ML-Based Collaborative Filtering for Cross-Cultural Music Recommendations

Yuzu Ido
yuzu@stanford.edu

Stephan Sharkov
stpshrkv@stanford.edu

Eunice Yang
eunicey@stanford.edu

1 Introduction

In the rise of music streaming services such as Spotify and Apple Music in the past decade, research on music recommender systems has increased significantly in both industry and academia. Given the breadth of access users now have to music on these services (generally on the order of tens of millions of unique songs), recommendation systems play an important role in filtering these choices, suggesting songs that fit user preferences, and maximizing user enjoyment on the platform. A key goal of these recommender systems is thus to generate predictions on items with which users have not interacted yet, and subsequently recommend the subset of items (such as songs or artists) that the users may particularly enjoy.

For many individuals, the music they initially listen to is highly influenced by their environment, such as the country they reside in or the family and friends around them. We believe this generally limits the average listener to music that largely exists within their own cultural background, such as Americans listening to mainly American music. In the 21st century, cultural globalization has taken a new form, but most music recommendation algorithms still typically suggest songs and artists with the same cultural cluster as those in the user's previous listening history.

Our project is thus twofold: (1) to use machine learning and collaborative filtering methods to generate useful predictions of potential artists of interest, and (2) to develop a novel algorithm that re-weighs predictions to maximize recommendations of artists coming from different countries than the user. The input to our models are individual users (particularly their prior music listening history), and we use different collaborative filtering methods (both neighborhood and model based) to output predictions and generate 10 artists from a different country of origin that they may enjoy.

2 Related Work

Collaborative filtering (CF) is a popular approach for recommender systems wherein a specific user's predictions for an item are generated based on information from many users (collaboration). The general concept for this approach was established by the Goldberg group in 1992 to create a mail and repository system called "Tapestry" [1].

Today's methods for collaborative filtering can be split into two main approaches, neighborhood-based and model-based. Neighborhood-based methods largely involve computing relationships between items or users. When item-oriented, the approach evaluates user preferences based on "neighboring" items' ratings [2,3]. Example methods include k-Nearest Neighbors (kNN) and k-Means.

Model-based methods assume that users and items can be represented as unknown latent factors, and these latent factors can be used to reproduce a new user-item rating matrix [4,5]. This is largely reliant on Matrix Factorization, a foundational technique that allows the users and items to be represented in a lower-dimensional space. This method has been popularly used in competitions such as the Netflix Prize and has demonstrated very positive results [2], especially for problems with sparse data. In theory, uncovering latent features can lead to higher accuracy because they better account for implicit variables, often resulting in improved performance over approaches like kNN. Some commonly used approaches include singular value decomposition and probabilistic matrix factorization [9].

Overall, collaborative filtering approaches are advantageous in that these ML models can help users identify new interests [6] and do not require information about item content [7]. This was especially important for our LFM-1B dataset, which did not have readily available features for the items. The most notable disadvantage of CF is an inability to handle new items not seen by the model during training, though this was not a significant concern for us during the implementation of this project.

We utilize and build on these CF methods to generate predictions of potential artists of interest for music streaming users. There is notably limited existing literature on recommender systems that generate modified ratings and provide recommendations based on identifiers that

CS229 Project 2021, Stanford, CA, USA.

are actually distinct from that of the user (in our case, country of origin). This is a novel aspect of our project, with unique implications for cross-cultural facilitation.

3 Dataset and Features

Our project relies on the LFM-1B database from last.fm, which is a large global online music service [8]. LFM-1B contains a dataset with one billion listening events from more than 120,000 users characterized by the user, artist, track, album, and timestamp. The complete dataset can be represented as a 120175×585095 matrix containing the playcount for every (user, artist) pair (**LFM-1B_LEs.mat**), where LE stands for listening event. We additionally use a file containing basic users demographic information, in particular the country of origin (**LFM-1B_users.txt**).

For our initial preprocessing, we first randomly selected 5000 users from the total 120175 users in the matrix; this allowed us to maintain a data set with a size that is both computationally manageable to run with our available resources and still large enough to provide the model with sufficient information to learn. Of these 5000 users, we only retain the 2312 users who reported a country of origin, as our project centers on using the user's country of origin to recommend artists from outside the user's culture (country). We also randomly selected 20000 artists from the 585095 total artists provided by LFM-1B.

Additionally, the basic users.txt information file contains 120322 users while the LEs.mat matrix contains 120175, as the creators of the dataset chose to exclude the subset of users who listened to fewer than 10 unique artists from the matrix. Thus, we also needed to find and delete the missing 147 users in our data frame of users so we could match up our indexing across the files.

From our 2312 users and 20000 artists, we generated an artist-country-playcount matrix based on the user-artist-playcount matrix, by aggregating the playcounts for each artist by all users from a given country. Furthermore, for each artist, the country with that artist's highest playcount will serve as our estimate for the artist's country of origin, as we make the assumption that people from the same country will listen to a particular artist most.

The top 10 countries of origin for users and artists from our 2312 users and 20000 artists are shown below (Table 1, Table 2). Because we randomly select 20,000 artists from 585,095 total artists, there are a substantial number of artists to whom none of our 2312 users listened; these are coded as NULL.

Table 1: User Country of Origin

Code	Country	% of Users
US	United States	19.2
RU	Russia	8.7
UK	United Kingdom	8.6
DE	Germany	7.9
PL	Poland	7.9
BR	Brazil	7.4
NL	Netherlands	2.3
FI	Finland	2.3
UA	Ukraine	2.2
AU	Australia	2.2

Table 2: Predicted Artist Country of Origin

Code	Country	% of Artists
NULL	No Listens	61.5
US	United States	5.8
PL	Poland	3.5
RU	Russia	3.5
UK	United Kingdom	3.2
DE	Germany	2.7
BR	Brazil	0.4
IT	Italy	1.7
FI	Finland	1.4
NL	Netherlands	1.1

Finally, the matrix is provided in a SciPy sparse matrix format, but we convert it to Pandas DataFrames and melt the 2313×20000 sparse matrix into a list of 46865 (user, artist, playcount) data point triples, meaning that the density of the dataset is 0.101%. The main benefits of the list format include compatibility with the Surprise recommendation scikit library and ease to split these data points for training and testing. In most of our experiments, we do a 5-fold cross-validation process in which we use 80% for training and 20% for testing. For stability purposes, we converted the raw playcounts, which range from 0 to thousands, to their base-2 logarithms.

4 Methods

4.1 Baseline

As a baseline algorithm, we predict

$$\hat{r}_{ua} = \mu + b_u + b_a$$

where μ is the mean of all (non-zero) ratings in R, and b_u and b_a the bias term for the user and artist respectively. The bias terms b_u, b_a, which represent the "observed deviations of user u and [artist a], respectively, from the average," are estimated by solving the least

squares problem

$$\min_b \left(\sum_{(u,a) \in R'} (R_{ua} - \mu - b_u - b_a)^2 + \lambda \left(\sum_u b_u^2 + \sum_a b_a^2 \right) \right)$$

as explained in [11].

4.2 Matrix Factorization

Our primary machine learning algorithm is matrix factorization (MF), specifically the probabilistic matrix factorization algorithm introduced in [10]. We are given a matrix R of playcounts, in which the entry R_{ua} is the number of times the user u has listened to the artist a. R has dimension $m \times n$, where there are m users and n artists. It is a sparse matrix because each user generally listens to a very small subset of the artists.

Matrix factorization aims to find two matrices, Q and P^T, such that $R \approx QP^T$. Q is an $m \times k$ matrix, in which each user is represented by one row with k features; similarly, P is an $n \times k$ matrix, in which each artist is represented by one row with k features; k is a hyperparameter of the algorithm. Then,

$$\hat{r}_{ua} = q_u p_a^T$$

the product of one user row q_u from Q and one artist column p_a^T from P^T, is our predicted playcount for that (user, artist) pair, giving an approximation for the original matrix R with predictions for empty entries.

Our loss function for matrix factorization is the sum of the squared errors with $L2$ regularization. Mathematically,

$$L = \left(\sum_{(u,a) \in R'} (R_{ua} - q_u p_a^T)^2 \right) + \lambda \left(\sum_u \|q_u\|_2^2 + \sum_a \|p_a\|_2^2 \right)$$

where R' is the set of all ratings in R and λ is a hyperparameter.

We perform stochastic gradient descent to minimize the loss. This algorithm starts with an initial set of parameters that repeatedly updates with a step in the direction of the deepest decrease of the loss for every example. The step size is determined by the hyperparameter α, also known as the learning rate.

Because we are doing stochastic gradient descent, instead of full batch gradient descent, we will take the gradient of

$$M = (R_{ua} - q_u p_a^T)^2 + \lambda(\|q_u\|_2^2 + \|p_a\|_2^2)$$

Let $\epsilon_{ua} = R_{ua} - \hat{r}_{ua}$. Then we see that

$$\begin{aligned} \nabla_{q_u} M &= 2\epsilon_{ua}(-p_a) + 2\lambda q_u \\ \nabla_{p_a} M &= 2\epsilon_{ua}(-q_u) + 2\lambda p_a \end{aligned}$$

Our parameters in the matrix factorization are the vectors q_u for each user and p_a for each artist, which get updated in each step as follows:

$$\begin{aligned} q_u &\leftarrow q_u - \alpha \nabla_{q_u} M \\ &\leftarrow q_u - \alpha \left(2\epsilon_{ua}(-p_a) + 2\lambda q_u \right) \\ p_a &\leftarrow p_a - \alpha \nabla_{p_a} M \\ &\leftarrow p_a - \alpha \left(2\epsilon_{ua}(-q_u) + 2\lambda p_a \right) \end{aligned}$$

The advanced version of matrix factorization involving the bias terms b_u and b_a is known as singular value decomposition (SVD), popularized in [5].

4.3 k-Nearest Neighbors

For comparison to matrix factorization, we are also using a k-nearest neighbors (kNN) collaborative filtering approach, discussed in [11]. We are considering the distance between all pairs of users (u, v) using the mean squared difference similarity, which can be defined as follows. Let I_{uv} be the set of artists to whom users u and v have both given ratings. Then, the mean squared difference is given by

$$\text{msd}(u, v) = \frac{1}{|I_{uv}|} \sum_{a \in I_{uv}} (R_{ua} - R_{va})^2$$

and the mean squared difference similarity is

$$\text{sim}(u, v) = \frac{1}{\text{msd}(u, v) + 1}$$

For a target user u and artist a, we find the set $N_a^k(u)$, the set of the k nearest neighbor users to u who have previously rated a, where k is a hyperparameter of the algorithm. Then our basic k-nearest neighbor prediction for user u, artist a is

$$\hat{r}_{ua} = \frac{\sum_{v \in N_a^k(u)} \text{sim}(u,v) R_{va}}{\sum_{v \in N_a^k(u)} \text{sim}(u,v)}$$

We also use an improved k-nearest neighbor prediction (kNN with means) that takes into account the users' mean ratings, as follows:

$$\hat{r}_{ua} = \mu_u + \frac{\sum_{v \in N_a^k(u)} \text{sim}(u,v)(R_{va} - \mu_v)}{\sum_{v \in N_a^k(u)} \text{sim}(u,v)}$$

5 Experiments, Results, Discussion

Our cross-cultural music recommendation system has two large phases. In the first, we compute predictions of ratings based on algorithms such as matrix factorization and k-nearest neighbors without considering countries of origin. This phase invites a quantitative evaluation of our performance.

Our first quantitative evaluation metric to compare the performance of various algorithms on the test set is the root-mean-squared error, or RMSE, defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|R'|} \sum_{(u,a) \in R'} (R_{ua} - \hat{r}_{ua})^2}$$

Our second quantitative evaluation metric is the mean absolute error, or MAE, defined as follows:

$$\text{MAE} = \frac{1}{|R'|} \sum_{(u,a) \in R'} |R_{ua} - \hat{r}_{ua}|$$

We perform 5-fold cross validation across the algorithms and measure their RMSE and MAE.

	Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Baseline	RMSE	1.88401947	1.90824763	1.87772364	1.88866466	1.89654485
	MAE	1.53309983	1.55769875	1.52827716	1.54377897	1.54589434
MF, $k = 20$, $\alpha = 0.005$, $\lambda = 0.02$	RMSE	2.03202189	1.98315977	2.02619297	1.96551696	1.98227644
	MAE	1.56913812	1.53247659	1.56497909	1.52706266	1.52407263
MF, $k = 100$, $\alpha = 0.005$, $\lambda = 0.02$	RMSE	2.00426576	2.01926088	2.03832999	2.00502754	2.01710413
	MAE	1.54090497	1.54148313	1.55648747	1.5394203	1.54881289
SVD, $k = 100$, $\alpha = 0.005$, $\lambda = 0.02$	RMSE	1.91043931	1.90560266	1.90608345	1.92647848	1.90336738
	MAE	1.53030798	1.53032941	1.52827149	1.54304269	1.52945145
Basic kNN, $k = 40$	RMSE	2.03764018	2.03821978	2.03985588	2.04109587	2.04734938
	MAE	1.63955094	1.63320747	1.63807946	1.64289437	1.65438092
kNN with means, $k = 40$	RMSE	2.00338222	1.98509416	1.9851912	2.00875524	1.99640174
	MAE	1.6052293	1.58795244	1.59095355	1.60552306	1.60672749
kNN with means, $k = 50$	RMSE	1.99748787	1.97569048	2.02128141	1.98701072	1.9828416
	MAE	1.60135872	1.58241625	1.62663366	1.58982409	1.59163167

When comparing these various algorithms, we see that in fact, the baseline has the best performance, with an average RMSE of 1.89104 across the 5 folds. Between matrix factorization with $k = 20$ and $k = 100$, we see that $k = 20$ has a slightly better performance with an average RMSE of 1.9978, over $k = 100$ with an average RMSE of 2.0168; this could be attributed to overfitting of the model to the test set. The SVD, which is an advanced version of MF, has an average RMSE of 1.91039. The kNN with means, which has an average RMSE of 1.99576, does show an improvement over basic k-nearest neighbors with the same $k = 40$, which has an average RMSE of 2.04083. Thus, we chose to focus on kNN with means, and ran with $k = 20, 30, 50$, and 60 to find that $k = 50$ gives the best performance, with an average RMSE of 1.99286.

There are a number of potential reasons why our baseline method performed the best based on RMSE and MAE. One possibility could be based on the scarcity of our data, which was a challenging aspect of working with sparse matrices populated highly by 0s. The density of our data was 0.1%, and this might have resulted in underfitting of our models and comparatively higher RMSEs. Another factor in the good performance of baseline is the consideration of the bias terms b_u and b_a. Note that when comparing MF and SVD with the same hyperparameters, $k = 100, \alpha = 0.005$, and $\lambda = 0.02$, the major difference lies in the use of the bias terms, resulting in average RMSEs of 2.0168 and 1.91039 respectively; this suggests that including bias gives better performance.

Now, the second phase in our music recommendation system takes the predicted ratings for every (user, artist) pair generated in the first phase and incorporates the cross-cultural aspect to generate our final recommendations. Specifically, given a target user, we will find the top ten predicted artists whose estimated country of origin (that is, the country in which they are most popular) is different from the user's. After generating our initial predictions and top ten artists for each user, we systematically eliminated artists from a user's top ten if the artist's predicted country of origin was the same as the user's. Missing artists were filled in with the next highest predictions to ensure users received ten artists. For our entire set of users, we made 1824 deletions from baseline predictions, 1667 deletions from kNN with means, 1720 deletions from kNN, and 2312 deletions from MF. This is an interesting sign that MF could be generating more ratings relatively closely clustered to the user's previous listening history (related to factors based on country or cultural location), even though its RMSE was still higher than baseline.

We will further qualitatively evaluate the performance in this phase by finding the artists whom a particular target user already likes and the artists recommended by our system, and comparing their styles of music as well as the cross-cultural diversity. For the target user with User ID 4, from the US, we find all artists which have a known playcount higher than 50 to understand their taste in music (Table 3).

Table 3: Known Favorite Artists

Artist	Genre
Scorpions	Rock
Christian Burns & Paul Van Dyk	Pop, Rock
Marc Marlberg with Kyau & Albert	Trance
Joan Sebastian	Regional Mexican
Rotfront	Reggae, ska
Blackthorne	Hard Rock
Dj Mangoo	Electronic
Vinylshakerz	Tech, Electro
Alternative Mix	N/A
Calin with Fantastic Plastic Machine	Big Beat, Electronic

Table 4: Our Recommended Artists

Artist	Genre	True Ctry	Pred Ctry
The Little Ones	Indie Pop, Rock	US	RU
Insolence	Nu Metal, Rapcore	US	RU
Sulin	Rap	PL	IT
Jesse McCartney	Pop, Pop Rock	US	FI
Missy Elliott	Rap, Hip Hop	US	FI
Scroobius Pip	Hip Hop	UK	DE
Short Dawg	Rap	US	PL
Enslaved	Heavy Metal	NW	UK
Oasis	Rock	UK	NZ
U.N.S.I.N.	Alt Metal	GR	RU

We then get the top ten predictions for User ID 4, for which the artist's predicted country does not matches the US (Table 4). However, we found the true countries of those artists on Google, and we can see that they generally do not match; furthermore, many are from the US, which we hoped to avoid. However, there are some regional and cultural similarities such as for "Oasis", "U.N.S.I.N" and others, as well as similarity of genres between recommended artists and the known favorite ones.

The predicted country was based on where each of the artists has the maximum user-based playcount, which in reality is not necessarily true. Potential improvement could include getting the maximum playcount statistics from all the users, rather than the 2312 we chose for running our algorithms. Another improvement might include consideration of the top three countries where the artist is most popular. Both of those provide more detailed information on the artists, but were not currently achievable for us due to limits in computation power.

6 Conclusion, Future Work

Overall, we were able to develop predictive models and use collaborative filtering approaches to recommend new artists. We also developed a reasonable approach to reweigh predictions based on our desire to maximize recommendations from different countries of origin, which has interesting implications for cross-cultural exchange in an increasingly globalized Internet. Both our models and our country reweighing could benefit from some adjustment—for example, even the SVD matrix factorization approach demonstrated a RSME slightly above baseline and our country popularity prediction accuracy could be improved. Some potential ideas that we believe are immediately applicable and interesting for future work include:

1) Expanding capability to a higher number of users and artists. We were limited in computational power, as converting between sparse and dense matrices was very RAM-expensive. These were challenging aspects of working with very sparse data that we were new to when starting this project. Building on our methods and using a larger subset of the available data will likely improve the final predictions.

2) Considering more advanced methods for reweighing predictions and penalizing based on country. Simply using country as a metric may not be completely sufficient to facilitate the extent of cross-cultural exchange we envision, as there are groups of countries with relatively similar cultures and languages (i.e. US and Canada). It would be interesting to try methods that actually variably adjust predictions with this country similarity in mind.

3) Developing models that incorporate content-based filtering, in support of our existing collaborative-based filtering. Though our LFM-1B dataset had limited features for artists, incorporating another datastream with information about the artists, such as demographics, genres, or even whole music files, could open the avenue to very interesting hybrid approaches and models.

7 Contributions

All authors contributed equally to this work. Each member of our team took lead on different aspects of our project as described below, though all members were still involved in implementation of almost all aspects. As preparation to begin the project, Stephan took the lead on setting up the environment and virtual machine for the project in the Google Cloud Platform, Eunice considered the requirements for data pre-processing, and Yuzu thought about the theory and implementation of the collaborative filtering. Once we met with our mentor Chris and solidified our direction, we met through Zoom multiple times where we all talked through the specific implementation details. After the milestone, we continue working together on the next parts of the project but split the main responsibilities. Yuzu took over practicing models and further implementation on a smaller dataset on Google Colab. Stephan led the process of scaling up our code for the bigger portion of the dataset and running it on Google Cloud. Eunice focused on implementing and factoring country information into our predictions after running the models. Meeting through Zoom multiple times (read: all the time), we helped each other when one of us would get stuck with a certain bug or error.

8 References

[1] Goldberg, David, et al. "Using collaborative filtering to weave an information tapestry." *Communications of the ACM* 35.12 (1992): 61-70.

[2] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 42.8 (2009): 30-37.

[3] Rosa, Ricardo Erikson Veras De Sena, et al. "Improving Prediction Accuracy in Neighborhood-Based Collaborative Filtering by Using Local Similarity." *IEEE Access* 8 (2020): 142795-142809.

[4] Billsus, Daniel, and Michael J. Pazzani. "Learning collaborative information filters." *ICML.* Vol. 98. 1998.

[5] Funk, Simon. "Netflix Update: Try This at Home." *The Evolution of Cybernetics*, 11 Dec. 2006, sifter.org/simon/journal/20061211.html.

[6] Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* (2009).

[7] "Collaborative Filtering Advantages & Disadvantages." Google Developers, Google, developers.google.com/machine-learning/recommendation/collaborative/summary.

[8] Schedl, Markus. "The LFM-1b Dataset for Music Retrieval and Recommendation." *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, Association for Computing Machinery, New York, NY, USA, 2016.

[9] Reddy, M. Sunitha, and T. Adilakshmi. "Music recommendation system based on matrix factorization technique-SVD." *2014 International Conference on Computer Communication and Informatics.* IEEE, 2014.

[10] Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization." *Advances in neural information processing systems* 20 (2007): 1257-1264.

[11] Koren, Yehuda. "Factor in the neighbors: Scalable and accurate collaborative filtering." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4.1 (2010): 1-24.

[12] Hug, Nicolas. "Surprise: A Python library for recommender systems." *The Open Journal, Journal of Open Source Software,* 5.52 (2020).

9 Quotewall

"do we even need the matrix? can we just use user_ids?"

"for loop is the solution to everything"

"our project idea centers around the idea . . . "

"waiting for your model to run is like watching paint dry"

"why is this project such a train wreck?" // "it's ok we're in the train wreck together"