Predicting the Performance of Collaborative Filtering Algorithms

Pawel Matuszyk
Otto-von-Guericke-University Magdeburg
Universitätsplatz 2
D-39106 Magdeburg, Germany
pawel.matuszyk@ovgu.de

Myra Spiliopoulou
Otto-von-Guericke-University Magdeburg
Universitätsplatz 2
D-39106 Magdeburg, Germany
myra@iti.cs.uni-magdeburg.de

ABSTRACT

Collaborative Filtering algorithms are widely used in recommendation engines, but their performance varies widely. How to predict whether collaborative filtering is appropriate for a specific recommendation environment without running the algorithm on the dataset, nor designing experiments? We propose a method that estimates the expected performance of CF algorithms by analysing only the dataset statistics. In particular, we introduce measures that quantify the dataset properties with respect to user co-ratings, and we show that these measures predict the performance of collaborative filtering on the dataset, when trained on a small number of benchmark datasets.

Categories and Subject Descriptors

Information Systems [**Recommender Systems**]: Data Mining—*Collaborative Filtering*

General Terms

Algorithms, Performance, Measurement

Keywords

Recommenders, Recommender Performance Prediction, Collaborative Filtering, Matrix Factorization

1. INTRODUCTION

The family of collaborative filtering (CF) algorithms is widely used in recommender systems, because of their accuracy and ability to cope with sparse data. However, CF performance varies strongly among different datasets, indicating that CF is appropriate for some but not all data. This gives raise to the question of deciding whether collaborative filtering is appropriate for a given dataset, without exposing the users of the recommendation engine to a potentially inappropriate algorithm and without running expensive simulations. In this study, we propose a method that predicts CF performance on a dataset without running nor simulating the execution of the algorithm.

Performance prediction for recommenders is challenged by the power laws governing recommendation environments. Hence, researchers investigate how well a recommender will assess the preferences of a given user (cf. [2, 3]), for whom little may be known. We rather study the dataset as a whole. Our method encompasses a visual analysis step, in which we identify properties that indicate whether a CF recommender has enough information for assessing user preferences. We then quantify these properties and incorporate the derived descriptors into an estimator that approximates the expected performance of CF algorithms on this dataset. In our study, we distinguish between neighborhood-based CF and non-negative matrix factorization, and we cover both types of CF recommendation.

The paper is structured as follows. In the next section we present related work. In section 3, we explain the visual analysis method and derive two properties a dataset should reveal for the CF algorithms to provide high-quality recommendations. In section 4, we present the measures that reflect and quantify those properties and show how we use them for prediction. In section 5 we report on our experiments on several datasets. In section 6 we conclude our work and discuss remaining open issues.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
WIMS'14, June 2-4 2014 Thessaloniki, Greece
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2538-7/14/06...$15.00.
http://dx.doi.org/10.1145/2611040.2611054

2. RELATED WORK

In [2, 3], Bellogin et al. investigate the prediction of the performance of recommender systems. In [2], they introduce a novel weighting scheme on the "clarity" of users. They show that this weighting scheme allows them to find good neighbours for a user and thus reduce prediction error, especially for small neighbourhoods. In [3] Bellogin et al. extended [2] with the concept of "query clarity" from the field of information retrieval. However, these methods are designed to improve the performance of a running system by predicting the error for a given user. Our goal is rather to predict the error of a recommender on an entire dataset prior to implementing and using it on operative data.

Griffith and O'Riordan [6] build regression trees on features derived from user data, including average rating, number of neighbours, average similarity to the top 30 neighbours, importance of a user etc. They predict the Mean Average Error (MAE) values using a decision rule derived from the regression tree. Then, they compare the MAE predicted by the rule to the MAE values computed by a collaborative filtering approach. This analysis delivers interesting insights on which user features are informative. However,

© Matuszyk 2014. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14) , http://dx.doi.org/10.1145/2611040.2611054

the computation of those features requires finding the neighbourhood of an active user. Therefore, the method is rather appropriate for predicting the performance of single users. A prediction for an entire dataset would be computationally very expensive, since it would require enumerating all users and aggregating the findings on them in a reasonable way.

Ekstrand and Riedl predict the performance of single recommendation algorithms to improve ensembles of methods [4]. They show that different types of recommenders make different errors and therefore, using ensembles can lead to overall performance improvement. To predict the behavior of the different recommender types, they introduce basic predictors, such as count, mean and variance of ratings. They show that these statistics correlate with performance, but stress that further features are needed to explain when a recommender works well [4]. In our work we create more sophisticated features and build a regression model that describes recommender performance well.

Some studies identify dataset descriptors that describe or predict interesting dataset properties. Niall and Rickard investigate ways of measuring sparsity and define six properties that a sparsity measure should have [7]; they conclude that only the Gini Index exhibits all of them. A further sparsity measure, especially designed for CF, is proposed in [1]. In our work, we also investigate sparsity; we use entropy-inspired measures [9] and apply them on user co-ratings.

3. VISUAL ANALYSIS

Our approach is designed for web platforms where users rate items. To judge the eligibility of CF methods, our approach only requires a dataset $\mathcal{D}$ with the users' ratings over a time horizon. It is possible to use as $\mathcal{D}$ the complete set of ratings recorded over a time period, but it is also possible to choose a random sample over the set of *users*, so that processing overhead is reduced. Sampling over the set of users is essential: if we sample directly over the set of ratings, it is likely that we mostly select users from the short head; these users are not representative of the dataset, hence they should not be used to estimate the eligibility of CF methods.

Once $\mathcal{D}$ is constructed, as next step we map the distribution of user ratings into the set of equivalence classes; we then inspect the co-rating structure of $\mathcal{D}$ using heatmaps. Through the juxtaposition of the co-ratings structure of different datasets, we demonstrate that the heatmap of co-ratings gives insights on CF eligibility, and we point to two properties that characterize datasets for which CF methods are appropriate. We quantify these properties in the next section and use them in our CF-eligibility estimator.

3.1 User Equivalence Classes and Co-Ratings

For our visual analysis of a set of ratings $\mathcal{D}$, we extract the set of users U who contributed the ratings, and partition U into equivalence classes with respect to the number of ratings a user has provided. In particular, let $u \in U$ be a user and let $R(u) \subseteq \mathcal{D}$ be the ratings provided by u. The "equivalence class" of u, denoted as $[u]$, is defined as:

$$[u] = \{u_x \in U || R(u_x)| = |R(u)|\} \quad (1)$$

where $|X|$ denotes the cardinality of an arbitrary set X. Hence, all users who gave the same number of ratings belong to the same equivalence class. We denote each equivalence class with the number of ratings of its users, i.e. the class $[u_8]$ encompasses all users who made exactly 8 ratings.

Using the notion of equivalence classes, we compute a *co-rating matrix* for $\mathcal{D}$. In particular, for each pair of users (u_x, u_y), we compute the number of ratings they have in common. Then, to each pair of equivalence classes $([u_x], [u_y])$ we set as *class co-ratings value* the average number of co-ratings among the users in these classes.

After computing the co-ratings matrix of $\mathcal{D}$, we visualize the heatmap of the co-ratings matrix. As shown in the example heatmap of Figure 1(a), classes are sorted by number of ratings. The darker the dot corresponding to a pair of classes is, the higher is the class co-ratings value for this pair. Since users with the same number of ratings are grouped into the same equivalence class, the class co-ratings values can provide insights on how sparse the co-ratings matrix is.

We propose to accompany the co-ratings matrix of $\mathcal{D}$ with the histogram of the cardinalities of the equivalence classes, as shown in the example Figure 1(b). Such a histogram delivers a more accurate interpretation of the heatmap, since it highlights the user classes of high importance, i.e. the highly populated ones. We elaborate on the insights gained by inspecting heatmaps and their corresponding class cardinality histograms on two example datasets.

3.2 Juxtaposing the Heatmaps of Datasets

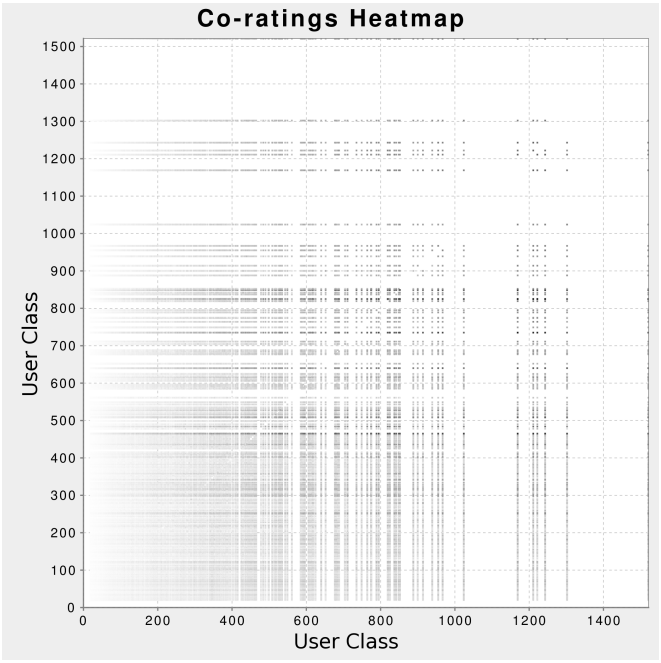
As a proof of concept, we depict the co-rating matrices and class cardinality histograms for two of the datasets we studied, MovieLens (Figure 1) and Epinions (Figure 2), using a a sample of 1000 users per dataset. The two co-rating matrices on Figures 1(a) and 2(a) are remarkably different.

In particular, the co-ratings heatmap of MovieLens on Figure 1(a) has many dark regions, which implies that in MovieLens there is a high number of co-ratings for many pairs of equivalence classes. The histogram (Fig. 1(b)) reveals a skewed distribution, as is typical for recommender systems, but it shows also that most of the users have contributed many ratings. In contrast, the heatmap of the Epinions dataset (Figure 2(a)) is almost empty, implying that there are almost no co-ratings. This is supported also by the corresponding histogram (Figure 2(b)), which reveals that most of the users have very few ratings.

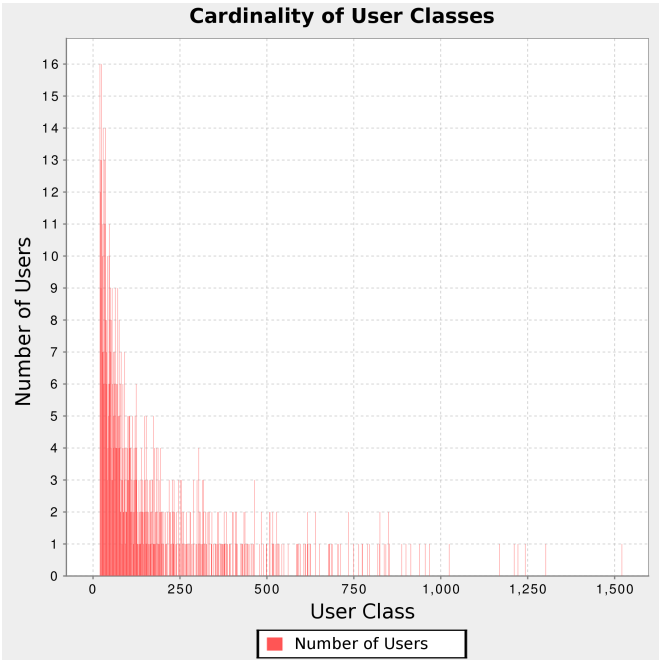
Figures 1 and 2 show that the two datasets MovieLens and Epinions, which are often used as benchmark datasets, are very different with respect to co-ratings distribution. Since CF algorithms assess the similarity of two users on the basis of the items they have both rated, we expect that CF algorithms will perform less well on the Epinions dataset than on the MovieLens dataset. However, to decide whether CF methods are appropriate for an arbitrary dataset $\mathcal{D}$, it is preferable to not only rely on the visual juxtaposition of $\mathcal{D}$ to MovieLens or Epinions. In the next section, we derive two properties that $\mathcal{D}$ should satisfy, so that CF algorithms are successful on it. We then derive measures from these properties and use them for CF-eligiblity estimation.

4. CF-PERFORMANCE ESTIMATION

CF algorithms rely on the co-ratings of users, so, intuitively, a co-ratings graph or matrix with *Low Sparsity* is important for high-quality recommendations. However, low sparsity of a dataset alone does not ensure that a high number of co-ratings can be found for all users. The distribution of co-ratings can be a skewed distribution i.e. *some users*, namely those who have rated a lot of items, are likely to be matched to other users as opposed to users with few ratings.

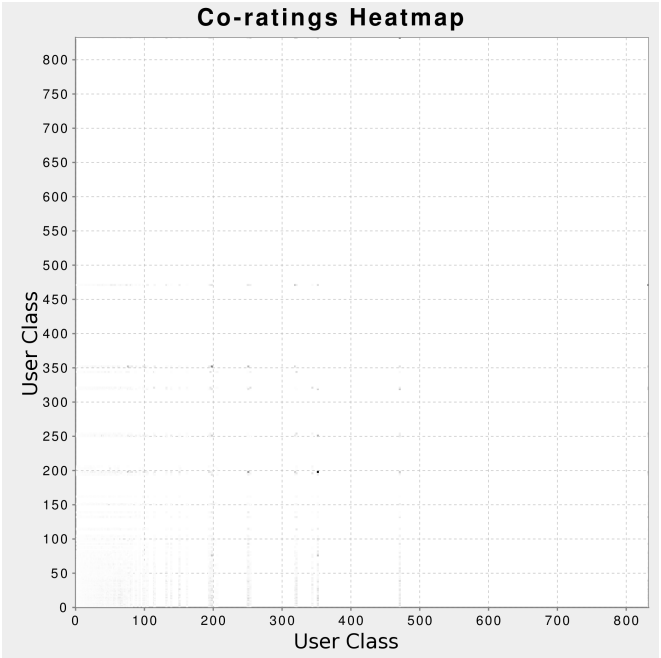


(a) Heatmap representing the co-rating structure of the MovieLens 1M dataset. Dark areas represent user classes with many co-ratings and white areas mean no co-ratings at all.

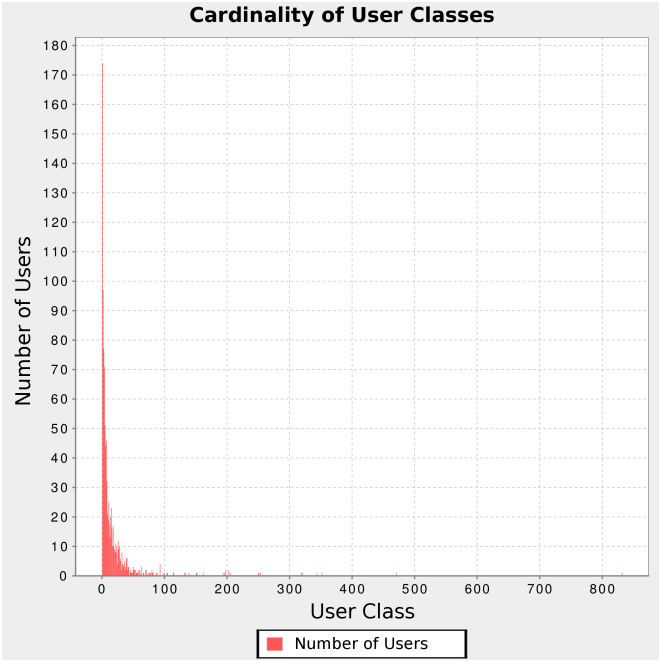


(b) Histogram of cardinalities of user classes. A typically for recommenders skewed distribution. The histogram allows to determine the biggest user classes.

Figure 1: Visualization of the Movie Lens 1M dataset.



(a) The Co-rating heatmap of the Epinions dataset shows an extreme sparsity (the heatmap is nearly blank). Lower performance of CF algorithms can be expected.



(b) Histogram of the Epinions dataset reveals an even more skewed distribution than MovieLens. The user classes with only few ratings are the biggest ones.

Figure 2: Visualization of the Epinions dataset.

Hence, it is reasonable to expect a better CF performance when the number of co-ratings of users follows a *Uniform Distribution*. We quantify the properties of *Low Sparsity* and *Uniform Distribution* and use them to learn an estimator of CF performance. This estimator can be applied on the dataset $\mathcal{D}$ without running CF algorithms on it.

4.1 Quantifying the Sparsity of Co-Ratings

To model the sparsity of a co-ratings matrix we use the following measure proposed in [1] (notation adjusted):

$$sparsity(\mathcal{D}) = 1 - \frac{\sum_{u \in U} |R(u)|}{|U| \cdot |Items|} \quad (2)$$

where, as before, U is the set of users recorded in $\mathcal{D}$, $R(u)$ is the set of ratings of user $u \in U$ and $Items$ is the set of items of the site, independently of whether they are referenced in $\mathcal{D}$ or not. The product of $|U|$ and $|Items|$ is the number of all possible ratings, under the assumption that a user can rate an item only once. The formula reflects the ratio of ratings that are "missing" out of the set of all possible ratings. It takes the value of one when no ratings are available, and the value of zero when no values are missing (complete matrix).

4.2 Quantifying the Distribution of Co-Ratings

To quantify how uniform is the distribution of co-ratings among user classes we use measures from the field of information theory. Let $\mathcal{U} = \{[u] | u \in U\}$ be the set of equivalence classes over the set of users U and $M_{X \times Y}$ the co-rating matrix as visualized e.g. in Figure 1(a) using a heatmap. $cor([u_x], [u_y])$ is then defined as the (x,y)-th entry of the matrix expressing the average number of co-ratings between classes $[u_x]$ and $[u_y]$, where $x \in \{1, ..., X\}$ and $y \in \{1, ..., Y\}$. (cf. dimensions of $M_{X \times Y}$). We need a measure that reaches its maximum when all equivalence classes are equiprobable. Possible measures are entropy ([9], notation adjusted):

$$E(\mathcal{D}) = -\sum_{x,y} \frac{cor([u_x], [u_y])}{\sum_{x,y} cor([u_x], [u_y])} log_2 \left(\frac{cor([u_x], [u_y])}{\sum_{x,y} cor([u_x], [u_y])} \right) \quad (3)$$

and Gini index (from [9], notation adjusted):

$$Gini(\mathcal{D}) = 1 - \sum_{x,y} \left(\frac{cor([u_x], [u_y])}{\sum_{x,y} cor([u_x], [u_y])} \right)^2 \quad (4)$$

Those measures are computed upon average number of co-ratings of user classes, where a uniform distribution is ideal i.e. all classes have the same number of co-ratings.

4.3 Building a CF-Performance Estimator

To predict the performance of CF on dataset $\mathcal{D}$, we use supervised learning. In particular, we run CF algorithms on benchmark datasets: for each dataset X, we derive an instance that contains the $sparsity(X)$ (cf. Eq. 2), the $Entropy(X)$ (Eq. 3) and the $Gini(X)$ (Eq. 4) value for the ratings in X, as well as the RMSE value for each CF algorithm on X (target variable). Thus, for each CF algorithm A under study, we derive one training set, from which we can learn a predictor ξ_A. Then, the expected RMSE value of A for $\mathcal{D}$ is the predictor's output $\xi_A(\mathcal{D})$.

Albeit it is inherently attractive to predict the performance of a CF algorithm through a learner, it is even more appealing to study how the variables of the training set predict the CF performance. To this purpose, we use Pearson correlation coefficient as measure: if there is a strong linear correlation between the input variables and the target variable, then linear regression can be used to express this correlation as a formula (cf. Section 5.3 for training $\xi_A(\mathcal{D})$).

5. EXPERIMENTS

We investigate how well our predictor with the proposed measures predicts the performance of different CF algorithms. As proof of concept we consider two CF algorithms, User-Based Collaborative Filtering with cosine similarity (UB-CF) and SVD++ [8] (denoted as MF for "Matrix Factorization" hereafter), as implemented in [5]. Our training dataset consists of four (!) instances, one per dataset with ratings.

We first describe how we build the training dataset: in 5.1.1, we show how we tune UB-CF and MF, since fair evaluation requires that each algorithm runs with the best parameter setting; in 5.1.2, we elaborate on the values of our measures for the four datasets. Then, in subsection 5.2, we discuss the values of the Pearson coefficients, identifying linear correlations. Accordingly, we perform linear regression (cf. 5.3) to come up with a formula that returns the expected performance of UB-CF and MF on an unknown dataset.

5.1 Building the Training Dataset

We use four benchmark datasets: MovieLens1M and Epinions (also described in subsection 3.2), Netflix and Flixter. Since it is essential to create comparable training instances, and since the benchmark datasets differ in size, we sampled 1000 users from each one randomly. As explained at the beginning of Section 3, random user sampling approximately retains the original distribution of equivalence classes of users with respect to their sizes, while random sampling of ratings would distort the ratings distribution and increase the sparsity of the co-ratings matrix.

The ratings distributions in the four benchmark datasets vary strongly, as can be seen in the juxtaposition of MovieLens1M and Epinions in subsection 3.2. Selecting dissimilar datasets is essential, because the training instances must be representative of the variety of recommender platforms.

5.1.1 CF-Algorithm Tuning with Grid Search

Since each benchmark dataset has its idiosyncrasies, the optimal setting of parameters for the CF-algorithms UB-CF and MF vary from one to the other. For our training instances, we use the optimal parameter settings per algorithm and dataset. To tune the parameters optimally, we perform a grid search over the parameter space. The grid search does not guarantee finding an optimal solution, but provides a good approximation of optimal settings. To increase the stability of the results we performed a 5-fold cross validation on each run within the grid search.

Figure 3 depicts the RMSE values achieved by UB-CF and MF on each dataset, when using the best parameter settings per dataset. It is apparent that MF, i.e. SVD++, performs consistently better on all datasets.

5.1.2 Creating the Training Instances

We compute the $sparsity(X)$ (cf. Eq. 2), $Entropy(X)$ (Eq. 3) and $Gini(X)$ (Eq. 4) for each of the four benchmark datasets, and we associate this tuple with the RMSE value achieved by UB-CF, respectively MF (cf. values on Figure 3). Thus, we produce one training dataset with four instances for each of our two selected CF algorithms.

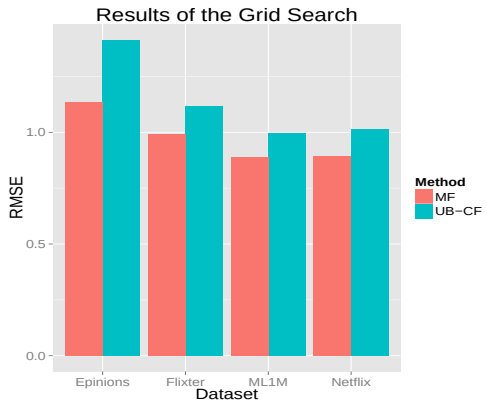


Figure 3: RMSE values of UB-CF and MF on the four benchmark datasets, as achieved by the best parameter settings found with grid search; lower values are better, and MF outperforms UB-CF in all cases.

5.2 Correlation of Measure Values to the Performance of the Selected CF Algorithms

To decide whether we can fit a linear regression model on our data, we first check for correlations between the three measures, corresponding to three variables (sparsity, entropy, gini) in our training dataset, and the target variable RMSE. We do so for both algorithms, UB-CF and MF, so there are two correlation coefficients, one per algorithm. The results are presented in Table 1. Both negative and positive correlations are relevant, therefore, we have to consider only absolute values of the correlation coefficient.

The Pearson's product moment correlation coefficients, presented in the second and third column of Table 1, show that there is a strong correlation between our measures and the results achieved by the CF algorithms, whereby Gini and Entropy reveal a stronger linear correlation than Sparsity.

In the last two rows of Table 1, we show the correlation between combinations of our measures and the target variable. Best absolute correlation (0.99693) is reached by a multiplicative combination of sparsity and Gini Index, namely $(1 - Gini) \cdot Sparsity$, which shows a stronger correlation with the RMSE value of UB-CF than any of its factors.

The computed coefficients are based on a training set with only four instances. We therefore perform a significance testing, shown in the last three columns of Table 1: the last two columns depict the p-values and the fourth column the respective alternative hypothesis [1]. Except for the correlation coefficients of sparsity, all remaining coefficients are statistically significant at significance level 0.03 and lower (marked in red), which proves that our measures describe the performance of the two representative CF-algorithms and can be used for performance prediction.

5.3 Performance Predictor

We derive a performance predictor on the basis of our training dataset by selecting the derived measure which achieves the strongest correlation and using it as basis for a linear regression model. For this training dataset, the model is:

$$RMSE = \alpha \cdot (1 - Gini) \cdot Sparsity + \beta \quad (5)$$

and the regression lines are shown in Figure 4.

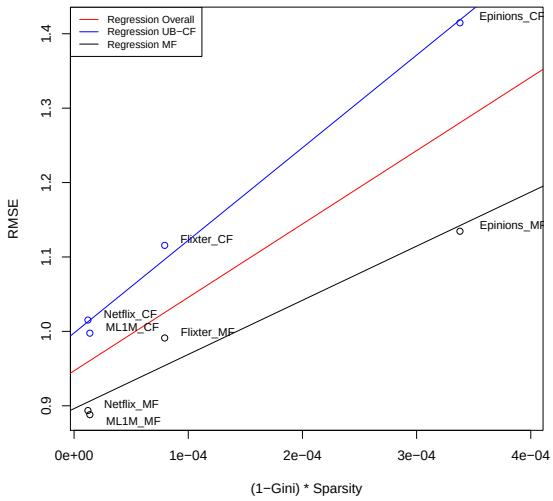


Figure 4: Linear Regression for UB-CF (blue) and MF (black): the index $(1 - Gini) \cdot Sparsity$ describes well the performance of MF and even better that of UB-CF on the training dataset.

We see in Figure 4 how the regression lines generalize the RMSE values of MF (black line) and UB-CF (blue line). All training instances are captured well. The performance of UB-CF is captured better than that of MF, but both regression lines show the same trend and relative positions of the training instances. Since all of these lines show similar tendencies, we conclude that the relation between our measure and the results of different algorithms is approximately the same. They differ only by a vertical shift, denoted as β in Eq. 5. The red regression line is computed on all instances and delivers a (rougher) prediction of RMSE for an arbitrary CF-algorithm.

As a small experiment on the predicted performance of our regression-based predictor, we learn a regression model on one of the two CF algorithms and measure how well it predicts the performance of the other algorithm. In particular, we check whether the difference between the predicted and real values is explainable just by a vertical shift. For this purpose, we use following linear regression formula:

$$RMSE = \alpha \cdot (1 - Gini) + \beta \cdot Sparsity + \gamma \quad (6)$$

In Table 2 we show the values learned for each parameter – once using UB-CF, once using MF for learning. In Table 3 we present the Pearson's product moment correlation between predictions made by linear regression learnt on UB-CF and MF separately and between the real RMSE values. In the first row of the table we see a linear regression learnt on the RMSE values of the UB-CF algorithm. The regression was learnt using the formula 6. The second column depicts the correlation of the predicted values to the

[1] P-values reflect the probability of obtaining the coefficients by chance (null Hypothesis: H0). If the probability is lower than a given significance level, H0 can be rejected.

Table 1: Pearson's product moment correlation coefficients between our measures and the target (RMSE) for UB-CF and MF: all coefficients except sparsity are significant at level lower than 0.03 (marked in red).

Measure	Correlation with RMSE		Alternative	p-value	
	UB-CF	MF	Hypothesis	UB-CF	MF
1-Gini	0.9969276652	0.9760339396	true correl > 0	0.001536	0.01198
Entropy	-0.9488311629	-0.9849657734	true correl < 0	0.02558	0.007517
Sparsity	0.737570808	0.7795095148	true correl > 0	0.1312	0.1102
(1-Gini) · Sparsity	**0.9969300691**	0.9758969096	true correl > 0	0.001535	0.01205
Entropy · Sparsity	-0.9409525839	-0.9733224195	true correl < 0	0.02952	0.01334

Table 2: Regression parameters, learned on the training data

Method	α	β	γ
UB-CF	1158.0332	0.925	0.1004
MF	621.7	1.13	-0.2

real values achieved by UB-CF. The correlation coefficient is here 0.99967. More importantly, the predictions also highly correlate with RMSE values of MF (0.98455 marked in red) that were not used for training the regression model. Similarly, the predictions made by regression trained on MF highly correlate with the real values of UB-CF (0.99649, in blue). All correlation coefficients are statistically significant at significance level lower than 0.01 (p-values in Table 3).

Table 3: Pearson's product moment correlation coefficients of RMSE predictions with real values.

Regression on	UB-CF		MF	
	Corr.	p-value	Corr.	p-value
UB-CF	0.99967	0.000167	0.98455	0.00773
MF	0.99649	0.00175	0.98768	0.00616

6. CONCLUSIONS

Collaborative filtering algorithms are widely used in recommendation engines, yet their performance varies with the properties of the dataset they are applied on. Until now, in order to asses how eligible a CF algorithm is, given a dataset, it was necessary to implement an algorithm and evaluate it in a series of experiments. We developed a method that predicts the performance of CF algorithms on an unknown dataset, using two measures of the data properties and a training sample derived from benchmark datasets.

Our approach encompasses computing the sparsity of the co-ratings matrix and the distribution of co-ratings among the users' equivalence classes. We use these scores to train an estimator on benchmark datasets, and then apply the estimator on the unknown dataset. In our experiments, we have shown that the performance of two CF algorithms (conventional user-user collaborative filtering and SVD++ [8]) can be described by a linear regression model that uses our scoring functions. In the future, we want to study the predictive power of our method on further CF algorithms.

Our method is based on training with benchmark datasets. Presently, the method is designed for datasets with rating range between 1 and 5. Since there are not many such datasets (four datasets known with this range), our training dataset contains only four instances. Albeit we have shown that our estimator can approximate the performance of CF-algorithms, it is preferable to use larger training datasets. To this purpose, we intend to investigate whether benchmark datasets with different rating ranges can be used together, although rating behaviour might vary, if the user can assign a score between 1 and 100, instead of 1 to 5. Nevertheless, the results presented on our small dataset show a high statistical significance.

7. REFERENCES

[1] D. Anand and K. Bharadwaj. Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. *Expert Syst. Appl.*, 38(5):5101–5109, 2011.

[2] A. Bellogín and P. Castells. A Performance Prediction Approach to Enhance Collaborative Filtering Performance. In *ECIR*, volume 5993 of *LNCS*, pages 382–393. Springer, 2010.

[3] A. Bellogín, P. Castells, and I. Cantador. Predicting the Performance of Recommender Systems: An Information Theoretic Approach. In *ICTIR*, volume 6931 of *LNCS*, pages 27–39. Springer, 2011.

[4] M. D. Ekstrand and J. Riedl. When recommenders fail: predicting recommender failure for algorithm selection and combination. In *RecSys*, pages 233–236. ACM, 2012.

[5] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A Free Recommender System Library. In *RecSys 2011*, 2011.

[6] J. Griffith, C. O'Riordan, and H. Sorensen. Investigations into user rating information and predictive accuracy in a collaborative filtering domain. In *SAC*, pages 937–942. ACM, 2012.

[7] N. P. Hurley and S. T. Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009.

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *14th ACM SIGKDD*, pages 426–434. ACM, 2008.

[9] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*, chapter 4, pages 158–164. Addison Wesley, 2006.