Improving Hit Song Prediction Networks with Musician Co-Collaboration Network Data

Trevor Hubbard
UMass Amherst
tmhubbard@umass.edu

Phillip Sifferlen
UMass Amherst
psifferlen@umass.edu

1. Introduction

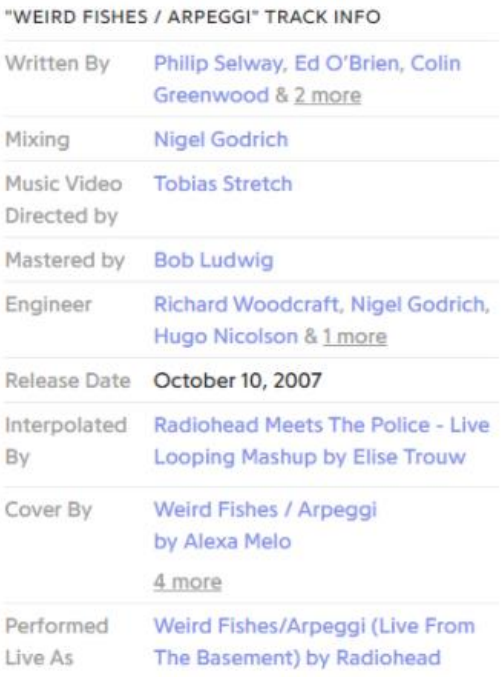
We're focusing on the Hit Song Science Problem, which is the problem of deciding whether a given song will become a "hit" or not. Recent approaches to this problem have attempted to use high-level acoustic features (such as a song's key, its tempo, and various descriptors like "valence" and "danceability") to make predictions about a song's popularity potential. Our approach intends on combining this data with social network information - more specifically, combining it with the output of a network auto-encoder encoding a co-collaborator graph we construct. We hypothesize that prediction accuracy will increase with this additional information, as popularity is influenced by more than just what the song sounds like: songs performed by already popular artists are generally more likely to become popular!

2. Problem Statement

The Hit Song Science Problem has no universally accepted definition of a "hit", but most research uses some sort of sales chart presence to discriminate between hits and non-hits. In our research, we define a "hit" as a song that has, at some point, charted on the Billboard Hot-100. So, hit-status is a binary variable: 0 if it's not a hit, or 1 if it's ended up on the chart. With this formulation, you can state the Hit Song Science Problem as: given information about a song (i.e., audio information & co-collaborator information), can an algorithm predict whether or not the song will be a hit?

In order for a neural network to learn which artists might make a hit, it'll need to know a *lot* about the artists in the industry et large. Naturally, then, we needed to create a network from artists collaborating on songs that represented the industry in a holistic way. In other music information retrieval (MIR) contexts, the Million Song Dataset (MSD) has been used as a basis for a "universe" of songs. In addition to the songs in the MSD, we would need to consider all of the songs that had been on the Billboard Hot 100 chart. We selected 1990-2010 as our range of years to examine, as these dates were well represented in the MSD.

We filtered the MSD to show only these dates, and extracted a list of the song titles and artist names. (There were 432,005 songs in this list!) Furthermore, we wrote Python scripts using the Billboard API[1] to scrape the Billboard Hot-100 charts for every song appearing between 1990-2010. (There were 7,530 songs like this!) From there, we wrote Python scripts to query both the Genius and Spotify APIs.



"WEIRD FISHES / ARPEGGI" TRACK INFO

Written By	Philip Selway, Ed O'Brien, Colin Greenwood & 2 more
Mixing	Nigel Godrich
Music Video Directed by	Tobias Stretch
Mastered by	Bob Ludwig
Engineer	Richard Woodcraft, Nigel Godrich, Hugo Nicolson & 1 more
Release Date	October 10, 2007
Interpolated By	Radiohead Meets The Police - Live Looping Mashup by Elise Trouw
Cover By	Weird Fishes / Arpeggi by Alexa Melo 4 more
Performed Live As	Weird Fishes/Arpeggi (Live From The Basement) by Radiohead

Figure 1: Genius Collaborator Information for the song Weird Fishes / Arpeggi by Radiohead

From Genius, we extracted collaborator info about each song, which could then be used in service of building the social network. From Spotify, and using the Spotipy module[2], we collected some high level audio features for each of the songs. The features collected include:

- duration
- key
- mode
- time signature
- acousticness
- danceability
- energy
- instrumentalness
- liveness
- loudness
- speechiness
- valence
- tempo

Some of these features are essentially the confidence values of Spotify's own prediction systems - "danceability", for instance, is some prediction about whether or not a song would be danced to. (The predictions themselves are based off of analyses of the spectrograms of the audio's waveforms.) Other features, like duration, are easily calculated by looking at metadata about the audio itself.

We were able to find accurate Genius information to match 241,545 of the songs in the combined MSD-Billboard dataset. From the information about these songs, we built a social network. Nodes in the network represented music industry professionals (artists, producers, writers, engineers, musicians, etc.), and edges between nodes represented songs that the artists worked on together.

Methods using only the high level acoustic information[3] have achieved ~75% accuracy on this problem. We expect that the network we're training will outperform these accuracies, but we're not quite sure *how* much of an impact the social network will have. Additionally, we predict that we'll be able to achieve decent (70%+) levels of accuracy using only the social network information, and no other data. Evaluating the performance of the network will be as simple as computing the accuracy of its predictions on a test set of songs.

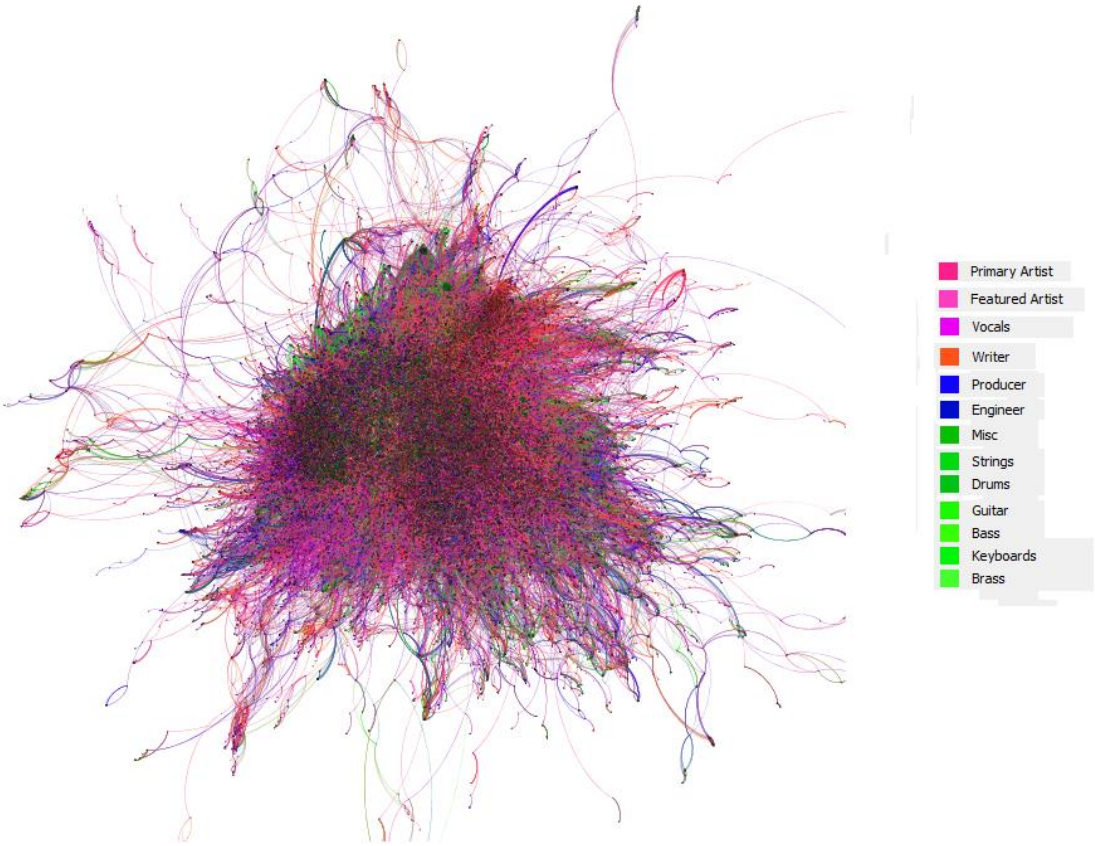


Figure 2: a zoomed-out view of the music industry co-collaboration network. Nodes are colored according to the key on the right, and edges are colored by mixing the colors of the nodes. The graph contains 69,385 nodes, and 704,872 edges.

3. Technical Approach

We've got two main technical challenges to overcome: encoding the social network into a vector space, and designing a network to predict whether or not this song is a hit.

The development of network autoencoders is a relatively new branch of research, so there's no "industry standard" algorithm that we can use to encode the network. Initially, work in this area focused on learning representations for the topological structure of the network - in node2vec[4], for instance, the network is trying to learn a representation that could find the same nodal neighborhood that a random walk of the graph could. More recent work[5] in the field has been focused on *also* encoding a node's attribute information into the vector embedding - in our case, this could be what class of artist the node is (primary artist, producer, writer, guitarist, etc.). In order to learn more about the current landscape of autoencoders, we think it'd be interesting to try multiple different encoding strategies, and compare their performance.

Once we've got a vector embedding of the social network, we plan on designing a Hit Predictor neural network that'll use this network embedding as training data. We're curious about comparing this network's performance to the performance of Hit Predictor trained on only the acoustic features, so we'll run a number of experiments to compare the two. Finally, once we're confident both of the networks are performing to the best of their ability, we'll try to merge the two networks together to see if combining the information will make a substantial impact on prediction accuracy.

The first network uses solely the high level audio features from Spotify as input, with binary classification of hit or non-hit, depending on whether the song has appeared in Billboard. As this network is to be used both in conjunction with, and contrasting against, the social network model, we decided to structure it closely to those used in previous papers[6] dealing solely with Spotify features. This will allow us to see what significant difference the social network addition has made. The network consists of two fully connected layers, thirteen input features, a 20 node hidden layer, and a single output class. The output layer is run through a sigmoid function and then rounded to get the predicted class. For training, we used the RMSprop optimizer with a binary cross entropy loss with logits for a loss function. This loss function makes sure to include the sigmoid function. To create the training data, we took all the 7,530 songs that were hits, and then randomly subsampled from the non-hits to create a set that had 50/50 representation of the classes.

4. Preliminary Results

Audio Feature NN: Currently, results from the neural network mirror what we have seen in the earlier papers[6]. The net gives an accuracy of 73% on the validation data with little to no overfitting. The final cross-entropy loss after 20 epochs was .5341. The precision and recall on the validation set were 0.697 and 0.822. The confusion matrix shows a decent amount of false positives, with about half as many false negatives.

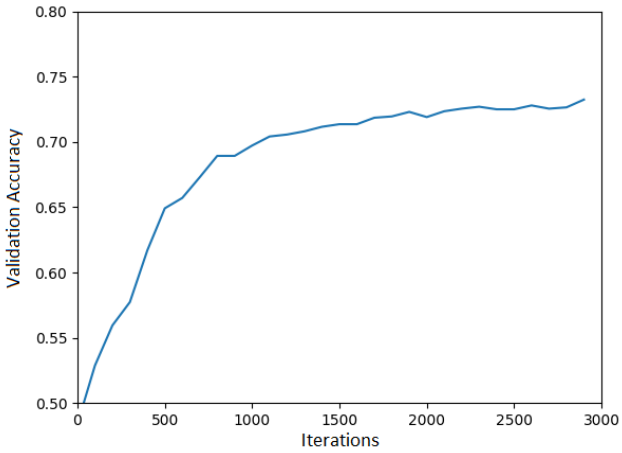


Figure 3: Accuracy of 2-Layer NN Trained on Spotify Audio-Features

		Predicted: Non-Hit	Predicted: Hit
Actual	Non-Hit	649	360
	Hit	180	829

Figure 4: Confusion Matrix of the 2-Layer NN

These are promising results with a relatively simple neural network. Note that the network's simplicity is primarily due to its initial use as a comparison/initial companion to the social network model. Therefore, we will also work further to build and tune this model to try and get the best accuracy possible.

Network Encoding: We needed to do a decent amount of work in the data scraping phase, so most of the progress made in this area was primarily done on scraping and cleaning the social network information, and then using that to construct a graph. We're still performing various analyses on the graph (trying to understand which artists are well-connected, where neighborhoods are forming, etc.), and only now beginning the auto-encoding of the graph. We've just started using node2vec to embed the network. We haven't yet begun learning from the high-dimensional representations it's outputting, but we have tried visualizing some of the clusters that appear in the embedding.

Figure's 5 and 6 demonstrates some of our results! It makes sense that the two clusters formed would represent the large "nucleus" in the middle and the various outlier nodes, respectively; after all, node2vec's embeddings attempt to encode neighborhood structures in graphs. We're excited to see what further analysis brings!

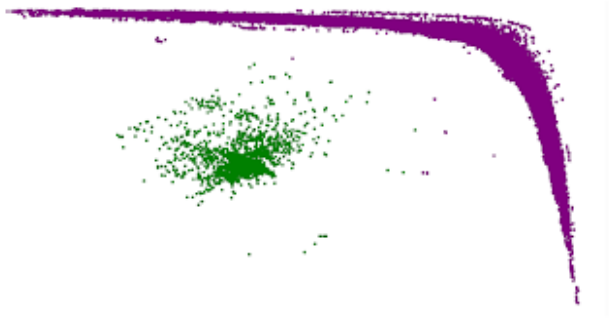


Figure 5: We plotted the node2vec embedding of each node of the social network, and colored the two primary clusters green and purple.

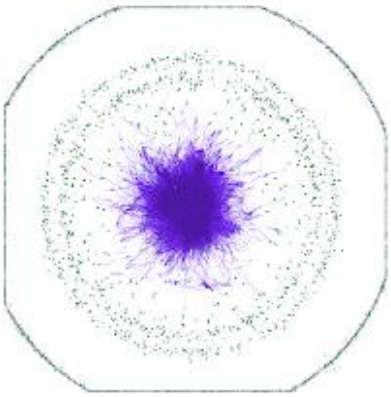


Figure 6: Here are the same colored nodes, but represented in the original social network.

References

[1] Guo, A. Python API for Billboard Data. Github.com. Retrieved from: https://pypi.org/project/billboard.py/

[2] Hilden, F. A Python package for using Spotify API. https://pypi.org/project/spotipy/

[3] Elena Georgieva, Marcella Suta, and Nicholas Burton. "Hitpredict: Predicting hit songs using spotify data." 2018.

[4] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks." *ArXiv.org*, 3 July 2016, arxiv.org/abs/1607.00653.

[5] Dave, Vachik S., et al. "Neural-Brane: Neural Bayesian Personalized Ranking for Attributed Network Embedding." *ArXiv.org*, 20 Aug. 2018, arxiv.org/abs/1804.08774.

[6] Middlebrook, Kai, and Kian Sheik. "Song Hit Prediction: Predicting Billboard Hits Using Spotify Data." *ArXiv.org*, 18 Sept. 2019, arxiv.org/abs/1908.08609.