Flatter Is Better: Percentile Transformations for Recommender Systems

MASOUD MANSOURY, DePaul University, USA
ROBIN BURKE, University of Colorado, Boulder, USA
BAMSHAD MOBASHER, DePaul University, USA

It is well known that explicit user ratings in recommender systems are biased toward high ratings and that users differ significantly in their usage of the rating scale. Implementers usually compensate for these issues through rating normalization or the inclusion of a user bias term in factorization models. However, these methods adjust only for the central tendency of users' distributions. In this work, we demonstrate that a lack of *flatness* in rating distributions is negatively correlated with recommendation performance. We propose a rating transformation model that compensates for skew in the rating distribution as well as its central tendency by converting ratings into percentile values as a pre-processing step before recommendation generation. This transformation flattens the rating distribution, better compensates for differences in rating distributions, and improves recommendation performance. We also show that a smoothed version of this transformation can yield more intuitive results for users with very narrow rating distributions. A comprehensive set of experiments, with state-of-the-art recommendation algorithms in four real-world datasets, show improved ranking performance for these percentile transformations.

CCS Concepts: • **Information Systems** → **Recommender Systems**; *Retrieval effectiveness*;

Additional Key Words and Phrases: Recommender systems, rating distribution, percentile transformation, flatness

ACM Reference format:
Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2021. Flatter Is Better: Percentile Transformations for Recommender Systems. *ACM Trans. Intell. Syst. Technol.* 12, 2, Article 19 (March 2021), 16 pages.
https://doi.org/10.1145/3437910

1 INTRODUCTION

Recommender systems have become essential tools in e-commerce systems, helping users to find desired items in many contexts. These systems use information from user profiles to generate personalized recommendations. User profiles are either implicitly inferred by the system through user interaction or explicitly provided by users [Adomavicius et al. 2005; Adomavicius and Tuzhilin 2015]. In the latter case, users are asked to rate different items based on their preferences and may have individual differences in how they use explicit rating scales: Some users may tend to rate

Authors' addresses: M. Mansoury and B. Mobasher, School of Computing, DePaul University, Chicago, IL, USA; emails: mmansou4@depaul.edu, mobasher@cs.depaul.edu; R. Burke, Department of Information Science, University of Colorado Boulder, Boulder, CO, USA; email: robin.burke@colorado.edu.

2157-6904/2021/03-ART19 $15.00
https://doi.org/10.1145/3437910

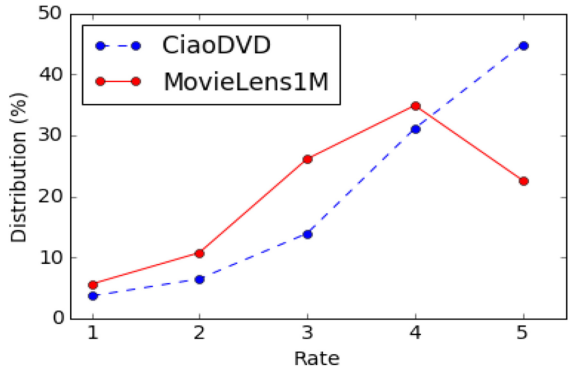


Fig. 1. Rating distribution of CiaoDVD and MovieLens datasets.

Table 1. Rating Profiles with Percentile Transformation

Alice	rating	⟨1, 1, 2, 2, 3, 3, 3, 4, 5⟩
Bob	rating	⟨3, 3, 4, 4, 4, 5, 5, 5, 5⟩
Alice	percentile	⟨20, 20, 40, 40, 70, 70, 70, 80, 90⟩
Bob	percentile	⟨20, 20, 50, 50, 50, 90, 90, 90, 90⟩

higher, while some users may tend to rate lower; some users may use the full extent of the rating scale, while others might use just a small subset [Herlocker et al. 1999].

When a user concentrates his or her ratings in only a small subset of the rating scale, this often results in ratings distributions that are skewed—most often toward the high end of the scale. This is because items are not rated at random, but rather preferred items are more likely to be selected and therefore rated due to selection bias [Marlin et al. 2007]. Figure 1 shows the overall rating distribution of two datasets that exhibit typically right-skewed distributions. Users in the CiaoDVD dataset, for example, have assigned less than 10% of the ratings to ratings 1 and 2 and some 70% of ratings are values 4 and 5. We can assume this is not because there are so many more good movies than bad, but rather that users are selecting movies to view that they are likely to enjoy and the ratings are concentrated among those selections. A drawback of this skew to the distribution is that we have more information about preferred items and less information about items that are not liked as well. It also means that a given rating value may be ambiguous in meaning.

As an example, assume that Alice and Bob both purchase an item X and rate it. Alice is a user who tends to rate lower and tends to use the whole rating scale, while Bob is a user who tends to rate higher and never uses ratings at the bottom of the scale. Their profiles, sorted by rating value, are shown in Table 1. After using item X, Alice is fully satisfied with it, but Bob is only partially satisfied. As a result, both rate the item X as 4 of 5 although they have different levels of satisfaction toward that item. These ratings, while identical, do not carry the same meaning. A transformation based on percentiles, shown in the bottom rows of the table, captures this distinction well: A rating of 4 for Alice is percentile 80, whereas for Bob, the same score has a score of 50. In addition, unlike the original profiles, where the users' ratings are distributed over different ranges, these profiles span the same numerical range from 20 to 90.

Rating normalization in neighborhood models [Resnick et al. 1994] and inclusion of a bias term in factorization models [Koren 2008; Koren et al. 2009] are two common techniques for managing rating variances among users. However, these techniques adjust only for the central tendency of

Table 2. An Example of User-item Matrix

												Similarity to U1				
	I1	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**	**I11**	rating	z-score	Gaussian	Decoupling	percentile
U1	1	1	1	—	1	1	—	2	2	3	3	—	—	—	—	—
U2	1	2	3	—	—	—	3	4	—	5	5	0.914	0.914	0.914	0.940	0.916
U3	—	—	—	—	1	3	—	2	5	—	4	0.567	0.567	0.567	0.580	0.567
U4	1	4	4	—	4	4	—	5	5	5	5	0.606	0.612	0.606	0.909	0.966
U5	3	3	—	3	2	2	—	2	4	5	—	0.734	0.758	0.734	0.564	0.571
U6	5	5	5	—	5	5	—	2	2	4	4	−0.531	−0.549	−0.531	−0.426	−0.933

users' rating distributions and do not fully compensate for different patterns of rating behavior that users exhibit. However, the percentile transformation proposed in this article takes into account the whole shape of the distribution, not only its central tendency, and therefore retains more information from the original user profile.

Table 2 shows a hypothetical rating matrix. In this table, users with different rating patterns are exhibited. Some users tend to rate lower (e.g., *U1*), some users tend to rate higher (e.g., *U4* and *U6*), some users show normal rating pattern (e.g., *U2* and *U5*), and, finally, some users do not show any pattern (e.g., *U3*). For illustration purposes, we show how different normalization methods affect the computation of user-user similarities (in this case similarities to user $U1$). Note that for calculating the similarity values based on z-score, Gaussian, Decoupling, and percentile, first we created z-score, Gaussian, Decoupling, and percentile matrix from Table 2, and then we used the corresponding matrix for calculating similarity values for each technique.

Based on users' characteristics and rating patterns, certain similarity values between users are expected. For example, *U1* and *U4* show different behavior when providing ratings to items. Based on their rating patterns, a rating of 3 provided by *U1* can be mapped to a rating of 5 provided by *U4*, or rating 1 provided by *U1* can be mapped to rating 3 provided by *U4*. Hence, a good transformation technique should be able to capture these differences in these users' behaviors. For this case, our percentile[1] technique assigns high similarity value to *U1* and *U4*. The same result can be observed between *U1* and *U6*. However, the other transformation techniques are unable to capture these differences when calculating similarity values. In other cases, where users have normal rating patterns or do not show a specific rating pattern, our percentile technique behaves similarly to other normalization techniques.

The above example also shows that, in general, original ratings, z-score, and Gaussian normalization techniques behave similarly even when comparing rating patterns that are very different. Decoupling normalization technique also does not work consistently well in these cases. However, our percentile technique yields more intuitive results for disparate profiles, while behaving similarly to the other transformations in other cases.

One can imagine the most informative rating distribution would be a flat, uniform, distribution. Users would provide ratings for items sampled uniformly across all of the items, and the profiles would then represent their preferences across the whole inventory and across all possible rating values. One way to think about the difference between the typical, skewed, distribution and a uniform one is in terms of information entropy. The worst case, a profile where every item is rated the same, carries no information that distinguishes the different items, and the assignment of a

[1]Results are based on first index percentile transformation. The same results are observed for median and last index percentile transformation.

rating to an item has low entropy. A profile where the rating values are distributed across the items with equal frequency has maximum entropy.

In this article, we formalize a rating transformation model as above that converts users' ratings into percentile values as a preprocessing step before recommendation generation. Each value associated with an item therefore reflects its rank among all of the items that the user has rated. Thus, the percentile captures an item's position within a user's profile better than the raw rating value and compensates for differences in users' overall rating behavior. Also, the percentile, by definition, will span the whole range of rating values, and as we show, gives rise to a more uniform rating distribution. To handle cases in which users use only a small part of the rating range, we also introduce a smoothed variant of the percentile transformation that preserves distinctions among users with different rating baselines.

We show that these two properties of the percentile transformation—its ability to compensate for individual user biases and its ability to create a more uniform rating distribution—lead to enhanced recommender system performance across different algorithms, different datasets, and different performance measures. We also show that the percentile transformation creates flatter rating distributions and that this is correlated with improved recommendation performance.

This article makes the following contributions:

1. We propose a rating transformation model that converts users' ratings into percentile values to compensate for skew in rating distributions and variances in users' rating behaviors.
2. We empirically evaluate the proposed percentile technique using state-of-the-art recommendation algorithms on four real-world datasets. Our experiments include both overall recommendation performance and recommendation performance on long-tail items.
3. We show the relationship between the uniformity of the rating distribution and the quality of recommendation; with flatter distributions being correlated with better recommendations.
4. We show that the smoothed version of the transformation overcomes the issue of identical ratings in rating transformations, and provides further improvement over the percentile alone.

2 BACKGROUND

It has long been noted that users differ in their application of explicit rating scales. Resnick's algorithm, perhaps the most well-known prediction method in recommendation, normalizes ratings by user mean when computing its predictions [Resnick et al. 1994]. Herlocker et al. [1999] used z-scores instead of absolute rating values in recommender systems and investigated its effectiveness on quality of recommendations. In this research, they compared the performance of three rating normalization techniques and showed that bias-from-mean approach performs significantly better than a non-normalized rating approach and slightly better than the z-score approach in terms of mean absolute error. This result is consistent with our findings.

Kamishima and Akaho [2010] proposed a ranking-based method that replaces the existing rating scheme with a ranking scheme. In this method, instead of rating the items, users order the items based on their preferences. Based on order statistics theory, preference orders expressed by users are converted into scores and then recommendation algorithms are applied on these scores to generate recommendations. This method proved effective, but it is not widely applicable, because order-based input is rare in recommender system interfaces, and requires more effort from users than rating assignment.

Jin and Si [2004] compared the impact of two normalization techniques for user ratings, namely Gaussian and Decoupling normalization techniques on the performance of recommender systems.

This research found that Decoupling normalization is more effective than Gaussian normalization. A more recent study by Kim et al. [2016] proposed a normalization model that learns the differences in users' rating dispositions using two phases of clustering and normalization. At the clustering phase, users are clustered based on their rating disposition and then at the normalization phase, users' ratings are normalized through predicting their rating disposition and adjusting their neighbors' ratings based on that disposition.

Adamopoulos and Tuzhilin [2013] incorporated weighted percentile into the neighborhood model to assign weight to the ratings provided by neighbors of the target user on the target item. In their approach, depending on the rank of the ratings provided by neighbors on a target item, when calculating the predicted ratings in traditional neighborhood model, the ratings will be weighted by a predefined percentile weight. Also, Dixit and Jain [2019] applied this weighted percentile technique on context-aware recommender systems to improve the quality of recommendations. Neither of these techniques involve transformation of rating inputs.

In the domain of trust relations in social networks, it has been shown that percentile values are more effective than absolute trust ratings. Hasan et al. [2009] showed that using percentile values instead of absolute trust ratings improves the accuracy of trust propagation model. They applied a method introduced by NIST[2] for converting predicted percentile values into trust rating in social networks.

3 PERCENTILE TRANSFORMATION

In statistics, given a series of measurements, percentile (or quantile) methods are used to estimate the value corresponding to a certain percentile. Given the Pth percentile, these methods attempt to put P% of the dataset below and (100-P)% of the dataset above. There are a number of different definitions in the literature for computing percentiles [Hyndman and Fan 1996; Langford 2006]. Although they are apparently different, the answers produced by these methods are very similar, and the slight differences are negligible [Langford 2006]. In this article, we use a definition from Hyndman and Fan [1996].

The percentile value, p, corresponding to a measurement, x, in a series of measurements, M, is computed with regard to the position of x in the ordered list M, $o(M)$, as follows:

$$p^z(x, M) = \frac{100 \times position^z(x, o(M))}{|M| + 1}, \tag{1}$$

where $position^z(x, o(M))$ returns the index of occurrence of x in $o(M)$, or the position in the order where x would appear if it is not present, and $|M|$ is the number of measurements in M. For more details, see Hyndman and Fan [1996].

This transformation assumes that values are distinct and there is no repetition in the series. However, with rating data, we often have a different situation. User profiles usually contain many repetitive ratings, and it is unclear how to specify the position of a rating. For example, in a series of ratings $v = \langle 2, 3, 3, 3, 3, 3, 5, 5, 5 \rangle$, it is not clear what the position of rating 3 should be. We could take the first occurrence, position 2, or the last occurrence 6, or something in between.

We explore the performance of our percentile technique by taking the index of the first, median, and last occurrence of repeated ratings in the ordered vector. Hence, the parameter z determines the index rule that we want to use for percentile transformation and can take values f, m, and l as *first*, *median*, and *last* index assignments, respectively. Each of these index assignments signifies a particular meaning when transforming rating profiles. The index of the last occurrence, for example, is the highest rank (most preferred) position occupied by an item with the given rating.

[2]National Institute of original and Technology, http://www.itl.nist.gov/div898/handbook/prc/section2/prc262.htm.

We experiment with all index assignments and show that the rule that yields a more uniform distribution will provide greater recommendation performance.[3]

Even in contexts where ratings are gathered implicitly, they are often converted into numeric scores representing user preference or relevance. For example, time spent on a page is often considered a measure of user interest [Yi et al. 2014] or number of seconds watched of a video [Zhao et al. 2019]. Profiles generated in these ways can also be normalized using the percentile transform as well, although they are less likely to have repeated entries.

For our purposes, the entire set of ratings provided by a user u is considered a rating vector for u, denoted by R_u with an individual rating for an item i, denoted as r_{ui}. Let $p(v, \ell)$ be the percentile mapping in Equation (1) from a rating value v in a list of values ℓ, using the first, median, and last index method. Then, the percentile value of a rating r provided by user u on an item i is computed by taking the rating r_{ui} and calculating its percentile rank within the whole profile of the user. For example, based on the last index rule, for the user Bob from Table 1, an item rated 3 would have percentile rank $100 * 2/(9 + 1) = 20$. We define the user-percentile function, Per_u^z, as follows:

$$\mathrm{Per}_u^z(u, i) = p^z(r_{ui}, R_u). \tag{2}$$

Analogously, we can consider profiles for an item, denoted by R_i, to be all of the ratings provided for that item by users, and we can define a similar transformation for item profiles in which the transformation takes into consideration the rank of the rating across all ratings for that item, an item-percentile function,

$$\mathrm{Per}_i^z(u, i) = p^z(r_{ui}, R_i). \tag{3}$$

Note that Per_u^z and Per_i^z might be quite different for the same user-item pair. For example, user x might be a strong outlier relative to the dataset, liking an item y that no one else does. $Per_i^z(x, y)$ would therefore be quite high. However, if user x has a strong tendency to high ratings in general, then $Per_u^z(x, y)$ might be significantly lower. This article concentrates on the user-oriented transformation: We plan to explore the properties of the Per_i^z transformation in future work.

3.1 Measuring Distribution Uniformity

One of our claims in this article is that the flatness of the rating distribution in a dataset is an indicator of how well collaborative recommendation will perform, and that the percentile transformation achieves flatter distributions. To test this hypothesis, we need a measure of how close a rating distribution is to uniformity.

One common technique for measuring the shape of a distribution is *kurtosis*. Kurtosis is regularly used for determining the normality of a distribution. A normal distribution has a kurtosis value of 3[4], and a value below 3 indicates a distribution closer to uniform. Although kurtosis can be used for measuring the uniformity of a distribution, it is not a robust measure and may be misleading. Therefore, to overcome this issue, we introduce a new technique for measuring the flatness of a distribution as an alternative along with kurtosis.

To determine the flatness of a ratings distribution we calculate Kullback-Leibler divergence (KLD) between the observed rating distribution and a uniform distribution in which each rating value occurs the same number of times. If there is a discrete set of rating values V (for example,

[3]See https://github.com/masoudmansoury/percentile for the code for computing these and other transformations described in this article.

[4]In some references, kurtosis is defined such that 0 reflects a normal distribution.

Table 3. Flatness Calculation of BookCrossing Dataset

Rating	Frequency	$D(v)$	$\log(\|V\|D(v))$
1	349	0.0029	−3.5275
2	606	0.0051	−2.9757
3	1,300	0.0109	−2.2125
4	1,944	0.0164	−1.8101
5	11,322	0.0953	−0.0481
6	8,934	0.0752	−0.285
7	19,776	0.1665	0.5096
8	29,233	0.2461	0.9005
9	21,221	0.1786	0.5802
10	24,113	0.203	0.7079
			$\mathcal{F}$=0.448

1,2,3,4,5), then we define the flatness measure $\mathcal{F}$ as

$$\mathcal{F}(D \parallel Q) = \sum_{v \in V} D(v) \log \frac{D(v)}{Q(v)}, \tag{4}$$

where V is the set of discrete rating values in rating matrix R and D is the observed probability distribution over those values. Q is a uniform distribution that associates a probability $1/|V|$ for each possible value in V (i.e., for each $v \in V$, $Q(v) = 1/|V|$). Therefore,

$$\mathcal{F}(D) = \sum_{v \in V} D(v) \log(|V|D(v)). \tag{5}$$

The $\mathcal{F}$ function measures the distance between the two distributions and hence how close the observed distribution is to the flat ideal, with a lower KLD value being indicative of a flatter distribution.

Table 3 illustrates the flatness calculation of BookCrossing dataset for original ratings. In this dataset, there are 10 rating values, $|V| = 10$. $D(v)$ is the probability distribution over each rating values calculated as

$$D(v) = \frac{frequency(v)}{\sum_{v \in V} frequency(v)}. \tag{6}$$

By using Equation (5), the flatness of this dataset will be $\mathcal{F} = 0.448$. Comparison between this flatness and the flatness of a uniform distribution (i.e., $\mathcal{F} = 0$) shows that the distribution of original ratings in BookCrossing dataset is far from a flat ideal.

The percentile, z-score, Gaussian, and Decoupling transformations yield real valued ratings, unlike the original discrete ratings chosen by users in these datasets. Evaluating our flatness measure at every point in these distributions yields results that are not comparable to the original discrete distribution.

To have comparable calculations of the $\mathcal{F}$ value across types of distributions, we created binned versions of the percentile, z-score, Gaussian, and Decoupling distributions, using the same number of bins as present in the original ratings. In a 10-star rating system, such as found in the BookCrossing data, the rating distribution covers 10 values, and hence we created 10 equal length bins for percentile, z-score, Gaussian, and Decoupling values and aggregated each bin by its mean.[5]

[5]As an example, we know that percentile values are between 0 and 100. Thus, we create 10 bins each of which with the length of 10 and aggregate each bin by mean of its distribution.

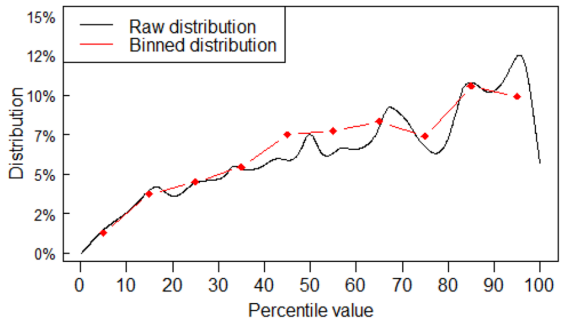


Fig. 2. Raw and binned percentile distributions for BookCrossing dataset.

Table 4. Statistical Properties of Datasets

Dataset	#users	#items	#ratings	Density	Ratings
BX	7,033	9,441	118,798	0.179%	1–10
CiaoDVD	17,595	16,113	72,042	0.025%	1–5
FilmTrust	1,508	2,071	35,497	1.137%	0.5–4.0
ML1M	6,040	3,706	1,000,209	4.468%	1–5

Figure 2 shows the percentile distribution (last index illustrated here) and its aggregated distribution for the BookCrossing dataset. The black curve is the percentile distribution and red line is its aggregated distribution with 10 bins. It shows that aggregating by mean retains the shape of the percentile distribution, while being comparable to the original ratings for computing flatness.

4 EXPERIMENTS

We evaluated the performance of percentile transformation on four publicly available datasets: BookCrossing, CiaoDVD, FilmTrust, and MovieLens. The characteristics of the datasets are summarized in Table 4. These datasets are from various domains and have different degrees of sparsity.

The ML1M is movie ratings data and was collected by the MovieLens[6] research group. The CiaoDVD[7] includes ratings of users for movies available on DVD. The FilmTrust is a small dataset collected from the FilmTrust website [Guo et al. 2013]. It contains both movie ratings and explicit trust ratings. Finally, the BX dataset is a subset extracted from the BookCrossing dataset[8] such that each user has rated at least five books and each book is rated by at least five users. The ML1M has the highest density and CiaoDVD has the lowest density.

4.1 Flatness

To evaluate the percentile transformation for its distributional properties, we evaluated its flatness and kurtosis compared to the original ratings distribution and a distribution based on the z-score, Gaussian, and Decoupling transformations over four datasets: BX, CiaoDVD, FilmTrust, ML1M.

First, we converted the original ratings into percentile, z-score, Gaussian, and Decoupling values. Then, we applied the binned flatness and kurtosis measures described above to these datasets to evaluate the transformations for their distributional properties. Table 5 shows the flatness ($\mathcal{F}$)

[6] https://grouplens.org/datasets/movielens/.
[7] https://www.cse.msu.edu/~tangjili/trust.html.
[8] http://www.informatik.uni-freiburg.de/~cziegler/BX/.

Table 5. Flatness ($\mathcal{F}$) and Kurtosis ($\mathcal{K}$) of Rating Distribution

Dataset	Method	Rating	z-score	Gaussian	Decoupling	Per_u^f	Per_u^m	Per_u^l
BX	$\mathcal{F}$	0.449	0.661	0.144	0.173	0.110	0.120	0.101
	$\mathcal{K}$	3.371	3.679	4.354	2.660	2.099	1.907	1.909
CiaoDVD	$\mathcal{F}$	0.317	0.468	0.339	0.267	0.208	0.218	0.153
	$\mathcal{K}$	3.619	3.781	4.919	2.470	2.384	2.343	2.044
FilmTrust	$\mathcal{F}$	0.355	0.248	0.2	0.122	0.053	0.034	0.086
	$\mathcal{K}$	3.206	3.317	4.839	2.611	1.938	1.831	1.850
ML1M	$\mathcal{F}$	0.153	0.562	0.881	0.213	0.057	0.055	0.130
	$\mathcal{K}$	2.648	2.920	7.380	2.687	2.078	1.829	1.961

Flatness of a uniform distribution is $\mathcal{F} = 0$.
Kurtosis of a uniform distribution is $\mathcal{K} < 3$.

and kurtosis ($\mathcal{K}$) values for each type of transformation on the four datasets (user profile transformation). As shown, the values for both measures are consistent across all three percentile transformations and datasets. As anticipated, the proposed percentile model makes the rating values flatter than rating transformations.

Thus, the proposed percentile transformation approach reduces skew in the rating distribution over the original ratings, z-score, Gaussian, and Decoupling values. Given these results, we expect to see better recommendation performance when we use percentile values as input for recommender systems since they have lower $\mathcal{F}$ and $\mathcal{K}$ values.

4.2 Methodology

We performed a comprehensive evaluation of the effect of the percentile transformation on the ranking performance of a number of recommendation algorithms. Due to the nature of our percentile technique, we experimented only with algorithms that make use of rating magnitude. The percentile transformation rescales rating values without changing their relative ordering, so it will have no effect when applied to ranking-based algorithms (for example, ListRank [Shi et al. 2010]). Implicit feedback algorithms that use unary data, such as Bayesian Personalized Ranking [Rendle et al. 2009], would also be inappropriate to use with percentile transform, because they use binary interaction information and ignore rating values.

Our experiments included user-based collaborative filtering [Resnick et al. 1994], item-based collaborative filtering [Sarwar et al. 2001], biased matrix factorization (`BiasedMF`) [Koren et al. 2009], singular value decomposition (`SVD++`) [Koren 2008], and non-negative matrix factorization [Lee and Seung 2001] However, in this article, for the purpose of presentation, we only report results on `BiasedMF` and `SVD++`.[9] None of the other conditions showed the percentile transform performing worse than the others. In some cases, it was not significantly different and in these cases, all transformation methods yielded the same results.

We performed fivefold cross validation, and in the test condition, generated recommendation lists of size 10 for each user. Then, we evaluated nDCG and precision[10] at list size 10. Results were averaged over all users and then over all folds. A paired t-test was used to evaluate the significance of results and based on paired t-test, the results shown in bold are statistically significant with a p-value of less than 0.05.

[9]Results on all algorithms and datasets are available at https://github.com/masoudmansoury/percentile.
[10]In addition, we evaluated recall and F-measure, also finding significant improvement in these metrics.

Table 6. Performance of Recommendation Algorithms at nDCG@10 and precision@10

Algorithm	Input	BX		CiaoDVD		FilmTrust		ML1M	
		Precision	nDCG	Precision	nDCG	Precision	nDCG	Precision	nDCG
BiasedMF	rate	0.005	0.009	0.005	0.019	0.039	0.078	0.107	0.116
	z-score	0.004	0.008	0.002	0.005	0.047	0.092	0.106	0.116
	Gaussian	0.001	0.002	0.001	0.001	0.014	0.031	0.007	0.007
	Decoupling	0.005	0.009	0.006	0.019	0.092	0.138	0.068	0.076
	Per_u^f	**0.007**	**0.014**	0.007	**0.027**	**0.158**	**0.317**	**0.132**	**0.151**
	Per_u^m	**0.007**	**0.014**	0.006	**0.024**	**0.160**	**0.314**	**0.137**	**0.156**
	Per_u^l	0.006	**0.012**	0.005	0.016	**0.165**	**0.302**	**0.136**	**0.156**
SVD++	rate	0.005	0.009	0.003	0.009	0.024	0.049	0.125	0.145
	z-score	0.003	0.006	0.003	0.011	0.035	0.072	0.125	0.145
	Gaussian	0.001	0.002	0.001	0.004	0.016	0.036	0.003	0.003
	Decoupling	0.005	0.009	0.006	0.013	0.048	0.085	0.012	0.011
	Per_u^f	0.006	**0.012**	0.006	**0.022**	0.044	**0.079**	**0.129**	**0.153**
	Per_u^m	0.006	**0.012**	0.006	**0.019**	0.049	**0.087**	**0.133**	**0.157**
	Per_u^l	0.005	**0.010**	0.004	**0.017**	**0.070**	**0.124**	**0.131**	**0.153**

Before reporting on the results here, we performed extensive experiments with different parameter configurations for each algorithm and dataset combinations. To determine sensible values for parameters, we followed the settings reported in the literature. In factorization models, for instance, we approximately set the number of factors and iterations based on the density of the dataset and convergence of the loss function, and we tested these parameters for sensitivity. We performed a grid search over *bias*[11] $\in \{0.0001, 0.001, 0.005, 0.01\}$, *factor* $\in \{50, 100, 150\}$, *iteration* $\in \{30, 50, 100\}$, and *learning rate* $\in \{0.0001, 0.001, 0.005, 0.01\}$. Results of extensive experiments show that, in general, across on all settings, our percentile technique works significantly better than the original ratings, z-score, Gaussian, and Decoupling values in terms of ranking quality.

4.3 Results

We include results for 14 experimental conditions: two recommendation algorithms evaluated over seven different inputs (the original ratings, the results of the three percentile transformations, the results of the z-score transformation, and the results of the transformations based on the Gaussian and Decoupling normalization introduced in Jin and Si [2004]). Table 6 shows the results for all the datasets and both algorithms, reporting the best-performing configuration for each dataset, algorithm, and input value.[12]

Results in Table 6 show that in terms of ranking quality (i.e., nDCG), the percentile technique produces recommendations that are consistently better than the other transformation techniques[13] over all the recommendation algorithms and datasets except for Per_u^l as input for BiasedMF on CiaoDVD dataset. On the densest dataset (ML1M), the average improvement by our percentile technique on BiasedMF is 33% and on SVD++ is 7%. The improvement on the FilmTrust dataset is 268% and 66%, on the CiaoDVD dataset is 182% and 95%, and on BX dataset is 58% and 48%, respectively.

[11] User, item, implicit feedback, and overall bias terms.
[12] We used LibRec 2.0 and librec-auto for all experiments [Guo et al. 2015; Mansoury and Burke 2019; Mansoury et al. 2018].
[13] We can think of the original ratings as a null transformation.

Table 7. Correlation between $\mathcal{F}$ / $\mathcal{K}$ and nDCG@10 for Each Algorithm

Dataset	$\mathcal{F}$		$\mathcal{K}$	
	BiasedMF	SVD++	BiasedMF	SVD++
BX	−0.28	−0.29	**−0.94**	**−0.93**
CiaoDVD	**−0.70**	−0.69	**−0.81**	**−0.89**
FilmTrust	**−0.81**	**−0.89**	**−0.81**	**−0.89**
ML1M	**−0.82**	−0.59	**−0.92**	**−0.73**

In addition to nDCG, Table 6 shows the performance of the transformation techniques in terms of precision, another measure of recommendation accuracy (without considering the ranking of recommendations). Again, our percentile technique outperforms other transformation techniques in most cases with significant results on denser datasets, FilmTrust and ML1M.

4.4 Flatness Analysis

Note that, in most cases, the results in Table 6 are consistent with our flatness hypothesis: *flatter profiles yield better recommendation quality*. We hypothesize that a transformation that produces a flatter distribution will compensate for skew in the rating distribution and generate improved recommendation performance. As we have seen, the percentile transformation generally leads to better performance and to flatter distributions, and the less-flat transformations have lower performance.

Besides the percentile values (the flattest distribution) that provide the best recommendation performance, our flatness hypothesis is also true about other transformations. For instance, based on the flatness values reported in Table 5, the Decoupling values provide flatter distribution than the original ratings, z-score, and Gaussian values on FilmTrust. Thus, we would expect better recommendation performance when Decoupling values are used as input for recommender systems compared to the original ratings, z-score, and Gaussian values. According to the results in Table 6, the performance of recommendation algorithms when the Decoupling transformation is applied to the input is significantly better than the original ratings, z-score, and Gaussian transformations across all metrics. The same results can also be observed for the original ratings on ML1M with slight improvement.

We examined this phenomena more closely using seven types of inputs for rating transformation: original ratings, first, median, and last percentile values, z-score, Gaussian, and Decoupling transformations.[14] We examined the $\mathcal{F}$ and $\mathcal{K}$ values for the training data under the different transformations and computed correlation against the recommendation performance using nDCG@10.

Table 7 shows the correlation between $\mathcal{F}$ and $\mathcal{K}$ values of each transformation (i.e., original rating, first index percentile, median index percentile, last index percentile, z-score, Gaussian, and Decoupling) and nDCG@10 of recommendation algorithms with those input values. It clearly shows significant negative correlation between performance and divergence from uniformity. (Note that a low $\mathcal{F}$ and $\mathcal{K}$ values correspond to a flatter distribution.) The flatter distributions (closer to zero for $\mathcal{F}$ and below 3 for $\mathcal{K}$) yield better performance for all three algorithms across all datasets. Except for the $\mathcal{F}$ value of BX, the $\mathcal{F}$ value of CiaoDVD on SVD++, and the $\mathcal{F}$ value of ML1M on SVD++, all of the observed correlations between $\mathcal{F}$ / $\mathcal{K}$ and nDCG are between −0.99 and −0.70,

[14]Because a limitation in LibRec 2.0, z-score, Gaussian, and Decoupling values are shifted to positive values by the addition of an offset.

Table 8. Performance of Recommendation Algorithms on Long-tail Items at nDCG@10

Dataset	Algorithm	Rate	z-score	Gaussian	Decoupling	Per_u^f	Per_u^m	Per_u^l
BX	BiasedMF	0.0008	0.0009	0.0007	0.0009	**0.0018**	**0019**	**0.0034**
	SVD++	0.0011	0.0009	0.0009	0.0009	**0.0024**	**0.0024**	**0.0043**
CiaoDVD	BiasedMF	0.019	0.005	0.002	0.005	**0.027**	**0.024**	0.016
	SVD++	0.009	0.011	0.004	0.012	**0.022**	**0.019**	**0.017**
FilmTrust	BiasedMF	0.058	0.067	0.015	0.013	**0.199**	**0.211**	**0.227**
	SVD++	0.0279	0.072	0.018	0.022	**0.060**	0.070	**0.101**
ML1M	BiasedMF	0.036	0.036	0.003	0.034	0.036	**0.038**	**0.039**
	SVD++	0.046	0.046	0.004	0.039	0.043	**0.047**	**0.050**

indicative of a strong inverse relationship: In general, flatter distributions give better algorithmic performance.

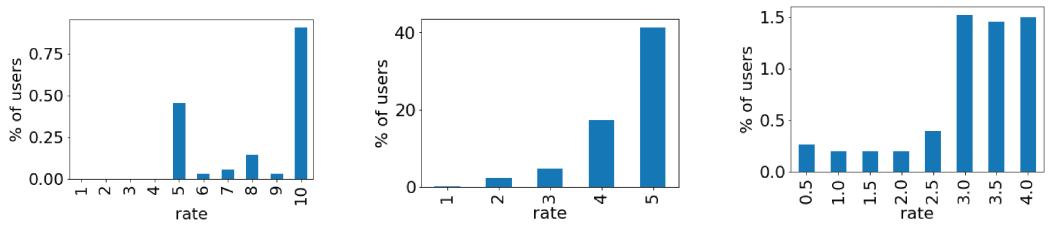
Further investigation showed that the low correlation of $\mathcal{F}$ on BX is due to the inaccurate flatness calculation on Gaussian values. Looking at the flatness values in Table 5 for Gaussian values show that the calculation based on $\mathcal{F}$ and $\mathcal{K}$ are inconsistent (high flatness for one measure and low flatness for another measure on the same dataset). This can be due to the fact that the distribution of Gaussian values is more complex and our flatness measurement techniques are unable to correctly calculate it. Also, we observed that omitting Gaussian values in correlation calculation in Table 7 yields much higher correlation values.

4.5 Long-tail Performance

One enduring challenge in collaborative recommendation is the ability to provide accurate recommendation about items in the "long tail" of popularity. These items are often of great interest to users [Brynjolfsson et al. 2006; Park and Tuzhilin 2001], but many algorithms do not recommend them with sufficient frequency [Jannach et al. 2015]. In this section, we examine the performance of recommendation algorithms on recommending long-tail items when using different input transformations. To do this, we follow the methodology in Cremonesi et al. [2010] for analyzing item popularity. In this methodology, for each user in test set, a list of items will be recommended, and then ranking quality will be measured only on long-tail items in the recommended list. The main goal of this methodology is to measure the effectiveness of a recommendation algorithm in recommending long-tail items.

For this evaluation, we need to determine the long-tail items from training data. To do this, we create cumulative popularity list of items sorted from most popular to less-popular items, then we define a cutting point such that it divides the items into short-head and long-tail items. For experiments in this article, we used cutting point of 20%, meaning that cumulatively 20% of most popular items are considered as short-head items and the rest of less popular items are considered as long-tail items.

Table 8 shows the performance of recommendation algorithms on long-tail items for different transformations. As shown in this table, some version of the percentile transform significantly outperforms all others for each dataset/algorithm combination in terms of nDCG@10. In only 4 of the 24 conditions are the improvements not significant: on CiaoDVD when Per_u^l is used as input for BiasedMF, on FilmTrust when Per_u^m is used as input for SVD++, and ML1M when Per_u^f is used as input. We can therefore conclude that the improvements in recommendation accuracy shown on the datasets overall are not accruing only to the popular items in the distribution but are shared among the difficult to recommend long-tail items as well.



(a) BX (b) CiaoDVD (c) FilmTrust

Fig. 3. Percentage of users who provided identical ratings.

5 SMOOTHED TRANSFORMATION

A drawback of the percentile transform is the handling of a *uniform* user profile that consists entirely of identical ratings, for example, $\langle 3, 3, 3, 3, 3, 3\rangle$. When a user rates every item with the same rating values, it is hard to determine user's preferences and attitudes: If the user is generous (tends to rate highly), then a rating value of 3 can be interpreted as *dislike*, while if user is stingy (tends to rate low), then the same value can be interpreted as *like*. But in the absence of a rating distribution for a given user, it is impossible to tell which assumption is correct.[15]

Figure 3 shows the percentage of users with uniform profiles at different rating values in three datasets[16]: BX, CiaoDVD, and FilmTrust. In CiaoDVD as the sparsest dataset, more than half of the users have uniform profiles, with almost 40% rating all items at 5. These profiles provide little information for a recommendation algorithm beyond the implicit association of user and item.

To overcome the problem of uniform profiles, we introduce the notion of a *smoothed* percentile transformation. Our inspiration for this method is additive (Laplace) smoothing as commonly found in naive Bayes classification. The effect of additive smoothing is to shrink probability estimates based on counts toward a uniform probability; here our goal is to shrink the percentile estimate toward a uniform (flat) distribution across the rating values. To create a smoothed version of the percentile, we add a small number of artificial ratings, k, at each rating level. In a five-star rating system, for example, possible rating values are 1, 2, 3, 4, 5, so a $k = 2$ smoothed transform of the profile $\langle 3, 3, 3\rangle$ yields the smoothed profile $\langle 1, 1, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5\rangle$.

After the smoothed profile is created, the percentile transformation is computed and then the artificial rating values are removed, leaving behind the altered percentiles for the original rating values. Thus, the profile consisting only of 3s, as in our example above, would have middling percentile scores, being transformed to $\langle 64, 64, 64\rangle$, using the last index method. If the profile had been $\langle 5, 5, 5\rangle$ instead, then the transformed version would be $\langle 93, 93, 93\rangle$. The effect of the smoothed transform is therefore to place the user profile in the context of the full rating scale.

We formalize our smoothed version of the percentile transformation for each index assignment as follows:

$$p^f(x, M) = \frac{100 \times (position^f(x, o(M)) + (k \times (index(x) - 1)))}{|M| + (|R| * k) + 1}, \tag{7}$$

$$p^m(x, M) = \frac{100 \times (position^m(x, o(M)) + (k \times (index(x) - 1)) + k/2)}{|M| + (|R| * k) + 1}, \tag{8}$$

[15]Note that this issue can be even more problematic for some other transformation techniques: z-score, for example, is undefined for uniform profiles.

[16]There are no uniform user profiles in ML1M.

Table 9. Performance of Recommendation Algorithms with Smoothed Percentile as Input at nDCG@10

Dataset	Algorithms	$SPer_u^f$	$SPer_u^m$	$SPer_u^l$
BX	BiasedMF	**0.014**	**0.014**	**0.013**
	SVD++	**0.014**	**0.013**	**0.012**
CiaoDVD	BiasedMF	**0.027**	**0.027**	**0.026**
	SVD++	**0.019**	**0.018**	**0.016**
FilmTrust	BiasedMF	**0.352**	**0.345**	**0.335**
	SVD++	**0.081**	**0.092**	**0.102**
ML1M	BiasedMF	**0.157**	**0.158**	**0.156**
	SVD++	**0.162**	**0.166**	**0.157**

$$p^l(x, M) = \frac{100 \times (position^l(x, o(M)) + (k \times index(x)))}{|M| + (|R| * k) + 1}, \tag{9}$$

where $index(x)$ returns the index of rating x in the full list of rating values available in the application. In rating system such as {0.5,1,1.5,2,2.5,3}, for example, $index(1) = 2$ or $index(2.5) = 5$. $|R|$ is the number of rating values available to users (i.e., in a five-star rating system, $|R| = 5$).

We repeated our prior experiments using these smoothed transforms, achieving the results shown in Table 9. On the FilmTrust dataset, the smoothed percentile showed significantly improvement over the percentile technique particularly on BiasedMF algorithm. On BX dataset, results are only slightly better than percentile values. One might attribute this result to the fact that there are few uniform profiles ratings in BX dataset. However, although ML1M does not have any users with uniform profiles, the smoothed percentile showed significant improvement over the percentile technique, indicative of effectiveness of smoothing even on non-uniform profiles.

On the CiaoDVD dataset, we expected significantly better results due to high number of users who provided identical ratings. However, the improvement by smoothed percentile is only slightly better than percentile transform. One possible reason for this result is because most of the users who provided identical ratings are cold-start users with few items rated.

6 CONCLUSIONS

In this article, we presented a rating transformation model that converts rating values to percentile values as a pre-processing step before model generation. This technique addresses two well-known problems in ratings distributions in recommender systems: the problem of user rating bias, due to variation in rating behavior, and the problem of right-skew, due to the selection bias toward preferred items. This simple pre-processing step produces improved recommendation ranking performance across multiple datasets, multiple algorithms, and multiple evaluation metrics. In addition, we introduced the *smoothed percentile* transformation to overcome the problem of identical ratings in users profiles. Experimental results showed that the smoothed percentile technique can improve recommendation performance beyond the percentile technique alone, even in cases where uniform profiles are not present.

In introducing these transformations and demonstrating their benefits for recommender system performance, we also introduced the concept of distribution flatness and produced suggestive evidence that distributional flatness may be a good indicator of the benefits of such rating transformations: flatter, indeed, seems to be better when it comes to rating value distributions for recommendation.

In future work, we plan to conduct additional experiments with the percentile transform, particularly the item-based version of the transform, which was introduced here but for which no result were presented. Early experiments indicate that on algorithms that are item oriented (for example, the Sparse Linear Method [Ning and Karypis 2011]), the item-oriented version of the transform is more appropriate.

We also plan to explore alternative approaches to enhancing the flatness of user profiles including negative sampling. Negative sampling has been shown to improve classification accuracy when the evidence is biased [Goldberg and Levy 2014]. For example, rather than adding artificial ratings just for the percentile computation and removing them afterwards, it may be useful to sample items with different average rating values and use them to augment uniformity of user profiles. This would have the effect of smoothing such low-information profiles both toward flatness and toward the population average for item preferences.

Finally, we plan to investigate the effectiveness of the proposed percentile transformation on generating fair recommendations to different groups of users with respect to sensitive attributes [Ekstrand et al. 2018; Mansoury et al. 2019]. Recently, for example, it has been shown that entropy of users profile can be one factor that leads to group unfairness in recommender systems [Mansoury et al. 2020]. Although percentile transformation does not change the entropy of an individual user's profile, our initial studies indicate that it will significantly increase the entropy of the aggregate profiles of user groups (e.g., male or female users). Therefore, we will study the effectiveness of percentile transformation as a pre-processing technique on improving the consumer-side group fairness.

REFERENCES

Panagiotis Adamopoulos and Alexander Tuzhilin. 2013. Recommendation opportunities: Improving item prediction using weighted percentile methods in collaborative filtering systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 351–354.

Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* 23, 1 (2005), 103–145.

Gediminas Adomavicius and Alexander Tuzhilin. 2015. Context-aware recommender systems. In *Recommender Systems Handbook*. Springer US, 191–226.

Erik Brynjolfsson, Yu Jeffrey Hu, and Michael D. Smith. 2006. From niches to riches: Anatomy of the long tail. *Sloan Manage. Rev.* 47, 4 (2006), 67–71.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 39–46.

Veer Sain Dixit and Parul Jain. 2019. Weighted percentile-based context-aware recommender system. *Applications of Artificial Intelligence Techniques in Engineering* (2019), 377–388.

Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. 2018. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In *Proceedings of the Conference on Fairness, Accountability and Transparency*. 172–186.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722*. Retrieved from https://arxiv.org/abs/1402.3722.

Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java library for recommender systems. In *Proceedings of the ACM User Modeling, Adaptation and PersonalizationWorkshops (UMAP'15)*.

Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2013. A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*.

Omar Hasan, Lionel Brunie, Jean-Marc Pierson, and Elisa Bertino. 2009. Elimination of subjectivity from trust recommendation. In *Proceedings of the IFIP International Conference on Trust Management*. Springer, Berlin, 65–80.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 230–237.

Rob J. Hyndman and Yanan Fan. 1996. Sample quantiles in statistical packages. *Am. Stat.* 50, 4 (Nov, 1996), 361–365.

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: An analysis of recommendation biases and possible countermeasures. *User Model. User-Adapt. Interact.* 25, 5 (2015), 427–491.

Rong Jin and Luo Si. 2004. A study of methods for normalizing user ratings in collaborative filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 568–569.

Toshihiro Kamishima and Shotaro Akaho. 2010. Nantonac collaborative filtering: A model-based approach. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 273–276.

Soo-Cheol Kim, Kyoung-Jun Sung, Chan-Soo Park, and Sung Kwon Kim. 2016. Improvement of collaborative filtering using rating normalization. *Multimedia Tools Appl.* 75, 9 (May 2016), 4957–4968.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

Eric Langford. 2006. Quartiles in elementary statistics. *J. Stat. Educ.* 14, 3 (Nov. 2006), 1–27.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems* (2001), 556–562.

Masoud Mansoury, Himan Abdollahpouri, Jessie Smith, Arman Dehpanah, Mykola Pechenizkiy, and Bamshad Mobasher. 2020. Investigating potential factors associated with gender discrimination in collaborative recommender systems. In *Proceedings of the 33rd International Flairs Conference*.

Masoud Mansoury and Robin Burke. 2019. Algorithm selection with librec-auto. In *Proceedings of the ECIR Interdisciplinary Workshop on Algorithm Selection and Meta-Learning in Information Retrieval (AMIR'19)*. 11–17.

Masoud Mansoury, Robin Burke, Aldo Ordonez-Gauger, and Xavier Sepulveda. 2018. Automating recommender systems experimentation with librec-auto. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 500–501.

Masoud Mansoury, Bamshad Mobasher, Robin Burke, and Mykola Pechenizkiy. 2019. Bias disparity in collaborative recommendation: Algorithmic evaluation and comparison. In *Proceedings of the RecSys Workshop on Recommendation in Multistakeholder Environments (RMSE'19)*.

Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. 2007. Collaborative filtering and the missing at random assumption. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 267–275.

Xia Ning and George Karypis. 2011. SLIM: Sparse linear methods for top-N recommender systems. In *Proceedings of the IEEE 11th International Conference on Data Mining (ICDM'11)*. IEEE, 497–506.

Yoon-Joo Park and Alexander Tuzhilin. 2001. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08)*. 11–18.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. ACM, 175–186.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*. 285–295.

Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 269–272.

Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: Dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems*. 113–120.

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: A multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.

Received June 2019; revised June 2020; accepted November 2020