Smart User Models for Ambient Recommender Systems

Gustavo González[1], Cecilio Angulo[2], Beatriz López[1], Josep Lluís de la Rosa[1]

[1] Institute of Informatics and Applications. Agents Research Lab
University of Girona. Campus Montilivi, Building P4. E-17071 Girona, Spain
{gustavog, blopez, peplluis}@eia.udg.es
[2] GREC. Knowledge Engineering Research Group. Technical University of Catalonia.
Campus Vilanova, Building VG2. E-08800 Vilanova i la Geltrú, Spain
cecilio.angulo@upc.es

Abstract. Smart User Models improve the quality of service personalization, they reduce the overload of processed information and can establish the user's preferences and emotional state in the next generation of open, distributed and networked environments in Ambient Intelligence. In this paper, we combine the flexibility of intelligent agents with the information processing capabilities of support vector machines in order to obtain the user's most relevant preferences, tastes and behaviors through an incremental learning process. A multi-agent architecture has been developed in order to manage services and user preferences in several domains. The set of functionalities and capabilities of each agent in the multi-agent Smart User Model is described and illustrated with a case study.

1 Introduction

The most generalized vision of Ambient Intelligence is an open, networked world with all kinds of objects. However, the center of this world is the user who needs to satisfy his/her preferences through personalized services in this open, distributed, heterogeneous, interconnected environment. The idea of the average user of Ambient Intelligence is the situational human being who makes decisions not only based on his/her preferences, tastes and interests, but also influenced by his/her perceptions of the context. The context is a multidimensional parameter that includes variables such as time, place, weather, and emotions.

Personalization in Ambient Recommender Systems can be achieved through internal representations of the users in the devices, i.e. user models. These artificial representations have been studied by the Human-Computer Interaction community for years, however, developing applications in open environments such as Ambient Intelligence and the Internet poses the challenge of making continuously updated user models and, what is more important, this implies using a unique model for all applications with which the user interacts. In order to contribute to this kind of future user models, we have combined the synergy of smart adaptive systems, intelligent agents and Support Vector Machines to

develop a Multi-agent Smart User Model (*SUM*). The *SUM* is able to deal with any type of objective, subjective or emotional user feature, whether explicit or implicit, in several domains.

Such artificial representations have been mainly studied by the Human-Computer Interaction community for years, however, the development of applications in open environments such as Ambient Intelligence and Internet poses the challenge of modelling the user once and continuously and, what is more important, the use of a unique model for all applications with which the user interacts. In order to contribute to such kind of future user models, we combine the synergy of smart adaptive systems, intelligent agents and Support Vector Machines to develop a Multi-agent Smart User Model (*SUM*). The *SUM* is able to deal with any type of objective, subjective or emotional user feature, whether explicit or implicit, in several domains. It continuously increases the knowledge about user preferences and interests in an unobtrusive way. Agent flexibility (reactivity, pro-activity and social-abilities) is a cornerstone in the implementation of *SUM* in Ambient Recommender Systems.

Ambient Recommender Systems pro-actively operate like an 'intelligent adviser' on the behalf of the users. Their added value is based on their ability to give suitable advice, recommendations, or predictions that are of interest to each user in his/her particular context. Particularly, our challenge is to develop a unique user model that influences the decision process of recommender systems in order to give the user relevant advice, recommendations or suggestions about an item or service in several domains. Figure 1 shows different geometric figures, which represent the domains (for instance restaurants, movies, music, news) at the domain level. These interact by means of wrapper agents with corresponding geometric figures which represent different recommender systems (restaurant recommender system, movie recommender system, music recommender system, news recommender system) at the computational level. At this level we can see the Multi-agent Smart User Model which interacts with the user through hand-held or desktop devices.

We are concerned with the ongoing work of an unobtrusive adaptive user model. Since our approach to user modeling encompasses communication (interoperability) and coordination (coherent actions) with recommendation processes, agent technology provides the appropriate flexibility to carry out all these activities. We present a developed prototype of a user model as a multi-agent system, which is able to provide services through its proactive, reactive and social capabilities in relation to other agents, applications and users. The Smart User Model is able to provide information about the user when a new application in the environment requires it (reactivity); it is able to search for new applications in which the user may be interested (pro-activity); and it can interact with other user models to obtain recommendations in a collaborative way (social-ability).

This paper is organized as follows: In section 2, we describe the Smart User Model. Section 3 is devoted to introducing the support vector machine kernel method and its relation to the *SUM*. In section 4 the multi-agent architecture

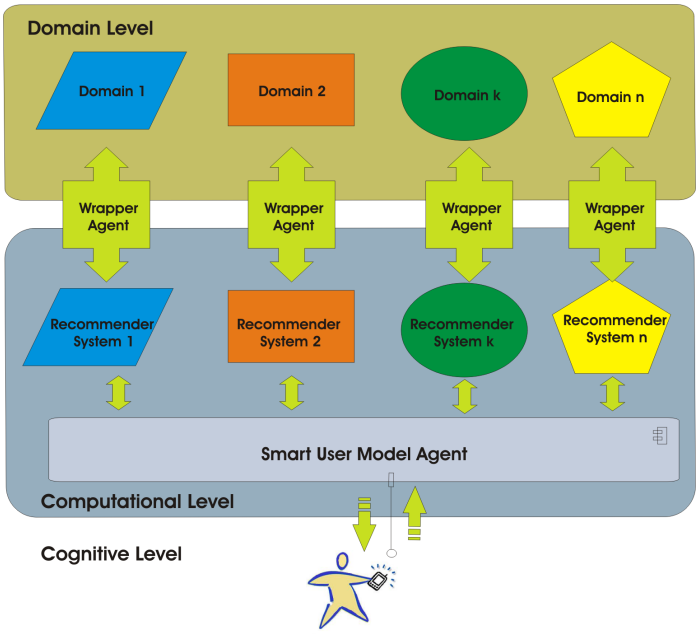

Fig. 1. The Smart User Model in Ambient Intelligence.

is explained. We continue in section 5 with an example, using our user model in multiple domains, and we end in section 5 with some conclusions and discussion.

2 Smart User Model

Broadly speaking, a Smart User Model should be able to deal with any type of objective, subjective or emotional user feature, whether explicit or implicit. For this purpose, the following *Smart User Model* has been defined in [1],

$$SUM = \left\{ \begin{array}{l} \left[\left(a_1^O, v_1^O\right), \ldots, \left(a_i^O, v_i^O\right), \ldots, \left(a_n^O, v_n^O\right)\right] \\ \left[\left(a_1^S, v_1^S\right), \ldots, \left(a_j^S, v_j^S\right), \ldots, \left(a_m^S, v_m^S\right)\right] \\ \left[\left(a_1^E, v_1^E\right), \ldots, \left(a_k^E, v_k^E\right), \ldots, \left(a_l^E, v_l^E\right)\right] \end{array} \middle| ; \right\} = \left\{U^O; U^S; U^E\right\}$$

where the collection of attributes-value pairs, $U^F = \left[\left(a_p^F, v_p^F\right)_p\right]$ represents n objective *(F=O)*, m subjective *(F=S)* and l emotional *(F=E)* user features. In this form, each user's behavior is obtained by a Smart User Model, SUM_r, defining his/her internal representation in the environment, to achieve ambient-aware personalization.

So that the *SUM* can be used more extensively in several application domains, we have initially defined the user model, *UMD*, for a given, existing application domain i as follows,

$$UMD_i = \left\{A_i^D, A_i^I, A_i^S\right\}$$

where A^D, A^I, and A^S are the sets of domain characteristics, interests and socio-demographic features, respectively, of the user required by the specific application.

Then, we establish a relationship between the general internal *SUM* and the user model for a specific application domain, UMD_i, by means of a weighted graph, where $UMD_i = G\left(SUM, UMD_i\right)$. This graph connects the user's internal features in the *SUM* with particular user features required at the application domain UMD_i. The emotional features of the *SUM*, U^E, modifies the weights used in the graph according to the user's emotional state (For more details see [1]). The methodology for managing objective and subjective user features is based on a combination of machine learning methods: inductive methods for generalization, in particular support vector machines, and deductive methods for specialization. This methodology can be used to both, learn user features from user information stored in Ambient Recommender Systems and deliver the user features to other Ambient Recommender Systems. Therefore, the user's *UMD* for each application is defined by shifting information to and from *UMD*'s of different existing domains according to the weighted graphs $G\left(SUM, UMD_i\right)$ defined by each application in which the user is involved. For more information on this methodology see [2], in which it is fully developed.

3 Support Vector Machines in User Modelling

The Support Vector Machine (SVM) is a type of learning machine for summarizing information and modelling from examples based on the statistical learning theory, which implements the structural risk minimization inductive principle in order to generalize data sets of limited size effectively [3,4]. There has been a great deal of research interest in these methods over the last years, because:

- They provide good data generalization.
- They are well suited to sparse data.
- They exhibit independence from the results from the input space dimension.

Although initially conceived for linearly separable two classes classification problems, new algorithms have already been derived to solve classification problems with non-separable data, regression, ordinal regression, and multi-class problems. Let $\mathcal{T} = \{(\mathbf{x_i}, y_i)\,;\, \mathbf{x_i} \in \mathcal{X}, y_i \in \{-1, +1\}\}$ be a training data set for a binary classification task, where classes are labeled as +1, -1. Let the decision function based on a hyperplane be $f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$. According to the statistical learning theory, good generalization is achieved by maximizing the margin between the separating hyperplane, $\mathbf{w} \cdot \mathbf{x} + b = 0$, and the closest data points for each class in the input space. This optimal hyperplane can be determined by solving a quadratic programming problem. The decision function can

thus be written as,

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{SV} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x}) + b\right)$$

In order to expand the method to non-linear decision functions, the original input space, $\mathcal{X}$, is projected to another higher dimension dot product space $\mathcal{F}$, called *feature space*, via a nonlinear map $\phi : \mathcal{X} \rightarrow \mathcal{F}$, with $dim(\mathcal{F}) >> dim(\mathcal{X})$. The optimal hyperplane is derived in this new space. Denoting the inner product in $\mathcal{F}$, (kernel) $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j}) = K(\mathbf{x_i}, \mathbf{x_j})$, the decision function is formulated in terms of this kernel.

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{SV} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b\right)$$

As an important consequence of the SVM procedure, just a few of the training patterns are significant for classification purposes, i.e. those with a non-zero weight α_i. These elements lie on the margin of the class and are known as support vectors. This means that the representation of the hypothesis generated by the SVM is given only by the points that, in the input space, are closest to the hyperplane and therefore they are the most difficult patterns to classify. The patterns that are not support vectors do not influence the position and direction of the decision function and are therefore not relevant to the hypothesis. Moreover, for this methodology the original space does not have to be an Euclidian space. By using appropriated kernels, any original space ('words', 'facial expressions', 'sensory data', 'clickstreams', 'preferences', 'attributes', 'physiological signals') can be transformed with minor restrictions into an useful feature space, $\mathcal{F}$.

Support Vector Machines are suitable for implementing efficient kernel methods to process very large and high-dimensional data sets produced by Ambient Recommender Systems in several domains. Several kinds of data sources for user modelling, such as weblogs, socio-demographic databases, transactional databases, preference and attribute databases, and sensory databases among others, can be pre-processed efficiently with SVM [5].

4 Multi-agent Smart User Model Architecture

To support our *SUM* approach in a web-based application, we propose a multi-agent architecture defined at two main abstraction levels [6]. At the highest level, there are two abstract agents exist (see Figure 2): the *Web Service Abstract Agent* (*WSAA*) and the *Ubiquitous Abstract Agent* (*UAA*). The *WSAA* provides autonomy capabilities in relation to automatically finding services for the user on the Internet [7]. It communicates with the applications in a specific domain. When applications are non-agent based, a *wrapper agent* operates like a middleware between the *WSAA* and the application. The *UAA* provides initialization, identification, interoperability, control, coordination and management of

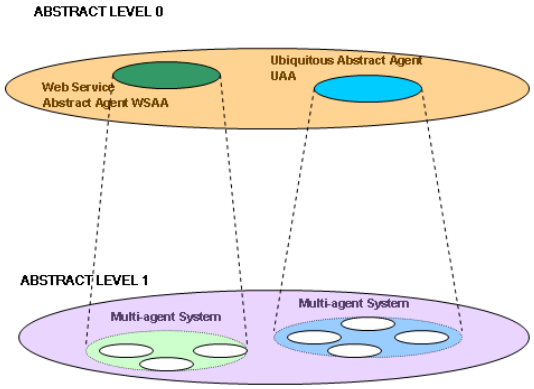

Fig. 2. Two-level Abstract Architecture for the Smart User Model.

the user preferences allowing a flexible and autonomous human-agent interaction. It is a generic and portable user model that works according to our *SUM* definition.

Coordination between the *WSAA* and the *UAA* is established mainly by two mechanisms: (i) The *WSAA* requests personalized information from the *UAA* to deal with the applications in the environment (ambient recommender systems); (ii) The *UAA* receives information from the *WSAA* regarding the success or failure of the application interaction. The *UAA* uses this relevant feedback to learn about the user's interests, so the corresponding *SUM* and the weighted graph $G\,(SUM, UM_i)$ of the application is updated.

Both the *WSAA* and the *UAA* have been designed to be implemented in a distributed platform. The *WSAA* can be stored in a server while the *UAA* can be stored in a mobile device. At the next abstract level, both abstract agents are implemented as multi-agent systems, as we explain in the next part of this section.

4.1 WSAA Architecture

The *WSAA* is made up of three types of agents, namely (see Figure 3):

The Accountant Agent, which keeps a register of user-interaction applications and domains. It also requests the *UAA Application Agents* to establish new applications (see subsection 4.2).

The Provider Agent, which uses contextual information and interacts with the *UAA Repository Agent* to obtain the user's pro-active behavior by finding new applications in non-registered domains in which the user could be interested.

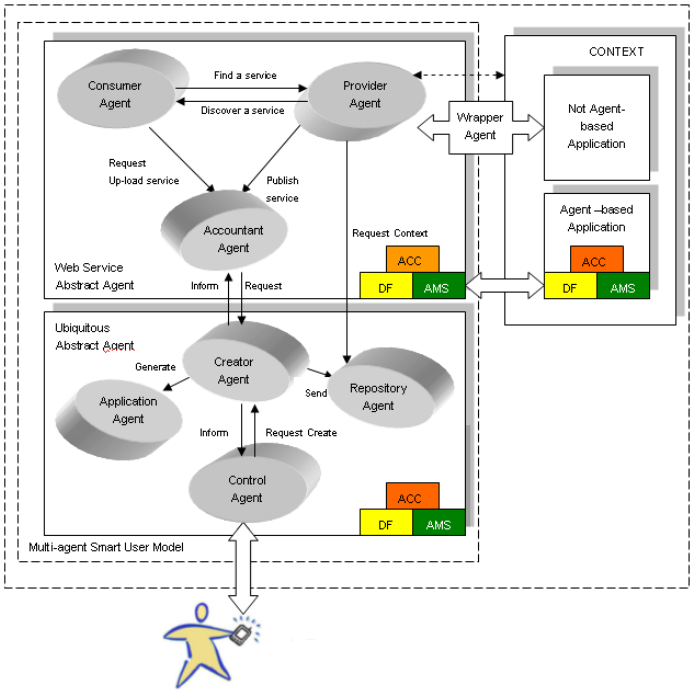

Fig. 3. Multi-agent System Architecture for the Smart User Model.

The Consumer Agent, which finds a user requested service by communicating with the *Provider Agent*, up-loading the service and creating an *Application Agent*.

4.2 UAA Architecture

The *UAA* has four types of agents, namely (see Figure 3): The *Control Agent, Creator Agent, Application Agents* and *Repository Agent* (see Figure 3).

The Control Agent's tasks are: (i) user login service; (ii) to dialogue with the user regarding his/her interaction with an application (suggested by the *WSAA* or requested by the user); (iii) to request to the *Creator Agent* to generate an *Application Agent* to manage the application confirmed by the user.

The Creator Agent is a temporal agent that manages the user information in previous-applications the first time that they are registered in the system. It has three goals: (i) to acquire the user profile by obtaining the entire information spread from the user's interaction with the ambient recommender systems, and to communicate it to the *Repository Agent*. The user's objective, subjective and emotional features are learned via the methodology

described in [1,2]; (ii) to generate *Application Agents* from past user interactions that will be in charge of the user's interaction and the application from now on; (iii) to register the previous applications in the multi-agent system by means of the *Control Agent*. To improve the *UAA's* performance, once the *Creator Agent* has carried out its functions it is removed.

Application Agents are dynamically created when there is interaction with an application, the number of *Application Agents* varies from user to user. They provide all the information about the user that an application requires (reactivity), by acquiring and saving the relationship graph between the *SUM* and the user model of the application UM_i. *Application Agents*have social abilities and are connected with other multi-agent *SUM*s, establishing a social network [8] of Smart User Models. When more than one agent has interacted with a certain application and therefore more than one possible graph in the social network can be found, the *Application Agent* composes the graphs by means, for instance, of trust measures [9]. Otherwise, if no agent in the social network has interacted with the application, the user needs to intervene for the graph to be established.

The Repository Agent provides database storage procedures to save the user knowledge represented in the *SUM*. Individual user information is kept in a non-redundant, complete and consistent way in order to share it when and where it is necessary.

5 A Case Study

In this section, we illustrate briefly with an example the functional operation of the architecture proposed. Let Juan Valdez be a user that has interacted in the past with the IRES recommender system in the 'restaurant domain' [10]. Now, Juan Valdez sets up his *SUM* and so the *UAA* starts.

In a first step, Juan Valdez initializes his Smart User Model through the *UAA* by registering his ID and his password through the *Control Agent*. Immediately the *Control Agent* requests to the *Creator Agent* for the registration. This last agent gathers the current information about the user in the restaurant domain and sends it to the *Repository Agent*. Then the *Control Agent* creates an *Application Agent* for the restaurant recommender system and registers the restaurant application made to the *Control Agent*

In the *UAA*, the *Control Agent*, prompts the user with information about cinema recommender systems and Juan Valdez selects one application. Then the *Control Agent* creates an *Application Agent* to deal with the new application. The *Application Agent* looks in the social network for a user that has dealt with the new application. It is the case that Paula Allende has already interacted with the new application. So Paula's and Juan's corresponding *Application Agents* dialog. Juan's *Application Agent* obtains the graph $G\,(SUM, UMD_i)$ corresponding to the relationship between Paula's *SUM* and her *UMD* in the 'cinema domain'. The weighted graph G is adapted to the Juan Valdez's *SUM* and his *Application Agent* is ready to deal with the recommendation process.

After a while, the *WSAA Provider Agent* gathers information about new recommender systems in the 'restaurant domain'. In this case, Juan Valdez has been using a recommender system for the city of Girona, and the *Provider Agent* has discovered a recommender system for the city of Barcelona. Since the user is traveling to this last city (contextual information), the *Provider Agent* believes that this information could be interesting for the user. Hence, the *WSAA* requests to the *UAA* to generate a new application in this new recommender system.

6 Conclusions

In the past, user modelling was focusing on developing domain-dependent software architectures for user models [11,12,13,14]. However, in recent years information technology has moved from single and centralized uses to distributed multipurpose systems, which are now increasingly embedded in a fully interconnected world [15,16]. Smart User Models must relate with the context in which they are used as well as being a cornerstone in cross-domain recommendation processes. For instance, if a user model helps a user to choose an appropriate restaurant, this same user model could help him/her to choose an appropriate cinema, although the domains are different.

Therefore, we are contributing to the next generation of ambient intelligent systems through Smart User Models which include, among other variables, the user's emotional features

In this paper, we have presented a multi-agent architecture for Smart User Models that aims to shift traditional user models towards this a new vision of distributed models. Our multi-agent Smart User Model can operate across multiple domains implemented by dynamically composing agent services. The architecture is defined at two abstract levels, one concerning services and another one dealing with user features that constitutes a distributed platform.

We are currently testing our hypothesis by using kernel-based methods [17,18] in order to construct automatic mapping of user features into the high-dimensional feature space of several domains. In particular, we are in the process of implementing a hybrid One-Class and RBF SVM as a learning component of the Multi-agent Smart User Model to classify user preferences in high-dimensional and heterogeneous databases from both a web service-based *intelligent learning directory*[1] and a agent-based restaurant recommender system[2].

The Multi-agent Smart User Model could contribute to quality of life with a re-usable and non-intrusive intelligent adaptive system able to understand the user's habits.

7 Acknowledgments

This research has been supported by the Spanish Science and Education Ministry project TIN2004-06354-C02-02.

[1] www.emagister.com
[2] http://arlab.udg.es/

References

1. González, G., López, B., de la Rosa, J.: Managing Emotions in Smart User Models for Recommender Systems. In: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'04)., Porto, Portugal (2004) 187–194
2. González, G., López, B., de la Rosa, J.: Smart User Models for Tourism: An Holistic Approach for Personalized Tourism Services. ITT Information Technology & Tourism Journal **6** (2004) 273–286
3. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
4. Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University press 2000 (2000)
5. Zhang, T., Iyengar, V.S.: Recommender Systems Using Linear Classifiers. Journal of Machine Learning Research **2** (2002) 313–334
6. Giret, A., Botti., V.: Towards an Abstract Recursive Agent. Integrated Computer-Aided Engineering **11** (2004)
7. Dale, J., Lyell, M.: Towards an Abstract Service Architecture for Multi-Agent Systems. In: Challenges in Open Agent Systems '03 Workshop., Melbourne, Australia. (2003)
8. Palau, J., et. al.: Collaboration Analysis in Recommender Systems Using Social Networks. In Klusch, M., Ossowski, S., Kashyap, V., Unland, R., eds.: Cooperative Information Agents VIII: 8th International Workshop, CIA 2004. Volume 3191 of Lectures Notes in Computer Science., Erfurt, Germany, Springer-Verlag Heidelberg (2004) 137–151
9. Montaner, M., López, B., de la Rosa, J.L.: Opinion- based Filtering Through Trust. In Matthias Klusch, S.O., Shehory, O., eds.: Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA'02). Lecture Notes in AI, Madrid (Spain), Springer-Verlag Berlin Heidelberg (2002) 164–178
10. Montaner, M., et.al.: IRES: On the Integration of Restaurant Services. AgentCities Agent Technology Competition: Special Prize, Barcelona (Spain). (2003. Available at: http://arlab.udg.es/GenialChef.pdf)
11. Fischer, G.: User Modeling in Human Computer Interaction. User Modelling and User-Adapted Interaction (UMUAI). **11** (2001) 65–86
12. Kobsa, A.: Generic User Modelling Systems. User Modelling and User-Adapted Interaction (UMUAI). **11** (2001) 49–63
13. Brusilovsky, P.: Adaptive Hypermedia. User Modelling and User-Adapted Interaction (UMUAI). **11** (2001) 87–110
14. Fink, J.: User Modelling Servers: Requirements, Design, and Evaluation. PhD thesis, Dept. of Mathematics and Computer Science, University of Essen, Germany (2003)
15. Muñoz, M.A., et.al: Context-Aware Mobile Communication in Hospitals. IEEE Computer **36** (2003) 38 – 46
16. Sadeh, N.M., et. al.: Creating an Open Agent Environment for Context-Aware M-Commerce. In et al., B., ed.: Agentcities: Challenges in Open Agent Environments. Lecture Notes in Artificial Intelligence, Agentcities, Springer Verlag (2003) 152–158
17. Angulo, C., Català, A.: Ordinal Regression with K-SVCR Machines. In Mira, Prieto, eds.: 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001. Granada, Spain. Volume 2084 of Lecture Notes in Computer Science. (2001)
18. Angulo, C., Parra, X., Català, A.: K-SVCR. A Multi-class Support Vector Machine. Neurocomputing. Special issue: Support Vector Machines **55** (2003) 57–77