SI 206: Data-Oriented Programming

Final Project Report
— Billboard X Spotify—

By: Amy Tang and Yilin Fang

Github repo:
https://github.com/atang2100/si206_finalproject

Original Goals

The main objective of this project is to analyze the most popular music in the United States of the week — as listed on the Billboard's *Hot 100 Chart* by comparing it to popular playlists on Spotify. Spotify, a well-known music streaming service company, generates and frequently updates its own "top songs" public playlists based on country and genre.

In this project, there are the following two main goals:

- Firstly, we want to assess whether or not the songs that are popular in the States are also popular elsewhere in North America (excluding the Caribbean countries) by finding the percentage of songs that overlapped between each of the Billboard Hot 100 songs and Spotify's *Top 50 - USA*, *Top 50 - Canada*, and *Top 50 - Mexico* playlists.
- Secondly, we want to determine the most popular genres that are on the Billboard chart, that is, by computing the percentage of overlapping songs between each of the following playlists created by Spotify: *Pop Rising* (Pop), *Rap Caviar* (Rap), *R&B Favourites* (R&B), *Hot Country* (Country), and *Viva Latino* (Latin Pop), and the Billboard Hot 100 Chart.

By the end of the project, we hope to gain a deeper understanding of the current pop music landscape in the U.S. and North America at large, and what people are actually listening to this week!

Achieved Goals

☐ Extract data on all of the songs that are on the current week's Hot 100 Chart from the Billboard website using BeautifulSoup	Achieved
☐ Convert all of the gathered data from the Billboard's website into 1 table in a database file that is readable in SQLite	Achieved
☐ Extract data on all of the songs that are on the 3 regional Spotify playlists using Spotipy and an API key	Achieved
☐ Extract data on all of the songs that are on the 5 genre Spotify playlists using Spotipy and an API key	Achieved
☐ Combine all of the data from the regional playlists into 1 table in a database and clearly labels the positions of songs on each chart	Achieved

☐ Combine all of the data from the genre playlists into 1 table in a database and clearly labels the positions of songs on each chart	Achieved
☐ Create visualizations based on calculated percentages	Achieved

Problems that We Faced

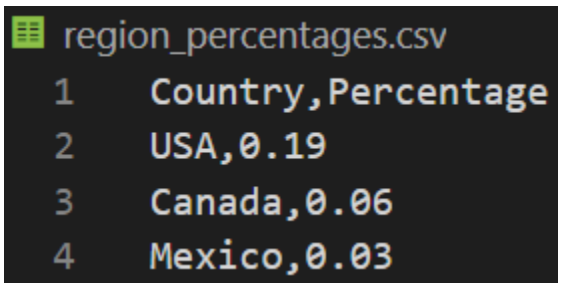
While working on this project, one of our problems faced was finding APIs that were free for use and were updated frequently. Our original idea for the project was to analyze flights and music events, but we struggled to find free flight APIs and the music events API was outdated with events in 2017. Therefore, we decided to switch the project to use the Spotify and Billboard data because the APIs were free and the data was frequently updated.

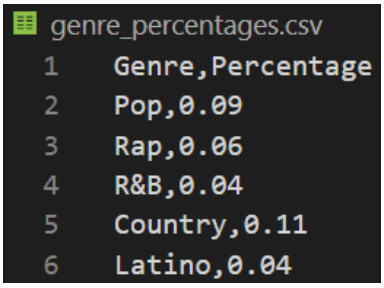
Another problem was figuring out how to pull data from the Spotify API. However, after doing some research, we found Spotipy which is an external library that helps pull data from the API. After searching up some documentation, it was easy to figure out and it solved the problem of figuring out how to pull from the API.

In addition, as we were computing the percentages, we realized that the data we collected may not be the best representation of the actual percentage of overlap since there are a lot of songs that are on multiple playlists. For example, since the Canada and the USA Spotify playlists have a lot of overlapping songs, the percentage we calculated for Canada does not represent the actual percentage of songs overlapped between the Canada Spotify playlist and the Billboard Chart.

Calculation Results

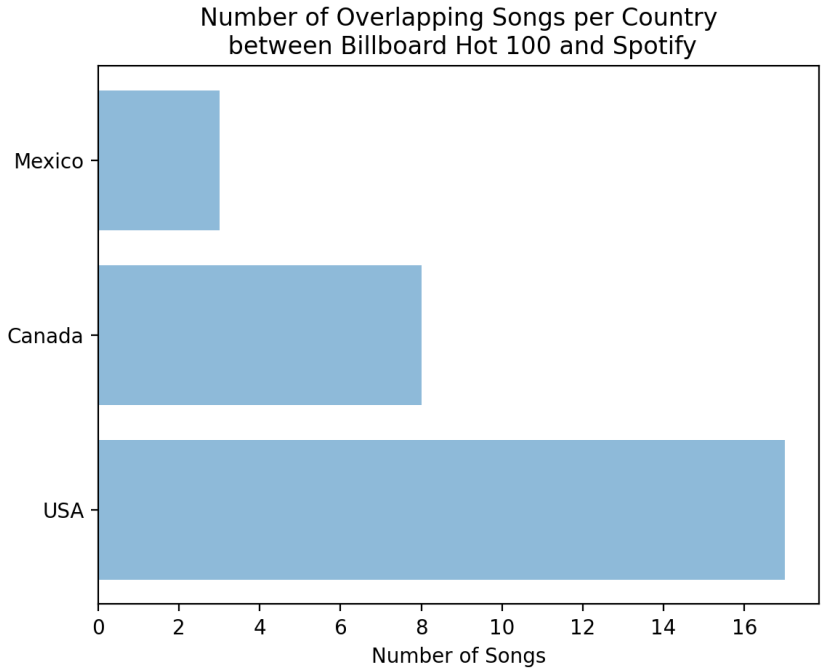
We calculated the percentage of overlapping songs between top songs for each country, top songs for each genre, and the Billboard Hot 100. The calculated results are in the files region_precentages.csv and genre_precentages.csv

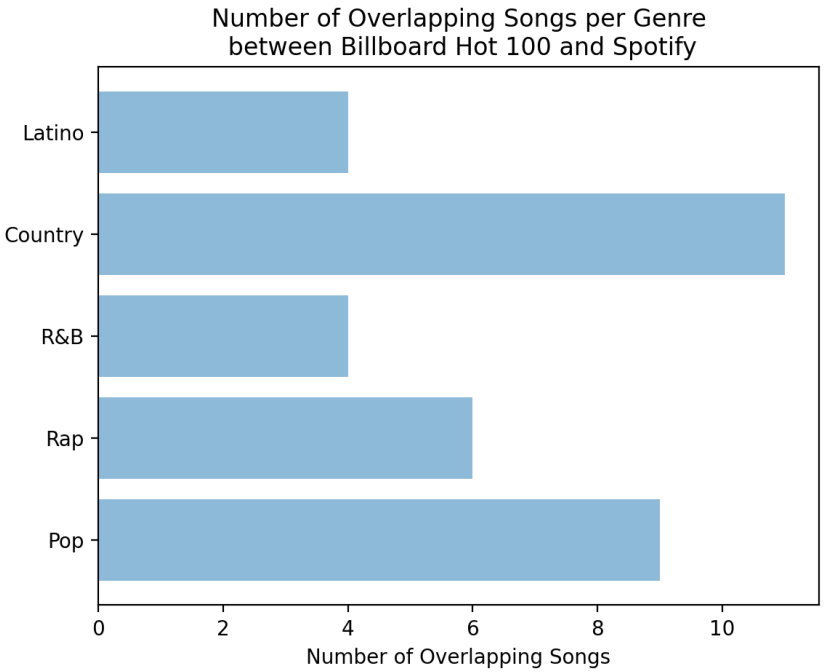




Visualizations

Two bar charts were generated for the overlapping songs between top songs for each country, top songs for each genre, and the Billboard Hot 100.





Instructions for Running Our Code

An external library was used for this project, so it is necessary to install this library (spotipy) before running the Python file.

1. pip install spotipy
2. python3 spotipy.py

Documentation Per Function

- spotify_topSongs
    - Input:
        - category - array of genres/regions
        - playlists - array of playlist URIs used to pull from the API
    - Output:
        - returns a list of tuples of this format: (song_uri, song_name, category, artist_name, artist_uri)
    - Purpose: pulls from the spotipy API and returns the data in a list of tuples
- add_song_by_region
    - Input:
        - cur, conn: connections to the database
        - song_data: list of tuples of songs
    - Output:
        - None
    - Purpose: inserts rows 25 at a time into the database
- insert_into_database_region
    - Input:
        - cur, conn: connections to the database
        - song_data: list of tuples of songs
        - start, end: indexes
    - Output:
        - None
    - Purpose: query and insert the data into the database
- add_song_by_genre
    - Input:
        - cur, conn: connections to the database
        - song_data: list of tuples of songs
    - Output:
        - None
    - Purpose: inserts rows 25 at a time into the database
- insert_into_database_genre
    - Input:
        - cur, conn: connections to the database
        - song_data: list of tuples of songs

        - start, end: indexes
    - Output:
        - None
    - Purpose: query and insert the data into the database
- setUpDatabase
    - Input:
        - db_name: name of the database
    - Output:
        - cur, conn: connections to the database
    - Purpose: create the database and create tables
- billboard_info
    - Input:
        - url: billboard top 100 url
    - Output:
        - a list of tuples with (title, name, weeks on board)
    - Purpose: get billboard data into a list of tuples
- add_billboard_songs
    - Input:
        - song_info: list of songs
        - cur: used to query
        - conn: used to commit changes
        - Start, end: indices
    - Output:
        - None
    - Purpose: insert data into billboard table 25 at a time
- insert_into_database_billboard
    - Input:
        - cur, conn: connections to the database
        - song_data: list of tuples of songs
        - start, end: indices
    - Output:
        - None
    - Purpose: query and insert the Billboard data into the database
- find_similar_songs_region
    - Input:
        - cur: used to query
    - Output:
        - return a tuple of an array of percentages and an array of regions
    - Purpose: calculate how many songs in the top-50 playlists for each country are in the billboard top 100 songs
- find_similar_songs_genre
    - Input:
        - cur: used to query
    - Output:

        - return a tuple of an array of percentages and an array of genres
    - Purpose: calculate how many songs in the top playlists for each genre are in the billboard top 100 songs
- make_bar_chart_region
    - Input:
        - totals: number of songs in region playlists that overlap
        - labels: array of genres
    - Output:
        - creates a bar chart
    - Purpose: creates a bar chart with the given labels and percentages
- make_bar_chart_genre
    - Input:
        - totals: number of songs in genre playlists that overlap
        - labels: array of genres
    - Output:
        - creates a bar chart
    - Purpose: creates a bar chart with the given labels and percentages

Resources

Date	Issue Description	Location of Resource	Result
04/21/2022	Did not know how to use the spotipy API to pull playlist songs	https://github.com/plamere/spotipy/blob/master/examples/playlist_tracks.py	This example showed how to use the playlist URI to pull and get songs
04/21/2022	Did not know how the JSON object was formatted after pulling from spotipy	https://developer.spotify.com/console/get-playlist-tracks/?playlist_id=&market=&fields=&limit=&offset=&additional_types=	This website visualized the JSON object so we could see the structure and index accordingly