Music Recommender System

Dhivya Govindarajan
dng2378@rit.edu

Yogesh Jagadeesan
yj6026@rit.edu

Nitish Ganesan
nkg8933@rit.edu

Pradeep Kumar
pd3792@rit.edu

ABSTRACT

The ever increasing size of data is being tremendously used by industries in a variety of useful ways. One such data that is too hard to ignore is the one from songs and albums. With their enormous number and a lot of complex attributes like beats and energy associated with them, music industries play a significant role in mining out useful information that would best benefit the listeners. Most listeners find it difficult to search for new tracks/albums that would suit their tastes as the numbers are just too many. In addition, they would have to keep track of every album or song being released if they ever need to improve their playlists. With the advent of big data and analytics, key attributes from the songs dataset can be mined to figure out patterns and suggest tracks that the listeners might like. As the user chooses to listen to or skip a song, the system will iteratively learn and improve suggestions. Over time, if enough users contribute to the learning of the system, it would attempt to suggest random songs based more on other users' preferences and less on the attributes of the soundtracks themselves. In this way, the system would continually learn and improve, thus constantly evolving into a more sophisticated recommender system that would make the lives of avid listeners a lot easier.

1. OVERVIEW

The Music recommendation system enables users to listen to music tracks that fit their tastes. The system allows users to login with their credentials, displays the music recommendations based on their tracks history and enhances their experience. The user's listening patterns are learned based on which the system recommends tracks that are similar to their favorite tracks. The system initially recommends tracks that are not often heard by the user before and progressively improves suggestions.

2. DESIGN CONSIDERATIONS

The following are the associated tables and the functional dependencies:

Song (song_id, song_title, song_genre, duration, play_count, language, cluster_id)
song_id ⇒ song_title
song_id ⇒ song_genre
song_id ⇒ cluster_id
song_id ⇒ duration
song_id ⇒ language
song_id ⇒ playcount

System_User(user_id, first_name, last_name, age, gender, native_language, password)
user_id ⇒ first_name
user_id ⇒ last_name
user_id ⇒ password
user_id ⇒ age
user_id ⇒ native_language
user_id ⇒ gender

Artist(artist_id, artist_name, popularity, artist_genre)
artist_id ⇒ artist_name
artist_id ⇒ popularity
artist_id ⇒ artist_genre

Instrument(instrument_id, instrument_name)
instrument_id ⇒instrument_name

User_to_Song(song_id, user_id, cluster_id)

Artist_to_song(artist_id, song_id)

Instrument_to_song(song_id, instrument_id)

User_to_Instrument(user_id, instrument_id)

The above mentioned tables are loaded in Oracle Database. The Song table comprises the properties of the song and contains unique entries. The User_System table has only the user data and login credentials. The Artist table contains the artists' details and the popularity. The relationship between the Artist and the Song is depicted through Artist_to_Song which is one-to-many relationship. The User _to_Song relationship table is updated whenever the user likes a song. The Instrument table comprises of collective and unique information of all the instruments from all the songs. As a song can be associated with many instruments, the entries are recorded in the Instrument_to_Song table.

Users' preferred instruments are in the User_to_Instrument table.

Initially, the following tables were present: Song, System_User, and Tasks. The Tasks table included the records of the songs played by the user. The tables were not in 1NF due to the presence of multi-valued attribute, Instruments and not all non-key attributes were dependent on the keys. Individual artists were identifed and populated in the Artists table. All unique values of the Instruments were extracted and copied to the Instruments table. The relationship tables, User_To_Instrument, Instrument_To_Song were created to record the instruments used in songs and also the instruments preferred by users, eliminating the need for the multi valued attribute. The attributes dependent on the key attributes were identified and additional tables, such as Artist_To_Song and User_To_Song were created and populated. The schema was hence brought to 2NF. Since there are no transitive dependencies, it is also in 3NF.

The CSV files from which data were to be populated were preprocessed. Unnecessary fields which wouldn't contribute to the application or mining were eliminated. Also, the application itself needed a separate user table that should include their login credentials and the songs they have listened/will listen to. So unique users were extracted and were populated in 'system_user' table. The tables that were created as a result of normalization were populated as well from the existing CSV data. For instance, the 'user_to_song' table needed information extracted from both the song table and the user table thus linking which user listened to which song. The 'Instrument_to_song' table required linking to be done from songs to instruments in order to determine what song used which instruments. Likewise, the table 'User_to_instrument' shows which users prefer what kind of instruments. The 'Artist_To_Song' table linked artists to songs even if they composed multiple songs. Figure 2 and Figure 3 shows the queries used and the entity relationship diagram.

3. ARCHITECTURE

The tables are loaded into the Oracle database while retaining all the constraints. The clustering is done using Weka Mining tool. The system features an intuitive user interface that allows users to login to access their personalized soundtracks. This was done using HTML, PHP and CSS. It displays a list of suggested and random tracks which gets updated as users like a song. As the user is given a soundtrack either randomly or from the list of suggested tracks, he/she has the option to like the song which would improve suggestions. Figure 1 shows a sample screenshot of the user interface.

4. IMPLEMENTATION

The approach adopted for this recommendation system is the clustering technique which is one of the mining techniques. Clustering is referred to as an unsupervised learning which assigns data points to groups so that the more similar data points are in the same groups. The similarity of the data points are determined using a distance measure. The selection of particular clustering algorithm depends on

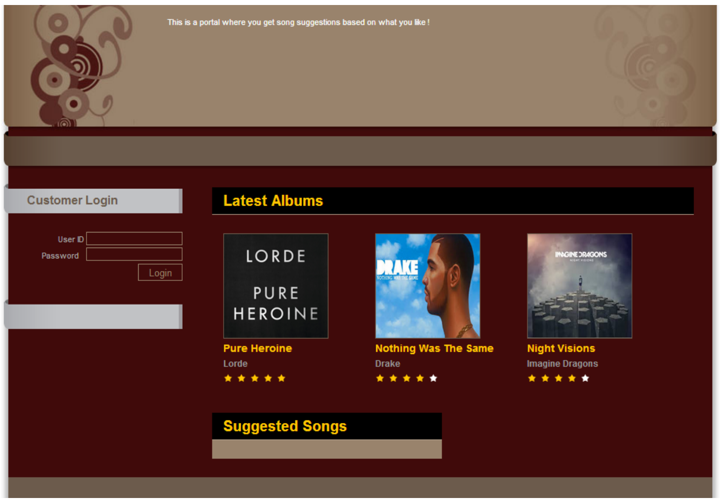
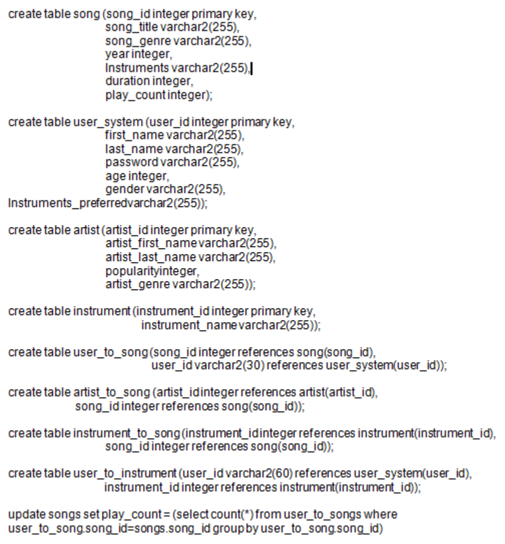


Figure 1: UI



```
create table song (song_id integer primary key,
                    song_title varchar2(255),
                    song_genre varchar2(255),
                    year integer,
                    Instruments varchar2(255),
                    duration integer,
                    play_count integer);

create table user_system (user_id integer primary key,
                    first_name varchar2(255),
                    last_name varchar2(255),
                    password varchar2(255),
                    age integer,
                    gender varchar2(255),
Instruments_preferredvarchar2(255));

create table artist (artist_id integer primary key,
                    artist_first_name varchar2(255),
                    artist_last_name varchar2(255),
                    popularityinteger,
                    artist_genre varchar2(255));

create table instrument (instrument_id integer primary key,
                           instrument_name varchar2(255));

create table user_to_song (song_id integer references song(song_id),
                             user_id varchar2(30) references user_system(user_id));

create table artist_to_song (artist_idinteger references artist(artist_id),
              song_id integer references song(song_id));

create table instrument_to_song (instrument_idinteger references instrument(instrument_id),
                    song_id integer references song(song_id));

create table user_to_instrument (user_id varchar2(60) references user_system(user_id),
                    instrument_id integer references instrument(instrument_id));

update songs set play_count = (select count(*) from user_to_songs where
user_to_song.song_id=songs.song_id group by user_to_song.song_id)
```

Figure 2: Queries

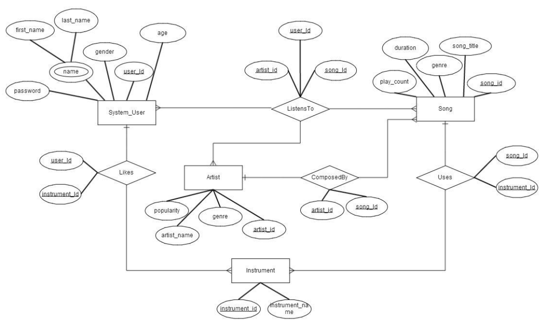
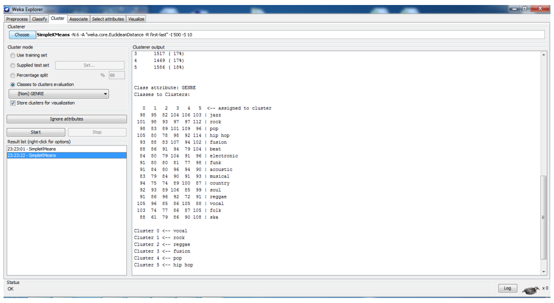


Figure 3: E-R diagram



SimpleKMeans -N 6 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Clusterer output
```
3      1517 ( 17%)
4      1469 ( 17%)
5      1586 ( 18%)


Class attribute: GENRE
Classes to Clusters:

   0   1   2   3   4   5  <-- assigned to cluster
  98  95  82 104 106 103 | jazz
 101  98  93  97  97 112 | rock
  98  83  89 101 109  96 | pop
 105  80  78  98  92 114 | hip hop
  93  88  83 107  94 102 | fusion
  88  86  91  94  79 104 | beat
  84  80  79 104  91  96 | electronic
  91  80  80  81  77  98 | funk
  91  84  80  96  94  90 | acoustic
  83  79  84  90  91  93 | musical
  94  75  74  89 100  87 | country
  92  93  89 106  85  99 | soul
  91  86  96  92  72  91 | reggae
 105  96  85  86 105  88 | vocal
 103  74  77  86  87 105 | folk
  88  61  79  86  90 108 | ska

Cluster 0 <-- vocal
Cluster 1 <-- rock
Cluster 2 <-- reggae
Cluster 3 <-- fusion
Cluster 4 <-- pop
Cluster 5 <-- hip hop
```

Figure 4: Song Clustering results

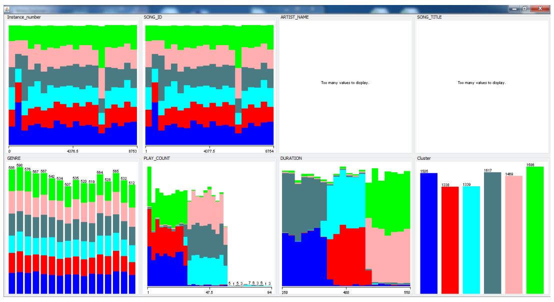


Figure 5: Clusters generated on applying K-Means

the characteristics of the data. Here, the dataset is clustered using the K-means algorithm. K-means algorithm is a method of cluster analysis which aims to partition 'n' observations into 'k' clusters in which each observation belongs to the cluster with the nearest mean. It is one of the simplest unsupervised learning algorithms that solves the clustering problem. The dataset with several music parameters are imported into the Weka explorer mining tool and the clustering is performed on the training data.

The songs are clustered into six clusters based on the genre, artist, duration and playcount. The major clusters are vocal, rock, reggae, fusion, pop and hip hop. The songs liked by the user and its cluster are stored in the 'user_to_song' table and are analyzed to find the major cluster of songs that each user listens to. The music tracks are recommended to users, based on the major cluster of songs that each user listens to. Random songs are also displayed from other clusters, to help the user discover new variety of music. The songs are recommended to the user through the user interface designed for the project. A new user gets random tracks to listen to and is provided with the like button on the UI to like the preferred songs. Initially the new user doesn't have any song suggestions since the system has no knowledge of the user's liking. Figure 7 shows the random tracks displayed for the new user. Based on the randomly tracks liked by the user, the songs are suggested to the user either on refreshing the suggestions area or during the next login attempt.
The results obtained through this approach are presented in Figures 4 and 5.

5. CONCLUSION AND FUTURE WORK

In this project, a music recommendation system was built to suggest songs based on the user's interest. A simple user interface is employed to present the suggestions to the users.

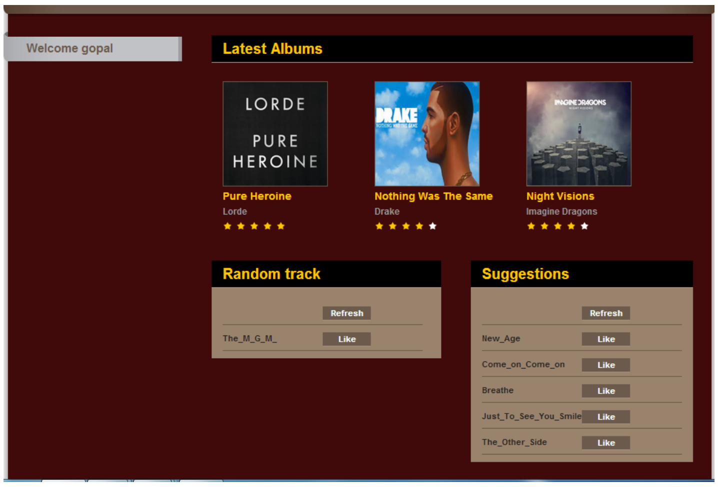


Figure 6: The UI displaying the song recommendations for the existing user

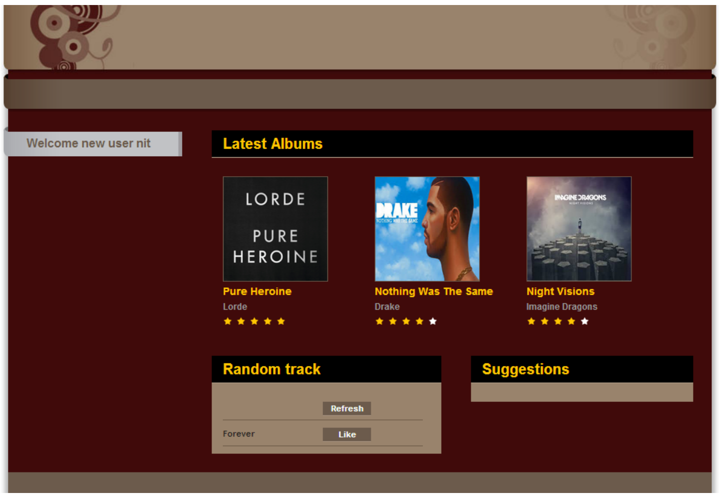


Figure 7: The UI displaying the random song recommendations for the new user

By applying the K-means algorithm, the songs are clustered into groups and the users selection of desired songs are analyzed based on which the songs are recommended to the user through the UI. The recommendation systems tease out various aspects in order to provide a personalized service to the users. The project can be extended by not only suggesting the songs but by also playing the songs, displaying the recently played tracks. While playing the songs the album art can also be made to display.