Who likes it more? Mining Worth-Recommending Items from Long Tails by Modeling Relative Preference

Yu-Chieh Ho
Dept. of Computer Science and Information Engineering, National Taiwan University
Taipei, Taiwan
yuchieh.ho@gmail.com

Yi-Ting Chiang
Institute of Information Science, Academia Sinica
Taipei, Taiwan
yitingchiang@gmail.com

Jane Yung-Jen Hsu
Dept. of Computer Science and Information Engineering, National Taiwan University
Taipei, Taiwan
yjhsu@csie.ntu.edu.tw

ABSTRACT

Recommender systems are useful tools that help people to filter and explore massive information. While the accuracy of recommender systems is important, many recent research indicated that focusing merely on accuracy not only is insufficient to meet user needs, but also may be harmful. Other characteristics such as novelty, unexpectedness and diversity should also be taken into consideration. Previous work has shown that more the sales of long-tail items could be more beneficial to both customers and some business models. However, the majority of collaborative filtering approaches tends to recommend popular selling items.

In this work, we focus on long-tail item promotion and aggregate diversity enhancement, and propose a novel approach which diversifies the results of recommender systems by considering "recommendations" as resources to be allocated to the items. Our approach increases the quantity and quality of long-tail item recommendations by adding more variation into the recommendation and maintains a certain level of accuracy simultaneously. The experimental results show that this approach can discover more worth-recommending items from Long Tails and improves user experience.

Categories and Subject Descriptors

H.3 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL

Keywords

Recommendation Diversity, Recommender System, Aggregate Diversity, Long Tail, Collaborative Filtering

WSDM 2014, February 24–28, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2556195.2566589 .

1. INTRODUCTION

Recommender systems [26], as a filter and guide for users when dealing with huge volume of information, were expected to predict the user preference based on his/her profiles or previous behaviours. Most works and competition such as Netflix Prize aims to construct accurate recommender systems (i.e. minimize the difference between the predicted rating values and the real rating values). Although accuracy is one of the most important metric for recommender system evaluation, many recent research indicated that focusing merely on accuracy not only is insufficient to meet user needs, but also may be harmful. Other characteristics such as novelty, unexpectedness and diversity should also be taken in to consideration [11, 19, 27].

Long-tail items, or Long Tails are items located in the tail of the sales distribution [3]. Although different manufacturers and retailers may have diverse business strategies in driving sales toward the head or the tail of the sales distribution [6], more sales of long-tail items could be beneficial to both customers and some business models [5, 9]. For example, one of the most crucial niche of online book stores is the availability of books hidden in Long Tails and it is also a good way to develop customer loyalty by helping the user to discover more long-tail items that he/she does not know yet. In addition, "uniqueness" could be the key concern in clothing selection and [5] showed that providing more diverse recommendations increases the sales on long tail items. On the other hand, one study in the domain of people to people recommendation indicated that in the context of online dating, expression of interests (EOI) sent to unpopular users got higher success rate [23]. Another study also indicates that promoting less popular candidates can improves user success rates, while only recommending popular candidates results in a decreased rate [16].

However, the majority of collaborative filtering (CF) based algorithms tends to recommend popular items because the popular items, on average, are likely to get higher predicted ratings than less popular ones from both memory-based and model-based techniques for rating prediction [2][7]. Therefore, the long-tail items have very little chance to be seen by the users, while popular items occupy the majority of the recommendations, if the system generates recommendations that only depends on the predicted rating values. This disproportion can get worse and worse, and more items will be trapped in the long tail.

Aggregate diversity, which is a measurement of the distribution of recommendations over all items, has been considered as one of the important aspects of recommender system, since it has been shown that recommender systems which utilizes aggregate diversity are able to significantly associate recommendations with the sales of long-tail items [5, 7, 20].

In this work, we focus on long-tail item promotion and aggregate diversity enhancement, and propose a novel approach which diversifies the results of recommender systems by considering "recommendations" as resources to be allocated to the items. Compared to the classical framework of a recommender system (Figure 1a), where the results of recommendation depend only on the ordering of predicted rating values, our framework (Figure 1b) allows the service providers to re-organize the recommendations according to their strategies. In our framework, the preference prediction acts as the first step instead of being the final step of a recommender system. A good predicting algorithm can guarantee the accuracy, by acting as the role of rating predictor in Figure 1a and 1b. In the proposed approach, predicted rating values are also used in the recommendation organizer, showed in Figure 1b, which can guarantee the degree of aggregated diversity by allocating the "resources" (i.e. recommendations) based on the given allocation strategy and current user-item interactions (i.e. user rating values and predicted rating values). Our approach increases the quantity and quality of long-tail item recommendations by adding more variation into the recommendation approach and maintains a certain level of accuracy simultaneously.

The structure of the rest of the paper is as follows. We start with a review of related work on recommendation diversity and the matrix factorization technique for recommender systems in Section 2. Then, we describe the intuitive idea and the detailed methodology of our approach in Section 3. Section 4 presents and discusses the evaluation results on five metrics: average prediction (accuracy), coverage, gini coefficient, the quantity and quality of long-tail item recommendations that our system generated. Finally, we conclude our contributions and address potential directions for future work in Section 5.

2. RELATED WORK

In the following section, we review the matrix factorization technique which we used for rating value prediction, and several related work in the field of recommendation diversity.

2.1 Matrix Factorization

Matrix factorization (MF) is an effective model based CF approach in terms of rating value prediction and is famous for its success in Netflix Prize competition [14]. Given a $m \times n$ user-item matrix constructed by user rating records, MF technique captures the interactions between users and items by factorizing (decomposing) the user-item matrix into a $m \times k$ user-factor matrix and an $k \times n$ item-factor matrix where k is the number of latent factors. It characterizes both items and users by vectors of factors inferred from existing rating patterns and predict the unknown rating values by computing the inner product of those factors (i.e. user-factor matrix and item-factor matrix). MF is more accurate and more memory- efficient than the classical memory based CF approaches [15].

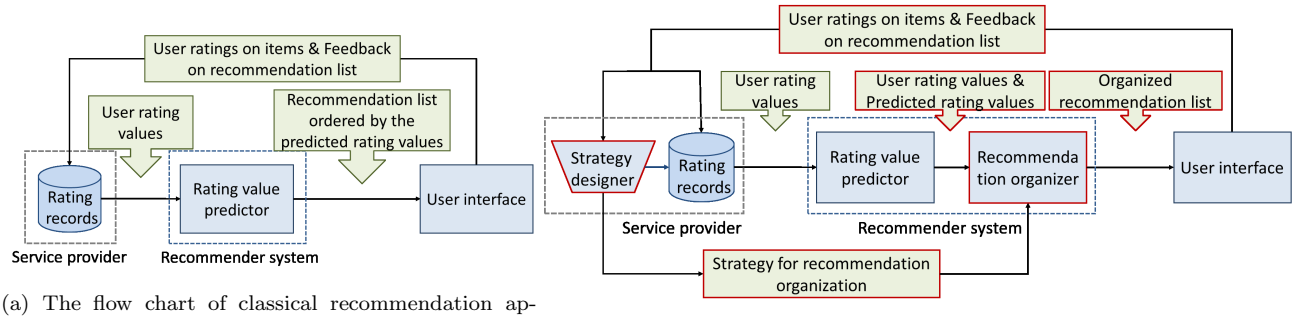
2.2 Recommendation Diversity

The diversity of recommendations has become one of the key dimensions in the research of recommender systems. Many researchers had worked on generating more diverse recommendations [4, 30, 29, 17, 12, 18, 2, 1, 16] and several studies on human factors and recommender systems [13, 19, 24] also suggested that it is necessary to maintain a certain level of diversity in the results of a recommender system even at the risk of accuracy loss. Generally speaking, we can classify the research on recommendation diversity into two categories: individual diversity and aggregate diversity.

The researches on individual diversity [4, 30, 29, 17] focused on the dissimilarity of items within a recommendation list to the individuals. [30] diversified the results of content based recommender systems by transforming the problem into a constrained optimization problem in quadratic form. The authors controlled the trade-off between similarity (to the user query) and diversity of the recommendation lists by a single parameter and proposed a solution for solving the optimization problem. [17] enhanced the individual diversity and novelty by building informative user profiles from user-item interactions and designing a hybrid mechanism which consists of accuracy-orientated, diversity-orientated, and novelty-orientated algorithms.

On the other hand, aggregate diversity focus on the coverage and the distribution of total recommendations among all items, especially the long-tail items with only few ratings. [22] and [21] improved the predictive accuracy and increased the recommendation of long-tail items by clustering the items according to their similarity on several derived variables (e.g. average rating values, number of ratings, etc) and computing the predictive rating values within each cluster. Different from most of the recommender systems which are triggered by the customers, [18] and [12] proposed an item-triggered recommendation approach to identify potential customers for long-tail items, which increases the recommendations on those cold sellers. This approach computes customer lists, ordered by the probability that the customers are willing to buy given items by solving a rare-class classification problem. In addition, [16] proposed a decision tree based approach to promoting the less popular candidates and showed that recommending less popular candidates can improve user success (the sender get reply from the message receiver) rates on online dating websites. Furthermore, [28] challenged the Long Tail Recommendation problem with a random walk based algorithm. The authors defined the concept of *absorbing time* to capture the longtailness of items and the relevance between users and items. Finally, [2] improved the aggregate diversity by re-ranking the item list according to given ranking strategies (e.g. the popularity of items and the average ratings of items), and [1] proposed a graph-theoretic approach which maximizing aggregate recommendation diversity by solving maximum flow or maximum bipartite matching problems.

In this work, we re-organize the recommendation lists by resource allocation, where recommendations are considered as resources, after the rating value estimation. We chose [2] and [1] as our benchmarks because their system share similar work flows and goals with our approach.



(a) The flow chart of classical recommendation approach.

(b) The flow chart of proposed approach. The red lines indicate the differences between the two approaches.

Figure 1: The framework of classical recommender system and the proposed approach.

3. METHODOLOGY

3.1 Game of Colors

The core idea behind our approach is "exploring individual users' unique taste" and we would like to give an intuitive analogy to our approach before digging into the technical details.

In this section, we make an analogy between item recommendation and pigment distribution in a painting class. Imagine that you are a teacher, and you want each of your student to complete a painting with the colors which show his/her unique taste. You design a game of colors to determine those colors and the two steps of this game are just a simplified version of approaches in Section 3.2.1 and Section 3.2.2.

First, every student gives a score between one and five to 36 colors. Next, you buy some color pigments. The quantity of each color is according to the ratings given by the students, the higher the ratings the more you purchase, and a minimum of one bottle for each color. (The idea of Section 3.2.1)

The purchased color pigments is then distributed to the students. In order to emphasize the uniqueness of each student, the distribution of the pigments will depend on comparison with the overall rating in the whole class. For example, suppose that the total amount of blue pigments and green pigments that you purchased are about the same, and one of the student has the score of 3.5 for blue and 3.7 for green. Although the student rates blue lower than green, the student will nevertheless receive more blue pigments than green pigments, if most of the other students rates higher than 3.7 for green, and lower than 3.5 for blue. Because in this case, blue is a more representative color regarding his/her unique taste.(The idea of Section 3.2.2)

After all 36 colors are distributed to each student, each student then sort their color pigments according to the amount they have received in descending order. Finally, the students can start to paint with only the top six colors, and can ask for more pigments if necessary.(top-N recommendation)

Therefore, this method of distribution depends on two factors:

1. The total amount of pigments in this color, which depends on the popularity of this color.
2. How much does the student like the color, compare with the others.

The first factor is decided by the ratings given to each color by the students in the first step, and the second factor is decided by comparing individual students' rating to the overall rating of the whole class. This color distribution will produce more diverse and individualized paintings than those that considers only the highest ratings given by the students.

3.2 Proposed Approach

The goal of our approach is to diversify the recommendations and maintain a reasonable level of accuracy by mining worth-recommending items from Long Tails. To achieve our goal, we compute a score named *5D-score* for each user-item pair, and the scores are used for re-organizing the recommendation lists. The role of 5D-score in our approach is just like the quantity of the color pigments that a student receives in the game of colors. We named it the "5D-score" because it is computed by considering the five "dimensions" that are critical to the evaluation of aggregate diversity and long-tail item promotion. The five dimensions are as follows:

1. Accuracy: the average predicted rating values of all recommended items.
2. Balance: the distribution of recommendations among all items after a round of recommendation (providing a top-N recommendation for each user).
3. Coverage: the percentage of recommended items among all items after a "round" of recommendation.
4. Quantity of long-tail item recommendation: the ratio of long-tail items among all recommended items.
5. Quality of long-tail item recommendation: the average predicted rating values of those recommended long-tail items.

Just like the "game of colors", our approach includes two sequential phases: the resource allocation phase and the recommendation phase (Figure 2). In the first phase, we consider the "opportunity of being recommended" as resources, and allocate them among all items according to some allocation strategies. In the second phase, each user receives a certain amount of resources from each item depending on how much the user likes a specific item, and the top-N recommendation list for a specific user is generated by the ranking of the amount of the resource he/she receives from each item.

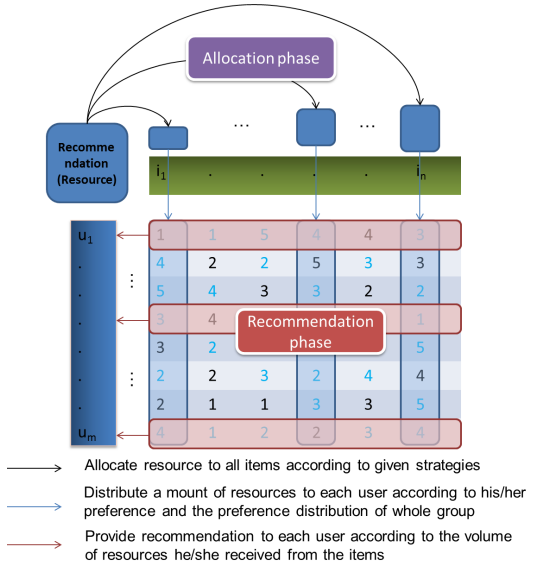


Figure 2: The diagram of allocation phase and recommendation phase.

The following section describe the details of the proposed approach and we summarize some notations here:

- $U = \{u_1, \ldots, u_m\}$: The set of users.
- $I = \{i_1, \ldots, i_n\}$: The set of items.
- I_u: The set of items rated by the user u.
- $R_u = \{r_{u,1}, \ldots, r_{u,n}\}$: The set of ratings. Where $r_{u,i}$ is the predicted or real rating of user u on item i.

3.2.1 Resource Allocation Phase

In the first phase, we would like to allocate certain quantity of resources on all items with the following objectives:

1. Items with higher rating values shall receive more resources (opportunity of being recommended) to guarantee the quantity of recommendations on those items with good reputations.
2. The allocation shall follow the first objective but in a balanced way. That is, non-popular items shall also receive reasonable amount of resources to keep their chance of being recommended.

Our approach begins with a complete $m \times n$ user-item matrix (Figure 2) where all the elements are filled by either ratings from the users (the black numbers) or predicted ratings (the blue numbers) computed by a recommendation algorithm (e.g. matrix factorization). We formulate the resource allocation approach as the following maximization problem:

$$\text{argmax}_{X=\{x_1,\ldots,x_n\}} \prod_{i \in I} x_i^{\text{allocation}(i)}, \tag{1}$$

where $x_1 + \ldots + x_n = k$, $k > n$ and $x_i \geq q > 0$.

allocation(i) is the allocation strategy to be plugged into the objective function and we want to find the n dimensional vector X (resource allocation among all items) which maximizes the objective function given an allocation strategy. We leverage the series multiplication as the fundamental form of our objective function because we would like to allocate the resources in a more balanced fashion (objective 2): Allocating all the resource to a single variable x_j would not be a proper solution for Equation 1 even when allocation(j) is larger than all the other indexes.

We define the allocation strategy as a function of i for flexibility and one can define his/her own function to facilitate other context. In this work, the allocation strategy is defined as:

$$\text{allocation}(i) = \sum_{u \in U} \frac{r_{u,i} - \overline{R_u}}{\sigma_u}, \forall i \in I, \tag{2}$$

where σ_u is the standard deviation of R_u and $\overline{R_u} = \frac{1}{n}\sum_i r_{u,i}$.

For each item i, Equation 2 is the summation of the normalized rating values and predicted rating values of each users (the z-scores of those ratings). As a result, allocation(i) models the preferences of all users on item i, and allocation(1), ..., allocation(n) model the distribution of preferences of all users over all items. By plugging this allocation strategy into Equation 1, the maximization process would assign larger values to the variables with larger indexes, which means the items with higher rating records will receive more (but not extreme) resources as we expected (objective 1).

3.2.2 Recommendation Phase

In the recommendation phase, we would like to distribute those resources that we have allocated on the items to each user according to their *relative preferences*, and then provide recommendations based on the results of distribution (i.e. 5D-scores.). The approach is designed to achieve the following objectives:

1. Improve the aggregate diversity by emphasizing the unique taste of individuals: The user who likes a specific item more than the other shall receive more resources (higher 5D-score) from this item.
2. Promote the long-tail items by giving them more chance to be seen by the user who likes them: When a user have similar preference on a popular item and a long-tail item, he/she is more likely to receive more resources from the latter.

The ratings and predicted ratings can represent users' preference, but they may contain rating bias. Therefore, we have to eliminate the rating bias by normalizing the rating values before the comparison. For example, although a movie was rated as three (out of five) points by a user named John, it could be a negative rating if John never gave a score lower than 3.5 to any movie. On the other hand, a rating of three points in John's rating record may denote a positive preference on that movie if he is very critical and most of his ratings are lower than 2.5. Equation 3 calculates the z-scores of the rating values of each user (row by row calculation on the user-item matrix) and models the preferences of all user on all items.

$$Z(u,i) = \frac{r_{u,i} - \overline{R_u}}{\sigma_u} \tag{3}$$

As mentioned in the beginning of section 3.2.2, we would like to distribute the resource according to the *relative preferences* of users on each item. The next step is to calculate

the 5D-score (Equation 4) based on the normalized preferences computed by Equation 3.

$$5\mathrm{D}(u,i) = \frac{Z(u,i) - \overline{Z(U,i)}}{\sigma_{Z(U,i)}} \times x_i, \quad (4)$$

where $\overline{Z(U,i)}$ and $\sigma_{Z(U,i)}$ are the mean value and the standard deviation of $Z(u_1,i),\ldots,Z(u_m,i)$ respectively. We use $\frac{Z(u,i)-\overline{Z(U,i)}}{\sigma_{Z(U,i)}}$ as the *relative preference* in this work.

Equation 4 is a "column by column" calculation on the user-item matrix. When we observe the 5D-scores within a single column j, they actually modeled the distribution of preferences from all users on item j. By Equation 3 and 4, we transformed the user preferences to *relative preferences* among all users and distributed the resources from items to users by calculating the product of those *relative preferences* and the quantity of resources $(x_1,\ldots,x_n)$ each item received. Finally, by sorting the items according to $\{5\mathrm{D}(u,1),\ldots,5\mathrm{D}(u,n)\}$, we generate the top-N recommendation for a user u.

Compared to the top-N lists generated merely according to the predicted rating values which are accurate but narrow and unbalanced, 5D-scores can help to promote the worth-recommending items in Long Tails and provide more diverse recommendations with sufficient accuracy.

The distribution of 5D-scores can help to promote the long-tail items (objective 2) because it tends to assign a higher value for a user from a cold seller (long-tail item) instead of a superstar (popular item) when this user has similar preference on the two items. Note that two z-scores with the same value (say $Z(u,i)$ and $Z(u,j)$) could have different meanings given that item i is a cold seller and item j is a superstar. For example, even user u have the same preference on the two items ($Z(u,i) = Z(u,j)$), $5\mathrm{D}(u,i)$ could still be higher than $5\mathrm{D}(u,j)$ because $Z(u,i)$ is relatively high among $\{Z(1,i),\ldots,Z(m,i)\}$ and relatively low when considering others' preferences on item j. In addition, this characteristic of 5D-scores also helps to diversify the recommendations because our approach emphasizes the unique taste of the users on the long-tail items (objective 1).

3.2.3 Accuracy Enhancement Algorithms

Providing recommendations according to 5D-scores can improve aggregate diversity and keep reasonable level of accuracy simultaneously. However, in some situations, we may need to keep a higher level of average predicted rating value. We have designed two algorithms which allow the user to balance the trade-off between accuracy and diversity. We will describe the two algorithms in this section.

The Rank by Rankings Algorithm, described in Algorithm 1, takes the advantages of two ranking strategies (original predicted rating values and 5D-scores). The algorithm ranks the items according to both strategies and generates two ranking results, then combines the ranking results together.

In line 5, we combine the two ranking results of predicted rating values and 5D-scores by multiplying two ranking indexes on each item and sort the items according to those products. Recall that for any two values $v_1 + v_2 = V$, we have $(\frac{V}{2})^2 \geq v_1 \times v_2$. This property helps us to capture those items with at least one extreme high ranking instead of those with two average rankings.

The goal of the Accuracy Filtering algorithm (Algorithm 2) is to pick the first N items that have higher predicted rating values than a threshold T from a list ranked by the 5D-score (or other ranking strategy). If there are less than N items that can fulfill this requirement, the algorithm will fill up the list with items that have the next-highest 5D-scores(line 3).

Algorithm 1: Rank by Rankings

Input : a: the active user.
$\{r_{a,i_1},\ldots,r_{a,i_n}\}$: R_u.
$\{5\mathrm{D}(a,i_1),\ldots,5\mathrm{D}(a,i_n)\}$: the 5D-scores.
N: length of recommendation list.
Output: Top-N recommendations for user a, $N \leq n$.
Data: $i.\mathrm{idx}_r$: raking index of item i according to R_u.
$i.\mathrm{idx\_5D}$: ranking index of item i according to 5D-scores.

1. Sort the items by $\{r_{a,i_1},\ldots,r_{a,i_n}\}$ in descending order.
2. Assign $i.\mathrm{idx}_r$ for each item i according to the current order.
3. Sort the items by $\{5D(a,i_1),\ldots,5\mathrm{D}(a,i_n)\}$ in descending order.
4. Assign $i.\mathrm{idx\_5D}$ for each item i according to the current order.
5. Sort the items by $\{1.\mathrm{idx}_r \times 1.\mathrm{idx\_5D},\ldots,n.\mathrm{idx}_r \times n.idx\_5\mathrm{D}\}$ in ascending order, and let the result be $\{i_{rr_1},\ldots,i_{rr_n}\}$.
`// New items are those items not in` I_a.
6. Top-N recommendations for user a is The top N new items in $\{i_{rr_1},\ldots,i_{rr_n}\}$.

Algorithm 2: Accuracy Filtering

Input : $\{i_1,\ldots,i_n\}$: sorted items (according to a given ranking strategy like 5D-scores or the results of Algorithm 1).
$\{r_{a,i_1},\ldots,r_{a,i_n}\}$: the real and predicted rating.
a: the active user.
T: the threshold of predicted rating value.
N: length of recommendation list.
Output: $L_{\text{Top-N}}$: Top-N recommendations for the active user, $N \leq n$.

1. Collect 2 sets of items I_c and I_r from $\{i_1,\ldots,i_n\}$, where $I_c = \{i | r_{a,i} \geq T, i \notin I_a\}$ and $I_r = \{i | r_{a,i} < T, i \notin I_a\}$ without loss the order.
2. If $|I_c| \geq N$, insert top N items of I_c into $L_{\text{Top-N}}$.
3. If $|I_c| < N$, insert I_c and top $(N - |I_c|)$ items of I_r into $L_{\text{Top-N}}$.

4. EXPERIMENTS

We conducted several experiments on MovieLens dataset [10] to show the effectiveness of our methods. We adopted three metrics: prediction, coverage, and gini coefficient to evaluate *accuracy*, *coverage* and *balance* of the recommendation results, and two metrics: long-tail recommendation and prediction on long-tail items to evaluate *quantity* and

quality of the recommended long-tail items. All the experimental results are computed under the setting of top-N recommendation. That is, the recommendation approaches will generate N best items for each user.

4.1 Datasets and Data Preprocessing

The MovieLens dataset is a public-domain dataset collected by the GroupLens lab in the Department of Computer Science and Engineering at the University of Minnesota. We conducted the experiments on MovieLens 100k (ML-100k) and MovieLens 1M (ML-1M)datasets, where ML-100k contains 100000 ratings (values from 1 to 5) from 943 users on 1682 movies and ML-1M contains 1,000,209 ratings from 6040 users on 3952 movies. In addition to the ratings, brief demographic information of the users is also available in the dataset. Researchers in the GroupLens lab have cleaned up the dataset and removed the users who rated less than 20 movies or have no complete demographic information.

We employed libFM [25] to predict the rating values of all the unrated movies based on the real rating values in the datasets.

4.2 The Metrics

Five metrics were adopted in our experiments. To begin with, we employed the prediction-in-top-N metric (in short, prediction) from [1] to evaluate the level of accuracy. This metric is defined as

$$\text{prediction-in-top-}N = \frac{\sum_{u \in U} \sum_{i \in L_N(u)} r^*_{u,i}}{\sum_{u \in U} |L_N(u)|},$$

where U is the set of users and $L_N(u)$ is the top-N recommended list for user u. $r^*_{u,i}$ is the predicted rating value that user u may rate on item i. A high prediction value indicates that the recommended items matched the user preferences.

We also evaluated coverage, which denotes the proportion of recommended items among all items. This metric was used to show the degree of diversity of the top-N item set generated by recommender systems. In this work, we defined coverage as

$$\text{Coverage} = \frac{|I_R|}{|I|},$$

where I is the set of items, and I_R is the set of recommended items. A higher coverage means that more items were recommended by the recommender system.

Furthermore, we employed the well-known Gini coefficient [8], which is used to measure the income distribution of a nation's residents, as our metric to measure the balance of recommendations among all items. Therefore, the balance of recommendations is essentially the recommendation frequency distribution of the items after the top-N recommendation are provided to each user.

In addition, in order to evaluate the degree of opportunity that the long-tail items can be recommended in a recommender system, we defined a metric named long-tail recommendation. In our work, long-tail items are items which were rated by less than L users and the long-tail recommendation was defined as

$$\text{long-tail} = \frac{\sum_{u \in U} |\,\text{longtail}(L_N(u))|}{\sum_{u \in U} L_N(u)},$$

where $\text{longtail}(L_N(u))$ is the set of long-tail items in $L_N(u)$.

Finally, since the quality of these recommended long-tail items should also be taken into consideration, we also computed the prediction on long-tail items to evaluate their quality.

4.3 Benchmarks and Our Approaches

We compared our approaches with three benchmarks. We first implemented four recommendation approaches as our approaches: **5D**, **5D_RR**, **5D_ACC**, and **5D_RR_ACC**, which are based on the 5D-score and the combination of rank by rankings (RR) algorithm and accuracy filtering (ACC) algorithm.

In addition, we implemented two approaches proposed in [1] and [2] as the benchmarks. The approach proposed in [1] is based on graph theory, so it is called **GBA** in our experiment. Similarly, the approaches adopted from [2] are called **RBT** approaches since they are based on ranking techniques. Moreover, the classical approach which simply selects top-N items according to the predicted rating values generated by MF was also implemented as a **baseline** benchmark.

In [2], several ranking methods were used to examine the ranking-based technique. We adopted the item popularity and the item average rating as the ranking methods and implemented two benchmarks, **RBT with itemPop** and **RBT with AvgRating**. We chose these two ranking methods because the item average rating method outperformed the others in respect of coverage and precision loss on MovieLens dataset in [2], and the experiment in [2] also showed that item popularity generated more long-tail recommendations on MovieLens dataset than the other methods.

[1] and [2] introduced *ranking-threshold* $T_R \in [T_H, T\text{max}]$ to control the trade-off between accuracy and diversity of the recommendations. The ranking-threshold plays the same role as the threshold T in our accuracy filtering (ACC) algorithm. The main difference between our work and [1] [2] is the way we sort the items before generating the recommendation list. RBT and GBA sort the items according to the predicted rating values while our approaches sort the items according to the 5D-scores. Therefore, the items within the interval of $[T_H, T_R)$ can never ranked higher than those within $[T_R, T\text{max}]$ by RBT. In our approaches, those items have better opportunities to achieve higher ranking because 5D-scores consider not only popularity of items but also relative preference of users. Similarily, although GBA maximizes the coverage by recommending maximum number of different items within $[T_R, T\text{max}]$, it has difficulty in improving the quantity of long-tail recommendation.

4.4 Parameters and Experimental Results

We compared the results of recommendation methods with our approach under different threshold settings. T_H and $T\text{max}$ were set to 3.5 and 5 in [1] and [2]. In our experiments, they were set to 2 and 5 respectively, and T_R was set to $[4.2, 3.8, ..., 2.2]$ for ML-100k and $[5, 4.8, ..., 3]$ for ML-1M. We decided the range of T_R by considering the average of median rating of each user (ML-100k: 2.148; ML-1M: 3.15), and the average of their 99th percentile(ML-100k) and 99.5th percentile(ML-1M) (ML-100k: 4.249; ML-1M: 4.727). This threshold was also applied to 5D_ACC and 5D_RR_ACC. The parameter k and q in equation 1 were set to $3N$ and 1, and allocation(i) was normalized to [0,1]. Fi-

nally, the value N of the top-N setting was set to 3, 5, and 10 respectively in this experiments.

In order to define the Long Tails, we analysed the distribution of ratings in the ML-100k and ML-1M datasets. There are about 70% of items were rated by less than 50(5%) users in ML-100k dataset and about 70% of items were rated by less than 250(4%) users in ML-1M dataset. Therefore, in our experiments, long-tail items are those items have less than 50/250 real ratings.

We first present the patterns of all approaches under all thresholds on three metrics: prediction, coverage and long-tail recommendation on ML-100k dataset in Figure 3.

We noticed an interesting phenomenon in Figure 3a: the prediction-in-top-N of our approaches with ACC had the best performance when the threshold was close to 2.8. The reason is that ACC took both 5D-scores and predicted rating values into consideration and thus retained the prediction-in-top-N, however, in the situations of higher threshold (system failed to find enough (N) items) or lower threshold (too many candidate items), the predicted rating values were ignored. As a result, prediction-in-top-N decreases.

In addition, the baseline approach showed its weakness in coverage (Figure 3b), since it only recommended best-rating items. RBT approaches performed worse than the baseline approach when $T_R \leq 2.4$, which is surprising. The reason may be that these approaches recommended only high-ranked items (according item popularity or item average rating) when the threshold value was small.

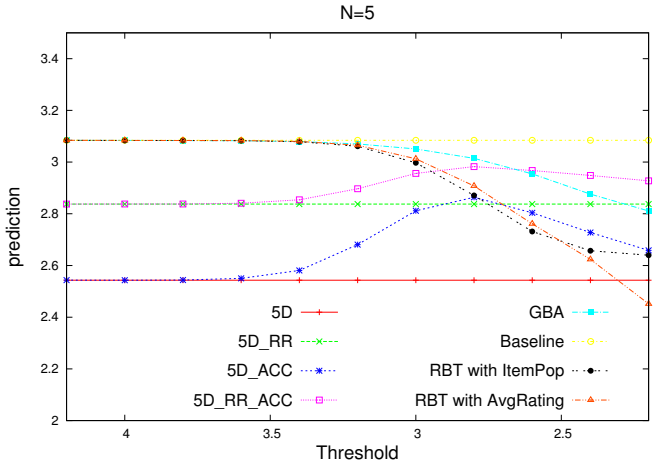
Finally, Figure 3c shows the results of long-tail recommendation. Comparing with the RBT with ItemPop approach, our approaches recommended more long-tail items in large threshold values and RBT with ItemPop outperformed ours in small threshold values.

In the following section, we show the average performances (under all thresholds) of 8 approaches on 5 Metrics over two datasets.

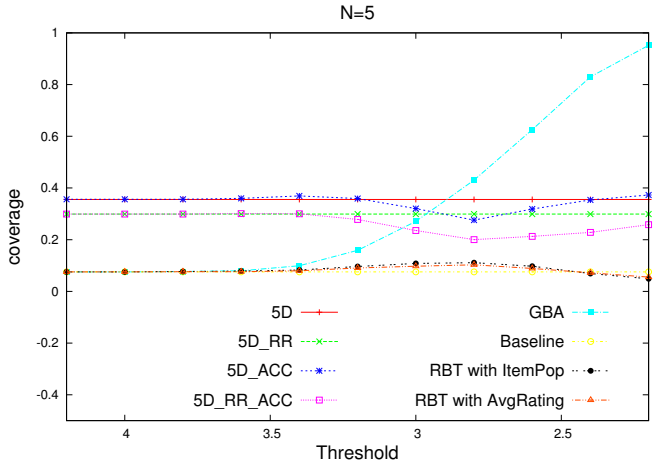
Figure 4 shows the result of the prediction. The baseline benchmark (the 6th bar), which only recommended items with high predicted rating values, maximized the prediction. 5D (the 1st bar)sacrificed considerable amount of prediction, but the prediction values significantly improved in 5D_RR (the 2nd bar)and 5D_RR_ACC (the 4th bar) with the help of RR and ACC algorithms. Comparing to ACC, RR would be a better choice to improve prediction-in-top-N.

According to Figure 5, GBA (the 5th bar) had the best performance on coverage and our approaches performed stably in this metric as well. Although we can see that 5D_RR and 5D_RR_ACC are outperformed by 5D and 5D_ACC (the 3rd bar)in most cases, all of our approaches surpassed the RBT approaches(the 7th and 8th bars) and the performances increased significantly with the increasing of N.

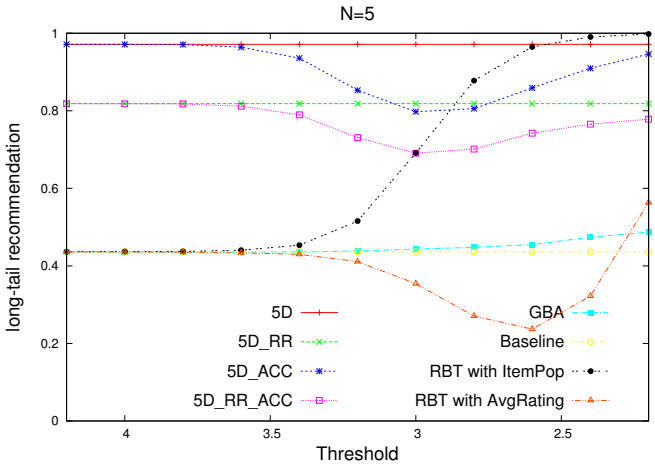
Figure 6 displays the Gini coefficient of the recommendations generated by these approaches. (Note that a smaller Gini coefficient means a more balanced recommendation set, which is preferred.) The results on Gini coefficient were similar to coverage. Firstly, RBT approaches and the baseline approaches performed badly in Gini coefficient. On the other hand, GBA showed its effectiveness in this metric. Moreover, 5D and 5D_ACC performed better than 5D_RR and 5D_RR_ACC again. Note that when N increased, GBA's advantage became less-significant. Our approaches outperformed GBA when $N = 10$.



(a) Prediction@5

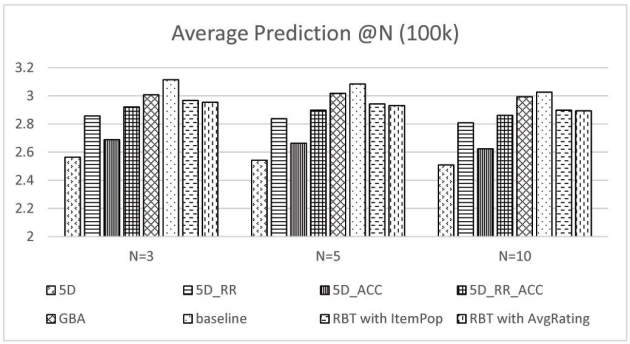


(b) Coverage@5

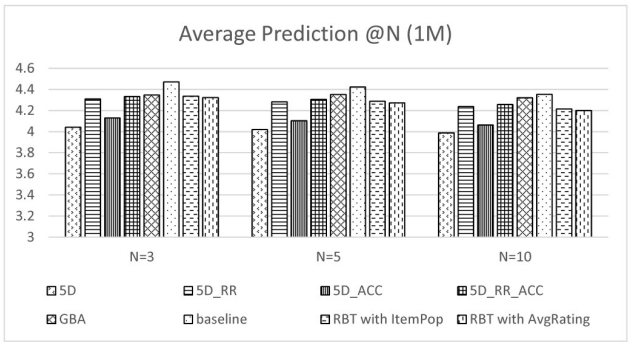


(c) Long-tail recommendation@5

Figure 3: The results of prediction, coverage and long-tail recommendation in Top 5 on MovieLens 100k.

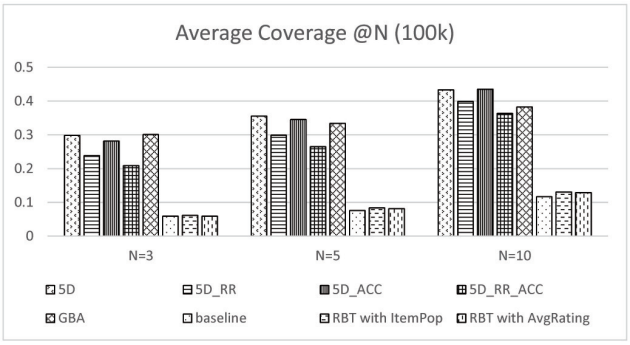


(a)

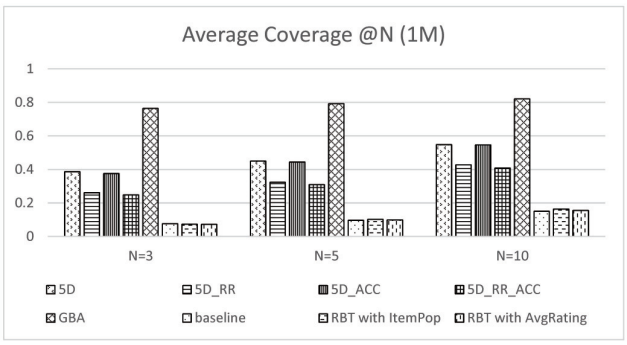


(b)

Figure 4: The results of average prediction-in-Top-N over MovieLens 100k(4a) and MovieLens 1M(4b)

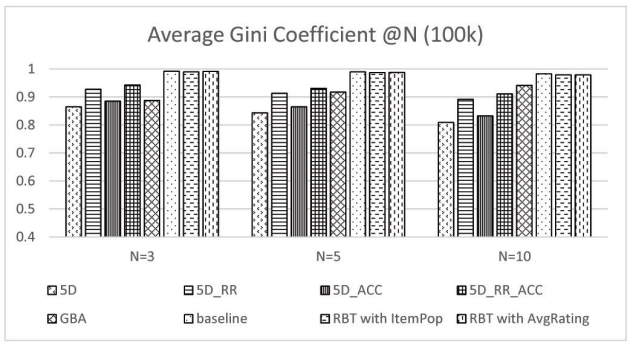


(a)

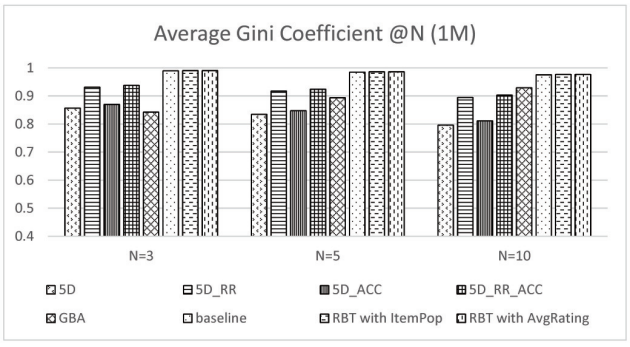


(b)

Figure 5: The results of average coverage-in-Top-N over MovieLens 100k(5a) and MovieLens 1M(5b)

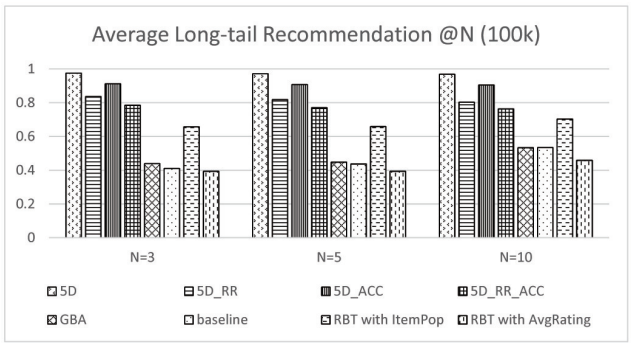


(a)

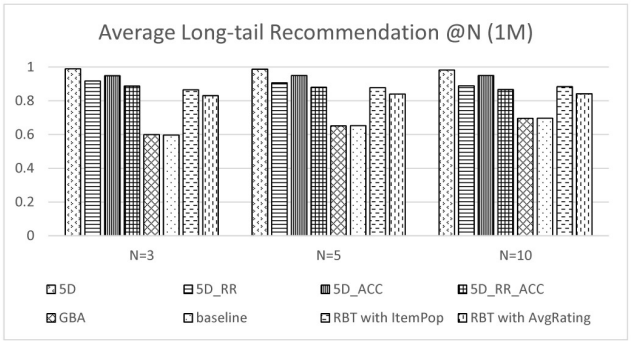


(b)

Figure 6: The results of average Gini coefficient-in-Top-N over MovieLens 100k(6a) and MovieLens 1M(6b)

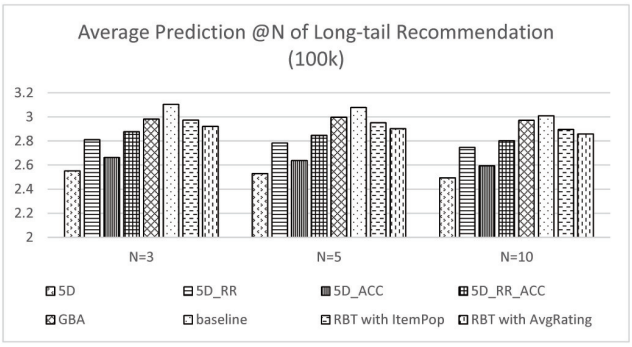


(a)

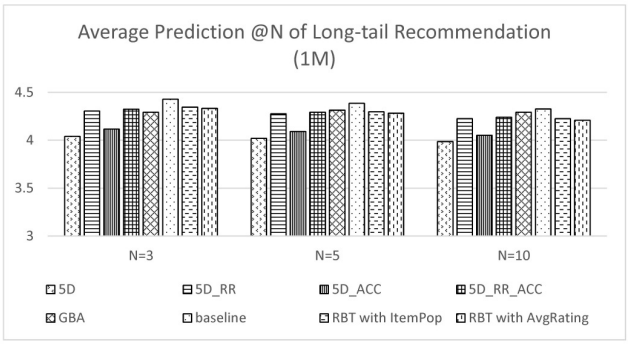


(b)

Figure 7: The results of average long-tail recommendtion-in-Top-N over MovieLens 100k(7a) and MovieLens 1M(7b)



(a)



(b)

Figure 8: The results of average prediction-in-Top-N of long-tail recommendation over MovieLens 100k(8a) and MovieLens 1M(8b)

Finally, the results of long-tail recommendation are shown in Figure 7 and the prediction on long-tail items are shown in Figure 8. In Figure 7, we can find that RBT with itemPop outperformed RBT with AvgRating, which is consistent with the result given in [2]. GBA had difficulty in recommending long-tail items since this approach does not focus on recommending items in Long Tails. Our approaches have significant advantage over the baseline approach and GBA and showed similar performance with RBT approaches on these two metrics.

To summarize, our approaches can improve long-tail recommendation and coverage/balance simultaneously. While other benchmarks performed well in either coverage/balance (GBA) or long-tail recommendation (RBT) only; our approach improve current recommendation results.

5. CONCLUSIONS AND FUTURE WORK

In this work, we designed a recommender system to promote long-tail items and enhance aggregate diversity. We proposed the concept of 5D-scores which takes accuracy, coverage, balance, the quantity of the recommended items, and the quality of the recommended items into consideration.

Our approach includes two sequential phases: the resource allocation phase and the recommendation phase. In the resource allocation phase, we re-distributed "the chance of being recommended" to all items. This distribution reserves some opportunities for the long-tail items to be recommended and keeps some privilege for the popular items with good reputations, thus guaranteeing a balance between the diversity and the accuracy of recommendations. In the recommendation phase, we further improve the aggregate diversity and promote long-tail items by considering the *relative preferences* of the users and the resource allocation from the resource allocation phase. Note that we consider not only the quantity, but also the quality of the long-tail items in our recommendations. The whole approach follows the idea of "exploring individual users' unique taste" and aims at discovering worth-recommending long-tail items which can attract the users and emphasize their individual taste.

As shown in the experiments, our approach can generate highly divergent and balanced recommendations with little amount of prediction loss. Moreover, our approach excels in promoting long-tail items on quantity as well as quality. Although we believe that our approach is data-independent, it would be interesting to conduct further experiments on public datasets to investigate the generality of our approach. Moreover, the application of our approach on an on-line recommendation system and user study would likely improve our understanding of the usability of our approach in more practical settings.

Last but not least, it would also be interesting to combine our approach with people-to-people recommender system. It has been shown that expression of interests (EOI) sent to unpopular users on online dating websites gets higher success rate [23] and promoting less popular users also improves the success rates [16]. We believe that our approach may able to help to discover more worth-recommending candidates for the users of online dating websites.

6. REFERENCES

[1] G. Adomavicius and Y. Kwon. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proceedings of workshop on novelty and diversity in recommender systems*, pages 3–10, 2011.

[2] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):896–911, 2012.

[3] C. Anderson. *The long tail: Why the future of business is selling less of more*. Hyperion Books, 2008.

[4] K. Bradley and B. Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth National Conference in Artificial Intelligence and Cognitive Science (AICS-01)*, pages 75–84, 2001.

[5] E. Brynjolfsson, Y. J. Hu, and D. Simester. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57(8):1373–1386, Aug. 2011.

[6] E. Brynjolfsson, Y. J. Hu, and M. D. Smith. Research commentary - long tails vs. superstars: The effect of information technology on product variety and sales concentration patterns. *Information Systems Research*, 21(4):736–747, 2010.

[7] D. M. Fleder and K. Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management Science*, 55(5):697–712, 2009.

[8] C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121):124–126, 1921.

[9] D. G. Goldstein and D. C. Goldstein. Profiting from the long tail. *Harvard Business Review*, 84(6):24–28, June 2006.

[10] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, 1999. ACM.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[12] H.-s. Huang, K.-l. Lin, J. Y.-j. Hsu, and C.-n. Hsu. Item-triggered recommendation for identifying potential customers of cold sellers in supermarkets. In *Workshop on the Next Stage of Recommender Systems Research, in conjunction with the 2005 International Conference on Intelligent User Interfaces*, 2005.

[13] N. Jones and P. Pu. User Technology Adoption Issues in Recommender Systems. In *Proceedings of the 2007 Networking and Electronic Commerce Research Conference*, pages 379–394, Riva del Garda, 2007.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, 2008. ACM.

[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[16] A. Krzywicki, W. Wobcke, X. Cai, M. Bain, A. Mahidadia, P. Compton, and Y. S. Kim. Using a critic to promote less popular candidates in a people-to-people recommender system. In *Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence*. AAAI, 2012.

[17] A. Lacerda and N. Ziviani. Building user profiles to improve user experience in recommender systems. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 759–764, New York, NY, USA, 2013. ACM.

[18] K.-L. Lin. *Item-triggered Recommendation*. PhD thesis, National Taiwan University, 2005.

[19] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA'06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[20] G. Oestreicher-Singer and A. Sundararajan. Recommendation networks and the long tail of electronic commerce. *MIS Quarterly*, 36(1):65–84, March 2012.

[21] Y.-J. Park. The adaptive clustering method for the long tail problem of recommender systems. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1904–1915, 2013.

[22] Y.-J. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 11–18, New York, NY, USA, 2008. ACM.

[23] L. Pizzato, T. Rej, J. Akehurst, I. Koprinska, K. Yacef, and J. Kay. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction*, 22:1–42, 2012.

[24] P. Pu, L. Chen, and R. Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4-5):317–355, 2012.

[25] S. Rendle. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, 3(3):1–22, May 2012.

[26] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, Mar. 1997.

[27] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[28] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen. Challenging the long tail recommendation. *Proc. VLDB Endow.*, 5(9):896–907, May 2012.

[29] M. Zhang. Enhancing diversity in top-n recommendation. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 397–400, New York, NY, USA, 2009. ACM.

[30] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 123–130, New York, NY, USA, 2008. ACM.