What music do you like?

A python project of billboard

By Group S

Yang Jia Qi, Sevena, MB954931
Chen Yuan Yuan, Vanessa, MB955283
Long Long, MB955398

Introduction

If you are a music lover, you must have heard about the Billboard charts. Billboard charts tabulate the relative weekly popularity of songs and albums in the United States and elsewhere. The results are published in its magazine and website. The charts may be dedicated to a specific music genre such as R&B, country, rock and pop, or they may cover all genres. In billboard charts, songs can be ranked according to sales, streams or airplay and for some main charts like *the Hot 100* song chart, all three data are used to compile the charts.

In this project, our team would like to use billboard charts to see the trend of a music genres popularity among a period of 60 years. Besides, we are going to find out "the best song" and "the best artist" from billboard charts. Since in our dataset, there are data about music features of "Hot 100" songs and lyrics for over 380,000 songs, analysis about certain artists' music features and words frequency can be included in the project as bonus.

For python libraries and packages we used in this project, they are listed as below:

- **Numpy**: to formalize data and do computation;
- **Pandas**: to clean and formalize data;
- **Matplotlib**: to visualize data:
- **Billboard API**: provided by GitHub contributor, used in crawling billboard chart data[1];
- **Ratelimiter**: to limit the website request times in certain periods when crawling data from billboard[2];
- **Seaborn**: to visualize data;
- **Wordcloud**: to calculate and rank the frequencies of each word in text and make it visual in a picture.

Data Collection

Billboard is in fact a big dataset for this project, to obtain the data we found a Billboard API at GitHub for us to quickly and efficiently realize the crawling work.

At first, we simply used a *For Loop* to realize the goal, but the website declined our visit since we made too frequent requests during crawling processing. After discussion we send an e-mail to the author of the API, he kindly inspired us to limit our request times. The Python package *ratelimiter* we found did great help. It would efficiently and effectively limit the crawling times by setting a slower and fixed speed of visiting times. Had been trying many times, we finally found a perfect rate to capture data, that is, visiting the Billboard URL 2 times per 15 seconds.

With the help of the two packages, we crawled 10 rankings from Billboard, all of them are released weekly, and half of them were from 1958, while the other were from 1990s.

Charts	**Attributes**	
Hot 100 Songs	**Date**	The chart released date
Hot 100 Artists	**Title**	The titles of songs
Hot 100 Country songs	**Artist**	The artist
Hot 100 Dance-electronic songs	**Weeks**	total weeks the song had stayed in the chart till now
Hot 100 Pop songs	**Current**	the current ranking
Hot 100 R&B/HipHop songs	**Peak**	the best ranking the song had ever achieved till now
Hot 100 Rock songs.	**Previous**	the rankings in previous week
Japan Hot 100		
K-pop Hot 100		
China-social Hot 100		

However, the data from the website could not satisfy all our requirements. We found a dataset, named *Hot 100 Audio Features*[3], containing music features value of popularity, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence and tempo. They are generated by Spotify Web API.

In addition, we found another data file from Kaggle, named *Lyrics data[4]*. Its columns included songs names, lyrics and idioms, which provides us a chance to gain a brief look at what people would like to express through songs by the most frequent words and phrases showed up.

In general, we used two ways to complete our datasets, including the primary data we crawled from Billboard and secondary data downloaded from other two websites. The size of final data files used in following analysis are up to 314MB.

Data Cleaning

After integrating all our datasets, it was time to do the cleaning work. Firstly, we must understand the data structures so that we can process them to fit in our analytical objective. For charts data, we used *dtypes()* to check the data type of each column in dataframes and found that columns [title] and [artist] are both 'object' with blank spaces at the beginning and the end of every record. In deleting those needless blank spaces, the method *string.strip()* played a greatly important role, specific code is written like this *hot100['title'].str.strip()*. All those ten charts from Billboard need such cleaning process so that the data can be purer.

Another problem was that there were columns like [peak] and [previous] whose type should be 'integer' but appeared to be 'object'. It was because there were strings 'None' in these columns. In order to uniform the data type of columns [weeks], [current], [previous] and [peak] to type *integer*, we applied the method *replace()* to substitute the 'None' with 0 firstly. And then, the method *dataframe.apply()* and *lambda* function were utilized for transforming the data type of column [peak] and [previous] from 'object' to 'int'.

Since many songs were jointly performed by multiple singers, data about artists in the column [artist] of chart Hot-100 were messy. For the sake of satisfying the visualizing requirement of artist popularity analysis, we decided to preserve the lead singer of the songs in the 'artist' column and removed other assisted singers. Hence, we used method *find()* to locate the position of string 'featuring' or 'Featuring' so that achieving the objective of preservation of artist name that is before those two strings. There is a kindly note that because of the similarity of data structure of all tables we collected and used, majority of cleaning process are operated for all tables.

Data Process and Visualization

Ⅰ. Trend of Music Genre From 1958 to 2019

To analyze this subject, what the most important is to categorize the songs in table Hot 100 into different genre styles. Pursuant to the classification of the five tables for five genre styles, including country song, dance electronic song, R&B/HipHop, pop and rock song, it is convenient for us to classify songs in table *Hot 100*.

We created new dataframes made up with two columns, named [title] and [type], to store all songs and their respective genre styles from those five tables. Then we used method *pd.merge()* to combine the table *Hot 100* with those new dataframes so that we can know the genre style of songs in table Hot 100. Finished the classification, the method *groupby()* as well as *matplotlib* library is very helpful for drawing line chart with five curves revealing the trend of music genre from 1958 to 2019.

Ⅱ. Percentage Changes of Music Genre for Diverse Eras

On the purpose of making a more delicate comparison of various music styles in different eras, we firstly needed to divide all records in table *Hot 100* into six groups according to the year. In this part, we used method *pd.concat()* to connect parts of tables which facilitated the calculation of percentages of each genre style for respective eras.

Secondly, after obtaining the percentages and saved them into lists, we defined a function named *stacked_bar* to paint out the stacked bar chart. In this function, we used *cumsum()* to achieve the cumulative bars and we also configured the parameters of chart, such as color, width, height and label position.

Ⅲ. Which Song Is Always the No.1

Wondering which song has been the most popular for the longest time, we filtered out all records of the top 1 in table Hot 100 for each week. And again, using *groupby()* to get the number of weeks that each song was at the No.1 according to the column [current].

Ⅳ. Who Is the Hottest Artist

To pick out the top 20 artists most-loved by listeners, based on *Hot 100 Artist* chart, we used the *groupby()* method to find every singers' total weeks being on this chart. Then we saved the singers names and their respective on-list weeks into the dictionary format that one was set as key and the other was value. With the help of seaborn, the bar chart demonstrated the final results sorted by descending order.

By the way, we also have tried to make the ranking based on the *Hot 100* Song chart, but the result had differences with the present one. After looking for the gist of Billboard's Hot Artist ranking, it was obviously that they calculated the total ranking according to every artists' income from records sold through both live and online, the frequencies of the released of songs and other activities. Therefore, their outcome is much more reasonable.

Ⅴ. Music Features of Top 5 Artists

Based on the findings of previous visualization, in this part we used file *Hot 100 Audio Features* to see music features of top 5 artists.

Firstly, a function "*get_features(artist)*" was defined to return list of specific artists' 7 music features. In this function, we first used *str.find()* in column [Song ID] to get all the rows of songs performed by the artist, because the data was not clean enough with name of song and its artist were combined together in the same column. Then we computed the average of each features (danceability, energy, instrumentalness, liveness, speechiness, valence and acousticness) of every song. Next, we created dataframe for data extracted and wrote codes to make spider chart for every artist.

Ⅵ. What People Mostly Frequently Express by Songs

To see the situation of English songs, we filter the lyric dataset by idioms and made a wordcloud with the help of the library WordCloud. Since the WordCloud requires a text format file, we need to rewrite it into a text, and encoded it into utf-8 format. After the first version of wordcloud had been made, we found that the library would not filter those meaningless words like interjections, auxiliary word, etc., but luckily it provides stopword function to allow us to set some unnecessary words so they would not show up in the picture. Finally we manually picked out a set of words including 'hey', 'oh', 'the', 'in', 'to', 'you', 'it', 'la', 'that', 'on', 'and', 'is', 'yeah', 'And', 'my', 'me', 'na', 'not', 'do', 'no', 'there', 'been', 've', 'be', "don\'t", 'are', "I'm" and 'of'.

Findings and Conclusion

Having went through all the work above, we understand how to launch a python project and realize that when meet with large quantity of data, python is quite a powerful and highly efficient tool.

Here are several findings we conclude about nowadays music through Billboard:

1. For music genres, people's appeals have significant changes in pop music and R&B/Hiphpo music, that the fans of former type has increased while the later one has declined.
2. In hottest song analysis, we found that the "Old Town Road" was the hottest song, which stayed at top one position for the longest time, nearly 20 weeks.
3. Through hottest artists' analysis, it is clearly that the top three most popular artists, Maroon 5, Drake and Eminem, have slight differences in their respective heat. We could say that they are in fierce competition and the ranking could change at any time.
4. In further analysis about artists features, it is intuitively that the five singers we pick have similar music features, while Drake and Eminem win more scores in speechiness and liveness. In general, songs with higher danceability and energy would be more popular.
5. In lyrics analysis, the most ten frequently written phrases are "let go", "know what", "your love", "we can", "your eye", "your heart", "all night", "your hand", "look at" and "right now". Obviously, people's attentions are focus on the other one in the relationship and express some wishes about two persons future.

At the same time, problems exist and more things we could do are found as well.

1. The Billboard API we use may meet an error about the data format, since the early released charts were a little different from the charts nowadays. We have made some adjustments on the original codes to smoothly crawl all the data. After the change, once a capturing loop reached it end, it would keep on catching data from the newest data rather than stop automatically. There could be more perfection in our codes.

2. WordCloud library
   The library is suitable for simply trying and could meet the lowest standard of calculation of words frequencies. However, after filtered by idiom, the dataset size for production of word cloud is 160MB, needing nearly 3 minutes to run out the result, which is quite time-consuming. Moreover, the meaningless word could not be identified automatically, users could only filter them by manual selecting.
3. Since the work we have done are limited in some descriptive analysis, we believe further mathematical analysis like correlation and general liner mode could be set up and used.

Reference

[1] Billboard API: https://github.com/guoguo12/billboard-charts
[2] Ratelimiter: https://pypi.org/project/ratelimiter/
[3] Billboard Hot Weekly Chart (Data.World):
https://data.world/kcmillersean/billboard-hot-100-1958-2017
[4] Song lyrics from six musical genres (Kaggle) :
https://www.kaggle.com/neisse/scrapped-lyrics-from-6-genres