Rocchio Algorithm to Enhance Semantically Collaborative Filtering

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

Rocchio Algorithm to Enhance Semantically Collaborative Filtering

Sonia Ben Ticha[1,2], Azim Roussanaly[1], Anne Boyer[1] and Khaled Bsaïes[2]
[1] *LORIA-KIWI Team, Lorraine University, Nancy, France*
[2] *LIPA Lab, Tunis El Manar University, Tunis, Tunisia*
{*sonia.benticha, azim.roussanaly, anne.boyer*}*@loria.fr, khaled.bsaies@fst.rnu.tn*

Keywords:
Hybrid Recommender System, Latent Semantic Analysis, Rocchio Algorithm.

Abstract:
Recommender system provides relevant items to users from huge catalogue. Collaborative filtering and content-based filtering are the most widely used techniques in personalized recommender systems. Collaborative filtering uses only the user-ratings data to make predictions, while content-based filtering relies on semantic information of items for recommendation. Hybrid recommendation system combines the two techniques. In this paper, we present another hybridization approach: User Semantic Collaborative Filtering. The aim of our approach is to predict users preferences for items based on their inferred preferences for semantic information of items. In this aim, we design a new user semantic model to describe the user preferences by using Rocchio algorithm. Due to the high dimension of item content, we apply a latent semantic analysis to reduce the dimension of data. User semantic model is then used in a user-based collaborative filtering to compute prediction ratings and to provide recommendations. Applying our approach to real data set, the MoviesLens 1M data set, significant improvement can be noticed compared to usage only approach, content based only approach.

1 INTRODUCTION

Thanks to computers and computer networks, our society is undergoing rapid transformation in almost all aspects. We buy online, read news online, listen music online, gather information by search engines and live a significant part of our social life on the Internet. However, the ongoing rapid expansion of the Internet, requires to help user to access to items that may interest her or him. Recommender Systems (RS) provide relevant items to users from a large number of choices. Several recommendations techniques exist in the literature. 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 described by a user-item ratings matrix, which is extremely sparse ($\geq 90\%$ of missing data).

Collaborative Filtering (CF) and Content-Based (CB) filtering are the most widely used techniques in RS. In CF, user will be recommended items that people with similar tastes and preferences liked in the past (Schafer et al., 2007). CB filtering assumes that each user operates independently and user will be recommended items similar to the ones he preferred in the past (Pazzani and Billsus, 2007). The major difference between CF and CB recommender systems is that CF relies only on the user-item ratings data to make predictions and recommendations, while CB relies on item content (semantic information) for recommendations. Hybrid approach (Burke, 2007) is another important technique, which combine collaborative and content-based methods to provide recommendations.

In this paper, we present a new approach: *User Semantic Collaborative Filtering (USCF)*, that takes into account the semantic information of items to enhance collaborative recommendations. User Semantic Collaborative Filtering (USCF) consists of two components as shown in Figure1: the first builds a new user model, the

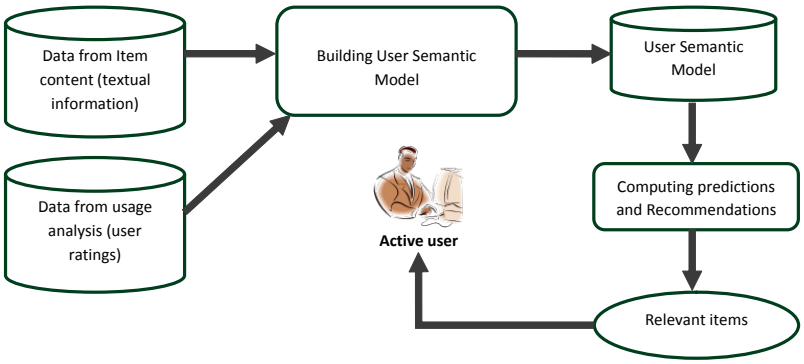


Figure 1: User Semantic Collaborative Filtering (USCF) Architecture.

User Semantic Model, by inferring user preferences for item content; the second computes predictions and provides recommendations by using the User Semantic Model in a user-based collaborative filtering algorithm (Resnick et al., 1994) to calculate the similarity between users. The originality of this work is in the building of the User Semantic Model. Indeed, assuming that items are represented by structured data in which each item is described by a same set of attributes, we build a *User Semantic Attribute Model* for each relevant attribute. With this aim, we define two classes of attributes: *dependent* and *non dependent* and we propose a suited algorithm for each class. User Semantic Model is then deducted from the horizontal concatenation of all User Semantic Attribute Model. In previous works (Ben Ticha et al., 2012; Ben Ticha et al., 2011) we have presented solutions based on machine learning algorithm to build a User Semantic Attribute Model for non dependent attribute.

In this work, we present a new approach for building a user semantic attribute model for dependent attribute by using Rocchio algorithm (Rocchio, 1971). Due to the high number of attribute values, and to reduce the expensiveness of user similarity computing, we apply a Latent Semantic Analysis (LSA) (Dumais, 2004) to reduce the size of the user semantic attribute model. We compare our results to the standards user-based CF, item-based CF and Content Based algorithms. Our approach results in an overall improvement in prediction accuracy.

The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 presents the Latent Semantic Analysis (LSA) algorithm. The algorithm of building the User Semantic Model is described in Section 4. Section 5 describes our approach to build User Semantic Attribute Model for non dependent attribute. Section 6 describes the recommendation component of our system. Experimental results are presented and discussed in Section 7. Finally, we conclude with a summary of our findings and some directions for future work.

2 RELATED WORK

Recommender Systems (RS) (Goldberg et al., 1992) have become an independent research area since the appearance of the first papers on collaborative filtering in the mid-1990s (Resnick et al., 1994; Shardanand and Maes, 1995; Hill et al., 1995). Collaborative Filtering (CF) (Ekstrand et al., 2011) is the most widespread used technique 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 (Su and Khoshgoftaar, 2009). In CF, the user model is usually described by a user-item rating matrix (users in lines and items in columns). Each value in the matrix is either a rating assigned by the user to the corresponding item, or a missing value if no information is available about the corresponding couple (user, item). Thus, CF systems try to predict ratings of items for the active user based only on the items previously rated by other users. BREESE et al. (Breese et al., 1998) have identified two classes of CF algorithms: *Memory-based* and *Model-based* algorithms. Memory-based algorithms use heuristic functions and the entire of the user-item ratings matrix to generate predictions. This allows them to be very reactive by integrating immediately modifications of users profiles into the system. User-based CF, introduced by Resnick et al. (Resnick et al., 1994), and Item-based CF, introduced by SARWAR et al. (Sarwar et al., 2001), are the most prevalent memory-based algorithms. They are both based on the k-Nearest-Neighbors algorithm. The first computes similarities between users and the second computes similarities between items. However, even if these methods work well with small-sized database, BREESE et al. (Breese et al., 1998) think that their scalability is problematic for big databases with great number of items and/or users. The model-based algorithms (Miyahara and Pazzani, 2000; Sarwar et al., 2002; Xue et al., 2005) constitute an alternative to this problem. These algorithms build descriptive models via a learning process. Thus, predictions are inferred from these models.

Content Based filtering (CB) (Lops et al., 2011) is

another important technique used in RS. Unlike CF recommendation methods, CB recommender systems rely on the content of items to provide personalized recommendations to active user. CB assumes that each user operates independently and recommends items similar to the ones he or she preferred in the past. Content-based systems focus on recommending items containing textual information, such as documents, web page, news or item attributes. That is why, Content-based filtering has its roots in information retrieval (Salton, 1989; Baeza-Yates and Ribeiro-Neto, 1999) and information filtering (Belkin and Croft, 1992) research. Furthermore, in content based filtering, item profile is usually represented by a Vector Space Model (VSM). Like collaborative filtering, there are two classes of CB algorithms : *Memory-based* and *Model-based* algorithms (Adomavicius and Tuzhilin, 2005). In memory-based algorithms (Pazzani and Billsus, 1997), various candidate items are compared with items previously rated by the active user and the best matching item(s) are recommended. The similarities between items are computed using heuristic formula like cosine measure. In model base techniques (Mooney and Roy, 2000; Pazzani and Billsus, 2007), predictions are provided based on a model learned from the underlying data using statistical learning and machine learning techniques.

However, CF and CB filtering must face many challenges (Adomavicius and Tuzhilin, 2005), like the *data sparsity* problem due to missing data in user-item matrix; the *scalability problem* for large datasets with the increasing numbers of users and items; the *cold start problem* when new user logs in, the system ignores his or her preferences. Furthermore, each technique introduced its own shortcomings. In CF technique, if new item appears in the database, there is no way to be recommended before it is rated, this problem is known also as *Cold-start problem*. *Neighbor transitivity* refers to a problem with sparse data, in which users with similar tastes may not be identified as such if they have any items rated in common. CB 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, several recommender systems use a hybrid approach by combining CF and CB techniques. The Fab System (Balabanović and Shoham, 1997) counts among the first hybrid RS. Many systems have been developed since (Burke, 2007). Moreover, because of the huge number of items and users, calculating the similarity between users in CF algorithm becomes very expensive in time computing. Dimension reduction of data is one of the solution to alleviate the expensiveness of users similarity computing (Sarwar et al., 2000). Mobasher et al. (Mobasher et al., 2004) combine values of all attributes and then apply a Latent Semantic Analysis (LSA) to reduce dimension of data. Sen et al. (Sen et al., 2009) are inferring user preferences for only one attribute, the item' tags, without reducing dimension. Manzato (Manzato, 2012) computes a user semantic model for only the movie genre attribute and applies a Singular Value Decomposition (SVD) to reduce the dimension of data.

3 LATENT SEMANTIC ANALYSIS (LSA)

Latent Semantic Analysis (LSA) (Dumais, 2004) is a dimensionality reduction technique which is widely used in information retrieval (Salton, 1989) and information filtering (Belkin and Croft, 1992) research. Given a term-document frequency matrix $M_{d,l}$ (d documents and l terms), in witch each document is described by a Vector Space Model (VSM), LSA is used to decompose it into two matrices of reduced dimensions and a diagonal matrix of singular values. Each dimension in the reduced space is a latent factor representing groups of highly correlated index terms.

Singular Value Decomposition (SVD) is a well known technique used in LSA to perform matrix decomposition. SVD decomposes the term-document frequency matrix M into three matrices D, Σ and T:

$$M = D_{d,r} * \Sigma_{r,r} * T^t_{r,l} \quad (1)$$

where D and T are two orthogonal matrices; $r = min(d, l)$ is the rank of matrix M. Σ is a diagonal matrix, where its diagonal entries contain all singular values of matrix M stored in decreasing order. D and T matrices are the left and right singular vectors. LSA uses a truncated SVD, keeping only the k largest singular values and their associated vectors, so

$$M \approx M^{'} = D_{d,k} * \Sigma_{k,k} * T^t_{k,l} \quad (2)$$

$M^{'}$ is the rank-k approximation of M. $M^{'}$ is what LSA uses for its semantic space. The rows in $D_{d,k}$ are the document vectors in LSA space and the rows in $T_{l,k}$ are the term vectors in LSA space. Each document in $D_{d,k}$ is represented by a set of k latent variables, instead of the original terms. Each term in $T_{l,k}$ is represented by a set of k latent variables instead of the original document. This, results in a much less sparse matrix. Furthermore, the generated latent variables represent groups of highly correlated terms in the original data, thus potentially reducing the amount of noise associated with the semantic information.

4 USER SEMANTIC MODEL

In this paper, we are interested only to items described by structured data. According to the definition of Pazzani et al. (Pazzani and Billsus, 2007), in structured representation, item can be represented by a small number of attributes, and there is a known set of values that each attribute may have. For instance, the attributes of a movie can be *title*, *genre*, *actor* and *director*. In the following, we will use the term

feature to refer to an attribute value. For instance, *Documentary*, *Musical* and *Thriller* are features of *movie genre* attribute.

4.1 Dependent and Non Dependent Attribute

In structured representation, each attribute has a set of restricted features. However, the number of features can be related or not to the number of items. That is why we have defined two classes of attributes:

- **Dependent attribute**: attribute, which having very variable number of features. This number is closely related to the number of items. So, when the number of items is increasing, the number of features is increasing also. For example: *directors* and *actors of movies*, *keywords*.
- **Non dependent attribute**: attribute, which having a very few variable number of features, and this number is not related to the number of items. Thus, the increasing number of items has no effect on the number of features. For example: *movie genre*, *movie origin* and *cuisine of restaurants*.

In addition, all 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 significant, in the evaluation criteria of user, than the *origin*. Experiments that we have conducted (see Section 7.4) confirmed this hypothesis. In this paper, we assume that relevant attributes will be provided by a human expert. Therefore, for each relevant attribute A, we build a *user semantic attribute model* that predicts the users preferences for its features (or group of features). This model is described by a matrix Q_A (users in lines and features (or group of features) of A in columns). In our approach, we design a suited algorithm for building the *user semantic attribute model* for each class of attribute.

For non dependent attribute, due to the low number of features, we have used a clustering algorithm. Section 4.2 briefly described the operating principle of our solutions that have been addressed in previous works (Ben Ticha et al., 2012; Ben Ticha et al., 2011).

For dependent attribute, we have explored techniques issues from information retrieval (IR) research. In (Ben Ticha et al., 2013) we have presented an approach based on user rating frequency. In this paper we present an other approach based on Rocchio algorithm (Rocchio, 1971).

The user semantic model for all relevant attributes, described by the matrix Q, is the result of the horizontal concatenation of all user semantic attribute models Q_A as shown in algorithm 1.

Algorithm 1 *Building User Semantic Model*

Input: *L: List of relevant attributes, users ratings, items content*
Output: *Q the User Semantic Matrix*

```
for A in L {This loop is fully parallelizable} do
    build Q_A the User Semantic Attribute Model of A
end for
for A in L do
    Q ← horizontal concatenation of (Q and Q_A)
end for
```

4.2 User Semantic Model for Non Dependent Attribute

Let us denote by S the set of items, U the set of users, s a given item $\in S$, u a given user $\in U$ and a rating value $r \in \{1, 2, ..., 5\} \equiv R$. U_s the set of users that rating the item s, then we define the rating function for item s by $\delta_s : u \in U_s \longmapsto \delta_s(u) \in R$. We denote also by F_A the set of features of attribute A, f a given feature $\in F_A$ and S_f the set of items associated to feature f. For instance if we consider the *movie genre* attribute, S_{action} is the set of all action movies.

An item s is represented by its usage profile vector $s_{up} = (\delta_s(u) - \overline{\delta_u})_{(u=1..|U|)}$, where $\overline{\delta_u}$ is the average rating of all rated items by user u. The idea is to partition all items described by their usage profile in K clusters, each cluster is labeled by a feature $f \in F_A$ (or a set of features).

The number K of clusters and the initial center of each cluster is computed by the initialization step of the clustering algorithm. In initial step, each cluster C_k consists of items in $\bigcup_{f\ labeling\ C_k} S_f$ and labeled by the set of corresponding features; so its center is the mean of its items described by their usage profile vector s_{up}. Moreover, an attribute can be mono valued or multivalued depending on the number of features that can be associated to a given item s. For example, the attribute *movie genre* is multivalued because a movie can have several genres while *movie origin* is a mono valued attribute because a movie has only one origin. Thus, if an attribute is multivalued, s can belong to several clusters C_k, while for mono valued attribute, an item should belong only to one cluster. Therefore, for multivalued attribute, the clustering algorithm should provide non disjointed clusters (a fuzzy clustering), whereas, for mono valued attribute, the clustering algorithm should provide disjointed clusters.

After running the clustering algorithm, we obtain K cluster centers; each center k is described by a vector $c_k = (q_{k,u})_{(u=1..|U|)}$. The K centers is modeling k latent variables issued from the features of the attribute A. Thus, the user semantic attribute model is described by the matrix $Q_A = (q_{u,k})_{(u=1..|U|,\ k=1..K)}$.

With non dependent attribute, the number of associated features is low, this is why the clustering is suitable. Moreover, the user semantic attribute model allows an important reduction of dimension and so reduce the expensiveness of user similarity computing. In (Ben Ticha et al., 2011), we have used the Fuzzy

CMean Algorithm on the movie *genre* attribute, we have obtained good performance because the user semantic attribute model has no missing values and all similarities between users were able to be computed. In (Ben Ticha et al., 2012), we have used the KMean clustering algorithm on the movie *origin* attribute. Because of the missing values in the user item rating matrix, we have proposed an algorithm for the initialization step of the KMean clustering using a movie origin ontology. We obtained good results compared to user-based CF but not as good as results for the *genre* attribute.

5 USER SEMANTIC MODEL FOR DEPENDENTS ATTRIBUTES

For a dependent attribute A, the set F_A of its features can be important and it augments with the increasing of the set of items S. In this paper, we present our solution to compute a user semantic attribute model for dependent attribute.

In addition to the formalism used in Section 4.2, we denote by F_{A_s} the set of features $f \in F_A$ associated to item s and by S_u the set of items $s \in S$ rated by user u. We define also, the rating function of user u as $\delta_u : s \in S_u \mapsto \delta_u(s) \in R$; and the Item Frequency Function of item s as $freq_s$ described in formula 3.

$$\forall s \in S, f \in F_A, freq_s(f) = \begin{cases} 1 \ if \ \ f \in F_{A_s}. \\ 0 \ otherwise. \end{cases} \quad (3)$$

The Frequency Item Matrix $F = (freq_s(f))_{s \in S \ and \ f \in F_A}$ is provided by computing $freq_s(f)$ for all items and all features. Table 1 provides an example of Frequency Item matrix for $S = \{i_1, i_2, i_3\}$ and $F_A = \{f_1, f_2, f_3, f_5\}$.

	f_1	f_2	f_3	f_4
i_1	0	1	0	1
i_2	0	0	1	1
i_3	1	1	0	1

Table 1: Example of Item Frequency Matrix

The building of user semantic attribute model consists of three steps:

1. Computing the TF-IDF measure on the Frequency Item Matrix F.
2. Reducing the dimension of feature space by applying a LAS.
3. Computing the user semantic attribute model on items in LSA space by using the Rocchio algorithm.

5.1 Computing the TF-IDF measure on the Frequency Item Matrix F

One of the best-known measures for specifying keyword weights in Information Retrieval is the TF-IDF (Term Frequency/Inverse Document Frequency) (Salton, 1989). It is a numerical statistic, which reflects how important a word is to a document in a corpus. In our case, we replace document by item and term by feature and compute TF-IDF on the Frequency Item Matrix F.

$$TF(f, s) = \frac{freq_s(f)}{max_j freq_s(j)} \quad (4)$$

where the maximum is computed over the $freq_s(j)$ of all features in F_{A_s} of item s.

The measure of Inverse Document Frequency (IDF) is usually defined as:

$$IDF(f) = \log \frac{|S|}{|S_f|} \quad (5)$$

where $|S_f|$ is the number of items assigned to feature f (ie $freq_s(f) \neq 0$). Thus, the TF-IDF of feature f for item s is defined as:

$$TFIDF(s, f) = TF(f, s) \times IDF(f) \quad (6)$$

In the flowing, we will denote by F_{TFIDF} the TF-IDF frequency matrix provided by computing TF-IDF measure on the item frequency matrix F, so $F_{TFIDF} = TFIDF(s, f)_{s \in S \ and \ f \in F_A}$.

5.2 Reducing Dimension of the TF-IDF Frequency matrix

For dependent attribute, the number of feature is correlated to the number of items. So it can be very elevated and even higher than the number of items. Thus, the semantic user attribute model can have dimension greater than the user rating matrix thereby aggravating the scalability problem. Therefore, in order to reduce the dimension of features space, we apply a LSA with rank k (see section 3) to the F_{TFIDF} matrix. The rank k is well below the number of features of attribute A ($k << |F_A|$). As shown in section 3, LSA uses a truncated SVD (see formula 2) keeping only the k largest singular values and their associated vectors. So, the rank-k approximation matrix of the F_{TFIDF} matrix is provided by formula 7.

$$F_{TFIDF} \approx I_{|S|,k} * \Sigma_{k,k} * V^t_{k,|F_A|} \quad (7)$$

The rows in I_k are the item vectors in LSA space and the rows in V are the feature vectors in LSA space. Thus, each item is represented in the LSA space by a set of k latent variables instead of the features of F_A. This, results in a much less sparse matrix and a reduced dimension of feature space.

5.3 Computing the User Semantic Attribute Model

Rocchio algorithm (Rocchio, 1971) is a relevance feedback procedure, which is used in information retrieval. It designed to produce improved query formulations following an initial retrieval operation. In a vector processing environment, both the stored information document D and the requests for information B can be represented as t-dimensional vectors $D = (d_1, d_2, ..., d_t)$ and $B = (b_1, b_2, ..., b_t)$. In each case, d_i and b_i represent the weight of term i in D and B respectively. A typical query-document similarity measure can then be computed as the inner product between corresponding vectors.

Rocchio showed in (Rocchio, 1971), that in a retrieval environment that uses inner product computations to assess the similarity between query and document vectors, the best query leading to the retrieval of many relevant items from a collection of documents is:

$$B_{opt} = \frac{1}{|R|} \sum_{R} \frac{D_i}{|D_i|} - \frac{1}{|NR|} \sum_{NR} \frac{D_i}{|D_i|} \qquad (8)$$

Where D_i represent document vectors, and $|D_i|$ is the corresponding Euclidean vector length; R is the set of relevant documents and NR is the set of non relevant documents.

We have applied the Rocchio formula (8) for computing the user semantic attribute profile of user u. We replace documents by items in S described in LSA space. Thus, the user semantic attribute model $Q_A(u)$ for user u and attribute A is given by formula 9.

$$Q_A(u) = \frac{1}{|R_u|} \sum_{R} \frac{s_i}{|s_i|} - \frac{1}{|NR_u|} \sum_{NR} \frac{s_i}{|s_i|} \qquad (9)$$

Where R_u is the set of relevant items of u. It is composed of all items in S_u having rating greater than the rating average of u ($\delta_u(s) \geq \overline{\delta_u}$). NR_u is the set of non relevant items of u. It is composed of all items in $S_u \setminus R_u$ ($(\delta_u(s) < \overline{\delta_u})$.

6 RECOMMENDATION

To provide recommendations for the active user u_a, we use the user-based CF algorithm (Resnick et al., 1994). User-Based CF predicts the rating value of active user u_a on non rated item $s \in S$ based on his or her nearest neighbors. The principle of the user-based CF consists of the following steps:

Similarity : compute the similarities between u_a and all others users. The similarity between two users u and v is equal to the cosine of the angle between $\vec{u}$ and $\vec{v}$ (see equation 10). The set of the nearest neighbors of u_a is equal to the B users with the highest similarity values. In the standard user-based CF algorithm, the users-items rating matrix $(\delta_u(s)_{(u \in U,\ s \in S)})$ is used to compute users' similarities. In our algorithm, the user semantic matrix Q is used instead for computing the similarities between users. As we have already mentioned, the matrix Q is the horizontal concatenation of user semantic attribute model Q_A for each relevant attribute A.

$$sim(u, v) = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \bullet \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|} \qquad (10)$$

Prediction : compute the prediction value $p(u_a, s) =$ of rating of user u_a on non rated item s. Prediction value is equal to a weighted aggregate of the b nearest neighbor ratings of user u_a (see equation 11).

$$p(u_a, s) = \overline{\delta_{u_a}} + \frac{\sum_{v \in V} sim(u_a, v)(\delta_v(s) - \overline{\delta_v})}{\sum_{v \in V} |sim(u_a, v)|} \qquad (11)$$

where V is the set of the B nearest neighbors (most similar users) to u_a that have rated item s. B can range anywhere from 1 to the number of all users.

Recommendation: recommend to the active user u_a a list $L(u)$ of the Top-N relevant items. Relevant items are those having predicted ratings greater than or equal a given threshold.

Although we apply a user-based CF for recommendation, our approach is also a model-based method because it is based on a new user model to provide ratings of active user on non rated items. Our approach resolves the scalability problem for several reasons. First, the building process of user semantic model is fully parallelizable (because the computing of user semantic attribute model is done in independent way for each other) and can be done off line. 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, so, the computing of similarities between users is less expensive than in the standard user-based CF. In addition, our approach allows inferring similarity between two users even when they have any co-rated items because the users-semantic matrix has less missing values than user item ratings matrix. Thus, our approach provides solution to the neighbor transitivity problem emanates from the sparse nature of the underlying data sets. In this problem, users with similar preferences may not be identified as such if they haven't any items rated in common.

7 PERFORMANCE STUDY

In this section, we study the performance of our algorithm, User Semantic Collaborative Filtering (*USCF* in plots), against the standards CF algorithms: User-Based CF (*UBCF* in the plot) (Resnick et al., 1994), and Item-Based CF (*IBCF* in the plot) (Sarwar et al.,

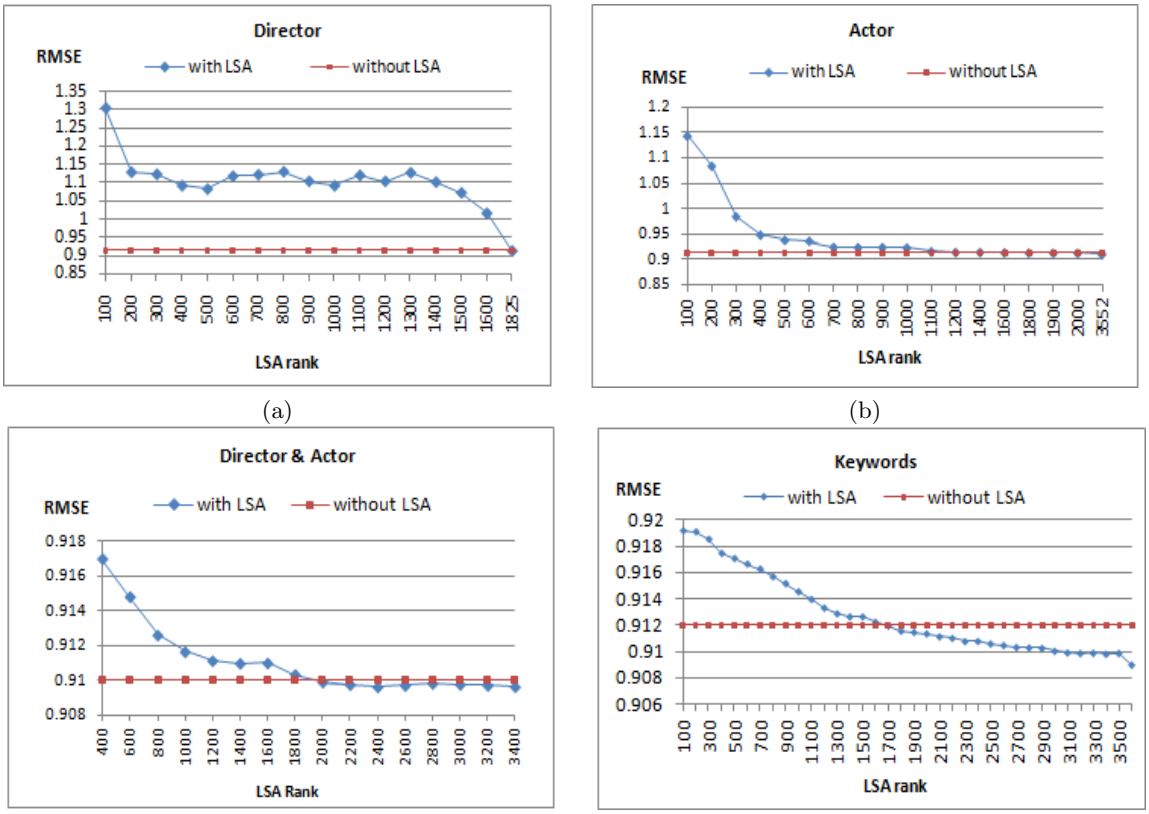


Figure 2: Impact of LSA on prediction accuracy of Rocchio algorithm.

2002); standard CB algorithm (*CB* in the plot) (Lops et al., 2011).

It should be noted that the building of User Semantic Attribute Model for the non dependent attributes *genre* and *origin* have been addressed respectively in previous works (Ben Ticha et al., 2011; Ben Ticha et al., 2012). Therefore, we will not detail the experiments conducted for these attributes in this paper.

7.1 Data set

We have experimented our approach on real data from the MovieLens1M data set (data set, 2014) of the MovieLens recommender system (MovieLens, 2014). The MovieLens1M provides the usage data set and contains 1,000,209 explicit ratings of approximately 3,900 movies made by 6,040 users. The ratings are user-provided star ratings, from 1 to 5 stars. For the semantic information of items, we use the HetRec 2011 data set (HetRec2011, 2011) that links the movies of MovieLens data set with their corresponding web pages at Internet Movie Database (IMDb) and Rotten Tomatoes movie review systems. We use *movie genre* and *movie origin* as non dependent attributes, *movie director*, *movie actor* and *movie keyword* as dependent attributes.

We have filtered the data by maintaining only users with at least 20 ratings and available features for all movies. After the filtering process, we obtain a data set with 6020 users, 3552 movies, 19 genres, 43 origins, 1825 directors, 4237 actors and 12367 keywords . The usage data set has been sorted by the timestamps, in ascending order, and has been divided into a training set (including the first 80% of all ratings) and a test set (the last 20% of all ratings). Thus, ratings of each user in test set have been assigned after those of training set.

7.2 Evaluation Metrics

Several metrics have been used to evaluate the prediction accuracy of recommendation algorithms (Herlocker et al., 2004; Shani and Gunawardana, 2011). There are mainly two classes of metrics. The first measure the ratings prediction accuracy and are used to evaluate the accuracy of algorithms predicting the rating values. The second measure the accuracy of recommendation and are used to evaluate algorithms that provide a Top-N list of relevant items. In this paper, we have used the widely accepted metrics for each class. Thus, for measuring the accuracy of rating prediction we have used the Root Mean Squared Error (RMSE)(see equation 12). RMSE computes the

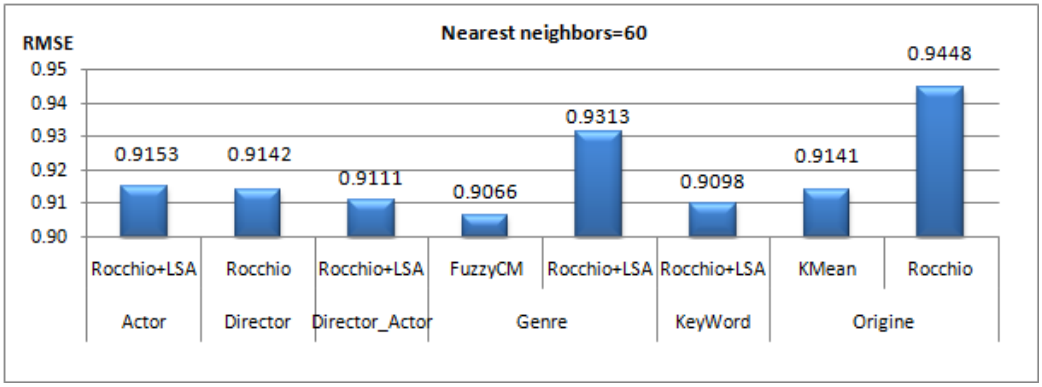


Figure 3: Impact of user semantic attribute algorithm on prediction accuracy.

square root of the average of the absolute difference between the predictions and true ratings in the test data set. RMSE disproportionally penalizes large errors. Lower the RMSE is, better is the prediction accuracy.

$$RMSE = \sqrt{\frac{\sum_{u,s}(p(u,s) - \delta_u(s))^2}{d}} \qquad (12)$$

Where d is the total number of ratings over all users, $p(u,s)$ is the predicted rating for user u on item s, and $\delta_u(s)$ is the actual rating for user u on item s in test data set.

For Top-N recommendation list, we have used the Precision metric. We have computed a Precision for each test user u in the test set by using formula 13. Then, the Precision for all users is equal to the average.

$$Precision(u) = \frac{|L(u) \cap R_{test}(u)|}{|L(u)|} \qquad (13)$$

where $L(u)$ is the list of N items recommended to user u. $R_{test}(u)$ is the set of relevant items for u in the test data set. In all our experiments, an item is relevant if its rating is greater than or equal to 4. The Precision measure describes the proportion of the recommendations were actually suitable for the user. Higher the Precision is, better is the recommendation accuracy.

7.3 Impact of LSA on Prediction Accuracy of Rocchio algorithm

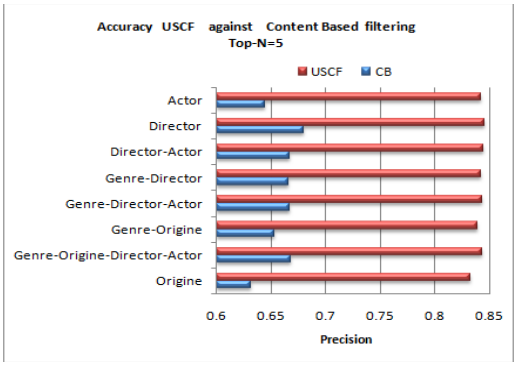
In Figure 2, the RMSE has been plotted with respect to the LSA rank. It compares the Rocchio approach *with* and *without* applying LSA (dimension reduction) on *director* attribute (Figure 2(a)), *actor* attribute (Figure 2(b)), combined attribute *director_actor* (Figure 2(c)) and *Keyword* attribute (Figure 2(d)). For attributes *director* and *actor*, the plots (Figures 2(a) and 2(b)) have the same look, the RMSE of Rocchio with LSA decreases until it reaches the RMSE value of Rocchio without LSA. So, LSA dimension reduction has no effect on improving the accuracy of Rocchio approach on *director* and *actor* attributes. The poor performance of these two attributes may be explained by the fact that their features are not highly correlated. Indeed, for the *director* attribute, for instance, the RMSE without reduction (1825 features) is equal to 0.9142 while the best value with LSA is equal to 1.017. For *actor* attribute, the best accuracy (RMSE=0.9153) of Rocchio with LSA is reached for LSA rank=1100, it is equal to the performance of Rocchio without LSA, with a dimension reduction about 75%.

However, for combined attributes *director_actor* (Figure 2(a)) and *Keyword* (Figure 2(d)) the LSA dimension reduction improves the accuracy of Rocchio approach. Because of features in *keyword* attribute are more correlated than in *actor* or *director* attribute, using LSA can potentially reduces the amount of noise associated with the semantic information. For *keyword* attribute, the best accuracy (RMSE= 0.909) is reached for LSA rank=3552 a reduction of about 71%. For rank equal to 1000, RMSE= 0.9145, so a dimension reduction about 91% for a loss of accuracy about 0.60% against the best accuracy of Keyword attribute.

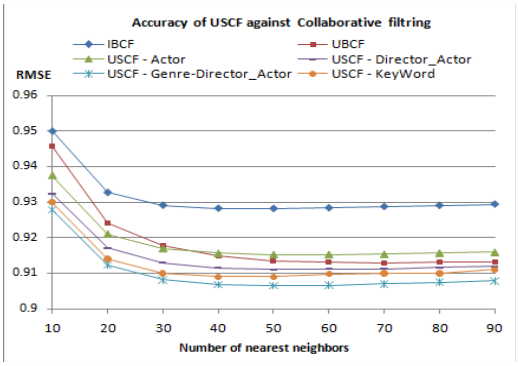
Although the LSA doesnt́ improve the accuracy, dimension reduction is significant. Thus, it allows to reduce the cost of users similarity computing, specially when the number of features is very high, as is the case of combined attributes *director_actor*.

7.4 Impact of Attribute Class on Prediction Accuracy

Figure 3 compares algorithms for building user semantic attribute model in term of RMSE. *Fuzzy C Mean* algorithm (FuzzyCM in plot) is a fuzzy clustering used for non dependent and multivalued attribute (here *genre*) and *KMean* algorithm (KMean in plot) is used on non dependent and mono valued attribute (here *origin*). Moreover, Rocchio algorithm (Rocchio in plot) is applied here for all attributes dependent and non dependent. For *origin* and *director* attributes, Rocchio without LSA provides best results than with dimension reduction. For *actor* attribute,



(a) Higher the Precision is, better is the recommendation



(b) Lower the RMSE is, better is the prediction

Figure 4: Evaluation of USCF against CB in terms of Precision (a) against standards CF in terms of RMSE (b).

LSA with rank equal to 1100 is performed, for *genre* attribute a factorization (LSA with rank equal to 18) is applied, for *keyword* attribute LSA with rank equal to 3400 is applied. Rocchio+LSA in plot means the Rocchio approach is performed with LSA. When analyzing this figure we note that, if we performed the *Rocchio* algorithm to non dependent attribute the performance compares unfavorably against the dependent attribute. Indeed, the best performance is achieved by *FuzzyCM* algorithm on *genre* attribute and the difference with Rocchio approach, even with factorization, is important (0.9066 for FuzzyCM and 0.9314 for Rocchio with LSA (rank=18)). This allows us to deduce that, using a suited algorithm for each attribute class provides best performance than applying the same algorithm for all attributes. Second, the *origin* attribute has the worst performance compared to the other three attributes and this for all algorithms. This is confirm our hypothesis that all attributes don't have the same relevance to users. The attribute *origin* can be less significant in the choice of users than the *genre*, *actor* or *director*, which is intuitively understandable.

7.5 Comparative results of USCF against CF and CB recommender system

Figure4 depicts the recommendation accuracy of USCF in contrast to those produced by pure CB (CB in plots) recommender system (Figure4(a)) using Precision metric to measure the recommendation accuracy; and standard Item-Based CF (IBCF) and User-Based CF (UBCF) (Figure4(b)). USCF-<*Attributes*> in plot means the list of relevant attributes involved in building the user semantic model Q. For each relevant attribute, the suited algorithm is applied. So, Fuzzy CMean for *genre*, KMean for *origin*, Rocchio with LSA (rank=1200) for combined attribute *director_actor* and Rocchio with LSA (rank=3400) for Keyword attribute. Pure CB algorithm exploits only item-content for recommendations. Thus, we have built an item-item similarity matrix based on Cosinus. Item is described by a Vector Space Model (VSM) composed by features of selected attributes as shown in Figure4(a). In Figure4(a), recommendations are computed for 60 nearest neighbors. We note that our algorithm USCF results in an overall improvement in accuracy against CB for all combinations of attributes. In Figure4(b), RMSE has been plotted with respect to the number of neighbors (similar users) for computing rating prediction (see section 6). In all cases, the RMSE converges between 50 and 60 neighbors, however, USCF results in an overall improvement in accuracy. In addition, the best performance is achieved by the combination *genre-director_actor*. This improvement can be explained by many reasons. First, taking into account the semantic profile of items in a CF recommendation process. Second, for non dependent attribute, User Semantic Attribute Model is built according to a collaborative principle; ratings of all users are used to compute the semantic profile of each user. Third, the choice of the attribute can have significant influence on improving the accuracy. Lastly, Users Semantic Model has few missing values, so, it allows inferring similarity between two given users even when they have any items rated in common.

8 CONCLUSION AND FUTURE WORK

The approach presented in this paper is a component of a global work, which the aim, is to semantically enhance collaborative Filtering recommendation and to resolve the scalability problem by reducing the dimension. For this purpose, we have designed a new hybridization technique, which predicts users' preferences for items based on their inferred preferences for semantic information. We have defined two classes of attributes: *dependent* and *non dependent* attribute, and presented a suited algorithm for each class for building user semantic attribute model.

The aim of this paper is to present our approach for building user semantic attribute model for dependent attribute. We have defined an algorithm based on Rocchio algorithm and have applied Latent Semantic Analysis (LSA) for dimension reduction. Our approach provides solutions to the scalability problem, and alleviates the data sparsity problem by reducing the dimensionality of data. The experimental results show that USCF algorithm improves the prediction accuracy compared to usage only approach (UBCF and IBCF) and Content based only approach. In addition, we have shown that applying Rocchio formula on non dependent attribute, decreases significantly the prediction accuracy compared to results obtained with machine learning algorithms. Furthermore, we have experimentally shown that all attributes don't have the same importance to users. Finally, experiments have shown that the combination of relevant attributes enhances the recommendations.

An interesting area of future work is to use machine learning techniques to infer relevant attributes. We will also study the extension of the user semantic model to non structured data in witch items are described by free text. Lastly, study how our approach can provide solution to the cold start problem in which new user has few ratings. Indeed, CF cannot provide recommendation because similarities with others users cannot be computed.

REFERENCES

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley.

Balabanović, M. and Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72.

Belkin, N. J. and Croft, W. B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38.

Ben Ticha, S., Roussanaly, A., and Boyer, A. (2011). User semantic model for hybrid recommender systems. In *The 1st Int. Conf. on Social Eco-Informatics - SOTICS*, pages 95–101, Barcelona, Espagne. IARIA.

Ben Ticha, S., Roussanaly, A., Boyer, A., and Bsaïes, K. (2012). User semantic preferences for collaborative recommendations. In Huemer, C. and Lops, P., editors, *E-Commerce and Web Technologies*, volume 123 of *Lecture Notes in Business Information Processing*, pages 203–211. Springer Berlin Heidelberg.

Ben Ticha, S., Roussanaly, A., Boyer, A., and Bsaïes, K. (2013). Feature frequency inverse user frequency for dependant attribute to enhance recommendations. In *The Third Int. Conf. on Social Eco-Informatics - SOTICS*, Lisbon, Portugal. IARIA.

Breese, J., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 43–52, Morgan Kaufmann, San Francisco. Madison, Wisconsin.

Burke, R. (2007). Hybrid web recommender systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer Berlin Heidelberg.

data set, M. (2014). http://grouplens.org/datasets/movielens/. July 2014.

Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

HetRec2011 (2011). In *2nd Int Workshop on Information Heterogeneity and Fusion in Recommender Systems*. The 5th ACM Conf. RecSys.

Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer US.

Manzato, M. G. (2012). Discovering latent factors from movies genres for enhanced recommendation. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 249–252, New York, NY, USA. ACM.

Miyahara, K. and Pazzani, M. J. (2000). Collaborative filtering with the simple bayesian classifier. In *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, PRICAI'00, pages 679–689, Berlin, Heidelberg. Springer-Verlag.

Mobasher, B., Jin, X., and Zhou, Y. (2004). Semantically enhanced collaborative filtering on the web.

In Berendt, B., Hotho, A., Mladenic, D., van Someren, M., Spiliopoulou, M., and Stumme, G., editors, *Web Mining: FromWeb to SemanticWeb*, volume 3209 of *Lecture Notes in Computer Science*, pages 57–76. Springer Berlin / Heidelberg.

Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA. ACM.

MovieLens (2014). www.movielens.org. July 2014.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, pages 325–341. Springer-Verlag, Berlin, Heidelberg.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *The 1994 ACM conference on Computer supported cooperative work*, page 175186.

Rocchio, J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The Smart Retrieval System - Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, Inc.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. In *Proceedings of the ACM WebKDD 2000 Web Mining for E-Commerce Workshop*.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT 02)*.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, number 4321 in Lecture Notes in Computer Science, pages 291–324. Springer Berlin Heidelberg.

Sen, S., Vig, J., and Riedl, J. (2009). Tagommenders: Connecting users to items through tags. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 671–680, New York, NY, USA. ACM.

Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 257–297. Springer US.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2.

Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., and Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 114–121, New York, NY, USA. ACM.