CS 6120: Music Genre Classification

Arinjay Jain
jain.arinj@northeastern.edu

Dhruv Doshi
doshi.dhru@northeastern.edu

Santosh Saranyan
saranyan.s@northeastern.edu

Abstract

This project focuses on the implementation and evaluation of various models, including Naïve Bayes as the baseline, Multiclass logistic regression, long short-term memory (LSTM), and LinearSVM, to classify songs into their respective genres based on the lyrics of each song. The dataset comprises over 138,000 English songs from various genres merged from 2 sources. It implements 2 word embedding techniques, Word2Vec and GLoVe, after preprocessing and cleaning the data. The performances of different models and embedding techniques are evaluated by calculating the accuracy, precision, recall, and F1 score. The results of this study can be used by music industry professionals to gain insights about the popularity and trends of different music genres and make data-driven decisions about music production and marketing. The project concludes by discussing the differences between the results obtained from each model and suggesting the most appropriate model for genre classification based on a song lyric task.

1 Introduction

One of the most exciting applications of NLP is the analysis of large volumes of text data to gain insights and knowledge. In recent years, the music industry has seen an increasing interest in using NLP techniques to classify songs into their respective genres based on the lyrics of each song (Li, 2020).

The aim of this project is to implement and evaluate various models like Naïve Bayes, Multiclass logistic regression, long short-term memory (LSTM), and LinearSVM. We have merged data from 2 sources comprising of lyrics from various genres such as rock, pop, country, hip-hop, etc. The merged dataset is then preprocessed and cleaned to remove irrelevant information to prepare it for modeling. We will implement 2 word embedding techniques, Word2Vec and GLoVe, and finally select the better-performing one. Next, we will train the models on the preprocessed data and evaluate its performance on a test dataset by calculating the accuracy, precision, recall, and F1 score. These models can be used by music industry professionals in order to get insights about the popularity and trends of different music genres and make data-driven decisions about music production and marketing.

2 Background/Related Work

Many related works in the past have proposed the classification of song lyrics based on genre as a technique to tag and automatically index music. They state that this type of indexing is useful for many music streaming services to arrange songs into genre-based playlists for recommendation purposes (A. Kumar et al. 2018).

Most of these works employ some form of a deep learning algorithm to perform the classification. Models based on some form of a deep learning algorithm such as Deep Neural Networks, Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) were among the popular ones for classification.

One work modifies a hierarchical technique called Hierarchical Attention Network (HAN) previously used for document classification (Z Yang, et al. 2016), for lyric-based classification, due to the hierarchical structure and composition of song lyrics (A. Tsaptsinos, 2017).

Some works even make the use of simpler models like an n-gram language model or a naïve bayes which perform quite decently for the task at hand (M. Fell, et al. 2014).

Another application was to simply use the lyrics to semantically analyze a song and discover various similarities between genre and artists. For this purpose, a technique called Probabilistic Latent Semantic Analysis (PLSA) was used (B. Logan et al. 2004).

Word embeddings were usually created using a GloVe (J. Pennington et al. 2014) or Word2Vec (T. Mikolov et al. 2013) model, but some works extracted additional features based on semantics such as emotions or based on style like word or letter repetitions (M. Fell, et al. 2014). Various word embeddings were also taken together such as combining Word2Vec and TF-IDF representations.

3 Data

The dataset used in this research consists of over 138,000 English songs by 2488 unique artists sourced from Kaggle, providing information on the song name, artist name, lyrics, and other relevant details. As the objective of this study is to predict the genre of the music, gold labels for each song are required, which were obtained from the publicly available Last.fm API. Although the API contained most of the genres for the songs, not all songs were covered. To mitigate this, the most common genre of the artist was selected as the missing genre for the songs, and songs from artists with no representation in the Last.fm API were excluded from the dataset.

Following this, the lyrics of each song were preprocessed for input into a Word2Vec model, enabling the generation of word embeddings that will subsequently be used in several language models. The preprocessing procedure consisted of converting the lyrics to lowercase and removing end-of-line tokens such as "\n".

We also noted that the lyrics contained metadata, specifying when different artists sang different parts of the song, where verses began, and where choruses occurred such as below:

"[Chorus]
See my head is so big they call me 'jack in the box',

[Hook]
Shine that light on me, put that light on me,
'cause I was sent to the world, by G-O-D"

To enhance the context of the actual words in the lyrics, we eliminated these meta tags and any other accompanying text. After obtaining pure lyrics in lowercase, we used the Natural Language Processing Kit (NLTK) library to tokenize sentences, remove period-endings from sentences, eliminate stop words, and finally tokenize sentences into words while lemmatizing them to obtain the base form of each word.

The next task was to cut down the number of genres. After getting data from the Last.fm API, we were left with a few hundred genres. To cut them down, we combined various sub-genres like classical rock, hard rock, etc into one genre called rock. We also normalized the genre names, for example, rhythm and blues, rnb, r&b, etc into just rnb. We then filtered out genres that had less than 1000 songs each. After all this, we were left with 16 genres.

Before we began training our models, we wished to explore our data more. So, we created a TF-IDF matrix from the songs and then visualized the most frequent words for each genre using a word cloud as shown in Fig. 1.

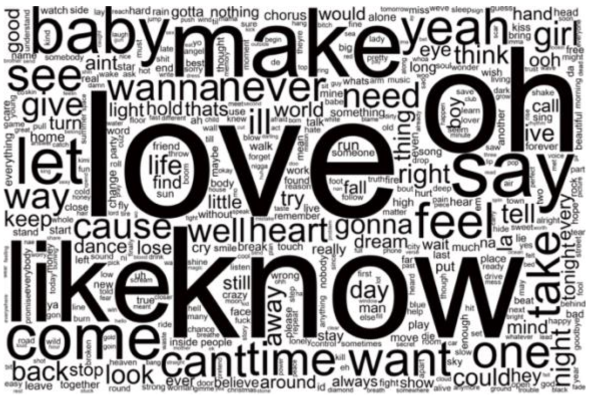

Figure 1 Word Cloud for 'pop'

4 Methods

4.1 Word Embeddings

For our word embeddings we used Word2Vec and GloVe embeddings. More specifically, for the Word2Vec approach made the use of the skip-gram technique. The skip gram technique uses a local context window to create word embeddings. It does well with word analogies, but the statistics of the corpus is very poorly utilized. This is due to the training of the skip gram model on the separated local context windows. This causes the skip gram model to not take advantage of the vast amounts of repetition in a corpus.

Unlike skip gram, the GloVe embeddings make good use of the corpus statistics in addition to doing well with word analogies. This is because the GloVe model uses global word to word co-occurrence counts to train the model for the word embeddings.

4.2 Multinomial Naïve Bayes

A Multinomial Naive Bayes (MNB) model will be used for text classification of songs into genre. The MNB model is a probabilistic algorithm that utilizes Bayes' theorem to predict the probability of a certain class given a set of features. The MNB model assumes that the presence of each feature is independent of the others and assigns a weight to each feature for each class. We will utilize pre-trained word embeddings to encode the lyrics of each song as a fixed-length feature vector. These vectors will then be used as input to the Multinomial Naive Bayes model and its results will be the baseline for our research and will be compared with the performance of other more complex models such as Multiclass Logistic Regression, Long Short-Term Memory (LSTM), and Multiclass Support Vector Machine (SVM).

4.3 Logistic Regression

Multiclass logistic regression is a popular linear classification model used for text classification. In our study, we will use word embeddings as features for training a multiclass logistic regression model. The input to the model will be a matrix of word embeddings, with each row representing a song and each column representing a feature dimension. The model will learn a set of weights for each feature dimension, which will be used to predict the probability of each song belonging to a particular genre. The model will be trained using a cross-entropy loss function, which will minimize the difference between the predicted probabilities and the actual labels of the songs. Once the model is trained, it can be used to predict the genre of new songs.

4.4 Support Vector Machines

SVM: Support Vector Machines (SVMs) are a popular choice for classification tasks like Genre classification based on lyrics. SVMs have several advantages which suit our case.

SVMs can handle high-dimensional data: Since the input we provide will be a song’s lyrics, it will be high-dimensional, and the model should be able to handle high-dimensional data well. SVMs are well-suited to handle high-dimensional data as they can identify the most relevant features and ignore the irrelevant ones. In the book "Introduction to Machine Learning with Python" by Andreas C. Müller and Sarah Guido, SVMs are described as a popular choice for classification tasks with high-dimensional data.

SVMs are effective in separating classes: When the classes are not linearly separable, SVMs work by finding the optimal hyperplane that separates different classes in the feature space.

They are described as an effective method for text classification tasks (Amit Nanavati, 2011).They state that SVMs are particularly useful for tasks where the number of features is much larger than the number of examples.

4.5 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a deep learning architecture based on Recurrent Neural Networks (S. Hochreiter et al. 1997). This architecture fixes a major problem present in traditional RNNs, which is the problem of vanishing gradients. This occurs during backpropagation of an RNN, when the gradients become exponentially small. This makes it difficult for RNNs to learn long term dependencies. LSTMs introduce memory cell mechanisms that allow them to store or forget information selectively, which helps in the prevention of the vanishing gradient problem. The state of a memory cell can be controlled by various gates which control how much new information is added (input), how much old information is removed (forget) and how much information is output from the current cell state to the next (output).

5 Experiments

5.1 Word Embeddings

Before training our models, we had to first create word embeddings for our song lyrics. We trained the word2vec embeddings using gensim's 'Word2Vec' function. We used the skip-gram method and had an embedding size of 200. We kept a window size of 3, due to the songs having short sentences, and trained it for 100 epochs.

For the GloVe embeddings, we used one of the pre-trained ones from J. Pennington et. al available on their website. We chose the 200-dimension embeddings consisting of 6 billion tokens called 'glove.6B.200d'. We loaded them into a word2vec format using genism's 'KeyedVectors'.

Both embeddings were then analyzed with the help of t-SNE projections to word visualize similarities. Sci-kit learn's 'TSNE' function was used to bring the dimension down to 2, and the top 25 most similar words for each word for the given list (words like crazy, rock, love, etc.) were extracted from the embeddings. In the plot, words that were similar were close to each other, and non-similar ones, far away.

Some common steps we took for training all our models were label encoding and feature vector calculation to make it easy to load into our models. Label encoding is the process of converting the categorical column 'Genre' into numerical data. This process is often used to categorical labels of a feature to numerical such that it can be processed by the machine learning algorithm. For example, for the target column 'Genre' if we have three categories: 'pop', 'rock' and 'hip-hop' then we will use label encoding to map these categories to numerical integer values: 0, 1, and 2 respectively. To calculate the feature vector of a song, we added the word embedding representation of every word in its lyrics and divide it by the length of the song. We used the train_test_split method from 'sklearn.model_selection' package to divide the data into an 80:20 split of training and testing data.

5.2 Multinomial Naïve Bayes

For the baseline Naïve Bayes classifier, we implemented a Gaussian Naïve Bayes classifier using the 'GaussianNB' model from the 'sklearn.naive_bayes' package. As we were using word embeddings with negative values to calculate the feature vectors for every song, we were unable to a regular Multinomial Naïve Bayes classifier as it cannot have negative values in its features. After having the training data (calculated using Word2Vec and GloVe embeddings) and labels, we fit 2 different models for each embedding type and calculated the accuracy for the test data as the baseline performance to beat by our other models.

5.3 Logistic Regression

For the second model, we implemented a multiclass Logistic Regression classifier. We did this using the 'LogisticRegression' model from 'sklearn.linear_model' package and created our own implementation using the library 'PyTorch'. In the PyTorch model our architecture consisted of a single linear layer which had an input layer size of 200 (word embedding size) with the output layer having 16 nodes. For both implementations of the Logistic Regression model, our experiments consisted of training the model with different regularizations (no regularization, L1 and L2) on both sets of embeddings and comparing their accuracy and F1 score with the testing data. The training was done using gradient descent with condition for end either convergence or a maximum of 1000 iterations whichever came first. In the PyTorch implementation the learning rate for gradient descent was varied for experimenting as well. For the PyTorch implementation, we check the accuracy, precision, recall and F1 score of classification by the model on every epoch and store the best accuracy and best F1 score. These stored values are then compared between the different models with different hyperparameters.

5.4 Support Vector Machines

We implemented the SVM model using Support Vector Classification class from the 'sklearn.svm' module of the 'scikit-learn' library. The 'SVC' class is used to create a Support Vector Machine (SVM) model for classification tasks. In this class, we can adjust the hyperparameter 'C' to customize the behavior of the model. The 'SVC' class implements SVM algorithm for binary and multi-class classification problems. The SVC class cannot handle Null values. Hence, we need to check and remove Null values from the dataset before moving forward. Now we fit the SVC model by passing in X_train and y_train and set the value of 'C' to default of 1.0. Next, we make predictions by passing in X_test and compare the predictions with y_test to calculate the accuracy. We also calculate the precision and recall for this model.

Tuning the SVM model: We can modify the value of the hyperparameter 'C' to customize the model. It is a regularization parameter that controls the cost of misclassification on the training set. We considered [0.1, 1, 10] possible values for 'C' and performed a grid search to find the best value using GridSearchCV function in the scikit-learn library.

5.5 Long Short-Term Memory

The last model to train was the LSTM model. This model was built and trained using 'PyTorch'. The words in the lyrics for each song were first replaced by their index in the word embeddings and were split into batches and loaded into the model using PyTorch's 'DataLoader' function. The batches were also padded using the 'pad_sequence' function to ensure each batch had vectors of equal length.

The model was built starting with an embedding layer whose weights are the word embeddings. Next, a dropout layer was added with a probability 'p' of 0.2. The output of the dropout layer was passed to LSTM layers using torch's 'nn.LSTM'. The number of hidden units was given as 64. The number of layers parameter was set to 2 indicating a stacked LSTM (the second LSTM gets the outputs of the first). Finally, the output of the LSTM was fed to a linear layer whose size corresponded to the number of genres (16). There was no softmax function used as the 'nn.CrossEntropyLoss' function was used to calculate loss during training. The optimizer used for the gradients was the Adam optimizer. The models were trained for 20 epochs each, and the best weights of the model was extracted by comparing the performance on a validation set at the end of each epoch.

5.6 Results

Results for Gaussian Naïve Bayes:

Embeddings	Accuracy	Precision	Recall	F1 Score
Word2Vec	0.2644	0.3116	0.2644	0.2442
GloVe	0.2369	0.2831	0.2396	0.2122

Results for Logistic Regression using SkLearn:

Embeddings	Regularization	Accuracy	Precision	Recall	F1 Score
Word2Vec	No	0.4301	0.4029	0.4301	0.3691
	L1	0.4307	0.4027	0.4307	0.3710
	L2	0.4301	0.4029	0.4301	0.3691
GloVe	No	0.4092	0.3794	0.4092	0.3434
	L1	0.4103	0.3803	0.4103	0.3454
	L2	0.4092	0.3794	0.4092	0.3434

Results for Logistic Regression with PyTorch:

Embeddings	Regularization	Accuracy	F1 Score
Word2Vec	No	0.2578	0.1257
	L1	0.2703	0.1609
	L2	0.2577	0.1483
GloVe	No	0.2772	0.1441
	L1	0.2578	0.1071
	L2	0.2778	0.1625

Results for SVM:

Embeddings	Accuracy	Precision	Recall	F1 Score
Word2Vec	0.4182	0.3803	0.4182	0.40
GloVe	0.2566	0.0065	0.2566	0.01

Results for LSTM:

Embeddings	Accuracy	Precision	Recall	F1 Score
Word2Vec	0.4941	0.4762	0.4941	0.4661
GloVe	0.4874	0.4645	0.4874	0.4566

6 Conclusion

We implemented multiple machine learning and deep learning classifiers to classify music into genres on the basis of its lyrics. For each model, we experimented with our own word2vec embeddings, trained on our corpus and with embeddings from a pretrained GloVe model. Our baseline performance was set by the Gaussian Naïve Bayes classifier.

The LSTM model with Word2Vec embeddings performed the best with an accuracy of 49.41% and an F1 score of 0.4661. The Word2Vec embeddings most likely had a better performance than the GloVe due to them being trained on the corpus while the embeddings for GloVe were pretrained.

LSTM models perform better than other models as they can capture the temporal dependencies between words or characters as well as capture the semantic meaning of words and phrases. In the case of music genre classification based on lyrics, this can be particularly useful as the sequence of words in a song can be important in determining its genre.

SVM and regression models, on the other hand, are better suited for tasks where the input features have a clear linear or nonlinear relationship with the output variable. In the case of music genre classification, there may not always be a clear linear relationship between the lyrics of a song and its genre, which could limit the performance of these models.

For future work we would want to try more sophisticated models such as transformer-based models (BERT and GPT) and ensemble models consisting of the combination of many different models.

References

Li, J. L. 2020. Automatic Song Genre Classification using Lyric Texts. *2020 3rd International Conference on Music and Audio Computing, (pp. pp. 43-47).*

A. Kumar. A. Rajpal. and D. Rathore. 2018. Genre Classification using Word Embeddings and Deep Learning. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, India, 2018, pp. 2142-2146, doi: 10.1109/ICACCI.2018.8554816.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. *arXiv preprint arXiv:1707.04678.* https://arxiv.org/pdf/1707.04678.pdf

Michael Fell. and Caroline Sporleder. 2014. Lyrics-based Analysis and Classification of Music. *25th International Conference on Computational Linguistics: Technical Papers, pages 620–631* https://aclanthology.org/C14-1059.pdf

Anna Boonyanit. and Andrea Dahl. 2021. Music Genre Classification using Song Lyrics. *Stanford University.*

B. Logan. A. Kositsky and P. Moreno. 2004. Semantic analysis of song lyrics. *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, Taipei, Taiwan, 2004, pp. 827-830 Vol.2, doi: 10.1109/ICME.2004.1394328.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. https://nlp.stanford.edu/projects/glove/

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Amit Nanavati, A. M. 2011. A Comparative Study of SVM and Decision Tree for Text Classification. International Journal of Computer Applications.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

A. Neisse. 2022. Song Lyrics from 79 Musical Genres. https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres (Dataset Used)

MUSIC GENRE CLASSIFICATION

GitHub repository link: https://github.com/arinjay97/Music_genre_classification

CODE AUTHORS
Arinjay Jain
Dhruv Doshi
Santosh Saranyan

FILES IN REPOSITORY
data: https://www.dropbox.com/s/90i3kyd6z2e3s52/data.zip?dl=0 folder containing all necessary datasets and text files to run the project. This file has been compressed as a zip file. Extract before use.

data_merging: This file contains python code to get information about a song's genre from last.fm's API and combine it with our dataset.

svm_model_implementation: This is a notebook file consisting of the implementation steps for preparing the data, training the SVM model, tuning it, and evaluating the model performance on test data using Word2Vec embeddings and GloVe embeddings each.

preprocessing_lyrics_dd_final: This is a notebook file consisting of the implementation steps for pre-processing the data like case folding, removal of stop words, lemmatization, etc. Finally, it saves the modified dataset as a csv file.

GNB and Logistic Regression: This notebook has implementation of sklearn Gaussian Naive Bayes and Logistic Regression classifiers as well as a PyTorch implementation of a Logistic Regression classifier. Experiments with different embeddings and hyper parameters are performed and their performance is calculated. To run thai file, install the libraries imported at the top of the notebook and run all cells in order.

Genre_Reduction: This is a notebook file that normalizes the names of the genres and combines various sub-genres into one. It filters out genres that have fewer songs and visualizes a word cloud for the most frequent words for each genre.

Word_Embeddings: This is a notebook file that trains and creates Word2Vec embeddings on the corpus. A word similarity plot is visualized for the Word2Vec and pre-trained GloVe embeddings with the help of t-SNE to reduce dimensions.

LSTM_Model: This is a notebook file that builds, trains and evaluates an LSTM model with PyTorch. Two models are trained, one for each word embedding.

SOURCE FOR THE DATASET
https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres