Dynamic Item Weighting and Selection for Collaborative Filtering

Linas Baltrunas and Francesco Ricci

Free University of Bozen-Bolzano
Domeninkanerplatz 3, Bozen, Italy
{lbaltrunas,fricci}@unibz.it

Abstract. User-to-user correlation is a fundamental component of Collaborative Filtering (CF) recommender systems. In user-to-user correlation the importance assigned to each single item rating can be adapted by using item dependent weights. In CF, the item ratings used to make a prediction play the role of features in classical instance-based learning. This paper focuses on item weighting and item selection methods aimed at improving the recommendation accuracy by tuning the user-to-user correlation metric. In fact, item selection is a complex problem in CF, as standard feature selection methods cannot be applied. The huge amount of features/items and the extreme sparsity of data make common feature selection techniques not effective for CF systems. In this paper we introduce methods aimed at overcoming these problems. The proposed methods are based on the idea of dynamically selecting the highest weighted items, which appear in the user profiles of the active and neighbor users, and to use only them in the rating prediction. We have compared these methods using a range of error measures and we show that the proposed dynamic item selection performs better than standard item weighting and can significantly improve the recommendation accuracy.

1 Introduction

The World Wide Web, interconnecting a myriad of information and business services, has made available to on-line users an over abundance of information and very large product catalogues. Hence, users trying to decide what information to consult or what products to select may be overwhelmed by the number of options that they can potentially access. Collaborative Filtering (CF) is a recommendation technique which emulates a simple and effective social strategy called "word-of-mouth" and is now largely applied in Web 2.0 platforms. In CF personalized recommendations for a target user are generated using opinions (item ratings) of users having similar tastes to that of the target user [1]. A CF system represents users with their ratings on a set of items (ratings vectors). When requested to generate a recommendation for a target user, a CF system first selects a set of similar users according to a similarity/correlation measure computed on their ratings vectors. Then, it generates rating predictions for items not rated yet by the target user. Finally the system recommends the items with the highest predicted rating.

Neighborhood formation is an important step of CF [1], and this paper concentrates on feature weighting and feature selection methods, aimed at improving the recommendation accuracy by tuning the user-to-user similarity metric. In fact, CF can be seen as an instance-based learning approach where users are instances described by their feature vectors, where the product ratings play the role of features. 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. This paper reviews and compares feature weighting methods. Moreover, we introduce three dynamic feature selection methods that are suited to the specific characteristics of data used by CF prediction.

Usually, in standard classification and regression learning, feature selection is preferred to feature weighting. In fact, feature selection decreases dimensionality of data, speeds up the learning process and could improve model interpretability [5]. But, the application of feature selection to CF rating prediction and recommendation faces two major problems. Firstly, real world data exploited in CF systems contains a huge amount of features, and secondly it is extremely sparse. In this paper we show how both problems can be tackled exploiting a rather simple dynamic feature selection approach based on feature weights computed by a number of alternative methods. In practice, features with largest weights are selected on-line, every time a rating prediction is required, depending on the target user, target item, i.e., the item whose rating prediction is sought, and the neighbor user whose similarity is needed. Our experiments show that dynamic feature selection is more effective than feature weighting.

The paper is organized as follows. Section 2 overviews some related work. Methods used to improve user-to-user similarity are discussed in section 3. Section 4 describes the feature weighting methods we used in our study, and Section5 describes how dynamic feature selection methods can be exploited in a CF. All the proposed methods are evaluated in Section 6, and Section 7 draws the conclusions and presents future work.

2 Related Work

Feature weighting is a well studied problem in Machine Learning, and Instance-Based Learning in particular. A number of studies reported accuracy improvements when features are weighted in the instance-to-instance distance computation (see [9] for a survey). In the context of CF, feature weighting raises some notable problems. The huge amount of features and data sparsity makes most methods inefficient. Breese et al. [2] adapted the idea of inverse document frequency [7] for feature weighting in CF. The key idea of that approach, called Inverse user Frequency, is that universally liked and known items do not give a lot of information about the true user preferences, therefore, the weights for such commonly rated items should be decreased. The approach showed better performances than the unweighted version of the method. The same idea of decreasing the weights of commonly liked items was implemented by using variance weighting [3]. Here, items with bigger variance are supposed to better distinguish

users' tastes, therefore, they receive larger weights (in user-to-user similarity). But, the authors report that this method slightly decreases the Mean Absolute Error (MAE) with respect to the non-weighted version.

In [10] Yu et al. introduced Information Theoretical approaches for feature weighting. They showed that Mutual Information and Entropy based methods perform better than Inverse User Frequency and the baseline CF. Moreover, Mutual Information gains 4.5% accuracy improvement and performs better (even when trained on a small number of instances) than the baseline CF. They also report that Inverse User Frequency, differently from an earlier report, decreases the accuracy. Since these papers are contradicting we decided to test ourselves the performance of some of these methods. [4] presents another automatic weighting schema for CF systems. This method tries to find weights for different items that bring each user closer to the similar users and further from dissimilar users. The method uses ideas from model-based approaches and obtains a reduction of MAE.

Feature selection algorithms were studied in Machine Learning for several decades [9]. Kohavi or Langley However, the huge search space of thousands and millions of items, and the fact items/features must be selected depending on the target item prediction, makes them hard to apply to CF. For these reason feature selection has not been applied to CF.

3 Improving the Similarity

The user-to-user similarity is the core computation step in user-based CF [1]. Similarity is used in the neighborhood formation and in the final rating prediction. 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 the Pearson Correlation Coefficient (PCC). PCC is preferred when data contains only positive ratings, and has been shown to produce better results in such cases [2]. PCC among 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 user x to item i, and $\bar{v_x}$ is the mean of the ratings of user x. The sum runs over all the items i that both users rated.

Feature weighting and feature selection are two methods that have been largely used in instance-based learning approaches to improve the prediction accuracy of classifiers [9]. In fact, a user-based CF system can be described as a collection of instance-based classifiers, one for each item whose rate is to be predicted. Given a target item (class) and a user whose rating must be predicted, the user's ratings on all the other items provide the instance description (predictive features or items). In this perspective, the rating prediction step of a CF system can be described as a classification or regression learning problem, i.e., one classification/regression problem for each item's rating prediction. The similarity measures we introduced above are based on user preferences, i.e. item

ratings. Hence, these items can be regarded as user's features, and the ratings are the feature values. In the rest of this paper we shall call features of the user the items managed by the CF system, and feature values the ratings assigned by the user to the corresponding item.

Feature weighting methods assign to each instance's feature a weight that measures how important is the feature in the overall instance-to-instance similarity. This translates in the CF case to weights assigned to items used to balance the importance of predictive items in the user-to-user similarity.

Feature selection methods operate in a different way. Instead of providing a weight for each feature, they decide whether a feature must be used or not in the similarity computation. In fact, feature selection could be seen as a particular case of feature weighting, where feature selection uses binary valued weights. We have applied both approaches to a weighted version of the PCC user-to-user similarity measure:

$$WPCC(x,y,j) = \frac{\sum_i (w_{ji}(v_{xi} - \bar{v_x}))(w_{ji}(v_{yi} - \bar{v_y}))}{\sqrt{\sum_i (w_{ji}(v_{xi} - \bar{v_x}))^2 \sum_i (w_{ji}(v_{yi} - \bar{v_y}))^2}}$$

where j denotes the target item of the rating prediction, and w_{ji} is the weight of the (predictive) item i when making a prediction for j. In feature weighting methods, the weight could be any real number, however, in this paper we use only positive weights. Feature selection is here implemented by assigning a weight equal to 1 for the item/feature that is selected, and 0 otherwise.

The role of the target item j needs some further explanation. In fact this gives to a feature weighting or selection method the flexibility to assign to each predictive item a weight that depends on the target item whose rating prediction is sought. Hence, the weights used for predicting the rating of item j (for all users) can be different from those used for making predictions on another item j'. In this way, we can exactly encode in a weight how much two items are correlated, and what is the role that an item should play in neighborhood formation when a prediction is sought for the second one. Without such a flexibility we could only express knowledge about how important an item would be for all rating predictions. In fact, there are some examples of item weighting approaches that do not make this distinction and compute the absolute importance of an item. An example of this approach is the variance weighting method [3] that will be explained shortly.

For efficiency reasons, in all the feature weighting and feature selection algorithms that we shall describe later, we first compute all the weights and later on we use these stored values in the user-to-user similarity computation. In practice, to store all the weights, we need an $M \times M$ matrix of weights, where M is the cardinality of the item set. In other words, one vector of weights of size M is used for each item.

4 Items Weighting Methods

Feature weighting tries to estimate how much a particular feature is important for deciding about the class membership of instances. In CF, item weights can be

learned while exploring training data consisting of user ratings, or using external information associated with the items. Based on this observation we classify the methods we are going to present in two groups. The methods in the first group determine item weights using statistical approaches and are aimed at estimating statistical dependencies within rating data. The second group of methods uses item descriptions to estimate item dependencies and finally infer the weights. In the rest of this paper we will consider Random, Variance, Mutual Information, Tag label based, and PCC based feature weighting approaches.

Random. The first method is the baseline and uses a random item weighting. Random weights in $[0, 1]$ are selected for each target item and predictive item combination.

Variance. Variance method is based by an observation originally made by [3], that knowing a user's ratings on certain items could be more valuable than other ratings in discerning a user's interests. For example, if all users rate a particular item with the maximum rating value, this information is not much valuable to determine a user preferences and therefore should not be exploited in the user-to-user correlation (similarity). Hence, the variance method gives higher weights to items with higher variance among the ratings provided by the users to that item:

$$w_{ji} = w_i = \frac{\sum_{u=1}^{n} (v_{ui} - \bar{v_i})^2}{n-1}$$

here $\bar{v_i}$ is the mean of the ratings of item i, and n is the number of users in the database. Variance feature weighting method uses only information on a single item, the item that is weighted. All the methods that we're presenting next explore internal relations between predictive items and target item.

IPCC. The first method in this group uses PCC as measure of dependency between two vectors, which represent the ratings of all users for the two items. Since PCC ranges from -1 to 1, and we are using only positive weights we transform PCC to IPCC as follow, to take values between 0 and 1:

$$w_{ji} = \frac{\frac{\sum_u (v_{ui} - \bar{v_i})(v_{uj} - \bar{v_j})}{\sqrt{\sum_u (v_{ui} - \bar{v_i})^2 \sum_u (v_{uj} - \bar{v_j})^2}} + 1}{2}$$

here u runs on all the users in the database that have rated both i and j, and $\bar{v_i}$ is the mean of item i ratings. The usage of Pearson Correlation Coefficient between items to determine item weights has not been used in previous researches.

Mutual Information. Mutual Information measures the information that a random variable provides to the knowledge of an other. In CF we compute the mutual dependency between target item variable and predictive item variable (the values are the ratings). 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 the $I(X;Y) = H(X) + H(Y) - H(X,Y)$, where $H(X)$ denotes the entropy of X, and $H(X,Y)$ is the joint entropy of X and Y. Using the above formula we compute weights as:

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

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

Tag weighting. All the previous methods exploit statistics of the users rating data to compute the feature weights. The last method we present here computes weights using items' description. The general idea is to increase the weight w_{ji} if the description of the target item j and the predictive item i are similar. In Tag weighting we increase the weight w_{ji} if the target item j is labelled with a tag that is also used to tag the predictive item i. In the movie recommendation dataset that we are going to use for our experiments, movies are tagged with movie genres (a movie can be tagged with more than one genre). Hence, we make the assumption that the larger the number of common tags (genres), the higher is the dependency. The weight of the predictive item i for a prediction of the ratings of item j is given by:

$$w_{ji} = 1 + \frac{\#\ co-labeled\ classes\ between\ i\ and\ j}{\#classes}$$

We note that [8] introduces the idea of using domain specific information to selectively choose the items to be used in the user-to-user correlation.

5 Item Selection

Feature selection is a technique commonly used in machine learning to filter out irrelevant or noisy features in classification learning tasks. While in classical classification learning each instance can belong to a single class, in CF rating prediction, given an active user (instance), the goal is to make a rating prediction for the largest number of items in the database. Therefore, instead of a single classification problem for each user (instance) we have a classification problem for each target user and target item pair. Hence, as we mentioned above, we would like to have one set of relevant items (predictive items) for each target item. The optimal feature selection problem requires to conduct a search procedure in the space of the subsets of the features [5]. 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 features , i.e., the predictive items. Conversely, we propose to use a dynamic approach that reuses the information provided by the feature weighting methods to select, for each target item and user pair, an appropriate set of predictive items. Hence, first we compute feature weights, using one of the methods described above, and then we filter out irrelevant features, i.e., the predictive items with the smallest weights, for any given target item. Therefore, every method used for feature weighting can be transformed into one or more corresponding feature selection method depending on how the weights are used to select items.

In the rest of the section we describe three different feature selection methods for CF systems. We will explain all three methods using an example of user-item rating matrix showed bellow:

	$i_1(w_{t1}=0.1)$	$i_2(w_{t2}=0.2)$	$i_3(w_{t3}=0.3)$	$i_4(w_{t4}=0.4)$	$i_5(w_{t5}=0.5)$	i_t
u_1	5	3	2	1	?	6
u_2	4	2	4	?	5	?

The table consist of two user and six items. Question marks indicate unknown ratings. Let us assume that we want to predict user's u_2 rating for the item i_t. Moreover, suppose that we have computed item weights beforehand, using one item weighting algorithm, and weights for all the items are showed in the first line in the brackets. For example, the weight of item i_1 for predicting the target item i_t is $w_{t1} = 0.1$

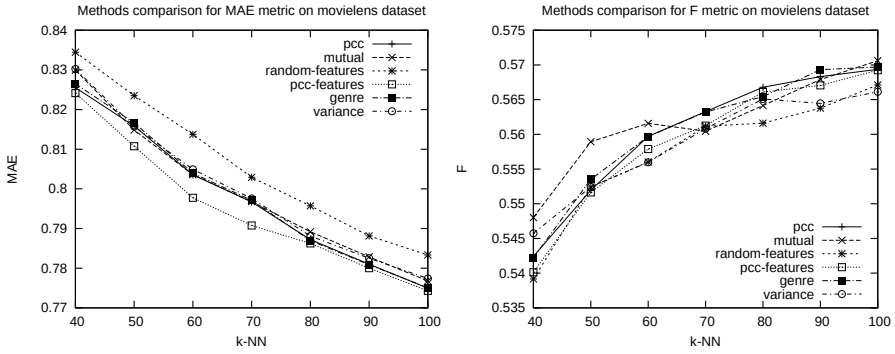
The first method, called **best-f-per-target**, selects the f items (predictive items) with highest weights. In other words, from the set of weights $W = \{w_{t1}, \ldots, w_{tM}\}$ the algorithm selects the f items with highest weights. Using the example above, If f is set to 3, then best-f-per-target would select items 5, 4 and 3. Best-f-per-target does not take into account, if the target user, or the neighbor user express any rating for the selected items. Therefore, some users can have no available ratings for the selected items.

To overcome this problem, we propose a second method, called **best-f-per-user** that selects the best f items, according to the weights, that are present in the target user's profile. Using the same example, in this case best-f-per-user would select the items 5, 3 and 2. Hence, in best-f-per-user predictive items are selected on a user base and are kept stable for all the predictions for a user.

The last method extends the idea of selecting the best items that are actually presents in the user profiles even further. The method **best-f-per-overlap** selects the f items with largest weights that are present in both the target user profile and in the neighbor user profile. In the example above, method best-f-per-overlap would select items 3, 2, and 1. Hence, in best-f-per-overlap the items used in each rating prediction change for every target item and user pairs.

6 Experimental Evaluation

This section evaluates feature weighting and feature selection methods for CF. All methods compute the set of feature weights and later use them in WPCC to



(a) MAE measure (b) F measure

Fig. 1. Performance of item weighting methods.

compute user-to-user correlation. To generate the prediction, WPCC is used in both, computing k nearest neighbors of the target users, and also to determine the predicted rating in the prediction step:

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

here the sum runs on the k-nearest neighbors of the user x, $N(k,x)$.

In our implementation, as previously done in [8], we do not take into account neighbors which have less than six overlapping rated items. In our experiments we used the MovieLens dataset, containing 100K ratings, for 1682 movies by 943 users. The data sparsity is 96%, and each rating can vary in the range $\{1,2,3,4,5\}$.

To evaluate the proposed methods the full dataset was randomly divided into train (80K) and test (20K) cases. We used the train dataset to learn the weights and also to make a prediction for the test ratings. We evaluated the performance of the proposed methods with a wide range of error measures.

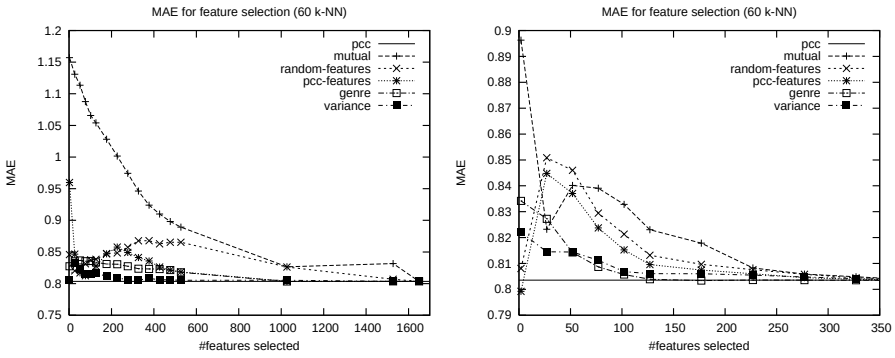
To measure the accuracy we used: MAE, High MAE, F measure, precision and recall [3]. To compute F, precision and recall measures, we considered items worth recommending (relevant items) only if their ratings were 4 or 5. Since, we are interested in recommending only the top items, we propose to modify the MAE error measure to see how an algorithm performs on predicting the highest ratings. We defined High MAE measure as the MAE obtained only on ratings with values 4 and 5.

In the first experiment we evaluated all feature weighting methods with a varying number of nearest neighbors. The baseline prediction method is the standard unweighed CF using PCC as user-to-user similarity measure (named *pcc* in all figures). Figure 1 shows the performances of the six proposed weighting methods with the number of neighbors ranging from 40 to 100.

It is clear from these results that there is no single best method for all error measures and nearest neighbors (for lack of space we do not show the results obtained with all the error measures). Some methods perform better than other in specific situations and sometimes they perform even worse than the standard CF. As expected, random feature selection performs worst and degrades the base line unweighed approach (Figure 1(a).) Considering only MAE we can observe that feature weighting based on IPCC performs better than other feature weighting or baseline method up to 80 neighbors. With 60 nearest neighbors it gets up to 1.5% improvement, and random-feature weighting loses 1.2% accuracy. In fact, the performance gains even for the best performing feature weighting methods are small. We should mention that in our experiments we did not observe any improvement for Mutual Information weighting, contradicting what is reported in [10]. That could be due to the fact that we used different datasets and different missing value interpretation. In [10] Yu et al. used the EachMovie dataset and discarded users, who have rated less than 20 items. Moreover, in our implementation when computing entropy we skipped undefined summation terms like $p(a)log(p(a))$ when, $p(a) = 0$. The interpretation of undefined terms is not clear in [10]. Considering F measure, it is clear that there is no single method, which outperforms all other methods. Random feature weighting is still the worst, however, the difference is not large. Here, differently from MAE measure, Mutual Information performs well, especially when the number of nearest neighbors is small.

We need to mention that, the weighting methods which depend on the target item, suffer from several drawbacks. First of all, the memory complexity is quadratic in the number of items, because they use $M \times M$ matrix to store the weights (M is the number of items). This makes such methods not easily applicable for big datasets with a lot of features/items. Another problem, is related to the computation of the nearest neighbors for a given user. To speed up this computation, most of the implementations index the nearest neighbors or cache the user-to-user similarity values for most of the active users. When the feature weights depend on the target item caching is hardly possible. In fact, in the worst case, in order to cache the user neighborhood we would have to store one neighborhood for each of the item and user pairs, as the user-to-user similarity uses weights that depends on the target item.

The second experiment evaluates the first item selection method we proposed, i.e., best-f-per-target. To compute the weights we used the five feature weighting methods described in Section 5. In figure 2(a) the performance of all these methods are depicted. We note that in general all the methods performed worse than the baseline, i.e., standard CF without item selection. The motivation is that users tend to rate only a small fraction of the total available items and the number of co-rated items between two users can be very small and even null. Selecting items, even if relevant, decreases the number of co-rated items even more. Imagine for instance, that there are two users that are perfectly correlated but have co-rated a few items. One user could be used to predict ratings of second user. But, when we select a small number of items we have a very



(a) "best-f-target" (b) "best-f-user"

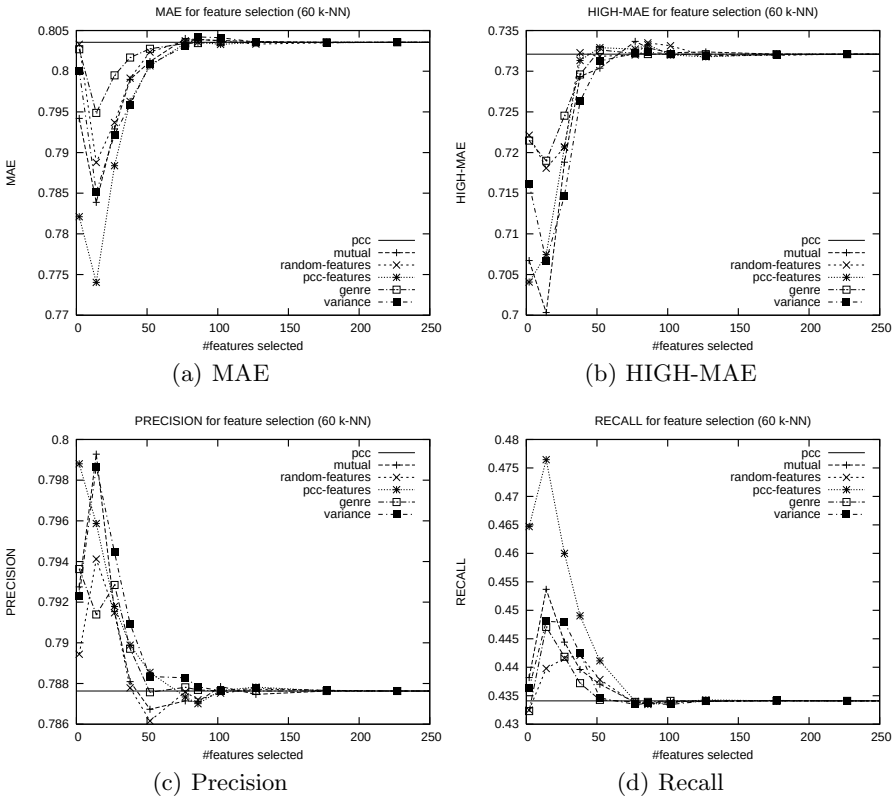
Fig. 2. MAE for two item selection methods.

small chance that these users will overlap on the selected items. Therefore, using this simple item selection procedure, many good neighbors could be discarded degrading the prediction accuracy. Anyway it is important to note that using a smaller number of items the accuracy can be pretty close to the standard CF system and when one uses more and more items the accuracy converge to that of the standard CF method.

In the second experiment we tested the best-f-per-user method . The accuracy (see Figure 2(b)) loss is smaller than best-f-per-target, and a faster convergence to the accuracy of the standard CF is obtained with very few items. However in general we do not really improve the accuracy. We observe that with 300 items we obtain the same accuracy of the baseline method. It can be explained by the fact, that virtually no users rate more than 300 items and much less ratings are needed to compute a reliable prediction.

In the final experiment we evaluated the best-f-per-overlap method. Figure 3 shows results for different accuracy metrics. Here, selecting a small number of items, we can observe a significant improvement of all the accuracy measures. If we select a larger and larger number of features every method results in the same accuracy as the base line method. We note that there is no single best performing method and the winner depends on the particular error measure we use. In fact, for MAE we decrease the error up to 3.7% using PCC based feature weighting method and for Recall measure the improvement is up to 9%. Random feature selection method and genre labelling method are the worst, however, they can also improve the performance of the baseline method.

It is worth noting that item selection based on best-f-overlap makes the user-to-user similarity much faster to compute since it is based on a small number of overlapping items. In fact, with only 14 overlapping items the best performances can be achieved. Moreover, for the same reason, it is possible to explain to



(a) MAE (b) HIGH-MAE

(c) Precision (d) Recall

Fig. 3. Performance of "best-f-overlap" item selection methods.

the user her correlation with her neighbors and increase trust on the system prediction providing transparency on the recommendation mechanism.

7 Conclusions

In this paper we compared a number of feature weighting methods, some original and some previously proposed in the literature. We introduced IPCC based weighting method and adopted tag weighting method to CF. Furthermore we introduced three feature selection methods applicable to CF prediction that are based on feature weighting. We evaluated the proposed methods along with a range of error measures and varying the number of nearest neighbors. The main conclusion of this work is that there is no single best feature weighting method, and accuracy improvements using weighting are very small. Considering instead the item selection methods, we show that dynamic best-f-per-overlap outperforms item weighting and the two other item selection methods. In summary,

best-f-per-overlap, using a small number of items, can achieve significant improvements for all considered error measures. This result is important because it shows that CF user profiles contain a lot of redundancy and even if the dataset is sparse the information is not uniformly distributed among users. This work also shows that CF performances can be improved with a careful selection of the item ratings, i.e., acquiring ratings for certain items can greatly improve the performance of the recommendation algorithm.

In the future we will compare feature weighting methods with other techniques such as instance (user) selection methods. As we noticed above, we observed that there is no single method outperforming the others in all cases. Therefore, we want to design a meta learning approach that tunes user-to-user similarity according to a number of contextual and reliability indicators.

The computation of the feature weights is an expensive step in all considered algorithms. Moreover, we need to recompute the weights when new ratings are registered to the system. At the moment we are working on a feature weighting method based on the RELIEF [6] feature estimator in order to avoid feature weights recomputation. Using this method with every new rating we could gradualy adapt the overal weighting schema.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
2. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *UAI*, pages 43–52. Morgan Kaufmann, 1998.
3. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237. ACM, 1999.
4. R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344, New York, NY, USA, 2004. ACM Press.
5. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
6. M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, 53(1-2):23–69, 2003.
7. G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
8. T. K. Shlomo Berkovsky and F. Ricci. Cross-domain mediation in collaborative filtering. In C. Conati, K. McCoy, and G. Paliouras, editors, *LNAI*, volume 4511 of *Lecture Notes in Artificial Intelligence*, pages 355–359. Springer, 2007.
9. D. Wettschereck, D. W. Aha, and T. Mohri. 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.
10. K. Yu, X. Xu, M. Ester, and H.-P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach*. *Knowl. Inf. Syst.*, 5(2):201–224, 2003.