Million Song Dataset Recommendation Project Report

Yi Li
Cornell University
yl2326@cornell.edu

Rudhir Gupta
Cornell University
rg495@cornell.edu

Yoshiyuki Nagasaki
Cornell University
yn253@cornell.edu

Tianhe Zhang
Cornell University
tz249@cornell.edu

***Abstract*—For our project, we decided to experiment, design and implement a song recommendation system. We used the Million Song Dataset [1] (MSDS) to find correlations between users and between songs to ultimately provide recommendations for songs to which users would prefer to listen. In this paper, we will discuss the problems we faced, the methods we considered to implement this system, and the results and analysis of our findings. We have also compared our findings with other experiments done in the field. From our findings, also we recognize numerous ways to improve the results of our recommendation system; we believe we have found a novel way to recommend songs that a user might enjoy.**

I. Introduction

With the advent of online services such as Pandora and iTunes, the music industry has shifted away from the traditional distribution model of selling physical copies of music to a cloud-based model that provides music for users to listen to. In this model, value is derived when these services present songs that the customer is interested in; either the customer purchases a subscription, or the customer pays for the song. In both cases, these music services can derive financial gain by improving their recommendations to potential customers of the songs they like. Thus, there is strong financial incentive to implement a good song recommendation system.

There are many interesting problems to tackle when building a song recommendation system. One issue that makes recommending songs difficult is that there is no straightforward similarity measure between songs; two songs that sound similar may be very different songs that have completely different audiences. Thus, we needed to figure out how to gain useful information that could be used to recommend other songs. Another issue is in identifying useful features of the songs; determining which features would be best for a learning algorithm is important, since good feature selection could improve recommendations. Lastly, we decided to use the MSDS, as well as data from participating partners[1], to train and test our recommendation system, and the scope of this data presented issues. The MSDS alone contained more than 280 gigabytes of data, which was a challenge to process and use efficiently.

[1]http://labrosa.ee.columbia.edu/millionsong/pages/additional-datasets

To address the problem of effectively measuring song similarity and recommendation, we used song metadata from the MSDS and user play history data from EchoNest[2]. After experimenting with several baselines, such as K-Nearest Neighbors (KNN) and matrix factorization, we created a program that utilized the user listening data and song metadata to suggest new songs that a user would most likely prefer. Metadata that we deemed had the greatest value in this metric were used as features. Lastly, we had several ways to address the large amount of data. The simplest way was to run our training and testing on a subset of the data. We also decided to filter our data in order to preserve the useful information while ignoring information with little value.

Our basic approach is to use a combination of clustering, song-based similarity metrics, and user-based similarity metrics to recommend songs. When recommending to a user in the test group, our algorithm produces three different sets of recommendations from these three different metrics. From these recommendations, as well as properties of the test user, we choose songs with the highest likelihood of being a song that the test user prefers.

Papers on related work in this field talk about how a specific implementation will improve the precision of recommendations. For example, the winner of the Kaggle MSDS competition[3] provided a Memory-based Collaborative Filtering method. Also, many other papers talked about different song similarity approaches, such as using acoustic data. These publications focused on deriving methods that provide a better precision. We surveyed some of these algorithms in our work and used them as baselines and in designing our final solution.

From our experimentation, we have derived several conclusions. First, we discovered the precision of our results tended to be low, which he attribute mainly to the lack of useful features and the fact that our program only recommends constant number of songs for a given user.

[2]The Echo Nest Taste profile subset, the official user data collection for the MSDS, available at: http://labrosa.ee.columbia.edu/millionsong/tasteprofile

[3]http://www.kaggle.com/c/msdchallenge

Second, we found that certain methods that we surveyed were more useful than others, when the properties of the users in the test set were considered (i.e. the number of songs in a users listening history). Lastly, we believe our method for calculating song similarity is a novel solution for the song recommendation problem, as it optimizes among numerous approaches to predict the best songs for users.

II. Problem Definition

The central issue is the recommendation of songs to a user. For a given user, we have their song history and play count for each song. From this, we want to produce a set of ten recommendations to the user. Then, we try to answer the question, "How do we use the song history of a user to provide recommendations that they will like?" Generally, this is done by looking at the songs that are most similar to the users songs, as well as the users who are most similar to the user according to their listening history.

In order to discern whether songs are similar to those in a query users listening history, our methods use the metadata collected from a song to weigh its similarity with another song. From this, we can choose the songs with the highest weights as the recommended songs. However, given that there is a lot of metadata about each song, a significant issue was deciding which features should be used to calculate song similarity. For this project, we focused on the features that we deemed were most likely to distinguish similar songs from different songs.

We also determine the best songs for a given user, in part, based on the song histories from similar users. However, this brings up another problem: How do we measure similarity among users? Our approach measured closeness among users by taking the cosine similarity measure between the two users' song histories. This is a major component in some of our algorithms.

Another problem that we considered is the cold-start and near-cold-start problem. The cold-start problem, in terms of song recommendations, refers to the situation in which the test user has no song history; since all our metrics use this for song prediction, a lack of song history is a difficult problem for our algorithms. It can be solved by simply recommending a subset of the most popular songs. This scenario also brings up another interesting situation, the near-cold-start problem. If the user has a very small song history, our algorithms may not return an accurate set of recommended songs. To address these issues and to try to achieve the best possible precision under any circumstance, we ran several experiments with our algorithms and devised a multi-faceted prediction algorithm.

Lastly, we have the issue of dealing with large quantities of data. The MSDS and accompanying datasets offer over 280 gigabytes of information about songs, their metadata, and user play counts. How do we deal with all the data? Often, we simply use a random sample of the data for training, validation, and testing. For example, we use the data in our user song history set with only the top one thousand songs in the song history. We also prune our data so that we focus on information that would maximize the value-to-performance tradeoff. Furthermore, we developed multithreaded, multiprocessor, and map-reduce implementations of certain methods to accelerate the processing of the tasks.

III. Data Analysis

The main source of data for this project was the Million Song Data Set (MSDS), released by Columbia Universitys Laboratory for the Recognition and Organization of Speech and Audio. It contains metadata and audio features for one million popular and contemporary songs, varying in genre, time period, and country of origin. Supplementary data provided by other groups and matched to songs in the MSDS was also used. This includes lyrics provided by *musiXMatch*[4], user-defined tags provided by *Last.fm*[5], and user listening data provided by the EchoNest.

A. User Listening Data

This data comes in the form of the number of times a given user has listened to a given song (play count). Throughout the project, we have roughly modeled user preferences by their played song history, so the algorithms we implemented that use this per-user song play count data learn and are evaluated based on this assumption.

B. Song Metadata

Among the large amounts of song metadata, we decided to focus on features that we deemed would be most relevant in determining similarity between songs, as well as those that we thought were interesting candidates for enhancing this metric. We decided that information like artist name, year composed, and some audio signal-related measurements like tempo would be most relevant in characterizing a song, whereas data on artist location would not matter significantly.

We used the lyrics data as features for those songs whose lyrics were provided. To respect copyright privileges, lyrics data for only about 23% of the MSDS songs were released in a bag-of-(stemmed)-words format. Ultimately, we decided to include this data, because we believed that it could exploit the semantic relationship in like-themed songs.

[4]http://labrosa.ee.columbia.edu/millionsong/musixmatch
[5]http://labrosa.ee.columbia.edu/millionsong/lastfm

IV. Methods and Algorithms

For all of our methods, we created sets of training, validation, and test data. For training data, we used unmodified user histories and song metadata. For validation and test data, we took the user histories and split it into two halves. The first half was used to test our algorithms, while the second half was used to evaluate our results. If a song recommendation for a user, predicted based on the history for that user in the first half, existed in the second set of data, then that recommendation was considered relevant. This is the same method used to create training, validation, and test data in [1].

A. Baseline algorithms

For this project, we devised a few baseline algorithms against which to compare our results. These methods provided a solid basis for evaluating the effectiveness of our later work. The first and simplest baseline method we implemented was a straightforward popularity-based algorithm that just recommends the most popular songs, as outlined in [1].

1) K-Nearest Neighbors (KNN): We also used the k-Nearest Neighbor algorithm on the user history data. For each test user, we found the set of users who were closest, using each song as a feature and weighted according to the play count for that song. From these results, we recommended songs (that the test user had not listened to) from the listening history of the closest users. Closeness between users is calculated by the minimum cosine distances between users song histories.

2) Matrix factorization: Singular Value Decompositions (SVD) are among the most successful recommender system algorithms. In its simplest form, SVD decomposes the user-song play count matrix M into a latent feature space that relates users to songs (and vice-versa). This is usually achieved by factoring M according to

$$M = U^T V$$

Where $U \in R^{k \times m}$ is a low-rank representation of the m users, and $V \in R^{k \times n}$ represents the n items. Once the representations U and V have been constructed, personalized recommendations are calculated for a user u by ranking each item i in descending order of the predicted feedback:

$$w_i = U_u^T V_i$$

The interpretation is that the users and songs can each be represented by k-dimensional vectors, where each dimension corresponds to one of k topics like (rock music, artist name etc). Each row of U, represents user u's degree of interest in each topic. Each row of V represnts the relevance of the song v to each topic.

The computation of SVD in our case is non-trivial due to the large number of missing values in M. The above decomposition is possible only if M is complete. Simon Funk has described a method to estimate SVD using only the known weights (playcount)[2]. The idea is to view the decomposition as a regression problem of finding the parameters of the vectors u and v, so as to minimize the error between the known actual song weights and the predictions.

$$Err(w) = argmin(\sum_{u \in U, s \in S}^{m,n} (V(w_{u,s}) - V) + R)$$

where $V(w_{u,s})$ is the predicted value.

As only few songs per user are known, an L_2 norm regularizer is added to prevent overfitting:

$$R = \gamma(||u||^2 + ||s||^2)$$

Now, we use adaptive stochastic gradient descent to minimize the above objective. We train the user and song vectors one feature at a time. Here are the update equations for user and song vector feature i:

$$u_i = u_i - \lambda(u'v' - u \cdot v)v_i + \gamma u_i$$
$$v_i = v_i - \lambda(u'v' - u \cdot v)u_i + \gamma v_i$$

Where u' and v' represent current feature values for user u and song v. γ is the regularization parameter and γ is the learning rate. For our experiments, we varied the latent factors from $k = 1$ to 20, and varied $\gamma \in \{10^{-3}, 10^{-4}\}$ with $\lambda_{init} = 10^{-2}$.

B. Main Algorithms

1) K-Means clustering: One of the approaches used was K-Means clustering. For all the users in the training data, we found clusters of users based on their song histories. From the clusters, we predicted for each user which cluster was the best fit according to proximity to its centroid. The most highly weighted songs in each cluster were selected as the songs to recommend for that particular user. Although this algorithm will approximate the accuracy of KNN, it is a lot faster and easier to tweak.

2) User Based Recommendations: Generally, given a new user, for which we want to obtain predictions, the set of items (in this case songs) to suggest is computed by looking at similar users. This strategy is typically referred to as user-based recommendations. In user-based recommendations, the scoring function, on the basis of which the recommendations are made, is computed as

$$sim(u,v) = \frac{u \cdot v}{||u||^{\frac{1}{2}} \cdot ||v||^{\frac{1}{2}}}$$

which is just a cosine similarity between two users u and v. One common way of finding the cosine similarity is finding the number of common items the users share between them. Thus,

$$sim(u,v) = \frac{\#\text{common items}(u,v)}{\#\text{items}(u)^{\frac{1}{2}} \cdot \#\text{items}(v)^{\frac{1}{2}}}$$

The strategy relies on the fact that each user belongs to a larger group of similarly-behaving individuals. Consequently, items frequently purchased (listened) by the various members of the group can be used to form the basis of the recommended items.

The overall procedure is as follows: We try to find the weight of each item I_i in the system for the new user u by finding the set of users V in the training set who have listened to the item I_i and then summing the user similarity using the similarity function. Thus, weight for song I_i, w_{I_i} is,

$$w_{l_i} = \sum_{v \in V} sim(u,v)$$

Finally, all the items are sorted by their weights and top-N are recommended to the user.

The cosine similarity weighs each of the users equally which is usually not the case. In fact, a user should be weighted less if he has shown interests to many variety of items basing it on the knowledge that he does not discern between songs based on their quality of the item, or he just likes to explore items. Likewise, user is weighted more if he listens to very limited set of songs. Thus, we re-modeled the similarity measure by parametrizing it with this weighing rule.

$$sim(u,v) = \frac{\#\text{common items}(u,v)}{\#\text{items}(u)^{\alpha} \cdot \#\text{items}(v)^{1-\alpha}}, \alpha \in [0,1)$$

We also, added another parameter to sharpen this normalization. The effect of this exponentiation is this: when γ is high, smaller weights drop to zero while larger weights are (relatively) emphasized. Thus our item weight equation is now,

$$w_{l_i} = \sum_{v \in V} (sim(u,v))^{\gamma}, \gamma \in [0,1)$$

3) Item Based Recommendations: User-based recommendations identify a neighborhood of users that, in the past, have exhibited similar behaviors. An alternative approach is to build recommendation models that are based on items. The key motivation behind this scheme is that a user will more likely purchase (or listen to) items that are similar to items he already purchased; thus, in this approach, the historical information is analyzed to identify relationships between the items so that the purchase of an item often leads to the purchase of another item. The algorithm first determines the similarities between the various items, then uses them to identify the items to be recommended. The key steps in this approach are (i) the method used to compute the similarity between items and (ii) the method used to combine the similarities and provide recommendations.

During the model building phase, for each item i, the k most similar items $\{i_1, i_2, .., i_k\}$ are computed and their corresponding similarities are stored. Now, for each user, that has listened to a set U of items, we first identify the set V of candidate items for recommendation by taking the union of the k most similar items that are already in U. Then, for each item $c \in C$, we compute its similarity to the set U as the sum of similarities between all items $u \in U$ and c, using only the k most similar items of u. Finally, as in user-based recommendation, the candidate items for recommendation are sorted and the top-N items are recommended.

$$w_c = \sum_{u \in U, c \in C} sim(u,c)$$

Item similarity: We used two approaches to calculate Item-Item similarity used in the Item based recommendations.

(i) Without metadata: We used the similar method as in user-user similarity by finding the common users that are shared by items i and j. Thus, similarity function is,

$$sim(i,j) = \frac{\#\text{common users}(i,j)}{\#\text{items}(i)^{\frac{1}{2}} \cdot \#\text{items}(j)^{\frac{1}{2}}}$$

(ii) With Metadata: the second approach involves using the song metadata to compute a cosine similarity score between two songs. The features used for each song in this calculation are artist name, tempo, loudness, energy, year the song was composed, the five most common lyrics in the song, and the five most heavily weighted user tags for the song. All the features, except tempo, loudness, and energy, were treated as binary features.

Lyrics were included as features in an attempt to take advantage of any possible semantic similarity between songs. For example, if a user enjoys love songs, then this approach could exploit the fact that love songs may often share love-themed lyrics.

4) Multi-faceted Prediction Algorithm: The multi-faceted prediction algorithm uses multiple algorithms to more precisely recommend songs for the user. We do this because we have observed that different methods work better under different circumstances. By combining the different methods and weighing a specific methods prediction more based on the size of the test users listening history, we can maximize our song recommendation systems precision.

The multi-faceted prediction algorithm uses the trends observed in Figure 7 and Figure 8. For different sizes of the test users song history, different methods of prediction give optimal recommendations. Specifically, when the test users song history is small, predictions based on song similarity metrics tend to perform better; when the test users song history is large, predictions based on similar user metrics tend to perform better. This has a direct application to the cold start or near cold start situation, in which predictions based on a test users song history tend to be noisy and inaccurate, since the similar users do not accurately represent the test user. In these situations, making recommendations based on song similarity is better. Thus, the multi-faceted prediction algorithm aims to use this phenomenon to make the best predictions. Using our validation sets, we tuned the weights to produce the optimal set of predictions.

V. Results

A. Precision Metric

The metric we used to determine the quality of our algorithms was the precision value of our song recommendations. Specifically, our metric was the proportion of relevant songs over a constant number of recommendations. This form of precision was used because it is similar to the metric used in the Kaggle competition. This allows us to make comparisons between our results and the results of related work from the competition. Furthermore, in recommendation systems, precision is the key metric in determining the quality of the recommendations. In [3], the authors who were building an online recommendation system of web searches believed precision was the best metric for recommendation systems. Thus, precision was selected as our metric for determining the quality of our algorithms results.

Due to the nature of our data, we found it impossible to run our program on the entire set of data provided by the MSDS. The following results reflect a subset of the song metadata and the song history of 10,000 users.

B. KNN

We begin by considering the KNN baseline method and compare its performance against the popularity baseline. Figure 1 shows that even a simple popularity-based recommendation method offers better precision than a KNN approach. This suggests that similarity based purely on user play count is not a good measure of predicting new songs; in the end, people would rather listen to popular music rather than what other similarly behaving users listen to.

C. Matrix Factorization

With the singular value decomposition (SVD) method (Figure 2), we found that precision was exceptionally poor. Although the theory behind this method is compelling, there is not enough data for the algorithm to arrive at a good prediction. The median number of songs in a users play count history is fourteen to fifteen; this sparseness does not allow the SVD objective function to converge to a global optimum. Ultimately, this results in poor prediction precision rates.

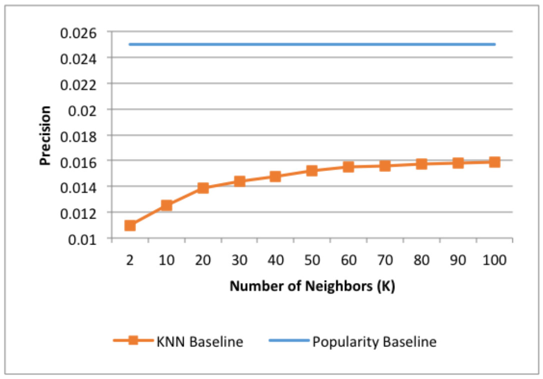

Fig. 1. **KNN baseline performance v. popularity baseline performance**

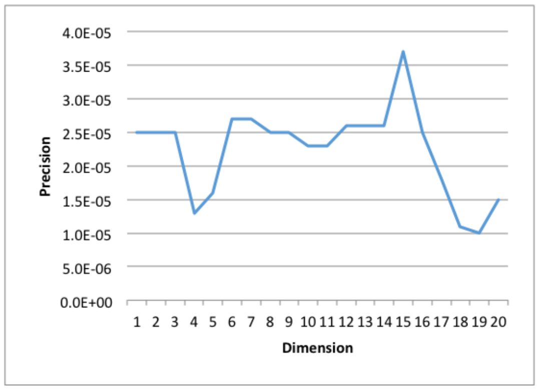

Fig. 2. **Precision of SVD with rate of** 0.510^{-4} **and** γ **of** 1.010^{-5}

D. K-Means

With K-Means, we start to see precision results that exceed the popularity baseline (Figure 3). This can be attributed to the fact that the K-Means method finds more generalized clusters of users with similar behavior (as compared to KNN) and offers predictions from this more general pool of user song histories, which ends up being slightly more accurate.

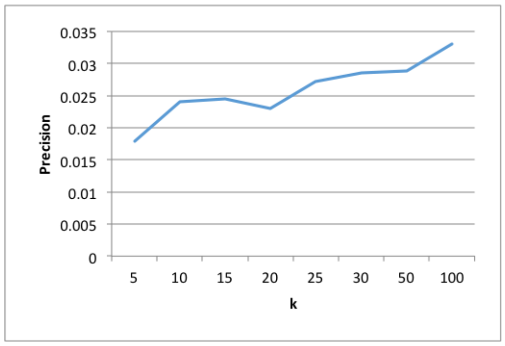

Fig. 3. **K-Means performance**

E. User Based Recommendations

Next, we consider the user-based song-recommendation method, which provided dramatic gains in precision over our other approaches. Figures 4 and 5 illustrate how different values of the α and γ parameters change the recommendation precision; we see that an α value of 0.8 and a γ value of 5 offers the best precision. The α and γ factors are the same factors expressed in the Methods section.

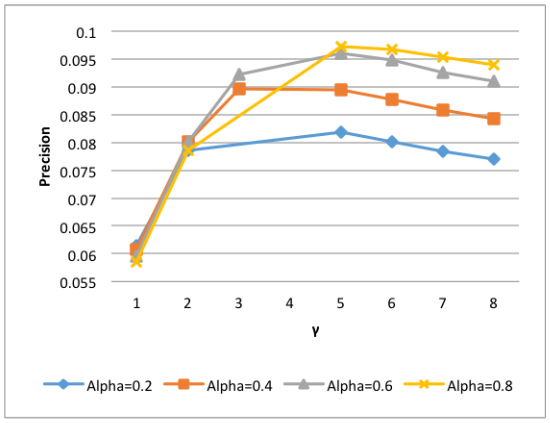

Fig. 4. **Precision of user-based approach v. γ**

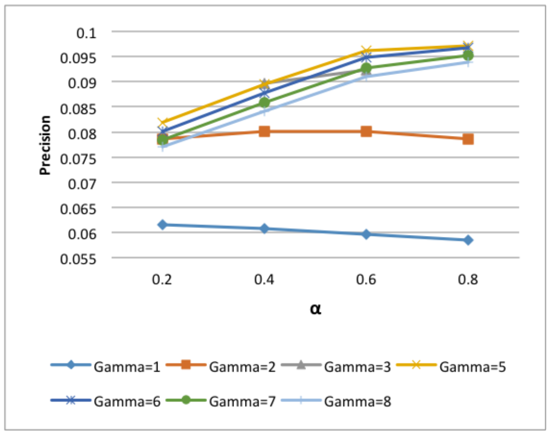

Fig. 5. **Precision of user-based approach v. α**

F. Item Based Recommendations

We also got comparably high precision with item-based recommendations, specifically using the song similarity metric that does not use the song metadata. The item-based recommendation algorithm that uses metadata to calculate song similarity yielded lower precision, but still performed better than the popularity baseline, KNN, and K-Means. The relative performance of the different methods is summarized in Figure 6.

Finally, we present some indication that the multi-faceted prediction algorithm that we implemented would offer improved results over a purely user-based recommendation method or item-based method, which are our two highest-performing algorithms. Figures 7 and 8 show the performance

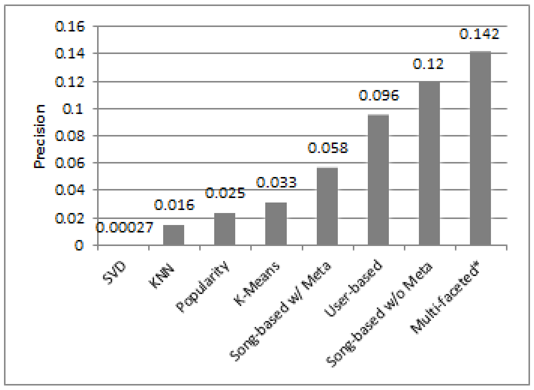

Fig. 6. **Comparison of recommendation methods**

of the user- and item-based methods, respectively, as the played song history count for a given user increases.

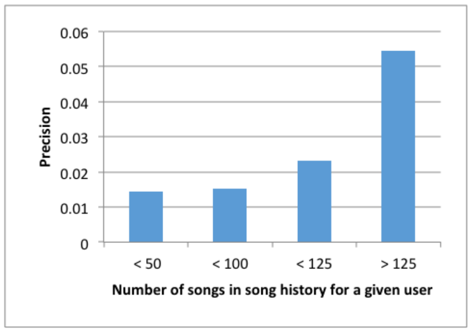

Fig. 7. **User-based v. listened song count**

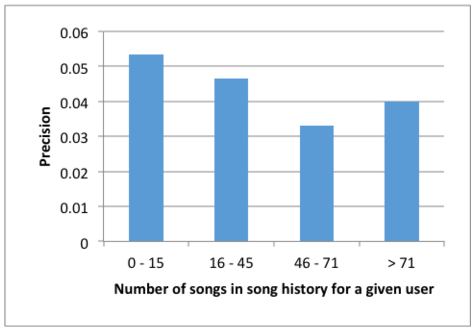

Fig. 8. **Item-based v. listened song count**

Given these trends, the benefit of a recommendation system that chooses songs based on different underlying algorithms (i.e. user-based or item-based) depending on the number of songs a user has listened to is apparent. When the song history of a user is too small to leverage the power of the user-based recommendation algorithm, we can offer recommendations based on song similarity, which yield better results with smaller song histories.

Figure 6 illustrates the effectiveness of the multi-faceted prediction algorithm. By combining user- and item-based methods in a way that stresses each ones strengths, we get the highest prediction precision overall.

VI. Discussion

An analysis of our results has provided a deeper understanding of our data and our questions. We used several of our methods to find similarities between users and songs. Using our metrics, we produced data that represents the correlation between song predictions based on song similarity and similar users and the precision of the song recommendations. An interesting property we discovered was how the size of the user history affected the precision of different metrics. Test users with smaller song histories tended to be have better precision with metrics based on song similarity, while those with larger song histories tended to have better precision with metrics based on user history.

Our results show a generally low level of precision produced by the song recommendations. Our multi-faceted prediction algorithm resulted in a precision of 14.2%. This shows that the song recommendation problem is a difficult one and can be attributed to the lack of high correlation between a users song history and their song preferences in the data set. Regardless, our song recommendation system uses a novel approach of incorporating the results of our multiple methods to produce an optimal set of song recommendations, while other researchers are mostly focused on optimizing individual algorithms that exclusively use a single approach.

VII. Related Work

A lot of related work has been completed on the Kaggle competition site relating to the MSDS. Although the specific implementations of the competitions various solutions were not revealed, we used the scoreboard of the competition as a point of comparison for our algorithm against others. The highest average precision achieved in the competition was 17%, while our highest average precision was 14.2%.
There have been a lot of published works as well, regarding the different methods to evaluate the data to produce song recommendations. Some of this work is detailed below.

A. About song similarity

There are a number of different ways to measure the similarities between songs, such as semantic embedding model [4] and acoustic similarity model [5]. However, music similarity measures are subjective, which makes it difficult to rely on ground truth. This issue is addressed in [6] and [7]. For our final algorithm, we inherently assumed that this metric was unreliable and used user song history to provide better results.

B. Meta-data

Meta-data provides rich information for the songs, which helps classify the music. However, meta-data suffers from a popularity bias. People are more likely to rate the artists and songs that are popular. Thus, little information can be found for those less known artists and songs. [8]

C. Algorithms

Numerous other methods have been considered and implemented [9]. One method that is very similar to our SVD approach is called Latent factor mode Bayesian personalized ranking (BPR). The factorization is similar:

$$M \approx \bar{M} = U^T V$$

At a high level, the algorithm learns U and V such that for each user, the positive (consumed) items are ranked higher than the negative (no-feedback items). This is accomplished by performing stochastic gradient descent on the following objective function:

$$f(U,V) = \sum_{\substack{u,i,j: \\ M_{u,i} > M_{u,j}}} ln(1+e^{-U_u^T(V_i-V_j)})+\lambda_i||U||_F^2+\lambda_2||V||_F^2$$

where u is a user, and i and j are positive and negative items, respectively, for user u. This is a different approach since it transforms the original problem into an optimization problem.

D. Comparison

For the MSDS problem, we compare our results to the results from the Kaggle competition. The winner , aio[10], has a recommendation system with mean average precision of 0.1791 for the top 500 songs. Our final precision is 0.142 for the top 10 songs. We believe our results are good because, when a user looks at a set of recommendations, he usually only focuses on the top songs, in the same way that we rarely go to the second page of results in a Google search.

VIII. Future Work

The song-with-metadata approach yields low precision compared to the user-based and song-without-meta methods. This may be attributed in part to low similarity scores generated by the cosine similarity distance metric, which, in turn, could be because the features chosen to represent a song are a poor indicator of what users value in their choices of music. Despite this, the precision attained is significantly higher than the popularity baseline, and it is conceivable that incorporating more features or choosing different combinations of features or simply playing with weight values for the chosen features could improve the method even more.

We believe that a very effective method of music discovery lies in compiling the results from a variety of different individual recommendation algorithms, like the results from our multi-faceted predictor have indicated. Our work looked at leveraging mostly user listening behavior and song characteristics to identify similarities between that data and that of a query user to make some recommendations. Generally, further work can be done to develop more recommendation algorithms based on different data (e.g. the how the user is feeling, which www.ragechill.com tries to capture), and these new methods can be used to further enhance multi-faceted prediction.

IX. Conclusion

Our work has given us a keen insight into recommendation systems for music. Our results have shown how we can use song metadata as well as data from users histories to provide the best set of songs to recommend. Furthermore, we observed the low precision results from both our own work and those from related work, which forced us to think about ways to improve overall precision. Ultimately, we devised a novel method to provide a recommendation system aimed to maximize precision of our song recommendations.

Acknowledgment

The authors would like to thank Thorsten Joachims, Joshua Moore!

References

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[2] http://sifter.org/∼simon/journal/20061211.html

[3] Sergey Brin, Lawrence Page.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, Volume 30, Issues 17, April 1998, Pages 107117

[4] Law, E., Settles, B., Mitchell, T.: Learning to tag from open vocabulary labels. In: ECML. (2010)

[5] Bertin-Mahieux, T ., Eck, D., Mandel, M.: Automatic tagging of audio: The state-of-the-art. In Wang, W., ed.: Machine Audition: Principles, Algorithms and Systems. IGI Publishing (2010) In press.

[6] Berenzweig, A.: A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. Computer Music Journal 28(2) (June 2004)

[7] Ellis, D.P.W., Whitman, B., Berenzweig, A., Lawrence, S.: The quest for ground truth in musical artist similarity. In: ISMIR. (2002)

[8] Pampalk, E., Dixon, S., Widmer, G.: On the evaluation of perceptual similarity measures for music. In: Intl. Conf. on Digital Audio Eects. (2003)

[9] Law, E., West, K., Mandel, M., Bay, M., Downie, J.S.: Evaluation of algorithms using games: the case of music tagging. In: Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR). (October 2009)

[10] Fabio Aiolli.: Preliminary Study on a recommender system for the Million Songs Dataset challenge. (2011)