Probabilistic Partial User Model Similarity for Collaborative Filtering

Amancio Bouza, Gerald Reif, Abraham Bernstein

Department of Informatics, University of Zurich
{bouza,reif,bernstein}@ifi.uzh.ch

Abstract. Recommender systems play an important role in supporting people getting items they like. One type of recommender systems is user-based collaborative filtering. The fundamental assumption of user-based collaborative filtering is that people who share similar preferences for common items behave similar in the future. The similarity of user preferences is computed globally on common rated items such that partial preference similarities might be missed. Consequently, valuable ratings of partially similar users are ignored. Furthermore, two users may even have similar preferences but the set of common rated items is too small to infer preference similarity. We propose first, an approach that computes user preference similarities based on learned user preference models and second, we propose a method to compute partial user preference similarities based on partial user model similarities. For users with few common rated items, we show that user similarity based on preferences significantly outperforms user similarity based on common rated items.

1 Introduction

Users are overwhelmed with the vast amount of items (e.g. books, movies, locations such as bars or restaurants) offered by online stores or web guides. They have to invest a lot of effort to discover relevant items. Keyword-based filtering approaches are not suitable to reduce the vast amount of items to a reasonable size. Additionally, keyword-based filtering does not consider the user's preferences or the item quality. Even the use of average item ratings over all users is not expressing a single user opinion adequately and for that reason only provides poor user support. For this reason, recommender systems play an important role in supporting people finding items they like. Recommender systems aim to filter relevant items according to personal user preferences. Relevant items are provided directly to the user instead of asking the user to search for relevant items. A good example of a recommender system is the Amazon online store which provides recommendations such as "People that bought product X, also bought product Y".

The fundamental assumption of recommender systems that are based on user-based collaborative filtering is that people who share similar preferences for the same items behave similar in the future. Consequently, each user can benefit from the past item experiences of these users. Typical recommender systems

compare user preferences based on the set of items that both users rated. We call these items the *common rated items*. However, we argue that the set of common rated items reflect the user preferences only partially. First, two users might share the same interest only in parts of the item domain. For example in the domain of restaurant recommendations, users might share similar ratings on Italian restaurants but not on Chinese restaurants. Traditionally, the similarity of the user preferences is computed globally over all common rated items and do not consider that the user preferences can overlap only in parts of the domain. Second, if two users live in different cities, the set of common rated items might be empty, if the users did not visit at least one time the same restaurant. In this case traditional recommender systems cannot compute the similarity of the preferences between these users.

In this paper we argue that the set of common rated items might be too small to infer the similarity of the user preferences and may reflect the user preferences only partially. In addition, partial similarities between users might be missed because the user preference similarity is computed globally. Thus, we suggest to compare the similarity among user preferences based on learned user preference models that are an accurate approximation of the users' real preferences. We propose a formal probabilistic framework that compares user preference models and enables the similarity computations of partial user preferences.

2 Related Work

The fundamental assumption of recommender systems is that people who share similar preferences for common items behave similar in the future. Thus, computing the user preference similarity is a key challenge. Several user preference metrics like Pearson correlation and cosine similarity have been proposed [15, 5, 10] that compute user preference similarity based on common rated items. But user-based collaborative filtering approaches face the challenge of rating sparsity [16]. In case the set of common rated items is small, no accurate user similarity can be inferred. Instead of common rated items, user profiles are used to compute user preference similarities. The Fab system [2] recommends documents. Its users are asked to create a user profile by selecting topics of interest. Users are similar if they share many topics of interest. Documents are then recommended that matches the user's profile and that have been liked by users with similar user profiles. In [8], user profiles are represented as topic preference vectors that describe the relevance of every topic for the user. The synergy between ontologies and recommender systems has been demonstrated in [12]. Quickstep [13] uses an ontology for user profiling. It learns by observing the user's behavior in what research domain the user is interested and recommends other papers of that research domain. The user profile consists of a preference vector containing the relevance of the corresponding category in the ontology. In [1], the relevance of item features is used to build the user's feature preference vectors instead. Instead of building user profiles, missing user ratings are predicted with a user model to overcome the sparsity of user ratings [11].

Computing partial user preferences has been discussed in [3]. Basu et al. suggests grouping items and computing user similarity within a group. A group is defined by a single feature. An item belongs to a feature group if the item provides the feature. A similar approach has been proposed in [6] where items are clustered based on item feature descriptions to build communities of interest.

3 Formal Framework of User Preference Model Similarity

In this section, we first introduce the notation used in this paper and the basic framework of recommender systems. Section 3.2 discusses the formal representations of the user's true preferences. In Section 3.3, we introduce our approach of how to define the similarity of user preference models followed by its application to predict item ratings. We continue in Section 3.5 with our second approach and the definition of the partial user preference similarity. We close with Section 3.6 that describes how to compute item ratings based on partial user preference similarities.

3.1 Formal Framework and Notation

The basic elements in a collaborative recommender system are the set of users N, the set of items I, the set of ratings that are provided by the users for some items and the set of rating concepts C from which users can choose to describe adequately their opinion about an item. We denote R as the $n \times m$ rating matrix with $n = |N|$ as the number of users and $m = |I|$ as the number of items such that the value of the ith row at position j corresponds to the rating of the ith user for the jth item. We refer to a certain user as U_i with $i \in \{1, \ldots, n\}$, to a certain item as I_j with $j \in \{1, \ldots, m\}$ and denote $r_{ij} \in C \cup \{\varnothing\}$ of the rating matrix R as the rating of user U_i for the item I_j. The value of rating r_{ij} is either a rating concept $c_k \in C$ or $\varnothing$, if no rating has been provided yet. $z = |C|$ is the number of rating concepts the user can assign to every item in I. The item rating vector R_i for the user U_i contains the rating for every item in I. Further, we refer to the set of items that user U_i has actually rated ($r_{ij} \neq \varnothing$) as the item subset $\overline{I}_i \subseteq I$. We denote the user for whom we compute the recommendations as the active user U_a with $a \in \{1, \ldots, n\}$.

Depending on the recommendation algorithm, we can distinguish between several sets of rating concepts C. These can be classified into four groups:

- *Nominal rating*: The task of item recommendation can be treated as a classification problem that associates an item with one ore more rating classes. Popular classes of rating concepts C are $\{relevant, irrelevant\}$ or $\{likes, likes\ not\}$.
- *Ordinal rating:* The rating concepts are interrelated and can be ordered. The typical example for ordinal rating concepts is the star-rating on a 1-5 integer scale: $\{\bigstar, \ldots, \bigstar\bigstar\bigstar\bigstar\bigstar\}$. With ordinal ratings only the assertion can be done that a 4-star rated item is better then a 2-star rated item.

– *Interval rating*: Items can be rated with a numeric value from $\mathbb{R}$. In general, such ratings can be normalized to a $[-1, 1]$ scale.
– *Ratio rating*: Items can be rated with a numeric value from $\mathbb{R}$. In general, such ratings can be normalized to a $[0, 1]$ scale.

In general, a recommender system is a function f which returns for the active user U_a the computed item rating vector R_a. The function f takes all ratings in R, the user U_a, all users in N, all items in I and the rating concepts in C as input. We can conceptualize a recommender system as follows:

$$\widehat{R}_a = f(R, I, C, U_a),\ r \in C \cup \varnothing$$

3.2 User Preference Model

Most recommender systems represent the true user preferences as item true rating vector R_i^{true} which contains for every item $V_j \in I$ the true rating $\widetilde{r}_{ij} \in C$:

$$R_i^{true} = \langle r_{i1}^{true}, \ldots, r_{im}^{true} \rangle$$

If all true item ratings of a user are known, providing recommendation comes down to a trivial sorting problem (sorting the vector R_i^{true}). In general, a recommender system has only partial knowledge about the user U_i's true preferences R_i^{true}. We assume that the provided ratings in R_i are equal to the corresponding true ratings in R_i^{true}. The user U_i provides only partial rating information for three reasons:

– *Costs*: Temporal or monetary costs restrict the amount of items that a user is able to consume and provide a rating for.
– *Usability*: The rating effort is too high because the item has to be found first (search costs) and then being rated with too much effort (usability).
– *Privacy*: The user has an interest in not to publish all personal item ratings

Since the true item rating vector R_i^{true} is only partially known, we adjust the representation of the user's preferences to a more general way and more adaptable to machine learning. We assume that every user is able to assign the proper rating concept to an item. More formally, every user $U_i \in N$ is able to assign a rating concept $c_k \in C$ to every item $I_j \in I$ using his mental rating function $u_i(I)$:

$$u_i(I_j) : I_j \mapsto c_k = r_{ij}^{true}, \forall I_j \in I$$

We can reason that the user U_i always associates the item I_j with the true rating concept c_k. Therefore, the probability that the rating function $u_i(I)$ provides the true rating concept c_k for all $I_j \in I$ is always 1.

$$P\big(u_i(I_j) = c_k | r_{ij}^{true} = c_k\big) = 1, \forall I_j \in I$$

Instead of guessing all the user U_i's ratings, we approximate the rating function $u_i(I)$ based on the known user U_i's ratings R_i over the subset $\overline{I}_i \subseteq I$. We assume, that the distribution of ratings of R_i represent the real rating distribution of R_i^{true}. The rating distribution is the frequency of appearance of every rating concept in C in the vector R_i. Hence, a computer program can learn an approximation of $u_i(I)$ based on R_i and the set of items I.

The learner faces the problem to hypothesize the rating concepts $c_k \in C$ for the items $I_j \in \overline{I_a}$. For this purpose, the learner has to find the hypothesis h from the hypotheses space of all possible hypotheses H that estimates best the proper rating concept c_k the active user U_a associates with item I_j. The performance measure P is used to determine the best hypothesis h. To summarize, the rating concept learning task is to find the set of hypotheses that associates all items $I_j \in \overline{I}_a$ with the proper rating concepts $c_k \in C$. The learner selects the hypotheses subset $\overline{H}_a \subset H$ and builds a hypothesized rating function $h_a(I)$ that selects the best hypothesis $h \in \overline{H}_a$ to hypothesize the active user U_a's rating for the item I_j. The hypothesized rating function is a classifier that is built by the learner. A perfect hypothesized rating function $h_a(I)$ for $I_j \in \overline{I}_a$ with $j = \{1, \ldots, m\}$ means:

$$h_a(I_j) = u_a(I_j), \forall I_j \in \overline{I}_a$$

In general, $h_a(I)$ is an approximation of $u_a(I)$ such that the probability that $h_a(I) = u_a(I)$ is usually below 1. That can be expressed as:

$$\alpha \sum_{k=1}^{z} \sum_{j=1}^{m} P\big(h_a(I_j) = c_k | u_a(I_j) = c_k\big) \leq 1$$

The sum is normalized to 1 with the normalization factor α. Hence, we can state that $h_a(I)$ approximates $u_a(I)$ by an error function $\varepsilon(I_j)$:

$$u_a(I) = h_a(I) + \varepsilon(I)$$

The performance of $h_a(I)$ depends on the hypotheses space H that is based on the human designer's choice and on the other side on the amount of experience E respectively amount of the user's item ratings. If an adequate hypotheses space H has been defined by a human designer, we argue that the error function $\varepsilon(I) \to 0$ because the performance of $h_a(I)$ increases with more experience E. Hence, we can state:

$$\boxed{u_a(I) \approx h_a(I)} \tag{3.1}$$

3.3 User Preference Model Similarity

We denote the similarity of preferences between user U_a and U_b as $sim(U_a, U_b)$. If user U_a and U_b always rate the same item identically, user U_a and U_b share the identical preferences. If user U_a and U_b always rate the same item differently, no preference similarity exists between the preferences of user U_a and U_b. We

define the similarity $sim(U_a, U_b)$ as the probability P that user U_a and user U_b both rate all items $I_j \in I$ with $j = \{1, \ldots, m\}$ equally. We denote $z = |C|$ as the number of rating concepts.

$$sim(U_a, U_b) \equiv \alpha \sum_{k=1}^{z} \sum_{j=1}^{m} P(r_{aj} = c_k \wedge r_{bj} = c_k)$$

The sum is normalized to 1 with the normalization factor α. Hence, we define the similarity on the interval $[0, 1]$ with 0, no similar preferences, and 1, identical preferences. Since not all ratings are known by the recommender system, the similarity $sim(U_a, U_b)$ can be computed only based on common rated items $I_j \in \overline{I}_a \cap \overline{I}_b$. The smaller the size of the intersecting item set is, the smaller the confidence of the accuracy and the smaller the accuracy of the similarity $sim(U_a, U_b)$ is.

Therefore, we have to generalize the idea of similarity. The user U_i's preferences can be explained with his personal rating function $u_i(I)$ as demonstrated in the previous Section 3.2. Hence, we can formulate the similarity $sim(U_a, U_b)$ between user U_a and U_b as the similarity of both personal rating functions $u_a(I)$ and $u_b(I)$:

$$\begin{aligned} sim(U_a, U_b) &= sim\big((u_a(I), u_b(I)\big) \\ &\equiv \alpha \sum_{k=1}^{z} P\big(u_a(I) = c_k \wedge u_b(I) = c_k\big) \\ &\equiv \alpha \sum_{k=1}^{z} P\big(u_a(I) = c_k | u_b(I) = c_k\big) P\big(u_b(I) = c_k\big) \end{aligned}$$

Because either $u_a(I)$ or $u_b(I)$ are known, we approximate both with $h_a(I)$ and $h_b(I)$ respectively according to Eq. 3.1. Hence, we can write the similarity $sim\big(u_a(I), u_b(I)\big)$ as:

$$\begin{aligned} sim\big(u_a(I), u_b(I)\big) &\approx sim\big(h_a(I), h_b(I)\big) \\ &\approxeq \alpha \sum_{k=1}^{z} P\big(h_a(I) = c_k | h_b(I) = c_k\big) P\big(h_b(I) = c_k\big) \end{aligned} \tag{3.2}$$

Compared to the computation of the similarity $sim(U_a, U_b)$ on common rated items $\overline{I}_a \cap \overline{I}_b$, we now can compute the similarity $sim(U_a, U_b)$ on the merged item set $\overline{I}_a \cup \overline{I}_b$. The probabilities are computed by applying $h_a(I)$ and $h_b(I)$ to the merged item set and comparing the predicted rating concepts.

3.4 Collaborative Filtering based on User Model Similarity

In user-based collaborative filtering, the similarity among users can be used to predict for the active user U_a the rating $\hat{r}_{aj}$ of the item I_j. The rating $\hat{r}_{aj}$ is the

Listing 1.1. Example of a conjunction of constraints of movie features

```
movie hasGenre Drama = yes
movie hasGenre Romance = yes
movie hasReleaseYear = 1942
movie isOfCountry = m:Country_USA
movie isPresentedIn Black_and_White = yes
movie isPresentedIn Color = no: 5
```

weighted deviation from the active user U_a's average rating. More specifically, the rating $\hat{r}_{aj}$ is the sum of U_a's average rating $\overline{r}_a$ and the normalized sum of the weighted difference of all the other user U_b's rating r_{bj} and their average ratings $\overline{r}_b$ [5]:

$$\hat{r}_{aj} = \overline{r}_a + \kappa \sum_{b \neq a}^{n} sim(U_a, U_b) * (r_{bj} - \overline{r}_b)$$

The factor κ is a normalization factor such that the similarities sum to unity. According to the previous section 3.3, it is feasible to approximate the similarity $sim(U_a, U_b)$ of user U_a and U_b with the similarity $sim\big(h_a(I), h_b(I)\big)$. Therefore, we predict the rating $\hat{r}_{aj}$ as follows:

$$\hat{r}_{aj} = \overline{r}_a + \kappa \sum_{b \neq a}^{n} sim\big(h_a(I_j), h_b(I_j)\big) * (r_{bj} - \overline{r}_b)$$

3.5 Partial User Preference Model Similarity

In the previous Section 3.3, we proposed a probabilistic approach to define the similarity of two user preference models. But as we have already argued, overall similarity of user preferences misses partial user preference similarity. In general, a user has various preferences. In the context of movies, a user may not only like action movies, but also romantic movies. The user's preferences consist of a set of single preferences. Each such preference is described by a conjunction of constraints of features as shown in Lst. 1.1. The constraints may be a specific value, no acceptable value exists or any value is acceptable.

It is not feasible to compare single preferences of two users because a single preference may be similar to a set of preferences of the other user. Hence, it is necessary to describe the problem of computing partial user preference similarity of user U_a and U_b as the problem of computing the similarity of user U_a's preference to a composition of user U_b's preferences.

According to Eq. 3.1 in Section 3.2, it is feasible to approximate hypotheses to preferences. A user U_i's preference model $h_i(I)$ consists of a subset of hypotheses $h_{a,q} \in \overline{H}_a$ with $q \in \{1, \ldots, |\overline{H}_a|\}$ of the hypotheses space H. Analogously to preferences, a hypothesis h is described as a conjunction of constraints of the set of features [14]. Hence, partial user preference similarity can be described on the basis of hypotheses and hypotheses space. We denote $w = |\overline{I}_{a,q}|$ as the number

of items in $\overline{I}_{a,q}$. We define partial user preference similarity $\partial sim(U_a, U_b|h_{a,q})$ as the similarity of user U_a's qth preference hypothesis $h_{a,q}$ and the user U_b's hypotheses space $\overline{H}_b$ respectively user U_b's user model $h_b(I)$:

$$\boxed{\partial sim(U_a, U_b|h_{a,q}) \equiv sim\big(h_{a,q}, h_b(I)\big)}$$

To compare the hypotheses $h_{a,q} \in \overline{H}_a$ with $q \in \{1, \ldots, |\overline{H}_a|\}$ of user U_a's preference model with the preference model $h_b(I)$ of user U_b, all hypotheses $h_{a,q}$ of user U_a's preference model have to be extracted. For each hypothesis $h_{a,q}$, an item set $\overline{I}_{a,q}$ is built with the items that matches $h_{a,q}$'s conjunction of constraints of item features. Note, that the item set I is partitioned into item sets $\overline{I}_{a,q}$ with $q \in \{1, \ldots, |\overline{H}_a|\}$ and are disjoint. That is because each item is associated with one rating concept by a well-defined hypothesis. The hypothesis $h_{a,q}$ associates every item $I_j \in \overline{I}_{a,q}$ with exactly one concept $c_k \in C$. We define the partial similarity of user U_a's hypothesis $h_{a,q}$ and user U_b's user preference model $h_b(I)$ as the probability P, that the user preference model $h_b(I)$ predicts rating concept c_k under the condition that the hypothesis $h_{a,q}(I)$ associates all items $I \in \overline{I}_{a,q}$ with c_k:

$$\begin{aligned} sim\big(h_{a,q}, h_b(I)\big) &\equiv P\big(h_b(I) = c_k \wedge h_{a,q}(I) = c_k\big) \\ &\equiv P\big(h_b(I) = c_k | h_{a,q}(I) = c_k\big) P\big(h_{a,q}(I) = c_k\big) \end{aligned}$$

The hypothesis $h_{a,q}$ associates all items with c_k that matches $h_{a,q}$'s conjunction of constraints of item features. Hence, the probability of $P(h_{a,q}) = 1$. Therewith, we can simplify the upper definition:

$$sim\big(h_{a,q}, h_b(I)\big) \equiv P\big(h_b(I) = c_k | h_{a,q}(I) = c_k\big), \forall I_j \in \overline{I}_{a,q} \tag{3.3}$$

3.6 Collaborative Filtering based on Partial User Model Similarity

Similar to Section 3.4, partial preference similarity can be used to predict for user U_a the rating $\hat{r}_{aj}$ of item I_j. To this goal, the proper hypothesis $h_{a,q}$ has to be identified that is chosen by the user U_a's preference model $h_a(I_j)$ given the item I_j. The well-defined hypothesis $h_{a,q}$ is identified by matching the conjunction of constraints of features with the item's feature vector. Thereafter, the similarity of $h_{a,q}$ and the user models $h_b(I)$ of other users U_b can be computed. Note, that all partial user preference similarities can be computed offline to provide a rating predictions online. The similarity $sim\big(h_{a,q}, h_b(I)\big)$ between two people is computed on the merged item set $\overline{I}_{h_{a,q}}$ that matches the hypothesis $h_{a,q}$ and have been rated by at least one of both people. The rating $\hat{r}_{aj}$ is the sum of U_a's average rating $\overline{r}_a$ and the normalized sum of the weighted difference of all the other user U_b's rating r_{bj} and their average ratings $\overline{r}_b$:

$$\hat{r}_{aj} = \overline{r}_a + \kappa \sum_{b \neq a}^{n} sim\big(h_{a,q}, h_b(I)\big) * (r_{bj} - \overline{r}_b) \tag{3.4}$$

The sum is normalized to 1 with the normalization factor κ.

4 Evaluation

In this section, we provide an empirical evaluation of our two proposed methods to compute user preference similarities and partial user preference similarities. First, we describe the used dataset and second, we describe the experimental settings. We close discussing the comparison of our approach with others.

4.1 Dataset

We created a movie dataset that consists of user ratings and movie descriptions. We used the *Netflix-Prize* dataset [4] that originally consists of 17 000 movies, 480 189 users and 100 480 507 ratings from a 1 to 5 integer scale. We enriched this dataset with movie informations from the Internet movie database *IMDb*. We built a movie ontology based on the *IMDb* movie descriptions and domain knowledge. The movie ontology consists among other things of 28 genres, 10 different movie awards including nominations, movie color information (i.e. black&white, colored) and 240 countries. We used this information to build the hypotheses space and to generate movie feature vectors to learn the user preference models.

Both datasets provide partially different movie informations like year releases or titles. Furthermore, movie titles tend to be used by several movies. Thus, we unified movie titles of both datasets. We defined a movie of one dataset identical to a movie dataset of the other dataset if the unified titles are identical and the difference between the year releases is minimal. We made the assumption that movies with the same title are not produced and published within several years. With this method, we automatically identified 10 128 Netflix movies in the *IMDb* with 83 029 805 ratings of 479 437 users. In a precedent data analysis, we measured the average number of ratings per user of 173.2 and the median of 80 ratings. The average rating is 3.56 and the median is 4.

4.2 Experimental Setting

To evaluate our approach we used two different settings: the first setting consists of users with few common rated items and the second setting consists of users with many common rated items. We assume that users with few ratings have less common rated items then users with many ratings. Hence, we selected randomly 500 users with 50 ratings and 500 users with 200 ratings for the first and second setting, respectively. We chose 50 ratings to have a reasonable amount of ratings to learn user preferences from and 50 are below the median and the average. We chose users with 200 ratings because it is more then the median or the average and thus a reasonable amount. We split both datasets in a training set and a test set with a ratio of 4:1.

For the partial user preference model approach *pUMsim*, we used the machine learning algorithm *Part* [7] that obtains rules from partial decision trees. Because every rule is also a hypothesis, the hypotheses extraction is relatively simple. For the model-based similarity approach we used *Part* and *SVM* to test if the applied machine learning algorithm has a significant impact on the computed similarity.

We compared our approach with three collaborative filtering approaches. We used Pearson correlation [15, 5] to measure the global user preference similarities among users. The other two approaches learn user preference models respectively classifiers. Both predict the item ratings for a specific user by classifying movies with the learned user models based on *Part* and *SVM*, respectively. We used the implementations of *Part* and *SVM* from the Weka library [17].

4.3 Comparison

We evaluate all approaches by measuring the *root mean square error* (RMSE) and *mean absolute error* (MAE) as suggested in [9]. MAE measures the average absolute error between the user's rating and the predicted rating. In contrast, the RMSE measures the average squared error and extracts the root. Hence, the RMSE metric is sensitive on very bad recommendations. Further, we calculate the precision, recall and the F1-measure. The precision describes the percentage of relevant items of all the set of predicted relevant items. The recall describes the percentage of all predicted relevant items compared to all relevant items. The F1-measure combines the precision and recall value to one single value. For this purpose, we classified items into two groups: relevant items with high ratings and irrelevant items with low ratings. We used the threshold of 3.5 to classify items as relevant or irrelevant.

The evaluation results are presented in Table 1. We tested the significance with a non parametric test for dependent samples because the results are not normal distributed. We applied the Wilcoxon signed-ranks test and tested all methods pairwise. Therefore, we applied Bonferroni correction to control the *family-wise error*. We set $\alpha = 0.01$ as significance level.

In Table 1 and the setting with few common rated items (50 rat./user), both classifiers based on *Part* and *SVM* perform worst regarding RMSE and MAE. In case of RMSE and MAE, computing user preference similarity based on user preference model similarity (*UMSim*) and partial user model similarity (*pUMSim*) significantly outperform the user similarity based on Pearson correlation. That is because Pearson correlation is computed on common rated items that are rarely found in this setting. In contrast, the user model similarity is computed on all rated items of the users and thus, is based on more examples. However, *UMSim* significantly outpferoms *pUMSim* regarding MAE, but does not regarding RMSE. We assume that the hypotheses partition the set of items into too small sets such that partial user preference model similarity is less accurate compared to user preference model similarity. No significant difference exists regarding precision, recall and the F1-measure in this setting.

In the setting with many common rated items (200 rat./user), both classifiers still perform significantly worst regarding RMSE and MAE. In contrast to the previous setting, *UMSim* based on SVM is not significantly better then Pearson correlation, but both are significantly better then *UMSim* based on Part regarding RMSE. However, they are not significantly better then *UMSim* based on Part regarding MAE. In this setting, *UMSim* and Pearson correlation are significantly better then *pUMSim*. Regarding precision, the classifier based on SVM performs

Table 1. Evaluation with the setting of 50 and 200 rated items per user

Setting	Algorithm	RMSE	MAE	Precision	Recall	F1-Meas.
50 rat./user	pUMSim (*Part*)	1.097698	0.898961	66.70%	68.23%	67.46%
	UMSim (*SVM*)	1.077945	0.889020	68.27%	68.74%	68.05%
	UMSim (*Part*)	1.075843	0.885730	67.82%	68.34%	68.08%
	CF (*Pearson Corr*)	1.131921	0.929923	65.76%	68.30%	67.01%
	SVM Classifier	1.309146	0.976800	65.07%	71.17%	67.98%
	Part Classifier	1.334507	1.003800	65.05%	70.98%	67.89%
200 rat./user	pUMSim (*Part*)	1.048786	0.843029	60.90%	66.83%	63.73%
	UMSim (*SVM*)	1.035611	0.835009	60.77%	67.33%	63.88%
	UMSim (*Part*)	1.032746	0.833374	60.89%	67.31%	63.94%
	CF (*Pearson Corr*)	1.035324	0.832373	60.56%	68.71%	64.38%
	SVM Classifier	1.230682	0.896450	58.54%	67.80%	62.83%
	Part Classifier	1.292360	0.953600	58.76%	64.72%	61.60%

significantly worse compared to others. The classifier based on Part performs significantly worse then other approaches only globally but not with the Bonferroni correction. In case of recall, Pearson correlation significantly outperforms all approaches. Our proposed approaches together with Pearson correlation significantly outperform the two classifiers only without Bonferroni correction.

In general, our proposed approaches perform similar to traditional collaborative filtering approaches. However, they significantly outperform traditional collaborative filtering in settings with few common rated items.

5 Conclusion

We proposed a probabilistic method to compute user preference similarities based on user preference models and partial user preference similarities based on hypothesized user preferences that are extracted from learned user preference models. We have empirically shown that user similarity based on preferences is significantly better then user similarity based on common rated items for users with few common rated items. However, the accuracy of partial user preference similarity is promising but needs further improvement. Note, the learned user models might not be total functions and the user model comparison incomplete. But we argue that these missing items are irrelevant because they do not match the user's preferences.

Due to user-user comparison, our proposed approach is suitable for small amount of users because it has order of $O(n^2)$ time complexity. Therefore, in future work, we investigate clustering and filtering to improve scalability. Further, we will improve the accuracy of the learned user preference models. For this purpose, we will increase the hypotheses space by applying more features and enrich them with background knowledge from domain ontologies. In a next step, we will consider regression-based machine learning algorithms to improve the rating predictions and refine the currently suggested probabilistic similarity

with a more semantical interpretation of what similar ratings are. In addition, we will consider global user preference similarity as background evidence for partial user preference similarity. In a last step, we will investigate how to prune hypotheses such that partial preference similarity relies on more examples. That leads to higher evidence of the accuracy of partial preference similarity. We will investigate how background knowledge in form of a domain ontology can be used to prune hypotheses since the features are related to instances in the movie ontology.

6 Acknowledgements

We would like to Thank Jörg-Uwe Kietz for his insightful comments on the ideas.

References

1. S. S. Anand, P. Kearney, and M. Shapcott. Generating semantically enriched user profiles for web personalization. *ACM Transactions on Internet Technology*, 2007.
2. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. In *Communications of the ACM*, 1997.
3. C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI*, 1998.
4. J. Bennett and S. Lanning. The netflix prize. *KDD Cup and Workshop*, 2007.
5. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in AI*, 1998.
6. I. Cantador, A. Bellogín, and P. Castells. A multilayer ontology-based hybrid recommendation model. *AI Communcations*, 2008.
7. E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *15th International Conference on Machine Learning*, 1998.
8. N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI /IAAI*, 1999.
9. J. L. Herlocker, J. A. Konstan, L. G. Reveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Sys.*, 2004.
10. D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.
11. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, 2002.
12. S. E. Middleton, H. Alani, and D. C. de Roure. Exploiting synergy between ontologies and recommender systems. In *WWW*, 2002.
13. S. E. Middleton, N. R. Shadbolt, and D. C. de Roure. Ontological user profiling in recommender systems. In *ACM Transactions on Information Systems*, 2004.
14. T. M. Mitchel. *Machine Learning*. 1997.
15. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW*, 1994.
16. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.
17. I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques*. 2005.