

Final Project: DJ WATSON

COSC 89.11: Cognitive Computing with Watson

Authors: Ryoya Wakamatsu, Ali Hagen, Zac Gottschall, Teddy Ni

DJ WATSON LINK: <http://djwatson.surge.sh/>



Business Model & Motivation

Putting together a perfectly ordered playlist is no easy feat. In fact, creating such an optimal playlist is nearly impossible for a human, particularly when playlists consist of a multitude of songs. This project aims to use cognitive computing in order to make this possible.

DJ WATSON is a paid service and/or feature of a preexisting paid service (such as Spotify) in which users pay either a one-time or recurring fee to use DJ Watson's services. DJ Watson can make paid appearances at events, ranging from small gatherings such as private weddings, to the world's largest festivals like Coachella. This will not only allow DJ WATSON to earn money from events/shows, but also allows more people to know about the product.

Knowledge Source

Positive data

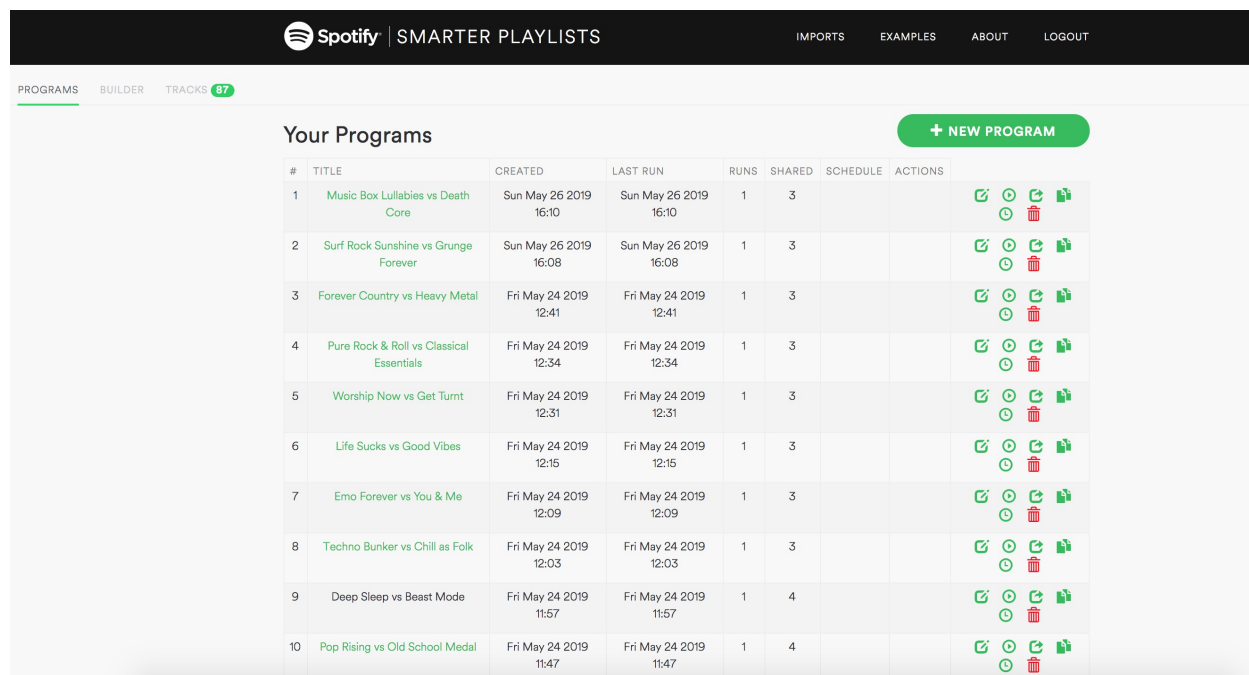
For the positive data, we used pairs of songs that are next to each other in albums. This made

sense to us since albums are put in a specific order so that the songs sound nicely together.









































Negative data

In order to create a significant amount of negative data without manually choosing songs, we used a playlist generator from an online application that uses Spotify's API.

(<http://playlistmachinery.com/>) With this, we made playlists that alternate between playlists that have completely opposite types of music to create pairs of songs that would not go well together. For examples, Techno Bunker and Chill as Folk. Here is a view of these playlists in the app:



The screenshot shows the Spotify 'Smarter Playlists' app interface. At the top, there's a navigation bar with 'Spotify | SMARTER PLAYLISTS' and links for 'IMPORTS', 'EXAMPLES', 'ABOUT', and 'LOGOUT'. Below this, a secondary bar shows 'PROGRAMS', 'BUILDER', and 'TRACKS 87'. The main content area is titled 'Your Programs' and features a '+ NEW PROGRAM' button. It contains a table with 10 rows of program data. Each row includes a number, a title, creation and last run dates/times, run and share counts, and a set of action icons (edit, play, delete, share).

#	TITLE	CREATED	LAST RUN	RUNS	SHARED	SCHEDULE	ACTIONS
1	Music Box Lullabies vs Death Core	Sun May 26 2019 16:10	Sun May 26 2019 16:10	1	3		   
2	Surf Rock Sunshine vs Grunge Forever	Sun May 26 2019 16:08	Sun May 26 2019 16:08	1	3		   
3	Forever Country vs Heavy Metal	Fri May 24 2019 12:41	Fri May 24 2019 12:41	1	3		   
4	Pure Rock & Roll vs Classical Essentials	Fri May 24 2019 12:34	Fri May 24 2019 12:34	1	3		   
5	Worship Now vs Get Turnt	Fri May 24 2019 12:31	Fri May 24 2019 12:31	1	3		   
6	Life Sucks vs Good Vibes	Fri May 24 2019 12:15	Fri May 24 2019 12:15	1	3		   
7	Emo Forever vs You & Me	Fri May 24 2019 12:09	Fri May 24 2019 12:09	1	3		   
8	Techno Bunker vs Chill as Folk	Fri May 24 2019 12:03	Fri May 24 2019 12:03	1	3		   
9	Deep Sleep vs Beast Mode	Fri May 24 2019 11:57	Fri May 24 2019 11:57	1	4		   
10	Pop Rising vs Old School Medal	Fri May 24 2019 11:47	Fri May 24 2019 11:47	1	4		   

Tools and Techniques

The two main tools we used were Brain.js and Spotify's API.

Brain.js



Neural Networks in JavaScript

DJ WATSON utilized Brain.js as the alternative to Watson's API. Brain.js is a Javascript library for Neural Networks.

We decided to use Brain.js because of its versatility, and integration ease. IBM Watson limited us to two features, which we strongly believed would limit our models' abilities.

Spotify's API

We utilized Spotify's API to collect data about songs, specifically the audio features that we used to train our model. In addition, we used the API to access playlist data, and hopefully in the future, use Spotify API's playback features.

We created a scraper that takes in a list of playlists, and generates an excel sheet of all the audio features associated with each song from Spotify's API. We used this excel sheet to generate our model.

Organization

We designed a web client and web server to handle our purposes. Our frontend is built in React, and our backend in Node/Express. Essentially, the frontend deals with the user input and display, while the backend handles all the data sorting and machine learning.

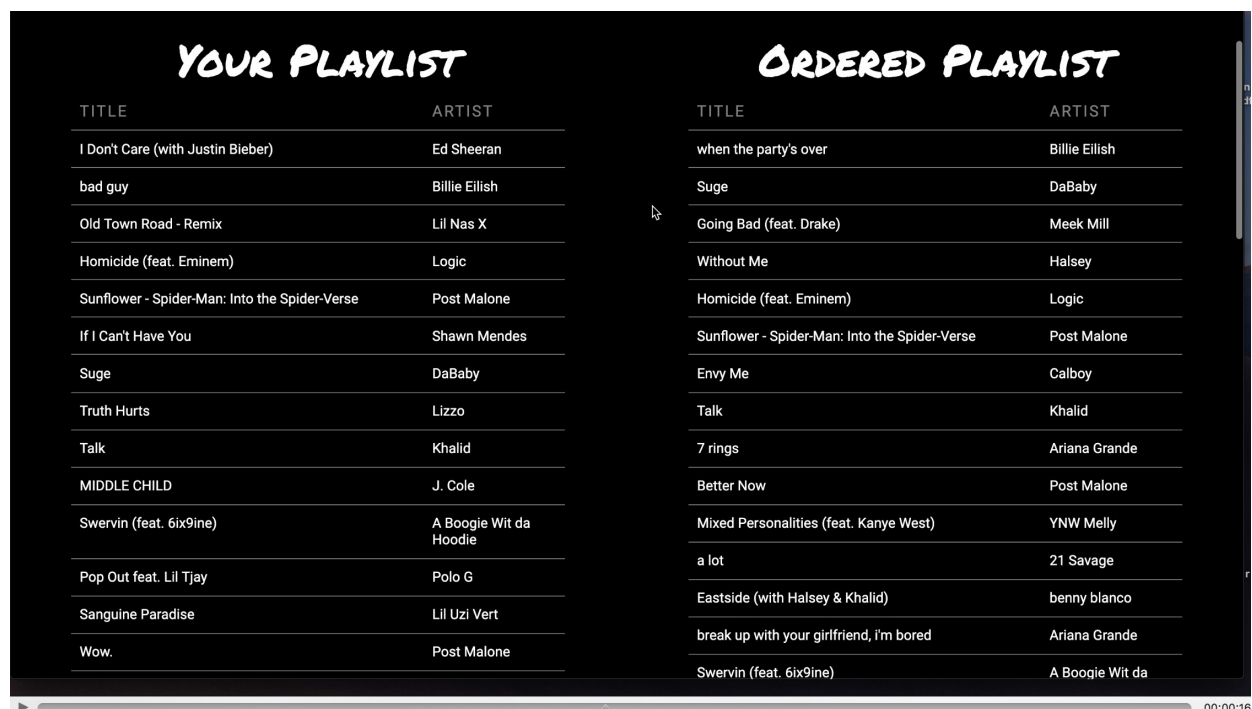
Difficulties Encountered

One initial problem we faced was realizing that Watson's Machine Learning lite plan would not allow us to input more than 3 features and we needed more. We resolved this problem though by finding the separate ML API that would work for all of the features we wanted to include.

With the enormous range of music out there on spotify, there are definitely genres that were left out of our training data. This evidently may have left some bias in our app's results but this was definitely something we were aware and concerned about. As a whole, our group discussed a lot the ways that our app may be biased towards liking certian music pairs, or how that in general is a subjective view. As a result, we were worried that it would be difficult to objectively tell whether DJ Watson was working or not. In the end, we agreed the best we could do was see that the app's results fit our general taste of which songs sound good together.

Analysis of Results

Here is an example we used of DJ Watson ordering the playlist United States Top 50 from Spotify:



YOUR PLAYLIST		ORDERED PLAYLIST	
TITLE	ARTIST	TITLE	ARTIST
I Don't Care (with Justin Bieber)	Ed Sheeran	when the party's over	Billie Eilish
bad guy	Billie Eilish	Suge	DaBaby
Old Town Road - Remix	Lil Nas X	Going Bad (feat. Drake)	Meek Mill
Homicide (feat. Eminem)	Logic	Without Me	Halsey
Sunflower - Spider-Man: Into the Spider-Verse	Post Malone	Homicide (feat. Eminem)	Logic
If I Can't Have You	Shawn Mendes	Sunflower - Spider-Man: Into the Spider-Verse	Post Malone
Suge	DaBaby	Envy Me	Calboy
Truth Hurts	Lizzo	Talk	Khalid
Talk	Khalid	7 rings	Ariana Grande
MIDDLE CHILD	J. Cole	Better Now	Post Malone
Swervin (feat. 6ix9ine)	A Boogie Wit da Hoodie	Mixed Personalities (feat. Kanye West)	YNW Melly
Pop Out feat. Lil Tjay	Polo G	a lot	21 Savage
Sanguine Paradise	Lil Uzi Vert	Eastside (with Halsey & Khalid)	benny blanco
Wow.	Post Malone	break up with your girlfriend, i'm bored	Ariana Grande
		Swervin (feat. 6ix9ine)	A Boogie Wit da

In order to analyze the results, we looked at specific pairs of songs that Watson did (or didn't) put together.

Correct Positive Example: Suge by DaBaby & Going Bad by Meek Mill feat Drake

- This pair of songs sound great together with a very similar vibe of rap and beat style that

flows nicely into each other.

Incorrect Example: when the party's over by Billie Eilish & Suge by DaBaby

- Very surprisingly, this is the first pair in the ordered playlist so DJ Watson calculated a high percentage match for these songs. However, Billie Eilish's song is much more slow of a tempo and a darker vibe so it does not go together well with Suge by DaBaby.

Correct Negative Example: bury a friend by Billie Eilish & Happier by Marshmello and Bastille

- This pair was correctly not placed next to each other in the ordered playlist since bury a friend is a very gloomy song with a beat that dramatically contrasts and does not sound nicely with Happier.

Missed Positive Example: Happier by Marshmello and Bastille & Eastside by benny blanco, Halsey and Khalid

- These songs are very similar flow well into each other but DJ Watson unfortunately did not place them next to each other in the playlist. This highlights the way a great pair could be overlooked if its percentage score is only slightly lower than another match.

Assessment of resulting system's effectiveness

DJ WATSON worked extremely well when the queried playlist consisted of a wide diversity of music types and genres. Upon inspection of the playlist, the songs were mostly ordered appropriately with songs flowing well from one to the next.

What DJ WATSON did struggle with were playlists that consisted of songs that were similar to each other. The scores that DJ WATSON assigns to the songs are extremely close and thus, the system fails to order them in an interesting way. It is worth noting, however, that in such a case, the songs already flow well into each other. Another limitation to the system was that there was a hard limit of 50 songs. If a user queried a playlist that consisted of more than 50 songs, DJ WATSON would not be able to generate a reordered list.

Points of Improvement and Further Work

There are many points of improvement for DJ WATSON. Firstly, we can personalize the model to the user by taking into account user feedback to retrain the model. Secondly, focusing the algorithm to weigh more heavily on the start and end of songs will allow transitions to be more smooth and natural. Thirdly, incorporating an automatic linkage to Spotify so that the user can play directly from the browser would be a great addition for the user experience. Lastly, we could improve the model further by utilizing different Machine Learning APIs.