A Comprehensive Recommendation For Music System

Zheyao Gao, Jinyang Li
Department of Electrical Engineering
Columbia University in the City of New York, U.S.A
zg2308@columbia.edu, jl5173@columbia.edu

Abstract—**With the rapid development of the era of big data, information is gradually showing an overload state. As one of the most effective methods to implement the balance of interests between information producers and consumers in recent years, the recommendation system (also known as personalized content distribution) play more and more important roles in recent years. According to estimates by Wall Street analysts, the purchase Conversion Rate[1] of Amazon can be as high as 60%. Therefore, exploring how to implement and improve a personalized and comprehensive recommendation is more meaningful.**
(Abstract)

Key words: Recommendatio, Conversion Rate, Personalized, Comprehensive.

I. INTRODUCTION

Because of fast development of the technology and exploration of the big data, more and more Internet companies, such as Facebook, Google, Amazon, are focusing on the advancement of recommendation systems, which can help them appeal to more users and make more profits. In this case, our team aims to construct a comprehensive recommendation systems based on Collaborative Filtering Algorithms and Content Based Algorithms. Besides, in this project, we also introduce a more efficient and innovative way to recommend users with songs by RGB features.

II. RELATED WORKS

Currently, there are two main recommendation algorithms. The most famous and practical method is Collaborative Filtering.
1.Collaborative Filtering is a method of automatically predicting (filtering) a user's interest by collecting preference or taste information from multiple users (collaboration). The basic assumption of the collaborative filtering approach is that if a person A has the same opinion as person B on a certain issue, then A is more likely to have a B opinion on another issue than a randomly selected people[2].
Besides, there are also additional ways to recommend. In our project, we chose Content Based Filtering (based on Cosine Similarity) to do recommendations, and Amazon, Walmart recommend users with their products in this way as well:
In a content-based recommendation system, it is used to describe items, and user profiles are used to indicate the kinds of items that the user likes. In other words, these algorithms attempt to recommend items that are similar to the items that the user liked (or are currently checking). Currently, most companies choose some very common and general music information as input features of each song. For example, they will take year, release, artist as the main features of song. Although, this indeed can catch users' interests somehow, we also want to explore more than general method that can bring us and users more surprises. In this case, in our project, we would like to utilize characteristics of songs, not of information again. Then we introduce timbre here to be selected as our features of songs:

2.Timbre:
Timbre is the quality of a musical note or sound that distinguishes different types of sound production or musical instruments. Timbre is one of the basic elements of music. In timbre listening, focus is on texture and color of music and sound rather than perceiving pitch and harmony[3].
Human ears possess an ability to distinguish musical colors. We can easily distinguish the sound of piano from the sound of guitar because they have different feeling or color in their sound. When required training is processed, human ears can extract a certain instrument using the knowledge of timbre of the sound of the specific instrument. Various audio features are used in recent researches in order to achieve automatic timbre recognition system using machine learning algorithms. Recognizing audio features can be a challenging problem since it is a continuous task unlike discrete data, image or text recognition. Acquisition and recognition need to be in sync of time frame in order to achieve correct predictions. In order to achieve a correct set up for the recognition, we need to explore more about the musical characteristics, timbre[4].
In a nutshell, in our project, we would take two main important timbre attributes as our features of songs (Timbre Average and Timbre Covariance). That may catch customers' hidden interests and tastes more than we can imagine and may represent styles of songs in some angles. Therefore, we calculate similarities between songs based on their timbre, which was extracted them into numbers that our software package can deal with.

III. SYSTEM OVERVIEW

1.Describing dataset:
There are two csv files that are used in our project:

1.1 Ratings.csv :
This csv file contains 12053779 rows of records, which occupies 869.9MB in our server. In each record, it provides

three fields of information: IDs of users, IDs of songs, and how many times the specific users played this the specific songs. We used this file to do Collaborative Filtering, which is the first phase of our project.

1.2 Song_features.csv:
This csv file contains 9839 rows of records, which occupies 8.2 MB in our server. In each record, it provides the information about the timbre of each songs. We used this file to construct Cosine similarity matrix between different songs and apply them with RGB features in order to show the similar colors of songs with similar styles(timbre).

2.System design idea:
As presented in the Figure 1, we first recommend different users with several songs that they may like potentially based on Collaborative Filtering. Then for each recommended song, we will also provide users with other similar songs. In our system, if they are satisfied with some of song that we recommend them and click on those songs, they can also select similar songs depending on the colors of songs besides our recommendations. Then, in the next time, our users can jump into songs with their favorite types quickly and efficiently. That can improve experience of users to our applications somehow.

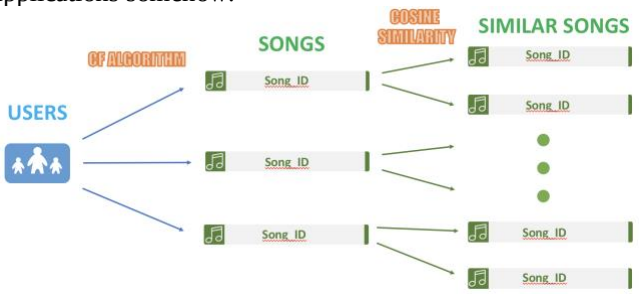

Figure 1

IV. ALGORITHM

These are algorithms that we used in our project:
1. Collaborative Filtering:
The collaborative filtering approach is based on collecting and analyzing a large amount of information about user behavior, activities or preferences, and predicting what the user likes based on the similarity of the user to other users
In the PySpark, ALS package is one way to do Collaborative Filtering, which was implemented by SVD Algorithms:

1.1 SVD algorithm:
In the Figure 2, this is an extremely sparse matrix. We record the score matrix as R. The elements in this matrix represent user's score on item. The question is: how to predict the unknown score?

User /item	1	2	3	4	5	.....
1	5	4	4.5	?	3.9	.....
2	?	4.5	?	4.5	?	.....
3	4.5	?	4.4	4	4	.....
4	?	4.8	?	?	4.5	.....
5	4	?	4.5	5	?	.....

Figure 2

For any matrix A, it has its full rank decomposition (as shown in the Figure 3)

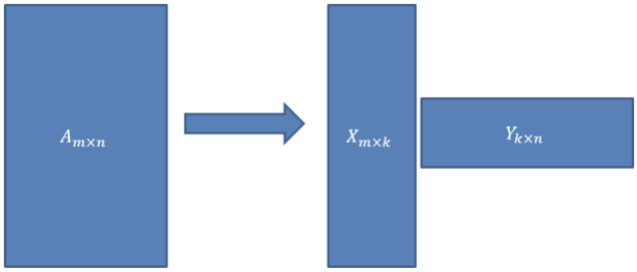

Figure 3

The rating matrix R (shown in the Figure 2) also has such a decomposition, so the rating matrix R can be expressed by the product of two matrices P and Q.

The U in the formula above represents the number of users and the I represents the number of products. Then the known score in R is used to train P and Q so that the result of multiplying P and Q can best fit the known ratings. Finally, the unknown ratings can be obtained through multiplying a row of P by a column of Q.

$$\hat{r}_{ui} = p_u^T q_i$$

2. PCA (Principle Components Analysis):
PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on[5]. The transformation is defined by a set of p-dimensional vectors of weights or coefficients that maps each row of data X to a new vector of principal component scores given by:

$$t_{k(i)} = \boldsymbol{x_i} * \boldsymbol{w_k}$$

In order to maximize variance, the first weight vector $\boldsymbol{w_1}$ has to satisfy:

$$\boldsymbol{w_1} = \arg\max\{\frac{\boldsymbol{w^T X^T X w}}{\boldsymbol{w^T w}}\}$$

The kth component can be found by subtracting the first k-1 principal components from X:

$$\hat{\boldsymbol{X}}_k = \boldsymbol{X} - \sum_{s=1}^{k-1} \boldsymbol{X w_s w_s^T}$$

To do dimensionality reduction, we only have to use first k row of **W:**

$$\mathbf{T_L} = \boldsymbol{XW_L}$$

where the matrix TL now has n rows but only L columns. In other words, PCA learns a linear transformation where the columns of p × L matrix W form an orthogonal basis for the L features (the components of representation t) that are decorrelated[6].

3. Timbre extraction

The musical signal is cut into frames(50ms), and for each frames we compute the shot-time spectrum. With these spectrums, we then calculate Mel Frequency Cepstrum Coefficients[7]

$$c_n = \frac{1}{2\pi}\int \log\left(S\left(e^{jw}\right)\right)e^{jwn}dw$$

These calculated coefficients change from frame to frame. The changes in these values can be plotted as an envelope across the sound. These envelopes are distinctive to the instrument[8] as illustrated below:

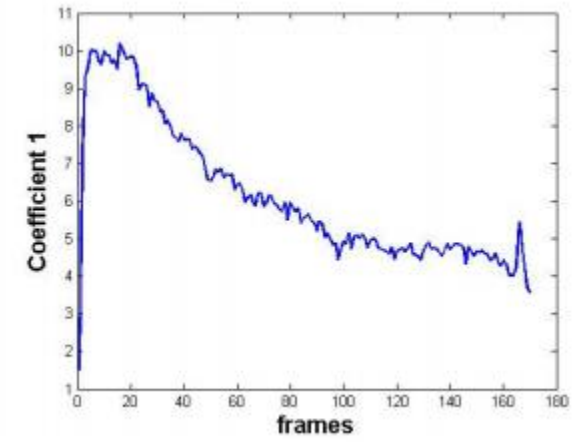

Figure 4

(Trend of the first MFFC on a piano)

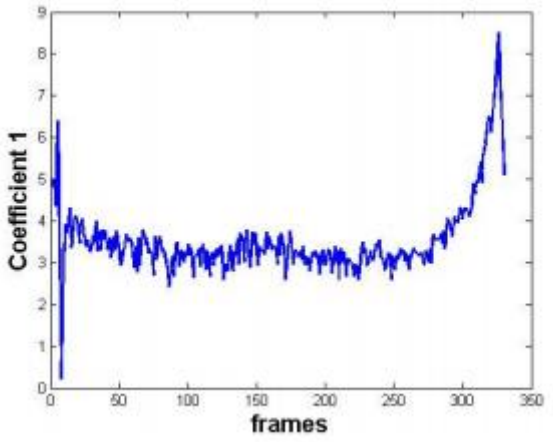

Figure 5

(Trend of the first MFFC on a flute)

In our project, we use the mean and covariance of MFFC as features to distinguish between songs

4. **Cosine similarity：**

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} * \mathbf{B}}{||\boldsymbol{A}|| * ||\boldsymbol{B}||}$$

It is thus a judgment of orientation and not magnitude. Since the feature vector in our project is RGB values, difference in direction turns out to be difference between colors. In this way, songs with similar colors will be more close to each other by using cosine similarity to measure the closeness of two songs. And that means songs with similar timbre features will obtain a higher similarity score.

V. Software Package Description

Our project consists of two parts: recommendation engine and web server.

For the recommendation engine part, we don't write api for the web server to update data in database automatically. To recommend songs for user with new data, you should get new play count data from web server, and use csv file (CF_recommendation.csv, Similarity.csv) for this als.py to generate new recommendations.

For the web server part, we save the recommendation data to the mysql workbench, and write api function for the web to fetch data from database. You only have to connect to the internet and run api.py, the all computers connected to the same internet can access through private network IP shown in the output of api.py. And our Front-end page was written by HTML5, CSS and JavaScript.

In this case, you will enter this Login Interface:

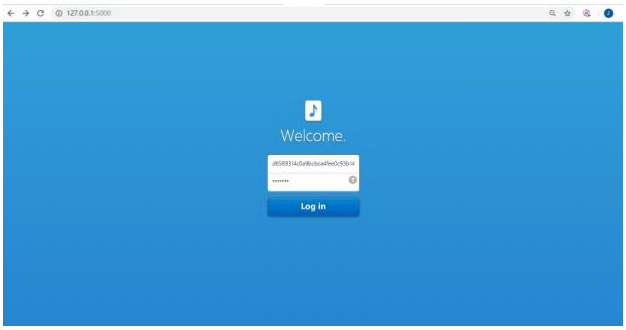

Figure 6

Users then can log in our website through inputting user name and password.

After logging in our application, as shown in the Figure 7, users will receive some hottest songs (sorted by number of play_counts) in the left side, and also receive songs that we recommend them based on CF Algorithms in the right side (There are several songs that they may like potentially according to their previous information about using this application.

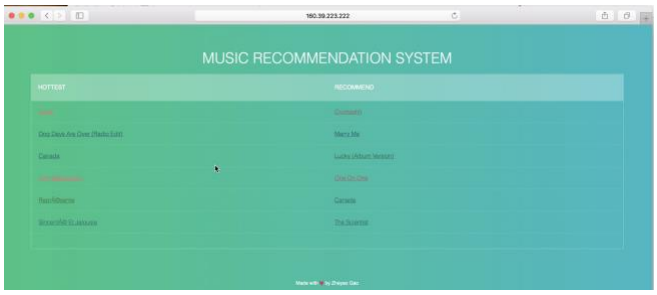

Figure 7

When users are interested in some of songs that we recommend them, they can also find other similar songs for this song. That is presented in the Figure 8.

Figure 8

Because we applied each song with RGB features for songs with similar timbre style and visualize them in the page, users can find their favorite style of music through their eyes and jump into that more quickly.

VI. Experiment Results

As shown in Figure 1, our project is made up by two parts. First part that is called 'Recommendations' in our project first page is implemented by CF Algorithm. Second part is finished by Cosine Similarity.

1.Recommend part (as shown in Figure x):

In the first part, since recommendation experiment can be considered as unsupervised algorithms, our metric of result is RMSE. We split our original dataset into train and test dataset. And our model yield RMSE result from testing dataset. As shown in the figure x, the RMSE is 12.95, when the maxIter is 10, which means the maxium of iterations is 10, and regParam is 0.3, which means we try to reduce phenomenon of overfitting here. Compared with the range of play_counts, which is (0, 9667), the RMSE result is satisfying.

```
# Build the recommendation model using ALS on the training data

als = ALS(maxIter=10, regParam=0.3, userCol="userId_digtal", itemCol="songId_digtal", ratingCol="pla
          coldStartStrategy="drop")
model = als.fit(training)
```

```
# Make predictions and output a RMSE to check the precision
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="play_count",
                                predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 12.952059997398477
```

Figure 9

2.Similar Songs We found for you:

In this page (as shown in the Figure 10), if our users are satisfied with songs that we recommend them by CF Algorithm and clicked on them, we will also recommend other songs with similar styles to users depended on cosine similarity. The Figure x is cosine similarity matrix of songs based on their timbre features. We show top 15 most similar songs to each song here for clearer presentation here.

In this part, we will only choose first 10 songs (sorted by the value of similarity shown in the matrix) as top 10 similar songs for the song that our users are interested in.

3.Besides, there are reasons for choosing top 6 songs in the first part of our application and top 10 similar songs in the second part to users:

3.1: The dataset is very big and the memory of our server is limited. Showing several songs recommended for users will be more efficiently.

3.2 Because we visualized our songs with colors based on timbre features. Showing too many songs with different colors in one page will cause users to be confused very much.

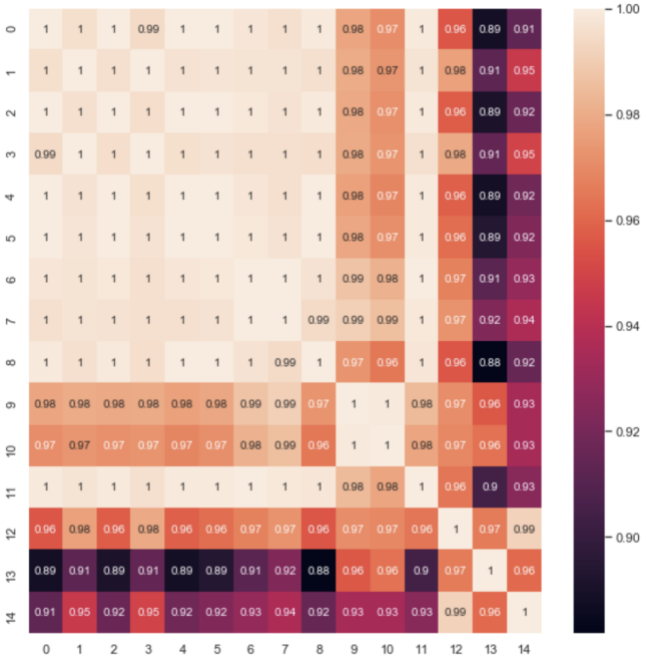

Figure 10

VII. Conclusion

1. When applying ALS package in the PySpark, we can reduce RMSE and improve the precison of our models through increasing values of maxIter. And we can prevent overfitting by increasing value of regParam.

2. Through PCA algorithm, we can project original features of songs into three dimensions first, and normalize values in each dimension into the range (0, 255) for convenience of showing colors of each songs based on their styles.

3. Our RGB features not only can be visualized for users to identify their favorite styles of songs, if users think too colorful webpages are confused, this RGB features can also be utilized to help back-end engineers to analyze some data problems or check the precision of classification somehow. Because more intuitive features always help data analyst or engineers to solve problems or find something new.

4. Division of Work:
In this project, our team is combined with two students: Zheyao Gao and Jinyang Li. And this is our division of Work during project:
Zheyao Gao: (50%)

1. In charge of Front-end and Back-end buildings
2. Data cleaning and pre-processing
3. Training the model and analysis

Jinyang Li: (50%)

1. In charge of Front-end and Back-end buildings
2. Data cleaning and pre-processing
3. Training the model and analysis

5. Future Work

5.1 We will fetch the dataset about lyrics of songs and provide each song with one topic based on LDA (Latent Dirichlet Allocation) analysis.

5.2 Given that people have different moods in different days, we will also introduce sentiment analysis through their tweets (by Data Streaming). To implement that function, we will add labels (Sad or Happy) into our new dataset about Twitter first. Then filtered out or add songs in the list of songs that we recommend our users in the page.

5.3 Sometimes, our users may be confused and overwhelmed by colors we applied with each song. Then we could design our UI better to make our pages cleaner. Or we can set Threshold of values of RGB feature after PCA,then cluster them and add labels instead of setting colors

ACKNOWLEDGMENT

WE SHOULD THANK PROFESSOR LIN FIRST. ALL OF KNOWLEDGE WE APPLIED IN OUR OBJECT HAVE BEEN ABSORBED THROUGH PROFESSOR LIN'S LECTURE. AND PROFESSOR LIN ALWAYS GIVE THE FEEDBACK AS QUICKLY AS HE CAN IN THE piazza THAT DID HELP OUR HOMEWORK AND UNDERSTAND OF SOME MAIN CONCEPTS IN THE BIG DATA AND MACHINE LEARNING. BESIDES, WE ALSO SHOULD APPRECIATE OF WONDERFUL COMMENTS AND SUGGESTIONS OF DOCTOR VISHAL ANAND IN TWO PRESENTATIONS. DOCTOR ANAND'S ADVISES HELPED US UNDERSTAND OUR MEANINGS AND USEFULNESS OF OUR PROJECT VERY MUCH. FINALLY, WE VERY APPRECIATED EVERY COMPREHENSIVE TUTORIAL ABOUT HOMEWORK IN THE LECTURES AS WELL. BECAUSE OF THESE DETAILED TUTORIAL DOCUMENTS, WE CAN FINISH PROJECT AND OPERATE MAINSTREAM ALGORITHMS OF MACHINE LEARNING AND BIG DATA VERY SMOOTHLY. THANK YOU FOR YOUR HELP!

REFERENCES

[1] Dvir, Nim; Gafni, Ruti (2018). "When Less Is More: Empirical Study of the Relation Between Consumer Behavior and Information Provision on Commercial Landing Pages". Informing Science: The International Journal of an Emerging Transdiscipline. 21: 019–039. doi:10.28945/4015

[2] An integrated approach to TV & VOD Recommendations Archived 6 June 2012 at the Wayback Machine.

[3] Acoustical Society of America Standards Secretariat (1994). "Acoustical Terminology ANSI S1.1–1994 (ASA 111-1994)". American National Standard. ANSI / Acoustical Society of America.

[4] Sang Hyun Park," Musical Instrument Extraction through Timbre Classification", NVIDIA Corporation Santa Clara, CA 95050

[5] Olliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

[6] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[7] J. -. Aucouturier, F. Pachet and M. Sandler, ""The way it Sounds": timbre models for analysis and retrieval of music signals," in IEEE Transactions on Multimedia, vol. 7, no. 6, pp. 1028-1035, Dec. 2005.

[8] Loughran, Roisin & Walker, Jacqueline & O'Neill, Michael & O'Farrell, Marion. (2008). The Use of Mel-frequency Cepstral Coefficients in Musical Instrument Identification.

[9] D. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

[10] H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representatives," Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670.