The Yahoo! Music Dataset and KDD-Cup'11

Gideon Dror
Yahoo! Labs

Noam Koenigstein*
Tel Aviv University

Yehuda Koren
Yahoo! Labs

Markus Weimer
Yahoo! Labs

ABSTRACT

The theme of the the KDD cup 2011 challenge was to identify user tastes in music by leveraging the actual Yahoo! Music dataset. Two datasets were sampled for the raw data: The larger dataset contained 262,810,175 ratings of 624,961 music items by 1,000,990 users was created for Track1 and and a smaller dataset with 62,551,438 ratings of 296,111 music items by 249,012 was created for Track2. A distinctive feature of the datasets is that there are four types of musical items: *tracks*, *albums*, *artists*, and *genres*, forming a four level hierarchy.

The challenge started on March 15, 2011 and ended on June 30, 2011 and attracted 2389 participants, 2100 of which were active by the end of the competition. The popularity of the challenge is related to the fact that learning a large scale recommender systems is a generic problem, highly relevant to the industry. In addition, The competition drew interest by introducing a number of scientific and technical challenges including dataset size, hierarchical structure of items, high resolution timestamps of ratings, and a non-conventional ranking-based task.

1. INTRODUCTION

People have been fascinated by music since the dawn of humanity. A wide variety of music genres and styles has evolved, reflecting diversity in personalities, cultures and age groups. It comes as no surprise that human tastes in music are remarkably diverse, as nicely exhibited by the famous quotation: "We don't like their sound, and guitar music is on the way out" (Decca Recording Co. rejecting the Beatles, 1962).

Yahoo! Music has amassed billions of user ratings for musical pieces. When properly analyzed, the raw ratings encode information on how songs are grouped, which hidden patterns link various albums, which artists complement each other, how the popularity of songs, albums and artists vary over time and above all, which songs users would like to listen to. Such an analysis introduces new scientific challenges. We have created a large scale music dataset and challenged the research world to model it through the KDD Cup 2011 contest[1]. The contest released over 250 million ratings performed by over 1 million anonymized users. The ratings are given to different types of items: tracks, albums, artists, genres, all tied together within a known taxonomy. Thousands of teams participated in the contest, trying to crack the unique properties of the dataset.

2. THE YAHOO! MUSIC DATASET

2.1 Yahoo! Music Radio service

Yahoo! Music[2] was one of the first providers of personal internet music radio stations, with a database of hundreds of thousands of songs. As a pioneer in online music streaming, it influenced many subsequent services. Yahoo! Music used to be the top ranked online music site in terms of audience reach and total time spent. The service was free with some commercial advertising in between songs that could be removed by upgrading to a premium account. Users could rate songs, artists, albums and even genres on a 5 star system, or using a slider interface. These ratings were used by Yahoo! Music to generate recommendations that match the user's taste, based on either the taxonomy of items or on recommendations of other users with similar musical tastes.

2.2 Ratings dataset

The KDD-Cup contest released two datasets based on Yahoo! Music ratings. The larger dataset was created for Track1 of the contest, and a smaller dataset was created for Track2. The two datasets share similar properties, whereas the Track2 dataset omits dates and times, and refers to a smaller user population. In addition, the distinctive nature of the tasks dictates using different kinds of test sets. In this section we will elaborate on the Track 1 dataset, which is the richer and larger of the two.[3] The dataset comprises 262,810,175 ratings of 624,961 music items by 1,000,990 users collected during 1999-2010. The ratings include one-minute resolution timestamps, allowing refined temporal analysis. Each item and each user have at least 20 ratings in the whole dataset. The available ratings were split into train, validation and test sets, such that the last 6 ratings of each user were placed in the test set and the preceding 4 ratings were used in the validation set. The train set consists of all earlier ratings (at least 10). The total sizes of the train, validation and test sets are therefore 252,800,275, 4,003,960, and 6,005,940, respectively. Figure 1 depicts the weekly number of ratings and the weekly mean ratings score vs. the number of weeks that passed since the launch of the service on 1999.

The ratings are integers between 0 and 100. Figure 2 depicts the distribution of ratings in the train and validation sets using a logarithmic vertical scale. The vast majority of the ratings are multiples of ten, and only a minuscule fraction are not. This mixture reflects the fact that several interfaces ("widgets") were used to rate the items, and different users had different rating "strategies". While different widgets have different appearances, scores have always been stored internally at a common 0–100 scale. We possess only the 0–100 internal representation, and do not know the exact widget used for creating each rating. Still, the popu-

*This research was done while at Yahoo! Labs.
[1] `kddcup.yahoo.com`

[2] `new.music.yahoo.com`
[3] More details on the Track2 dataset are given in Sec. 3.

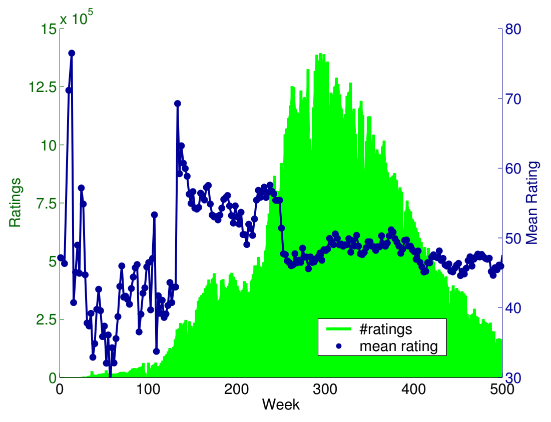

Figure 1: Yahoo! Music dataset: number of ratings and mean rating score vs. time in weeks

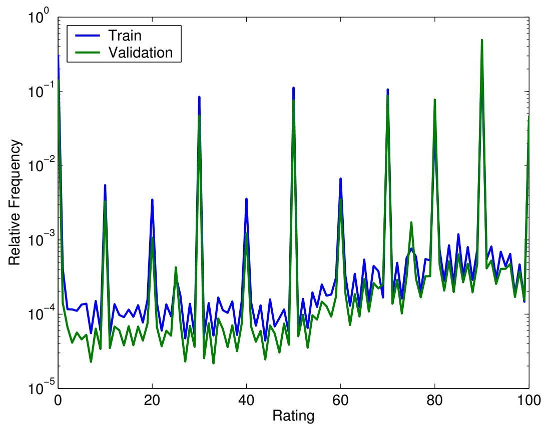

Figure 2: Ratings distribution

larity of a widget used to enter ratings at a 1-to-5 star scale is reflected by the dominance of the peaks at $0, 30, 50, 70$ and 90 into which star ratings were translated. Notice that the distributions of ratings of the train and validation sets differ significantly on lower ratings. This is caused by the fact that the validation set has exactly 4 ratings per user, thus significantly down-sampling the heavy raters who are responsible to many of the lower ratings (see also Figure 5 and related text).

An interesting aspect of the data is the fact that widgets have been altered throughout the years. Figure 3 depicts the relative frequency of each of the three types of ratings. In the first group are the ratings corresponding to the five dominant peaks of Fig. 2 ($0, 30, 50, 70$ and 90). The second group includes the remaining peaks ($10, 20, 40, 60, 80$ and 100), and the third group contains the remaining ratings (those not divisible by 10). Abrupt changes in the relative frequencies of the three groups may be clearly observed on the 125-th week as well as on the 225-th week. These dates are also associated with a dramatic change in mean rating, as can be observed in Fig. 1.

We calculated the mean rating of each user, as well as the mean rating of each item. Figure 4 depicts these two distributions. The location of the modes (at 89 and 50 respectively), as well as the variances of the two distributions are quite distinct. In addition, the distribution of the mean

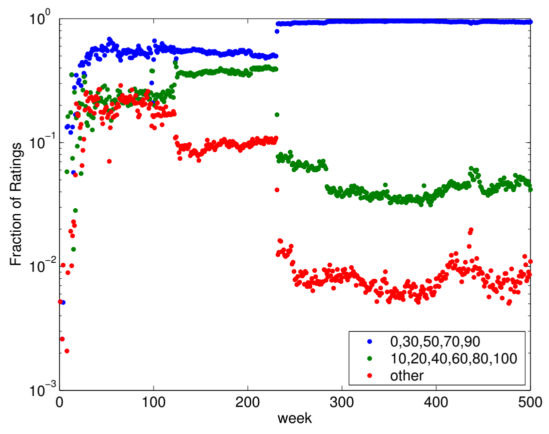

Figure 3: Relative frequency of the three groups of ratings as a function of time

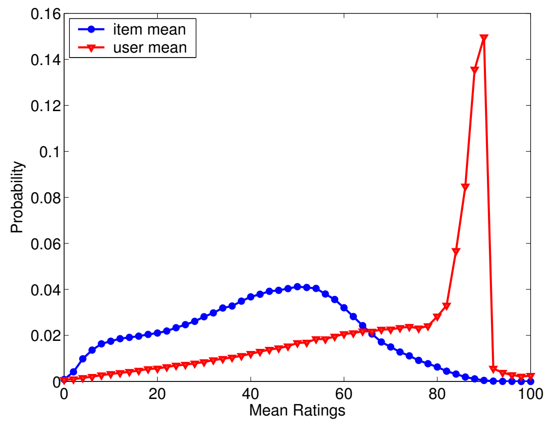

Figure 4: The distributions of item and user mean ratings

user ratings is significantly more skewed.

Different rating behavior of users accounts for the apparent difference between the distributions. It turns out that users who rate more items tend to have considerably lower mean ratings. Figure 5 substantiates this effect. Users were binned according to the number of items they rated, on a linear scale. The graph shows the median of the mean ratings in each bin, as well as the inter-quantile range in each bin plotted as a vertical line. One of the explanations for this effect is that "heavy" raters, those who explore and rate tens of thousands of items, tend to rate more items that do not match their own musical taste and preferences, and thus the rating scores tend to be lower.

A distinctive feature of this dataset is that user ratings are given to entities of four different types: *tracks*, *albums*, *artists*, and *genres*. The majority of items (81.15%) are tracks, followed by albums (14.23%), artists (4.46%) and genres (0.16%). The ratings however, are not uniformly distributed: Only 46.85% of the ratings belong to tracks, followed by 28.84% to artists, 19.01% to albums and 5.3% to genres. Moreover, these proportions are strongly dependent on the number of ratings a user has entered. Heavier raters naturally cover more of the numerous tracks, while the light raters mostly concentrate on artists; the effect is shown in Fig. 6. Thus, unlike the train set, the validation

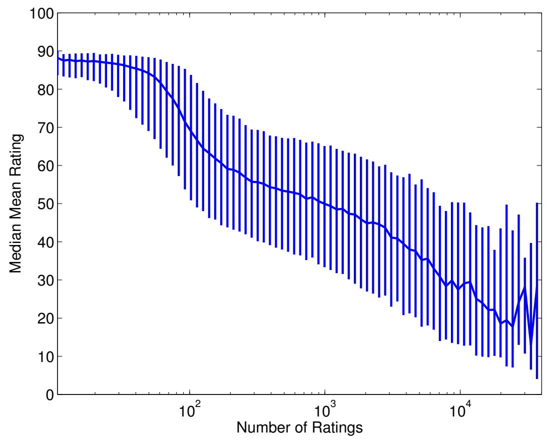

Figure 5: Median of user ratings as a function of the number of ratings issued by the user. The vertical lines represent inter-quartile range.

and test sets, which equally weight all users, are dominated by the many light-raters and dedicate most of their ratings (51.61%) to artists rather than to tracks.

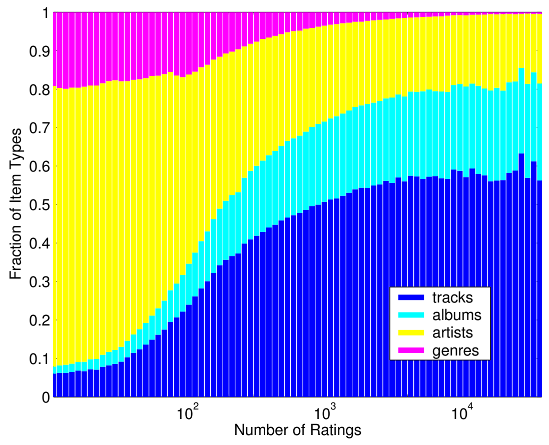

Figure 6: The fraction of ratings the four item types receive as a function of the number of ratings a user gives.

All rated items are tied together within a taxonomy. That is, for a track we know the identity of its album, performing artist and associated genres. Similarly we have artist and genre annotation for the albums. There is no genre information for artists, as artists may switch between many genres in their career. We show that this taxonomy is particularly useful, due to the large number of items and the sparseness of data per item (mostly attributed to "tracks" and "albums").

3. TASK DESCRIPTION

The competition had a two-track structure offering two different tasks. Track1 calls for predicting users' rating to musical items. Items can be tracks, albums, artists and genres. Each user and item have at least 20 ratings in the dataset (train, validation and test sets combined). A detailed description of the provided dataset was given in Sec. 2.

The main dataset statistics are described in Table 1. Each user has at least 10 ratings in the training data. Then, each user has exactly four ratings in the validation data, which come later in time than the ratings by the same user in the training data. Finally, the test data holds the last 6 ratings of each user. The contestants were asked to provide predictions for these test scores. The evaluation criterion is the Root Mean Squared Error (RMSE) between predicted ratings and true ones. Formally:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}_1|}\sum_{(u,i)\in\mathcal{T}_1}(r_{ui}-\hat{r}_{ui})^2} \qquad (1)$$

Where $(u,i) \in \mathcal{T}_1$ are all the rating pairs in Track1 test data set, r_{ui} is the actual rating of user u to item i, and $\hat{r}_{ui}$ is the predicted value for r_{ui}. This amounts to a traditional collaborative filtering rating prediction task.

Track2 involves a less conventional task. A similar music rating dataset was compiled, covering 4 times less users than Track1. Once again each user and item have at least 20 ratings in the dataset (train and test sets combined); see Table 1 for the dataset main statistics. The train set includes ratings (scores between 0 to 100) to both tracks, albums, artists and genres performed by Yahoo! Music users.

For each user participating in the Track2 test set, six items are listed. All these items must be tracks (not albums, artist or genres). Three out of these six items have never been rated by the user, whereas the other three items were rated "highly" by the user, that is, scored 80 or higher. The three items rated highly by the user were chosen randomly from the user's highly rated items, without considering rating time. The three test items not rated by the user are picked at random with probability proportional to their odds to receive "high" (80 or higher) ratings in the overall population. Note that many users do not participate in this test set at all.

The goal of such a task would be differentiating high ratings from missing ones. Participants were asked to identify exactly three highly rated items for each user included in the test. The evaluation criterion is the error rate, which is the fraction of wrong predictions.

We had several objectives in designing the second track of the contest. Initially, we wanted to add a dataset of a lower scale, in order to appeal to competitors with less capable systems, which might get discouraged by the large size of the Track1 dataset. More importantly, the Track2 dataset provided us an opportunity to experiment with a less established evaluation methodology, which may better relate to real life scenarios. The proposed metric is related to the common recommendation task of predicting the items that the user will like (rather than predicting the rating value of items). Furthermore, the task of Track2 requires extending the generalization power of the learning algorithm to the truly missing entries (items that were never rated by the user), as required in real life scenarios. Finally, the way we drew negative examples (in proportion to their dataset popularity), discourages known trivial solutions to the top-K recommendation task, where most popular items are always suggested, regardless of the user taste.

We decided to exclude timestamps (both dates and times) from the Track2 dataset. Our purpose was to center the focus of the time-limited competition on the a specific aspect of the problem. In particular, we assume that there are

Task	#users	#items	Train	Validation	Test
Track1	1,000,990	624,961	252,800,275	4,003,960	6,005,940
Track2	249,012	296,111	61,944,406	N/A	607,032

Table 1: Datasets size statistics

many strong short-term correlations between various actions of a user, which would greatly help in separating rated from unrated items. For example, some users have the tendency to listen to multiple tracks of the same album in a row. We wanted to encourage solutions catering to the longer-term user taste, rather than analyzing short term trends, thus we hid the temporal information in Track2.

At both tracks, the test set was internally split into two equal halves known as Test1 and Test2, in a way not disclosed to the competitors. Participants were allowed to submit their test set predictions every 8 hours, and receive feedback on their Test1 performance that was reflected in the public leaderboard. The true rankings of competitors were based on Test2 scores, which were not disclosed.

4. CONTEST CONDUCT

The contest took part over a period of three and a half months from March 15 till June 30, 2011. The datasets were made available to contestants on the first day of the competition. In addition, the contestants could register for the contest and have access to sample data to prepare their software two weeks in advance to the beginning of the competition.

During the competition, each team was allowed to submit one solution every eight hours for each track of the competition. The submission system provided the contestants with immediate feedback on the performance of the submitted solution on the test set. The contest website also featured a public leaderboard that showed an up-to-date ranking of the best submissions so far (based on scores on the Test1 set).

The registration system required users to form teams and to be part of only one team at a time. Teams could leave the competition by marking themselves as inactive. The contestants made heavy use of that functionality, but not to actually leave the contest. Instead, it was used mostly to regroup: Out of the 2389 users that registered for the contest, 2100 were still part of an active team at the end of the competition. During the competition, 4252 teams were registered (more than users!) of which 1287 were active at the end of the competition.

Each of the two contest tracks had about the same number of teams submitting to it: 1878 teams ever submitted to Track1, 1854 to track2. The same is not true for the number of submissions. During the competition, there were 6344 solutions submitted to Track1, while Track2 received "only" 5542. Table 2 shows a histogram of the number of submissions per track and how many teams submitted that many times. Figure 7 shows the number of submissions per day and track. As can be clearly seen from this plot, the contestants were eager to (re-)submit the best solutions on the last day of the competition.

The almost equal number of submissions and teams submitting is somewhat suprising, because the leadership of Track2 was much more volatile than that of Track1 as can be seen in Figure 8 and 9. During the competition, the top

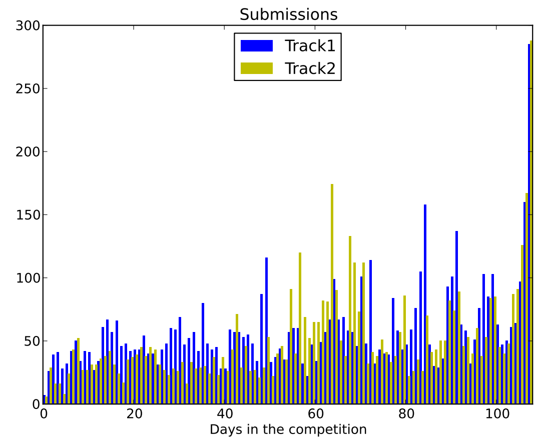

Figure 7: Submissions per day

spot on the public leaderboard changed 52 times for Track1 and almost twice as often, 96 times, for Track2. Even more drastically, only 12 teams ever held the top leaderboard spot in Track1 during the competition, while 35 teams held that position for some period of time in Track2. We believe that this greater volatility in the leadership of Track2 can be attributed to the novel task to be solved in that track of the competition.

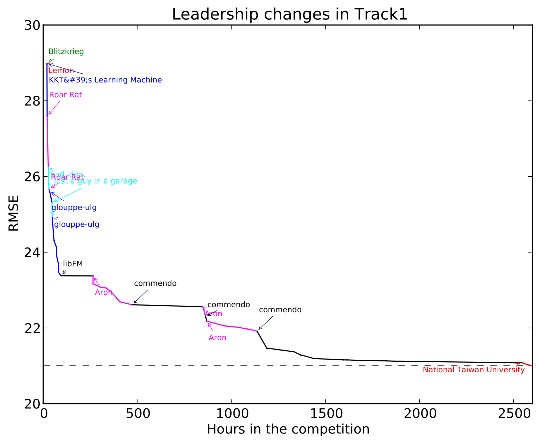

Figure 8: Leadership and RMSE over time for Track1

According to the rules of the competition, the last submission of each team is used fot determining the winners, not the best. Thus, the question arises whether all teams managed to submit their best solution in time or whether an earlier submission actually would have won. Much to the relief of the teams involved, the order of the top spots

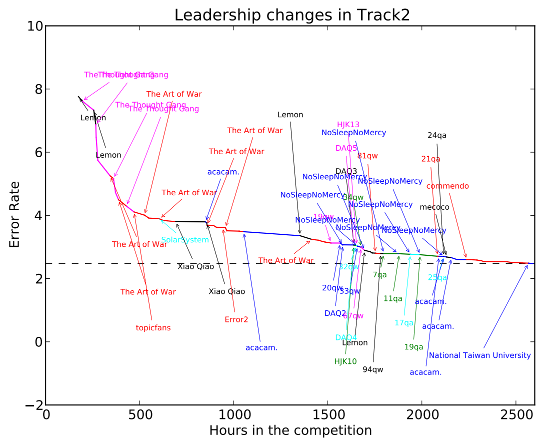

Figure 9: Leadership and error rate over time for Track2

Submissions	Teams Track1	Teams Track2
0-9	1728	1738
10-19	80	65
20-29	36	29
30-39	13	2
40-49	8	7
50-59	4	6
60-69	0	1
70-79	4	1
80-89	1	1
90-99	2	2
100-109	0	1
110-119	1	0
120-129	1	1

Table 2: Submissions statistics: Number of submissions per team

in the leaderboard would be the same if we would have used the best submission. Similarly, for the top spots the publicly reported rankings based on the Test1 set, equal the ranking based on the undisclosed Test2 set, meaning that contestants did not strongly overfit the Test1 set.

5. LESSONS

The KDD-Cup'11 workshop published 13 papers describing the techniques employed by top teams. These papers describe many interesting algorithms, enhancing different recommendation methods. We encourage the reader to look at these papers for more details. Here, we will highlight some of the higher-level lessons that we take away from the competitors' works.

The Track1 solutions followed much of the techniques that were found successful at the Netflix Prize competition [1], which aimed at the same RMSE metric. In a sense, reported Track1 results reinforce many findings on the Netflix data, which still hold despite the usage of a relatively different dataset. Solutions were created by blending multiple techniques, including nearest neighbor models, restricted Boltzmann machines and matrix factorization models. Among those, matrix factorization models proved to perform best. Modeling temporal changes in users' behavior and items' popularity was essential for improving solution quality. Among nearest neighbors models, item-item models outperformed user-user models. Such a phenomenon is notable due to the fact that the number of items in our dataset is large and roughly equals number of users, hence one could expect that in such a setup user-user and item-item techniques will deliver similar performance. The given item taxonomy helped in accounting for the large number of sparsely rated items, however its effect was relatively low in terms of RMSE reduction.

The best result achieved on Track1 has RMSE=21. This pretty much confirms our pre-contest expectations based on Netflix Prize experience. Note that Netflix Prize results need to get calibrated in order to account for the different rating scales. While in the Netflix Prize the ratings range (difference between highest (=5) and lowest (=1) rating) is 4, in our dataset the ratings range is 100. Hence, if one linearly maps our ratings to the 1–5 stars scale, RMSE of the same methods will shrink by a factor of 25. This means that on Netflix score range, the best score achieved at Track1 is equivalent to 0.84, which is strikingly close to the best score known on the Netflix dataset (0.8556). Comparing the fraction of explained variance (known as R^2), reveals more of a difference between Track1 and Netflix Prize. The total variance of the ratings in the Netflix test set is 1.276, corresponding to an RMSE of 1.1296 by a constant predictor. Three years of multi-team concentrated efforts reduced the RMSE to 0.8556, thereby leaving the unexplained ratings variance at 0.732. Hence the fraction of explained variance is $R^2 = 42.6\%$, whereas the rest 57.4% of the ratings variability is due to unmodeled effects (e.g., noise). Moving to the Yahoo! Music Track1 test set, the total variance of the test dataset is 1084.5 (reflecting the 0-100 rating scale). The best submitted solution could reduce this variance to around 441, yielding $R^2 = 59.3\%$. Hence, with the Yahoo! Music dataset a considerably larger fraction of the variance is explainable by the models.

Track2 of the competition offered a less traditional metric: error rate of a classifier separating loved tracks from unrated ones. We were somewhat surprised by the low error rate achieved by the contestants — 2.47% for best solution. This means that for over 97% of test tracks, the model could correctly dictate whether they are among the loved ones. However, such a performance should be correctly interpreted in light of real life expectations. Our chosen metric contrasted the same amount of positive ("loved items") and negative examples. In real life, one would argue that positives and negatives are not balanced, but rather negatives grossly outnumber positives. This makes the task of identifying the "top-3" items harder. For future works, we would try other ranking metrics, especially those contrasting all unrated items with the rated ones. One example would be recall@K, averaged across all users. A benefit of the metric we employed was its being neutral to popularity effects, which tend to greatly influence ranking-based metrics. As long as one wants to resist the high impact of popularity effects, alternative ranking metrics should be adjusted as well. For example, when measuring recall@K, one can weight items inversely to their popularity, such that retrieving a popular item liked by the user is less rewarding than retrieving an unpopular item liked by the user.

A key technique shared by the Track2 solutions is sampling the missing ratings. This way, the proposed methods do not only model the observed ratings, but also those ratings sampled from the missing ones. This greatly improved performance at Track2. This also confirms recent works [3, 11] showing that models that account also for the missing ratings provide significant improvements over even sophisticated models that are optimized on observed ratings only. We note that as dictated by the test set balanced structure, competitors resorted to balanced sampling. That is, the number of sampled missing ratings equals to the number of non-missing ratings for the same user. This also bodes well with practices used when learning unbalanced classification models, where often training is performed on a balanced number of positive and negative examples. Yet, we would like to explore what would have happened had the objective function been different, such as the aforementioned average recall@K metric, which is not utilizing an equal number of positive and negative examples. Would we still want to construct a balanced training set with equal number of positive and negative examples per user, or maybe we would like to use the full set of missing values as in [3, 5, 11]?

Another theme shared by many Track2 solutions is the usage of pairwise ranking objective functions. Such schemes do not strive to recover the right score for each item, but rather maintain the correct ranking for each pair of items. Pairwise ranking objective seems more natural and elegant in our ranking-oriented setup, where item ordering matters, not individual scores. A notable recommendation algorithm of this family is BPR [10], which was indeed employed by several teams. Yet, in future research we would like to further experiment with the value of pairwise ranking solutions, compared to the arguably cheaper regression-based solutions, which try to recover each single user-item score [5, 11]. In fact, the top two leading teams on Track2 [7, 8] utilized regression techniques which minimize squared loss of recovered user-item scores. Their results were as good (or even better) than teams that utilize pairwise ranking techniques.

Comparing results of Track1 and Track2 reinstates the known observation that achievable RMSE values on the test set lie in a quite compressed range. This was also evident at the Netflix contest where major modeling improvements could only slightly reduce RMSE. This observation becomes much more striking when considering comparable improvements on the error rate metric applied at Track2. Let us analyze the influence of three components that improve modeling accuracy.

1. **Dimensionality of the matrix factorization model** On the RMSE front it was observed that after reaching 50-100 factors, there is barely an improvement when increasing the number of factors; see, e.g., Figure 7 of [4]. However, the case is very different with Track2, where the error rate metric was used. Competitors have resorted to very high dimensionalities of matrix factorization models in order to improve performance. For example, team National Taiwan University reports error rate dropping from 6.56% to 4.25% when increasing dimensionality from 800-D to 3200-D (see Table 2 of [8]).

2. **Utilizing the given item taxonomy** We have observed a quite subtle RMSE reduction from around 22.85 to 22.6 by utilizing the taxonomy; see [4]. Similar RMSE reduction is also reported ain Table 3 of team InnerPeace solution [2]. When moving to the error rate metric, utilization of the taxonomy results in much more significant improvements. Team The Art of Lemon [7] reports a matrix factorization model (BinarySVD) achieving around 6% error rate, which is reduced all the way to 3.49% by considering the items taxonomy (BinarySVD+). Similarly, team The Thought Gang reports (Table 3 of [9]) reducing the error rate from 6.02% to 3.71% by accounting for taxonomy in a latent factor model.

3. **Blending multiple models** At Track1, best reported solution by an individual model yields a test RMSE of 22.12 [2]. Others could achieve RMSE=22.6 by a matrix factorization model [4]. When blending multiple predictors, the best reported RMSE on the same test set is close to 21, an improvement of just around 5%. On the Netflix dataset, blending helped even less (percentage-wise). However, on the Track2 error rate metric blending had a much stronger impact. Best individual model (BinarySVD+) yielded an error rate of 3.54% [7]. Yet, after blending, best known solution has an error rate of 2.47% – over 30% improvement thanks to blending.

In conclusion, we have found a consistent evidence that the same modeling improvements that only modestly impact RMSE, have a substantial effect on the error rate metric. Similar evidence was also hinted elsewhere [6]. Given the large popularity of the RMSE metric in recommenders research, it is important to interpret RMSE improvements well. Clearly, small RMSE changes can correspond to significantly different ranking of items. For example, RMSE minimizing algorithms tend to diminish variance by concentrating all predictions close to averages, with small deviations from the average dictating item rankings. We would like to leave here the question of whether small improvements in RMSE terms can have a significant impact on the quality perceived by real users?

6. REFERENCES

[1] J. Bennett and S. Lanning. The netflix prize. In *Proc. KDD Cup and Workshop*, 2007.

[2] T. Chen, Z. Zheng, Q. Lu, X. Jiang, Y. Chen, W. Zhang, K. Chen, Y. Yu, N. N. Liu, B. Cao, L. He, and Q. Yang. Informative ensemble of multi-resolution dynamic factorization models. In *KDD-Cup'11 Workshop*, 2011.

[3] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. 4th ACM Conference on Recommender Systems(RecSys'10)*, pages 39–46, 2010.

[4] G. Dror, N. Koenigstein, and Y. Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proc. 5th ACM Conference on Recommender Systems (RecSys'11)*, 2011.

[5] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. 8th IEEE Conference on Data Mining (ICDM'08)*, pages 263–272, 2008.

[6] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 426–434, 2008.

[7] S. Laiy, L. Xiang, R. Diao, Y. Liu, H. Gu, L. Xu, H. Li, D. Wang, K. Liu, J. Zhao, and C. Pan. Hybrid recommendation models for binary user preference prediction problem. In *KDD-Cup'11 Workshop*, 2011.

[8] T. G. McKenzie, C.-S. Ferng, Y.-N. Chen, C.-L. Li, C.-H. Tsai, K.-W. Wu, Y.-H. Chang, C.-Y. Li, W.-S. Lin, S.-H. Yu, C.-Y. Lin, P.-W. Wang, C.-M. Ni, W.-L. Su, T.-T. Kuo, C.-T. Tsai, P.-L. Chen, R.-B. Chiu, K.-C. Chou, Y.-C. Chou, C.-C. Wang, C.-H. Wu, H.-T. Lin, C.-J. Lin, and S.-D. Lin. Novel models and ensemble techniques to discriminate favorite items from unrated ones for personalized music recommendation. In *KDD-Cup'11 Workshop*, 2011.

[9] A. Mnih. Taxonomy-informed latent factor models for implicit feedback. In *KDD-Cup'11 Workshop*, 2011.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. 25th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'09)*, pages 452–461, 2009.

[11] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proc. 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 713–722, 2010.