Text Mining Project Report

507 – Artificial Intelligence

Tomeh Lara

Introduction: Music lyrics can be classified into various genres based on the themes and subjects they explore, the instrumentation and production techniques used, and the artist's history and style. In this project, we aim to develop machine learning and deep learning models that can classify music lyrics by genre.

To achieve this, we have a dataset of music lyrics and their associated genres, and we use it to train and evaluate the models. We will explore different models and techniques, and compare their performance on the task of genre classification.

Our goal is to build a model that can accurately classify music lyrics by genre, and to understand the factors that influence the classification performance. This will not only help us better understand the characteristics of different music genres, but it could also have practical applications, such as helping music streaming platforms recommend songs to users based on their preferred genres.

Implementation details: the data is divided into data for training, evaluating, and testing, after building the different models, only two columns were kept in the data, which are Lyrics and Genre, and the rest of the columns were deleted due to the large size of the data. Four models were used, Multinomial NB, Random Forest, Support vector Classifier, and AdaBoost Classifier, in addition to two deep learning models, but the deep learning models was not as good as expected. For data processing, various methods were used, such as converting letters to lowercase letters, deleting numbers, contractions, and punctuation marks from the text, removing stop words, and also word embeddings algorithm either count vectorizer or tf-idf vectorizer, the stemming using Porter stemmer and lemmatization processes by WordNet lemmatizer. In each attempt, the values of these things were changed, and the obtained results were documented.

In the end, the model svc achieved the best results most of the time, followed by Multinomial Naïve Bayes model, as for random forest and AdaBoost models, their

results were almost always five to eight percent lower than the results of the previous models, while the deep learning models were causing overfitting because the accuracy results in the training set was much higher than the results in the validation and test set, so the models were not trained as much.

Conclusion: after conducting our experiments and analyzing the results, it is clear that our models did not perform well on the task of classifying music lyrics by genre. Despite trying several different models and techniques, we were unable to achieve satisfactory accuracy levels, and the model consistently struggled to distinguish between different genres. There could be several reasons for this poor performance. One possibility is that the dataset we used was not representative of the full range of music genres, and did not contain enough examples of each genre to allow the model to learn the characteristics of each genre. Another possibility is that the task of genre classification is inherently difficult, and requires more complex models or additional data to achieve good performance.

Overall, our results suggest that there is still much work to be done in the area of music lyrics genre classification, and more research is needed to develop effective models for this task. In the future, it may be worth exploring other approaches, such as using larger datasets or more advanced machine learning techniques, in order to achieve better results.

The tables below show the parameter values and models' results containing the score of the validation set and test set to ensure not overfitting.

	Machine Learning Models			
	Multinomial NB	Random Forest	SVC Model	AdaBoost Classifier
	Lowering text, remove contractions, remove stop words, porter stemmer, no lemmatize, validation size = 0.1, tf-idf vectorizer, Max features = 20000			
Validation score	0.49	0.41	0.49	0.41
Test score	0.49	0.41	0.48	0.40
	Keep contractions, keep stop words, porter stemmer, no lemmatizer, validation size = 0.2, tf-idf vectorizer, Max features = 30000			
Validation score	0.49	0.43	0.5	0.42
Test score	0.48	0.41	0.49	0.41
	Keep contractions, keep stop words, no stemmer, wordnet lemmatizer, validation size = 0.2, tf-idf vectorizer, Max features = 30000			
Validation score	0.49	0.43	0.5	0.4
Test score	0.48	0.42	0.49	0.41
	Remove contractions, keep stop words, porter stemmer, no lemmatize, validation size = 0.2, count vectorizer, Max features = 20000			
Validation score	0.47	0.43	0.47	0.41
Test score	0.49	0.42	0.47	0.41
	No preprocessing, validation size = 0.2, tf-idf vectorizer, Max features = 20000			
Validation score	0.49	0.43	0.5	0.4
Test score	0.49	0.42	0.49	0.4
	No preprocessing, validation size = 0.2, count vectorizer, Max features = 20000			
Validation score	0.49	0.43	0.47	0.41
Test score	0.49	0.41	0.46	0.41
	Lowering text, remove contractions, remove stop words, porter stemmer, no lemmatizer, validation size = 0.3, tf-idf vectorizer, Max features = 40000			
Validation score	0.49	0.41	0.48	0.4
Test score	0.48	0.39	0.48	0.39

	Deep Learning Models
Parameters values	**Lowering text, remove contractions, remove stop words, porter stemmer, no lemmatize, validation size=0.1, tf-idf vectorizer, Max features = 20000, Dropout rate = 0.5, optimizer = Adam, batch size=32**
Validation accuracy	**0.41**
Val f1 score	**0.4**
Test accuracy	**0.39**
Test f1 score	**0.39**
Parameters values	**keep contractions, keep stop words, porter stemmer, no lemmatize, validation size=0.2, tf-idf vectorizer, Max features = 30000, Dropout rate=0.3, optimizer = Adam, batch size=64**
Validation accuracy	**0.46**
Val f1 score	**0.46**
Test accuracy	**0.46**
Test f1 score	**0.46**
Parameters values	**Keep contractions, keep stop words, no stemmer, wordnet lemmatizer, validation size = 0.2, tf-idf vectorizer, Max features = 30000, Dropout rate = 0.3, optimizer = rmsprop, batch size=64**
Validation accuracy	**0.45**
Val f1 score	**0.45**
Test accuracy	**0.45**
Test f1 score	**0.45**