C. Kaudan, J. Vasylenko, W. Wang, A. Patel, T. Larsen

IEOR 135 Project Write-up

April 30, 2018

Spotify Recommender Challenge

Over the last three months we have been working on Spotify's Million Playlist Challenge, a competition launched by Spotify with help from The University of Massachusetts, Amherst and Johannes Kepler University Linz In Austria. The challenge description was simple: using the training dataset provided, consisting of one million user-generated Spotify playlists, we were tasked with recommending 500 songs to each of the 1,000 incomplete playlists in a separate test set, which each had between 0 and 100 seeds songs from which we were to base our recommendations. Many of the competitors in this data challenge are researchers from both industry and academia, but as a group of five ambitious undergraduates we excitedly took on the challenge.

The first problem we had to solve was learning how current Music Recommendation Systems are implemented. In order to overcome this challenge, we spent the first few weeks of our project pouring through existing research, learning about the recommender systems being used by major companies like Netflix, Spotify, and Pandora, just to name a few.

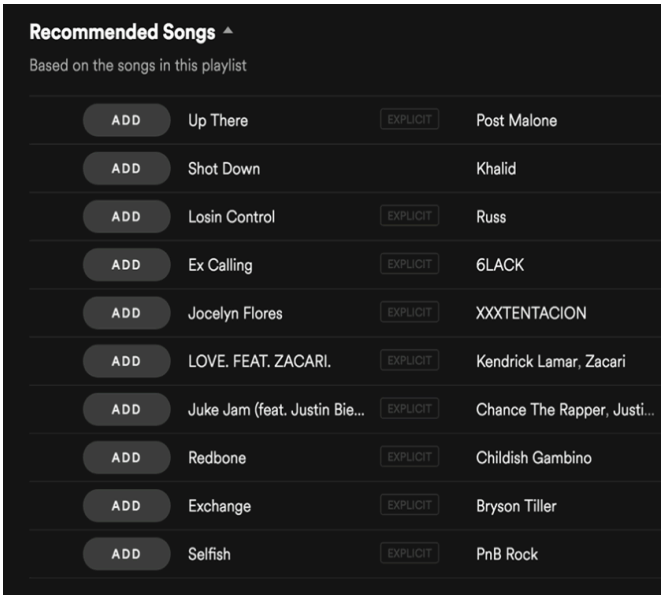
The algorithm we settled on is known as Collaborative Filtering, which succeeds especially in cases where the input data is implicit feedback, meaning instead of an explicit star or numeric ratings the data only includes a 1 if a particular song is in a given playlist and a 0 otherwise. In theory, collaborative filtering is simple: if person 1 likes songs A, B, and C, and we know that person 2 likes songs A and B, we can infer that person B has a high probability of liking song C as well.

The second major hurdle we had to overcome was scaling our solution to include all of our data. After gaining a solid understanding of Recommender Systems, we first implemented

our solution on 6 out of the 1000 data slices using Pandas and NumPy's linear algebra capabilities to transform our data into a dataframe of 1s and 0s of dimension number of playlists by number of unique songs, and a recommender class we built in order to generate predictions. In order to scale our model to the full data set we used Amazon AWS to host our data, and Apache SPARK running on Databricks to handle our computation. We initialized our cluster to autoscale between two and eight worker nodes in order to increase speed and minimize our overall costs, and SPARK's MLlib to implement an Alternating Least Squares approach to Collaborative Filtering.

Our third and final major hurdle was learning the technologies in our new development stack. Hosting our data was relatively simple, but learning to optimize our cluster settings and converting our code from Pandas and NumPy to MLlib provided much more of a challenge than even we expected. After developing a working implementation and optimizing our parameters, we were amazed by our results. We first inputted a small playlist of only five songs and set our output to be the 10 best recommendations to extend the input playlist. Here is how our recommendations compared to those that Spotify would have made for the same input playlist:

songid	artist_name	track_name
13.0	Future	Mask Off
6.0	Aminé	Caroline
5.0	KYLE	iSpy (feat. Lil Y...
15.0	Travis Scott	goosebumps
4.0	Post Malone	Congratulations
8.0	Migos	Bad and Boujee (f...
7.0	Lil Uzi Vert	XO TOUR Llif3
10.0	Big Sean	Bounce Back
2.0	DRAM	Broccoli (feat. L...
0.0	Kendrick Lamar	HUMBLE.



Our predictions, shown at left, are very similar to those made by Spotify on the right! We both included songs by Post Malone and Kendrick Lamar, and the remaining songs were all very similar. Perhaps most significantly, we were able to accurately map genres even though all we had was a dataframe indexed by song IDs and playlist numbers, and populated by 1s and 0s.

Our next steps are to improve on our current model before officially submitting to the Spotify Challenge. In our development phase we also worked on two other models, the first of which was a Neural Net approach to Collaborative Filtering using Keras and Theano, and the second was a content-based approach using the last.fm API. While both approaches yielded exceptionally high accuracies, given our time frame we were not able to figure out how to scale them to the size of our data. In our final model, we would like to combine elements from all three approaches to create a unique approach to recommendation.

With all of the data that exists today, recommender system research is slated to expand greatly over the coming years. While we may not have developed a new state of the art technology over the course of this project, we hope that by synthesizing our approaches we can add knowledge to this exciting and rapidly growing field. Every time we're at home scrolling through Netflix movie recommendations our at the gym listening to the songs Spotify is recommending us, we'll do so with a greater appreciation for what's going on behind the scenes, and in the backs of our minds we'll continually be thinking how this system could be improved.