User Semantic Preferences for Collaborative Recommendations

Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, Khaled Bsaies

User Semantic Preferences for Collaborative Recommendations

Sonia Ben Ticha [1,2] , Azim Roussanaly [2], Anne Boyer [2], Khaled Bsaies[1],

[1] URPAH Teams Tunis, Tunisia
[2] KIWI Team LORIA laboratory Nancy, France

Abstract. Personalized recommender systems provide relevant items to users from huge catalogue. Collaborative filtering (CF) and content-based (CB) filtering are the most widely used techniques in personalized recommender systems. CF uses only the user-rating data to make predictions, while CB filtering relies on semantic information of items for recommendation. In this paper we present a new approach taking into account the semantic information of items in a CF system. Many works have addressed this problem by proposing hybrid solutions. In this paper, we present another hybridization technique that predicts users 'preferences for items based on their inferred preferences for semantic information. With this aim, we propose a new approach to build user semantic profile to model users' preferences for semantic information of items. Then, we use this model in a user-based CF algorithm to calculate the similarity between users. We apply our approach to real data, the MoviesLens dataset, and compare our results to standards user-based and item-based CF algorithms.

Keywords. Recommender systems, collaborative filtering, semantic information, user modeling

1 Introduction

Personalized Recommender Systems (RS) provide relevant items to users from a large number of choices by defining a profile for each user. In this work, user model is based on an analysis of usage. Collaborative filtering (CF) and content-based (CB) filtering are the most widely used techniques in RS.

The fundamental assumption of CF is that if users X and Y rate n items similarly, and hence will rate or act on other items similarly [8]. However, CF techniques must face many challenges [10], like the data sparsity problem; the scalability problem for big database with the increasing numbers of users and items; the cold start problem when new user logs in, the system ignores his or her preferences, or when new item appears in the database, there is no way to be recommended before it is rated. CB [3.] assumes that each user operates independently. In CB RS, user will be recommended items similar to the ones he preferred in the past.

To overcome the disadvantages of both techniques and benefit from their strengths, hybrid solutions have emerged. In this paper, we present a new approach taking into account the semantic information of items in a CF system. In our approach, we design a new hybridization technique, called User Semantic Collaborative Filtering (USCF), which predicts users' preferences for items based on their inferred preferences for semantic information. We assume that items are described by a structured data in which there is a small number of attributes and there is a known set of features that each attribute may have. The originality of this work is in the used method to build the user semantic model.

Our contribution is summarized as follows: (i) we propose a new approach for building user semantic model, that inferred the user preferences for semantic information of items, (ii) we define a classification of attributes and propose a suited algorithm for each class, (iii) for each relevant attribute, we build the user semantic attribute model using the suited algorithm, then the user-semantic model is the concatenation of all user semantic attribute model, (iv) we provide predictions and recommendations by using the user semantic model, in a user-based CF algorithm [6.], for computing similarity between users, (iv) we perform several experiments with real data from the MoviesLens data sets, which showed improvement in the quality of predictions compared to only usage CF and a hybrid algorithm.

The rest of the paper is organized as follows: Section 2 summarizes the related work. Standard user-based CF is described in Section 3. Section 4 describes our USCF approach and experimental results are given in Section 5. Finally, we conclude with a summary of our findings and some directions for future work.

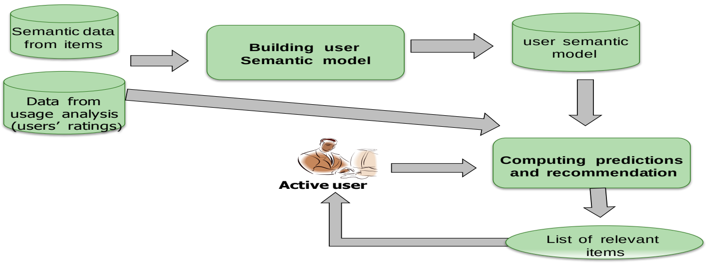
Semantic data from items
Building user Semantic model
user semantic model
Data from usage analysis (users' ratings)
Active user
Computing predictions and recommendation
List of relevant items

Fig. 1. Architecture of our system: USCF approach

2 Related Work

CF is the most widespread used technique in recommender systems. It was the subject of several researches [1][6.][7.]. Purely CB recommender systems are less widespread, techniques used are from information retrieval and information filtering research. Notable works can be found in [12][3.].

Several RS use a hybrid approach by combining CF and CB methods, which helps to avoid certain shortcomings of CB and CF systems. Many systems have been developed since [2][11]. In [9], authors use TF/IDF measure to calculate the weight of feature for each user. For computing this weight, they use only items liked by the user; which forces to define a rating value threshold to select the items preferred by user. This solution has two shortcomings; first, the threshold value is very subjective. Second, two users share the same tastes when, not only, they like the same things, but also, when they hate the same things; but in this approach, items not liked by users are not selected. In [17], authors are inferring user preferences for item ‘tags using several measures. This work is suitable only for item ‘tags and cannot be used for others kinds of attributes.

3 User based CF algorithm

User-based [6.] and Item-based [7.], algorithms are the most prevalent CF memory-based methods [1]. They are both based on the k-Nearest-Neighbors algorithm. The first computes similarities between users and in the second, similarities are computed between items. In our approach we have applied the user-based CF algorithm for recommendation. Its principle consists of the following steps:

1. *Calculatng the similarities sim(u_a,v):* which reflect the correlation between the active user u_a and all others users *v*. The similarity is computed by the Pearson correlation (1), introduced by Resnick et al. [6.].

$$sim(u_a, v) = \frac{\sum_i (r_{u_a,i} - \bar{r}_{u_a})(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u_a,i} - \bar{r}_{u_a})^2}\sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}} \quad (1)$$

   where the i summations are over the items that both users u_a and *v* have rated and, $\bar{r}_{\mathrm{v}}$ is the average rating of the rated items of the user *v*.

2. *Computing the predictions pr(u_a,i):* predicts the rating value of active user u_a on non rated item *i*. In the user-based CF algorithm, a subset of nearest neighbors of u_a are chosen based on their similarity with him or her (2).

$$pr\left(u_{a,}, i\right) = r_{u_a} + k \sum_{v \in V} sim(u_a, v)(r_{v,i} - \bar{r}_v) \quad (2)$$

   V denotes the set of the nearest neighbors (most similar) of u_a that have rated item *i*. *V* can range anywhere from 1 to the number of all users.

4 User Semantic Collaborative Filtering (USCF) approach

In this work, we design a new hybrid recommender system, USCF, which predicts users' preferences for items based on their inferred preferences for semantic information. Our system consists of two components as shown in Fig. 1. Originality of this work is in the building of the user semantic model and its use in a CF algorithm.

1. *Building the user semantic model*: constructs the user semantic model by inferring user semantic preferences from user ratings and features item. This model is represented by the *users-features matrix* Q that provides inferred user ratings for relevant attributes features.
2. *Computing predictions and Recommendations:* we predict for each user a list of relevant items based on the user-based CF algorithm. Similarities between users are computed, by using the user-feature matrix Q instead of the user-item rating matrix.

4.1 Building the users semantic model

Pazzani et al. [3.] have identified three alternative item representations. Item can be represented by *structured data* in which there is a small number of attributes, each item is described by the same set of attributes, and there is a known set of values that each attribute may have; *unstructured* data such as, news articles, is an unrestricted text in which there are no attribute names with well-defined values; or *semi-*

structured data in which there are some structured attributes and some free-text. In this paper we are interested only to items described by structured data. The others representations will be addressed in future work. In the following, we will use the terms *feature* to designate a value of an attribute. We define two classes of attributes:

- **Dependent attribute**: attribute which having very variable number of features. This number is directly correlated to the number of items. Thus, when the number of items is increasing, the number of features is increasing also. For example: *directors* and *actors of movies*, *user tags*, and *key words of news*.
- **Non dependent attribute**: attribute which having a very few variable number of features, and this number is independent of the number of items. Thus, the increasing number of items has no effect on the number of features. For example: *movie genres*, *cuisine of restaurants*.

In our approach we define suited inferring user semantic preferences algorithm for each class of attributes. For the dependent attribute, we propose techniques issues from information retrieval and information filtering research like TF/IDF. For non dependent attribute, we use machine learning algorithms. The aim of this paper is to present our solution for the non dependent attributes. The dependent attributes will be addressed in future works.

All item attributes do not have the same degrees of importance to users. There are attributes more relevant than others. For instance, the movie genre can be more important, in the evaluation criteria of user, than the release date. Experiments that we conducted (see section 5) confirmed this hypothesis. In this paper, we assume that relevant attributes will be provided by a human expert.

Algorithm: User Semantic Model. *Builds the user semantic model*

Input: *I: item user ratings matrix (N lines and M columns); LRA: list of Relevant Attributes; F: Item semantic profile for all relevant attributes*

Output*: Q: user semantic model (N lines and L columns)*

Method:

```
(1) Initialization : Q= empty matrix
(2) For each relevant attribute in LRA do
        //This loop is fully parallelizable
(3)     if A is non dependent attribute then
(4)         $Q_A$= User_Semantic_non_dependent_attribute_Model()
        else
(5)         $Q_A$= User_Semantic_dependent_attribute_Model()
        end if
(4) end for
(5) for each relevant attribute A in LRA do
(6)     Q= horizontal concatenation of Q and $Q_A$
(7) end for;
(8) Normalization of Q
```

Fig. 2. User semantic model algorithm

Therefore, for each relevant attribute A, we build the corresponding user semantic attribute model. This model is described by the user-feature matrix Q_A (users in line and features of A in column) and provides the inferring user preferences for the features of the attribute A. The user semantic model (shown in Fig. 2), described by the user-feature matrix Q (users in line and features of all relevant attributes in column), is the horizontal concatenation of all user semantic attribute models. In the following, we will detail our method for building a user semantic attribute model for non dependent attribute. The method for dependent attribute will be addressed in future works.

User semantic attribute model for non dependent attribute.
User semantic attribute model provides user feature preferences for one attribute.User item ratings matrix U (N lines (users) and M columns (items)) is provided by an analysis of usage. $U_{u,i}=r_{u,i}$ is the rating of user u on item i; it can be either a missing value or a number on a specific scale if user u rated item i. So, we define:

- The usage analysis profile of user u is given by the following ratings vector: $U_u=(r_{u,1},r_{u,2},...,r_{u,i},...,r_{u,M})$, line u of matrix U.
- The usage analysis profile of item i is given by the following ratings vector $I_i=(r_{1,i},r_{2,i},...r_{u,i},...,r_{N,i})$, column i of matrix U, hence matrix $I=U^T$.

We have, also, semantic information about items provided by the features of each attribute. We define the semantic attribute based profile of item i on attribute A by the following features vector: $F_{Ai}=(b_{Ai,1},\ b_{Ai,2},...,b_{Ai,LA})$, L_A is the number of features of attribute A, where:

$$b_{\mathrm{Ai},f} = \begin{cases} 0\ if\ f\ is\ not\ a\ feature\ of\ item\ i \\ 1\ if\ f\ is\ a\ feature\ of\ item\ i \end{cases} \tag{3}$$

Thus, matrix F_A (M lines and L_A columns) provides the item semantic attribute model. Otherwise, we must distinguish between two kinds of attributes: *multi-valued* and *mono-valued* attribute. For a same item, if an attribute can have many values (features), then it is a multi-valued attribute (a *movie* can have many *genres)*; while if it must have only one feature it is called mono-valued attribute (a *movie* has only one *director*). This is reflected formally in the matrix F_A by:

- If A is multi-valued, then each line F_{Ai} of F_A will have many values equal to 1,
- If A is mono-valued, then each line F_{Ai} of F_A will have only one value equal to 1.

User semantic attribute model will be computed from the matrix U and F_A. For example, assume that we have a movies Data set with users ratings, and we want to inferred the preference, $q_{u,action}$, of user u on the *action movies*. This means computing an aggregation overall ratings of user u on all action movies:

$$q_{u,genre=action} = AGGR_{i.genre=action} r_{u,i} \tag{4}$$

The aggregation function can be a simple function like the average (AVG), or more complicated mathematical function like TF/IDF, or special user-defined function. For non dependent attribute, we choose to define a special user function, so we use a clustering algorithm to learn the user semantic attribute model.

Clustering algorithm
The idea is to partition all items in K clusters; each cluster is labeled by a feature or a set of features, hence K is less than or equal the number L_A of features. After running

the clustering algorithm, we obtain K cluster centers; each center $C_{A,k}$ is a vector that is equal to the mean of all items in cluster k and modeled the inferred users' preferences for the feature(s) associated to cluster k.

```
Algorithm: The k-means algorithm for partitioning
Input: I: item user ratings matrix (N line and M columns); A: mono-valued attribute;$F_A$: Item semantic profile for A.(M lines, $L_A$ columns )
Output:  $K_A$:the number of clusters; $C_A(q_{Ak,u})_{k=1,K;u=1,N}$: composed by K vectors, each vector is the center of a cluster, and $q_{Ak,u}$ is the inferred preference of user u on feature(s) labeled cluster k
Method:
(1) Initialization : provides the number of clusters $K_A$, and a set of $K_A$ means $m_1,\dots,m_{KA}$
(2) repeat
(3)     Assignment step: (re)assign each item to the cluster to which it's the most similar, based on the mean value of the items in the cluster;
            $C_k = \{i : distance(i, m_k) \leq distance(i, m_j), \forall\ 1 \leq j \leq K_A\}$ Where each i goes into exactly one $C_k$ , even if it could go in two of them.
(4)     Update step :update the cluster means, i.e., calculate the mean value of the items into each cluster;    $m_k = \frac{1}{|C_k|} \sum_{j \in C_k} j$
(5) until no change;
```

Fig. 3. The k-means clustering algorithm.

In fact, because items are described by their usage analysis profile, the ratings vector I_i, (see above in this section), the vector $C_{A,k}=(q_{Ak,1}, q_{Ak,2},\dots, q_{Ak,u},\dots,q_{Ak,N})$, where N is the number of users, and $q_{Ak,u}$ is the inferred preference of user u on feature(s) labeling the cluster k. For example, assume that we have a movies dataset and we want inferring users' preferences on *movie genre*. The non dependent attribute *genre* has L_{genre} features, if each cluster is labeled by a feature, then we will have L_{genre} clusters. Assume that the feature *action* is labeled the cluster l, then after running the clustering algorithm, the center of cluster l provides the *action*-users profile $C_{genre,1}=(q_{genre1,1},\dots,q_{genre1,u},\dots,q_{genre1,N})$ where $q_{genre1,u}$ provides the inferring preference of user u on *action movie* that labeling the cluster 1. Therefore, the user semantic attribute model for the non dependent attribute A is the matrix $Q_A=q_{u,Ak}$, u=1..N, k=1..K, which is other one than the transposed of the matrix $C_A=q_{Ak,u}$, k=1..K and u=1..N. Thus, $Q_{u,A}$ is the user semantic attribute profile of user u. However, the question is what clustering algorithm to use? As we have already said, we have two kinds of attributes, the multi-valued attribute and the mono-valued attribute. For multi-valued attribute, a same item can belongs to many clusters, while, for mono-valued attribute, an item must belong to only one cluster. So, for multi-valued attribute, the clustering algorithm must provide non disjointed clusters, whereas, for mono-valued attribute, the clustering algorithm must provide disjointed clusters. For this case, when we have a multi-valued attributed, we must use a fuzzy clustering algorithm, and with a mono-valued attribute we use a standard clustering algorithm. In previous work [13], we addressed the multi-valued attribute and we choose the Fuzzy C Mean algorithm as a fuzzy clustering algorithm. In this paper, we present our solution for the mono-valued attribute.

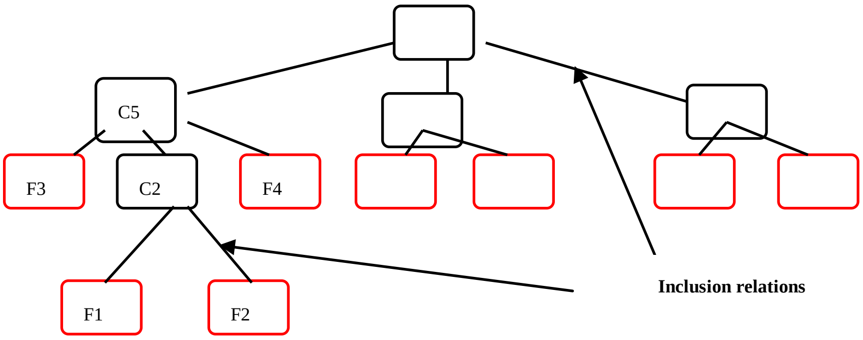

Fig. 4. Ontology of an attribute

After a study of several clustering algorithms we choose de K-Mean clustering algorithm [14] for its simplicity. The K-means algorithm takes the input parameter, K, and partitions a set of n objects into K clusters. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's center. The result of K-mean algorithm is depending on the number K of clusters, and the initial set of K means. In a standard K-mean algorithm, the initialization step is to choose randomly K observations from the data set and to use these as the initial means. In this paper, we design an algorithm for the initialization step of the K-mean algorithm shown in Fig. 3.

K-Mean initialization algorithm

Due to the data sparsity, we don't have sufficient data to infer user preferences for all features. This is the case for features assigned to very few items with a high number of ratings. That is why, we define two thresholds: MinNbRaIt that defines the minimum number of item ratings; MinNbItClust which indicates the minimum number of items by cluster in the K-mean initialization step. These thresholds are used in the K-Mean initialization algorithm (see Fig. 5).

Furthermore, we define two rules (eq. 5 and eq.6) describing the selection criterion.

$$C_k = \{item\ i:\ ratings\_Number(\ i._{attribute\ =f_k}\ ) \geq \mathrm{MinNbRaIt}\} \quad (5)$$

$$|C_k| >= \mathrm{MinNbItClust} \quad (6)$$

Algorithm: K-Mean initialization using ontology

Input: *A: mono-valued attribute; Ontology of A; F_A: Item semantic profile of A.(M lines, L_A columns); I: item user ratings matrix (N line and M columns);selection criterion (MinNbRaIt: minimum number of ratings; MinNbItClust: minimum number of items by cluster)*

Output*: K_A: the number of clusters;$m_{k=1...KA}$: mean of each cluster k*

Method:

```
(1)initialization: $LIC = \{C_k\}_{k=1...LA}$;  $C_k = \{item\ i:\ ratings\_Number(\ i._{attribute\ =f_k}\ ) \geq MinNbRaIt\}$
and labeled by feature $f_k$; LFC=∅
(2) repeat for each $C_k$ in LIC
(3)         if $|C_k| >= MinNbItClust$ then
(4)              add $C_k$ to LFC
(5)              remove $C_k$ from LIC
            else
```

```
(6)         if not exist, create cluster $C_j$ labeled by the father concept of the $C_k$ label in ontology
(7)         add items of $C_k$ to $C_j$
(8)         add $C_j$ to LIC
(9)         remove $C_k$ from LIC
        end if
 (10) until LIC=∅;
(11)$K_A$=|LFC|
(12)Compute the mean $m_k$ of each cluster $C_k$ in LFC, k=1...$K_A$
```

Fig. 5. K-Mean initialization algorithm using ontology

However, user preferences cannot be determined for features no checking the selection criteria defined by equations 5 and 6. That’s why we propose a solution to infer these preferences from ontology. We assume having an ontology describing the non dependent attribute. The concepts (in black in Fig. 4) of the ontology are interconnected hierarchically, and the leaf nodes describe the features of the attribute (in red in Fig. 4). We need only the inclusion relation between the nodes. For example, features F1 and F2 are included in the concept C2; features F3 and F4 and concept C2 are included in concept C5.

Each feature does not check the selection criteria defined above, will be replaced by its closest ancestor meeting the criteria, in the ontology. In the example described in Table 1, it is assumed that the threshold MinNbItClust is equal to 6, hence, features F1 and F3 satisfy the selection criterion, so a cluster will be assigned to each. However, as F2 does not satisfy the criterion, it will be replaced by its father C2, in the ontology; Similarly, C2 does not satisfy the criterion itself, it will be replaced by C5. In addition, F4 does not check the criterion; it will also be replaced by C5. The number of items assigned to the concept C5 is equal to 8 (5+3) and it’s greater than MinNbItClust. As, C5 satisfies the criterion, a cluster will be associated to it. The cluster assigned to C5 will represent the features F2 and F4. By using this initialization algorithm, we will be able to infer user preferences for the concept C5 which will group features F2 and F4.

Table 1. Example, $MinNbItClust = 6$

Feature	Nb items with $ratings\_Number >= MinNbRaIt$
F1	10
F2	5
F3	12
F4	3

Algorithm of user semantic attribute model for non dependent attribute

Fig. 6 summarizes the algorithm for building the user semantic attribute preferences for non dependent attribute.

Algorithm: User Semantic non dependent attribute Model.

Input: *I: item user ratings matrix (N line and M columns);A: non dependent attributeF_A: Item semantic profile for A*

Output*: K_A: the number of clusters; Q_A: the user semantic attribute model for A, (N lines and K_A columns)*

```
Method:
(1) Initialization : $Q_A$= empty matrix
(2) if A is mono-valued attribute then
(3)         $C_A$= K-Means_clustering_algorithm()
     else
(4)         $C_A$=Fuzzy_C_Mean_algorithm() [26]
     End if
(5) $Q_A$ = transposed matrix of $C_A$
```

Fig. 6. User semantic attribute algorithm for non dependent attribute

4.2 Computing predictions and Recommendation

To compute predictions for the active user, we use the user-based CF algorithm described in section 3. In our algorithm, we use the user feature matrix Q instead of users-items matrix U to compute users' similarities (see formula (1)).

4.3 USCF algorithm

Our approach resolves the scalability problem for several reasons. First, the building process of user-semantic model is fully parallelizable and can be done offline. Second, this model allows a dimension reduction since the number of columns in the user semantic model is much lower than those of user-item rating matrix. Third, the computing of similarities between users can be done offline, thus, only the computing of predictions will be done online. Beside the scalability problem, our algorithm alleviates the data sparsity problem by providing solution to the neighbor transitivity problem. In this problem, users with similar preferences may not be identifies as such if they haven't any items rated in common. Indeed, the number of missing values is much lower in the user feature matrix Q than in user-item ratings matrix U; thus, all similarities between users can be computed. This is not the case with the matrix U, because similarities between users who have no co rated items cannot be computed.

5 Performance study

In this section, we study the performance of the USCF algorithm against the standard User-Based CF [6.] (UBCF), the standard Item-Based CF (IBCF) [7.] and Average User Feature CF algorithm (AvgUFCF). For IBCF algorithm, we compute predictions using the Adjusted Cosine correlation measure which provides, according to [7.], best prediction accuracy. The AvgUFCF is building user semantic model by using the average (AVG) as an aggregation function (see formula (4)). We evaluate these algorithms in terms of predictions relevancy by using the Mean Absolute Error (MAE) [4]. MAE is the most widely used metric in CF research literature, which computes the average of the absolute difference between the predictions and true ratings (7).

$$MAE = \frac{\sum_{u,i}\left|pr_{u,i} - r_{u,i}\right|}{d} \tag{7}$$

d is the total number of ratings over all users, $pr_{u,i}$ is the predicted rating for user u on item i, and $r_{u,i}$ is the actual rating. Lower the MAE is, better is the prediction.

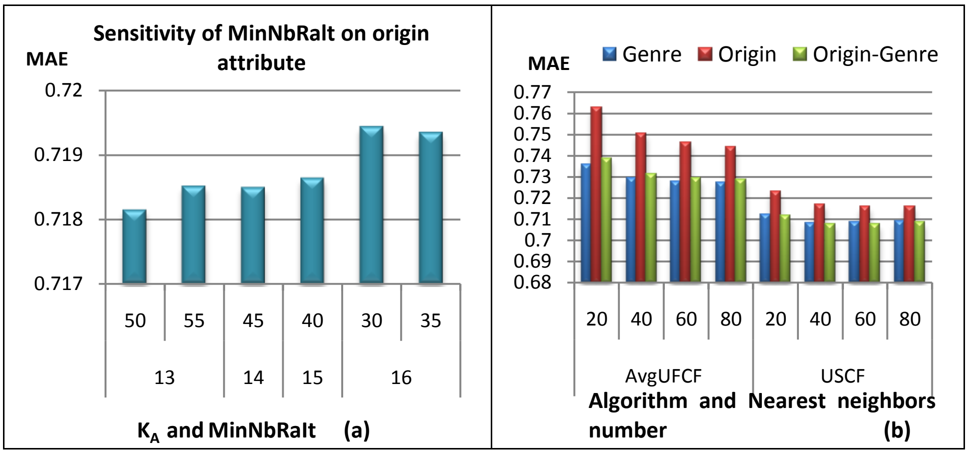

Fig. 7. Comparative results for USCF: (a) by varying the MinNbRaIt; (b) between USCF and AvgUFCF on all relevant attributes.

5.1 Experimental datasets

We have experimented our approach on real data from the MovieLens1M dataset of the MovieLens recommender system [5.]. The MovieLens1M provides the usage data set and contains 1,000,209 explicit ratings of approximately 3,900 movies made by 6,040 users. For the semantic information of items, we use the HetRec 2011 dataset [15] that links the movies of MovieLens dataset with their corresponding web pages at Internet Movie Database (IMDb) and Rotten Tomatoes movie review systems. We use the *genre* and the *origin country* of movies as non dependent attributes. Movie' genre is a multi-valued attribute whereas origin country is a mono-valued attribute. We use also the W3C movie ontology [16] for describing the origin of movie.

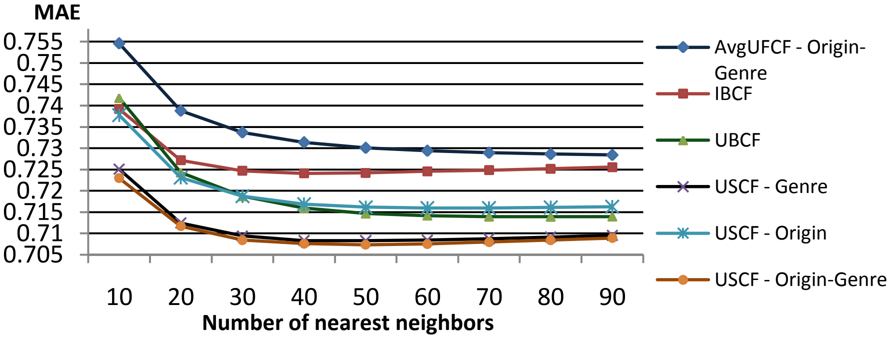

Fig. 8. Prediction accuracy for USCF v. IBCF, UBCF and AvgUFCF

We filtered the data by maintaining only users with at least 20 ratings and the movies origins existing in the ontology. After the filtering process, we obtain a data set with 6027 users, 3559 movies, 19 genres, 44 origins and a user-item rating matrix U with approximately 95% of missing values. The usage data set has been sorted by the timestamps, in ascending order, and divided into a training set (including the first 80% of all ratings) and a test set (the last 20% of all ratings). Thus, the ratings of each user, in the test set were assigned after those of the training set. In addition, after having tried several distance measures in the clustering algorithms, the cosines distance provides the best result. In the following, all results are obtained using this measure.

5.2 Results

It should be noted that the inferring user preferences for the attribute *genre* (multi-valued attribute see section 4.1) have been addressed in a previous work [13]. Therefore, we will not detail the experiments conducted for this attribute in this paper.

In Fig. 7 (a), the MAE has been plotted with respect to the MinNbRaIt parameter. It compares the K-Mean initialization algorithms on the attribute *origin* for MinNbIt-Clust =9 and 60 neighbors. We note that the accuracy of recommendations improves with the decreasing number of clusters K_A. In addition, the MAE converges for 50 ratings; this shows the impact of MinNbRaIt on the accuracy of the recommendations. In section 4.1 we have hypothesized that attributes don’t have the same importance to users. The plots in Fig. 7 (b) confirm this hypothesis; they show that the attribute *genre* provides better results than the attribute *origin*, for both algorithms USCF and AvgUFCF and regardless of the number of neighbors. Therefore, we can conclude that the attribute *genre* is more relevant than the attribute *origin.* Fig. 8 depicts the prediction accuracy of our algorithm, USCF, in contrast to those produced by IBCF, UBCF and AvgUFCF. The MAE has been plotted with respect to the number of neighbors in the *k*-Nearest-Neighbors algorithm and with the best parameters of each algorithm. In all cases, the MAE converges between 60 and 70 neighbors, however, our algorithm results in an overall improvement in accuracy. This improvement can be explained by many reasons. First, taking into account the semantic profile of items in the recommendation process. Second, user semantic model is built according to a collaborative principle; ratings of all users are used to compute the semantic profile of each user. It is not the case of the AvgUFCF algorithm; this may explain its results despite taking into account the semantic aspect. Third, the choice of the attribute can have significant influence on improving the accuracy. Lastly, users-features matrix Q has few missing values, so, it allows inferring similarity between two given users even when they have any items rated in common.

6 Conclusion and future work

In this paper, we have proposed a hybrid solution taking into account the semantic information in CF algorithm. The contribution of our solution over the solutions proposed in literature is the identification of the link between users’ ratings and items’ features. This link was defined by the user semantic model that inferring the user semantic preferences from the usage data. The originality of this work is in the used method to build the user semantic model. Indeed, we define two classes of attributes, the *dependent attribute* and the *non dependent attribute* and we propose an approach for inferring user semantic preferences for each class.

Our approach provides solutions to the scalability problem. Indeed, the built of the user semantic model is fully parallelizable and can be done offline. Therefore, it alleviates the data sparsity problem by reducing the dimensionality of data. The experimental results show that the USCF algorithm improve the prediction accuracy compared to usage only approach (UBCF and IBCF) and hybrid algorithm (AvgUFCF).

Furthermore, we have experimentally shown that, all the attributes don't have the same importance to users.

An interesting area of future work is to use machine learning techniques to automatically determine the relevant attributes. We will also further study the extension of the user semantic model to the dependent attribute and non structured data; study the use of the user semantic model to solve the cold start problem in which new items cannot be recommended to users because they haven't any rating; and lastly, study the impact of using other machine learning algorithms for building the user semantic attribute model for non dependent attribute and comparing their results.

References

1. J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering", Proc of the 14th Conf. on UAI '98, July 1998.
2. R. Burke, "Hybrid web recommender systems", in The Adaptive Web. LNCS, vol. 4321, , Springer-Verlag, 2007.
3. M. J. Pazzani, D. Billsus: "Content-based recommendation systems", in The Adaptive Web. Lecture Notes in Computer Science, Vol. 4321, pp. 325-341 Springer-Verlag, 2007.
4. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems", ACM Trans. on Information Systems, vol. 22, no. 1, 2004.
5. MovieLens, http://www.movielens.org/, retrieved: January, 2012.
6. P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews", Proc. of Conf. on Computer Supported Cooperative Work, ACM, pp. 175–186, North Carolina, 1994.
7. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," Proc. 10th Int'l WWW Conf., 2001.
8. X. Su, and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques". Adv. in Artif. Intell. Jan 2009.
9. P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Feature-weighted user model for recommender systems". Proc of the 11th Int'l Conf. on User Modeling, Greece, July 2007.
10. G. Adomavicius, and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", IEEE Trans on Knowledge and Data Engineering, vol.17, no. 6, pp. 734-749, June 2005.
11. A. Belén Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonteand A. Peleteiro "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition", In Information Sciences, vol. 15, pp. 4290-4311, Nov. 2010.
12. C. Musto, F. Narducci, P. Lops, M. Degemmis, and G. Semeraro, "Content-based personalization services integrating folksonomies". Proc. E-Commerce and Web Technologies, 10th Int'l Conf., September 2009, Linz, Austria.
13. Ben Ticha S., Roussanaly A. and Boyer A, "User Semantic Model for Hybrid Recommender System", In The First Int'l Conf. on Social Eco-Informatics- SOTICS October 2011 - Barcelona, Spain
14. Han J. and Kamber M. "Data Mining:Concepts and Techniques", Elsevier, 2006
15. 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011) at the 5th ACM Conf. on Recommender Systems (RecSys 2011)
16. A. Bouza: "MO – the Movie Ontology".. 2010. movieontology.org.
17. S. Sen,, J. Vig., and J. Riedl 2009. "Tagommenders: connecting users to items through tags". In Proc. of the 18th Int'l Conf on WWW, Madrid, Spain, April 2009.