User-Feature Model for Hybrid Recommender System

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

User-Feature Model for Hybrid Recommender System

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

E-mails: Sonia.benticha@ensi.rnu.tn, azim.roussanaly@loria.fr, anne.boyer@loria.fr, khaled.bsaies@fst.rnu.tn

***Abstract*—Recommender systems provide relevant items to users from a large number of choices. In this work, we are interested in personalized recommender systems where user model is based on an analysis of usage. Collaborative filtering and content-based filtering are the most widely used techniques in personalized recommender systems. Each technique has its drawbacks, so hybrid solutions, combining the two techniques, have emerged to overcome their disadvantages and benefit from their strengths. In this paper, we propose a hybrid solution combining collaborative filtering and content-based filtering. With this aim, we have defined a new user model, called *user-feature model*, to model user preferences based on items' features and user ratings. The *user-feature model* is built from the user item model by using a fuzzy clustering algorithm: the Fuzzy C Mean (FCM) algorithm. Then, we used the *user-feature model* in a user-based collaborative filtering algorithm to calculate the similarity between users. Applying our approach to the MoviesLens dataset, significant improvement can be noticed comparatively to the main CF algorithm, denoted as user-based collaborative filtering.**

***Index Terms*— Collaborative filtering, Content-based filtering, Fuzzy clustering, Hybrid recommender system**

I. INTRODUCTION

Recommender systems provide relevant items to users from a large number of choices. Several recommendations techniques exist in the literature [6]. Among these techniques, there are those that provide personalized recommendations by defining a profile for each user. In this work, we are interested in personalized recommender systems where the user model is based on an analysis of usage. This model is usually represented by a user-item rating matrix, which is extremely sparse (> 90% of missing data).

Collaborative filtering (CF) has been the first personalized recommender system [9]. In collaborative filtering, user will be recommended items that people with similar tastes and preferences liked in the past. Content-based filtering (CB) is another important technique, of recommender systems, it assumes that each user operate independently. In content-based recommender systems, user will be recommended items similar to the ones he preferred in the past. Content-based filtering uses techniques developed in information retrieval and, information filtering research. The major difference between CF and content-based recommender systems is that CF only uses the user-item ratings data to make predictions and recommendations, while content-based recommender systems rely on the features of items for predictions.

However, each technique introduces some shortcomings. In CF techniques, if a new item appears in the database, there is no way to be recommended before it is rated, this problem is known as Cold-start [8]. Neighbor transitivity [18] refers to a problem with sparse databases, in which users with similar tastes may not be identified as such if they have any items rated in common. On the other hand, if a user's taste is unusual, he cannot find neighbors, and gets inaccurate recommendations, this problem is known as Gray Sheep[18]. Content-based filtering suffers a problem of over-specialization where a user is restricted to seeing items similar to those already rated.

To overcome the disadvantages of both techniques and benefit from their strengths, hybrid solutions have emerged. Most of these hybrid systems are process-oriented: they run CF on the results of CB or vice versa. CF exploits information from the users and their ratings[8]. CB exploits information from items and their features. However, they miss the relation between, user ratings and items' features. This link may explain the user interests for an item.

In this paper, we propose a hybrid solution combining collaborative filtering and content-based filtering. Our solution defines a new user model, *user-feature model*, to model user preferences based on items content. Therefore, our user model is the link between the user ratings and the items' features and defines user-features preferences.

The *user-feature model* is built from the user-item model by using a fuzzy clustering algorithm: the Fuzzy C Mean (FCM) algorithm [3].

We used the *user-feature model* in a user-based CF algorithm [18] to calculate the similarity between users. We compared our results to the main CF algorithm, denoted as user-based CF (UB) [15]. The results obtained demonstrate the superiority of the proposed approach.

Our contribution is summarized as follows: (i) We construct a novel *user-feature model*, representing the link between user's preferences and item's features, (ii) We use a fuzzy clustering algorithm, FCM, for the construction of this model, (iii) We provide predictions and recommendations by using

the user-feature model, in a user-based CF algorithm, for computing similarity between users, (iv) We perform several experiments with MoviesLens data sets, which showed improvement in the quality of predictions compared to user-based CF.

The rest of the paper is organized as follows: section 2 summarizes the related work. The proposed approach is described in Section 3. Experimental results are given in Section 4. Finally, Section 5 concludes this paper.

II. Related Work

Recommender systems have become an independent research area in the middle 1990s after the apparition of the first paper on personalized recommender systems based on collaborative filtering [9]. Collaborative filtering is the most widespread used technique in recommender systems. It was the subject of several researches [15], [5], [16], [1].

Purely content-based recommender systems are less widespread. Techniques used are from information retrieval and information filtering research. Notable works can be find in Pazzani[14] and Ferman [7].

The Fab System of Balabanovic[2] counts among the first hybrid recommender systems. Several other systems have been developed since [4], [11], [17]. Most of these hybrid systems are process-oriented: they run CF on the results of CB or vice versa. In [20], authors integrate semantic similarities for items with user-rating similarities. The combined similarity measure was used in an item-based CF to generate recommendations. These works ignore the dependency between user-ratings and items' features. Taking account of the link between them can improve the quality of recommendation. In [19], this dependency was computed using the *term frequency/inverse document frequency* (TF-IDF) measure that is the best-known measures for specifying keyword weights in Information Retrieval. The authors use this measure to calculate the weight of feature for each user. In our approach, we used a fuzzy clustering algorithm, Fuzzy C Mean, to compute the estimated user-rating for each feature. The result of the Fuzzy C Mean algorithm is used to provide a new user profile based on the items' features. Thus, users are modeled by the user-feature model that defined the dependency between user rating and items' features (semantic of items).

III. Proposed approach

A. Notations

In this section, we provide details about the used terminology. Table I, summarizes the symbols used in this paper.

- The user profile is defined by a rating vector: $U_u=(r_{u,1}, r_{u,2}, \ldots, r_{u,i}, \ldots, r_{u,m})$
- The item profile is defined by:
  - a rating vector: $I_i=(r_{1,i}, r_{2,i}, \ldots r_{u,i}, \ldots, r_{N,i})$
  - an item-features vector: $F_i=(b_{i,1}, b_{i,2}, \ldots, b_{i,D})$ where:

TABLE I
DESCRIPTION OF THE USED SYMBOLS

Symbol	Meaning	Description
N	Number of users	
M	Number of items	
L	Number of features	
U	The user-item ratings matrix	With in general 93% to 95% of missing values
$I=U^t$	The item-user ratings matrix, U transposed	
U_u	The ratings vector of user U_u for all items	The user's profile
I_i	The ratings vector of item i by all users	The item's profile
F	the item-feature matrix	No missing value
$b_{i,f}$	The value of item-feature matrix	0 or 1
A	The user-feature matrix	result of our approach
A_u	The user-feature profile of user u	
?	Missing value	
$r_{u,i}$	Rating of user u on item i	
f	Feature	
i	Item	
u	User	
$P_k(I_i)$	The degree of item I_i of being in the cluster k	Fuzzy C Mean
m	The fuzzy parameter	Fuzzy C Mean

$$b_{i,f} = \begin{cases} 0 \; if \; item \; i \; hasn't \; feature \; f \\ 1 \; if \; item \; i \; has \; feature \; f \end{cases}$$

B. Architecture

We propose a system that takes into account the dependence between user's ratings and items' features. This dependency is represented by the user-feature matrix.

The aim of this study is to know whether the fact of taking into account the relationship between user's ratings and the features of items, in the recommender process, can improve the relevance of the recommended items.

The outline of our methodology consists of 2 steps as shown in Fig. 1:

1. *The user-feature matrix construction step*: by using a fuzzy clustering algorithm: the Fuzzy C-Mean [3] we build a user-feature profile from the item-user ratings matrix *I* and the item-feature matrix *F*. Obviously, for achievement reasons, this processing can be offline.
2. *The recommendation step*: we provide for each user a recommendation list of relevant items based on the user-based CF algorithm [15]. The similarity between two users is computed, in our algorithm, by using the user-feature matrix instead of the user-item rating matrix.

In the following sections, we describe each step in detail.

C. Construction of user-feature matrix

The user-feature matrix A(N lines and L columns), describes the user-feature profile for each user. The user-feature profile provides preferences of user *u* for all features

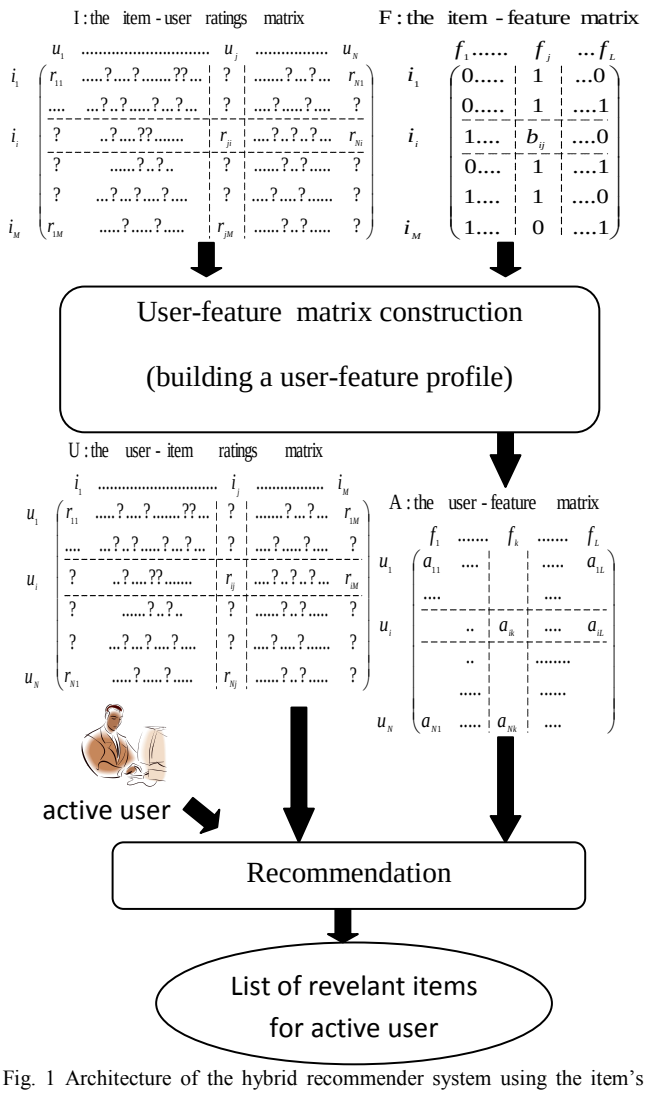

Fig. 1 Architecture of the hybrid recommender system using the item's features: *the Fuzzy user-feature CF*

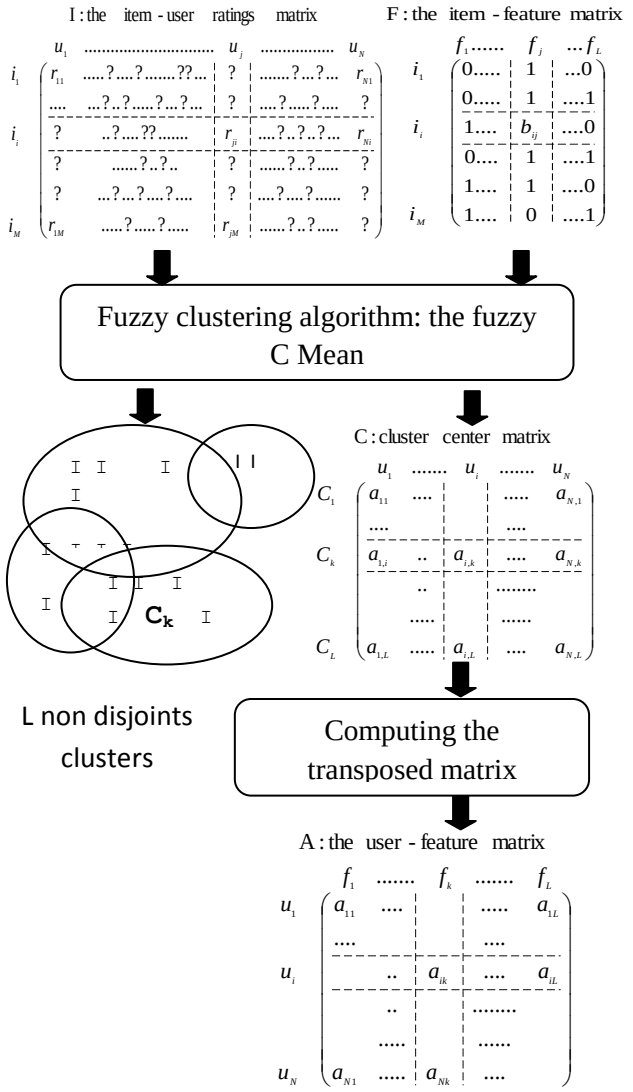

Fig. 2. The fuzzy clustering algorithm provides L non-disjoint clusters. C_k is the centroid of the cluster k, C_k is a generalized profile of the feature f_k, defined in N dimensional space. $C_k=A^t_k$

from his preferences for items. It is defined by the vector $A_u=(a_{u,1},\ldots,a_{u,f},a_{u,L})$. L is the number of features, $a_{u,f}$ indicates the preference of user *u* for feature *f* and will be computed by our algorithm.

In this section we describe in detail the steps of the construction of the user-feature model defined by the user-feature matrix A.

For building the matrix A, our algorithm computes before the transposed matrix A^t of A. The vector $A^t_f=(a_{1,f},a_{2,f},\ldots,a_{u,f},\ldots a_{N,f})$ is the profile of a feature *f* computed from the users' ratings. This profile can be performed by a generalization model like a partitioning method.

Then, the clustering algorithm classifies the set of items by features, so that, items within cluster have high similarity compared to their features. Thus, the center of each cluster defines the profile of the corresponding feature and is modeled by the vector A^t_f.

Since an item belongs to several features, we need to use a fuzzy clustering algorithm. In literature there are many fuzzy clustering algorithms. Initially, we choose the Fuzzy C Mean (FCM) algorithm [3]. In future works, we will test other algorithms and will compare the different results. FCM algorithm is very similar to the k-mean algorithm, but it provides non-disjointed clusters.

The construction of user-feature matrix consists of 2 steps as shown in Fig 2:

1. Performing of the features profiles by using the Fuzzy C Mean algorithm. This step provides L non-disjoint clusters represented by their centroid C_k, $k \in [1,L]$ and, for each item i a coefficient, $p_{k,i}$, giving the degree of membership of item I_i to cluster k.
2. Computing of the transposed matrix of C: C is an LxN dimension matrix; each line defined the profile of the corresponding feature. The transposed of C gives the

matrix A that is the user-feature matrix. A gives for each user *u* his features' preferences.

In the followings sections we present in details the FCM algorithm and its initialization step.

Fuzzy C Mean Algorithm (FCM)

The FCM algorithm is one of the most widely used fuzzy clustering algorithms. This technique was originally introduced by Jim Bezdek in 1981[3]. The FCM algorithm attempts to partition a finite collection of elements $E=\{X_1, X_2, \ldots, X_M\}$ into a collection of L fuzzy clusters with respect to some given criterion. Given a finite set of data, the algorithm returns

- a list of L cluster centers C_k such that $C_k=v_i, i=1,\ldots,N$
- a partition matrix P such that: $P=p_{ij}$, $i=1,\ldots,L$ and $j=1,\ldots,M$, p_{ij} is a coefficient $\in[0,1]$ giving the degree to which the element X_j belongs to the i-th cluster. Usually, the sum of those coefficients for any given element X is defined to be 1 as shown in equation (1).

$$\forall X \left(\sum_{k=1}^{nb\,clusters} p_{k,x} = 1 \right) \qquad (1)$$

The center of a cluster is the mean of all elements, weighted by their degree of belonging to the cluster (equation (2)).

$$C_k = \frac{\sum_X p_{k,x}^m X}{\sum_X p_{k,X}^m}. \qquad (2)$$

The coefficient of belonging is related to the inverse of the distance to the cluster center. In equation (3) the coefficients are normalized and fuzzyfied with a real parameter m>1 so their sum is 1.

$$p_{k,x} = \frac{1}{\sum_j \left(\frac{distance(C_k, X)}{distance(C_j, X)} \right)^{2/(m-1)}} \qquad (3)$$

The FCM algorithm consists of the followings steps:

-Choose a number of clusters,

-Assign to each element coefficients of belonging to the clusters,

- Repeat until the algorithm has converged:
  * Compute the center for each cluster, using the formula gives by equation (2)
  * For each element, computes its coefficients for being in the clusters, using the formula gives by equation (3).

In our algorithm the collection of elements is the items, X is replaced by I_i and the number of clusters is L.

Foe the distance measure, we use the Manhattan distance given by the formula of the equation (4)

$$distance(X,Y) = \|X - Y\| = \sum_{i=1}^{i=n} |x_i - y_i|. \qquad (4)$$

Where X and Y is two vectors in an n-dimensional real vector space.

The number of clusters

In most clustering methods, we must study the number of clusters to choose. Indeed, the results of some techniques could be influenced by this number. In our case, this problem does not arise; the number of clusters is equal to the number of features. Our aim is to provide a profile for each feature based on users ratings. This profile is given by the cluster centroid.

Initialization of the Fuzzy C Mean Algorithm

Like the K-mean algorithm, the FCM algorithm needs an initialization of the partition matrix or the clusters centers.

In our algorithm, we initialize the partition matrix with respect to the formula given in equation (1).We use, for that, the item-feature matrix *F*, then the degree to which the item i belongs to a cluster k is given by the equation (5)

$$p_{k,i} \begin{cases} = \frac{1}{\sum_{f=1}^{L} b_{i,f}}, if \;\; b_{i,f_k} = 1 \\ = 0, \quad if \; b_{i,f_k} = 0 \end{cases} \qquad (5)$$

Example:

	f_1	f_2	f_3
i_1	0	1	1
i_2	1	0	1
i_3	1	1	1
i_4	1	1	0

In this example, we have three clusters. $p_{3,4}=0$ because $b_{4,3}=0$, that means item 4 hasn't feature 3, and $p_{2,4}=0.5$ and $p_{1,4}=0.5$ then, $p_{1,4}+p_{2,4}+p_{3,4}=1$. We assume that all the features of an item have the same weight. This assumption can be changed if we have the information about the importance of each feature in an item.

D. Recommendation

For the recommendation process we use the user-based CF [15] algorithm that is a memory based algorithm. Memory-based CF algorithms use the entire or a sample of the user-item matrix to generate predictions. Every user is part of a group of people with similar interests. By identifying the so-called neighbors of the active user, predictions on new items for him or her can be produced.

The used-based *CF* algorithm, a prevalent memory-based *CF* algorithm, based on the KNN algorithm (K Nearest Neighborhood) consists of the following steps:

- *Calculate the similarity* $w_{u,v}$: which reflects the correlation between the two users u and v. The similarity is computed by the Pearson correlation introduced by Resnick et al. [15].
- *Compute the predictions:* produce predictions is the most important step in a collaborative filtering system. In the user-based *CF* algorithm, a subset of nearest neighbors of the active user are chosen based on their similarity with him or her, and a weighted aggregate of their ratings is used to generate predictions for the active user.
- *Recommendation*: the system recommends to the active user, the items with predicted ratings greater than a given threshold.

Equation (6) gives the Pearson correlation, formula that used in user-based CF algorithm.

$$sim(u,v) = w_{u,v} = \frac{\sum_i (r_{ui} - r_u)(r_{vi} - r_v)}{\sqrt{\sum_i (r_{ui} - r_u)^2}\sqrt{\sum_i (r_{vi} - r_v)^2}} \quad (6)$$

Where the *i* summations are over the items that both the users *u* and *v* have rated and r_u is the average rating of the rated items of the user *u*.

In user-based CF algorithm, the user-item matrix is used to compute user similarities. In our algorithm, we use the feature-user matrix instead. This allows inferring similarity between two users even when they have any co-rated items. Thus, our approach provides solution to the neighbor transitivity problem emanates from the sparse nature of the underlying datasets. In this problem, users with similar preferences may not be identifies as such if they haven't any items rated in common.

Furthermore, Pearson correlation is the most widely used measure in user-based CF research. That is why we chose for computing users similarities. In addition, we will be able to compare our results with those of the user-bases CF algorithm described in [15].

We use the equation (7) to calculate the similarity between users instead of the formula of equation (6).

$$sim(u,v) = w_{u,v} = \frac{\sum_f (a_{uf} - a_u)(a_{vf} - a_v)}{\sqrt{\sum_f (a_f - a_u)^2}\sqrt{\sum_f (a_{vf} - a_v)^2}} \quad (7)$$

Where the *f* summations are over the features that users *u* and v have both a value, and a_u is the average of $a_{u,f}, f=1,\ldots,L$

For computing the prediction, $pr_{u,i}$, for user *u* on non-rated item *i*, we used formula of equation (8).

$$pr_{u,i} = \mathrm{r}_u + k\sum_{v \in V} sim(u,v)(r_{vi} - r_v)$$
$$where \quad k = \frac{1}{\sum_{v \in V} |sim(u,v)|} \quad (8)$$

V denotes the set of H users that are the most similar to user *u* and who have rated item *i* (H can range anywhere from 1 to the number of all users).

Sim(u,v) is calculated in our algorithm by using formula of equation (7). That's mean; we use the user-feature matrix for computing the correlation between users.

The outline of our user-based CF consists of the following steps:

-Computing similarities between users, by using the user-feature matrix

-Computing predictions for active user by using the formula of equation (8)

-Recommend items that predicted rating is greater than a given threshold.

IV. Performance Study

In this section, we study the performance of our model named *Fuzzy user-feature CF* against the simple CF algorithm, denoted as user-based CF (UB) described in [15]. We have implemented all these methods in Java. We evaluate these techniques in terms of relevancy of predictions.

A. The used corpus and experiments

In order to compute the prediction relevancy in our system, we used the GroupLens [12] dataset. The latter is composed of 100.000 ratings of real users, 943 users, 1682 items and 19 features. Items are movies, and features are the movie's genres. A same film may have several genres, for example, the movie "*Toy Story*" has three genres: *Animation, Children's* and *Comedy*. Moreover, each user has rated at least 20 items.

The dataset has been divided into a training set (including 80% of all ratings) and a test set (20% of votes). We use the

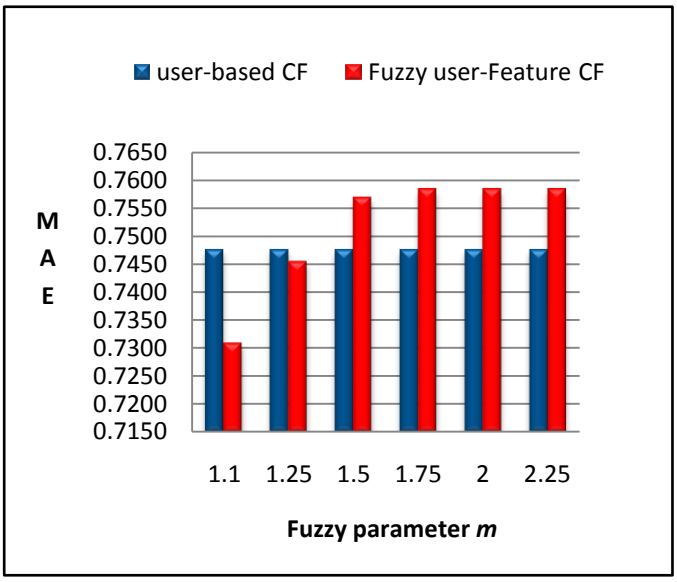

Fig 3. Comparison of prediction quality using the MAE between our algorithm and the used-based CF

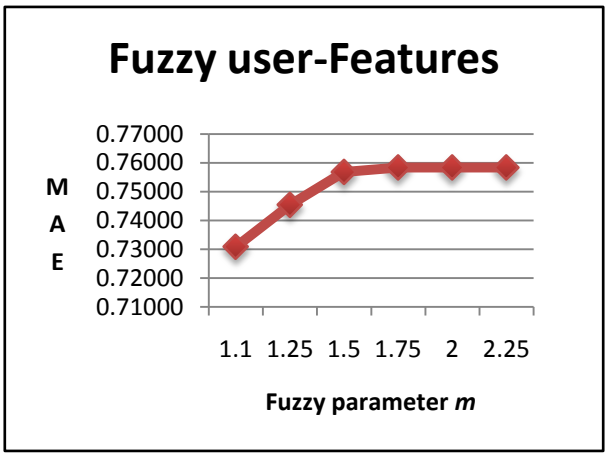

Fig 4. MAE of Fuzzy user-Feature against the fuzzy parameter m

five training and test set (*u1 to u5*) provided by GroupLens for cross validation. Thus, we repeat the experiment with each training and test set and we average the results.

TABLE II
COMPARISON OF PREDICTION QUALITY WITH VARYING THE FUZZY PARAMETER

	UB	*Fuzzy user-feature CF (m)*				
		1.1	1.25	1.5	1.75	2.0
MAE	0.7476	0.7309	0.7454	0.7568	0.7584	0.7584
RMSE	0.9486	0.9280	0.9460	0.9597	0.9619	0.9618

For the first step of our algorithm, *user-feature model construction* (see Fig 1), the data has been cleaned up, items that have less than 5 ratings, are removed from the dataset. Then, after the cleaning step, we have retained 1348 items and 18 genres.

We have executed the FCM algorithm for different values of the fuzzy parameter *m*. The number of iterations was set at 500. For the most values of *m*, the FCM has converged expect for *m*=1,25.

B. Results

We compare our algorithm with the user-based CF described in [15] by using the *Mean Absolute Error* (MAE)[10] and the *Root Mean Squared Error (*RMSE*)*.

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, as shown in equation (9).

$$MAE = \frac{\sum_{u,i} \left| pr_{u,i} - r_{u,i} \right|}{d} \qquad (9)$$

Where *d* is the total number of ratings over all users, $p_{u,i}$ 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.

RMSE is becoming popular partly because it is the *Netflix prize* [13] metric for movie recommendation performance:

$$RMSE = \sqrt{\frac{1}{d} \sum_{\{u,i\}} \left(pr_{u,i} - r_{u,i} \right)^2} \qquad (10)$$

RMSE amplifies the contributions of the absolute errors between the predictions and the true values.

Table II and Fig 3 demonstrate that our algorithm compares favorably against user-based CF for small values of *m.* The reason is when *m* is close to 1, then the cluster center closest to the items is given much more weight than the others and the FCM is similar to K-means. For large values of *m*, the *Fuzzy user-feature* CF converge and the MAE remain unchanged as shown in Fig 4.

We can conclude that the fact of taking account the features of item in the recommendation process improve the quality of recommendation. This can be explained by the fact that our approach allows inferring similarity between two given users even when they have any items rated in common, since we use the user-feature matrix instead of the user-item rating matrix for computing similarity in the recommendation process.

V. CONCLUSION AND FUTURES WORKS

In this paper, we have proposed a hybrid solution combining collaborative filtering and content-based techniques. The contribution of our solution over the solutions proposed in the literature is the identification of the link between user ratings and items' features. This link was defined by the user-feature model that modeled the user-feature preferences. The user-feature model, allows inferring similarity between two given users, even when they have any items rated in common. Thus, our approach provides solution for the *Neighbor transitivity problem*, in which users with similar tastes may not be identified, if they have not both rated any of the same items. Besides, our solution alleviates *the data sparsity problem* by reducing the dimensionality of data. In fact, the number of features is less than the number of items, so the user feature matrix dimension is smaller than the user-item ratings matrix dimension.

The results obtained are encouraging; they demonstrate the superiority of the proposed approach compared to the main CF algorithm: user-based CF [15].

As futures works, we will apply our approach to multi-criteria items. For example in the case of the MovieLens data sets, in addition to the genre, one can add the main actors in films. Otherwise, we will apply others Data Mining algorithms to construct the user-feature model.

VI. REFERENCES

[1] C.C. Aggarwal, J.L. Wolf, K-L. Wu, and P.S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering," *Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, Aug. 1999.*

[2] M. Balabanovic and Y. Shoham, "Fab: "Content-Based, Collaborative Recommendation," *Comm. ACM, vol. 40, no. 3, pp. 66-72, 1997.*

[3] J. C. Bezdek. "Pattern Recognition with Fuzzy Objective Function Algorithms". *Plenum Press, 1981*.

[4] D. Billsus and M. Pazzani, "A Personal News Agent that Talks, Learns and Explains," *Proc. Third Ann. Conf. Autonomous Agents,1999*.

[5] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.

[6] R. Burke, 2002. "Hybrid Recommender Systems: Survey and Experiments". *User Modeling and User-Adapted Interaction* 12, 4 (Nov. 2002), 331-370.

[7] M. Ferman, J. Errico., P. Van Breek, and I. Sezan, "Content-based filtering and personalization using structured metadata". *In Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (Portland, OR, USA, 2002), ACM Press, pp. 393–393.*

[8] Gediminas Adomavicius, Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE Transactions on Knowledge and Data Engineering, v.17 n.6, p.734-749, June 2005.*

[9] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[10] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl,"Evaluating collaborative filtering recommender systems,"*ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[11] P. Melville, R.J. Mooney, and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations," *Proc. 18th Nat'l Conf. Artificial Intelligence, 2002*.

[12] MovieLens, http://www.movielens.org/

[13] Netflix prize, http://www.netflixprize.com/.

[14] M. Pazzani and D. Billsus, "Learning and Revising User Profiles:The Identification of Interesting Web Sites," *Machine Learning,vol. 27, pp. 313-331, 1997.*

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews", in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175–186, Chapel Hill, North Carolina, (1994). ACM.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms". In *Proceedings of the 10th international Conference on World Wide Web* (Hong Kong, Hong Kong, May 01 - 05, 2001). WWW '01. ACM, New York, NY, 285-295.

[17] I. Soboroff and C. Nicholas, "Combining Content and Collaboration in Text Filtering," *Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering, Aug. 1999*.

[18] X. Su, and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques". *Adv. in Artif. Intell.* 2009 (Jan. 2009), 2-2.

[19] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Feature-Weighted User Model for Recommender Systems". In *Proceedings of the 11th international Conference on User Modeling* (Corfu, Greece, July 25 - 29, 2007).

[20] Mobasher, B., Jin, X., and Zhou, Y.: "Semantically enhanced collaborative filtering on the web". In *B.Berendt, et al.(eds): Web mining: From web to semantic web. LNAI* volume 3209, springer (2004)