

Item Weighting Techniques for Collaborative Filtering

Linas Baltrunas and Francesco Ricci

Abstract. Collaborative Filtering (CF) recommender systems generate rating predictions for a target user by exploiting the ratings of similar users. Therefore, the computation of user-to-user similarity is an important element in CF; it is used in the neighborhood formation and rating prediction steps. In this paper we investigate the role of item weighting techniques. An item weight provides a measure of the importance of an item for predicting the rating of another item and it is computed as a correlation coefficient between the two items' rating vectors. In this paper we analyze a wide range of item weighting schemas. Moreover, we introduce an item filtering approach, based on item weighting, that works by discarding in the user-to-user similarity computation the items with the smallest weights. We assume that the items with smallest weights are the least useful for generating the prediction. We have evaluated the proposed methods using two datasets (MovieLens and Yahoo!) and identified the conditions for their best application in CF.

1 Introduction

Recommender systems are powerful tools helping on-line users to tame information overload and providing personalized recommendations on various types of products and services [20]. Collaborative Filtering (CF) is a successful recommendation technique which automates the so-called “word-of-mouth” social strategy. When generating a recommendation for a target user, CF analyzes the experiences/evaluations of similar users that are modeled as vectors of item ratings, and it recommends the items that the similar users liked the most [24]. The user-to-user similarity is usually determined by a pair-wise comparison of *all* the available ratings explicitly expressed by the users about items in the product catalog. The assumption is that two highly

Linas Baltrunas and Francesco Ricci
Free University of Bozen-Bolzano
Domeninkanerplatz 3, Bozen, Italy
e-mail: {lbaltrunas, fricci}@unibz.it

correlated users will have similar tastes also on items they have not rated yet. It is therefore evident that user-to-user similarity is a core computation step in CF [2].

Note that in this paper we are focusing on user-to-user CF, as opposed to item-to-item CF. The latter computes the rating prediction for a target user and item pair as a weighted average of the ratings the user gave to *similar items*, and therefore user-to-user similarity is not exploited.

In CF, user-to-user similarity computation is used in the neighborhood formation and in the final rating prediction. As we noted above, two users are considered similar if they have expressed similar opinions (ratings) on a set of co-rated items. In standard CF each of the co-rated items contributes equally to the similarity. But it seems natural to conjecture that some items could be more important than others when computing the user-to-user similarity. For example, if the movie “Dead on Arrival” gets the lowest possible rating from all the users, then it should not be very useful in understanding whether two users have similar preferences. These type of items, i.e., with very similar ratings in the users’ population, should contribute less than others or could be even discarded from the similarity computation. Moreover, additional information about the *content* of the items could potentially improve the performance of CF. For example, consider two users who have very similar tastes for action movies but very different tastes for dramas and documentary movies. In general, these users would have a small correlation when comparing all their co-rated items. However, when considering only the action movies the correlation would be higher. When predicting a rating for an action movie, co-rated action movies could be given a higher importance (weight).

In this paper we propose to address the previously mentioned issues by using item weighting, i.e., by extending the user-to-user similarity definition so that it is possible to *weight* the influence of each item. This is not an easy task. There are many sources of information potentially useful for computation of item weights. Moreover, it is not clear how this information can be transferred into item weights. For instance, it is questionable whether it would be better to consider rating data statistics, or to exploit information coming from external sources, such as item descriptors or contextual information. We also need to determine how strongly the additional information should influence the similarity. Finally, we observe that after having found good information sources (for determining the importance of items), there is no well-accepted way (formula) to extend the standard user-to-user correlation measure in order to take into account these weights.

In this paper we address the issues mentioned above, and in particular:

- We analyze a wide range of weighting schemas — some new and some already used in previous research. These are based on various types of information; one group of methods explores the data dependencies between item ratings, whereas another tries to exploit the information contained in the descriptions of the items.
- We empirically investigate three approaches to incorporate the computed weights into the user-to-user correlation measure and we provide an analysis of their behavior. We also conduct some experiments aimed at understanding to what extent the additional information contained in the weights should modify the prediction formula, compared to the standard similarity metric.

- Additionally, we analyze the behavior of an item filtering method based on the computed weights. This study evaluates whether it is possible to improve the prediction accuracy by using just a small subset of items, i.e., those that are very relevant to the target item.

In this paper we show that item weighting can improve the accuracy of the baseline CF (for both datasets), for all the neighborhood sizes, and for all the error measures we used. This improvement is obtained by a particular weighting method based on Singular Value Decomposition (SVD-PCC) of the rating matrix. The other weighting schemas did not uniformly improve the baseline, however, in some situations they can even produce larger improvements over the baseline than SVD-PCC. We also show that the largest reduction of the mean absolute error was achieved by using item filtering together with a particular weighting method but at the cost of sacrificing coverage. We also identified a weight incorporation method that uniformly performs better than the others.

The rest of the paper is structured as follows. Section 2 gives an overview of the state of the art in item weighting and item filtering. In Section 3 we introduce our definition of a user-to-user similarity that incorporates item weighting. There, we explain and motivate different weighting schemas (Subsection 3.1), and different approaches to integrate the weights into the similarity measure (Subsection 3.2). In Section 4 we introduce our new approach to item filtering. Section 5 provides the experimental evaluation and the discussion of proposed methods. Section 6 summarizes the work presented in this paper and draws some conclusions.

2 Related Work

CF can be seen as an instance-based learning approach where users are instances described by their feature vectors, and the product (item) ratings play the role of feature values. Hence, the rationale for the application of feature weighting/selection techniques in CF is that some items may be more important than others in the similarity computation [8].

Feature weighting is a well studied problem in Machine Learning, and in Instance-Based Learning in particular [17]. A number of studies reported accuracy improvements when features are weighted in the instance-to-instance distance computation ([26, 3] offer an excellent survey). In the context of CF feature weighting can be called item weighting, since features of a user are actually item ratings. The huge amount of items and the data sparsity makes most of these classical methods inefficient.

Item weighting and item selection have not been widely studied in CF, and only few researchers tried to apply these methods. In [8], the authors adapted the idea of inverse document frequency, a popular Information Retrieval measure [21], to item weighting in CF. The key idea of their approach, which is called Inverse User Frequency, is that universally liked and known items do not give much information about the true user preferences. Therefore, the weights for such commonly rated items should be decreased. This approach showed better performances than the

unweighted version. The same idea of decreasing the weights of commonly liked items was implemented by using variance weighting [12]. Here, items with bigger variance are supposed to distinguish better the taste of the users and, therefore, they receive larger weights in the user-to-user similarity formula. Yet their results were not encouraging and the authors report just a small increase of the Mean Absolute Error (MAE) with respect to the non-weighted version.

In [28] Information Theoretical approaches for item weighting were introduced. The authors showed that Mutual Information and Entropy based methods perform better than Inverse User Frequency and standard CF. Moreover, Mutual Information obtained a 4.5% accuracy improvement and performed better (even when trained on a small number of instances) than the standard CF. They also reported that Inverse User Frequency, differently from [8], decreased the accuracy. [14] presents another automatic weighting schema for CF systems. This method tries to find weights for different items that bring each user even closer to the similar users and farther from dissimilar users. That method uses ideas from model-based approaches and can decrease MAE. As it is clear from this short review, the quoted papers offer contradicting results. For this reason we decided to test ourselves the performance of some of these methods on two different data sets.

In addition to feature weighting, feature selection (filtering) algorithms have also been very popular in Machine Learning [26, 15, 16], and they are now receiving new interest because of their exploitation in Information Retrieval ranking methods based on learning [19, 10]. In CF, there have been just a couple of attempts to use item selection. An interesting approach to item selection was proposed in [1]. To generate a recommendation the authors proposed to use only the items (ratings) that were experienced in the same context of the target item prediction. The authors made item selection only in the subset of the ratings where cross-validation showed improvement in the accuracy. But, due to sparsity and small size of their data, only a small improvement over the baseline CF method was obtained.

In our previous work [4] we showed that the straightforward application of item selection methods based on the *filter* approach [26] fails because of the high sparsity of the data. Besides, *wrapper* methods seem not to be applicable because of the large number of features and therefore the large complexity of evaluating the exponential number of possible subsets of the features. Therefore, in [5] we developed a new filter-based approach that tries to overcome these problems. The item selection is made according to a pre-computed set of weights and considers only the items that are present in the profiles of both users whose similarity is sought. That approach improved all the error measures we used [5]. In this paper we evaluate a simpler approach that does not depend on the pair of users whose similarity is sought. While computing user-to-user similarity It simply tries to discard the items with lowest weights.

3 Weighted Similarity

In order to use item weighting in user-to-user similarity we first compute the item weights using a weighting schema, and then we incorporate these weights into the

standard user-to-user similarity measure. To compute the item weights we use a wide range of weighting schemas that are described in Subsection 3.1. Each weighting schema computes a real valued $n \times n$ weight matrix W (n is the number of items). An entry w_{ij} is the weight (importance) of item i for predicting the ratings of item j (for all the target users). In order to be able to compare the behavior of different weighting schemas, in this paper we normalized the weights to range in $[0, 1]$.

The role of the target item j needs some further explanation. In fact this gives to an item weighting schema the flexibility to assign to each predictive item a weight that depends on the target item whose rating predictions are sought. Hence, the weights used for predicting the ratings for item j (for all the users) can be different from those used in the predictions for another item j' . In this way, we encode in a weight how much two items are correlated, and what role an item should play when a prediction is sought for the second one. Without such a flexibility we could only express more general knowledge about the importance of an item for all the rating predictions. In fact, an example of this approach is provided by the variance weighting method that is explained in Subsection 3.1. Finally, given a weighting schema, there are several ways to integrate the weights into an (unweighted) similarity measure. These are methods described in Subsection 3.2.

3.1 Weighting Schemas

Item weighting tries to estimate how much a particular item is important for computing the rating predictions for a target item. In CF, item weights can be learned while exploring training data consisting of user ratings or using additional information associated with the items, i.e., their descriptions. Based on this observation we can partition weighting methods in two groups. The methods in the first group determine the item weights using statistical approaches and are aimed at estimating statistical properties of the ratings and the dependencies between items. For instance, the variance of the item ratings (provided by a users' population) belongs to the first group of methods. The second group of methods uses item descriptions to estimate item dependencies and finally infer the weights. For instance, the similarity between item descriptions can be used to infer item-to-item dependencies and these can be converted into weights.

In the rest of this paper we will consider the following item weighting approaches: Random, Variance, Mutual Information, Genre, Pearson Correlation Coefficient (PCC), and PCC computed on low dimensional item representations. Let us now precisely define these methods.

Random. The first method is used as a reference for comparing different approaches. It assigns to each predictive (i) and target (j) item combination a random weight sampled from the uniform distribution in $[0, 1]$: $w_{ij} = \text{random}([0, 1])$

Variance. The variance method is based on an observation originally made in [12]. It gives higher weights to items with higher variance among the ratings provided by the users to that item:

$$w_{ij} = \frac{\sum_u (v_{ui} - \bar{v}_i)^2}{\#users\ who\ rated\ i}$$

Here the sum runs over all the users who rated the item i , v_{ui} is the rating given by user u to item i , and \bar{v}_i is the mean of the ratings provided by all the users to the item i . The variance feature weighting method uses only information on the item whose ratings have to be predicted. All the methods that we are presenting next, explore relations between predictive items and the target item.

IPCC. The first method in this group uses Pearson Correlation Coefficients (PCC) as a measure of the dependency between two vectors representing the ratings of all users for two items:

$$w_{ij} = \frac{\sum_u (v_{ui} - \bar{v}_i)(v_{uj} - \bar{v}_j)}{\sqrt{\sum_u (v_{ui} - \bar{v}_i)^2 \sum_u (v_{uj} - \bar{v}_j)^2}}$$

Here the index u runs over all the users that have rated both items i and j , and \bar{v}_i is the mean of the ratings on item i . We observe that IPCC (Item Pearson Correlation Coefficients) builds a symmetric weight matrix W , i.e., the importance of the item i in the prediction of the ratings for the item j is equal to the importance of the item j when it is used to predict the ratings for the item i .

Mutual Information. Mutual Information (MI) measures the information that a random variable provides to the knowledge of another. In CF, following [28], we compute the mutual dependency between the target item variable and the predictive item variable (the values are the ratings). The Mutual Information of two random variables X and Y is defined as:

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

The above equation can be transformed into $I(X;Y) = H(X) + H(Y) - H(X,Y)$, where $H(X)$ denotes the entropy of X , and $H(X,Y)$ is the joint entropy of X and Y . Using the above formula Mutual Information computes the weights as follow:

$$w_{ij} = - \sum_{r=1}^5 p(v_i = r) \log p(v_i = r) - \sum_{r=1}^5 p(v_j = r) \log p(v_j = r) \\ + \sum_{r'=1}^5 \sum_{r''=1}^5 p(v_i = r', v_i = r'') \log p(v_i = r', v_i = r'')$$

Here the probabilities are estimated with frequency counts in the training dataset, hence for instance $p(v_i = r) = \frac{\#users\ who\ rated\ i\ as\ r}{\#users\ who\ rated\ i}$ estimates the probability that the item i is rated with value r . We observe that r is running through all rating values, and in our data sets this is equal to 5.

SVD-PCC. The methods described previously, i.e., IPCC and Mutual Information, compute a form of statistical dependency between two items. Note that such

statistics between two items are computed using the pairs of ratings that all the users expressed for *both* items. When the data is extremely sparse there can be only few users who have rated both items, and the computation of the statistical dependency based on this incomplete information could be inaccurate. Moreover, the fact that two items were not co-rated by the users should not imply that they are not similar [7]. In order to avoid such problems we decided to reduce the dimensionality of the ratings matrix and compute item-to-item correlation in the transformed space rather than on the original raw data. This approach is based on Latent Semantic Analysis (LSA) [9], originally used for analyzing the relationships between a set of documents and the contained terms. LSA is now widely applied to CF to generate accurate rating predictions, and the user-to-user similarity is computed in a space with lower dimension than the number of items and users [22]. LSA uses Singular Value Decomposition (SVD), a well-known matrix factorization technique (see [11] for a reference).

SVD factors a rating matrix $R = (v_{ij})$, of size $m \times n$ (m users and n items) into three matrixes as follows: $R_{m \times n} = U_{m \times r} \cdot S_{r \times r} \cdot V_{r \times n}^T$ where, U and V are two orthogonal matrices, r is the rank of the matrix R and S is a diagonal matrix having all the singular values of matrix R as its diagonal entries. Note, that all the singular values are positive and ordered in descending order. Selecting the $k < r$ largest singular values, and their corresponding singular vectors U and V , one gets the rank k approximation $R_{m \times n} = U_{m \times k} \cdot S_{k \times k} \cdot V_{k \times n}^T$ of the rating matrix R .

Each column of the matrix $I = V_{k \times n}^T$ gives a dense representation of the corresponding item in a k dimensional space. The choice of k is an open issue in the factor analytic literature [9] and it is typically application dependent. We set $k = 40$ in our experiments (see Section 5 for more details). To compute the similarity of item i and item j we compute the PCC between two columns of I :

$$w_{ij} = \frac{\sum_k (I_{ki} - \bar{I}_i)(I_{kj} - \bar{I}_j)}{\sqrt{\sum_k (I_{ki} - \bar{I}_i)^2 \sum_k (I_{kj} - \bar{I}_j)^2}}$$

where the sum runs over the row index of the matrix I and \bar{I}_j denotes the average value of the j -th column.

Genre Weighting. All the previous methods exploit statistics of the users' rating data to determine item weights. The last method that we present here computes the weights using descriptions of the items. In the movie recommendation data sets, which we are using for our experiments, the movies are tagged with movie genres. Hence, we make the assumption that the larger the number of common genres, the higher is the dependency. Hence, the weight of the predictive item i for predicting the ratings of the target item j is given by:

$$w_{ij} = \frac{\# \text{comon genres of items } i \text{ and } j}{\# \text{genres}}$$

Genre weighting is related to the methods presented in [6], where item description information is used to selectively choose the items to be used in the user-to-user similarity.

3.2 Weight Incorporation

The two most popular measures of user-to-user similarity are: the cosine of the angle, formed by the rating vector of the two users; and Pearson Correlation Coefficient (PCC). PCC is preferred when data contains only positive ratings, and has been shown to produce better results in such cases [8]. The PCC between the users x and y is defined as:

$$PCC(x, y) = \frac{\sum_i (v_{xi} - \bar{v}_x)(v_{yi} - \bar{v}_y)}{\sqrt{\sum_i (v_{xi} - \bar{v}_x)^2 \sum_i (v_{yi} - \bar{v}_y)^2}}$$

where v_{xi} denote the rating given by the user x to the item i , and \bar{v}_x is the mean of the ratings of the user x . The sum runs over all the items i that both users have rated.

As we mentioned in the Introduction, the motivation for using weights in the user-to-user similarity is to improve the prediction accuracy. This approach has been used in instance-based learning to produce weighted the Euclidean metric [3]. We must observe that the effect of the weights on the Euclidean metric has a clear (geometric) interpretation: the features with larger weights produce a greater impact on the similarity. For PCC the situation is not as clear as for the Euclidean metric. Because of the more complicated nature of PCC, there exists no single well accepted approach to extend it with weights. In our study we analyzed three different versions of *weighted* PCC. The first version which we call normalized Weighted PCC ($WPCC_{norm}$) makes a natural extension of the unweighted PCC:

$$WPCC_{norm}(x, y, j) = \frac{\sum_i w_{ij}(v_{xi} - \bar{v}_x)(v_{yi} - \bar{v}_y)}{\sqrt{\sum_i w_{ij}(v_{xi} - \bar{v}_x)^2 \sum_i w_{ij}(v_{yi} - \bar{v}_y)^2}}$$

where j denotes the target item of the rating prediction, and w_{ij} is the weight of the (predictive) item i when making a prediction for j . Here \bar{v}_x is the average rating of the user x .

The second approach we consider is used by SASTM[23] and it extends the previous basic approach by replacing the standard average of the user ratings by a weight-normalized version:

$$WPCC_{wavg}(x, y, j) = \frac{\sum_i w_{ij}(v_{xi} - \hat{v}_x)(v_{yi} - \hat{v}_y)}{\sqrt{\sum_i w_{ij}(v_{xi} - \hat{v}_x)^2 \sum_i w_{ij}(v_{yi} - \hat{v}_y)^2}}$$

where $\hat{v}_x = \frac{\sum_i w_{ij}x_i}{\sum_i w_{ij}}$ and we note that this new normalized average depends on the target item j . The idea here is that the *weighted* average of the user ratings should

better estimate the user average. Note that we compute the average rating using all the ratings of the user, rather than just overlapping items.

The third method that we consider in this paper was used in [14], and it differs from the previous ones as it does not normalize the similarity metric:

$$WPCC_{no-norm}(x, y, j) = \frac{\sum_i w_{ij} (v_{xi} - \bar{v}_x)(v_{yi} - \bar{v}_y)}{\sqrt{\sum_i (v_{xi} - \bar{v}_x)^2 \sum_i (v_{yi} - \bar{v}_y)^2}}$$

In this approach, the users who co-rated highly weighted items would have higher correlation. Because of the lack of the normalization this correlation measure depends on the absolute values of weights, rather than on the relative differences between the weights. Therefore, the similarity that is computed on the items with small weights will be small.

Further discussion on the different weight incorporation methods can be found in the experimental part of this paper.

4 Item Selection

In item selection instead of using the precise values of the weights for tuning the similarity function, a simpler decision is taken to either use or not to use an item, based on the value of its weight. In fact, item selection could be seen as a particular case of item weighting, where just binary weights are used.

In fact, finding the optimal subset of items to be used in the user-to-user similarity would require an extensive search through the space of all the subsets of the items [15]. Applying this to a recommender system scenario would require to conduct a search procedure for every target item (the item playing the role of the class to be predicted) and for a large number of subsets of the predictive items. This is clearly extremely expensive, and therefore, we propose to use a more parsimonious approach (filter method) [15]. It uses the information provided by an item weighting method to select, for each target item an appropriate set of predictive items. Hence, first we compute the item weights using one of the weighting schemas described above, and then we select only the relevant items, i.e., the items with the largest weights, for any given target item.

Users tend to rate a small part of the items in a data set. Selecting a small number of items for the similarity computation further reduces the amount of information used to compute it. Therefore, in [5] we proposed to select the items that are present in the profiles of both users, whose similarity must be computed. We showed that this approach can increase the prediction accuracy (for many different error measures), and can use a smaller number of nearest neighbors.

In this paper we propose and evaluate a different approach. Instead of performing a dynamic item selection which depends on the target user we simply filter out (and never consider in the similarity computation) the items with the smallest weights.

In other words, if a weight is less than a given threshold, we set it to zero. Note, that instead of using binary weights as we did in [5], we still use in the user-to-user similarity computation the weights in $[0, 1]$. Moreover, the set of items used in a prediction depends only on the target item and it is the same for all the users.

5 Experimental Evaluation

In this section we evaluate the item weighting and item selection methods presented so far. We observe that to generate the prediction, one of the three variations of *WPCC* was used in computing the nearest neighbors of the target users and also in the prediction step:

$$v_{xj}^* = \bar{v}_x + \frac{\sum_{y \in N(k,x,j)} WPCC(x,y,j) \times (v_{yj} - \bar{v}_y)}{\sum_{y \in N(k,x,j)} |WPCC(x,y,j)|}$$

Here the sum runs over the k -nearest neighbors $N(k,x,j)$ of the user x . Note that the neighbors depend on the target item j because the user-to-user similarity depends on the target item. In our implementation of CF, as previously done in other studies, when making a rating prediction for a user x we take into account only the neighbors with at least a minimum number of co-rated items with the target user (six in our case) [6].

In the experiments, we used two data sets with ratings in $\{1, 2, 3, 4, 5\}$.¹ The MovieLens [18] data set contains 100K ratings, for 1682 movies by 943 users, who have rated 20 and more items. The data sparsity of this dataset is 96%. The Yahoo! [27] Webscope movies data set contains 221K ratings, for 11915 movies by 7642 users. The data sparsity of this dataset is much higher, 99.7%. To evaluate the described methods we used holdout validation, where both datasets were divided into train (80%) and test (20%) subsets. We used the train data to learn the weights and also to generate the prediction for the test ratings. Because of the high computational complexity, when computing weights, we were not able to perform a multi-fold cross-validation.

To measure the accuracy we used Mean Absolute Error (MAE), coverage and F_1 (in the rest of the paper denoted as F) measures [13]. To compute F, we considered an item worth recommending only if the user rated it as 4 or 5. Coverage is the fraction of the ratings in the test set for which predictions can be made. Note that when computing MAE we followed the standard practice and did not take into account the ratings that an algorithm was not able to predict.

To compute Singular Value Decomposition we used the SVDLIB [25] library and selected $k = 40$ biggest singular values. As mentioned earlier, the selection of k is a problematic issue. As stated above, in our experimental setting the parameter selection using cross-validation was too expensive. Selecting a too small k can lead to a big loss of information, however, selecting too big k could result in not removing

¹ We want to thank both Grouplens and Yahoo! for making their data sets available.

the noise present in the data. Computing similarity in the higher dimensional space might also not be effective because of the “curse of dimensionality”. Our selection was based on some trials and the analysis of the outcome of previous reports [9, 7].

5.1 Weight Incorporation

We started our experiments by evaluating the performance of the three weight incorporation methods described in Subsection 3.2. In a first experiment we used all the weighting schemas and computed F and MAE for the three weight incorporation methods. For lack of space we focus here only on SVD-PCC weighting. The comparison of three weight incorporation methods for the two considered datasets, using the *SVD-PCC* weighting schema, is depicted in Figure 1.

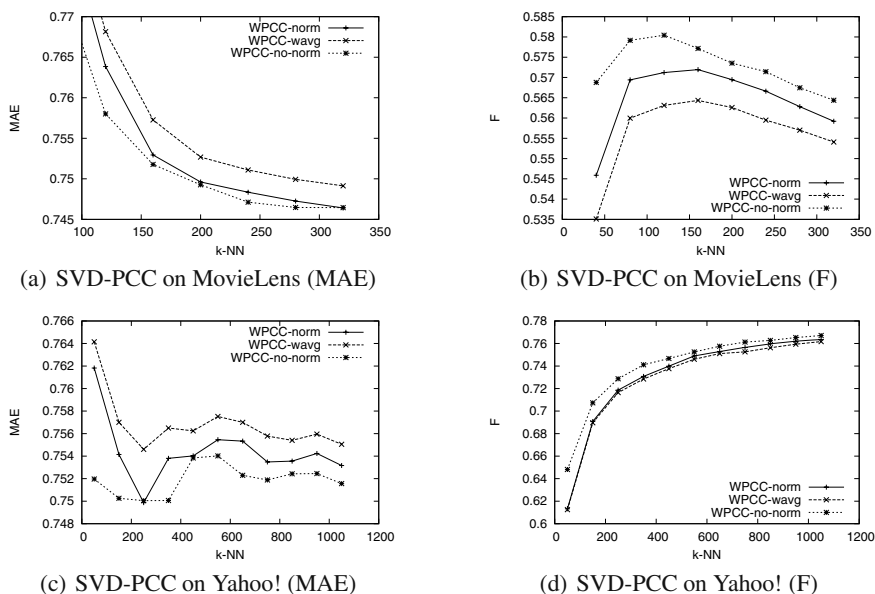


Fig. 1 Performance of weighted PCC measures

We observe here that most of the time $WPCC_{no-norm}$ performs better than the other methods independently of the data set, and independently of the error measure we used. Because of lack of space we do not show the results of this experiment with the other weighting schemas but they confirm this result, i.e., $WPCC_{no-norm}$ always performs better than the other two.

This is quite surprising, as $WPCC_{norm}$ or $WPCC_{WAVG}$ are used in the majority of the literature and software packages [23, 28]. In fact, because of the complicated behavior of PCC such results are hard to interpret. Our explanation is that $WPCC_{no-norm}$ gives a higher absolute correlation to a user who has expressed

ratings on more correlated items to a target item. For example, consider the following three users with ratings:

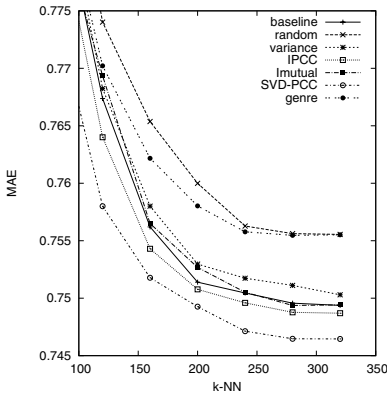
w_{ti}	0.8	0.7	0.6	0.5
	i_1	i_2	i_3	i_4
u_1	?	?	1	5
u_2	2	4	2	4
u_3	1	5	?	?

Here the first line of the table gives the item weights which are used in the similarity computation. Computing the user-to-user similarity on this toy example gives an insight into the nature of the different correlations. We have the following: $WPCC_{norm}(u_1, u_2) = 1$, $WPCC_{norm}(u_3, u_2) = 1$, $WPCC_{no-norm}(u_1, u_2) = 0.55$, $WPCC_{no-norm}(u_3, u_2) = 0.75$. One can see that $WPCC_{norm}$ takes into account only the relative size of the weights that are used in the similarity computation and not their absolute value. In fact, the similarity of u_1 and u_2 is equal to the similarity of u_3 and u_2 , even if the weights used for u_1 and u_2 are smaller. Therefore, computing the user-to-user similarity on highly correlated items (to a target item) or weakly correlated items does not change the final similarity and it remains equal to 1. On the contrary, $WPCC_{no-norm}$ takes into account the absolute values of the weights. This leads to a higher user-to-user correlation if computed on the item ratings which have bigger weights, i.e., are more correlated with the target item. Therefore, the users whose correlation is computed on the items with small weights will likely not be in the target user neighborhood.

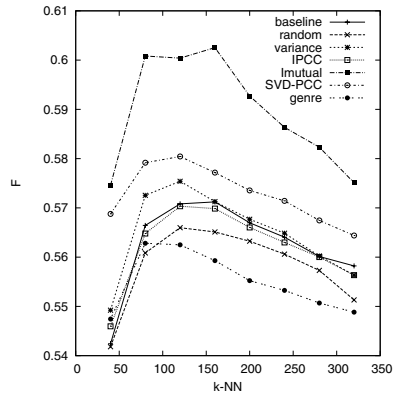
5.2 Evaluating Weighting Schemas

The second experiment evaluates all the weighting schemas described in Subsection 3.1 using $WPCC_{no-norm}$ as the weights incorporation method. In Figure 2 the performance of all these approaches varying the number of nearest neighbors are shown for the two considered data sets. Here, the baseline method uses standard (unweighted) PCC. The performance is shown while varying the number of nearest neighbors.

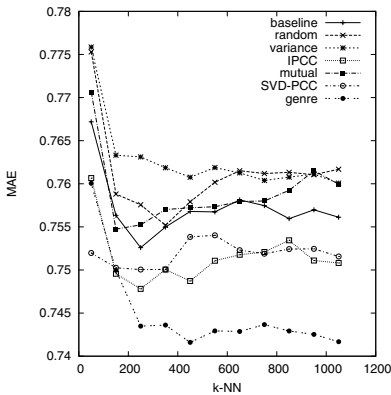
It can be seen that SVD-PCC improves MAE and F of the baseline prediction for both data sets and for all the neighborhood sizes. The situation with other weighting schemas is not so clear. For example, IPCC can reduce MAE, especially on the Yahoo! data set, and sometimes outperforms the SVD-PCC schema. The improvements of IPCC over the baseline method are smaller for the MovieLens dataset (considering MAE), however, this is achieved for all the neighborhood sizes. Surprisingly, the situation is different for F measure. While SVD-PCC still outperforms the baseline method, IPCC performs similarly to or even worse than the baseline. This is an interesting observation since IPCC is similar to SVD-PCC; the only difference is that SVD-PCC computes the correlation on a lower dimension item representation rather than on the original raw data.



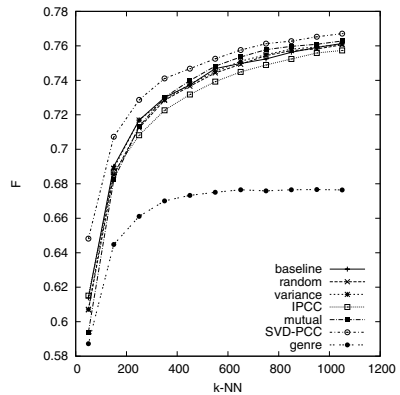
(a) MAE for MovieLens



(b) F for MovieLens



(c) MAE for Yahoo!



(d) F for Yahoo!

Fig. 2 Performance of Item weighting using $W_{PCC_{no-norm}}$ on two data sets

Another interesting behavior can be observed for the weighting schema based on Mutual Information for the MovieLens dataset (called “mutual” in all the Figures). This method performs similarly to the baseline with respect to MAE, however it gets a considerable improvement of F measure. After investigating the method further, we concluded that the increase of the F measure is due to a *large* increase of recall (up to 10%) at the cost of a small decrease in precision. This big improvement of recall is not clear and requires further investigation. Moreover, Mutual Information performs similarly to the baseline method for the Yahoo! data set. This result is different from the one reported in [28] where authors observed a big reduction of the MAE. Note that in [28] authors used EachMovie data set and trained the weights on a small random subset of the users. We guess that the differences could also be explained by the unstable behavior of Mutual Information weighting method. Variance weighting in general does not improve baseline CF, which confirms the

results of [12]. This method improves F measure, but it also increases MAE error. As expected, the random weighting schema always decreases the performance of the baseline CF.

Another interesting result can be observed for the genre weighting schema. In the MovieLens dataset it performs even worse than random item weighting and in fact it is the worst performing method. On the contrary, in the Yahoo! data set we have observed a significant reduction of MAE compared to the baseline method. It can be explained by analyzing the way the genre weights are computed. It is often the case that two movies do not share a single genre. Therefore, the genre overlap is 0, making the normalized weights also equal to 0. Hence, when using the genre weighting schema we perform a lot of item filtering together with item weighting.

5.3 Evaluation of Item Filtering

As explained in Section 4, in item filtering we set to zero all weights that are smaller than a given threshold. Our conjecture is that in this way we can reduce MAE by sacrificing the coverage. In Figure 3 the performance of item filtering, with SVD-PCC weighting schema, for all three possible weight incorporation methods is shown. We measure MAE and coverage for different filtering thresholds both for MovieLens and Yahoo! datasets. We fixed the neighborhood size for the MovieLens and Yahoo! datasets to be 200 and 300 respectively. For these neighborhood sizes the weighting methods have good MAE without decreasing F measure.

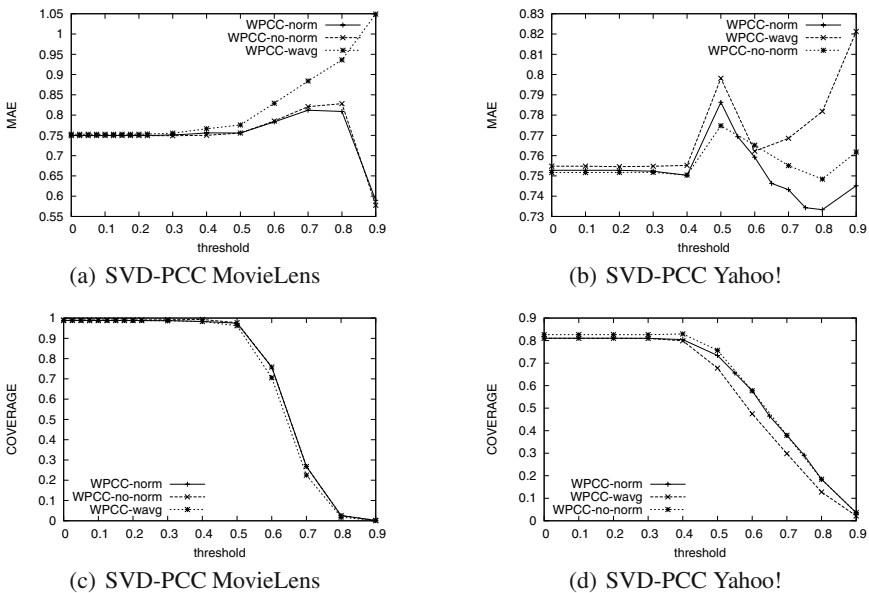


Fig. 3 Performance of item filtering

Note that some changes in the performance of item filtering can be detected only when the applied threshold is larger than 0.3. For the Yahoo! data set the normalized (in $[0, 1]$) weights are distributed as shown in Figure 4(a). In fact, only 8% of the weights have a value smaller than 0.3 and therefore filtering out these weights has almost no influence on the performance of the algorithms. Conversely, excluding items with weights larger than 0.3 decreases the performance. However, when most of the items are excluded (threshold equal to 0.9 for MovieLens and 0.8 for Yahoo!), the CF algorithms can make a prediction just for a very small number of items (low coverage), and MAE decreases. Hence, using item filtering with the SVD-PCC weighting schema, we can make more accurate predictions only for a very small number of items. In conclusion, using SVD-PCC, one can increase accuracy only by discarding most of the items and sacrificing the coverage.

5.4 Increasing the Influence of the Weights

In the previous evaluation of item weighting we used three different methods for weight incorporation into PCC. One of these approaches improved the performance of CF over the baseline method. In the experiments we are illustrating now, we tried to evaluate if amplifying the importance of the weights it is possible to obtain some improvement. Raising each weight to a power higher than one would make a bigger relative separation between weights. We want to evaluate if this larger separation of the weights can improve the accuracy of the prediction.

The Probability Density Function (PDF) of the weights distribution is shown in Figure 4. Here the weights computed by PCC-SVD for the Yahoo! dataset raised to different powers from 1 to 3 (x axis) are shown. Subfigure 4(a) shows the original (normalized) weight distribution, where most of the weights are distributed in the range $[0.2, 0.8]$. Subfigure 4(b) shows the PDF of the same weights were each weight was raised to the power of 2. In this case, weights which are close to 1 marginally decrease their value, whereas smaller weights are more strongly pushed towards 0. Therefore, we get a bigger relative separation between large weights (close to 1) and small weights (close to 0). If we raise the weights to a higher power, all the weights

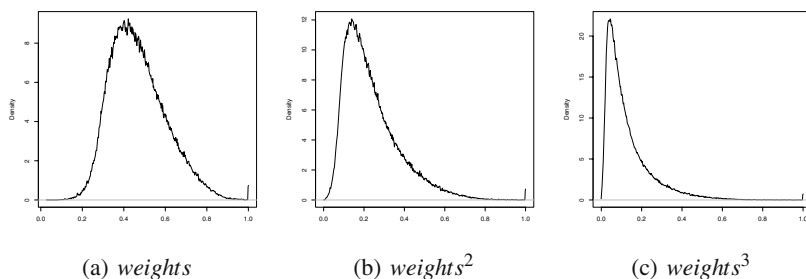


Fig. 4 Probability density estimation (PDF) of the weight distribution for weight transformations (raise to the power)

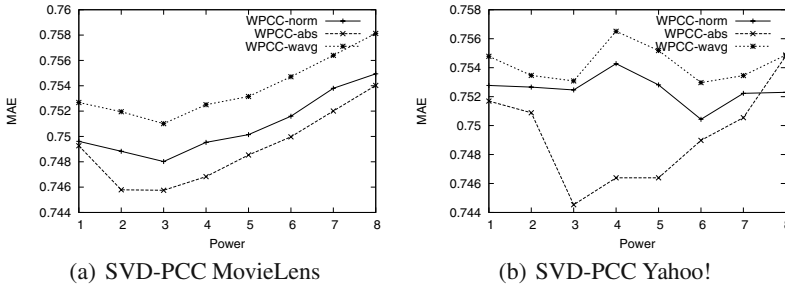


Fig. 5 Performance of SVD-PCC w.r.t. different powers of weights

would move towards zero, except the weights of the items perfectly correlated with the target item, i.e., with value equal to 1.

Figure 5 shows the results of an experiment where we varied the power to which all the weights are raised. Here we used the PCC-SVD weighting schema, and fixed k equal to 200 for the MovieLens dataset and 300 for the Yahoo! dataset. As we expected, raising the weights to a small power (two or three) decreases MAE. For the lack of space we do not show the corresponding results for the F measure, but similarly to results shown before the best performance is obtained raising the weight to the power of 2 and 3.

6 Conclusions

This paper analyzes a wide range of item weighting techniques suitable for CF. Each item weighting technique analyzes the data to get additional statistics about the relationships between the items. This information is then used to fine-tune the user-to-user similarity and to improve the accuracy of the CF recommendation technique. We have observed that the newly-introduced SVD-PCC weighting schema — designed to work for sparse data — performs better than the baseline CF method in the two datasets that was considered, and for all the accuracy measures that we used (MAE, F, Coverage). All the other methods do not have such a stable behavior and they outperform the baseline CF only for some of the error measures, and sometimes they perform even worse. We also experimentally evaluated three different weight incorporation techniques and explained their behavior. In the data sets that we used $WPC_{no-norm}$ showed a better performance than the other techniques.

This paper gives also an insight into the limitations of the weighting techniques when used for item filtering. The SVD-PCC weighting schema, which has good item weighting performance, showed worse results than the baseline method when applied to item filtering. On the other hand, item filtering with the genre weighting schema resulted in a reduction of MAE at the cost of a reduction of the coverage. Such an approach could be useful for applications where it is important to provide a small number of recommendations. If well used, it could also provide a powerful technique for determining a small number of well targeted recommendations.

In the future, we want to analyze item filtering in depth and understand if it can be used to improve accuracy with other weighting schemas. If successful, this approach could be used in distributed or cross-domain recommender systems, where it is important to understand what portion of items are relevant and should be retrieved from the other domain in order to improve the recommendation [6].

Weight amplification techniques needs some further study. In our last experiment, we tried to re-scale the original weights transforming them with a power function. This is just one possible approach and a better analysis of this issue is required. In fact, the weights computed by our proposed methods provide just a relative measure of the importance of the items, and determining the correct factor that must be used in the PCC metric requires some more experiments.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems* 23(1), 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: 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* 17(6), 734–749 (2005)
3. Aha, D.W.: Feature weighting for lazy learning algorithms. In: *Feature extraction, construction and selection: a data mining perspective*, vol. SECS 453, pp. 13–32. Kluwer Academic, Boston (1998)
4. Baltrunas, L., Ricci, F.: Dynamic item weighting and selection for collaborative filtering. In: Berendt, B., Mladenic, D., Semeraro, G., Spiliopoulou, M., Stumme, G., Svatek, V., Zelezny, F. (eds.) *Web Mining 2.0 International Workshop located at the ECML/PKDD 2007*, pp. 135–146 (2007)
5. Baltrunas, L., Ricci, F.: Locally adaptive neighborhood selection for collaborative filtering recommendations. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) *AH 2008. LNCS*, vol. 5149, pp. 22–31. Springer, Heidelberg (2008)
6. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-domain mediation in collaborative filtering. In: Conati, C., McCoy, K., Paliouras, G. (eds.) *UM 2007. LNCS*, vol. 4511, pp. 355–359. Springer, Heidelberg (2007)
7. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: Shavlik, J.W. (ed.) *ICML*, pp. 46–54. Morgan Kaufmann, San Francisco (1998)
8. Breese, J.S., Heckerman, D., Kadie, C.M.: Empirical analysis of predictive algorithms for collaborative filtering. In: Cooper, G.F., Moral, S. (eds.) *UAI*, pp. 43–52. Morgan Kaufmann, San Francisco (1998)
9. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6), 391–407 (1990)
10. Geng, X., Liu, T.Y., Qin, T., Li, H.: Feature selection for ranking. In: *SIGIR 2007: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 407–414. ACM, New York (2007)
11. Golub, G.H., Van Loan, C.F.: *Matrix Computations* (Johns Hopkins Studies in Mathematical Sciences). The Johns Hopkins University Press (1996)
12. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: *SIGIR*, pp. 230–237. ACM, New York (1999)

13. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.: Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22, 5–53 (2004)
14. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: *SIGIR 2004: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 337–344. ACM Press, New York (2004)
15. Kohavi, R., John, G.H.: Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2), 273–324 (1997)
16. Langley, P.: Selection of relevant features in machine learning. In: *Proceedings of the AAAI Fall Symposium on Relevance*. AAAI Press, Menlo Park (1994)
17. Mitchell, T.M.: *Machine Learning*. McGraw-Hill, New York (1997)
18. MovieLens dataset, <http://www.grouplens.org/>
19. Radlinski, F., Joachims, T.: Query chains: learning to rank from implicit feedback. In: *KDD 2005: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 239–248. ACM Press, New York (2005)
20. Resnick, P., Varian, H.R.: Recommender systems. *Communications of the ACM* 40(3), 56–58 (1997)
21. Salton, G., Mcgill, M.J.: *Introduction to Modern Information Retrieval*. McGraw-Hill Inc., New York (1986)
22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender systems – a case study. In: *Proceedings of the WebKDD 2000 Workshop at the ACM-SIGKDD Conference on Knowledge Discovery in Databases* (2000)
23. SAS Institute Inc., *SAS OnlineDoc(TM)*, Version 7-1 Cary, SAS Institute Inc., NC (1999)
24. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) *Adaptive Web 2007*. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)
25. Doug Rohde’s SVD C Library, version 1.34, <http://tedlab.mit.edu/~dr/svdlib/>
26. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artif. Intell. Rev.* 11(1-5), 273–314 (1997)
27. Yahoo! Research Webscope Movie Data Set. Version 1.0, <http://research.yahoo.com/>
28. Yu, K., Xu, X., Ester, M., Kriegel, H.P.: Feature weighting and instance selection for collaborative filtering: An information-theoretic approach*. *Knowledge and Information Systems* 5(2), 201–224 (2003)