31–36, Budapest, Hongary. European Research Consortium for Informatics and Mathematics.

Porcel, C., Moreno, J. M., and Herrera-Viedma, E. (2009). A multi-disciplinar recommender system to advice research resources in University Digital Libraries. *Exp. Syst. with Appl.*, 36(10):12520–12528.

Resnick, P. and Varian, H. R. (1997). Recommender Systems. *Commun. of the ACM*, 40(3):56–58.