Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy

Gideon Dror
Yahoo! Research
Haifa, Israel

Noam Koenigstein*
School of Electrical Eng.
Tel Aviv University

Yehuda Koren
Yahoo! Research
Haifa, Israel

ABSTRACT

In the past decade large scale recommendation datasets were published and extensively studied. In this work we describe a detailed analysis of a sparse, large scale dataset, specifically designed to push the envelope of recommender system models. The Yahoo! Music dataset consists of more than a million users, 600 thousand musical items and more than 250 million ratings, collected over a decade. It is characterized by three unique features: First, rated items are multi-typed, including tracks, albums, artists and genres; Second, items are arranged within a four level taxonomy, proving itself effective in coping with a severe sparsity problem that originates from the unusually large number of items (compared to, e.g., movie ratings datasets). Finally, fine resolution timestamps associated with the ratings enable a comprehensive temporal and session analysis. We further present a matrix factorization model exploiting the special characteristics of this dataset. In particular, the model incorporates a rich bias model with terms that capture information from the taxonomy of items and different temporal dynamics of music ratings. To gain additional insights of its properties, we organized the KddCup-2011 competition about this dataset. As the competition drew thousands of participants, we expect the dataset to attract considerable research activity in the future.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

recommender systems, collaborative filtering, matrix factorization, yahoo! music

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. Inspired by the success of the Netflix Prize contest [5], 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. At the time of writing, half throughout the contest, the contest has attracted thousands of actively participating teams trying to crack the unique properties of the dataset.

Noteworthy characteristics of the dataset include: First, it is of a larger scale compared to other datasets in the field. Second, it has a very large set of items (over 600K) – much larger than similar datasets, where usually only the number of users is large. This is mostly attributed to the large number of available music tracks. Third, there are four different categories of items, which are all linked together within a defined taxonomy thereby alleviating the unusually low number of ratings per item. Finally, given the recently shown importance of temporal dynamics in modeling user rating behavior [13, 26], we included in the dataset a fine resolution of rating timestamps. Such timestamps allow performing session analysis of user activities or determining the exact order in which ratings were given. Prior to the release of the dataset, we have conducted an extensive analysis and modeling of its properties. This work reports our main results and methodologies. In the following we highlight our main modeling contributions.

Music consumption is biased towards a few popular artists and so is the rating data. Therefore, collaborative filtering (CF) techniques suffer from a cold-start problem in many of the less popular artists in the "long tail" [7]. The problem becomes even more severe when considering individual tracks and albums. We tackle this item sparsity issue with a novel usage of music taxonomy information. Accordingly, we describe a method for sharing information across different items of the same taxonomy, which mitigates the problem of predicting items with insufficient rating data.

Our model, which is based on matrix factorization, incorporates temporal analysis of user ratings, and item popularity trends. We show the significance of a temporal analysis of user behavior—

*This research was done while at Yahoo! Research.

[1] kddcup.yahoo.com

RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.

within a refined session-based resolution—in improving the model predictive accuracy. In particular, we show how to perform such an analysis in a computationally friendly framework.

Last but not least, we have invested much efforts in uncovering biases, which is based on our experience that a CF model would significantly benefit from accounting for user and item biases. Indeed empirical evidence shows that biases explain a significant part of the observed rating behavior. Hence, we provide detailed parameterizations of biases combining conventional user and item biases with the readily available taxonomy and temporal information.

2. RELATED WORK

There are several approaches to music recommendations [6, 22]:

- *Collaborative Filtering (CF)* methods utilize user feedback (explicit or implicit) to infer relations between users, between items, and ultimately relate users to items they like. Unlike the other approaches they are content agnostic and do not use domain knowledge.
- *Signal Filtering* methods (also known as "content filtering") analyze the audio content for characterizing tracks and establishing item-to-item similarities [2]. For example, Mel-Frequency Cepstral Coefficients (MFCCs) are often used to generate feature vectors, which can be used to find other items with acoustic resemblance [3, 18].
- *Context Based Filtering* methods (or, "attribute-based filtering") characterize items based on textual attributes, cultural information, social tags and other kinds of web-based annotations [16, 19, 23].
- *Human Annotation* methods are based on time consuming annotation of music data such as in Pandora[2]. Human annotation methods are mostly based on the acoustic similarities, but may also include social context.

Given the nature of our dataset, we focus on a CF approach. In general, CF has proved more accurate in predicting rating datasets devoid of intrinsic item information such as the Netflix dataset [4]. However, algorithms based on collaborative filtering typically suffer from the cold-start problem when encountering items with little rating information [24]. Several hybrid approaches are suggested for merging content filtering and CF; see, e.g. some of the more recent approaches [1, 8, 9]. They allow relying on item attributes when rating information is not sufficient, while still enjoying the improved accuracy CF offers as more ratings are gathered. In our system, we use a taxonomy to share information between rated items. For example, the representation of tracks with a little rating data naturally collapses to the representation of their respective album and artist as described in Sec. 5.3 and 6.1.

We employ a Matrix Factorization (MF) model, which maps items and users into comparable latent factors. Such techniques became a popular choice for implementing CF and a survey can be found at [14]. Typically, in MF a user is modeled by a factor vector $p_u \in \mathbb{R}^d$, and an item is modeled by a factor vector $q_i \in \mathbb{R}^d$. A predicted rating $\hat{r}_{ui}$ by user u to item i is given by

$$\hat{r}_{ui} = \mu + b_i + b_u + p_u^T q_i \quad (1)$$

where μ is the average rating, and b_u and b_i are the user and item biases respectively. The term $p_u^T q_i$ captures the affinity of user u to item i. Sec. 5 and 6 discuss these components in depth.

User preferences and item popularity tend to drift over time. Thus, a few recent works [13, 26] highlighted the significance of a delicate modeling of temporal dynamics when devising collaborative filtering models. Our approach is related to [13]. The two notable differences are: (1) This work applies more refined session analysis rather than working at a coarser day resolution. (2) We employ a much more memory efficient method for modeling the way user factor vectors drift over time.

We have also benefited from techniques suggested by Piotte and Chabbert [21] in their Netflix Prize solution. First, we adopted their method of using Nelder Mead optimization [20] for automatically setting meta-parameters, and have taken it a step further by using a parallel implementation; see Sec. 7.1. Second, we have used their idea of modeling smooth temporal dynamics by learning to combine several basis functions.

[2] www.pandora.com

Train	Validation	Test
252,800,275	4,003,960	6,005,940

Table 1: Train, validation and test set sizes

3. THE YAHOO! MUSIC DATASET

Yahoo! Music[3] offers a wealth of information and services related to many aspects of music. We have compiled a dataset of user ratings of music items collected during a decade of using the Yahoo! Music website. The dataset was released within the first track of the KDD Cup 2011 contest. It comprises of 262,810,175 ratings of 624,961 items by 1,000,990 users. The ratings include both date and one-minute resolution timestamps, allowing refined temporal analysis. Each item and each user has 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. All earlier ratings (at least 10) comprise the train set. Table 1 details the total number of ratings in the train, validation and test sets.

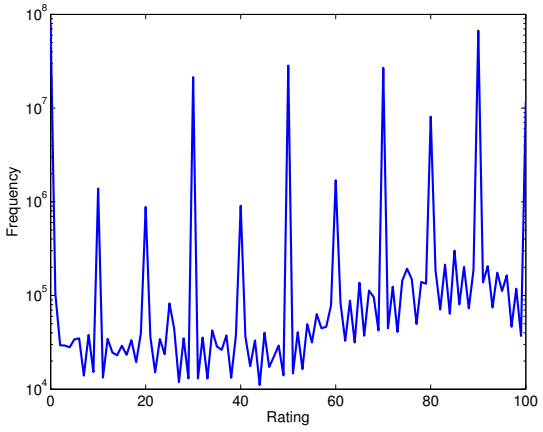

Figure 1: The distribution of ratings. The approximately discrete nature of the distribution is evident

The ratings are integers between 0 and 100. Figure 1 depicts the distribution of ratings in the train set 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, some of them changing throughout the years while allowing different usage options. 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 popularity of a widget used to enter ratings at a 1-to-5 star scale is reflected by

[3] new.music.yahoo.com

the dominance of the peaks at $0, 30, 50, 70$ and 90 into which star ratings were translated.

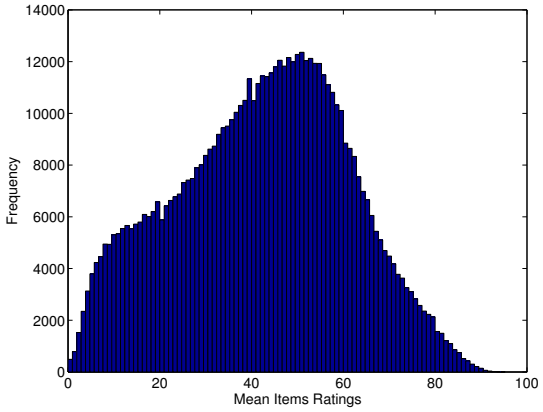

Figure 2: The distribution of item mean ratings

Not surprisingly, most items have intermediate mean rating and only few items have a mean rating on extreme high or low ends of the scale. Indeed, the distribution of item mean ratings follow a unimodal distribution, with a mode at 50 as shown in Fig. 2.

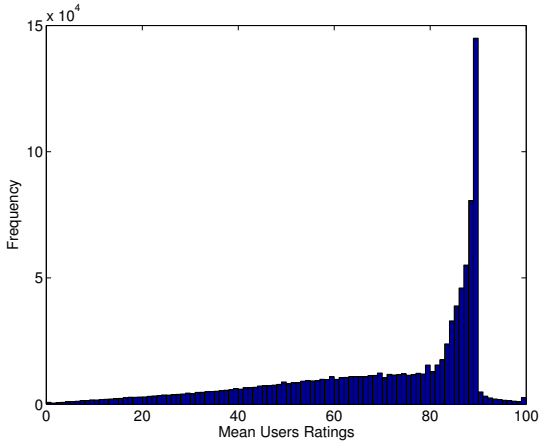

Figure 3: The distribution of user mean ratings

Interestingly, the distribution of item mean ratings presented in Fig. 2 is very different from the distribution of mean ratings of users, depicted in Fig. 3: the distribution is now strongly skewed, with mode shifted to a mean rating of 89. 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. Fig. 4 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 among the tens of thousands of items rated by the "heavy" raters, the majority do not match their taste.

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

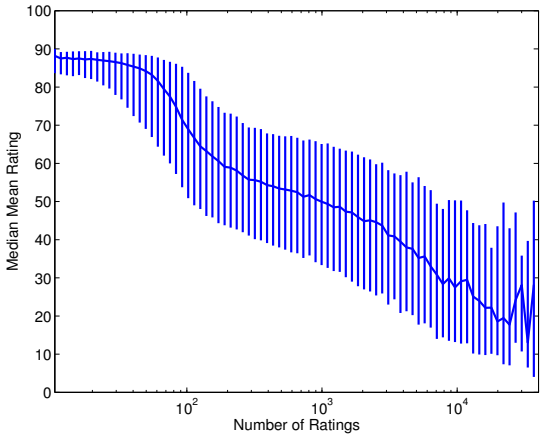

Figure 4: Median of user ratings as a function of the number of ratings issued by the user. The vertical lines represent inter-quartile range.

naturally cover more of the numerous tracks, while the light raters mostly concentrate on artists; the effect is shown in Fig. 5. Thus, the validation and test sets, which equally weight all users, are dominated by the many light-raters and dedicate most of their ratings to artists rather than to tracks; see Table 3.

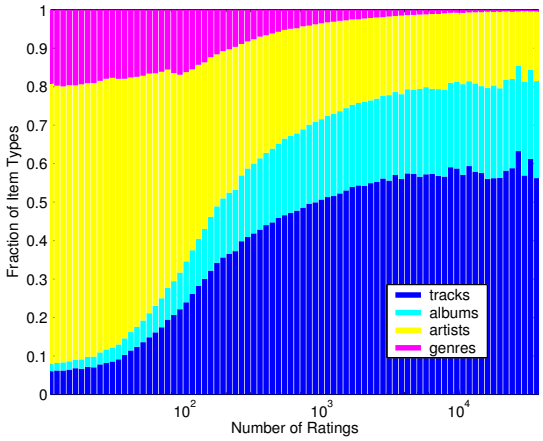

Figure 5: The fraction of ratings the four item types receive as a function of the number of ratings a user gives.

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").

4. NOTATION

We reserve special indexing letters for distinguishing users from items: for users u, and for items i. A rating r_{ui} indicates the rating given by user u to item i. We distinguish predicted ratings from known ones using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}.

For tracks, we denote by $album(i)$ and $artist(i)$ the album and the artist of track i respectively. Similarly, for albums, we denote by $artist(i)$ the artist of album i. Tracks and albums in the Yahoo!

Music dataset may belong to one or more genres. We denote by $genres(i)$ the set of genres of item i. Lastly, we denote by $type(i)$ the type of item i, with $type(i) \in \{track, album, artist, genre\}$.

5. BIAS MODELING

5.1 Why biases?

When considering their vast explanation power, biases are among the most overlooked components of recommender models. In the context of rating systems, biases model the portion of the observed signal that is derived either solely by the rating user or solely by rated item, but not by their interaction. For example, user bias may model a user's tendency to rate on a higher or lower scale than the average rater, while the item bias may capture the extent of the item popularity. A general framework for capturing the bias of the rating by user u to item i is described as

$$b_{ui} = \mu + B_i + B_u \tag{2}$$

where μ is the overall mean rating value (a constant), and B_i and B_u stand for item and user biases, respectively.

Since components of the user bias are independent of the item being rated, while components in the item bias are independent of any user, they do not take part in modeling personalization, e.g., modeling user musical taste. After all, ordering items by using only a bias model (2) necessarily produces the same ranking for all users, hence personalization—the cornerstone of recommendation systems—is not achieved at all.

Lack of personalization power should not be confused with lack of importance for biases. There is plenty of evidence that much of the observed variability in rating signal should be attributed to biases. Hence, properly modeling biases would effectively amount to cleaning the data from signals unrelated to personalization purposes. This will allow the personalization part of the model (e.g., matrix factorization), where users and items do interact, to be applied to a signal more purely relevant for personalization. Perhaps the best evidence is the heavily analyzed Netflix Prize dataset [5]. The total variance of the ratings in this dataset is 1.276, corresponding to a Root Mean Squared Error (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 (known as R^2) is 42.6%, whereas the rest 57.4% of the ratings variability is due to unmodeled effects (e.g., noise). Now, let us analyze how much of the explained variance should be attributed to biases, unrelated to personalization. The best published pure bias model [12] yields an RMSE=0.9278, which is equivalent to reducing the variance to 0.861 thereby explaining 32.5% of the observed variance. This (quite surprisingly) means that the vast majority of the 42.6% explainable variance in the Netflix dataset, should be attributed to user and item biases having nothing to do with personalization. Only about 10% of the observed rating variance comes from effects genuinely related to personalization. In fact, as we will see later (Sec. 7), our experience with the music dataset similarly indicates the importance role biases play. Here the total variance of the test dataset is 1084.5 (reflecting the 0-100 rating scale). Our best model could reduce this variance to around 510.3 ($R^2 = 52.9\%$). Out of this 52.9% explained variance[4], once again the vast majority (41.4%) is attributed to pure biases, leaving about 11.5% to be explained by personalization effects. Hence, the big importance one should put on well modeling biases.

In this section, we present a rich model for both the item and user biases, which accounts for the item taxonomy, user rating sessions, and items' temporal dynamics. The model adheres to the framework of Eq. (2). In the following we will gradually develop its B_i and B_u components. The predictive performance of the various components of the bias model is discussed in Sec. 7.

5.2 A basic bias model

The most basic bias model captures the main effects associated with users and items [11]. Following bias template (2), we set the item bias B_i as a distinct parameter associated with each item denoted by b_i, and similarly the user bias B_u as a user-specific parameter b_u. This gives rise to the model

$$b_{ui} = \mu + b_i + b_u \tag{3}$$

5.3 Taxonomy biases

We start enhancing our bias model by letting item biases share components for items linked by the taxonomy. For example, tracks in a good album may all be rated somewhat higher than the average, or a popular artist may have all her songs rated a bit higher than the average. We therefore add shared bias parameters to different items with a common ancestor in the taxonomy hierarchy. We expand the item bias model for tracks as follows

$$B_i^{(0)} = b_i + b_{album(i)} + b_{artist(i)} + \frac{1}{|genres(i)|} \sum_{g \in genres(i)} b_g \tag{4}$$

Here, the total bias associated with a track i sums both its own specific bias modifier (b_i), together with the bias associated with its album ($album(i)$) and its artist ($artist(i)$), and the mean bias associated with its genres($\frac{1}{|genres|} \sum_{g \in G} b_g$).

Similarly for each album we expand the bias model as follows

$$B_i^{(0)} = b_i + b_{artist(i)} + \frac{1}{|genres(i)|} \sum_{g \in genres(i)} b_g \tag{5}$$

One could view these extensions as a gradual accumulation of the biases. For example, when modeling the bias of album i, the start point is $b_{artist(i)} + \frac{1}{|genres(i)|} \sum_{g \in genres(i)} b_g$, and then b_i adds a residual correction on top of this start point. Similarly, when i is a track another item-specific correction is added on top of the above. As bias estimates for tracks and albums are less reliable, such a gradual estimation allows basing them on more robust initial values.

Note that such a framework not only relates items to their taxonomy ancestors, but (indirectly) also to other related items in the taxonomy (e.g., a track will get related to all other tracks in its album, and to lesser extent to all other tracks by the same artist).

Also, note that while artists and genres are less susceptible to the sparsity problem, they also benefit from this model as any rating to track and album also influences the biases of their corresponding artist and genre.

The taxonomy of items is also useful for expanding the user bias model. For example, a user may tend to rate artists or genres higher than songs. Therefore, given an item i the user bias is

$$B_u^{(0)} = b_u + b_{u,type(i)} \tag{6}$$

where b_u is the user specific bias component and $b_{u,type(i)}$ is a shared component of all the ratings by user u to items of type $type(i)$.

[4]Unlike the Netflix dataset case in which tremendous efforts were invested, here we can safely assume that eventually the explained variance will exceed 52.9% by subsequent works and by blending multiple predictors.

5.4 User sessions

A distinctive property of the Yahoo! Music dataset is its temporal information. Each rating is marked by a date and a timestamp with resolution down to minutes. We used this information for modeling temporal dynamics of both items and users. We start by modeling user sessions. Unlike movies, in music it is common for users to listen to many songs and rate them one after the other. A rating session is therefore a set of consecutive ratings without an extended time gap between them. There are many psychological phenomena that affect ratings grouped in a single session. These effects are captured by user session biases.

One example is the fact that the order in which the songs were listened by the user might determine the ratings scores, a phenomenon known as the drifting effect [10]. Users tend to rate items in the context of previous items they rated. If the first song a user hears is particularly good, the following items are likely to be rated by that user lower than the first song. Similarly, if the user did not enjoy the first song, the ratings of subsequent songs may shift upwards. The first song therefore may serve as a reference rating to all the following ratings. However, with no absolute reference for the first rating, the user sometimes find it hard to rate, and some users tend to give it a default rating (e.g., 70 or 50). Consequently, all the following ratings in that same session may be biased higher or lower according to the first rating. Another source for session biases is the mood of the user. A user may be in a good/bad mood that may affect her ratings within a particular session. It is also common to listen to similar songs in the same session, and thus their ratings become similar.

To take such effects into account, we added a session bias term to our user bias model. We thus marked users' consecutive ratings with session numbers separated by a time gap of at least 5 hours in which the user was idle (no rating activity). We denote by $session(u,i)$ the rating session of the rating r_{ui}, and expand our user bias model to include session biases

$$B_u^{(1)} = B_u^{(0)} + b_{u,session(i,u)} \tag{7}$$

The session bias parameter $b_{u,session(i,u)}$ models the bias component common to all ratings of u in the same session he rated i.

5.5 Items temporal dynamics

The popularity of songs may change dramatically over time. While users' temporal dynamics seem to follow abrupt changes across sessions, items' temporal dynamics are much smoother and slower, thus calling for a different modeling approach.

Given item i and the time t since i's first rating, we define a time dependent item bias as a linear combination of n temporal basis functions $f(t) = (f_1(t), f_2(t), ...f_n(t))^T$ and expand the item bias component to be

$$B_i^{(1)} = B_i^{(0)} + c_i^T f(t) \tag{8}$$

where $c_i \in \mathbb{R}^n$ is an item specific vector of coefficients.

Both $f(t)$ and c_i are learned from data using the standard RMSE-minimizing stochastic gradient descent (SGD), which is also used for learning the other model components. In practice, a 2-week coarse time resolution is sufficient for the rather slow changing item temporal dynamics, therefore the basis functions are only estimated at a small number of points and can be easily learned. This process does not guarantee an orthogonal or normalized basis, however it finds a basis that fits the patterns seen in the dataset.

We have found that a basis of 4 functions is sufficient to represent the temporal dynamics of item biases in our dataset. Figure 6 depicts the learned basis functions $\{f_i(t)\}_{i=1}^4$. Since the coefficients of the basis function can be be either positive or negative, it is hard to give a clear interpretation to any specific basis function. However, an interesting observation is that basis functions seem to have high gradients right after an item was released, indicating more dynamic temporal effects in this time period. It is also interesting to note that after a long time period (above 360 weeks), the temporal basis functions converge into relatively steady values. This indicates that at a longer perspective, items seem to have either a positive or a negative bias, with much less temporal dynamics.

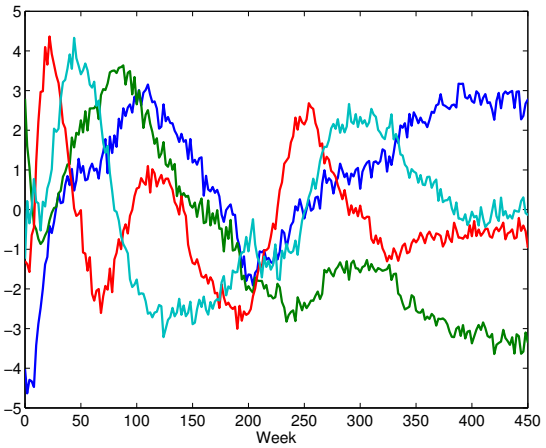

Figure 6: Items temporal basis functions $\{f_i(t)\}_{i=1}^4$ vs. time since an item's first rating measured in weeks

5.6 Full bias model

To summarize, our complete bias model, including both enhanced user and item biases is (for a track i)

$$\begin{aligned} b_{ui} =& \mu + b_{u,type(i)} + b_{u,session(i,u)} + b_i + b_{album(i)} \\ &+ b_{artist(i)} + \frac{1}{|genres(i)|} \sum_{g \in genres(i)} b_g + c_i^T f(t_{ui}) \end{aligned} \tag{9}$$

where t_{ui} is the time elapsed from i's first rating till u's rating of i.

Learning the biases is performed by SGD together with the other model components as described in the next section. The extended bias model dramatically reduced the RMSE even before any personalization components were added into the model (see results in Sec. 7). Biases were able to absorb much of the effects irrelevant to personalization. Such a "cleaning" proved to be a key for accurately modeling personalization in later stages.

6. PERSONALIZATION MODEL

Our initial personalization model is based on a classical matrix factorization approach. Each user u is associated with a user-factor vector $p_u \in \mathbb{R}^d$, and each item i with an item-factor vector $q_i \in \mathbb{R}^d$. Predictions are done using the rule

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i \tag{10}$$

where b_{ui} is the bias model (9), and $p_u^T q_i$ is the personalization model which captures user's u affinity to item i. In this section, we expand this basic personalization model to encompass more patterns observed in the data.

6.1 Taxonomy in personalization

Musical artists often have a distinct style that can be recognized in all their songs. Similarly, artists style can be recognized across different albums of the same artist. Therefore, we introduce shared factor components to reflect the affinity of items linked by the taxonomy. Specifically, for each artist and album i, we employ a factor

vector $v_i \in \mathbb{R}^d$ (in addition to also using the aforementioned q_i). We expand our item representation for tracks to explicitly tie tracks linked by the taxonomy

$$\tilde{q}_i \stackrel{\text{def}}{=} q_i + v_{album(i)} + v_{artist(i)} \tag{11}$$

Therefore, q_i represents the difference of a specific track from the common representation of all other related tracks, which is especially beneficial when dealing with less popular items.

Similarly, we expand our item representation for albums to be

$$\tilde{q}_i \stackrel{\text{def}}{=} q_i + v_{artist(i)} \tag{12}$$

One may also add shared factor parameters for tracks and albums sharing the same genre, similarly to the way genres were exploited for enhancing biases. However, our experiments did not show an RMSE improvement by incorporating shared genre information. This indicates that after exploiting the shared information in albums and artists, the remaining information shared by common items of the same genre is limited.

6.2 Session specific user factors

As discussed earlier, much of the observed changes in user behavior are local to a session and unrelated to longer term trends. Thus, after obtaining a fully trained model (hereinafter, "Phase I") we perform a second phase of training, which isolates rating components attributed to session-limited phenomena. In this second phase, when we reach each user session, we try to absorb any session specific signal in separated component of the user factor. To this end we expand the user representation into

$$\tilde{p}_u = p_u + p_{u,session} \tag{13}$$

where the user representation $\tilde{p}_u$ consists of both the original user factor p_u and and the session factor vector $p_{u,session}$. We learn $p_{u,session}$ by fixing all other parameters and making a few (e.g., 3) SGD iterations only on the ratings given in the current session in order to learn $p_{u,session}$. After these iterations, we are able to absorb much of the temporary per-session user behavior into $p_{u,session}$, which is not explained by the model learned in Phase I. We then move to a final relaxation step, where we run one more iteration over all ratings in the same session, now allowing all other model parameters to change and shed away any per session specific characteristics. Since $p_{u,session}$ already captures much of the per-session effects of the user factor, the other model parameters adjust themselves accordingly and capture possible small changes since the previous rating session. After this relaxation step, we reset $p_{u,session}$ to zero, and move on to the next session, repeating the above process.

Our approach is related to [13], which has also employed day specific factor vectors for each user. However, there are two notable differences. First, we apply a more refined session analysis rather than working at a coarser day resolution. Second, we employ a much more memory efficient method: since we discard the per session components $p_{u,session}$ after iterating through each session, there is no need to store session vectors for every session in the dataset. At the time of prediction, we only use the last session vector. We therefore avoid the high memory consumption that occurs in previous approaches. For example, our dataset consists of 13,844,810 ratings sessions (for all users). Using a 100-D factorization model with single precision floating point numbers (4 bytes), it would have taken more than 5.5GB of memory to store all the user session factors, significantly larger than the 400MB required to store only a single session factor for each user.

6.3 Learning the model

Our final prediction model takes the following form

$$\hat{r}_{ui} = b_{ui} + \tilde{p}_u^T \tilde{q}_i \tag{14}$$

where b_{ui} is the detailed bias model as in (9), $\tilde{q}_i$ is our enhanced item factor representation as described in (11) and (12), and $\tilde{p}_u$ is defined in (13).

As previously alluded, learning proceeds by stochastic gradient descent (SGD), where all learned parameters are L2-regularized. SGD visits the training examples one-by-one, and for each example updates its corresponding model parameters. More specifically, for training example (u, i), SGD lowers the squared prediction error $e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2$ by updating each individual parameter θ by

$$\Delta\theta = -\eta \frac{\partial e_{ui}^2}{\partial \theta} - \lambda\theta = 2\eta e_{ui} \frac{\partial \hat{r}_{ui}}{\partial \theta} - \lambda\theta \tag{15}$$

here η is the learning rate and λ is the regularization rate.

Our dataset spans over a very long time period (a decade). In such a long period musical taste of users slowly drifts. We therefore expect model parameters to change with time. We exploit the fact that the SGD optimization procedure gradually updates the model parameters while visiting training examples one by one. It is a common practice in online learning to order training examples by their time, so when the model training is complete, the learned parameters reflect the latest time point, which is most relevant to the test period. Since we perform a batch learning including several sweeps through the dataset, we need to enhance this simple technique.

We loop through the data in a cyclic manner: we visit user-by-user, whereas for each user first we *sweep forward* from the earliest rating to the latest one, and then (after also visiting all other users) we *sweep backward* from the latest rating to the earliest one, and so on. This way, we avoid the otherwise discontinuous jump from the latest rating to the first one when starting a new iteration. This allows parameters to slowly drift with time as the user changes her taste. The process always terminates with a forward iteration ending at the latest rating.

7. EVALUATION

We learned our model on the train dataset using a stochastic gradient descent algorithm with 20 iterations. We used the validation dataset for early termination and for setting meta-parameters; see Sec. 7.1. We then tested the results in terms of RMSE as described in Sec. 7.2.

7.1 Optimizing with Nelder-Mead

For each type of learned parameter we set a distinct learning rate (aka, step size) and regularization rate (aka, weight decay). This grants us the flexibility to tune learning rates such that, e.g., parameters that appear more often in a model are learned more slowly (and thus more accurately). Similarly, the various regularization coefficients allow assuming different scales for different types of parameters.

We have used the validation dataset to find proper values for these meta parameters. Optimization of meta-parameters is a costly procedure, since we know very little on the behavior of the objective function, and because every evaluation requires running the SGD algorithm on the entire dataset. The fact that we have multiple learning and regularization parameters further complicates the matter. For optimizing more than 20 meta-parameters we resorted to the Nelder-Mead simplex search algorithm [20]. Though not guaranteed to converge to the global minimum [15], Nelder-Mead search is a widely used algorithm with excellent results on real world scenarios [21, 25]. To speed up the search we implemented

#	Model Name	RMSE
1	Mean Score	38.0617
2	Items and Users Bias	26.8561
3	Taxonomy Bias	26.2553
4	User Sessions Bias	25.3901
5	Items Temporal Dynamics Bias	25.2095
6	MF	22.9533
7	Taxonomy	22.7906
8	Final	22.5918

Table 2: Root Mean Squared Error (RMSE) of the evolving model. RMSE reduces while adding model components.

	Track	Album	Artist	Genre
%Test	28.7%	11.01%	51.61%	8.68%
MF	27.1668	24.5203	20.9815	15.7887
Taxonomy	26.8899	24.3531	20.8766	15.7965
Final	26.85	24.1854	20.566	15.4801

Table 3: RMSE per item type for the three personalized models. We also report the fraction of each item type in the test dataset.

a parallel version of the algorithm [17]. We consider such an automated meta-parameters optimization process as key ingredient in enabling the development of a rich and flexible model.

7.2 Experimental results

After the parameters optimization step of Sec. 7.1, we fixed the meta-parameters and re-built our final model using both the train and validation sets. We then report our results on the test set. We measured the RMSE of our predictions as we gradually add components to the bias models, and then as we gradually add components to the personalization model. This approach allows isolating the contribution of each component in the model. The results are presented in Table 2.

The most basic model is a constant predictor. In the case of the RMSE cost function, the optimal constant predictor would be the mean train rating, $\hat{r}_{ui} = \mu$; see row 1 of the table. In row 2 we present the basic bias model $\hat{r}_{ui} = \mu + b_i + b_u$ (3). In row 3 we report the results after expanding the item and user biases to include also taxonomy terms, which mitigate data sparseness by capturing relations between items of the same taxonomy; see Sec. 5.3. We then added the user session bias of Sec. 5.4. This gave a significant reduction in terms of RMSE as reported in row 4. We believe that modeling session biases in users' ratings is key in explaining ratings behavior in domains like music in which users evaluate and rate multiple items at short time frames. In row 5 we add the item temporal bias from Sec. 5.5. This term captures changes in item biases that occur over the lifespan of items since their first ratings. This bias is especially useful in domains in which item popularity easily changes over time such as in music, or datasets in which the ratings history is long. The result in row 5 reflects the RMSE of our final bias model (defined in Sec. 5.6), when no personalization is yet in place.

We move on to personalized models, which utilize a matrix factorization component of dimension 50. The model of (10) yields RMSE of 22.9235 (row 6). By adding taxonomy terms to the item factors, we were able to reduce this result to 22.8254 (row 7). Finally, in row 8 we report the full prediction model including user session factors (as in Sec. 6.2). The relatively large drop in RMSE, even when the model is already fully developed, highlights the significance of temporal dynamics at the user factor level. The result given in row 8 puts our model in par with the leading models in the KDD Cup 2011 competition (as of the time of writing), which

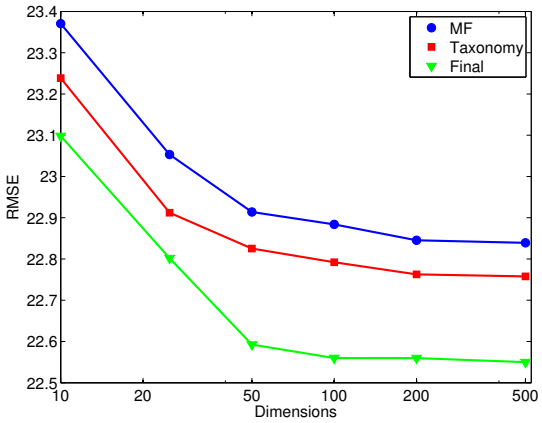

Figure 7: RMSE vs. dimensionality of factors (d). We track regular MF, MF enhanced by taxonomy and the final model.

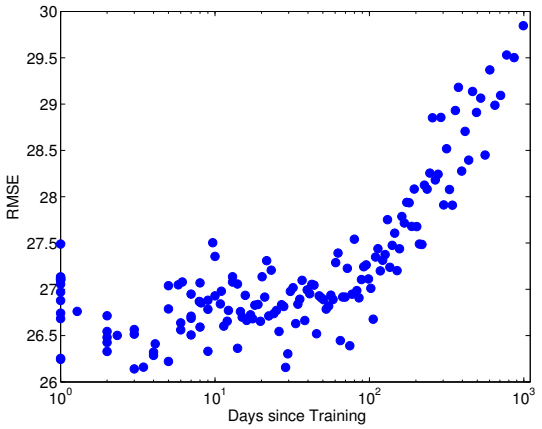

Figure 8: RMSE vs. number of days elapsed since last user's training example

is notable considering the fact that we are not blending multiple models, neither impute information taken from the test set.

Let us consider the effect of the taxonomy on the RMSE results of each item type. Table 3 breaks down the RMSE results per item type of the three personalized models. It is clear that incorporating the taxonomy is most helpful for the sparsely rated tracks and albums. It is much less helpful for artists, and becomes counterproductive for the densely rated genres.

We further investigate the performance of our model as more factors are added. Figure 7 depicts the RMSE vs. factors dimensionality. Since we carefully tune the regularization terms, there is no overfitting even as the dimensionality reaches 500. However, there are clearly diminishing returns of increasing dimensionality, with almost no improvement over 100 dimensions. Note that the reduction in RMSE given by the taxonomy and session factors remains steady even as the dimensionality increases.

Lastly, we investigate the relation of the test RMSE to the time distance from the train set. Figure 8 depicts the mean RMSE and the elapsed time for each bin. Ratings with a zero elapsed time, which mostly correspond to user sessions artificially split between train and test set, were excluded from this analysis, as they are not relevant for any real recommender system. The plateau on the left part of the figure suggests that the performance of the model is stable for about three months since the time it was trained, whereupon the RMSE of its predictions gradually increases. Thus the model needs updating only once in a month or so in order to exhibit uniform performance.

8. DISCUSSION

In this work, we introduced a large scale music rating dataset that is likely to be the largest of its kind. We believe that data availability is a key in enabling the progress of Web science, as demonstrated by the impact of the Netflix Prize dataset release. In the same spirit, we decided to share a large industrial dataset with the public, and to increase its impact and reach by including it in the KddCup-2011 contest.

While releasing real commercial data to the research world is critical to facilitating scientific progress, this process is far from trivial and definitely not risk free. On the one hand privacy advocates tend to push for over-sanitizing the data to the extent of putting an intermediary between the public and the dataset. The 2006 privacy crisis related to the AOL query data release and the more recent claims concerning privacy of the Netflix data clearly demonstrate this point. On the other hand, scientists are eager to receive the data in a form as complete as possible to facilitate unrestricted analysis. This has put us through a real dilemma, having to justify the release despite what seems to be a no win situation.

After conducting a thorough due diligence, we opted to release a sampled dataset, where both users and items are fully anonymized, which in our opinion maintains a good balance between privacy needs and scientific progress. The resulting dataset favors the application of collaborative filtering (CF) methods, which can excel without relying on item identities or attributes. We aimed at structuring the data in a way that offers a potential to sharpen current CF methods, by posing new scientific challenges and methodologies not offered by most prior datasets. The developments permitted by using the dataset, such as those discussed in this paper, are likely to be applicable at other setups, not necessarily music-related, reaping the benefits of using the domain-free CF approach.

Prior to releasing a dataset it is essential to go through data organization and sanitization. Consequently, we conducted an intensive analysis of the dataset to ensure a successful public release. The efforts reported in this work are based on our pre-release extensive analyzing and modeling efforts. We formulated a detailed collaborative filtering model, specifically designed to account for the dataset properties. The underlying design process can be valuable in many other recommendation setups. The process is based on gradual modeling of additive components of the model, each trying to reflect a unique characteristic of the data. Within this process, a major lesson is the need to dedicate significant efforts to estimating and modeling biases in the data, which tend to capture much of the observed data variability. Analyzing the effect of each component on the performance of the model supports this approach.

Finally, we are encouraged by the large number of teams (over 1000) currently analyzing the dataset within the KddCup-2011 contest, and look forward to observing new progress and getting new insights from the contest community.

9. ACKNOWLEDGEMENTS

The authors would like to thank Prof. Yuval Shavitt for his support.

10. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[2] X. Amatriain, J. Bonada, Àlex Loscos, J. L. Arcos, and V. Verfaille. Content-based transformations. *Journal of New Music Research*, 32:2003, 2003.

[3] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proc. 3rd International Symposium on Music Information Retrieval*, pages 157–163, 2002.

[4] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9:75–79, 2007.

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

[6] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, 2008.

[7] O. Celma and P. Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In *2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, 2008.

[8] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, pages 176–185, 2010.

[9] A. Gunawardana and C. Meek. Tied boltzmann machines for cold start recommendations. In *RecSys*, pages 19–26, 2008.

[10] M. Kendall and K. D. Gibbons. *Rank Correlation Methods*. Oxford University Press, 1990.

[11] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[12] Y. Koren. The bellkor solution to the netflix grand prize. 2009.

[13] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

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

[15] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex algorithm in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1996.

[16] P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[17] D. Lee and M. Wiswall. A parallel implementation of the simplex function minimization routine. *Comput. Econ.*, 30:171–187, 2007.

[18] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Int. Symposium on Music Information Retrieval*, 2000.

[19] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Trans. on Audio, Speech and Language Processing*, 18(2):407–412, 2010.

[20] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4), 1965.

[21] M. Piotte and M. Chabbert. The pragmatic theory solution to the netflix grand prize. 2009.

[22] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[23] M. Schedl and P. Knees. Context-based Music Similarity Estimation. In *Proc. 3rd International Workshop on Learning the Semantics of Audio Signals (LSAS 2009)*, 2009.

[24] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM Press, 2002.

[25] M. Wright. Direct search methods: Once scorned, now respectable. In D. Griffiths and G. Watson, editors, *Numerical Analysis*, pages 191–208. Addison Wesley, 1995.

[26] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *KDD*, pages 723–732, 2010.