AVATAR: Modeling Users by Dynamic Ontologies in a TV Recommender System based on Semantic Reasoning

Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrер, Martín López-Nores, Belén Barragáns-Martínez
Department of Telematic Engineering
University of Vigo, 36310, Spain
{yolanda, jose, agil, mramos, mlnores, belen}@det.uvigo.es

Abstract
In this paper a user modeling technique in the context of the AVATAR system is presented. AVATAR is a personalized TV contents recommender based on a process of semantic reasoning on the viewer preferences and the TV programs. This semantic reasoning requires a formal representation of the user preferences, and for that reason our approach uses typical Semantic Web technologies. So, the proposed user modeling technique is based on the representation of each profile like an ontology built incrementally, as the system receives information about the user viewing behaviour. In addition, we also describe the information used by AVATAR in order to measure the level of interest of the viewers, and to quantify the success or failure in previous recommendations. Our proposal is useful to compare different user profiles, leading to a possible reuse of suggestions previously recommended for viewers with similar preferences. In this regard, the use of ontologies in the proposed technique allows to apply the knowledge of the TV domain, beyond the simple syntactic comparison of former approaches.

Key Words
Modeling users, recommender systems, ontologies.

1. Introduction
Nowadays, the TV viewers can access a large amount of contents, due to the migration from the analogue to digital TV. The digital TV does not only provide the users with more channels, but also allows them to take an active role by using interactive applications that are broadcast together with the conventional TV programs. In this context, users tend to get disoriented and not manage to find the most appealing contents. To effectively address this problem, the TV viewers need tools -which we will refer to as TV recommender systems (Schafer J.B. et al. 1999)- which offer personalized TV programs (Lilianna A. et al. 2004) to them according to their preferences, avoiding tedious searching processes.

The TV recommenders have been interesting for the research community, leading to the appearance of different approaches. Some of the most used techniques are Bayesian methods, decision trees, collaborative filtering and content based strategies, among others. All these techniques share a common drawback related to the reasoning capabilities. No existing recommender uses the knowledge of the TV domain for reasoning on the user preferences and generic descriptions of the TV contents. On this semantic reasoning is focused the approach used in AVATAR (AdVanced Telematic search of Audiovisual contents by semantic Reasoning). In order to represent the knowledge of the TV domain, we propose to extend the use of the Semantic Web technologies (Daconta M. et al. 2003) to our application context. More specifically, we have implemented an ontology, according to the OWL (Web Ontology Language) language, which stores a hierarchy of classes and properties together with specific instances, describing different categories of TV programs and their main characteristics. Regarding to the descriptions of TV contents, the TV-Anytime (www.tv-anytime.org) specification has been used in our approach, because of the standardization is an important issue to promote an extended use of the proposed system.

One of the main research lines in the field of TV personalization is the User Modeling. This technique allows the acquisition, representation and utilization of information about users, including personal data, their preferences about TV programs and their viewing behavior. In this paper we focus on the method used by AVATAR for modelling the user interests, by describing both the information stored in the user profiles and their updating process. In this regard, a set of indexes whose aim is to measure the level of interest of the users, and to quantify the success or failure of the recommendations offered in the past, are defined. Given that we have implemented an ontology about the TV domain, our proposal uses the knowledge represented in it for modelling tasks. This approach allows to find semantic similarities between the programs defined in different user profiles, by using
the knowledge of the OWL ontology, beyond a simple syntactic comparison. So, it is possible to recommend a user programs that have been of interest for viewers with similar preferences.

This paper is organized as follows: in Sect. 2 some existing approaches in the fields of personalization tools and user modeling are described. In Sect. 3 we focus on the user modeling technique proposed, based on the use of ontologies built incrementally, and we also describe the organization of the user profiles, by presenting a specific syntax for that purpose. Finally, in Sect. 4 the main conclusions are shown and the future work is discussed.

2. Related Work

As we mentioned in Sect. 1, in the state-of-art there exist two widely used recommendation techniques: (i) content based methods (Ricci F. et al 2002) and (ii) collaborative Filtering (Rashid A.L. et al. 2002, Resnick P. et al. 1994). The former method is based on recommending the users TV programs that are similar to ones watched in the past, whereas the collaborative filtering focuses on suggesting the viewers contents that have been interesting for other users with similar preferences. Although these techniques are appropriate for a recommendation process, they also have some disadvantages. On one hand, during the initial phase of system working, there are no users registered. So, the collaborative filtering approach is hampered because of the no availability of enough user profiles. On the other hand, the content based methods could be scarcely useful because the suggested programs are too similar to contents the user already knows. This way, there are some proposals that focus on a hybrid approach (Burke R. 2002, Cotter P. et al. 2000), where both techniques are combined with the aim of meeting their different advantages.

To secure high quality recommendations the technique chosen for modeling the user interests plays a key role. In this regard, there exist many different approaches in the state-of-art. Taking into account the relevance of the inference and reasoning processes in AVATAR, we focus on user modeling techniques based on knowledge (Kobsa A. et al. 1991). In this approach, it is necessary for the system to acquire knowledge about the user preferences (e.g. by questionnaires), and then they must be modeled in a knowledge base. One of the most relevant knowledge based strategies is the “stereotyping”, proposed by Elaine Rich in 1979 (Rich E. 1979). This technique needs first to identify groups of users with similar common characteristics, which have to be relevant in the specific application domain of the system. Then, these characteristics called stereotypes, must be formalized in an appropriate representation system. For that goal, Rich proposes a hierarchical organization of stereotypes, to favor the inference of new information from the known data. Our proposal also uses a hierarchical information organization for modeling the user preferences, by using the knowledge stored in our OWL ontology about the TV domain, but differs from the stereotyping approach because we model individual user preferences, instead of user groups. So, we propose to apply the use of ontologies to represent the viewer preferences about TV programs. This way, the ontology that represents a user profile - which we will refer to as ontology-profile - is built incrementally, as the system receives additional information about the viewing behavior of the user (i.e. the actions carried out by the viewers and the programs selected and rejected when the recommendations are shown to them).

The ontologies have been used in collaborative approaches out of the TV domain. In (Middleton S. 2003) the advantages of these techniques are described. This work defines a recommender of research papers, according to the interests of the users registered in the system. Our approach differs from it because we do not only propose the use of ontologies to model the viewer interests, but also to compare the preferences of different users from a semantic point of view, by using the knowledge represented in our OWL ontology. The dynamic user modeling has been tackled in previous works (Stock O. et al. 1996). This paper presents a recommender system, called AlFresco, appealing to users interested in Italian paints and frescoes. The user can interact with the system by browsing links and typing sentences. This recommender uses a dynamic user modeling that is built incrementally, as AlFresco receives information about user preferences. This model consists of two separate modules: one modeling the user's knowledge, and one representing the potential interests of the user. The latter one is an activation/inhibition network where each node represents an area of interest, such as a specific painting style or a period of time. So, the activation of a node -caused for example because the user has asked for something related to a specific area- causes (i) the activation of nodes that represent areas related to the former one, and connected to it by activation links, and (ii) the inhibition of the nodes representing incompatible areas connected by inhibitory links. If the activation/inhibition networks are compared with ontologies, the network nodes would be the ontology classes, and the links that connect different nodes would be the properties between these classes. Our proposal differs from (Stock O.
et al. 1996) because our system does not only consider the properties defined explicitly (links between the network nodes), but also other properties related to the former ones implicitly, from a semantic point of view. This approach will be useful to reuse recommendations made previously for other users with similar preferences.

3. User Modeling based on Ontologies

The architecture of the AVATAR system is described in (Blanco-Fernández Y. et al. 2004c), where different knowledge inference strategies are combined in a multi-agent environment. So, we can identify three agents: (i) Bayesian agents, (ii) agents based on profiles matching and (iii) agents based on semantic reasoning about descriptions of TV programs (TV-Anytime metadata) and user preferences. This reasoning requires a formal representation of the viewer interests to favor inferential processes from the known information, leading to personalized recommendations.

The information stored in user profiles must be both personal data of the viewers and their preferences about TV contents. For that reason, these profiles consist of two separate parts:

(i) **Static part:** It stores personal data about users, such as their name, age, gender, nationality, etc. It is a reduced information to avoid that the viewer has to fill tedious forms. This information is provided for the registering phase and can be modified when the user needs to update some data.

(ii) **Dynamic part:** It is called PSC (Personalized Semantic Context) because it defines the personalized information about the programs that the user likes and dislikes. This part of the profile must be flexible enough to be updated when the interest of the viewers changes.

For that reason, our approach uses the implemented OWL TV ontology, to build dynamically an ontology-profile for each user. So, if a user likes a program classified into a specific class of the TV ontology, AVATAR stores this class in his/her ontology-profile, together with all the superclasses related to it, according to the TV ontology. For example, if a user likes the action movies, AVATAR must include in his/her ontology-profile, the following classes hierarchy: Cinema_Programs \(\rightarrow\) Movies \(\rightarrow\) Action_Movies.

This way, the static part of user profiles is a set of properties created to store personal data about the viewers, whereas to build the dynamic part we reuse the hierarchy of classes and properties defined in the existing OWL ontology. In our approach, each ontology-profile has been implemented by the Protégé-2000 tool, according to the precepts of the OWL language. The ontology allows to represent hierarchically the knowledge about the user preferences, instead of using plain lists which fail at expressing the existing relations in a specific application domain. For example, assume a user enjoys watching programs about Formula 1 races (instance of the class F1_Broadcasting_Programs) and informative programs about motorbikes races (instances of Motorbike_Broadcasting_Programs). In this scenario, it is interesting to store in his profile these contents, together with the common class for both: Sport_Programs in this example. This user will be more similar to a viewer that likes other sport programs -although these programs are not exactly motorbike races or Formula 1- than another one that enjoys, for example, with movies. So, it is possible to discover semantic relations between preferences of different users. The ontology knowledge allows the system to reuse programs that are of interest for other users with similar preferences, beyond a simple syntactic comparison between the content of their profiles.

In the following section the organization of each ontology-profile will be described. For that, we propose a syntax that allows to add new information to the existing data stored in the profile. This syntax is a standard way for referencing the hierarchy of classes and properties defined in the ontology.

3.1 Organization and Syntax of User Profiles

The user preferences in AVATAR can be provided by the viewer explicitly, or they can be inferred by the system. So, the information stored in the user profile can be more general or more specific. For example, a user can say that he enjoys watching action movies (general information), or he can have watched a specific movie, from which AVATAR extracts more complete information about his preferences (e.g. data about actors that take part in this TV content).

This way, regarding the information contained in the user profiles, two possible situations can be differentiated:

- First, AVATAR could only know a class related to the the TV programs that user likes or dislikes. As we mentioned at the beginning of this section, this class must be defined in the user ontology-
profile, together with the complete hierarchy of classes related to it, according to the knowledge defined in the TV ontology. To represent this hierarchy -classes together with their subclasses- in the syntax described here, the succession of these classes is stored joining them using the $\sqsubseteq$ symbol (Let A and B be two classes defined in an ontology. If A is superclass of B, then A subsumes B and this is represented by B $\sqsubseteq$ A because of the subsumption principle). Given that several classes can have the same superclass in the hierarchy defined in the TV ontology, our syntax proposes to use the symbol $\bigcap$ to connect many subclass successions to a specific class. In our notation, this class is C in $\bigcap$.

For example, if a user profile stores information about action movies, and now it is necessary to add data related to comedy movies (both classes are subsumed by the Movies class), this profile must be updated as shown below:

Movies $\sqsubseteq$ Comedy_Movies $\bigcap$ Movies $\bigcap$ Action_Movies

When the only information about the user preferences consists of general data provided by himself, the user profile only contains classes (at least one). In this case, the information to make personalized recommendations is very reduced, hampering the inference and reasoning processes.

- Secondly, it is possible that the system does not only know one class (to which a specific TV program belongs), but also a set of properties that define characteristics related to instances of this class. Note that OWL defines two types of properties: (i) Object properties that establish relations between instances of two classes, and (ii) Datatype properties that relate an instance to a specific data type (string, integer numbers, etc). Besides, the domain of a property limits the individuals to who this property can be applied, whereas the range identifies the individuals that the property can have as value. In our notation, a class is joined to the set of properties referred to it using dots (.)

Besides, if the range of an Object property is the same class that the domain of other property (Object or Datatype), is possible to go on making joinings. This structure will be called sequence of properties. The last property in a sequence will always be a Datatype property that defines a specific value of a data type. In our syntax this value is represented in square brackets, after the Datatype property (e.g. hasTitle [Wheel of the Fortune]).

On the other hand, several sequences of properties can be joined to a class by using the $\bigcap_p$ symbol, if this class is the domain of the first property of all these sequences. Note that the number of sequences of properties joined to a class depends on the information that AVATAR knows about user preferences. The more data the system knows, the more properties can be defined in the profile, and so is possible to facilitate the progress of inferential processes.

The organization of user profiles that we have just described appears formalized by the Extended Backus-Naur Form in Fig. 1. As shown here, an ontology-profile always contains at least one class and it can contain 0 or more properties. Note that the set of classes defined in the profile must obey the subsumption principle, as we explained before. Regarding the sequence of properties, remember the constraint that the range of a property be equal to the domain of the next one in the sequence.

To conclude this section, we show an example of the dynamic part of a user profile in AVATAR. Assume that it contains information about (i) a soccer match where the Liverpool team participates, (ii) the comedy movie entitled “The Mask” where Jim Carrey works as starring actor and (iii) an informative program about sport, presented by Carl Lewis. So, after querying the knowledge hierarchy represented in the OWL TV ontology this viewer profile is shown in Table 1 (note that here the TV program presented by Carl Lewis and the soccer match are instances of classes subsumed by the Sport_Programs class).

<table>
<thead>
<tr>
<th>Table 1: Excerpt from a User Profile in AVATAR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sport_Programs</strong> $\sqsubseteq$ Sport_Broadcasting Soccer_Broadcasting (hasTeam [Liverpool])</td>
</tr>
<tr>
<td>$\bigcap_{\text{Sport_Programs}}$ Informative_Sport_Programs (hasPresenter. Name [C. Lewis])</td>
</tr>
<tr>
<td>Movies $\sqsubseteq$ Comedy_Movies (hasTitle [The Mask]) $\bigcap_p$ (hasStarringActor. Name [J. Carrey])</td>
</tr>
</tbody>
</table>

\[ \text{Table 1: Excerpt from a User Profile in AVATAR} \]

\[ \text{profile, together with the complete hierarchy of classes related to it, according to the knowledge defined in the TV ontology. To represent this hierarchy -classes together with their subclasses- in the syntax described here, the succession of these classes is stored joining them using the $\sqsubseteq$ symbol (Let A and B be two classes defined in an ontology. If A is superclass of B, then A subsumes B and this is represented by B $\sqsubseteq$ A because of the subsumption principle). Given that several classes can have the same superclass in the hierarchy defined in the TV ontology, our syntax proposes to use the symbol $\bigcap$ to connect many subclass successions to a specific class. In our notation, this class is C in $\bigcap$.} \]

\[ \text{For example, if a user profile stores information about action movies, and now it is necessary to add data related to comedy movies (both classes are subsumed by the Movies class), this profile must be updated as shown below:} \]

\[ \text{Movies $\sqsubseteq$ Comedy_Movies $\bigcap$ Movies $\bigcap$ Action_Movies} \]

\[ \text{When the only information about the user preferences consists of general data provided by himself, the user profile only contains classes (at least one). In this case, the information to make personalized recommendations is very reduced, hampering the inference and reasoning processes.} \]

\[ \text{- Secondly, it is possible that the system does not only know one class (to which a specific TV program belongs), but also a set of properties that define characteristics related to instances of this class. Note that OWL defines two types of properties: (i) Object properties that establish relations between instances of two classes, and (ii) Datatype properties that relate an instance to a specific data type (string, integer numbers, etc). Besides, the domain of a property limits the individuals to who this property can be applied, whereas the range identifies the individuals that the property can have as value. In our notation, a class is joined to the set of properties referred to it using dots (.).} \]

\[ \text{Besides, if the range of an Object property is the same class that the domain of other property (Object or Datatype), is possible to go on making joinings. This structure will be called sequence of properties. The last property in a sequence will always be a Datatype property that defines a specific value of a data type. In our syntax this value is represented in square brackets, after the Datatype property (e.g. hasTitle [Wheel of the Fortune]).} \]

\[ \text{On the other hand, several sequences of properties can be joined to a class by using the $\bigcap_p$ symbol, if this class is the domain of the first property of all these sequences. Note that the number of sequences of properties joined to a class depends on the information that AVATAR knows about user preferences. The more data the system knows, the more properties can be defined in the profile, and so is possible to facilitate the progress of inferential processes.} \]

\[ \text{The organization of user profiles that we have just described appears formalized by the Extended Backus-Naur Form in Fig. 1. As shown here, an ontology-profile always contains at least one class and it can contain 0 or more properties. Note that the set of classes defined in the profile must obey the subsumption principle, as we explained before. Regarding the sequence of properties, remember the constraint that the range of a property be equal to the domain of the next one in the sequence.} \]

\[ \text{To conclude this section, we show an example of the dynamic part of a user profile in AVATAR. Assume that it contains information about (i) a soccer match where the Liverpool team participates, (ii) the comedy movie entitled “The Mask” where Jim Carrey works as starring actor and (iii) an informative program about sport, presented by Carl Lewis. So, after querying the knowledge hierarchy represented in the OWL TV ontology this viewer profile is shown in Table 1 (note that here the TV program presented by Carl Lewis and the soccer match are instances of classes subsumed by the Sport_Programs class).} \]
AVATAR does not only need to know the user preferences, but also the level of interest related to them. So, the next section focuses on the metrics used by AVATAR to update the user profiles and to make personalized recommendations.

3.2 Updating the User Profiles in AVATAR

Each recommendation agent described in the AVATAR architecture (Blanco-Fernández Y. et al. 2004c) is able to get personalized TV programs for the users, ordered according to their relevance and to the information stored in his profile. Notwithstanding, once these contents are shown to them, the system must infer data from the actions that the viewers carry out (e.g. the programs that the user has selected or rejected). So, once the recommender has presented the first suggestion to the viewer, the system must update a list of indexes whose values will determine the order of future recommendation.

In what follows, we suppose that AVATAR has shown a first recommendation to the user, containing different TV programs (instances of the classes defined in the TV ontology). From these suggestions, the system must extract the classes related to the instances contained in them. Each one of these classes must be added to the user ontology-profile, together with the appropriate classes hierarchy, and the indexes related to them must be computed and stored in the profile. This way, the value of an index for the \( C_i \) class, is computed by adding the weight related to each instance of this class contained in the recommendation suggested by the AVATAR system. Note that the weight of each instance is updated by adding to its former weight \( W_n(Inst_k) \), the new value \( W_n(Inst_k) \), that is calculated by the expressions provided here.

Once the index referred to each class in the user ontology-profile has been computed, the values are normalized to be in \([-1,1]\). So, the positive values identify classes that are of interest for the user, whereas the negative ones are related to classes that the viewer does not like.

3.2.1 Degree Of Interest (DOI)

This factor measures the degree of interest of the user referred to each class contained in his ontology-profile. The mathematical expression to compute the DOI index is the following:

\[
DOI(Inst_k) = \sum_{k=1}^{N} W_n(Inst_k) \tag{1}
\]

In Eq. 1, N is the number of instances of the \( C_i \) class contained in the recommendation shown to the end user, \( Inst_k \) is the k-th instance of the \( C_i \) class, and \( W_n(Inst_k) \) is the weight of this instance. This way, the system is able to quantify the interest of the user in, for example, an actor (instance of the Actors class) or in a specific action movie (instance of the Action_Movies class in the TV ontology).

AVATAR provides the user with an interface where he can access the different classes related to his preferences (e.g., Action_Movies or Actors), the instances of these classes (the movie entitled Rocky II or the actor Clint Eastwood), and the properties referred to them. In addition, the indexes of the instances contained in the user ontology-profile can be set by the user manually, or updated by AVATAR in an automatic way. This approach allows the recommender to correct possible errors in the offered suggestions, to attain a fast convergence between the real user preferences and the inferred ones by AVATAR. To compute the field \( W_n(Inst_k) \) shown in Eq. 1 three different indexes must be considered:

- **Index Of Feedback (IOF):** This factor identifies the feedback information provided by the user once the recommendation is offered to him. For each program contained in the recommendation, the value of this index is 1 if the user selects it to be watched, and -1 if this content is rejected.
- **Index Of Viewing (IOV):** This index is the ratio between the time a user watches a TV program suggested by the system, and its total duration. This factor is computed only if a program has been selected by the user; otherwise, its value is 1.

Let us imagine the following scenario to validate the usefulness of this index. AVATAR shows a list of personalized programs to the user. He selects one of them because he thinks it is interesting. However, when few minutes have passed, the user suspends the viewing of this content, because he discovers that it is not of interest for him. AVATAR could have included the category of this program into the positive preferences of this user, because of his selection, but this action would be wrong. Using the IOV index, whose expression appears in Eq. 2, this error could be fixed, since the weight assigned to the aforementioned program would be reduced.

\[
IOV(Inst_k) = \frac{ViewingTime(Inst_k)}{Duration(Inst_k)}
\]  

(2)

- **Antiquity of Viewing (AOV):** This index represents the time that has passed (number of days) since the user selected a recommended TV program, until he finally watched it. As we mentioned for the IOV index, this factor is only computed if a program has been selected by the user; otherwise, its value is 1. This factor is based on the following: the user will watch first those TV contents that are the most interesting for him, out of all the stored programs that he has chosen for viewing.

Taking into account the above indexes, DOI will be computed by Eq. 3 and Eq. 1 presented before.

\[
W_n(Inst_k) = W_o(Inst_k) + \frac{IOV(Inst_k) \ast IOF(Inst_k)}{AOV(Inst_k)}
\]

(3)

Assume the following scenario in AVATAR regarding the DOI index. We suppose that a specific class defined in the user ontology-profile has a very high degree of interest (i.e. a value close to 1), given that the user has selected many instances of this class recommended by the system. However, when AVATAR suggests this class again, the user rejects these instances. It is not hard to see that the DOI index referred to this class must be reduce greatly. On the contrary, once the reduction has been made, the value of this index could be greater than the remaining ones defined in the user profile. So, AVATAR would continue recommending this kind of programs because of the high DOI index associated. To prevent this error, our prototype uses the criterion presented next, by which we have obtained good results in the recommendation process: if the DOI index referred to a class is greater than 0.75 and one instance of this class is rejected by the user, the DOI index will take the value of the average of the positive DOI indexes, whose value is less than the analyzed one, contained in the user ontology-profile. If there does not exist any positive DOI index less than the analyzed one, the updating process is made by the mathematical expressions presented above. This is an effective rule because in this case, the user ontology-profile contains classes with very high DOI indexes (greater than 0.75 and greater than the analyzed index), and so the system will include their instances in the final suggestion, instead of the instances of the class the user has just rejected.

### 3.2.2 Confidence Index

This index quantifies the success or failure obtained by the AVATAR system in the recommendations offered in the past. The mathematical expression to calculate the contribution of each instance is shown in Eq. 4.

\[
W_n(Inst_k) = W_o(Inst_k) + \frac{IOF(Inst_k)}{Order(k)}
\]

(4)

In Eq. 4, \(Order_k\) represents the order of the k-th program in the recommendation presented to the user, and \(IOF(Inst_k)\) is the index of feedback described before. If the user chooses TV contents situated in the first positions of the recommendation (order = 1,2,...), the confidence index will be increased greatly, whereas if he selects programs in the last positions, the value of this index will be less than the previous one. If the programs ranked in the first positions are rejected, the confidence index for these instances, and for their classes, will be decreased strongly (here \(IOF(Inst_k)\) would be -1). The decrease will be lower if the rejected programs are ranked in the last positions.

### 3.2.3 Relevance index

This index is used to order the programs contained in the final recommendation offered to the users. Those classes contained in the user ontology-profile with high values of the relevance index, will provide the suggestion with more instances, and they will be ranked in the first positions. Examples of recommendations in AVATAR can be posed in
recommendation based on Bayesian classifiers- and in (Blanco-Fernández Y. et al. 2004b)-
personalization based on semantic reasoning.

The relevance index is computed by combining the DOI and confidence indexes. This factor is related to a class, and is calculated by adding the contribution of each one of its instances contained in the shown recommendation. This contribution is computed multiplying (in absolute value) the DOI index and the confidence one referred to the instance, as shown in Eq. 5, where $S_k$ is 1 if the DOI index for $Inst_k$ is positive and -1 otherwise.

$$W_r(Inst_k) = W_r(Inst_k) + S_k \cdot |DOI(Inst_k) \cdot Conf(Inst_k)|$$

Finally we show a collection of scenarios where the evolution of the relevance index can be seen, against the actions carried out by a user (selected and rejected programs). In the horizontal axis of Fig. 2 three different selections (S1, S2 and S3) of a user are shown. In the vertical one the relevance index related to a generic class C is represented. Assume the following scenarios described in the figure:

- **Scenario 1**: In this case, we suppose that the user selects all the programs related to the class C in his first choice (S1). In the next choices, this behavior model is repeated again, so the relevance referred to C is maximum and with constant value 1.

- **Scenario 2**: Now the user chooses all the instances of the class C contained in the shown recommendation, however he does not watch the whole content. So, the IOV index reduces the value of DOI, and in consequence, the value of the relevance assigned to this class is less than 1. In Fig. 2 is seen that the relevance is increased after each user selection, from which we can deduce that the viewing time is greater each time. This way, the system continues providing the viewer with instances of C in the offered suggestion, because this class seems to be of interest for him.

- **Scenario 3**: Firstly, the user selects some instances of the class C, and he rejects the remaining ones. So, a medium level of the relevance index is obtained. In the next recommendation, AVATAR advocates instances of this class again. Now, the user rejects one of the suggested programs, causing a decrease in the relevance index. However, this class continues belonging to the user favorite preferences. For that reason, AVATAR recommends programs classified into the class C and in S3, the user rejects them one more time. So, the relevance is decreased, reaching negative levels. This allows the system to identify a category of programs that the viewer does not like.

![Figure 2: Relevance index against the user choices](image)

To conclude this section related to the user profiles updating, note that the indexes assigned to a class described here, must be spread through the hierarchy defined in the user *ontology-profile*. We are working on this issue at the moment.

4. Conclusions and Further Work

In this paper a user modeling technique has been presented in the context of the AVATAR system, a personalized TV contents recommender. Taking into account that AVATAR is based on an approach of semantic reasoning on the descriptions of TV programs and the user preferences, is not hard to see that this system needs some knowledge representation method to conceptualize the TV domain. In this regard, our approach proposes to extend the use of the Semantic Web technologies to the TV context. So, we have implemented an ontology -according to the OWL language- that defines a set of classes, instances and properties to define the main concepts and characteristics of our application domain. This knowledge base is necessary to favor inferential processes, whose goal is to discover semantic relations among the instances contained in the ontology.

Our proposal for user modeling focuses on the reuse of the knowledge base available in AVATAR.
So, we propose to represent each user profile like an OWL ontology built incrementally, as the system receives additional data about the user viewing behavior. This behavior consists of several elements: (i) the programs selected or rejected by the viewer, from which AVATAR infers information and (ii) the user preferences declared explicitly, among others.

Besides the reuse of the TV ontology knowledge, our proposal defines several indexes to maintain user profiles permanently updated. These indexes take into account relevant issues in a personalization context, such as (i) the user interest in the classes and specific instances defined in his ontology-profile -the degree of interest index-, (ii) the quantification of the success or failure of the system in previous recommendations -the confidence index-, (iii) the ratio between the time that a user watches a TV program, and its total duration -the IOV index-, among others. Regarding these indexes, our proposal also considers the possibility that a user can modify his ontology-profile manually, by accessing each class and each instance. This approach favors a fast convergence between the real preferences of the viewers and the interests inferred by the system.

Our user modeling proposal is especially useful to be applied in one of the most used methods in personalization systems, the collaborative filtering. This approach recommends a user those TV programs that have been of interest for viewers with preferences similar to him/her. The use of ontologies in the field of the user modeling allows to get an enhanced collaborative filtering, by using the knowledge of the TV ontology.

Our future work is focused on this issue. We are now designing an algorithm of profiles matching, based on a semantic comparison between a given profile, and a set of a stereotypical profiles. Our approach discovers semantic relations between the instances defined in the user ontology-profile. For that purpose, our user modeling proposal is very useful, because it allows to compare different profiles, from a semantic point of view, beyond a simple syntactic comparison.

5. Acknowledgements

This work is being partially supported by the TSI2004-03667 project of the R+D Spanish plan.

References:


Blanco-Fernández, Y. et al., 2004(c): “AVATAR: An advanced multi-agent recommender system of personalized TV contents by semantic reasoning”. In Proceedings of the Fifth International Conference on Web Information Systems.


