A Knowledge Based Recommender System Based on Consistent Preference Relations

Luis Martínez, Luis G. Pérez, Manuel J. Barranco, and M. Espinilla

Department of Computer Science, University of Jaén, Jaén, Campus Las Lagunillas s.n. 23071-Jaén, Spain
martin@ujaen.es,lgonzaga@ujaen.es,barranco@ujaen.es,mestevez@ujaen.es

Summary. E-commerce companies have developed many methods and tools in order to personalize their web sites and services according to users' necessities and tastes. The most successful and widespread are the recommender systems. The aim of these systems is to lead people to interesting items through recommendations. Sometimes, these systems face situations in which there is a lack of information and this implies unsuccessful results. In this chapter we propose a knowledge based recommender system designed to overcome these situations. The proposed system is able to compute recommendations from scarce information. Our proposal will consist in gathering user's preference information over several examples using an incomplete preference relation. The system will complete this relation and exploit it in order to obtain a user profile that will be utilized to generate good recommendations.

1 Introduction

In the last years Internet development has grown beyond all expectations. New services have arisen in order to meet the users' necessities. As a consequence of this development nowadays people can accomplish a great number of activities such as watching films, buying books or flowers, chatting with other people, etc.

Usually these services are designed to offer a wide range of items and/or activities in order to be able to cater for the necessities or requirements of millions of potential users [14, 15, 19]. For instance, Amazon (http://www.amazon.com) sells over eight millions of books of any genre: scientific, business, or historical books as well as comics, novels or mystery books. iTunes Store (http://www.apple.com/itunes/store/) offers over three and a half millions of songs of a wide variety of artist such as The Killers, Bob Dylan, U2, or Sheryl Crow.

Although these services are designed to offer interesting items or services that fulfil the necessities or requirements of millions of potential users, many of them have problems to identify, and therefore, satisfy the necessities of a particular user. An e-bookshop can offer a wide range of mystery books in

L. Martínez et al.: *A Knowledge Based Recommender System Based on Consistent Preference Relations*, Studies in Computational Intelligence (SCI) **117**, 93–111 (2008)
www.springerlink.com

order to offer interesting ones for any user who likes this genre. However, it is not easy for this shop gets to know or finds out which particular books or kind of books each user likes. In such cases, the user has to search among all the books in order to find those ones that are more interesting for him/her. Due to the fact that the e-shops offer a huge variety of books, the search processes could be tedious and the user could waste much time exploring alternatives that he/she will never like and it is possible that the user gives it up and tries to find what he/she wants in a traditional shop where he/she can receive some pieces of advice from the shop assistant.

As a consequence of these problems, many tools have arisen to assist people in their searches. The most famous and successful ones are the Recommender Systems. These systems were first developed in the e-commerce area. Quite often e-commerce customers have to face situations in which the web site offers them a huge range of items that potentially could meet their requests, however only a small set of them really fulfil their necessities and many times they are hard to find out. These systems were developed with the aim of leading these customers towards interesting items by means of recommendations, limiting the offered items or sorting them according to the customers' necessities or tastes.

In the literature we can find different techniques to generate recommendations. Essentially, all these techniques have the same aim and accomplish the same phases to make the recommendations. First of all, before any recommendation process begins, they need a data set stored. The sources of information and its nature can be very varied. Such information is provided by customers, users, experts and it is related to their opinions, preferences, descriptions... The recommendation process starts when a user wants to find out a new item and the Recommender System has already stored the previous dataset with information regarding the user him/herself and/or other users. Then, an algorithm combines the information provided by the user about his/her necessities and the information stored in the Recommender System to generate recommendations about which items are the most suitable for him/her. Depending on the algorithm used to generate the recommendations we can classify them into:

- *Demographic Recommender Systems [12].* In this type of systems, the recommendations are based on demographic information. A specific customer will receive recommendations according to the information they have about the people who belong to the same demographic group.
- *Content-Based Recommender Systems [15].* They gather information about the features of the items user has liked in the past and use this information to find other items that the user could like.
- *Collaborative Recommender Systems [7].* These systems predict the users' preferences as a weighted aggregation of other users' preferences, in which the weights are proportional to the similarity between users on the basis of their ratings.

- *Knowledge Based Recommender Systems [4].* These systems infer the recommendations using the knowledge they have about the users, the items and how the features of these items fulfil the users' expectations.
- *Utility Based Recommender Systems [8].* They make recommendations based on the computation of the utility of each item for the user.
- *Hybrid Recommender Systems [3, 5].* The aforementioned Recommender Systems present some problems and drawbacks. Some authors have proposed to combine these techniques to smooth out these disadvantages and therefore improve the accuracy of the recommendations.

To choose the most suitable items for a user, these systems use information about the items, the users, their necessities, tastes... Sometimes this information is scarce and insufficient. Classical Recommender Systems, Collaborative and Content-based, are unable to make accurate recommendations in such cases. For instance, both of them need historical information about which items the user has liked in the past. If this information is not available (for example, it is a new customer) then, they cannot find out which items could be recommended. To smooth out these drawbacks some proposals have been presented. One of them is the Knowledge Based Recommender Systems. In these systems users state their preferences choosing an example that represent their preferences. The system defines a user profile based on the description of the example, and then, the system finds out which items are the most suitable one according to the user profile.

In this chapter we shall propose a Knowledge Based Recommender model that tries to improve the gathering process and the recommendations of the classical Knowledge Based Recommender Systems by using more examples and employing preference relations. To accomplish the gathering process, the user provides his/her preferences over a small set of items. This set contains examples that the user has chosen to represent his/her necessities. The user's preferences are expressed by means of an incomplete preference relation in which the user only supplies a row (or a column) of the relation. This incomplete preference relation will be completed by means of a method based on a consistency property, and from this relation, the system will compute a user profile that will be used to generate the recommendations. Thus, the gathering process is easier for users since they do not have to provide much information, given that the system can compute and complete by itself.

2 Preliminaries

In this section we shall review some preliminaries needed to understand the model that will be presented in the following section. First of all, we shall study the lack of information in Recommender Systems. Secondly we shall present a brief review of Knowledge Based Recommender Systems. Thirdly, we shall describe the preference relations. And finally we shall show a method to complete an incomplete preference relation by using the consistency property.

2.1 Problems in Classical Recommendater System Models

In the real world these systems face situations in which the information about user's necessities and tastes is not available or is scarce. For instance, some recommender systems ground their recomendations in the historical information about the user. If they are dealing with a new user, they will not be able to generate any recomendations. Even though, if they have historical information about the user, it may not be useful or enough for the current search, i.e., the user is looking for something that is neither related to his/her necessities in the past nor the necessities of other users. Moreover, the border that differentiates when the recommender system has enough information to generate recomendations and when it needs more information is incredibly blurry [5].

These problems particularly concern Classical Recommender Systems, both the Collaborative and the Content-based ones, which require historical information about their users. Some of the most common problems are [5]:

- *The new user ramp-up problem.* If the user has few ratings, Recommender Systems may not be able to make recommendations. This problem is presented in both Collaborative and Content-based Recommender Systems.
- *New item ramp-up problem.* In Collaborative Recommender Systems, items with few ratings are unlikely recommended, even though, they could be interesting for the users.
- *Grey sheep problem.* We can find this problem in Collaborative Recommender Systems. There might exists users whose ratings are not consistently similar with any group of users, and for this reason, they will rarely receive any accurate recommendation.
- *Quality dependent of large historical data set.* Many times, to obtain acceptable recommendations, a good and large historical dataset is needed.

These problems can cause recommender systems to lead the user towards false positives (items that are not truly interesting for him/her). If the user purchases the recommended item and finds out that he/she does not like it, the user will be unlikely to use the recommender system again [17] and this can cause a loss of money and customers. To sort out these problems some solutions have been presented, such as the Hybrid Recommender Systems [5] or the Knowledge Based Recommender Systems [4]. The aim of the first ones is to overcome the drawbacks of these Recommender Systems combining them to smooth out the above problems. The most usual combination is between the Collaborative and the Content-Based Recommender Systems. For instance, this kind of Hybrid Recommender Systems does not suffer from the new item ramp-up problem. However, the Knowledge Based Recommender Systems face the problem of lack of information from another point of view. These systems exploit the information provided by the user about their necessities and the knowledge that the system has about the items that can be recommended, to find out which items match the user real expectations.

In this chapter we develope a variation of a Knowledge Based Recommender Systems. In the next subsection we shall explain in further detail the working of a classical knowledge based recommender system.

2.2 Knowledge Based Recommender Systems

These Recommender Systems attempt to arise recommendations by exploiting the knowledge they have gathered about the items, the users, ... The algorithms used to infer these recommendations are usually based on case based reasoning [11]. These algorithms deal with three types of knowledge:

- *Catalog knowledge.* Knowledge that the Recommender System has about the items and their features.
- *Functional knowledge.* These systems need to know how items might meet the user's necessities.
- *User's knowledge.* The system needs to gather information about the user's necessities in order to find which items satisfy his/her necessities.

The acquisition of user's knowledge is the most challenging and important process in Knowledge Based Recommender System. For instance, this knowledge can be gathered through general demographic information, but the better and more knowledge we have about his/her necessities, the more accurate recommendation will be made. That is the reason why the most usual way to obtain this knowledge is directly requiring an example of the user's necessities. With this example the system is able to define a user profile that describes user's necessities. Then, it can find which items satisfy these necessities and they are returned as recommendations.

The main advantage of this kind of Recommender Systems is that they do not suffer from problems such as, the *new user* or *new item ramp-up problem* or those ones that are related to historical data about the users. As a consequence of this fact they are suitable in situations where there is no historical information (or it is very scarce) about the user.

Even though, these systems are easy to use, they present some drawbacks in the gathering process users' preferences, i.e., the user's knowledge. First of all, in some contexts users can find thousands and thousands of items related to their necessities. Many times it could be so difficult to find an example of what the user needs as to find directly what he or she really needs. And secondly, although the user can find an example of his/her necessities, it is possible that this example does not match exactly with his/her real expectations. The user profile defined from this example will not faithfully represent his/her necessities and therefore, he/she will not obtain a suitable recommendation. In order to solve this problem, this type of Recommender Systems let the user refines his/her user profile modifying, removing, or adding some features. Nevertheless, this process could be hard, time-consuming and not all the users can be willing to do so.

2.3 Incomplete Preference Relations

The numerical preference relations have been widely use to model preferences for problems such as decision-making problems [6,10,16]. In this representation the intensity of preference between any two alternatives of a set of feasible ones, $X = \{x_1, \ldots, x_n\}$ $(n \geq 2)$, is measured with a scale $[0, 1]$.

Definition 1. *[2] A numerical preference relation P on a set of alternatives X is a function on the alternative set $X \times X$ that is defined as following:*

$$\mu_P : X \times X \rightarrow [0, 1] .$$

Every value in the matrix P represents the preference degree or intensity of preference of the alternative x_i over x_j:

- $p_{ij} = 1/2$ indicates the maximum grade of indifference between x_i and x_j $(x_i \sim x_j)$.
- $p_{ij} = 1$ indicates that x_i is absolutely preferred to x_j
- $p_{ij} > 1/2$ indicates that x_i is preferred to x_j $(x_i \succ x_j)$

based on this representation we also know that $p_{ii} = \frac{1}{2} \forall i \in \{1, \ldots, n\} (x_i \sim x_i)$.

In an ideal situation the information provided by the user should be consistent and complete, however, many times in real situations this is not possible or suitable. For instance, users could be under time pressure or some alternatives could be unknown. In these situations, it would be more suitable to represent his/her preferences by means of an incomplete preference relation.

In our case, we know that time is a key issue in the gathering process of Knowledge Based Recommender Systems. Therefore, we shall propose that the users of our recommender system will provide preferences about different examples by using a preference relation that is a structure easy to exploit in order to obtain a user profile. However, to avoid a time-consuming process, instead of expecting the user provides a complete preference relation, the system will require an incomplete one, just a row (or a column) of the preference relation.

From the incomplete preference relation the system extracts as much information as it can. To do so, the system will fill it up by using a method that ensures that the resulting relation is not only complete, but also consistent. In the following section we shall review a method to complete this kind of relations.

2.4 A Method for Filling Preference Relation Based on the Consistency Property

The concept of consistency is usually characterized by the idea of transitivity. Transitivity represents the idea that the preference value obtained by comparing directly two alternatives should be equal to or greater than the preference

value between those alternatives obtained using an indirect chain of alternatives [13, 18]. Some of the suggested transitivity properties that we can find in the literature are the *Triangle condition* [13], the *Weak transitivity* or the *Additive transitivity* [18].

The last one seems a suitable property to characterize consistency in numerical preference relations and has been used successfully to construct consistent numerical relations from incomplete ones [1, 9].

Definition 2. *[1, 9] A numerical preference relation is "additive consistent" when for every three options on the problem $x_i, x_j, x_k \in X$ their associated preference degrees p_{ij}, p_{jk}, p_{ik} fulfil the following expression [9]:*

$$(p_{ij} - 0.5) + (p_{jk} - 0.5) = (p_{ik} - 0.5)\, \forall i, j, k.$$

A simple and practical method for filling a complete preference relation from an incomplete one that only has got the values of a row (or a columnn) is the following one [1, 9]:

Step 1. Let $X = \{x_1, \ldots, x_n\}$ be a discrete set of alternatives. The expert must provide a row (or a column) of the preference relation

Step 2. To utilize the known elements in P to determine all the unknown elements, and thus get a consistent preference relation, P', using the following expressions obtained from definition 2:

1. $p_{ij} + p_{jk} + p_{ki} = \frac{3}{2}$
2. $p_{i(i+1)} + p_{(i+1)(i+2)} + \cdots + p_{(j-1)j} + p_{ji} = \frac{j-i+1}{2} \forall i < j$

Step 3. End.

Example. Let's Suppose that we have a set of four alternatives $\{x_1, x_2, x_3, x_4\}$. If we know that $\{p_{12} = 0.55, p_{13} = 0.7, p_{14} = 0.95\}$, we shall have the following preference relation:

$$P = \begin{pmatrix} 0.5 & 0.55 & 0.7 & 0.95 \\ & 0.5 & & \\ & & 0.5 & \\ & & & 0.5 \end{pmatrix}.$$

If we use the previous algorithm we obtain:
$p_{21} = \frac{3}{2} - p_{12} - p_{22} = \frac{3}{2} - 0.55 - 0.5 = 0.45,$
$p_{31} = \frac{3}{2} - p_{13} - p_{33} = \frac{3}{2} - 0.7 - 0.5 = 0.3,$
...
therefore:

$$P' = \begin{pmatrix} 0.5 & 0.55 & 0.7 & 0.95 \\ 0.45 & 0.5 & 0.65 & 0.9 \\ 0.3 & 0.35 & 0.5 & 0.75 \\ 0.05 & 0.1 & 0.25 & 0.5 \end{pmatrix}.r$$

3 A Knowledge Based Recommender System Based on Numerical Consistent Preference Relations

Here, we will present our model for a Knowledge Based Recommender System that employs incomplete preference relations in order to build the user profile.

The main advantage of this kind of Recommender Systems is that they are suitable for casual exploration, i.e., they do not require to have any historical information (for instance, the items that the user has liked in the past) to make suitable recommendations. However, in this type of Recommender System the processes for building the user profile are usually more complex than in other kind of recommender systems such as the Collaborative or the Content-based ones. Besides, this user profile plays a key role in order to obtain an accurate recommendation. The more accurate it is gathered, the better recommentions are obtained.

Taking into account that the Knowledge Based Recommender System will deal with user's preferences and descriptions of the items, in our proposal both of them are modelled by means of numerical values. In future works we will study other types of preference modelling that can be more appropriate such as intervals, linguistic assessments and so on.

The Knowledge Based Recommender System that we propose has three phases (see Fig. 1):

(a) *Gathering user preference information.* The target of this phase is to obtain the preference information from the user. For this purpose, we need a small number of preferred items (4 or 5) that represents user's necessities and an incomplete preference relation provided over them.
(b) *Building the user profile.* The preference relation and the items' descriptions are used to build the user profile. First of all, the system fills up the incomplete preference relation and obtains partial profiles. These profiles express the user's preferences related to a specific item. Afterwards, they are aggregated to obtain the user profile.
(c) *Recommendation.* Eventually, the system recommends the items that are the closest to the user profile, i.e., the items that are the best to fulfil the user's real expectations.

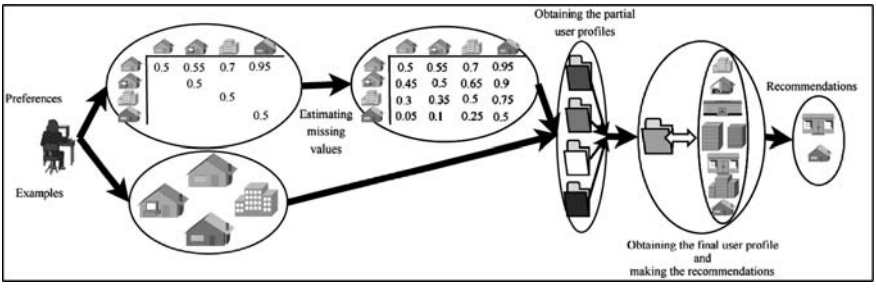

Fig. 1. Recommendation Model

3.1 Gathering User Preference Information

Initially, the user, u, chooses four or five items as examples of his/her preferences or necessities. Moreover, this process would be as difficult as (or more) finding directly the item(s) he/she likes. In order to make easier this choice, the system suggests a subset of representative items in which the user must select the examples of his/her necessities. This subset should be big enough to have items that represent any kind of user's necessities, these items ought to be "well-known" for almost everybody, but not too big because users could find this task too teadious. If he/she had to choose these examples from all the item database, he/she would waste much time exploring useless alternatives. We must remark that there is not any correlation between "well-known" and "preferred", i.e., in this set of well-known items we will find preferred items as well as items which the users do not like.

Let $X = \{x_1, x_2, \ldots, x_m\}$ be the set of items to be recommended and each one is described by a vector of features $x_i = \left\{c_i^1, \ldots, c_i^t\right\}$, the system offers a subset $X_r = \{x_1^r, x_2^r, \ldots, x_{m'}^r\}$ $(m' \leq m)$ that contains the most representative or well-known items of X $(X_r \subseteq X)$. The aim of this step is to obtain the user preference information. To do so, it must accomplish these two steps:

1. *Acquiring an incomplete preference relation.* The user provides it over the set of examples that represents his/her necessities.
2. *Filling the preference relation.* To exploit the above preference relation and define the user profile it is required that the system fills it up in order to build a complete and consistent preference relation.

Now, we shall present these steps in depth.

Acquiring an Incomplete Preference Relation

Once the user, u, selects from the subset of X_r four or five items, $X_u = \{x_1^u, \ldots, x_n^u\}$, as examples of his/her preferences, he/she has to choose one of them as the closest example to his/her necessities. Then, the user compares this example with the other ones assessing his/her preferences in a numerical value belonging to the interval $[0, 1]$.

Although, the user is only required to give a row of the preference relation $(p_{11}, \ldots, p_{1n})$, the system needs a complete preference relation to generate better recommendations. Therefore, the system will complete a consistent one by using the algorithm presented in Sect. 2.4. This way of computing the user preferences provides us two advantages:

(a) The user only provides the minimum and necessary information (a row or a column of a preference relation).
(b) The preference relation has not got inconsistent values because the algorithm will build a consistent preference relation from the incomplete preference relation.

Filling Up the Preference Relation

After the user has provided one row (or column) of the preference relation, the system can fill up the relation applying the properties presented in [1]. The result is:

$$P' = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p^*_{21} & p^*_{22} & \cdots & p^*_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p^*_{n1} & p^*_{n2} & \cdots & p^*_{nn} \end{pmatrix},$$

where p_{1j} is a value that the user has provided about the preference of example x^u_1 over the example x^u_j, and p^*_{ij} is an estimated value for the preference of the example x^u_i over x^u_j. By definition, p_{ii} has the value 0.5 (that means indifference).

3.2 Building the User Profile

The next phase of this model is to build the user profile. To accomplish this task the system will build a set of partial profiles, one for each item. Then, the partial user profiles will be combined to obtain the final user profile that will be used to compute the recommendations. This phase consists of two steps (see Fig. 2):

1. *Building partial user profiles.* The system will compute the partial user profiles from user's preference.
2. *Computing the user profile.* This user profile represents the knowledge about the user's necessities and it will be utilized to obtain the most suitable items for the user.

Now, we shall explain these steps in further detail.

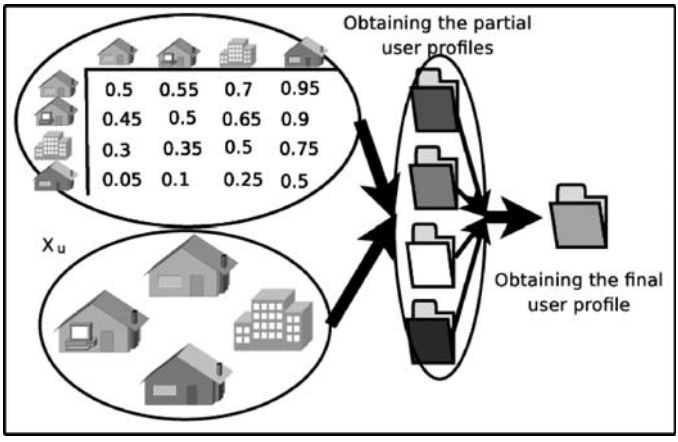

Fig. 2. Building the user profile

Building Partial User Profiles

Before building the user profile, the system will obtain partial profiles for each item that was chosen as example of the user's necessities. This partial profile will represent the user's preference regarding each item. For a given item, x_j^u, the system will build a partial profile, pp_j, related to this item aggregating the vectors of features of the other items different from x_j^u. That way, for the item x_j^u, the system will combine the description of the items $\{x_i^u, \forall i \neq j\}$. Our aim is to build a partial user profile that take into account that some items are closer to the user needs or tastes than others. To measure the importance of each item the system will use the filled preference relation, so that, for the partial user profile for the item x_j^u, the importance of the item $\{x_i^u, i \neq j\}$ is p_{ji}. To aggregate the vector of features of each item (its description) the system will use the IOWA operator (Induced OWA operator) proposed in [21].

The IOWA operator is used to aggregate tuples of the form (v_i, a_i). Within these pairs, v_i is called the order inducing value and a_i is called the argument value. The following procedure for performing the IOWA aggregation was suggested:

$$F_W\left(\langle v_1, a_1\rangle, \ldots, \langle v_l, a_l\rangle\right) = W^T B_v,$$

where $B_v = (b_1, \ldots, b_l)$ is the result of ordering the vector $A = (a_1, \ldots, a_l)$ according to the value of the order inducing variables, v_i, and W^T is the column vector of weights which satisfies:

$$W = (w_1, \ldots, w_l)$$
$$w_i \in [0,1] \quad \forall i \sum_{i=1}^{l} w_i = 1.$$

Our goal in this step is to obtain partial profiles $\left\{pp_j = \left(c_{pp_j}^1, \ldots, c_{pp_j}^t\right)\right\}$, one for each item x_j^u, aggregating the vectors $\left\{(c_i^1, \ldots, c_i^t), \forall i \neq j\right\}$ that describe the item $\{x_i^u, \forall i \neq j\}$. Each element $c_{pp_j}^k$ is obtained by aggregation of the $n-1$ elements $\left\{c_i^k, \forall i \neq j\right\}$. In this process, we need to choose order inducing variables, such as the IOWA operator suggest. For this purpose, we will take the column j of the preference relation $(p_{1j}, p_{2j}, \ldots, p_{nj})$. So, for every attribute we apply the following function:

$$c_{pp_j}^k = F_W\left(\langle p_{1j}, c_1^k\rangle, \ldots, \langle p_{nj}, c_n^k\rangle\right) = W^T B_v.$$

Then, the vector $B_v = (b_1, \ldots, b_{n-1})$ is given by an ordering, from the greatest to the smallest value, of the elements of the set $\left\{c_i^k, \forall i \neq j\right\}$ according to such order inducing variables, $(p_{1j}, \ldots, p_{nj})$ where p_{ij} represents the preferences of the example x_i^u over the example x_j^u.

In the literature there are different methods to compute the weighting vector $W = (w_1, \ldots, w_{n-1})$. For instance, We could associate it with a linguistic quantifier [21]. The selection of the quantifier will depend on the type of problem, items, etc.

Computing the Final User Profile

Now, we have a set of partial user profiles, $\{pp_1, \ldots, pp_n\}$, the system will aggregate them in order to obtain an unique and final user profile that state the user's preferences and tastes (see Fig. 3). This aggregation process is very similar to the previous one and the system will also use the IOWA operator. For every attribute the system will apply the following function:

$$c_{fp}^k = F'_W\left(\left\langle p_1, c_{pp_1}^k\right\rangle, \ldots, \left\langle p_n, c_{pp_n}^k\right\rangle\right) = W'^T B'_v,$$

where the vector $B'_v = (b'_1, \ldots, b'_n)$ is given by an ordering, from greatest to smallest value, of the elements of the set $\left\{c_{pp_i}^k\right\}$ according to the order inducing variables, $(p_1, \ldots, p_n)$ and the weighting vector $W' = (w'_1, \ldots, w'_n)$.

The inducing variables $(p_1, \ldots, p_n)$ represents the importance of each alternative. The most important alternative, which is the closest to the user's needs, will have the greatest value and the furthest alternative, the smallest value. To obtain these values we need to compute the importance of each partial user profile. The importance of the partial user profile, pp_i, is computed by using the following function:

$$p_i = \frac{1}{n-1} \sum_{j=1|j\neq i}^{n} p_{ji}.$$

This function computes the importance, p_i, as a mean of the preferences provided by the user over the item x_i. These preferences are obtained from the preference relation that was filled up in Sect. 3.1.

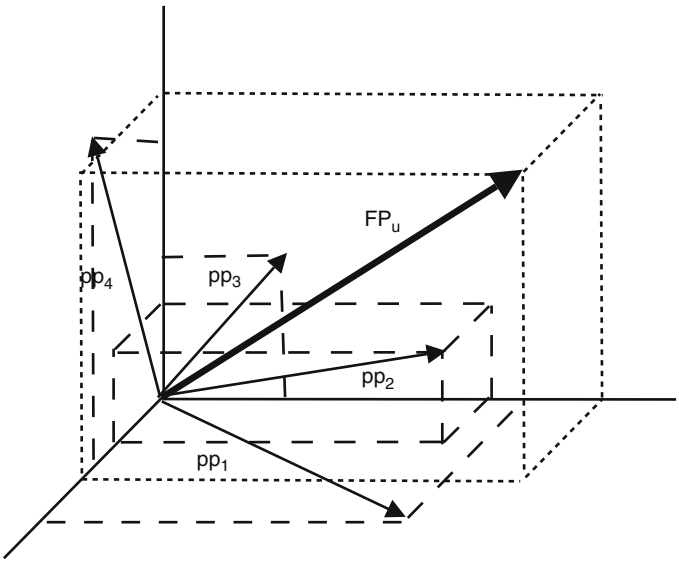

Fig. 3. Final User Profile. Vector representation

Finally, the system obtains the user profile, FP_u, for the user, u, that will be used in the recommendation phase:

$$FP_u = \left\{c_{fp}^1, \ldots, c_{fp}^t\right\}.$$

3.3 Recommendation

Once the user profile $FP_u = \left\{c_{fp}^1, \ldots, c_{fp}^t\right\}$ has been computed, the system will recommend the most suitable items to the user's necessities and tastes. The system has a item database $X = \{x_1, x_2, \ldots, x_m\}$ in which the system keeps all the items that can be recommended. Each item $x_i \in X$ is described by a set of features $x_i = \left\{c_i^1, \ldots, c_i^t\right\}$. To compute a score that measures the similarity between an item, x_i, and the user profile we shall used a similarity function based on the cosine of two vectors [22]. To acomplish these computations we shall deal with the user profile and the descriptions of items as vectors composed by t features defined in a t-dimensional space. Then, we shall define the similarity function based on the cosine of two vectors (see Fig. 4):

Definition 3. *The similarity between the user profile, FP_u and the item x_i is obtained as*

$$Similarity\,(FP_u, x_i) = cos\left(\overrightarrow{FP_u}, \overrightarrow{x_i}\right) = \frac{\overrightarrow{FP_u} \cdot \overrightarrow{x_i}}{||FP_u|| \cdot ||x_i||}.$$

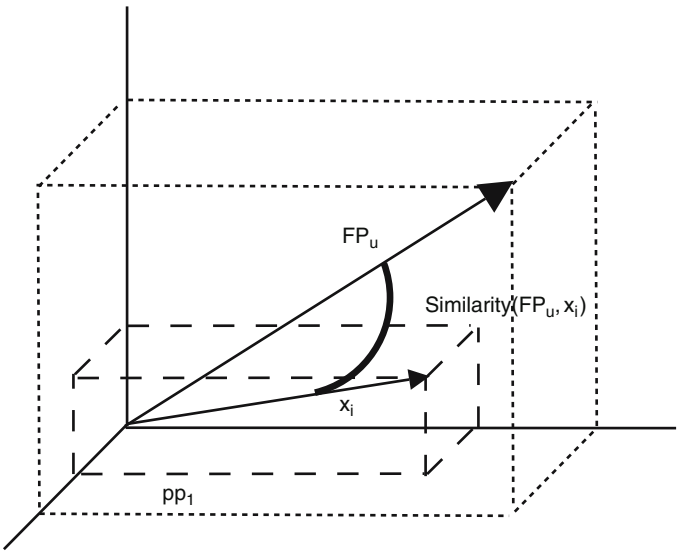

Fig. 4. Similarity between the final user profile and an item

The final recommendation(s) will be those items that are closest to the final user profile, FP_u, i.e., its overall similarity is greater. It is very likely that among the closest items the user could find the items that were chosen as examples of his/her necessities. These items must be left out from the final solution because their aim was to represent something close to what the user really needs, not to fulfil his/her necessities.

4 Example

In this section, we shall apply our model to a specific problem where a user wants to obtain some recommendations. The items that can be recommended are stored in a database $X = \{x_1, x_2, \ldots, x_m\}$. Each item is described by a vector of features, $x_i = \left\{c_i^1, \ldots, c_i^t\right\}$, in which each feature is assessed in the interval $[0, 1]$ (see Table 1).

The system will show the set X_r of the most "well-known" examples of the system, and the user will select the four closest examples of his/her necessities (see Table 2):

The examples chosen by the user are $X_u = \{Product\ 11, Product\ 15, Product\ 23, Product\ 24\}$. Moreover, the user provides his/her preferences about these examples. In our case, he/she provides the preference of the first item over the other ones:

Table 1. Item database

Item ID	Description
Product 1	(0.74, 0.37, 0.26, 0.41, 0.39, 0.86, 0.22, 0.050, 0.62, 0.62)
Product 2	(0.36, 0.52, 0.74, 0.28, 0.42, 0.14, 0.76, 0.12, 0.36, 0.59)
Product 3	(0.55, 0.012, 0.81, 0.88, 0.45, 0.97, 0.13, 0.60, 0.88, 0.49)
Product 4	(0.20, 0.18, 0.61, 0.93, 0.28, 0.49, 0.78, 0.88, 0.49, 0.67)
...	...
Product 11	(1.0, 0.2, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0)
...	...
Product 15	(1.0, 0.3, 1.0, 1.0, 0, 0, 1.0, 0, 1.0, 1.0)
...	...
Product 21	(0.82, 0.30, 0.89, 0.46, 0.38, 0.12, 0.26, 0.27, 0.57, 0.49)
...	...
Product 23	(0.5, 0.1, 0.4, 0.8, 1.0, 1.0, 1.0, 0.4, 0.9, 0.9)
Product 24	(0.1, 0.3, 0.3, 0.9, 1.0, 0, 0.78, 0, 0.85, 0.95)
...	...
Product m	...

Table 2. Given examples

Item	Description
...	...
Product 11	(1.0, 0.2, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0)
...	...
Product 15	(1.0, 0.3, 1.0, 1.0, 0, 0, 1.0, 0, 1.0, 1.0)
...	...
Product 23	(0.5, 0.1, 0.4, 0.8, 1.0, 1.0, 1.0, 0.4, 0.9, 0.9)
Product 24	(0.1, 0.3, 0.3, 0.9, 1.0, 0, 0.78, 0, 0.85, 0.95)
...	...
Product m'	...

$$P = \begin{pmatrix} 0.5 & 0.25 & 0.4 & 0.65 \\ & 0.5 & & \\ & & 0.5 & \\ & & & 0.5 \end{pmatrix}.$$

Now, with these preference values the system must find and recommend the most suitable items among all the items of its items database (see Table 1).

First of all, the system fills up the user's preference relation using the algorithm reviewed in Sect. 2.4 and obtains a complete and consistent preference relation:

$$P' = \begin{pmatrix} 0.5 & 0.25 & 0.4 & 0.65 \\ 0.75 & 0.5 & 0.65 & 0.9 \\ 0.6 & 0.35 & 0.5 & 0.75 \\ 0.35 & 0.1 & 0.25 & 0.5 \end{pmatrix}.$$

In the next phase the system will compute the user profile, but before, it must compute the weights that will be used to obtain the partial profiles and the final user profile. To obtain these weights we shall use the following function based on the use of a non-decreasing linguistic quantifier, Q [20]:

$$w_i = Q\left(\frac{i}{m}\right) - Q\left(\frac{i-1}{m}\right), \quad i = 1, \ldots, m,$$

where m is the number of values we are going to aggregate, and Q is the linguistic quantifier "*at least half*" [20]:

$$Q(x) = \begin{cases} 0 & si\ x < a \\ \frac{x-a}{b-a} & si\ a \leq x \leq b \\ 1 & si\ x > b \end{cases} \quad with\ a = 0,\ b = 0.5.$$

The above function obtains the weighting vectors, W and W', that will be utilized to obtain the partial user profiles and the final user profile respectively.

Table 3. Partial profiles

Partial profile	Description
$pp_{Product\ 11}$	$(0.83, 0.23, 0.8, 0.93, 0.33, 0.33, 1, 0.13, 0.97, 0.97)$
$pp_{Product\ 15}$	$(0.67, 0.13, 0.6, 0.87, 1, 1, 1, 0.6, 0.93, 0.93)$
$pp_{Product\ 23}$	$(1, 0.27, 1, 1, 0.33, 0.33, 1, 0.33, 1, 1)$
$pp_{Product\ 24}$	$(0.83, 0.23, 0.8, 0.93, 0.33, 0.33, 1, 0.13, 0.97, 0.97)$

The values obtained for the first vector are $W = \{0.67, 0.33, 0\}$ and for the second one $W' = \{0.5, 0.5, 0, 0\}$.

With these weights and using the complete and consistent preference relation the system aggregates the items descriptions to obtain the partial profiles. For example, to obtain the first value of partial profile related to the first example, $pp_{Product\ 11}$, the system shall compute:

$$c^1_{pp_{Product\ 11}} = F_W\left(\langle 0.75, 1\rangle, \langle 0.6, 0.5\rangle, \langle 0.35, 0.1\rangle\right) = 0.83.$$

We can see the partial profiles in Table 3.

To obtain the final user profile we shall aggregate the partial profiles using the weights W'. For instance, to obtain the first value of the final user profile the system shall compute:

$$c^1_{fp} = F'_W(\langle 0.57, 0.83\rangle, \langle 0.23, 0.67\rangle, \langle 0.43, 1\rangle, \langle 0.77, 0.83\rangle) = 0.83.$$

Where the inducing variables $\{p_1, \ldots, p_4\}$ are calculated, from the preference relation, as follows:

$$p_1 = \frac{1}{3}\sum_{j=1|j\neq 1}^{n} p_{ji} = \frac{1}{3}(0.75 + 0.6 + 0.35) = 0.57,$$

$$p_2 = \frac{1}{3}\sum_{j=1|j\neq 2}^{n} p_{ji} = \frac{1}{3}(0.25 + 0.35 + 0.1) = 0.23,$$

$$p_3 = \frac{1}{3}\sum_{j=1|j\neq 3}^{n} p_{ji} = \frac{1}{3}(0.4 + 0.65 + 0.25) = 0.43,$$

$$p_4 = \frac{1}{3}\sum_{j=1|j\neq 4}^{n} p_{ji} = \frac{1}{3}(0.65 + 0.9 + 0.75) = 0.77.$$

If we compute all the values we shall obtain the following final user profile (see Table 4).

The last step in our model is the recommendation phase. In this phase the system will compute the similarity of the final user profile with the description

Table 4. Final user profile

Final profile
$(0.83, 0.23, 0.8, 0.93, 0.33, 0.33, 1., 0.13, 0.97, 0.97)$

Table 5. Recommendations

Item	Similarity
Product 4	0.914
Product 21	0.897
Product 3	0.895
...	...

of each item of the item database and it will recommend those items that are the closest to the user's necessities. The system will use the function defined in Sect. 3.3 that is based on a cosine measure. The results of this comparisons can be seen in Table 5.

Therefore, according to these results the closest item to the user necessities is the item *Product* 4, the second one is the *Product* 21, the next one is the *Product* 8 and so on.

5 Conclusions

When people visit an e-shop, they usually can find thousands of items related to their necessities, but only a few of them can fulfil their real expectations and sometimes it is hard to find them. The Recommender Systems assist them in finding these items among all of them. There are different types of Recommender Systems, such as the Content-based and the Collaborative ones. These kind of Recommender Systems make good recommendations as long as they have enough information about the users, their necessities or the items. However, when this information is scarce or not available, they are unable to make recommendations.

In this chapter we have presented a model for Knowledge Based Recommender System that provides a technology to avoid this problem. It gathers the information from the users using a numerical preference relation structure that only requires to be filled with a small number of values and then, using the consistency property the system will complete the preference relation in order to exploit it and to obtain better recommendations but without forcing the users to spend much time in the generation of his/her profile.

Acknowledgements

This work is partially supported by the Research Projects TIN-2006-02121, JA031/06 and FEDER funds.

References

1. S. Alonso, F. Chiclana, F. Herrera, E. Herrera-Viedma, J. Alcalá-Fdez, and C. Porcel. A consistency-based procedure to estimate missing pairwise preference values. *International Journal of Intelligent Systems*, 23(2):155–175, 2008.
2. B. De Baets, B. Van de Walle, and E. Kerre. Fuzzy preference structures without incomparability. *Fuzzy Sets and Systems*, (76):333–348, 1995.
3. C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and contentbased information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, 1998.
4. R. Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(32), 2000.
5. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
6. F. Chiclana, F. Herrera, and E. Herrera-Viedma. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems*, (97):33–48, 1998.
7. D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
8. H.R. Guttman. *Merchant Differentiation through Integrative Negotiation in Agent-mediated Electronic Commerce*. Master's thesis, School of Architecture and Planning, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 1998.
9. E. Herrera-Viedma, F. Herrera, F. Chiclana, and M. Luque. Some issues on consistency of fuzzy preference relations. *European Journal of Operational Research*, (154):98–109, 2004.
10. J. Kacprzyk. Group decision making with a fuzzy linguistic majority. *Fuzzy Sets and Systems*, 18(2):105–118, 1986.
11. J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
12. B. Krulwich. Lifestyle finder: intelligent user profiling using large-scale demographic data. *AI Magazine*, 18(2):37–45, 1997.
13. R.D. Luce and P. Suppes. *Handbook of Mathematical Psychology*, chapter Preferences, Utility and Subject Probability, pages 249–410. Wiley, New York, 1965.
14. L. Martínez, L.G. Pérez, and M. Barranco. A multi-granular linguistic content-based recommendation model. *International Journal of Intelligent Systems*, 22(5):419–434, 2007.
15. R.J. Mooney and L. Roy. Content-based book recommending using learning text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204, 2000.
16. S.A. Orlovsky. Decision-making with a fuzzy preference relation. *Fuzzy Sets Systems*, 1:155–167, 1978.

17. B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.
18. T. Tanino. *Non-Conventional Preference Relations in Decision Making*, chapter Fuzzy Preference Relations in Group Decision Making, pages 54–71. Springer-Verlag, New York, 1988.
19. Y. Wang, Y. Chuang, M. Hsu, and H. Keh. A personalized recommender system for cosmetic business. *Expert Systems with Applications*, (26):427–434, 2004.
20. R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.
21. R.R. Yager. Induced aggregation operators. *Fuzzy Sets and Systems*, 137(1): 59–69, 2003.
22. Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 13–22. ACM/Springer, 1994.