Neural Language Architectures for Music Information Retrieval and Recommendation

Amit Gattadahalli, Tres Pimentel
amitg3@berkeley.edu, trespimentel@berkeley.edu

ABSTRACT
State of the art (SOTA) music recommendation systems typically use a variety of features including song audio, historical user data, playlist analysis, and features derived from natural language processing (NLP) techniques that investigate how people talk about certain songs online. This approach is both architecturally complex and computationally expensive. In this paper, we investigate the ability of lyrics and associated neural language architectures for detecting several key characteristics of songs that can be further operationalized for use in both content-based and collaborative music recommendation as a supplement approach to current SOTA systems. We scoped the problem to prioritize primary language detection, topic density estimation, and genre classification as key tasks.

INTRODUCTION
Platforms such as Spotify, Apple Music, and Pandora have cemented their place in the music and entertainment industry with their ability to deliver high quality music streaming to customers on-demand. In addition to returning user-requested music, one of primary drivers that has enabled each of these platforms to grow into multi-billion dollar giants is the ability to generate logical musical recommendations according to an analysis of user preferences. This capability has enabled various features such as custom radio stations and autonomously curated playlists, while also minimizing user responsibility for manually selecting their own music.

Music recommendation is a complex field that lies at the intersection of music information retrieval, preference based prediction, and geographic analysis with an added priority of identifying similar content in a big data ecosystem. While NLP is an important discipline within data science and music, it is not frequented as a primary driver of a music recommendation system - although this notion is beginning to change [6]. Instead, current SOTA approaches largely prioritize utilizing audio data to detect various sound-centric musical characteristics before making similarity-based recommendations. While these approaches have proven effective, they are computationally expensive in terms of data preprocessing and model training/development.

The exclusion of processing song lyric data articulates a gap in current systems abilities to make recommendations that include consideration of song meaning and language choice, which could in turn provide valuable data for serving improved recommendations to users. As an alternative method, we look to leverage both classical machine learning (ML) and deep learning (DL) approaches in order to investigate the viability of song lyrics and associated neural language architectures for detecting key characteristics that can serve as primary inputs for music recommendation. In particular, we look to explore the ability of utilizing lyrics for detecting primary language, topic density, and genre to provide computationally lightweight supplements to SOTA recommendation systems.

DATASET DESCRIPTIONS
As a result of modern efforts in music recommendation largely prioritizing usage of audio data for musical characteristic determination and recommendation, our team's initial research in viable labeled datasets for model development showcased a similar sentiment. While popular datasets do exist for music information retrieval, such as George Tzanetakis' GTZAN [7] dataset and the Million Songs Dataset [8] developed by researchers at Columbia University, a vast majority of these datasets contained audio-based features in various tabular and unstructured formats (mel spectrograms) while either having incomplete or no lyric data/features for songs. With limited data availability and usage of Genius.com's lyricsgenius Python API to provide more complete lyric data, we utilized the following Kaggle datasets for model development and evaluation on our priority detection tasks.

Language Detection
Song Lyrics from 79 Musical Genres [9]
Description: 383570 songs, complete lyric data, language labels for 52 languages (downsampled to 11 classes: Spanish, Portuguese, English, Kinyarwanda, Italian, French, German, Other, Finnish, Swedish, Romanian), artist name, song name

Genre Classification
Spotify Multi-Genre Playlist Data [10]
Description: 26753 songs, audio features provided by Spotify, 7 genres (Alternative, Indie, Rock, Hip Hop, Pop, Rock, Metal), 2000+ sub-genres, incomplete lyric data (supplemented with lyricsgenius API), artist name, song name, song duration

Topic Density Estimation [11]
Music Dataset: 1950-2019 → Derived From → Original Dataset Here
Description: 28372 songs, Subset of Lyrics (stopword removal and lemmatization was applied by author, reversed with lyricgenius API), audio features provided by Spotify, genre, topic density across 16 topics (downsampled to 11 topics: Family/Religion, Music, Violence, Explicit Content, Dating/Love, Emotion, Travel/Money, Time, Sensory Perception, Energize Audience, Other) provided by Latent Dirichlet Allocation (LDA) Topic Model, hard class topic assignment based off max predicted topic density

DATA PREPROCESSING, MODEL DEVELOPMENT, & RESULTS ON DETECTION TASKS

Language Detection
For this specific task, our goal was to predict a song's primary language given its lyrics. Upon identifying our dataset of choice for model development, initial EDA yielded a severe class imbalance across the 11 downsampled classes. To both speed up training time and avoid overinfluence of classes with higher label counts on loss functions and associated learned decision boundaries, our team's first treatment for class imbalance was downsampling from our majority classes to provide a modified subset of the initial dataset. In particular, 3000 examples were randomly sampled from each of the English, Portuguese, and Spanish classes respectively while all examples were retained from the remaining 8 classes. Then, a randomized 80-20 Train - Validation/Test Split was executed where 80% of the examples in the subset were retained for the train set and 20% of the examples were retained for the validation/test set with the first half of examples being reserved for the validation set and the latter half being reserved for the test set. Our final treatment for leftover class imbalance was to introduce class weights computed as:

$$class\ weight\ i\ = \frac{\#\ of\ Training\ Examples\ in\ Majority\ Class}{\#\ of\ Training\ \ Examples\ in\ Class\ i}$$

This class weights treatment allowed each class to have an equal influence over the loss function to be minimized during training.

	Original Dataset		Resampled Train Set		Resampled Val/Test Set		
Language Label	Count	Percentage	Count	Percentage	Count	Percentage	Class Weights
English	191812	52.51%	2371	19.53%	629	20.73%	1.017
Portuguese	157393	43.09%	2405	19.81%	595	19.61%	1.003
Spanish	9917	2.71%	2412	19.87%	588	19.38%	1
Kinyarwanda	1679	0.46%	1332	10.97%	347	11.44%	1.81
Italian	1432	0.39%	1156	9.52%	276	9.10%	2.09
French	1225	0.34%	968	7.98%	257	8.47%	2.49
German	844	0.23%	700	5.77%	144	4.75%	3.45
Other	638	0.17%	509	4.19%	129	4.25%	4.74
Finnish	145	0.04%	114	0.94%	31	1.02%	21.16
Swedish	112	0.03%	97	0.80%	23	0.76%	24.87
Romanian	97	0.03%	74	0.61%	15	0.49%	32.59

Figure 1: Label Percentages in Original Dataset vs Resampled Train/Validation/Test Set, Class Weights During Training

Following the creation of labeled lyric/language train and validation/test sets, text preprocessing steps to clean lyric data included casting all lyrics to lowercase, removal of line breaks (‘\n’), and removal of double spaces.

Further exploratory data analysis (word cloud generation) of lyrics segmented by language label yielded the initial model hypothesis that embedding-based models whose layers construct meaning-based representations of lyrics may not be well suited for the task of language detection, and simpler approaches which consider word token choices may have stronger generalization capabilities out of sample.

Figure 2: English (Left) vs Spanish (Right) Wordcloud of Song Lyrics, Size Indicates Term Frequency

Assessments of lyric wordclouds conditional on language labels supported this hypothesis due to evidence that word tokens from one language to another are reasonably unique. Even language pairings with high degrees of lexical similarity such as Spanish - Portuguese (89%) and Spanish - Italian (82%) (lexical similarity scores provided by The Language Doctors [12]) contained enough spelling variation between shared loan words to allow for differentiation between song examples with alternate labels. Our team’s featurization process of choice for capturing language choice was term density features, a simple variant of bag of words, across a corpus of words defined by lyrics in all training examples outside of the “Other” language category. While bag of words featurization captures the number of times a word appears in a text example, term density captures the following relationship:

$$Term\ Density\ for\ Word\ X\ in\ Example\ = \frac{\#\ of\ Times\ Word\ X\ Appears\ in\ Example}{\#\ of\ Word\ Tokens\ in\ Example}$$

Feature representation in this manner had many immediate benefits for our language detection task. First, this enabled examples belonging to non-Other language labels to have high values in feature subspaces that captured term density for words that belong to that language while having low values in feature subspaces that captured term density for words that belong to different languages. For examples that fell in the “Other” category, this resulted in low term density values across all features. Holistically, this enabled a reasonable separation of classes in feature space that would enable simple neural architectures to develop highly generalizable decision boundaries during the model training process. Term density features also enabled model generalizability across examples of variant length by representing all examples on similar scale whereas the performance of a bag of words model would have been highly dependent on the token length of examples it was trained on.

Following the featurization of our training, validation, and test examples into term density representation, we trained a shallow neural network in Tensorflow/Keras with 2 hidden layers (100 nodes each, ReLu activation) and 1 output layer (11 nodes, softmax activation) to identify model parameters that minimized the categorical cross entropy loss on our training data. The performance metric we aimed to maximize was measured via average class recall defined as:

$$average\ class\ recall\ = \frac{(\sum_{i=1}^{k} \frac{\#\ of\ correctly\ predicted\ exampled\ in\ class\ i}{\#\ of\ examples\ in\ class\ i})}{\#\ of\ classes}$$

This metric allowed us to optimize the performance of our model in detecting each class out of sample with the goal of comprehensively being able to detect a high proportion of all classes in aggregate.

Our optimal term density model was trained with a learning rate of 0.005 and a batch size of 8, and was able to achieve a validation accuracy and average class recall of 0.9796 and 0.9702 respectively. In parallel, to further assert our hypothesis that term density representation models would outperform embedding representation based models, we also trained a competing Deep Averaging Network (DAN) and Weighted Averaging Network (WAN) to compare performance. Both networks were initialized with a custom embedding matrix in 300 dimensional embedding space, with a corresponding row for every word in the term density training corpus in addition to a row for mapping unknown tokens (121320 rows). Our DAN model computed the centroid of the embeddings for the first 1000 tokens in a song to build a single vector embedding representation of the input before passing this vector into an identical feed-forward architecture as our term density model. While the WAN model also featured an identical final feed-forward architecture, it utilized a modified attention-based approach for building an embedding based representation of the input. Following the identification of embeddings for the first 1000 tokens of an input example, our WAN model utilized an attention layer with 10 nodes, where each node represented a query vector while the embeddings represented key and value vectors to identify how much weight to attribute to each embedding when building a vector representation. The intuition of multiple attention nodes in a layer was similar to that of a Convolutional Neural Network (CNN) with multiple kernels of the same size acting on the same input. While a CNN setting allows these kernels to learn to identify different features due to random initialization, this attention layer in parallel aims to identify different optimal ways of combining embeddings into single vector representations. The output of this attention layer was a 10 x 300 dimensional matrix representing 10 different attention-based embedding representations of the input. A singular attention query was applied to these embeddings in order to develop a final embedding representation of the attention layer output before proceeding into the feed-forward portion of the network for classification. The results of these competing models on the validation set were:

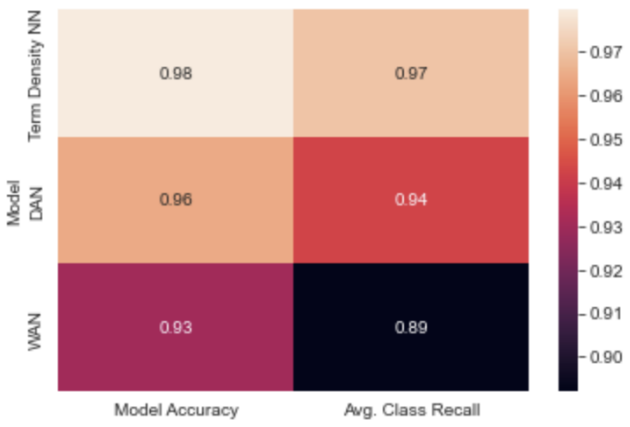


Figure 3: Language Detection Model Performance Comparison

As hypothesized, the term density model outperformed the embedding-based models in both overall model accuracy and class recall, indicating that models which develop meaning-based representations of text input result in a loss of signal in the task of detecting language.

When evaluating our term density model on the test set to get an unbiased estimation of model performance, our model exhibited consistently strong results in terms of class recall, precision, and F1 with the exception of the "Other" class. This may have been a result of general class diversity as the "Other" class was composed of 42 languages with low label counts indicating the possibility that different examples in this class could have been feasibility confused with a broad range of the 10 alternative classes during prediction.

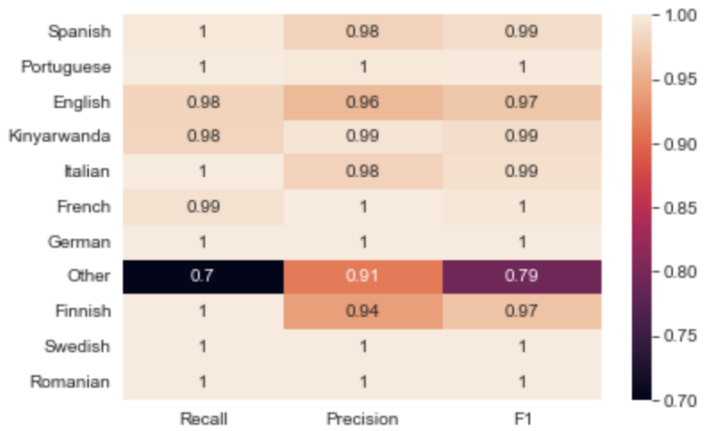


Figure 4: Class Recall, Precision, and F1 Score of Term Density Model on Test Set

Topic Density Estimation

For this task, our goal was to predict the density distribution of topics for a song given its lyrics. In particular, for a given song we aimed to identify the proportional topic density for the following 11 topics as stated above: Family/Religion, Music, Violence, Explicit Content, Dating/Love, Emotion, Travel/Money, Time, Sensory Perception, Energize Audience, Other. While unsupervised methods such as LDA exist for making this determination, we aimed to utilize the density outputs developed by the original dataset authors' LDA models as labels to recast this task into a supervised machine learning problem. The decision to predict density distribution in this context rather than a hard class assignment was based on the intuition that a given song will be made up of a variety of topics in a variable amount of proportions. Therefore, the ability to accurately detect this distribution will provide interpretable characteristics and summarization regarding the meaning of the song with minimal information loss in comparison to a hard class assignment which would inherently neglect the existence of subtopics. In addition, the ability to characterize topics within music will enable the encoding of lyric-based semantic meaning in lower-dimensional space as an input feature in recommendation.

Due to the initial incompleteness of lyric data for this dataset, we utilized the lyricsgenius API to scrape lyric data for the songs within our overall dataset. Of the original 28372 songs, this process successfully identified lyrics for 23143 songs. Random sampling of 100 songs both under and over 1000 word tokens in length yielded a 100% accuracy rate in successful lyrics pulled for songs under 1000 word tokens, but only a 72% accuracy rate in successful lyrics pulled for songs over 1000 word tokens in comparison. In the latter case, several examples had long text documents that were scraped rather than the song lyrics in question resulting in token counts that were in the hundreds of thousands of tokens for individual examples. This analysis motivated the decision to discard all examples larger than 1000 tokens to minimize the risk of erroneous examples in the dataset. Following additional text preprocessing including casting text to lowercase, removal of non-alphanumeric or non-punctuation data, removal of song section annotations, and removal of ad libs, there were 22272 examples remaining from the original dataset of which 18923 examples were set aside for the training set, 1705 examples were set aside for the validation set, and 1644 examples were set aside for the test set (approximately 85-7.5-7.5 Train/Validation/Test Split)

In order to measure out of sample performance in an interpretable manner, our team constructed the following performance metric dubbed average density deviation:

$$average\ density\ deviation\ =\ \frac{\sum_{i=1}^{k} |predicted\ density\ of\ Topic\ i\ -\ actual\ density\ of\ Topic\ i|}{k}$$

This metric allowed us to directly evaluate, topic agnostic, the expected deviation between our model's predicted topic density and the actual topic density for any given topic. Given the intent to maximize this metric out of sample, the natural choice for a Tensorflow/Keras supported loss function to minimize during training was the Kullback-Leibler Divergence (KL Divergence) which seeks to minimize the deviation between two distributions (P(Y) = True, Q(Y) = Predicted) across all training examples:

$$KL\ Divergence\ =\ \sum_{i=1}^{k} P(Yi)\ *\ log(P(Yi))\ -\ P(Yi)\ *\ log(Q(Yi))$$

$$KL\ Divergence\ =\ Cross\ Entropy\ (P, Q)\ -\ Entropy(P)$$

Utilization of a KL Divergence based loss function aligned particularly well to our prediction task at hand due to the goal of minimizing the deviation between predicted topic density distributions and actual topic density distributions in practice, which was a reasonable parallel to the goal of identifying model parameters that minimize deviation between distributions in training.

Unlike the language detection task, our team hypothesized that embedding-based models would be crucial for successfully executing this task due to the intuition of meaning-based representations of input being highly deterministic of proportional topic makeup. Our team experimented with 5 competing architectures to identify an optimal model that generalized best out of sample in terms of average density deviation: DANs, WANs, CNNs, DAN-WAN-CNN Hybrid (Hybrid), and Transformer (BERT).

Our DANs and WANs were identical layouts to their parallel implementation in the language detection task, with the former computing the centroid of the first 1000 token embeddings before proceeding into the feed-forward layers while the latter utilized a full attention layer with multiple attention nodes to build separate representations of the input tokens before utilizing a final attention query to return a single vector representation as input into the feed-forward layers. Our CNNs utilized kernel sizes of 3, 5, 10, and 25 (equal number of filters for each, global max pooling conditional to each kernel size) on the embeddings of the first 1000 tokens of an input to develop a word-order conscious, meaning-based representation of the input before proceeding into the feed-forward layers. Our hybrid architecture was an attention-based conglomeration of our DANs, WANs, and CNNs where we ripped off the feed-forward portions of each network architecture and used each scheme to develop a vector representation of the input. With 3 separate vector representations, the hybrid model utilized a multi-node attention layer and a final attention query (identical to the WANs) to develop a final vector representation that balanced the output of the DANs, WANs, and CNNs before proceeding to the feed-forward portion of the network. Finally, our BERT model utilized BERT (base uncased) to develop contextual embeddings of the first 512 tokens of the input before leveraging DAN, WAN, CNN, CLS, or Pooled Token based schemes to develop a single vector representation of the input before proceeding into the feed-forward layers for classification. In this context, the softmax outputs of our classification layer represented the predicted proportional density of a topic given the input.

Across the 5 competing network architectures for topic density estimation, we utilized identical feed-forward layers in order to directly assess the ability of each network for building a meaning-based representation that was deterministic of topic density distribution. Additional experimentation also included the evaluation of

embedding size (100, 300, 500, 1000) and prebuilt (Word2Vec) vs custom embeddings particularly in the case of the DANs, WANs, CNNs, and Hybrids. Results across all 5 architectures when evaluated on the validation set are shown below.

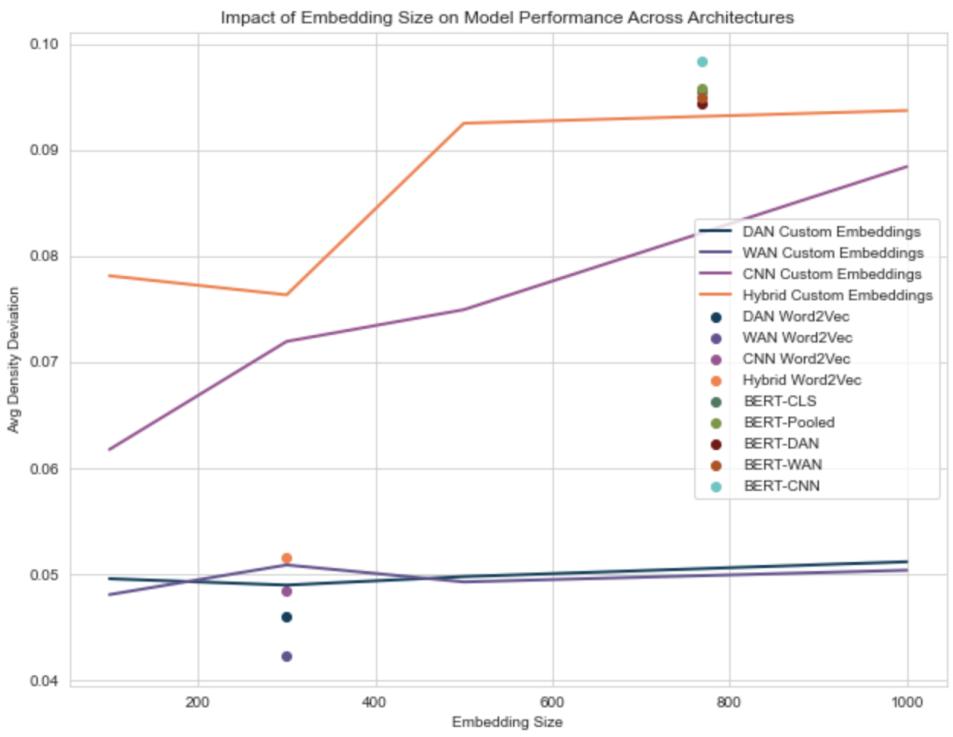


Figure 5: Topic Density Model Performance Comparison on Validation, Impact of Embedding Size

When comparing model performance across architectures and various embedding sizes, increasing embedding size tended to have an adverse effect on predictive performance both out of sample and in sample. This could have been a product of insufficient data as larger embedding sizes equate to larger network architectures and ultimately more parameters that must be optimized during the training process. This was further highlighted when evaluating the performance of BERT-based architectures which significantly underperformed in comparison to the other 4 competing architectures. While BERT and transformer architectures in general tend to be the SOTA across many predictive language tasks, they also need larger amounts of data to fine-tune due to large network sizes. Finally, this analysis highlighted the benefit of both transfer learning and attention mechanisms as our WAN-Word2Vec Model performed best out of sample with an average density deviation of 0.0422 and WAN architectures generally outperforming competing architectures during experimentation.

Following the selection of an optimal model (WAN-Word2Vec), our team evaluated this model on the test dataset to get an unbiased estimation of model performance while also directly comparing to the baseline of predicting equal density across all topics. Our model achieved an average density deviation of 0.049 on the test data in comparison to the baseline performance of 0.108, representing an overall 54.52% improvement over the baseline. In addition, when evaluating model vs baseline performance and error conditional on topics, our model achieved between a 39.65% - 74.85% improvement over the baseline dependent on topic.

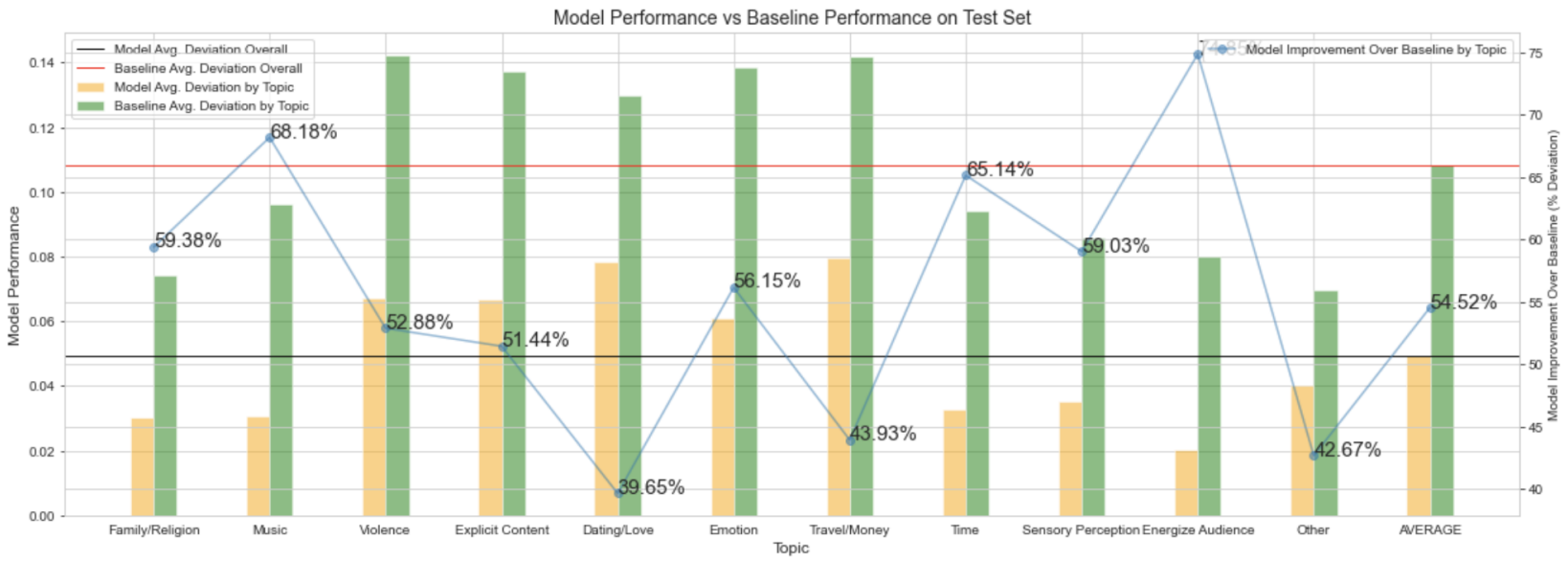


Figure 6: Model Performance on Test Set vs Baseline (Predicting Equal Density Across All Topics)

Genre Classification

Music genre classification is a common problem currently being researched by professionals and academics within the music information retrieval industry [2][3]. Frequent approaches primarily consist of training ML models on labeled mel-spectrograms of songs, which provide a visual image-based representation of audio frequency and sound waves. These computer vision models serve as the current SOTA for genre classification when evaluated out of sample[5]. While our team does not hypothesize that models primarily using lyrics as features will outperform audio-to-image based genre classification models, we sought to explore whether the introduction of lyric based features could be used as supplemental features to improve upon solely audio-to-image based models for genre classification and ultimately recommendation. Throughout the course of our experimentation, statistical audio features (valence, tempo, loudness, energy, danceability, etc.) provided by Spotify were used as a proxy for mel-spectrograms as input into neural architectures.

The multilingual dataset used for model development and evaluation includes seven primary genres: Rock, Metal, Alternative, Pop, Hip Hop, Indie, and Blues. Over 2000 sub-genres were also provided with this dataset which could potentially support the development of multi-label genre classification models, but for the sake of scoping this problem we elected to prioritize the ability to classify a song into one of the seven primary genres.

Several competing networks were selected for experimentation in this task, including embedding-based models (DANs, WANs), term density based models, audio statistics based models, and multimodal schemes which ingested features across language and audio mediums into a single network architecture. A multitude of text preprocessing steps were also included as part of the model exploration process to determine sequences of data cleaning schemes that enabled optimal model performance. Examples of preprocessing steps explored include removal of punctuation, tags, artist names, case standardization, and stopword removal. Stopword

removal in particular appeared to have the greatest positive impact on model performance and was a primary text preprocessing step for the final genre classification model that achieved the highest validation accuracy and recall during evaluation.

Analysis of models seen in the figure below yielded the following observations specific to better performing models:

1. Multimodal approaches that incorporated lyric and audio features
2. Inclusion of attention mechanisms on word embeddings to develop vector representations of lyric tokens
3. Inclusion of term density features to capture proportional word choices in lyrics while deprioritizing word meaning and context
4. Term density feature models outperformed audio feature models
5. Comparable model performance between custom and prebuilt (Word2Vec) embeddings

The latter observation above proved interesting because although Word2Vec had been developed to generate suitable representations of words that preserved relationships within its corpus, it only allowed us to capture tokens for approximately half the words in our dataset. This led to the hypothesis that the reason for comparable performance using custom embeddings despite random initialization was a product of these embeddings capturing a much larger proportion of input lyrics (83000 unique words in total) while also allowing for representation of specific slang words that Word2Vec may have neglected.

Model Type	Data + Format (token quantity, embedding length)	Test Set Accuracy
DAN	Word2Vec Tokens (1000, 300)	0.177
WAN	Word2Vec Tokens (1000, 300)	0.216
DAN	Custom Embeddings (1000, 300)	0.277
WAN	Custom Embeddings (1000, 300)	0.324
Feed Forward	Audio Features	0.425
Multimodal Audio + WAN	Audio Features + Word2Vec (600,300)	0.453
Multimodal Audio + WAN	**Audio Features + Custom Embeddings (600, 300)**	**0.652**
Multimodal Audio + WAN	Audio Features + Custom Embeddings (700, 700)	0.64
Multimodal Audio + DAN	Audio Features + Custom Embeddings (700, 700)	0.417
Feed Forward	Term Density Features	0.623
Multimodal Audio + Term Density	Audio Features + Term Density	0.636
Multimodal Audio + Term Density + DAN	Audio Features + Term Density + Word2Vec (1000, 300)	0.646
Multimodal Audio + Term Density + WAN	Audio Features + Term Density + Word2Vec (1000, 300)	0.649

Figure 7: Subset of Genre Classification Models and Accuracy on Test Set

	Indie	Metal	Pop	Rock	Alternative	Hip Hop	Blues
Indie	112	2	20	22	62	3	2
Metal	2	120	1	11	1	1	1
Pop	18	2	142	12	11	14	0
Rock	26	38	24	240	52	7	15
Alternative	29	2	14	23	65	1	3
Hip Hop	2	0	5	5	0	111	1
Blues	2	1	1	12	1	2	55

Figure 8: Confusion matrix for top genre model. Actual song genres are on the rows, predicted song genres are on the columns.

The optimal genre classification model identified during model evaluation featured a multimodal approach incorporating custom word embeddings, term density, and audio features as input, and achieved 65.2% accuracy and 68.6% recall on the test set. This architecture utilized a singular attention query to identify word significance when developing an embedding-based representation of the lyrics.

As highlighted by the confusion matrix in Figure 8, genre classification is a fluid task with subjective difficulty in characterizing a song within one genre vs another. Humans, for example, often find difficulty in making genre determinations between Indie, Alternative, and Rock due to the inherent similarities across genres. This uncertainty likely resulted in labeling bias which in turn was learned by our models during the training process, which is reflected by model mis-characterizations between the Indie, Alternative, and Rock genres specifically.

With continued difficulty in attributing songs to singular genres, multilabel classification or genre density estimation approaches that aim to capture a broader perspective of a song's musical influence may prove to be more deterministic of a song's genre-based characteristics. Evidence provided by our optimal model's confusion matrix increasingly shows that this could be a particularly interesting and fruitful area of research. In addition, using this current genre classification framework as a foundation, the incorporation of more data, deeper models, variable architectures, and usage of mel-spectrograms in addition to embeddings, audio statistics, and term density may be a feasible approach for generally improving predictive ability as well as the SOTA in this task.

MUSIC CHARACTERISTIC - BASED RECOMMENDATION

With the ability to characterize language, topic, and genre utilizing neural language architectures, we aimed to operationalize this information alongside audio-based characteristics in order to provide a suitable framework for a content-based music recommendation system. In the context of music recommendation specifically, key considerations included: system streamlining to make recommendations efficiently, quantifying song similarity, and making logical recommendations that users are likely to listen to. To successfully handle the above considerations, we propose a novel clustering and nearest neighbors search based approach that incorporates song popularity to efficiently identify similar songs that a user is likely to listen to in relation to an arbitrary song that they have rated favorably.

Song Characterization - Characteristic Embeddings

This recommendation scheme would begin with a characterization of a large corpus of songs such that each song has a characteristic-based vector representation. These song characteristic embeddings would be developed by feeding song lyrics into our architectures above (alongside audio features as necessary) to determine primary language, topic density distribution, and predicted likelihood across genre. The latter is returned as opposed to a hard genre assignment to introduce more variability into the embeddings while also supporting the intuition that a song may have musical influence across genres resulting in a single genre representation to be insufficient. By returning these predictions and preserving audio features, each song will have a characteristic embedding representation that captures language, song meaning, genre, and sound information.

Characteristic Embedding Clustering

Following song characterization, these song characteristic embeddings would be grouped into logical clusters. This clustering can be accomplished using a variety of different methods including unsupervised learning (KMeans, DBSCAN, Hierarchical), language based (songs are grouped according to primary language), artist based (each artist is treated as a cluster), composite schemes that utilize a variety of clustering techniques together, etc. Following the clustering of characteristic embeddings,

each cluster will be assigned a centroid, computed as the average of the embeddings within the cluster, which will serve as the single point representation of the cluster. In practice, as new songs are introduced to the database alongside their associated characteristic representations, they can be independently assigned to their respective clusters with the primary impact being modifications to the cluster centroid in question.

Nearest Neighbor Search, Popularity, and Recommendation

Consider the introduction of a favorably rated song by a user with the intention of identifying similar songs. A trivial approach would be to compare this song's characteristic embedding to every other songs' characteristic embedding under a distance metric of choice before returning the top K songs to the user according to closeness in characteristic embedding space. One major flaw with this approach is the breadth of the search space which leads to increasing computational cost as more songs are introduced to a database, highlighting scalability issues. Rather than evaluating the distance between the favorable song's embedding against every other song, one technique for reducing the search space is only computing the distance between the song and cluster centroids. In this case, the cluster with the closest centroid to our incoming embedding would represent our reduced search space. Our embedding will then be evaluated against every song embedding in the closest cluster utilizing the same distance metric that was used to evaluate centroid distance.

At this stage, a recommendation system could return with top K most similar songs within the closest cluster to the candidate embedding. However, one remaining issue is that a similar song in terms of characteristics does not necessarily imply that a user will enjoy this song. To control for this effect, a collaborative filtering based method could be employed that measures song popularity for every characterized song within the database which operationalizes the idea that if a high volume of other users think favorably of a song in aggregate then the likelihood of enjoying and wanting to listen to this song increases for users. With the inclusion of a normalized popularity metric alongside normalized distance computations (closer songs are higher to 1 such that normalized distance can be interpreted as normalized similarity), the following recommendation score can be computed with the highest K scores being returned to the user:

$$Recommendation\ Score\ =\ Normalized\ Popularity\ *\ Normalized\ Similarity\ to\ Candidate$$

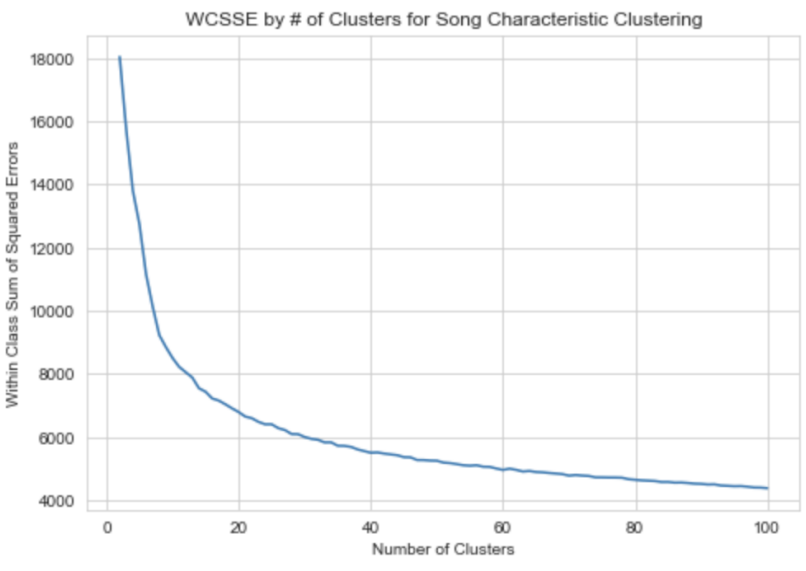
Recommendation Scheme in Practice

To illustrate this recommendation scheme, our team constructed a novel version that followed the above criteria. We began by characterizing a sample of 14661 songs across genres with complete lyric data and audio features. Each song was fed through our models to create a 39 dimensional characteristic embedding representation of the song (11 - topic, 11 - language, 7 - genre, 10 - audio). A min-max normalization was then executed strictly on the audio features to ensure they were represented on a similar scale as the other characteristics. To incorporate popularity into recommendation, our team also stored min-max normalized popularity scores provided by Spotify which rank scores from 0 (least popular) to 100 (most popular) according to listenership.

Following the characterization of songs into embedding representations, our team trained a KMeans clustering model on this dataset using Euclidean Distance as the distance metric of choice, measured by:

$$Euclidean\ Distance\ (x,z)\ =\sqrt{(x\ -\ z)^T(x\ -\ z)}\ =\ square\ root\ of\ sum\ of\ squared\ deviations\ in\ every\ dimension$$

To determine cluster quantity, we trained KMeans models at increasing numbers of clusters and evaluated the Within Class Sum of Squared Errors (WCSSE) which measures the squared distance between a point and the centroid of it's cluster to quantify the overall variance of the resulting clusters:



$$WCSSE\ =\ \sum_{i=1}^{k}\sum_{j=1}^{m}(x_{ij}\ -\ \bar{x}_i)^2$$

Common practice for cluster quantity selection using WCSSE as an evaluation metric indicates picking an arbitrary number of clusters past the "elbow" of the WCSSE plot. Visual inspection of our WCSSE plot yielded a K = 30 for our cluster quantity of choice. Following the training of of K = 30, KMeans clustering modeling, we developed the recommendation approach outlined above which procedurally executed the following given a candidate embedding:

- Identify closest cluster centroid using Euclidean Distance
- Compute Euclidean Distance between candidate embedding and every embedding in the closest cluster
- Max-Min Normalization of distance to return Normalized Similarity
- Compute Normalized Similarity * Normalized Popularity to return recommendation scores for each embedding within the closest cluster
- Sort embeddings by recommendation score and return the top K

Across an evaluation of 100 random recommendations, our recommendation model averaged 0.02 seconds to return top K recommendations agnostic of K.

Figure 9: WCSSE Plot for Evaluating Cluster Quantity on Characteristic Embeddings

Recommendation Examples

See below for examples of recommendations returned by our recommendation model for candidate songs featuring variable primary genres:

	Artist Name	Track Name	Popularity	Similarity	Recommendation Score
1	Ariana Grande	34+35	0.88	0.9299	0.8183
2	Doja Cat	Kiss Me More (feat. SZA)	0.98	0.8225	0.8060
3	Lil Nas X	MONTERO (Call Me By Your Name)	1.00	0.7782	0.7782
4	MEDUZA	Paradise (feat. Dermot Kennedy)	0.89	0.8661	0.7708
5	DJ Khaled	POPSTAR (feat. Drake)	0.84	0.9121	0.7662

Figure 10: Hip Hop Recommendation Test: Drake - Laugh Now Cry Later

	Artist Name	Track Name	Popularity	Similarity	Recommendation Score
1	AC/DC	You Shook Me All Night Long	0.80	0.8968	0.7174
2	Train	Hey, Soul Sister	0.83	0.8599	0.7137
3	Nelly Furtado	Promiscuous	0.82	0.8642	0.7086
4	Bon Jovi	You Give Love A Bad Name	0.80	0.8788	0.7030
5	Counting Crows	Mr. Jones	0.77	0.9101	0.7008

Figure 12: Pop Recommendation Test: Taylor Swift - Shake it Off

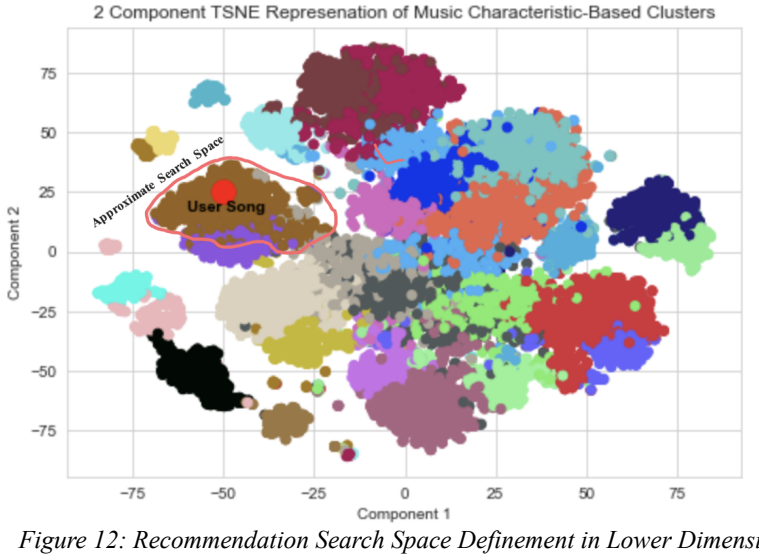


Figure 12: Recommendation Search Space Definement in Lower Dimensions

	Artist Name	Track Name	Popularity	Similarity	Recommendation Score
1	Train	Drops of Jupiter (Tell Me)	0.78	0.9640	0.7519
2	Red Hot Chili Peppers	Under the Bridge	0.81	0.9026	0.7311
3	Queen	Bohemian Rhapsody - Remastered 2011	0.82	0.8612	0.7062
4	The Police	Every Little Thing She Does Is Magic	0.74	0.9021	0.6676
5	Ed Sheeran	Perfect	0.86	0.7733	0.6650

Figure 13: Indie Recommendation Test: Arctic Monkeys - Do I Wanna Know?

CONCLUSION

We have examined the viability of neural language architectures across a variety of relevant tasks to music information retrieval and recommendation. Namely:

- ***Language Detection,*** the comparison of term density feature models vs embedding-based models for detecting primary song language given lyrics
- ***Topic Density Estimation***, evaluation of various embedding based models, embedding sizes, and transformer architectures for predicting topic density distributions and mimicking unsupervised topic models given lyrics
- ***Genre Classification***, evaluation of term density, audio, embedding-based, and multimodal approaches that combine language and sound features for determining musical genre
- ***Recommendation,*** a novel approach that operationalizes above model prediction capabilities alongside audio features in a recommendation scheme utilizing classical machine learning techniques (KMeans, Nearest Neighbors) and metric normalization to serve logical musical recommendations to users based off favorably ranked candidate songs

Our team's experimentation and model results indicate that neural language architectures can be used in the above tasks to provide relevant and useful information in the context of music information retrieval and music recommendation. Plans for immediate future work and additional areas of relevant exploration include:

- Inspection into other multimodal approaches for genre classification, mel spectrograms + audio + language as an example
- Exploration into recommendation quality scoring schemes to evaluate recommendation performance and validate whether we are detecting an optimal set of attributes as inputs into a characteristic embedding-based recommendation system
- Research into other open source datasets that are suitable for similar tasks to evaluate model generalization capabilities across datasets

CITATIONS

[1] Alexandros Tsaptsinos. Lyrics-based music genre classification using a hierarchical attention network. In ISMIR, 2017.

[2] Cory McKay, John Ashley Burgoyne, Jason Hockman, Jordan BL Smith, Gabriel Vigliensoni, and Ichiro Fujinaga. Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features.

[3] Rudolf Mayer and Andreas Rauber. Musical genre classification by ensembles of audio and lyrics features. In ISMIR, pages 675–680, 2011.

[4] Yeshwant Singh and Anupam Biswas. Robustness of musical features on deep learning models for music genre classification. Expert Systems with Application. Volume 199

[5] Wang Hongdan, Siti SalmiJamali, Chen Zhengping, Shan Qiaojuan, Ren Le. An intelligent music genre analysis using feature extraction and classification using deep learning techniques,. Computers and Electrical Engineering. Volume 100

[6] Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. Retrieving Similar Lyrics for Music Recommendation System. In Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017), pages 290–297. 2017

[7] Tzanetakis, George and Essl, Georg and Cook, Perry. Automatic Musical Genre Classification Of Audio Signals The International Society for Music Information Retrieval. 2001

[8] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[9] Neisse, Anderson. Song lyrics from 79 musical genres. https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres

[10] Lior, Ido. Spotify Multi-Genre Playlists Data. https://www.kaggle.com/datasets/siropo/spotify-multigenre-playlists-data

[11] Moura, Luan; Fontelles, Emanuel; Sampaio, Vinicius; França, Mardônio (2020), “Music Dataset: Lyrics and Metadata from 1950 to 2019”, Mendeley Data, V3, doi: 10.17632/3t9vbwxgr5.3

[12] Is Portuguese Similar To Spanish | Simple Answer. The Language Doctors. https://thelanguagedoctors.org/is-portuguese-similar-to-spanish/

[13] Keunwoo Choi, Gyorgy Fazekas, Kyunghyn Cho, Mark Sandler. A Tutorial on Deep Learning for Music Information Retrieval.

[14] Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen. Deep content-based music recommendation. Advances in Neural Information Processing Systems 26 (NIPS 2013)

[15] Schedl, Markus. Deep Learning in Music Recommendation Systems. Front. Appl. Math. Stat., 29 August 2019
Sec. Mathematics of Computation and Data Science. 2019.

[16] Jean-Julien Aucouturier and François Pachet. Representing Musical Genre: A State of the Art. Journal of New Music Research 2003, Vol. 32, No. 1, pp. 83–93.

[17] Derek A. Huang, Arianna A. Serafini, Eli J. Pugh. Music Genre Classification. https://github.com/derekahuang/Music-Classification

[18] Zichao Yang , Diyi Yang , Chris Dyer , Xiaodong He , Alex Smola , Eduard Hovy. Hierarchical Attention Networks for Document Classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies .

[19] David M. Blei, Jon D. McAuliffe. Supervised topic models. arXiv:1003.0783.