MyDJ: A Personal Curated Song Recommendation System

Ibrahim Ahmed
ahmedibr@msu.edu

Aasiruddin Walajahi
walajahi@msu.edu

Ikechukwu Uchendu
uchendui@msu.edu

***Abstract*— We propose a music recommendation system called *MyDJ*. *MyDJ* implements a web interface that collects general information about a user and their artist preferences. Using this data, a factorization machine model predicts artists that a user would most likely listen to. Users are able to view this information on the website and update their artist preferences for more accurate predictions.**

I. INTRODUCTION

Today's music industry is driven by streaming services, spearheaded by the cord-cutter culture. Music streaming services are competing to offer the largest variety and amount of music to maximize consumer listening time. A large portion of music consumption is dominated by streaming services (*Spotify, Apple, Google*), all of whom are driven by proprietary recommendation systems. Music streaming revenue in 2018 is estimated to be over \$9.8b USD[1]. As such, it is imperative that each music platform retains its user base with a continuous stream of new music recommendations that caters to the users' preferences.

Recommendation systems have grown in recent years for two main reasons: the abundance of data available and the increase in computational power. These developments have ushered the technology industry into an age of big data where predicting preferences is an achievable goal.

II. RELATED WORK

A. Spotify Recommendation System

Spotify uses a combination of three different data mining techniques when recommending songs, as outlined in [2]

1) Collaborative Filtering: Since *Spotify* persists all user data, they can create robust user-item collaborative filtering models which take user information and *activity* into account.

2) Natural Language Processing: Spotify crawls the internet searching for information about songs or artists in various sources such as blog posts, testimonials, and reviews. *Spotify* uses Natural Language Processing (NLP) to analyze all text collected about a song or artist to compare them to other songs or artists in *Spotify's* library. These similarity measures can then be used to recommend songs to a user.

3) Audio Models: When a new song is released, there is not much user data available in *Spotify's* database. To circumvent this lack of data, *Spotify* utilizes audio based models to compare songs. Using Convolutional Neural Networks (CNN), *Spotify* extracts characteristics about songs such as their estimated time signature, key, mode, tempo, and loudness. Using these characteristics, *Spotify* recommends songs to users based on their listening history.

B. The Music Genome Project (Pandora)

This was an attempt by Pandora to develop a system that is able to "capture the essence of music at the most fundamental level" [3]. The system used 450 attributes, or "genes", to describe the characteristics of a song. Each song is initially separated into 5 sub-categories: Pop/Rock, Hip-Hop/Electronica, Jazz, World Music, and Classical. Each of the subcategories have a different number of attributes associated from the 450. Unlike *Spotify's* recommendation system where the songs are characterized using a computerized algorithm, *Pandora* uses expert musicians to assign attributes to each song. Upon receiving a set of song vectors, Pandora's matching algorithm returns a list of similar songs.

III. PROBLEM STATEMENT

Recommendation systems are notorious for the cold start problem: the system cannot make reliable inferences for new users. To tackle this issue, we gather information about users and the artists they enjoy listening to. This gives us enough information to serve them with recommendations based on similar users in our data set.

To develop a music recommender system, we had to analyze a real music listening data set consisting of thousands of users. We use the *Last.fm* 360K users data set which provides us with the personal information of each user along with their corresponding listening history. Using this data set, we train a recommendation system to determine potential artists a user might like.

IV. METHODOLOGY

A. Data Collection

Our training data is comprised of the *Last.fm* 360K users data set [1]. It supplied us with a 543MB CSV file containing 17 million rows of listening information for 360,000 users on the *Last.fm* platform. One thing to note is that this data set

[1]https://www.upf.edu/web/mtg/lastfm360k

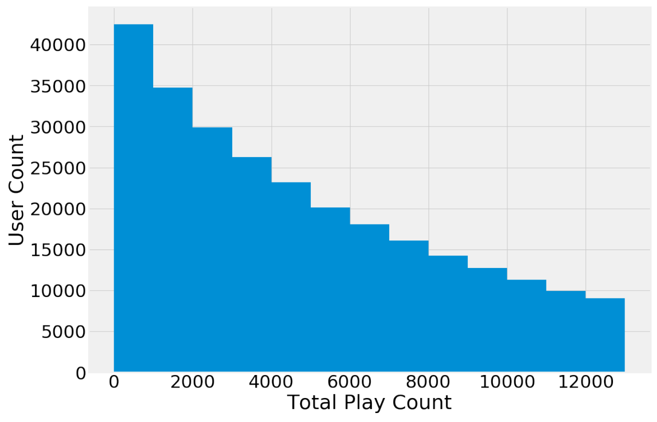


Fig. 1. Histogram of number of plays for each user. For the majority of the users, we notice a play count between 0 and 1000.

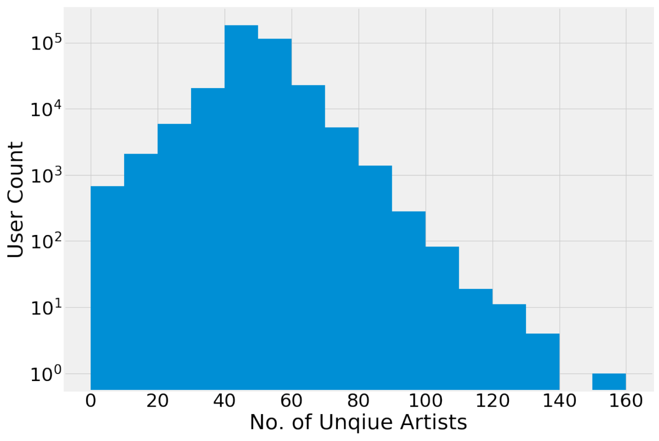


Fig. 2. Histogram of the unique number of artists a user has listened to. For the majority of users, they listened to about 40-60 unique artists.

does not contain information about individual songs. Each row of the data set contains the follow information:

- User ID
- User Gender
- User Age
- User Country
- User Signup Date
- Artist ID
- Artist name
- Total play (No. of times the user played the artist)

Upon closer inspection of the data, we notice some key characteristics about the *Last.fm* data set. There are a handful of artists dominating the *Last.fm* data set with the highest user listen count belonging to the English Rock band *Radiohead* at 77,254.

Figure 1 shows a histogram of the number of plays for each user. In the figure, the x-axis denotes bins that group users based on their total play count. The y-axis plots the number of users who fall in the corresponding play count bin. The histogram is segmented into 14 buckets, each with a range of 1,000. The highest play count in the data set for a user is 787,884 while the lowest is 1. As expected, a minuscule amount of users have over 12,000 total plays while the majority of users have between 0 and 2,000 plays.

Figure 2 shows us the distribution of the number of unique artists listened to by each user. In the figure, the x-axis denotes the bins that groups users based on the number of unique artists they listened to. The y-axis plots the number of users who fall within the corresponding unique artist bin. The histogram is segmented into 17 buckets, each with a range of 10. The highest unique artist count is 166 while the lowest is 1. The average number of unique artists listened to across the data set is 49. We notice that the majority of users listen to between 40-50 artists followed by 50-60 artists.

B. Data Preprocessing

The data set contained some entries with null values due to very nature of the *Last.fm* platform. We removed these entries to ensure data integrity. Also, due to the quantity of data available, rows with any missing values were removed. After removing the users with null values, we still had data from 358868 users to train with. Finally, total plays and age were centered.

C. Factorization Machine

A Factorization Machine (FM) model is a general predictor for any real-valued feature vector. [4] The power of FMs lie in the fact that they model all interactions between features using factorized parameters. [4] Modelling these interactions allows FMs to estimate interactions in problems with sparse data, such as our recommendation system.

1) Model: Rendle [4] defines the model equation for a factorization machine of degree $d = 2$ as:

$$\hat{y} := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

where the model parameters that have to be estimated are:

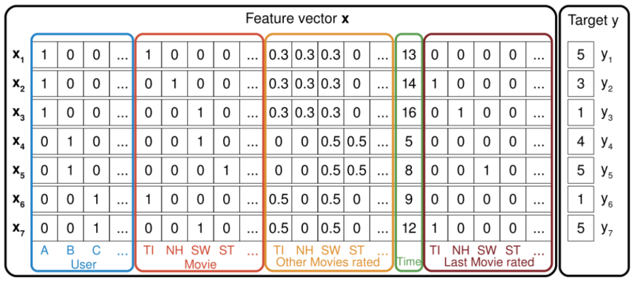
$$w_0 \in R, \ \mathbf{w} \in R^n, \ V \in R^{n \times k}$$

and $\langle v_i, v_j \rangle$ is the product of two vectors of size k.

In our problem, $\hat{y}$ denotes the number of times a user is predicted to play a certain artist.

2) Feature Matrix: Our model utilized various user features to increase the accuracy of our predictions. We collected the user's age, country, gender, and artist played for features, and artist total play count as the ground truth. To support categorical variables in our model, features such as user ID, country, and artist name were turned into one-hot vectors, similar to Figure 3 .

3) Training: We used the TensorFlow Factorization Machine module (TFFM) [5] to implement a second-order FM. TFFM provided us with an API to create, train, and save an FM. The FM was trained via Adam with a learning rate of



Feature vector **x**	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		TI	NH	SW	ST	...		
x_1	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	y_1
x_2	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	y_2
x_3	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	y_3
x_4	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	y_4
x_5	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	y_5
x_6	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	y_6
x_7	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	y_7

Fig. 3. Example data set from Rendle [4] that depicts sparse, real valued feature vectors. In our problem, we have a similar set up except that *artist* is in the place of *movie*. Instead of *other movies rated*, *time*, and *last movie rated*, we have *age*, *country*, and *gender*.

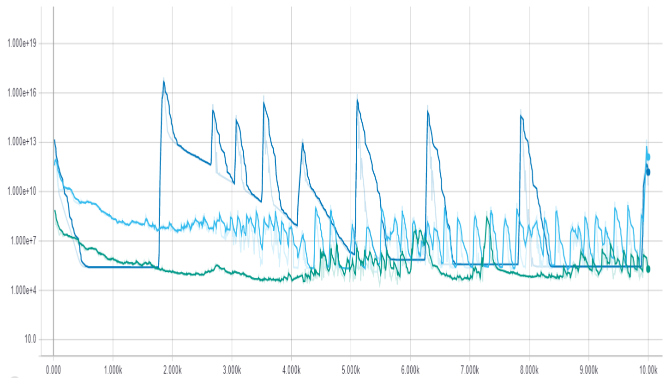
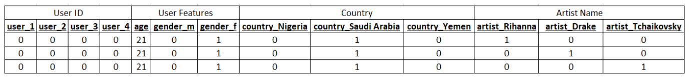


Fig. 4. Graph of the loss function for learning rates 0.1 (dark blue), 0.01 (light blue), and 0.001 (green). The horizontal axis represents the number of epochs, and the vertical axis represents the log-scaled mean squared error.

0.001 for $10,000$ iterations. Figure IV-C.3 depicts a training graph that experiments with various hyperparameters.

All training was conducted on the MSU Engineering Compute servers. These machines have 32-core CPUs and 756 GB of shared RAM. Unfortunately, due to resource competition, we were not able to harness enough compute to train on more than 50,000 rows of our data set. Once trained, we saved the model, top artists from our 50k subset, and the column mapping for our sparse data matrix. These files are used on the website and Google App Engine API for inference.

4) Inference: Given initial user features, we construct a matrix similar to Figure 5. Each column of user features is the same since we are predicting artist play count (artist preferences) for a unique user. Feature vectors for every top artist in the data set are created. We only predict play counts for the top n artists because many artists have a minuscule number of plays. Using the predicted plays, we are able to return the top n artists that a user would most likely listen to by sorting the output.



User ID				User Features			Country			Artist Name		
user_1	user_2	user_3	user_4	age	gender_m	gender_f	country_Nigeria	country_Saudi Arabia	country_Yemen	artist_Rihanna	artist_Drake	artist_Tchaikovsky
0	0	0	0	21	0	1	0	1	0	1	0	0
0	0	0	0	21	0	1	0	1	0	0	1	0
0	0	0	0	21	0	1	0	1	0	0	0	1

Fig. 5. Example feature matrix built for inference. This user is a 21 year old female from Saudi Arabia.

D. Web Application

The *MyDJ Music Player* can be accessed here.

MyDJ Music Player was built using Angular 7 and is hosted on Google Firebase. The web application is a single page application (SPA) and utilizes JavaScript to manipulate the page.

MyDJ Music Player consists of three major pages:

- Sign-up page - The sign-up page collects the users name, age, gender, and country. This data is used when making song recommendations.
- Artist selection page - The artist selection page is where the user selects their favorite artists. This list of favorite artists is fed into the recommendation model.
- Music Player page - The music player page is where the recommendations made to the user are realized. The website gets a list of recommended artists from the backend API and begins playing songs from these artists in an endless playlist on this page.

The *MyDJ Music Player* utilizes the Napster API to stream songs by the artists that are recommended to the user. The recommended artists are returned by a Flask application hosted on the Google App Engine Web API. The Flask application uses a cached version of the trained model to make predictions.

All data entered into the *MyDJ Music Player* are stored in the users browser so that they can return to the application if their window closes or their computer reboots.

V. EXPERIMENTAL EVALUATION

A. Software Utilized

The project consists of two main parts:

1) Music Recommendation System: The Recommendation System is written completely in Python. We utilize the scikit-learn and TensorFlow machine learning libraries along with NumPy and Pandas for data manipulation. The application backend API is developed with the Flask Framework in Python. The trained model and REST API are hosted on Google App Engine.

2) MyDJ Music Player: MyDJ Music Player is built using Angular 7 and is hosted on Google Firebase. The *MyDJ Music Player* utilizes the Napster API to stream songs that are recommended to the user.

B. Evaluation Measure & Results

The most basic evaluation measure for our FM model was root mean square error (RMSE). After training for 10,000 iterations, our RMSE was 2,333.07 for the training set and 3,107.23 for the testing set. Note that these RMSE values represent the error in how many times a user is predicted to play an artist.

We devised a more accurate measure using an assumption about our data. After centering the artist play count ground truth column, we assume that a user favors an artist if the total play count is greater than or equal to zero. We then predict the play count for every user in our testing set. If the prediction and the ground truth label have the same sign ($\pm$), we consider that a correct prediction. The count of correct predictions divided by the number of rows in our testing set returns a percent accuracy. This process is depicted in Algorithm 1. Of course, this method is still flawed because we do not have data about artists that users do not like. In other words, a negative play count does not make sense in reality. After 10,000 iterations, The training accuracy was 56.18%, and the testing accuracy was 41.42%.

Algorithm 1: Calculating the percent accuracy of the Factorization Machine Model.

```
  Input  : Testing data set $X$
1 $score = 0$
2 foreach feature row $i$ in $X$ do
3     Let $p$ be the predicted play count for row $i$
4     Let $c$ be the ground truth play count for row $i$
5     if $p$ has the same sign as $c$ then
6         $score = score + 1$
7 end
8 return $score/|X|$
```

VI. Future Work

A. Using Song Data Instead of Artist Data

In *MyDJ*, the factorization machine utilizes user listening history based on artists to make recommendations. The challenge that comes with using artist data instead of song data is that there is a loss of information when training our models. Users listen to fewer unique artists than they do unique songs which results in a skew towards specific artists. If song data is used instead, users' listening preferences would be much more apparent, and the number of samples would be much higher.

In addition to providing a larger training set, training on song data instead of artist data means song recommendations can be made instead of artist recommendations. Artists usually create a plethora of music which can span many different genres. Therefore, recommendations made on the artist level may not always satisfy the user since *MyDJ* chooses a random song to play when selecting from the predicted artists. If songs were recommended instead, they would have a higher chance of satisfying the users taste because they are chosen with higher confidence than when choosing a random song from a predicted artist.

B. Retraining Models Based On User Feedback

The *MyDJ Music Player* provides a method to collect feedback from a user when playing a recommended artist in the form of two buttons: a ”thumb up” button and a ”thumb down” button. By utilizing the feedback received by these buttons, the model can be improved by penalizing recommendations which have too many ”thumb down” ratings. This allows for continuous improvement of the model, and can be used to generate test sets for evaluating model performance.

C. Persistent User Database

Currently, *MyDJ* does not store user data. Instead, it stores all user information and artist preferences in the browsers localstorage which does not persist to any database. This means that once a user logs out of their session, all of their data is lost. Instead of persisting user data to localstorage, a database solution such as *MariaDB* or *PostgreSQL* could be used for long term storage of user accounts.

By using a long term storage mechanism to store user data, more information can be collected based on users listening activity which can subsequently be fed into the recommendation model. Having a user sign in method that is persisted provides the potential for more features to be offered in the *MyDJ Music Player* such as

- Allowing users to create custom playlists to store songs they like
- Recording user playback history to feed into the original model
- Recording playback time per-song per-user to measure user satisfaction per song
- Preserve user preferences to avoid having to regenerate recommendations for returning users

Extending the *MyDJ Music Player* to use a persistent database would be simple given the modular nature of the codebase.

VII. Conclusions

Music has been a part of humanity for generations. It is something that humans listen to during moments of elation, depression, motivation, and celebration. This, in combination with the increase in the ease of access of internet across the world, make music streaming a lucrative industry. The Music streaming industry is currently one of the largest entertainment industries in the world and has the potential to grow exponentially as many underdeveloped countries increase internet access. Estimated revenue worldwide is predicted to grow approximately 3 billion dollars over the next five years. As a result, there has been a sharp increase in recommendation systems research.

This project successfully developed a recommendation model utilizing a factorization machine model that learns relationships between user information and artist preferences. The *Last.fm* 360K user dataset was used to train our model. An accompanying web interface was developed that allows users to provide their personal information and artist preferences, which is then sent to our model that predicts artist recommendations. Once the web interface receives predicted artists, it streams an endless playlist of music based on them.

ACKNOWLEDGMENT

We would like to thank Dr. Pang-Ning Tan for his counsel and support throughout the course of this project. We would also like to thank Google for providing free credits to the Google Cloud Platform.

APPENDIX

The following is a list of links to datasets used in *MyDJ*:

- *MyDJ* Github Repo containing the entire website codebase, several iPython Notebooks, and copy of the final report documents
- 50K Users random sample state set to 1234 as used in the Last.FM 360k Factorization Machine.ipynb notebook
- Lst.fm 360k dataset

REFERENCES

[1] Statista. Music streaming - worldwide — statista market forecast, 2018.
[2] Sophia Ciocca. How does spotify know you so well?, 2018.
[3] Wikipedia. Music genome project, 2018.
[4] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.
[5] Alexander Novikov Mikhail Trofimov. tffm: Tensorflow implementation of an arbitrary order factorization machine. https://github.com/geffy/tffm, 2016.