How the Distribution of the Number of Items Rated per User Influences the Quality of Recommendations

Michael Grottke
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Lange Gasse 20
90403 Nuremberg, Germany
E-mail: Michael.Grottke@fau.de

Julian Knoll
Technische Hochschule Nürnberg
Georg Simon Ohm
Hohfederstraße 40
90489 Nuremberg, Germany
E-mail: Julian.Knoll@th-nuernberg.de

Rainer Groß
Technische Hochschule Nürnberg
Georg Simon Ohm
Hohfederstraße 40
90489 Nuremberg, Germany
E-mail: Rainer.Gross@th-nuernberg.de

***Abstract*—With an ever-increasing amount of information made available via the Internet, it is getting more and more difficult to find the relevant pieces of information. Recommender systems have thus become an essential part of information technology. Although a lot of research has been devoted to this area, the factors influencing the quality of recommendations are not completely understood.**

This paper examines how the quality of the recommendations made by collaborative filtering recommender systems depends on the distribution of the number of items rated per user. Specifically, we show that its skewness plays an important role for the quality attained by an item-based collaborative filtering algorithm.

***Keywords*—classification accuracy, collaborative filtering, item-based recommender system, simulation study, skewness.**

I. Introduction

Globalization and growing specialization have made the process of obtaining information more and more complex. Considering the Internet and the huge amount of information provided over it, the availability of certain information is often foiled by the flood of data, making it difficult or impossible to find the information of interest. Recommender systems were invented to counteract this problem by providing a user with few pieces of information that might be relevant for him or her, selected from the mass of data.

Collaborative filtering (CF) is one way to come up with recommendations. This approach computes recommendations based on data describing the attitude (e.g., ratings) or the behavior (e.g., mouse clicks) of users with respect to items. Such a data set is referred to as a user-item matrix (UI matrix), because it can be represented as a table with users in rows and items in columns. Using the available data, a CF algorithm selects a list of items that are potentially of interest to a specific user.

It is hardly surprising that the quality of such recommendations depends on the data used to compute the recommendations. Cacheda et al. [1] showed that the density of the sparse UI matrix (i.e., the proportion of filled-in cells) has an influence on the quality of recommendations: increasing the density leads to a gain in quality, which is plausible, because the recommendations can then be based on a larger amount of information.

978-1-4673-7328-9/15/$31.00

In a current project, we are taking a broader approach to investigating the influence of the input data on the performance of various recommender algorithms. For example, even if the density of the UI matrix is the same for two data sets, the distribution of the filled-in cells within the matrix might differ substantially. Understanding how this distribution affects the quality of recommendations may help to select an appropriate algorithm for a given data set, and it might guide the development of new algorithms that are able to cope with specific situations.

In this paper, we focus on the *skewness* of the distribution of the number of items rated per user. We make the following main contributions. First, we present an approach for generating data sets with different skewness of this distribution from actual observations. Second, we show how receiver operating characteristics curves both for individual users and for an entire test data set can be determined in the simulation of item-based CF algorithms. Third, we carry out a simulation study demonstrating that the skewness plays an important role for the quality attained by such algorithms.

The remainder of this work is structured as follows. In Section II, we give an overview of the related work. The theoretical background concerning recommender algorithms and metrics for evaluating recommendation quality is presented in Section III. After describing the simulation study conducted in Section IV, we discuss its results in Section V. Concluding remarks and an outlook on our future work are contained in Section VI.

II. Related work

A. Overview of recommender systems

In their survey of CF techniques, Su and Khoshgoftaar [2] distinguished CF from content-based filtering and hybrid approaches.

CF uses the "wisdom of the crowd" to recommend items based on the assumption that if two users similarly value a set of items they will also value other items similarly [3]. Valuation can be measured either explicitly (typically by collecting user ratings) or implicitly (typically by monitoring user behavior, such as mouse clicks or the time spent on Web sites), and are then stored in the UI matrix [4]. CF techniques can further be split into neighborhood-based CF

(also known as memory-based or heuristics-based CF), and model-based CF [5]. Neighborhood-based approaches make use of the similarities either between the users (user-based CF) or between the items (item-based CF) in the UI matrix to generate item recommendations. Model-based CF algorithms first develop a model and then provide item recommendations using the parametrized model. Examples are Bayesian and regression-based CF [2].

In contrast to CF techniques, which use the similarities of user ratings for item recommendations, content-based recommender systems generate recommendations by finding matches between the properties of items.

Both approaches have disadvantages: While CF requires the UI matrix, content-based filtering needs information on the item properties [2]. Hybrid techniques, such as content-boosted CF [5], combine both approaches, thereby attempting to overcome their limitations. Moreover, they try to include every available information into the recommendation process [6] and also take advantage of more complex methods (e.g., tensor factorization, matrix factorization, factorization machines, graph-based approaches).

B. Quality of recommendations

Herlocker et al. [7] examined different metrics that can be used to measure the quality of recommendations. Employing 432 variations of recommender algorithms and their parameters on the MovieLens 100k dataset, they correlated eleven metrics over the recommendations. They identified three classes of metrics leading to similar results.

Cacheda et al. [1] compared the result quality of eleven recommender algorithms for different densities of the UI matrix. To this end, they made use of the MovieLens 100k data set as well as a subset of the Netflix data set and constructed training data sets with varying densities by sampling between 10% and 90% of the ratings. They found that the algorithms perform the better the denser the UI matrix is. However, model-based approaches are better able to cope with sparse conditions than memory-based ones.

Zaier et al. [8] examined the influence of the neighborhood size on the quality of a user-based CF algorithm. According to their results, the quality of recommendations improves with an increase in neighborhood size (in the following referred to as z). This research was conducted for the MovieLens, Netflix and BookCrossing data sets, which are close to a power-law distribution, and the Jester data set, whose distribution is not specified in the paper. They concluded that the quality of recommendations based on data sets following a power-law distribution converges more slowly with a dynamically growing neighborhood size than for data sets that follow some other distribution. Although the authors focused on the influence of the distribution of the number of items rated per user on the quality of recommendations, they neither explicitly specified the underlying distribution nor investigated the performance under different well-defined distributional conditions.

Steck [9] used eight metrics for analyzing the application of two recommender algorithms (namely, Allrank and MF-RMSE) to the rating prediction task and the ranking task. He concluded that the performance results for an algorithm depend less on the task or the metrics considered; rather, the distribution of the ratings taken into account plays a crucial role. Here, distribution relates to the proportion of ratings for items that the user deliberately chooses among the ratings for all items in the catalog.

C. Stability of recommender algorithms

Adomavicius et al. [10] defined the stability of a recommender system as the degree to which the predictions for items remain the same when new ratings which are in complete agreement with the prior predictions are submitted to the system. They investigated the stability of five memory-based and one factorization algorithm, varying data sparsity, the number of new ratings added, the rating distribution, and data normalization. In this context, "rating distribution" refers to the kind of scale used, such as ratings from 1 to 5 (as in the MovieLens and Netflix data sets) or from -10 to 10 (as in the Jester data set). The distribution of the number of items rated per user, which is the focus of our study, was not considered by these authors.

III. Theoretical background

A. Item-based collaborative filtering

For this first investigation into whether the distribution of the number of items rated per user influences the quality of recommendations, we decided to study the item-based CF algorithm. This algorithm is well-known, and it serves as the basis for implementations in professional environments [11]. Moreover, the fact that it is a rather simple algorithm makes it easier to comprehend and check the results obtained. However, our approach can easily be applied to other algorithms, and we will do so in our future work.

Item-based CF uses the similarity between items in the UI matrix to make predictions and recommendations. The UI matrix is a table consisting of n rows for the users $u_1, \ldots, u_n$ and m columns for the items $v_1, \ldots, v_m$. Its cell (i, j) contains the rating $w_{i,j}$ that user u_i gave for item v_j. Table I shows an example UI matrix for $n = 5$ users and $m = 6$ items. The ratings are on an ordinal scale from 1 (very bad) to 5 (very good); missing values are indicated with NA (not available).

TABLE I. Example UI matrix

	v_1	v_2	v_3	v_4	v_5	v_6
u_1	NA	1	NA	NA	4	2
u_2	NA	4	NA	5	5	5
u_3	2	3	4	4	5	NA
u_4	1	5	5	3	4	4
u_5	2	3	4	1	NA	2

1) Similarity calculation: In a first step, the similarities between all pairs of items need to be calculated. The most commonly used metrics are cosine similarity, Jaccard correlation [12], and Pearson correlation [2]. Which metric is reasonable in a given context depends, among other things, on the measurement scale of ratings. As the data used in the following study is ordinally scaled, the rank correlation coefficient due to Spearman [13, p. 141] can be applied. Spearman's correlation coefficient s between items v_j and v_l

is calculated in analogy with Pearson's correlation coefficient; however, it does not employ the original ratings of the items by the ith user, $w_{i,j}$ and $w_{i,l}$, but their corresponding rank values $r_{i,j}$ and $r_{i,l}$. Moreover, $\overline{r}_{v_j}$ and $\overline{r}_{v_l}$ represent the average rank values of the ratings received by the jth and lth item, respectively:

$$s_{v_j,v_l} = \frac{\Sigma_{i=1}^{n}(r_{i,j} - \overline{r}_{v_j})(r_{i,l} - \overline{r}_{v_l})}{\sqrt{\Sigma_{i=1}^{n}(r_{i,j} - \overline{r}_{v_j})^2}\sqrt{\Sigma_{i=1}^{n}(r_{i,l} - \overline{r}_{v_l})^2}}.$$

In calculating s_{v_j,v_l}, the ith user is only taken into account if s/he rated both v_j and v_l (i.e., if both $r_{i,j}$ and $r_{i,l}$ are available). If the ratings given by one user feature identical values (ties), average ranks can be employed.

Predictions and recommendations for a specific user are generated based on the determined similarities. This user is below referred to as the active user u_a.

2) Item-based prediction: For data with ordinal scale level an item-based prediction $\hat{w}_{a,l}$ of the rating of user u_a for item v_l can be calculated based on the following formula [2]:

$$\hat{w}_{a,l} = \frac{\Sigma_{j=1}^{m}(r_{a,j} \cdot s_{v_l,v_j})}{\Sigma_{j=1}^{m}|s_{v_l,v_j}|}$$

The summations are over all other items rated by user u_a.

3) Item-based recommendation: Top-N recommendations consist of the set of the N items considered to be most interesting for the active user. To this end, a ranking of the items is calculated, and the top N items are selected. The item-based top-N algorithm [14] uses as a basis the similarities between the items (see *1)* above), and it works in five steps:

1) It is assumed that a user can evaluate items only if s/he has previously "purchased" them. Hence, items that were not "purchased" by the active user u_a are omitted. These are those items v_j where the rating $w_{a,j}$ is not available (NA).
2) For each one of the remaining items v_j, the z most similar items are selected (where z is the neighborhood size). An item v_l is the more similar with v_j, the higher the similarity coefficient s_{v_j,v_l}.
3) Those items that the active user has already "purchased" are omitted from the list of z most similar items obtained in step 2), because these items are not to be recommended again.
4) The remaining items in all most-similar-items lists produced are then grouped by item, and for each item the sum of the similarities with the respective items already "purchased" by the user u_a are calculated.
5) Finally, all items are ranked in descending order based on the scores obtained in step 4). The first N items are selected as the list of top-N recommendations.

B. Metrics for evaluating recommendation quality

Herlocker et al. [7] distinguished between two main classes of metrics: those for evaluating the prediction accuracy, and those for evaluating the accuracy of classifications. The former metrics measure how close the predicted ratings from the recommender system are to the true user ratings. Mean absolute error (MAE) is frequently used for measuring prediction accuracy [7], but it is less appropriate for evaluating the task of finding "good items" [7]. In fact, this is exactly the task that we are concerned with in this article, because we are interested in measuring recommendation quality. To this end, metrics for the accuracy of the classification are appropriate.

TABLE II. CATEGORIZATION OF ITEMS BASED ON THEIR RELEVANCE AND THEIR RECOMMENDATION BY AN ALGORITHM

	not recommended	recommended	total
irrelevant	h_{00}	h_{01}	$h_{0\cdot}$
relevant	h_{10}	h_{11}	$h_{1\cdot}$
total	$h_{\cdot 0}$	$h_{\cdot 1}$	h

These metrics are ratios of correct and incorrect decisions about the item recommendations of a recommender system [7]. In particular, we make use of the metrics precision and recall as well as the receiver operating characteristics (ROC) curve and the precision recall curve (PRC). To explain these concepts, we consider Table II, in which a total of h items are categorized based on their actual relevance to a user and the fact whether or not they have been recommended (i.e., whether or not relevance to the user has been assumed) by a recommendation system.

1) Precision and recall: Precision is defined as the ratio of relevant and recommended items (h_{11}) to all recommended items ($h_{\cdot 1}$). For example, if an item-based top-20 recommendation algorithms selects 10 relevant items, precision equals $h_{11}/h_{\cdot 1} = 10/20 = 0.5$.

Recall, also known as the true positive rate, is the proportion of relevant and selected (recommended) items (h_{11}) among all relevant items ($h_{1\cdot}$). For example, if the above-mentioned item-based top-20 recommendation algorithm selects 10 relevant items, while a total of 50 items are relevant for the user, recall equates to $h_{11}/h_{1\cdot} = 10/50 = 0.2$.

Cremonesi et al. [15] described in detail a possible approach for implementing precision and recall in the context of the Netflix data set, containing 100 million movie ratings on a scale from 1 star to 5 stars. When applying these metrics to the specific problem, the key question to be answered is: "Which items are relevant to a user, and which are not?" Cremonesi et al. considered a movie to be relevant for a specific user if this user gave it a 5-star review. In the simulation study conducted by these authors to evaluate the precision and recall of various recommender algorithms, they focused on *one* actual 5-star rating per user, and assessed the accuracy of the respective algorithm in classifying this movie as relevant. Precision and recall could thus be assessed both for each individual user and – by averaging the individual results – for the entire data set.

2) ROC curve and PRC: In the context of classification in general and recommender systems in specific, the ROC curve is a graphical representation of the trade-off between the true positive rate (i.e., the recall) on the y-axis and the false positive rate (i.e., the percentage of recommended items among the irrelevant ones, $h_{01}/h_{0\cdot}$) on the x-axis. From a predicted ranking of items produced by a top-N recommender algorithm, the ROC curve can be created following the approach described by Herlocker et al. [7]. Starting at the origin, a line segment of length $1/h_{1\cdot}$ is drawn vertically if the first-ranked

item is indeed relevant; otherwise, a horizontal line segment of length $1/h_0$. is drawn. This is repeated for all other items in the ranked list. Basically, this approach amounts to setting N equal to one and subsequently increasing it, comparing the true positive rate and false positive rate of the related top-N algorithm at each step.

A perfect recommender will produce an ROC curve consisting of a vertical line from the origin to (0,1) and a horizontal line from (0,1) to (1,1). In contrast to this, random recommendations that do not separate between relevant and irrelevant items will lead to an ROC curve that lies on the bisecting line of the diagram. Therefore, the area under the ROC curve (area under curve, AUC) is a measure of the quality of the recommendations. The better the classification ability of the recommender, the larger the AUC. The AUC value can be interpreted as the probability that a relevant item is actually classified as such.

Similar to drawing the ROC curve, it is possible to visualize the trade-off between precision and recall. The resulting diagram is referred to as the precision recall curve (PRC).

It should be noted that the approach by Cremonesi et al. [15], described above, does not allow to produce meaningful ROC curves and PRCs for individual users. As these authors only consider one relevant movie for each user, the recall attained by a recommender algorithm is either zero or one.

IV. Simulation study

A. Data set

The majority of publications on recommender algorithms is based on two data sets: MovieLens and Netflix. Both data sets contain movie ratings on a scale from 1 star to 5 stars. While the Netflix data set was originally published for a contest between 2006 and 2009, it is currently not available from any public source. However, MovieLens is still providing different data sets on its website, with a number of ratings ranging from 100 thousand to 20 million (100k, 1M, 10M and 20M).

Due to the better availability of the MovieLens data, we decided to use it for conducting our research work, selecting the MovieLens 10M data set, the largest one available at that time. In early April 2015, after our simulation study had already been carried out and when we were just about to finalize this paper, MovieLens published its 20M data set. However, we are confident that the overall results of our simulation study would not have been different if we had based it on this larger data set.

The 10M data set used here contains about 10 million ratings from 71.5 thousand users, relating to about 10 thousand movies. All users selected by MovieLens had rated at least 20 movies. Unlike the 100k and 1M data sets, the 10M MovieLens data set does not offer any demographic information about the users.

B. Basic setup

Our simulation study was aimed at investigating how the distribution of the number of items rated per user influences the performance of the item-based CF algorithm. Its basic setup is depicted in Figure 1.

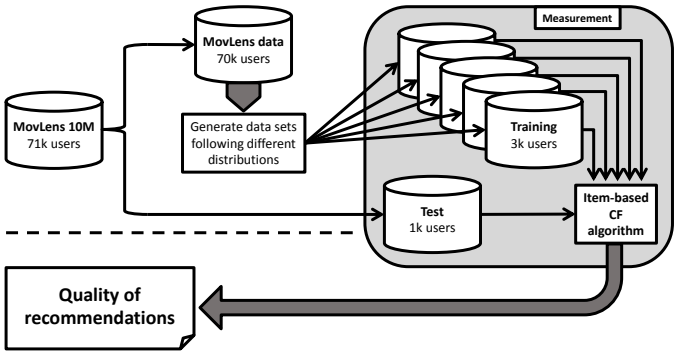

Fig. 1. Setup of simulation study

Throughout our analyses, the same test data set was used. We generated it from the MovieLens 10M data set by randomly sampling 1000 users from those users who gave at least five movies a 5-star rating.

All of the training data sets, always consisting of 3000 users, were sampled from the rest of the MovieLens 10M data set.

We conducted a total of seven simulations, each one concerned with a specific theoretic distribution of the number of items rated per user (see Section IV-C). These seven theoretical distributions did *not* imply differences in the expected density of the UI matrix, unlike in the work by Cacheda et al. [1]. Within each simulation, we sampled 20 different training data sets from this distribution, as explained in Section IV-D. The item-based CF algorithm was carried out for each training data set to create recommendations for every user in the test data set. The quality of the resulting recommendations can be determined for each individual training set. Also, the results for the 20 training data sets used in one simulation can be combined, allowing a more accurate measure of the performance of the item-based CF algorithm for a certain underlying distribution of the number of items rated per user.

C. Determining the theoretical distributions

We used the distribution of the number of ratings per user in the original MovieLens 10M data set as a starting point for specifying the seven different distributions employed in our simulation study. While each user in the data set rated at least 20 items, as mentioned above, most discrete distributions, such as the negative binomial distribution (see [16, pp. 199–235]), cover the domain $\mathbb{N}_0^+$. However, we found that the number of ratings per user X could well be modeled with a *location-shifted* negative binomial distribution with probability mass function

$$P(X = 20 + y) = \frac{\Gamma(y + k)}{\Gamma(k)y!} p^k (1 - p)^y, \quad y \in \mathbb{N}_0^+, \qquad (1)$$

and parameters $0 < p < 1$ and $k > 0$. Since the expected value of this distribution is given by (cf. [16, p. 207])

$$\mu_X = \frac{k(1 - p)}{p} + 20, \qquad (2)$$

the maximum likelihood estimates obtained from fitting the MovieLens 10M data set, $\hat{p} = 4.5445 \cdot 10^{-3}$ and $\hat{k} = 0.5620$,

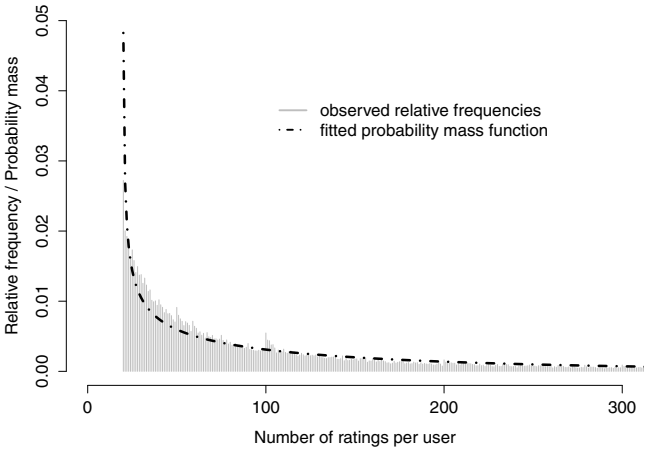

Fig. 2. Observed relative frequencies of the number of ratings per user, and probability mass function of the fitted negative binomial distribution

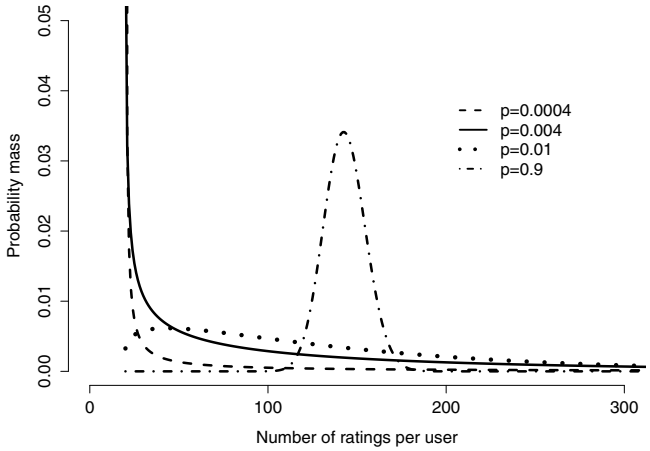

Fig. 3. Probability mass functions of different negative binomial distributions

implied an estimated mean of $\hat{\mu}_X = 143.1$; on average, every user rated about 143 movies. The relative frequencies of the various values of X as well as the fitted probability mass function (drawn as a single curve to allow comparison in one diagram) are shown in Figure 2.

All 20 data sets sampled in one simulation were based on the same location-shifted negative binomial distribution. Between the simulations, the parameter p was adapted to change the characteristics of the distribution, such as its skewness (see below). However, for all distributions simulated we kept the expected value of X (and hence the expected density of the UI matrix) fixed at the value obtained for the MovieLens 10M data set. As is easily seen from Eq. (2), this could be achieved by setting parameter k in accordance with the currently-chosen value of p:

$$k = \frac{123.1 \cdot p}{1 - p}. \tag{3}$$

While the expected value of X was thus kept constant, the skewness of the distribution of X, measured by its third standardized moment $\sqrt{\beta_{1,X}}$ [16, p. 208], varied as p was changed:

$$\sqrt{\beta_{1,X}} = \frac{2 - p}{\sqrt{k(1 - p)}} = \frac{2 - p}{\sqrt{123.1 \cdot p}}.$$

Since $\sqrt{\beta_{1,X}} \to +\infty$ as $p \to 0$, while $\lim_{p \to 1} \sqrt{\beta_{1,X}} = 1/\sqrt{123.1} = 0.0901$, varying p in $(0, 1)$ allowed us to generate both distributions that are extremely right-skewed and almost symmetric distributions (as well as intermediate cases). Specifically, for the seven distributions used in our simulations p was set to 0.0004, 0.004, 0.008, 0.01, 0.04, 0.5, and 0.9. The probability mass functions of four of these distributions are depicted in Figure 3 (again using curves rather than bars representing individual probability masses, making it possible to draw them in a single diagram).

It should be noted that although μ_X was fixed, the skewness is not the only metric of the distribution depending p in our approach. For example, its variance (see [16, p. 207]) varied with p as follows when setting k according to Eq. (3):

$$\sigma_X^2 = \frac{k(1 - p)}{p^2} = \frac{123.1}{p}.$$

D. Sampling from a theoretical distribution

To generate a training data set (approximately) following the desired theoretical location-shifted negative binomial distribution from the reduced MovieLens 10M data set (minus the test data), we sampled 3000 users (without replacement) from the 70k users, using unequal sampling probabilities. The basic approach to determining these sampling probabilities was as follows. For each specific realization x of the number of items rated per user X, the probability mass assigned according to Eq. (1) was equally allocated among those users in the data set who had rated exactly x items.

However, according to this probability mass function all integer values greater than or equal to 20 occur with a non-zero probability, while the MovieLens 10M dataset only contains users with between 20 and 7359 items rated. Moreover, even for x values below 7359, there are many cases in which none of the users in the data set rated exactly x items.

Let x_{no} denote such an x value. For each x_{no}, we re-assigned the related probability mass to the neighboring x values for which the data set actually contained observations. More exactly, we split the probability mass $P(X = x_{no})$ between the next-lower and the next-higher x values observed, taking into account their distances to x_{no}. The probability of the next-lower value $P(X = x_{no,l})$ and the probability of the next-higher value $P(X = x_{no,h})$ were thus increased by adding

$$P(X = x_{no}) \cdot \left(1 - \frac{x_{no} - x_{no,l}}{x_{no,h} - x_{no,l}}\right) \quad \text{and}$$
$$P(X = x_{no}) \cdot \left(1 - \frac{x_{no,h} - x_{no}}{x_{no,h} - x_{no,l}}\right),$$

respectively.

TABLE III. THEORETICAL AND AVERAGE EMPIRICAL MOMENTS

p	Avg. $\hat{\mu}_X$	σ_X^2	Avg. $\hat{\sigma}_X^2$	$\sqrt{\beta_{1,X}}$	Avg. $\sqrt{\hat{\beta}_{1,X}}$
0.0004	153.6	307750	190841	9.011	6.247
0.004	142.1	30775	29330	2.844	2.770
0.008	142.9	15387	15665	2.007	2.000
0.01	142.6	12310	12396	1.794	1.781
0.04	142.6	3077	3175	0.883	0.915
0.5	142.4	246.2	277.9	0.191	0.198
0.9	142.6	136.8	164.2	0.105	0.107

Based on this approach we are able to use a real-world data set to approximate drawing a sample from a different theoretical distribution. Table III compares moments of the theoretical distributions with the averages of the respective empirical moments, calculated from the 20 training data sets in each simulation.

Obviously, the greatest differences are observed for $p = 0.0004$. This is likely caused by the extreme skewness of the related negative binomial distribution, which implies that a sample from this distribution should include many users who submitted exactly 20 ratings, as well as a rather large number of users who rated a huge number (like several thousands) of items. However, the latter kind of users is quite rare in the MovieLens 10M data set, which explains why the real data set is not able to perfectly accommodate simulating the desired theoretical distribution. Nevertheless, the empirical measures differ significantly from the ones obtained for the simulations of the next theoretical distribution (with $p = 0.004$), which shows that the goal of simulating a highly-skewed distribution has partially been successful.

For the rest of the considered distributions the average empirical measures are rather close to the theoretical ones. For example, all averages of the estimated expected value ($\hat{\mu}_X$) agree nicely with the fact that we fixed the expected value of all theoretical distributions at $\mu_X = 143.1$.

E. Simulating the application of the recommender algorithm

In our simulation study, the application of the item-based CF algorithm is simulated in the following steps:

1) For each user u_a in the test data set, five randomly-selected 5-star ratings are set to "not available" (NA). Similar to Cremonesi et al. [15], we consider these items to be actually relevant for the user, because of his/her 5-star rating.
2) Moreover, we sample 995 items from all those items not rated by u_a (and thus considered irrelevant), and mix them with the five relevant items from step 1).
3) Employing the item-based CF algorithm, we rank the 1000 test items using the item-based similarities resulting from each training data set. As a result, we obtain a rank for most of the 1000 test items. Sometimes few items cannot be ranked due to a lack of data precluding the calculation of all similarities needed.

F. Metrics of recommender quality considered

To evaluate the quality of the recommender task "finding good items", we make use of the ROC curve, the AUC and the PRC. It should be noted that unlike Cremonesi et al. [15] we are able to create meaningful ROC curves (or PRCs) for individual users, because in our simulations the number of items actually relevant to each user that are to be identified by the recommender algorithm is five rather than one.

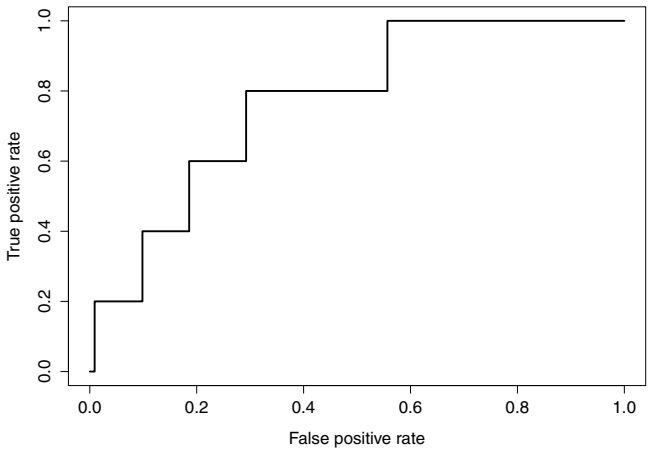

Fig. 4. ROC curve for an individual user

To plot an ROC curve for an individual user after simulating the application of the recommender algorithm described in the last section, we make use of the ranks obtained in step 3). Based on these ranks, we follow the generic approach by Herlocker et al. [7], described in Section III-B. Let us assume that 1000 items were recommended, with the relevant items taking ranks 10, 100, 188, 295, and 559. Starting from the origin, we draw 9 horizontal line segments of length $1/1000$ (which is identical to one horizontal line segment of length 0.009). For our first relevant item at rank 10, we then draw a vertical line segment of length $1/5$. Next, we draw 89 horizontal line segments of length $1/1000$ (i.e., one horizontal line segment of length 0.089), followed by a vertical line segment of length $1/5$ for the second relevant item, and so on. Figure 4 shows the corresponding ROC curve.

This ROC curve for an individual user is a non-decreasing step function. Since our test data set consists of 1000 users, we can aggregate the individual ROC curves to one ROC curve representing the recommendation performance for the whole training data set. Repeating this process for each of the 20 training data sets simulated based on one theoretical distribution, results in 20 ROC curves. These curves can again be aggregated to obtain an overall ROC curve, which embodies aspects of the recommender quality of the recommendation algorithm for the given theoretical distribution of the number of items rated per user; Figure 5 shows the curves generated in our simulations with $p = 0.004$.

PRCs can be generated analogously to ROC curves. The PRC for the individual user already considered in the above example can be seen in Figure 6. For $p = 0.004$, the PRCs for all 20 training data sets and the overall PRC are depicted in Figure 7. The different curves for the various training data sets (see Figure 5 and 7) do not feature substantial variability. The curves obtained for the other theoretical distributions of the number of items rated per user show a similar picture. This

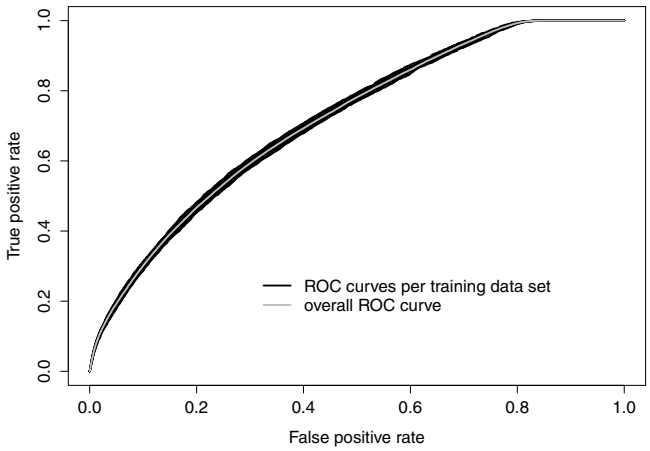

Fig. 5. ROC curves for the training data sets, and overall ROC curve

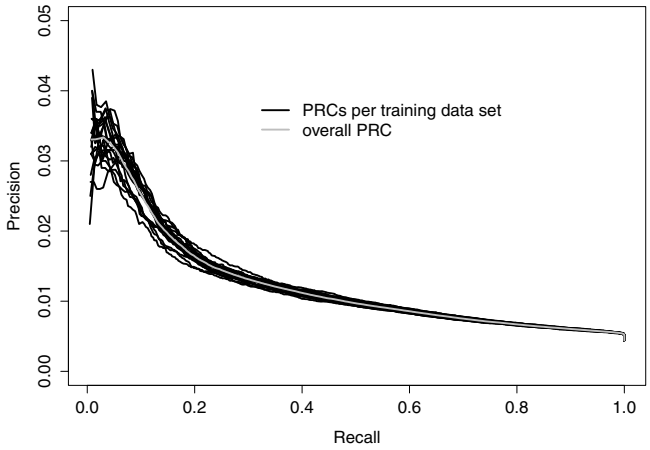

Fig. 7. PRCs for the training data sets, and overall PRC

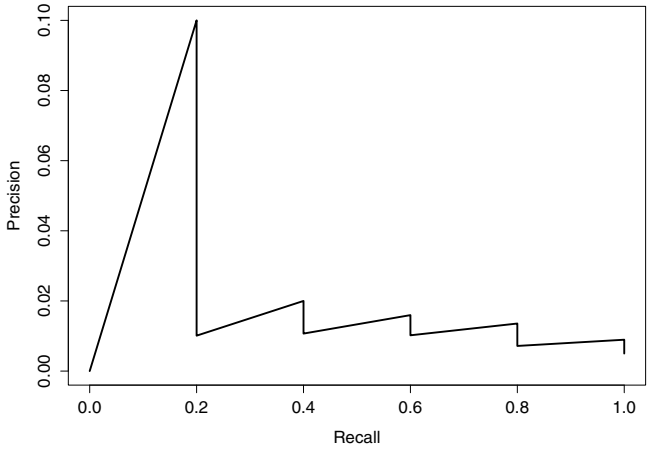

Fig. 6. PRC for an individual user

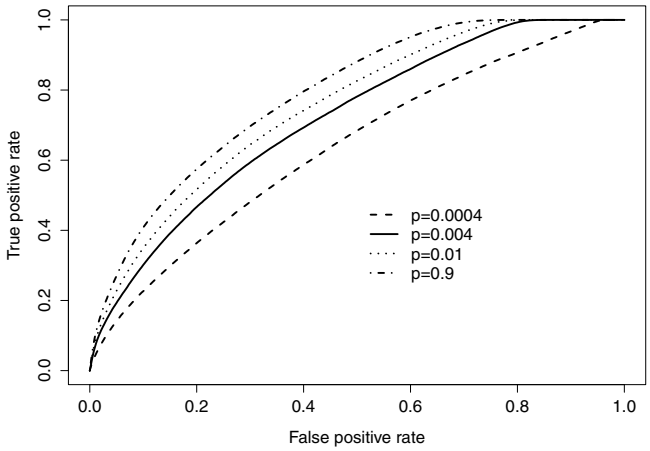

Fig. 8. ROC curves for four theoretical distributions

indicates that simulating 20 data sets from each distribution suffices for drawing valid and meaningful conclusions about the performance of the recommender algorithm based on the aggregated curves.

As mentioned above, a recommender algorithm may not be able to rank each relevant item. This occurs when the relevant item v_j rated by the user u_a has not been "purchased" by any other user in combination with other items "purchased" by u_a. As a consequence, the similarities required cannot be calculated, and the corresponding item is not rankable.

To capture the extent of the inability of a recommender algorithm to rank a relevant item, we propose the metric no recommendation rate (NRR), which represents the fraction of relevant items that could not be ranked. It is possible that for a specific underlying distribution of the number of items rated per user a recommender algorithm has a high NRR, while its predictive performance as measured by AUC is very good. Therefore, these metrics complement each other in the evaluation of the behavior of an algorithm.

V. Results

Figure 8 shows the overall ROC curves obtained for four of the theoretical distributions of the number of items rated per user. Moreover, for all seven distributions studied the AUC is listed in the second column of Table IV. As can be seen, with increasing values of p the area under the ROC curve tends to increase as well. This seems to be a quite reasonable result.

TABLE IV. AUC and NRR for all seven theoretical distributions

p	AUC	NRR
0.0004	63.65%	2.62%
0.004	71.71%	7.55%
0.008	73.72%	9.14%
0.01	75.01%	9.37%
0.04	77.70%	11.14%
0.5	78.76%	11.62%
0.9	78.69%	11.70%

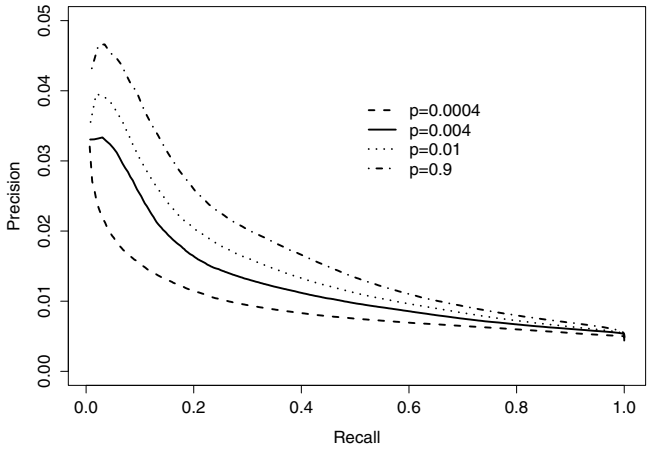

Fig. 9. PRCs attained for four theoretical distributions

Remember that the lower the value of the parameter p, the more right-skewed the distribution is. Generally-speaking, right-skewed distributions have a lot of users who rated few items and a few users who rated a larger number of items. In this case, the similarity between two items calculated from the data is less meaningful because the number of users who co-rated these items tends to be low.

In contrast, for a symmetric distribution there are many users with a moderate number of rated items, which implies a larger number of co-ratings. As a consequence, there is a broader basis for the calculation of the similarity coefficients, and the recommendations are of a higher quality. Interestingly, the AUC calculated for the distribution with $p = 0.9$ is slightly lower than the one obtained for $p = 0.5$. As can be seen from Table III, there is a relatively small difference in the skewness of these two distributions. For these almost-symmetric distributions the higher variance of the one with $p = 0.5$ might be beneficial.

Figure 9 shows the PRCs for the same four data sets as in Figure 8. The result is quite similar to the one for the ROC curves. For distributions with a higher value of p the item-based CF algorithm tends to perform better than for distributions with a lower value of p.

However, it should be noted that there are also drawbacks of a symmetric distribution. The NRR values listed in the third column of Table IV indicate that a larger value of p (related to a more symmetric distribution) is associated with a higher NRR. This implies that the ability of the item-based CF algorithm to come up with any ranking for relevant items is the lower the more symmetric the underlying distribution is. Under such a distribution, there are fewer users with an extremely large number of ratings than under a more right-skewed distribution. While this leads to a larger number of co-ratings per item, the proportion of items with at least one co-rated item tends to be lower. As a consequence, the recommendations made are generally of a better quality, but the percentage of relevant items being ranked at all is smaller.

VI. Conclusions and future work

In this paper, we showed that the density of the UI matrix is not the only aspect of the distribution of the number of ratings per user influencing the recommendation quality. Other aspects, such as the skewness of this distribution, play an important role as well. This suggests that better attention should be paid to the training data base of a recommender system and its underlying distribution.

The next steps of our project will be devoted to investigating how various recommender algorithms perform under different distributional conditions. Moreover, we will explore the question of how to develop algorithms that are capable of coping with specific situations, such as a high skewness. We are planning to employ the new MovieLens 20M data set for these analyses.

References

[1] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms," *ACM Transactions on the Web*, vol. 5, no. 1, pp. 1–33, 2011.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, Article ID 421425, 19 pages, 2009.

[3] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[5] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston: Springer, 2011, pp. 107–144.

[6] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys*, vol. 47, no. 1, pp. 1–45, 2014.

[7] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[8] Z. Zaier, R. Godin, and L. Faucher, "Evaluating recommender systems," in *Proc. 4th International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, 2008, pp. 211–217.

[9] H. Steck, "Evaluation of recommendations," in *Proc. Seventh ACM Conference on Recommender Systems*, 2013, pp. 213–220.

[10] G. Adomavicius and J. Zhang, "Stability of recommendation algorithms," *ACM Transactions on Information Systems*, vol. 30, no. 4, pp. 1–31, 2012.

[11] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

[12] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *IEEE Intelligent Systems*, vol. 50, no. 18, pp. 16–23, 2003.

[13] L. Fahrmeir, R. Künstler, I. Pigeot, and G. Tutz, *Statistik*, 2nd ed. Springer-Verlag, 1999.

[14] G. Karypis, "Evaluation of item-based top n recommendation algorithms," in *Proc. International Conference on Information and Knowledge Management*, 2001, pp. 247–254.

[15] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc.Fourth ACM Conference on Recommender Systems*, 2010, pp. 39–46.

[16] N. Johnson, S. Kotz, and A. Kemp, *Univariate Discrete Distributions*, 2nd ed. New York: Wiley, 1992.