Unsupervised Music Mapping for Content-Based Music Recommendation

University of West Florida

Department of Mathematics & Statistics

COT6990

Connor Baugh

970594482

May 6, 2021

Abstract

The standard music recommendation systems utilized by the global music streaming industry are in need of a redesign. The systems that music streaming platforms have been relying on for the past decade suffer from a lack of music discoverability, leaving users bored with their recycled recommendations and frustrated by the lack of new and undiscovered content. While traditional content-based recommendation systems have consistently been outperformed by collaborative-filtering methods, we propose multiple unsupervised techniques for mapping music data to a high fidelity lower-dimensional latent space to support content-based systems and offer a potential solution to the "cold start problem" faced by the music streaming industry today.

I. Background & Motivation

The science of music recommendation has become an increasingly significant research topic in the past 20 years. With the advent of the internet and the digital age, the demand for new and easily accessible music grew rapidly and consumers wanted a one-stop-shop for all of their music needs. The early 2000's saw the rise of music streaming platforms such as Napster and Apple's iTunes Store, which were effectively just sites where consumers could download their favorite music to their mp3 players.

The introduction of personalized music recommendation really came about with the launch of Pandora Internet Radio in 2005. Pandora's goal was to create individualized radio stations for its users that only played "good" music, filtering out the bad. Pandora did this by utilizing hired musicians who painstakingly labeled every song in its catalog with attributes. When a user liked or disliked a song, an algorithm would run a search in the background for the attributes assigned to that song and either recommend more songs with shared attributes or filter any songs with those attributes out.

When Spotify launched in October of 2008, the science of music recommendation was forever changed. Spotify was the first music streaming platform to utilize collaborative-filtering to recommend new music to its users, an algorithm that was popularized by the online shopping and movie-streaming giants Amazon and Netflix, respectively. This worked by comparing a user's listening history with the entire database of users' listening histories and deriving a list of songs that were likely to be both new and well-liked by that user. With the combination of a new collaborative-filtering-based recommendation system and the platform’s freemium business model that allowed for free use of basic features with advertisements, Spotify became the premier music streaming service to millions of users worldwide.

Since the introduction of Spotify, several other music streaming platforms, namely Apple Music, Amazon Music, Google Play and Tidal, have launched in the aim to emulate Spotify or capture niche markets of consumers that Spotify hadn’t managed to convert. However, all of these music streaming platforms also use some form of collaborative-filtering in their music recommendation systems.

The motivation for this paper comes from known user frustration with Spotify’s music recommendation system. More specifically, users feel that Spotify’s music recommendations lack discoverability. This issue stems from a core weakness of collaborative-filtering, that is collaborative-filtering will only work well when there is an abundance of listening data. Known as the “cold start problem,” new songs that have not been listened to yet can’t be recommended to the user base. Additionally, any songs that are only listened to in niche groups are more difficult to recommend because of a lack of listening data. Since the majority of Spotify’s music catalog is comprised of niche music with little accompanying listening data, the platform’s music recommendation system tends to cycle through the same popular music and deliver recommendations that are fairly predictable with only the occasional new song for the user.

II. Objective

The objective of this paper is to determine if new content-based music recommendation systems supported by the unsupervised latent feature-mapping of music meta-data potentially offer the solution to the problems faced by Spotify's current collaborative-filtering based music recommendation system. By combining the capabilities of content-based music recommendation with an unsupervised latent feature-mapping, this hypothetical music recommendation system could recommend both new and undiscovered songs to its users based on a learned embedding of the songs' meta-data, effectively eliminating the "cold start problem" currently faced by Spotify.

III. Methodology

To create our database, we employ Spotify's web API where we can extract any number of songs with accompanying meta-data. Spotify's API includes a multitude of meta-data attached to each song ranging anywhere from how danceable that song is to what pitch a specific note in the song is. The core libraries that Spotify provides for music analysis are "Get Audio Features" and "Get Audio Analysis" in the Tracks API console. There we can find arbitrary data features such as "Danceability" and "Energy" or high-level features like "Timbre" and "Pitch," respectively.

The selection of features is trivial, as any redundancy or multicollinearity present in the dataset will be removed when the dimensionality reduction techniques are applied in IV.

Therefore, we select all of the features that we believe may explain some variation in the data. To that end we select a total of 12 features to use in our dataset: Danceability, Energy, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Popularity, Pitch and Timbre. The following is a compilation of Spotify's descriptions of the selected features:

Danceability: *"Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable."*

Energy: *"Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy."*

Loudness: *"The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db."*

Speechiness: *"Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely* *of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks."*

Acousticness: *"A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic."*

Instrumentalness: *"Predicts whether a track contains no vocals. 'Ooh' and 'aah' sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly 'vocal'. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0."*

Liveness: *"Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live."*

Valence: *"A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)."*

Tempo: *"The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration."*

Popularity: *"The popularity of the track. The value will be between 0 and 100, with 100 being the most popular. The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently."*

Pitch: *"Content is given by a 'chroma' vector corresponding to the 12 pitch classes C, C#, D to B, with values ranging from 0 to 1 that describe the relative dominance of every pitch in the chromatic scale (e.g. a C Major chord would likely be represented by large values of C, E and G [i.e. classes 0, 4, and 7]). Vectors are normalized to by their strongest dimension (e.g. noisy sounds are likely represented by values that are all close to 1 and pure tones are described by one value at 1 [the pitch] and others near 0)."*

Timbre: *"The quality of a musical note or sound that distinguishes different types of musical instruments, or voices. Represented as a vector that includes 12 unbounded values roughly centered around 0 (e.g. the first dimension represents the average loudness of the segment, the second emphasizes brightness, the third is more closely* *correlated to the flatness of a sound, the fourth to sounds with a stronger attack, etc.). Best used in comparison with each other"*

While the first 10 of our selected features are static and apply to the entirety of each song, both Pitch and Timbre are time-sensitive, with each vector representing only a segment of a song that may only last a few milliseconds. Therefore, it is necessary for our data to be represented as multiple multivariate time-series. The central challenge in converting the database to time-series format is that every song has a different number of segments and it would not be possible or appropriate to interpolate the data because each song has a different duration length and hence, are on different time scales. To solve this problem, we represent the time scale of each song as a percentage of total duration (e.g. a segment starting at 18 seconds in a 3-minute song is 10% into the song). In addition, we dictate a specific number of increments to break each song into, averaging the segment data within each increment (e.g. if a 3-minute song is broken down into 10 increments then all segments between 0 second and 18 seconds will be averaged into the first increment). To save memory, we chose a number of 10 increments per song.

With the feature data now formatted correctly, we begin the extraction of song meta-data from Spotify's API. In addition to the feature data, we also include each song's ID, artist ID, song name, and genre for indexing purposes. For the sake of data balancing as well as for future visualizations and comparisons, we decide to pull an even number of songs from 20 different genres: Alternative, Bluegrass, Blues, Classical, Country, EDM, Folk, Funk, Hip-Hop, Indie, Jazz, Latin, Metal, Pop, Punk, R-N-B, Reggae, Rock, Rockabilly, and Soul. Due to time limitations regarding the expiration of the Spotify API's access tokens, which expire after 1 hour, we were constrained to only extracting 200 songs per genre. This, after dropping all the duplicates, results in a database of 3251 unique songs and their meta-data.

In order to create our latent-feature mapping of our music meta-data, we propose 3 methods of dimensionality reduction: Principal Component Analysis (PCA), Autoencoder, and t-Distributed Stochastic Neighbor Embedding (TSNE). These methods can be used alone or in combination to create a low-dimensional embedding of our high-dimensional data.

Principal Component Analysis (PCA) is a linear transformation that finds the directions of maximum variation in high-dimensional data and projects it into a new subspace with the same or fewer dimensions than the original space. The new directions that contain the maximum variation are called principal components, all of which are orthogonal to the others.

Autoencoders are a family of neural networks in which the inputs are also the outputs. The goal of autoencoders is to learn to reconstruct the input after compressing it into some latent space. The advantage of autoencoders over PCA is that autoencoders are able to preserve non-linear relationships between high-dimensional data in the low-dimensional subspace.

TSNE is an algorithm that creates embeddings of high-dimensional data points to a lower dimension. The goal of TSNE is to preserve local structure from the high-dimensional space in the low-dimensional space. In doing this, TSNE effectively maintains the spacial structure within neighborhoods of data points, but it fails to preserve distances between different neighborhoods.

IV. Experiments & Results

We begin our experiments by testing each dimensionality reduction technique individually before testing them in combination with each other. We start with PCA which will act as our baseline model. We will then move on to test our Autoencoder, TSNE, PCA + TSNE, and lastly our Autoencoder + TSNE. We standardize our database prior to each experiment. At the end of each experiment we present the final result in the 2-dimensional latent space with the data points labelled by their respective genre. We will analyze the result of each experiment in V.

Our PCA model takes in our flattened time-series data and outputs 2 principal components. The explained variance of the 2 components is 29%. The output of our PCA model is displayed below in Figure 1:

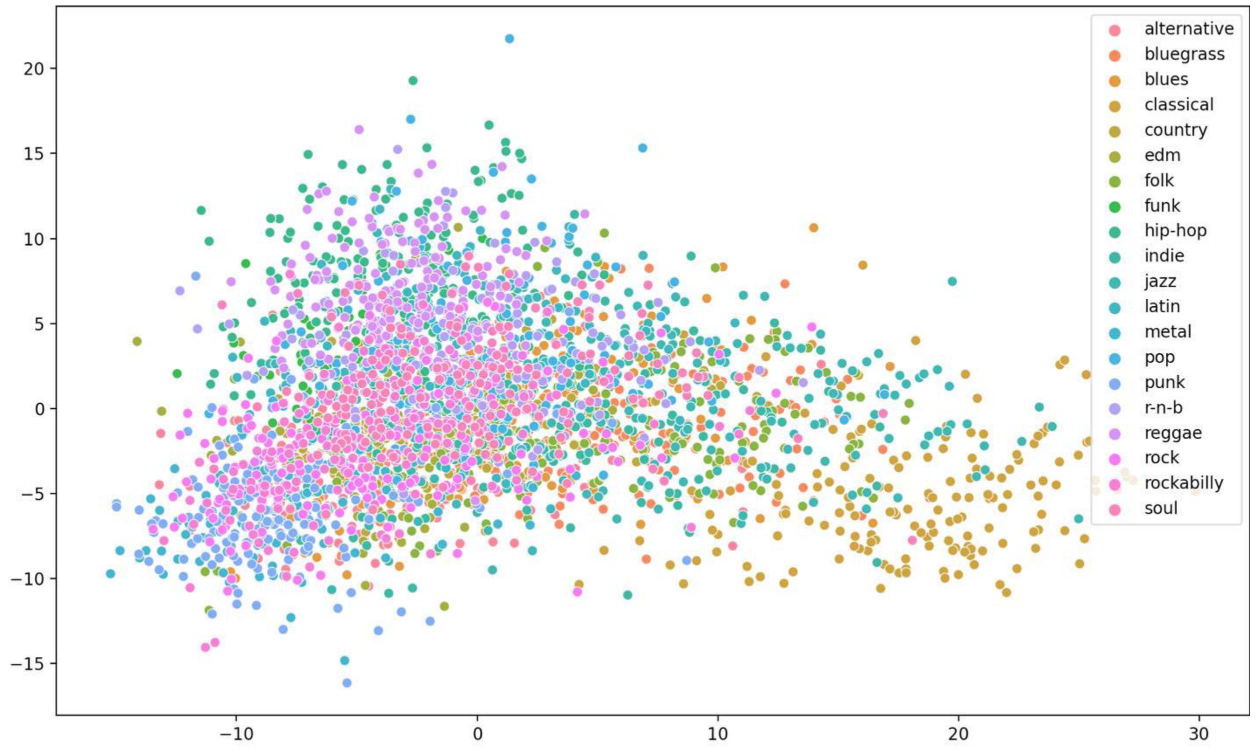

Figure 1: PCA output

Our autoencoder takes in our flattened time-series data and outputs a reconstruction of the same data. To get our latent feature-mapping, we split our autoencoder into an encoder and a decoder. The encoder takes in our inputs and outputs our data in latent space. Our autoencoder's architecture is: 340, 256, 128, 64, 32, 2, 32, 64, 128, 256, 340, where 340 is the length of our flattened time-series data. It uses Relu as its activation function, He Uniform as its initializer, Adam as its optimizer with a learning rate of 0.001, and the loss function is Mean Absolute Error. It trains for 400 epochs with a batch size of 32 and has a final validation loss of 0.5994. The learning curve for our autoencoder is shown in Figure 2 and the output of our autoencoder is presented in Figure 3:

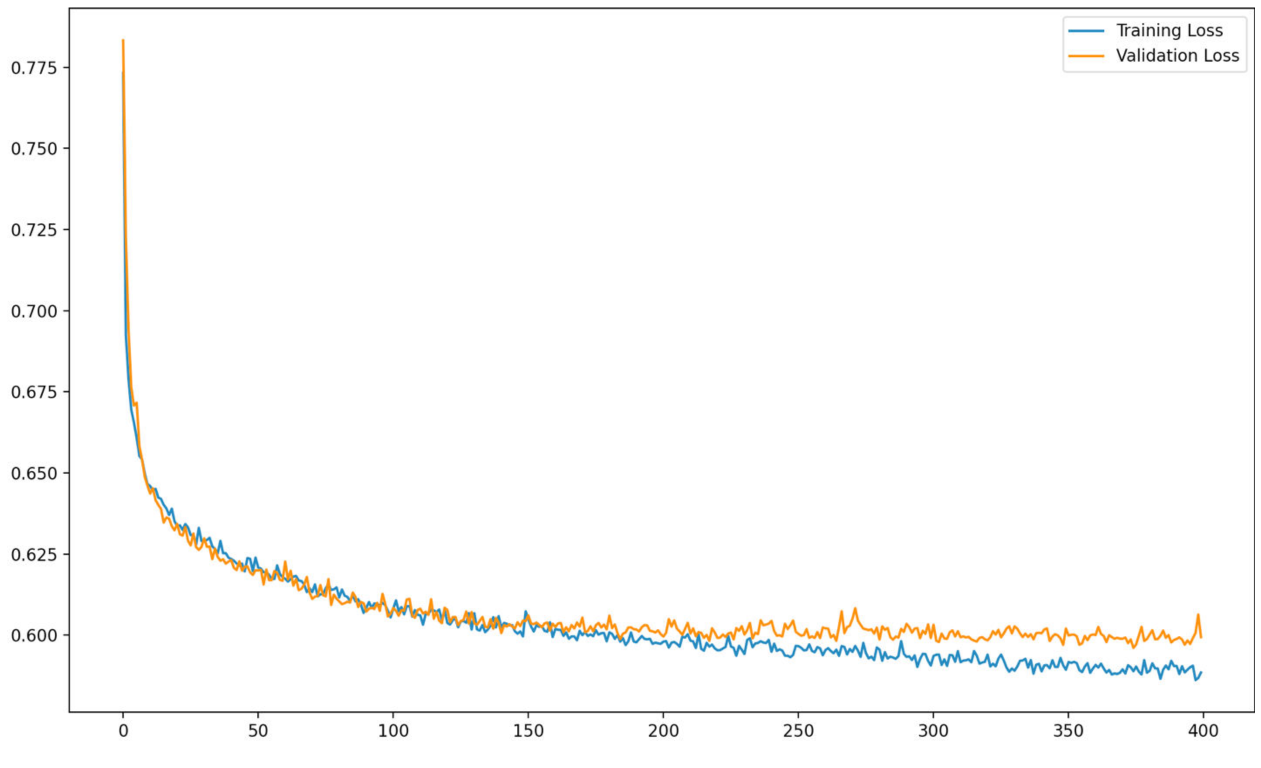

Figure 2: Autoencoder learning curve

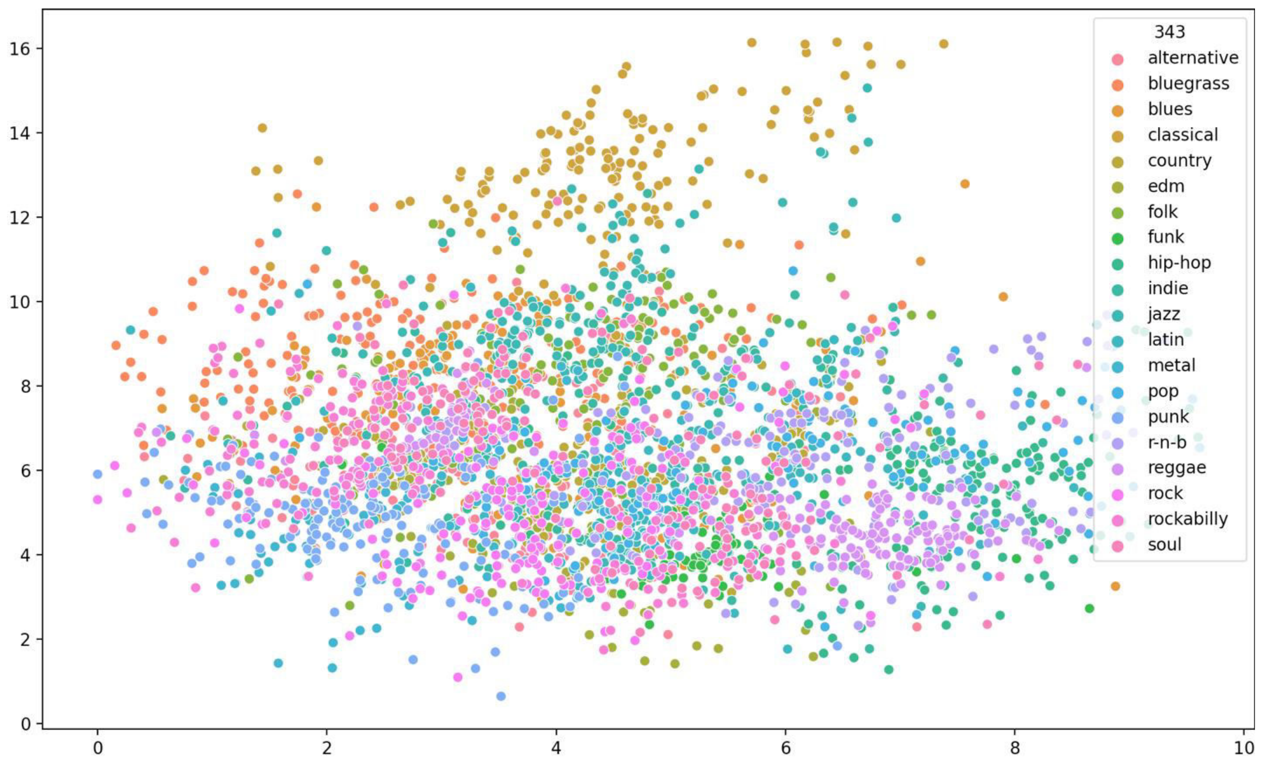

Figure 3: Autoencoder output

Our TSNE model takes in our flattened time-series data and outputs our data in 2-dimensional latent space. It uses a perplexity of 40, early exaggeration of 12, a learning rate of 600, and it runs for 1000 iterations. The output of our TSNE model is shown in Figure 4 below:

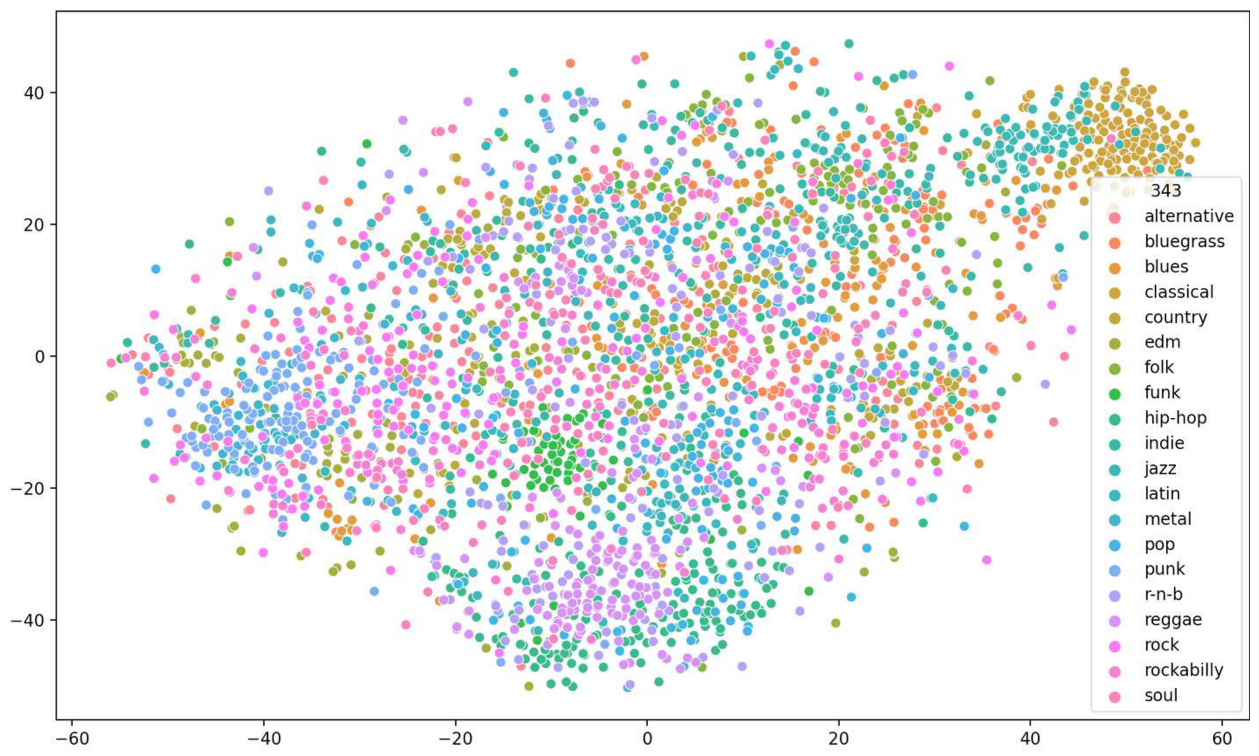

Figure 4: TSNE output

Our PCA + TSNE model works in 2 parts. First the model takes in our flattened time-series data and utilizes PCA to compress the data into 50 components, which explain 85% of the variation in our data. Then, the model takes in the 50 principal components output by PCA and uses TSNE to output a 2-dimensional representation of our data in latent space. All factors in the TSNE portion of the model remain identical to the basic TSNE model. The output of the combinatorial model is displayed in Figure 5:

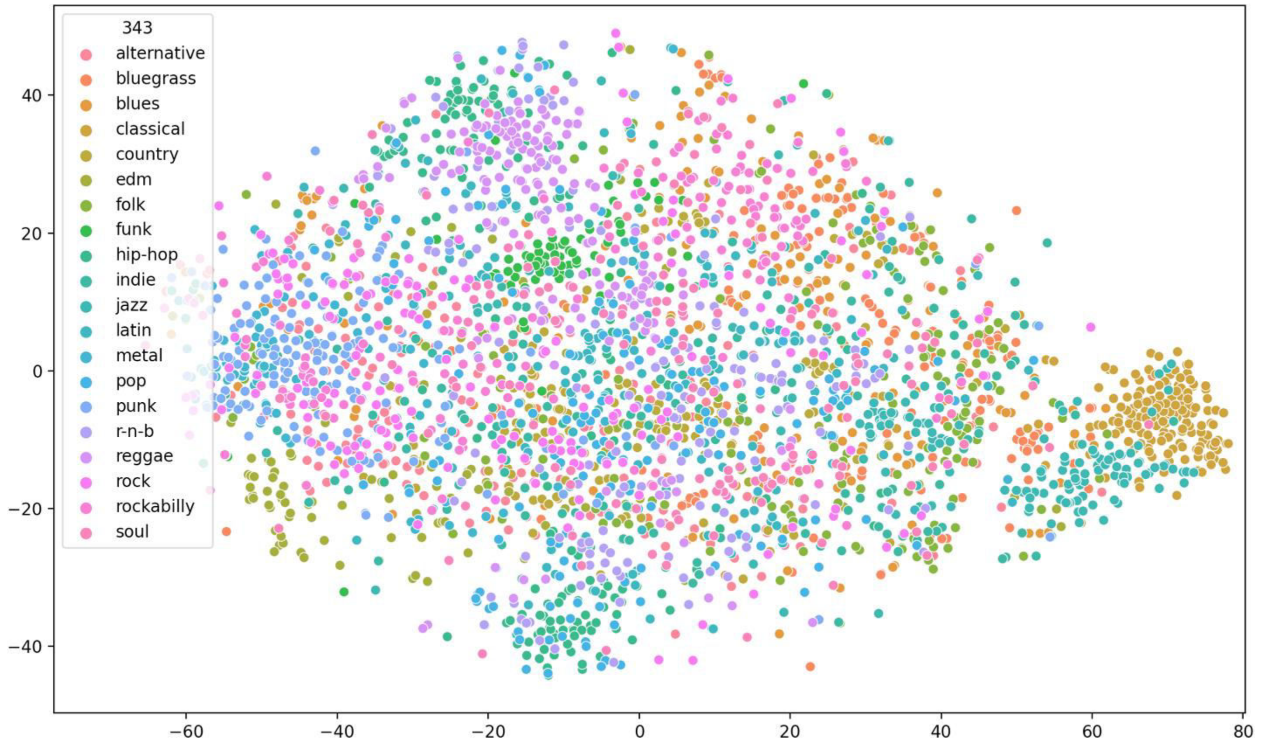

Figure 5: PCA + TSNE output

Similar to our previous experiment, our Autoencoder + TSNE model works in 2 parts. First the model takes in our flattened time-series data and employs an autoencoder to learn a 50-dimensional latent representation of our data. For this model, our autoencoder's architecture is: 340, 256, 128, 64, 50, 64, 128, 256, 340. It trains for 400 epochs with a batch size of 48 and has a final validation loss of 0.3172. All other parameters in the model remain the same as the original autoencoder model. Next, the model takes in that 50-dimensional data and applies TSNE to output our data in 2-dimensional latent space. The parameters in the TSNE section of the model are identical to the previous 2 tests. The learning curve for our autoencoder is displayed in Figure 6 and the output of the combinatorial model is shown in Figure 7:

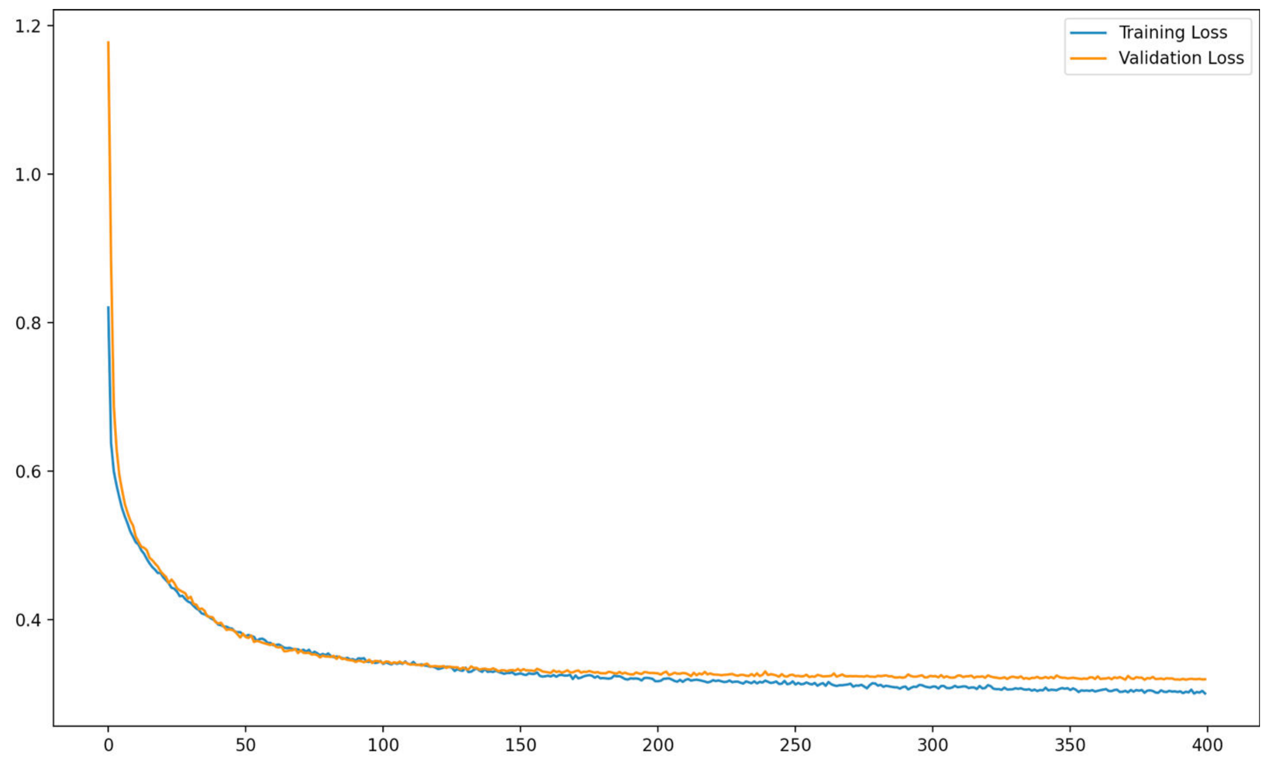

Figure 6: Autoencoder learning curve

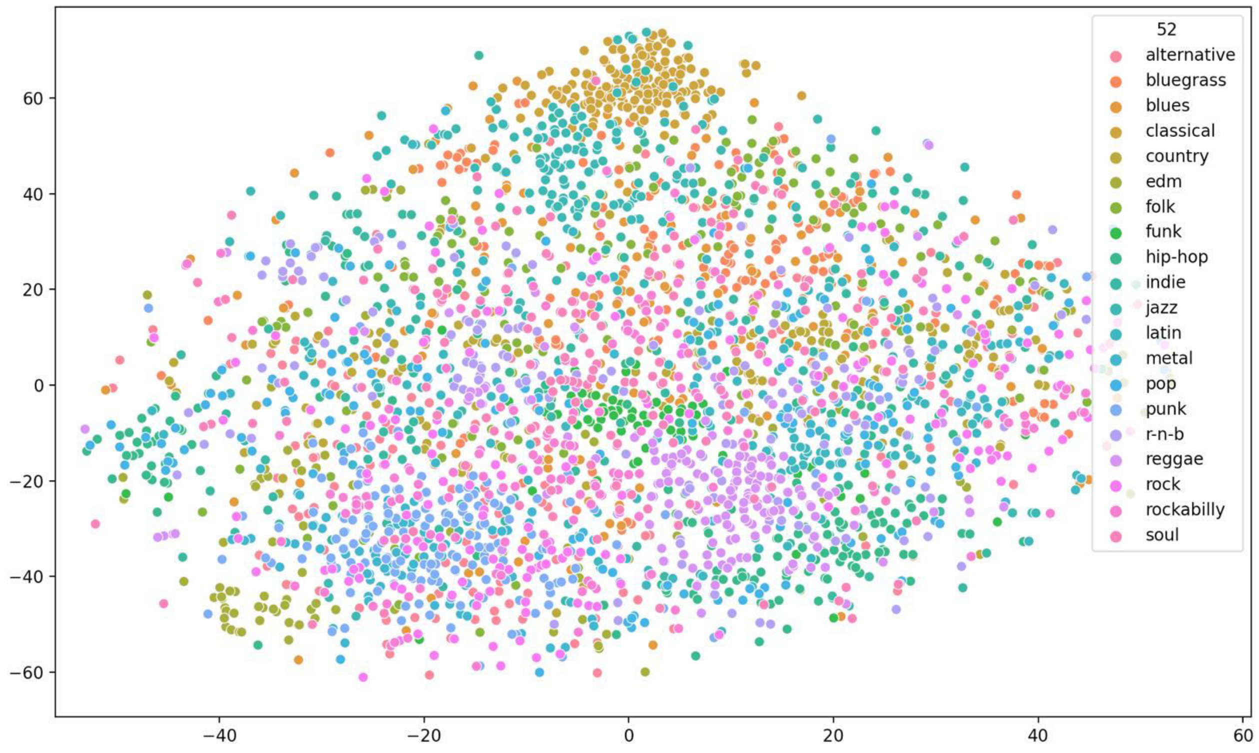

Figure 7: Autoencoder + TSNE output

V. Analysis

The analysis of the results of our experiments will be qualitative because the nature of these experiments is subjective. We do not collect any hard data for comparison. Instead, we examine each experiment's result and determine the effectiveness of each based on 3 key factors: whether well-defined clusters are formed, whether the over-arching neighborhoods of data points make sense, and whether the songs in the local neighborhoods sound similar to their neighbors. We start with the analysis of PCA, which will act as our baseline for the rest of the experiments.

The result for our experiment with PCA for dimensionality reduction can be seen in Figure 1. Observing the plot of the latent space in Figure 1, we find that most data points are clustered together in a dense ball with a sparse tail of data points spread out to the right. Due to the nature and linearity of the PCA algorithm, no well-defined clusters are observed. However, there are noticeable offshoots of data points in three separate locations that may indicate the presence of natural clusters in the high-dimensional data. These offshoots also seem to be mostly homogenous by genre, with the long tail to the right being made up of primarily classical and jazz music and the other offshoots being made up of hip-hip, reggae, punk and rock music.

The over-arching neighborhoods in the PCA results don't make much sense, most likely because the space is made up of 2 principal components that only explain a small part of the variance in the data. The data is so compressed to the principal axes that any homogenous neighborhoods of music or intelligently grouped neighborhoods (e.g. r-n-b music grouped with soul music) are either non-existent or overlapping with the others. The only evidence of homogeneity is observed on the edges of the latent space.

Similarly, the PCA embedding fails to group similar sounding songs together. For instance, a 1 x 1 space observed in the embedding includes a heterogenous mixture of almost every genre available in the database. Listening to a sample of the songs in this space, there are no distinguishable aspects in any of the songs that would make one consider the group as sounding similar.

The result for our experiment with our autoencoder for dimensionality reduction can be seen in Figure 3. Observing the latent space in Figure 3, we find a more spread out representation of the music data than in Figure 1. For instance, the offshoots observed in the PCA results are more pronounced, with the group of classical music essentially forming its own sparse cluster. While not perfectly defined, we can observe that there are at least 10 semi-defined and homogenous clusters present in the embedding. These are represented by the following genres (moving counterclockwise in Figure 3): classical, jazz, bluegrass, rockabilly, punk, rock, soul, funk, reggae, and hip-hip.

The over-arching neighborhood of data points in our autoencoder result makes significantly more sense than in PCA. Specifically, almost all major homogenous clusters border clusters of a similar genre. For example, funk is predominantly surrounded by soul, r-n-b, reggae and hip-hop clusters.

The local neighborhoods of songs also seem to make sense to some degree. While not perfectly similar in terms of instrument use, vocals, or genre, local neighborhoods of songs do tend to follow a general vibe or mood. For instance, on decently sized subspace in the embedding represents sad songs, with the “degree” of sadness increasing in one direction.

The result for our experiment with TSNE for dimensionality reduction can be seen in Figure 4. Observing the latent space in Figure 4, we find many of the same clusters found in our autoencoder result. One key difference between the two embeddings however is that the TSNE embedding manages to tighten the clusters more so than the autoencoder does. With the tightening of the latent space as a whole, we can now observe even more semi-defined clusters in the TSNE latent space than in previous results. For instance, we can now make out clusters for country, pop, latin, and folk.

The over-arching neighborhood of data points, while more defined, are comparable to that of the autoencoder embedding. As for the local neighborhoods of data, songs within these neighborhoods follow a similar pattern to those in the autoencoder embedding, specifically that they tend to be grouped by the vibe that they give off more so than by any distinct feature like instruments used or play-style.

The result for our experiment with the combination of PCA and TSNE for dimensionality reduction can be seen in Figure 5. Observing the latent space in Figure 5, we find a representation of our music data that is very similar to our TSNE embedding in Figure 4. One key difference between Figures 4 and 5 is that the music data seems to be more evenly distributed within the embedding, with data points seeming to be approximately equidistant between from their nearest neighbors. Another key difference is the emergence of a semi-defined cluster for EDM.

The over-arching neighborhood of data points are comparable to that of the TSNE embedding, maintaining fairly defined neighborhoods of data. As for the local neighborhoods, songs within these neighborhoods also follow similar patterns to those in the TSNE embedding. However, the similarity between neighboring songs is significantly improved, with many songs by the same artists tending to be near each other. The embedding still seems to be indifferent to distinct features like instrument-use and vocal pitch, but the result a significant step in the right direction.

The result of our experiment with the combination of our autoencoder and TSNE for dimensionality reduction can be seen in Figure 7. Observing the latent space in Figure 7, we find a representation of our music data that closely resembles our TSNE and PCA + TSNE embeddings. The most noticeable difference in Figure 7 from the previous 2 results is the rejoining of the classical/jazz music cluster seen in previous embeddings with the conglomerative blob of data points and semi-homogenous clusters. In fact, all of the off-shooting clusters seen in the previous results have sunk back into the collective blob of data. However, even with this difference in the embedding, the data still maintains the semi-homogenous clusters seen in previous results.

The over-arching neighborhood of data points are comparable to that of the previous 2 embeddings, appearing virtually identical to the structures observed in both the TSNE and PCA + TSNE results. As for the local neighborhoods of music data, the songs within these neighborhoods follow a very similar pattern to that in the PCA + TSNE embedding, that being the similarity in mood, but also in often clustering songs by the same or similar artists near each other. The embedding also seems to be fairly indifferent to distinct features of each song, tending to group songs by features like vibe that are most likely components of the songs' tempos, valences, and timbre vectors.

VI. Conclusion

The current state of music recommendation in the global music streaming industry suffers from a fundamental weakness stemming from the collaborative-filtering techniques that it employs. The inability of these systems to recommend new and undiscovered music to the platforms' users constitutes a growing frustration within the consumer base. The analyses of our experiments studying the potential for the support of unsupervised latent feature-mapping of music meta-data in content-based music recommendation systems uncovered intriguing results. We have shown that the potential for unsupervised latent feature-mapping of music data via traditional dimensionality reduction techniques is valid and may offer a solution to the "cold start problem" faced by the music streaming industry today. Given more time to experiment, it would be appropriate to further investigate to what degree some latent space can differentiate different pieces of music and to what extent that it could be used in regard to content-based approaches to music recommendation.

References

Aucouturier, Jean-Julien, et al. "(PDF) The Way It Sounds: Timbre Models for Analysis and Retrieval of Music Signals." *ResearchGate*, 2005, www.researchgate.net/publication/3424372_The_way_it_Sounds_timbre_models_for_analysis_and_retrieval_of_music_signals.

Limited, Spotify. "Web API Reference." *Spotify for Developers*, 2018, developer.spotify.com/documentation/web-api/reference/#category-tracks.

Koh, Mark, director. *Audio Analysis with the Spotify Web API*. *YouTube*, 3 Feb. 2018, youtu.be/goUzHd7cTuA.

Madathil, Mithun. "Music Recommendation System - Spotify Collaborative Filtering and Feedback System." Umea University, 2017.

Van den Oord, Aaron, et al. "Deep Content-Based Music Recommendation." Ghent University, 2013.