Item Popularity and Recommendation Accuracy

Harald Steck
Bell Labs, Alcatel-Lucent
Murray Hill, NJ
Harald.Steck@alcatel-lucent.com

ABSTRACT

Recommendations from the long tail of the popularity distribution of items are generally considered to be particularly valuable. On the other hand, recommendation accuracy tends to decrease towards the long tail. In this paper, we quantitatively examine this trade-off between item popularity and recommendation accuracy. To this end, we assume that there is a selection bias towards popular items in the available data. This allows us to define a new accuracy measure that can be gradually tuned towards the long tail. We show that, under this assumption, this measure has the desirable property of providing nearly unbiased estimates concerning recommendation accuracy. In turn, this also motivates a refinement for training collaborative-filtering approaches. In various experiments with real-world data, including a user study, empirical evidence suggests that only a small, if any, bias of the recommendations towards less popular items is appreciated by users.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications–*Data Mining*

General Terms

Algorithms

Keywords

Recommender Systems

1. INTRODUCTION

The idea of recommender systems (RS) is to automatically suggest items to each user that s/he may find appealing. The performance of recommender systems can be assessed with respect to various criteria, like accuracy, diversity and serendipity, among others (e.g. see [9, 23] for an overview).

Recommendations from the long tail and serendipitous recommendations are generally considered to be particularly valuable to users [23, 1]. On the other hand, it is well-known that recommendation accuracy tends to decrease towards the long tail. In this paper, we present a quantitative approach for offline-testing of this trade-off between recommendations from the long tail (i.e. less popular items) and recommendation accuracy.

As to combine accuracy and item popularity into a single new performance measure, we start with the assumption that there is a popularity bias in the available historical data, and then modify the accuracy measure as to correct for this popularity bias. This serves two goals. First, if the *assumed* popularity bias is equal to the (unknown) *true* popularity bias in the data, then the resulting accuracy is corrected for the popularity bias in the data. Second, when the *assumed* popularity bias increasingly exceeds the (unknown) *true* popularity bias in the data, the accuracy measure increasingly favors items from the long tail. While the first objective (bias correction) has a long history in the statistics literature on nonresponse in surveys and related sampling techniques [21, 8, 3, 22], we use these same methods in our second objective as to tune the accuracy measure towards the long tail.

The correction for popularity bias is not only applicable to the test metric (see Section 2), but also to the training objective function (see Section 3). Section 4 illustrates the huge quantitative changes caused by an increased popularity bias in numerous experiments on the Netflix data. In Section 5, we carried out a user study as to determine the degree of bias towards the long tail, as appreciated by the users.

2. POPULARITY BIAS

The basic idea of *popularity bias* is that users are more likely to provide feedback on popular/mainstream items than on unpopular/niche items. In other words, the observed user feedback may be biased towards popular items, compared to the user's true interest. The exact definition is provided in Section 2.1. The resulting accuracy measure, which we name *popularity-stratified recall*, is able to correct for such a popularity bias in the data. It may also be used to overcompensate for the popularity bias, which effectively results in an accuracy measure that is biased towards the long tail.

2.1 Popularity-Stratified Recall

In this section, we formalize popularity bias for offline testing, which in turn implies the corresponding modification to training (see Section 3). Conceptually, it is important to distinguish between the observed data and the (unknown) complete data, where each user has provided feedback on all

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.
Copyright 2011 ACM 978-1-4503-0683-6/11/10 ...$10.00.

items. Note that recommendation accuracy experienced by the user in a real-world application is determined by the performance regarding the complete data rather than the available data (as the latter contains only items already rated by, and hence known to, the user), see also [24].

While there are various accuracy measures (e.g. see [9, 23]), we follow [13, 24, 4] and use *recall* as our test measure. As suggested in [24, 4], and as supported by our user study in Section 5, recall on all items is well aligned with recommendation accuracy as perceived by the user.

While recall can be applied naturally to binary feedback data (e.g. purchases, clicks), explicit feedback data (e.g. rating values 1,...,5) have to be discretized into two values: **relevant** and not relevant to a user. For instance, in our experiments with Netflix data, we consider 5-star ratings as relevant to a user (i.e. the user definitely likes these movies),[1] while we consider all items with other or no rating values as not relevant.

In [24], it was shown that recall on all items on available data provides a nearly unbiased estimate of recall on the (unknown) *complete* data, under the following assumption: the *relevant* ratings are missing at random, while an arbitrary missing data mechanism may apply to all other rating values (as long as they are missing with a higher probability). This is much milder than assuming that all the ratings are missing at random, which is implied by testing on observed ratings only (e.g. using root mean square error–RMSE).

We now relax this assumption further [25]: concerning the *relevant* ratings, we assume that the probability of a user u providing feedback depends on the popularity of item i, $p_{\text{obs}}(i|u) = p_{\text{obs}}(i)$, where we define the **popularity** of an item by the number $N^{+}_{\text{complete},i}$ of *relevant* ratings it obtained in the (unknown) complete data. Let $N^{+}_{\text{obs},i}$ be the number of relevant ratings observed in the available data; then the (empirical) probability of observing a relevant rating for item i is

$$p_{\text{obs}}(i) = \frac{N^{+}_{\text{obs},i}}{N^{+}_{\text{complete},i}}. \qquad (1)$$

Following the established approaches on nonresponse in surveys (see Section 2.3), we modify the usual recall measure by introducing a weight proportional to the inverse response probability, s_i. This results in our definition of **popularity-stratified recall** (for user u):

$$\text{recall}_u@k = \frac{\sum_{i \in S_u^{+,k}} s_i}{\sum_{i \in S_u^{+}} s_i} \quad \text{with} \quad s_i = \frac{1}{p_{\text{obs}}(i)}, \qquad (2)$$

where S_u^{+} denotes the set of *relevant* items of user u with observed ratings in the observed data; $S_u^{+,k}$ is the subset of relevant items ranked in the top k items based on the predictions of the RS. The average recall across all users is defined as

$$\text{recall}@k = \sum_u w^u \text{recall}_u@k, \qquad (3)$$

where we allow for different weights w^u across users. These weights are normalized, $\sum_u w^u = 1$, like in [24].

Theorem: *Under the assumption that there are no other hidden causes/confounders besides the popularity bias, the popularity-stratified recall as defined in Eqs. 2 and 3 provides a (nearly)*[2] *unbiased estimate of the true recall (i.e. on the unknown complete data),*

$$\text{recall}_{\text{complete},u}@k = \frac{N^{+,k}_{\text{complete},u}}{N^{+}_{\text{complete},u}}, \qquad (4)$$

$$\text{recall}_{\text{complete}}@k = \sum_u w^u \text{recall}_{\text{complete},u}@k, \qquad (5)$$

where $N^{+}_{\text{complete},u}$ is the number of relevant items of user u, and $N^{+,k}_{\text{complete},u}$ is the number of relevant items ranked in the top-k in the (unknown) complete data.

Proof: This follows analogous to [24]. □

In our experiments, we chose $w^u \propto \sum_{i \in S_u^{+}} s_i$ as a generalization of the definitions in [13, 24]. This choice of weights has interesting properties. First, recall across all users and all relevant test items becomes

$$\text{recall}@k = \frac{\sum_u \sum_{i \in S_u^{+,k}} s_i}{\sum_u \sum_{i \in S_u^{+}} s_i}, \qquad (6)$$

where the enumerator and denominator can each be expected to be a fairly large number. Hence, the bias of this ratio estimator can be expected to be negligible (see Section 2.3). Note that the bias of Eq. 3 may be somewhat larger for other choices of weights w^u, as the sample size for an individual user might be small. Apart from that, note that $\sum_{i \in S_u^{+}} s_i$ is an estimate that is proportional to the (unknown) number of relevant items of user u in the complete data, $N^{+}_{\text{complete},u}$. This choice hence weights each user by his/her (true) *activity* in the complete data, when activity is measured in terms of the number of relevant ratings assigned by user u.

Apart from that, and analogous to [24], note that–*for fixed data and fixed k*–the average recall in Eq. 3 is proportional to the average popularity-stratified precision, which is defined as $\text{prec}@k = \sum_u w^u_{\text{prec}} \sum_{i \in S_u^{+,k}} s_i / k$ for $w^u_{\text{prec}} \propto w^u / \sum_{i \in S_u^{+}} s_i$. For instance, recall with user weights $w^u \propto \sum_{i \in S_u^{+}} s_i$, like in our experiments, is proportional to precision with uniform weights across users. Precision computed from the observed data, however, cannot provide an unbiased estimate concerning the (unknown) complete data, as the fraction of observed relevant ratings is unknown [24]. Note that this proportionality does not contradict the usual tradeoff between recall and precision, which is entailed by varying k.

2.2 Power-law of Popularity

Given that complete data is unavailable (at low cost), p_{obs} in Eq. 1 cannot be calculated in practice. We now consider the case where p_{obs} is a smooth function of the items' popularities, in particular a *power-law*

$$p_{\text{obs}}(i) \propto \left(N^{+}_{\text{complete},i}\right)^{\gamma} \qquad (7)$$

with exponent $\gamma \in \mathbb{R}$. This is motivated by the power-law behavior of the observed relevant ratings (see Figure 2 a). Eq. 7 ensures in turn that also the (unknown) complete data follows a power-low distribution concerning the relevant ratings. This results in the stratification weights

$$s_i \propto 1/(N^{+}_{\text{obs},i})^{\beta} \quad \text{where} \quad \beta = \frac{\gamma}{(\gamma + 1)}. \qquad (8)$$

[1]When we considered both 4 and 5 star ratings as relevant, we obtained qualitatively identical results/conclusions.

[2]Concerning the remaining bias of ratio estimators, see Section 2.3. Note that this bias diminishes in large samples.

Note that the unknown proportionality factor cancels out in Eq. 2. The popularity-stratified recall depends on a free parameter, β (or equivalently γ). The usual recall measure is obtained if $s_i = 1$ for all items i, or equivalently $\gamma = \beta = 0$. If $\beta = 0.5$, the probability of observing relevant ratings increases *linearly* with item popularity ($N^+_{\text{complete},i}$). The extreme case when $\beta \to 1$, i.e. $\gamma \to \infty$, has several interesting properties: first, one would observe in the available data only relevant ratings of the item with the largest popularity in the complete data. This does not agree with empirical evidence. Second the stratification weights s_i are inversely proportional to the number of observed relevant ratings $N^+_{\text{obs},i}$; this means that every item has the same overall contribution to the recall-measure in Eq. 3, independent of its number of observed relevant ratings. In other words, once an item obtains its first relevant rating, it is weighted the same as each of the other items, which may have obtained thousands of relevant ratings. This extreme case ($\beta \approx 1$) obviously has low robustness against statistical noise as well as against manipulation and attacks. As this is the limiting case, we nevertheless provide experimental results for this extreme scenario in Section 4. The realistic value of β can be expected to be less extreme, as also supported by the user study in Section 5.

This may also be viewed from the perspective of the well-known bias-variance trade-off in machine learning and statistics. Due to this trade-off, regularization is typically used to reduce the variance (at the price of increased bias) of the trained model, and hence to avoid over-fitting. As discussed in the previous paragraph, the different item popularities give rise to another such bias-variance trade-off: recommendations from the long tail of the popularity distribution may have a small bias, but this comes at the price of large variance, which reduces prediction accuracy. The parameter β may hence be interpreted as an additional regularization parameter that determines as to which degree the items from the long tail are recommended, i.e. the trade-off between popularity bias and variance. From the discussion in the previous paragraph, it is intuitively clear that the variance has to be restricted to some degree, which is achieved by $\beta \ll 1$. Note that a shift in the popularity bias variance trade-off towards smaller variance, and hence larger bias towards popular items was also found very useful in various other domains (e.g. bestseller list, PageRank [14]).

Even though it does not fit with our framework, we also considered to measure item popularity in terms of the number of (any) ratings it received (rather than *relevant* ratings), and found that the experimental results were very similar.

If the test set is a random subset of the observed data, then $N^+_{\text{obs},i}$ may be determined from the test set. Given that the training set is typically much larger than the test set, it may be more robust to determine $N^+_{\text{obs},i}$ based on all available data, i.e. the training and test set combined. We used the latter choice in our experiments, but the former led to very similar results in the experiments.

Obviously, stratification like in Eq. 2 carries over to other measures, like ATOP [24] or the area under the ROC curve.

2.3 Related Work on Popularity Bias

There is a rich body of work in statistics on non-response in surveys, e.g. see [21, 8, 3, 22] for an overview. Non-response can introduce a severe bias in the survey data, as the given answers to the survey-questions and the person's ability/decision to respond to the survey may be confounded. Accounting for bias in survey data has long been recognized as a crucial problem in the statistics literature. The basic idea underlying numerous sophisticated approaches (e.g. propensity score [18] weighting [17]) is to adjust the *weight* of each response as to reduce the bias of the estimated quantity. This weight is typically chosen inversely proportional to the estimated response probability (e.g. see [12] for asymptotic properties of such an estimator).

Apart from that, it is generally difficult or impossible to define *(exactly) unbiased* ratio estimators, except for special cases [3]. On the other hand, 'nearly' unbiased ratio estimators are often easy to define, where 'nearly' means that their bias decreases quickly with sample size [3]. Note that recall, which we will use as accuracy measure in this paper, is a ratio estimator; however, the sample size in the context of recommender systems is typically large enough so that the bias in Eq. 6 can be ignored.

3. MODEL TRAINING

In this paper, we focus on low-rank matrix factorization models, as they were found to be one of the most accurate single models for collaborative filtering.

3.1 Review

This section briefly reviews the two approaches used in the experiments: MF-RMSE, which optimizes the popular RMSE, and AllRank, which is geared towards recall [24]. They are both based on a low-rank matrix-factorization (MF) model: the matrix of predicted ratings $\hat{R} \in \mathbb{R}^{i_0 \times u_0}$, where i_0 denotes the number of items, and u_0 the number of users, is modeled as

$$\hat{R} = r_m + PQ^\top, \tag{9}$$

with matrices $P \in \mathbb{R}^{i_0 \times j_0}$ and $Q \in \mathbb{R}^{u_0 \times j_0}$, where $j_0 \ll i_0, u_0$ is the rank; and $r_m \in \mathbb{R}$ is a (global) offset.

For computationally efficient training, the square error (with the usual L2-norm regularization) is used (see [24]):

$$\sum_{\text{all } u} \sum_{\text{all } i} W_{i,u} \cdot \left\{ \left(R^{\text{o\&i}}_{i,u} - \hat{R}_{i,u} \right)^2 + \lambda \left(\sum_{j=1}^{j_0} P^2_{i,j} + Q^2_{u,j} \right) \right\}, \tag{10}$$

where $\hat{R}_{i,u}$ denotes the ratings predicted by the model in Eq. 9; and $R^{\text{o\&i}}_{i,u}$ equals the actual rating value in the training data if observed for item i and user u; otherwise the value $R^{\text{o\&i}}_{i,u} = r_m$ is imputed. The key is in the training weights,

$$W_{i,u} = \begin{cases} w(R^{\text{obs}}_{i,u}) & \text{if } R^{\text{obs}}_{i,u} \text{ observed} \\ w_m & \text{otherwise} \end{cases}, \tag{11}$$

where $w(R^{\text{obs}}_{i,u}) = 1$ for the observed ratings in the training data. In **AllRank** [24], the weight assigned to the imputed ratings is *positive*, i.e. $w_m > 0$ [24]. In contrast, **MF-RMSE** is obtained for $w_m = 0$. This seemingly small difference has the important effect that AllRank is trained on *all* items, while MF-RMSE is trained only on the *observed* ratings. Hence, *MF-RMSE is geared toward the popular objective of minimizing RMSE on observed ratings, while AllRank was found to achieve a considerably larger top-k hitrate or recall on all items than the various state-of-the-art approaches, see results in [24] and compare to [13, 4].*

We use *alternating least squares* for computationally efficient training of AllRank [24] and MF-RMSE.

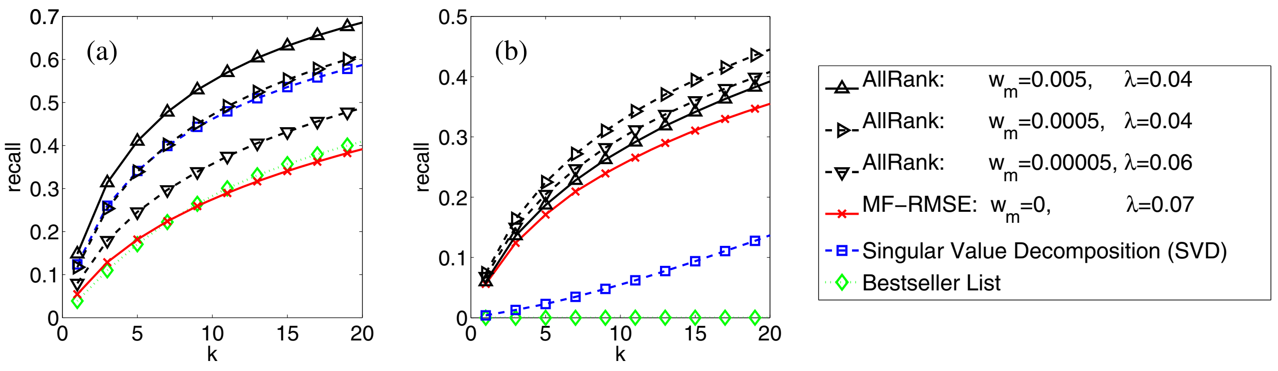

Figure 1: Recall@k on Netflix probe set (graph a), and when the 10% most-rated items were removed (graph b). In both cases, the largest recall is achieved by AllRank [24]. The low recall of SVD and bestseller-list in graph (b) reflects that their recommendations are the most biased towards popular items.

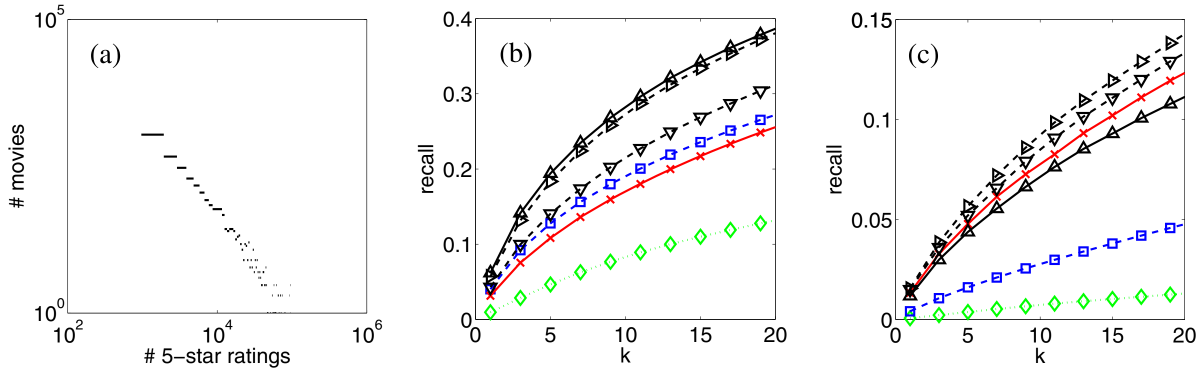

Figure 2: Netflix data [2]: (a) the number of relevant (i.e. 5-star[1]) ratings per movie in the training data shows a close to power-law distribution (i.e. straight line in the log-log plot). Popularity-stratified recall on the probe set for (b) $\beta = 0.5$, i.e. probability of observing a relevant rating increases *linearly* with item popularity; (c) unrealistic extreme case where $\beta = 1$, i.e. power-law exponent $\gamma \to \infty$. See legend in Figure 1. As β increases, the optimal w_m in AllRank decreases, but interestingly stays positive even in the extreme.

While there are several state-of-the-art approaches (e.g. [7, 11, 13, 16, 19]) that achieve a lower RMSE on *observed* ratings than MF-RMSE does, note that their test performances on *all* (unrated) items is quite similar to each other, e.g. see [13, 4]. This is interesting, as some of these approaches, like [16, 19, 13, 20], actually account in some sense for the fact that ratings are missing not at random from the observed data–but in the context of *observed* ratings (as to minimize RMSE). Note that AllRank does not only apply to explicit feedback data, but can also be used for implicit feedback data, similar to [10, 15].

3.2 Popularity-Stratified Training

The popularity-stratified test based on the power-law distribution immediately implies a simple modification to the AllRank training, which we name **AllRank-Pop**: the observed ratings of item i in the training data are assigned the weight $w(R^{\text{obs}}_{i,u}) \propto (1/N^{+}_{\text{tr},i})^{\beta_w}$, where $N^{+}_{\text{tr},i}$ is the number of relevant ratings of item i in the training data; the exponent $\beta_w \in [0, 1]$ can be expected to be similar in value to β in Eq 8. Note that AllRank-Pop with $\beta_w = 0$ is equivalent to AllRank. For numerical stability, one may add a small number, like 1, to the denominator. As to maintain the overall balance between observed and missing ratings in the training data, we use the normalization $\sum_{(u,i)\text{obs.}} w(R^{\text{obs}}_{i,u}) = \#\text{ratings}$, i.e. the cumulative weight of all observed ratings in the training data is invariant under β_w (including $\beta_w = 0$, which corresponds to the original case where all observed ratings have unit weight). This ensures that the relative weight between the observed ratings and the imputed ratings stays put for different β_w.

An alternative to *decreasing* the weight of observed ratings for popular items and using a fixed weight for all missing ratings, is to instead *increase* the weight of the missing ratings for popular items and to use a fixed weight (of 1) for all observed ratings: a missing rating of item i may be assigned the weight $w_m(i) \propto {N^{+}_{\text{tr},i}}^{\beta_w}$ with normalization $\langle w_m(i) \rangle_i = w_m$, where $\langle \cdot \rangle$ denotes the average. In our experiments, increasing as opposed to decreasing the training weights led to slightly better test results (see Figure 3). In our Netflix experiments, we retained the parameter settings for AllRank-Pop, i.e. $r_m = 2$, $w_m = 0.005$ and $\lambda = 0.04$, and only changed the new parameter β_w; modifying r_m, w_m or λ did not yield significant improvements.

4. EXPERIMENTS AND DISCUSSION

In this section, we discuss the popularity bias from several perspectives. While we use the Netflix data [2] as example

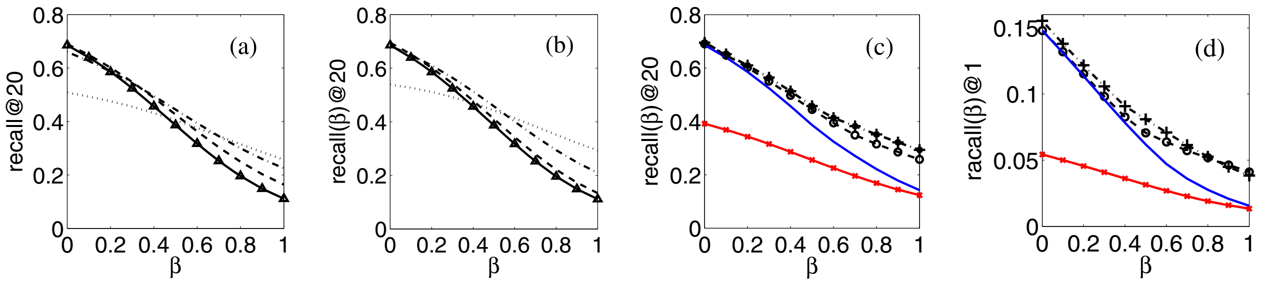

Figure 3: Popularity-stratified Recall(β)@k on Netflix data: (a) AllRank-Pop trained with $\beta_w = 0, 0.2, 0.5, 0.9$ (curves in this order from bottom to top at $\beta = 1$ in the graph); (b) AllRank-Pop (using increasing instead of decreasing weights, as discussed in Section 3.2) results in similar test results compared to graph (a), yet slightly higher recall@20 for larger β_w; (c) and (d) show the maximum recall achievable by AllRank-Pop (for increasing (+) and decreasing ($\circ$) weights), AllRank and MF-RMSE (curves in this order from top to bottom) for $k = 20$ and $k = 1$, respectively. AllRank-Pop improves over MF-RMSE in recall for all $\beta \in [0, 1]$ by over 75% at $k = 20$, and over 170% at $k = 1$.

in this paper, note that we obtained qualitatively identical experimental results/conclusions on the MovieLens data [5]. Note, however, that the degree of popularity bias may vary across different kinds of items (e.g. movies vs. songs). The Netflix Prize data [2] contains 17,770 movies and almost half a million users. About 100 million ratings are available. Ratings are observed for about 1% of all possible movie-user pairs. The ratings take integer values from 1 (worst) to 5 (best). The provided data are already split into a training and a probe set. We removed the probe set from the provided training set as to split these data into a training set and a disjoint test set.

4.1 Recall

The experimental results concerning popularity-stratified recall are obtained on all items, i.e. we used a random subsample of 1,000 additional items with no ratings in the test or training set for each user, and show results for the top 20 items, as to allow direct comparison to results in [13, 24, 4].

Figure 1 provides a first assessment of the popularity bias inherent in the recommendations of various approaches: graph (a) shows (the usual) recall on the probe set, while graph (b) shows (the usual) recall on the probe set when the 10% most-rated items were removed, i.e. recommendations of a popular movie are not counted towards recall. In both cases, AllRank–trained with different weights $w_m > 0$– clearly achieves the largest recall, even though the advantage over MF-RMSE is reduced in graph (b). This shows that an increased weight w_m in AllRank results in an increased bias towards popular movies. Nevertheless, in graph (b) a *small positive weight* is still beneficial compared to MF-RMSE, which ignores missing ratings. While singular value decomposition (SVD) and the bestseller list (items ranked according to number of ratings in training set) achieve surprisingly high recall in graph (a), their recall is–not surprisingly– considerably reduced in (b). SVD implicitly imputes a zero value (with weight $w_m = 1$) for all missing ratings, and was successfully used for *latent semantic indexing* [6]. Note that SVD achieved a higher recall than any state-of-the-art RMSE-based approach (including SVD++ [13]) in the experiments in [4] when 0,...,2% of the most-rated items were removed from the Netflix probe set.

Note that removing the 10% most-rated items in graph (b) may be quite extreme from two perspectives: first, over 75% of the ratings are assigned to the top 10% most-rated items in the Netflix data; second, Wal-Mart carries about 2.5% of the (music) albums released each year, and 0.5% of all the music available [1], p. 156. This suggests that the head of the popularity distribution often contains only a few percent ($\ll 10\%$) of the items.

While the removal of the most popular items from the test set provides a simple way of testing an RS's bias towards popular items, it has several disadvantages: (1) there may not exist a sharp cut-off in a real application, (2) the problem of the popularity bias is merely shifted to the remaining items, and (3) it is not reflected by the distribution shown in Figure 2 (a): instead of an abrupt change, the data points agree well with a straight line in this log-log plot. This suggests a power-law behavior in the Netflix data. If the power-law exponent concerning the complete data were known, then the power-law exponent γ (and hence β) of the popularity bias in Eq. 7 could be calculated.

We now consider *popularity stratified recall*, as defined in Section 2.1 with power-law exponent $\gamma \in [0, \infty)$ (or equivalently $\beta \in [0, 1]$) as our test measure in the remainder of this paper. While Figure 1(a) shows recall for $\beta = 0$, **Figure 2** summarizes the results for $\beta = 0.5$ (graph b), and $\beta = 1$ (graph c). Obviously, as β increases, recall decreases considerably for all recommender systems. Moreover, AllRank achieves a larger recall than MF-RMSE for all $\beta \in [0, 1]$– however, the optimal weight w_m decreases as β increases (but it is always a *positive* number, rather than zero, like in MF-RMSE). As in Figure 1 (b), the tendency of SVD and bestseller list to focus on popular items is reflected by the sharply reduced recall for large β.

Figure 3 shows the improvement of AllRank-Pop over AllRank and MF-RMSE regarding recall across all values of $\beta \in [0, 1]$: graph (a) shows that AllRank-Pop trained with an increased β_w achieves a considerable increase in recall at large β, while the decrease in recall at small β is rather small (unless β_w is close to 1). Graph (a) also indicates that the training parameter β_w that optimizes recall(β)@20 on the test set is close to β; only in the extremes, we find some deviations: for $\beta = 0$, the optimal value is $\beta_w \approx 0.15$, and for $\beta = 1$, $\beta_w \approx 0.9$. While graph (a) shows the results for the AllRank-Pop variant that *decreases* the weight of observed ratings for popular items, graph (b) refers to the AllRank-Pop variant that *increases* the weight of missing ratings for

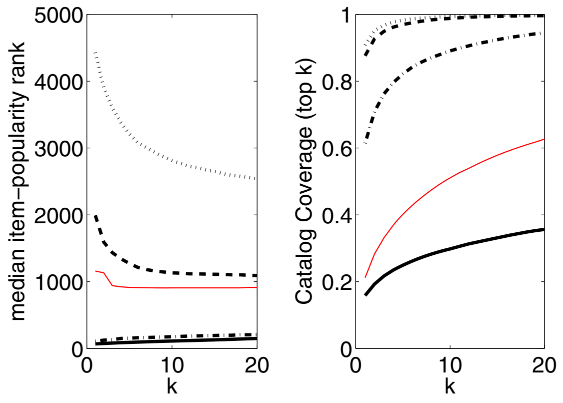

Figure 4: Median popularity-rank (left) and catalog coverage (right) for the recommended top-20 movies (across all users) in the Netflix data. We compared MF-RMSE (thin solid line) with AllRank-Pop (thick lines) trained with various $\beta_w = 0, 0.5, 0.9, 1.0$ (curves in this order from bottom to top in both graphs).

popular items, as discussed in Section 3.2. Both variants obviously achieve similar test results, with the latter variant resulting in slightly higher recall for larger β_w. The various curves in graphs (a) and (b) are each summarized by a single curve in graph (c), which reflects the maximally achievable recall(β)@20 of AllRank-Pop. For comparison, graph (c) also shows the maximally achievable recall(β)@20 of AllRank (for weights $w_m = 0.005, ..., 0.00005$), as well as the recall of MF-RMSE: it is apparent that AllRank improves over MF-RMSE considerably in recall for small β (small popularity bias). AllRank-Pop additionally improves over AllRank in particular for large β (large popularity bias). As a result, AllRank-Pop considerably outperforms MF-RMSE across all values $\beta \in [0, 1]$ (any degree of popularity bias). This holds also for recall@1 in graph (d).

4.2 Popularity and Coverage

In this section, we take a look at the popularity of the recommended movies. Let us define the *popularity rank* of an item as its rank based on the number of relevant ratings it has obtained in the training data. For instance, in the Netflix data the movie with the most 5-star ratings is assigned rank 1, and the movie with the fewest 5-star ratings has rank 17,770 (ties are broken at random).

Figure 4 (left) shows the median popularity-rank of all the recommended top-k items across all users. As expected, when AllRank-Pop is trained with an increased β_w, the median popularity-rank of the top-20 recommended items increases–from a few hundred at $\beta_w = 0$ to over 2,500 at $\beta_w = 1$. For comparison, the top-20 items recommended by MF-RMSE have a median popularity-rank of slightly below 1,000 (across all users). Note that the observed ratings in the training data have a median popularity rank of 426. This shows that MF-RMSE tends to recommend movies from the long tail of the popularity distribution, while the median popularity of AllRank-Pop's recommendations can be controlled via the free parameter $\beta_w \in [0, 1]$.

Instead of the median, **Figure 5** shows the full distribution of the top 20 recommendations (across all users). The left graph shows two interesting properties of AllRank-

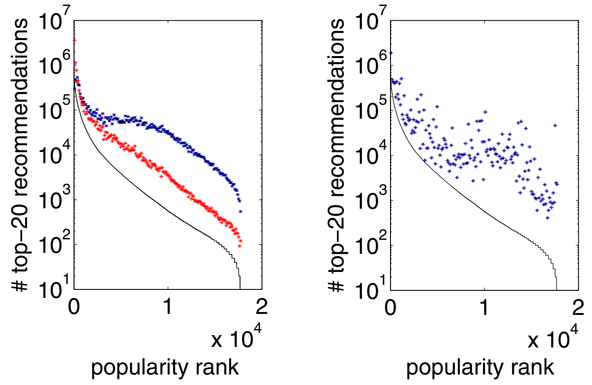

Figure 5: How often is an (un)popular movie recommended in the top-20 (across all users)? The left graph shows *two* results for AllRank-Pop, trained with $\beta_w = 0.5$ (lower point cloud) and $\beta_w = 1$ (upper point cloud). The right graph shows the results for MF-RMSE, which are less well aligned with the items' popularity ranks. Each point aggregates 100 movies. The item popularity in the training data is shown as continuous curve, and shifted vertically in the log plots to aid visual comparison.

Pop's recommendations: (1) as already indicated by the median, increasing β_w shifts the distribution towards less popular items; (2) the top 20 recommendations of AllRank-Pop (across all users) show a distribution that drops off quite *steadily and monotonically* with the observed item popularity in the training data. Note that this distribution is the *average across all users*, and hence this does not contradict personalization, which is at the core of an RS. The improved recall of AllRank in the previous section may be partially attributed to its recommendations being well calibrated regarding item popularity. Note that the information as to how many users refer to each item was also found useful in various other areas (e.g. bestseller list, PageRank [14]).

In contrast, MF-RMSE tends to recommend items with frequencies that are not well aligned with item-popularity (the points are quite spread out in Figure 5(right)): in fact, there are only a few items from the long tail that are recommended frequently, while many items are never recommend (in the top-20 across all the users in the Netflix data). This is apparent from **Figure 4 (right)**, which shows the catalog coverage for the top-k recommendations. The catalog coverage is defined as the fraction of items that are recommended at least once (across all users), from among all the items in the catalog. The top-20 recommendations of MF-RMSE cover only slightly more than 60% (i.e. almost 40% of the movies are never recommended), while AllRank-Pop achieves a coverage of over 90% already for quite small β_w.

4.3 User Perspective

In this section, we take a user-centric view. The median popularity of the items rated by each user is shown in **Figure 6 (left)**. It shows that the majority of users are mainly focused on popular movies while the number of users interested in less mainstream movies drops off quickly–note the log scale, and the median (dashed line) in the graph.

This distribution of users raises the question whether rec-

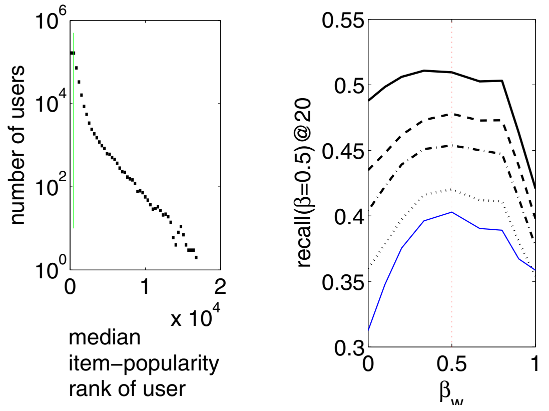

Figure 6: Netflix data [2]: (left) distribution of users based on the popularity of movies they rated; (right) popularity-stratified recall with $\beta = 0.5$ for five user groups (split by median popularity of rated movies): while recall is lower for groups who rated less popular movies, about the same popularity stratification $\beta_w \approx 0.5$ is best across all five groups.

ommendation accuracy can be improved by recommending popular items to the users who have liked popular items in the training data, and less mainstream items to the users who were off the beaten path in the training data. To this end, we split the users into five groups of approximately equal size (about 100,000) based on the median item-popularity of their rated movies. Interestingly, we did not find any improvement when testing for these five groups separately: **Figure 6 (right)** shows the popularity-stratified recall@20 with $\beta = 0.5$ for each of the five groups: the largest recall is achieved by AllRank-Pop trained with $\beta_w \approx 0.5$ *across all five groups* (with one minor exception), i.e. there is no apparent trend that the maximum of the recall curve shifts towards larger β_w values (and hence less popular movies in the recommendations) for the user groups who had rated less popular movies in the training set (lower curves in Figure 6).

We obtained the same behavior for popularity-stratified recall with other values, like $\beta = 0$ (where $\beta_w \approx 0$ was best across all groups) or $\beta = 1$ (where $\beta_w \approx 1$ was best across all groups). This suggests that, on *average* for each group, the *same* popularity stratification may be used. The remaining challenge is to determine the optimal value of β or β_w (see next section).

Apart from that, we also split the users into five groups based on their activity, i.e. the number of ratings they had assigned in the training data. In an analogous experiment, we found that there is no noticeable improvement in recall when using a different popularity stratification parameter β_w for the various user groups.

5. USER STUDY

An offline test is useful if it provides a good approximation to the utility function of interest in the real-world application, e.g. user satisfaction. Obviously, the results may depend on the domain of the items (e.g. movies vs. songs) or the kind of user population. In particular, in this user study we wanted to learn which value of the AllRank-Pop parameter β_w provides a good balance between recommendations from the head vs. the long tail of the popularity distribution of the movies in the Netflix data.

Table 1: User study: Which of the five algorithms helped you best in finding a movie to watch this weekend? We collected feedback from 20 users, resulting in significant results with p-value $5\cdot10^{-6}$. The users rated 36 movies on average and ranked the anonymized algorithms according to this question. Some users provided only a partial ranking. The cumulative user feedback indicates only a small popularity bias (i.e. small $\beta_w = 0, ..., 0.33$ is preferred).

	AllRank-Pop with $\beta_w =$				MF-RMSE
	0	0.33	0.67	0.9	
avg. rank	1.95	2.13	3.28	3.38	4.38
If ranked first, # pairwise ranking constraints ...					
... fulfilled	57	50	30	27	10
... violated	15	15	37	40	65

To this end, we conducted the following user study with the Netflix data [2]: users (i.e. employees from various departments) could enter their ratings for their movies on an internal website–by going through the list of all the movies or by searching for their titles. Based on the ratings a user had entered, the recommendations were instantly computed via alternating least squares with fixed item-matrix P (e.g. see [10, 24]), and then displayed in a table, where the top 20 recommendations were immediately visible, while up to 200 recommendations could be accessed by scrolling. The user could switch between five anonymized recommender systems, and eventually rank them on the website according to the following question: Which of the five algorithms helped you best in finding a movie to watch this weekend?

Due to the unexpected large agreement in the obtained feedback, we stopped this user study after 20 users had provided feedback. In **Table 1**, we aggregated the obtained rankings in two ways: (1) the average rank of the algorithm (i.e. 1 is best, 5 is worst), and (2) if an algorithm were ranked first, what would be the total number of fulfilled (vs. violated) *pairwise ranking constraints* (e.g. A is ranked above B) in all the users' rankings? For instance, in a ranked list of all five algorithms, each algorithm is involved in 4 pairwise ranking constraints, and there are 10 pairwise constraints (at most) for each user. As some users provided only a partial ordering, the number of pairwise constraints may differ across algorithms.

Several comments on these results are in order. First, even though unknown to the users, the popularity stratification according to the AllRank-Pop parameter β_w emerges clearly from the user feedback, i.e. the users' preferences are aligned with the popularity bias of the recommendations. Second, the user feedback indicates that there is only a *slight* popularity bias: AllRank-Pop with $\beta_w = 0$ (which is equivalent to AllRank [24]) and $\beta_w = 0.33$ are clearly preferred on average. In fact, these are also the only algorithms ranked first by more than one user: 10 and 7 times, respectively, out of 20 total. This is highly significant. The significance test with the null hypothesis that each of the five algorithms has the same probability of being ranked first by a user, yields a p-value of $5 \cdot 10^{-6}$ for AllRank-Pop with $\beta_w \in \{0, 0.33\}$

vs. AllRank-Pop with $\beta_w \in \{0.67, 0.9\}$ or MF-RMSE, using the (exact) Pearson confidence interval. This suggests that AllRank-Pop trained with a rather small value of β_w is significantly better than AllRank with a larger value of β_w and than (the vastly popular) MF-RMSE.

There was no obvious relationship between the users' preference of an RS and the number of their ratings in this user study, as was also found in the offline test in Section 4.3.

These results are also interesting from the perspective of other performance measures in the RS literature: the user's trust in the system and serendipity (recommendation of surprising yet relevant items to a user). As mentioned e.g. in [23], the users' trust may be built up by recommending some movies that they recognize and like (and hence are not serendipitous). For the task of building trust, obviously, recommendation of popular items is particularly useful, as they have an increased recall for most users. If recommendations are biased too much towards the long tail, the resulting decreased recall may result in a loss of the user's trust in the system, as well as in loss of serendipity due to the lack of the items' relevance to the user–despite their surprising nature (e.g. randomness). After all, a user has to trust in serendipitous recommendations. From this perspective, the results of the user study may be slightly biased towards smaller β_w, as the users may have paid more attention to trust than to serendipity, as this was a new website.

6. CONCLUSIONS

In this paper, we quantitatively examined the popularity-bias-variance trade-off: while recommendations from the long tail of the popularity distribution are considered more valuable, variance/noise also increases towards the long tail, degrading recommendation accuracy. In various experiments and a user study on Netflix data (movies), we found highly significant evidence that only a small, if any, bias of the recommendations towards less popular items (i.e. the long tail) is appreciated by users.

Acknowledgments

I am greatly indebted to Tin Ho for her encouragement and support, and to all participants of the user study.

7. REFERENCES

[1] C. Anderson. *The Long Tail*. Hyperion, New York, 2006.

[2] J. Bennet and S. Lanning. The Netflix Prize. In *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining*, 2007.

[3] W. G. Cochran. *Sampling Techniques*. Wiley, 1977.

[4] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-N recommendation tasks. In *ACM Conference on Recommender Systems*, pages 39–46, 2010.

[5] MovieLens data. homepage: `http://www.grouplens.org/node/73`, 2006.

[6] S. Deerwester, S. Dumais, G. Furnas, R. Harshman, T. Landauer, K. Lochbaum, Lynn Streeter, et al. Latent semantic analysis / indexing. homepage: `http://lsa.colorado.edu/`.

[7] S. Funk. Netflix update: Try this at home, 2006. `http://sifter.org/ simon/journal/20061211.html`.

[8] R. Groves, D. Dillman, J.L Eltinge, and R.J.A. Little. *Survey Nonresponse*. Wiley, 2002.

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

[10] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM)*, 2008.

[11] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–78, 2010.

[12] J. K. Kim and J. J. Kim. Nonresponse weighting adjustment using estimated response probability. *The Canadian Journal of Statistics*, 35:501–14, 2007.

[13] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 426–34, 2008.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[15] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM)*, 2008.

[16] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDDCup*, 2007.

[17] J. M. Robins, A. Rotnitzky, and L.P. Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association (JASA)*, 89:846–66, 1994.

[18] P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70:41–55, 1983.

[19] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *International Conference on Machine Learning (ICML)*, 2007.

[20] R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 2010.

[21] C. F. Särndal and S. Lundström. *Estimation in Surveys with Nonresponse*. Wiley, 2006.

[22] C. F. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer, 1992.

[23] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–97. Springer, 2011.

[24] H. Steck. Training and testing of recommender systems on data missing not at random. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 713–22, 2010.

[25] H. Steck and Y. Xin. A generalized probabilistic framework and its variants for training top-k recommender systems. In *PRSAT Workshop at RecSys Conf.,* `http://ceur-ws.org/#Vol-676`, 2010.