<u>**NLP Project**</u>

<u>**Micha Briskman – 208674713, Shlomo Gulayev - 318757382**</u>

Music-Genre-Classification-using-lyrics

Abstract

This project aims to build a system that can identify the genre of a song based on its lyrics. We create a set of songs with four labels - Rock, Hip-Hop, Country, Pop. Then we design four models to classify the songs into their genres - Bert model, Glove model, Word2Vec model and SVM model. All our models use recurrent neural networks (except SVM) to predict the genre of the song. We implemented pytorch and used transformers text embeddings in Bert model. The Bert model uses LSTM. For Glove model we implemented pytorch and used Glove embeddings. W2V uses W2V embeddings. They both use LSTM. For SVM model we implemented scikit learn algorithm and used our own word embeddings.

Introduction

In the field of Natural Language Processing, the classification of genres of a song solely based on the lyrics is considered a challenging task, because audio features of a song also provides valuable information to classify the song into its respective genre. Previously researchers have tried several approaches to this problem, but they have failed to identify a method which would perform significantly well in this case. SVM, KNN and Naive Bayes have been used previously in lyrical classification research. But classification into more than 10 genres have not been particularly successful, because then the clear boundary between the genres is often lost. So, we try to use a dataset of four genres. Hence, we try to approach this problem as a supervised learning problem applying several methods. We analysed the relative advantages and disadvantages of each of the methods and finally reported our observations. With the advent of deep learning, it has been observed that Neural Networks perform better than the previously used models.

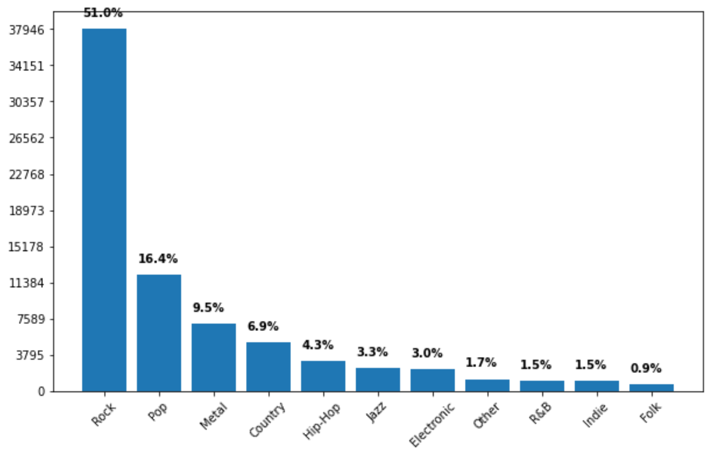
Dataset

The dataset for this problem was not abundant mostly due to copyright issues. However, after comparing datasets from several sources, we found out a data set which was most suited for our purpose. The dataset is basically a collection of 380000+ lyrics from songs scraped from https://github.com/hiteshyalamanchili/SongGenreClassification/blob/master/dataset/original_cleaned_lyrics.zip . However, the dataset was so large that we didn't have enough ram in the notebook to process it, so we used only 97,000+ lyrics.

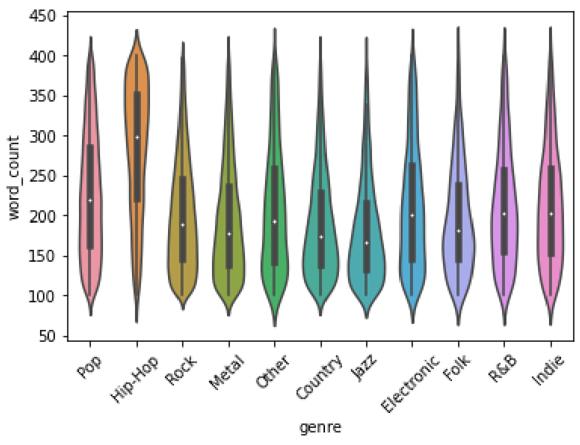
The structure of the data is index/song/year/artist/genre/lyrics. The data was not properly structured according to our needs there were additional genres we didn't need because they were underrepresented. So, we had to process our data before it could be fitted to any model for classification. Initially, we had to remove some irrelevant data from our dataset, making it more compact and easier to access. Like we removed artist and song year information thus creating just lyrics and genre mapping in our dataset. **Then we extracted songs of four genres - Rock, Hip-Hop, Pop, Country.** We extracted 2557 songs from each genre, making the dataset practical and easy to analyse. Then we removed some songs which had very few words in its lyrics (kept more than 100 and under 400 words). Then we tokenized the lyrics text using NLTK tool in Python for SVM model, and Spacy library for the other models. We also did some pre-processing of data for each of our models, which would be explained later.

Data Analysis

Before pre-processing we analysed the data and identified the features of data which is the first step of any machine learning problem. This analysis helped us understand the features of the data that would be most useful for the task in our hand. Then we calculated the average number of unique words in each genre. This would help us understand any correlation between the words used in lyrics and the genre type.



<u>Word count -</u> hip-hop has the most word count, and jazz has the lowest word count.



<u>Genre Analysis and Cleaning</u>

We removed some genres that were underrepresented: Folk, indie, R&B, other, metal, jazz, electronics.

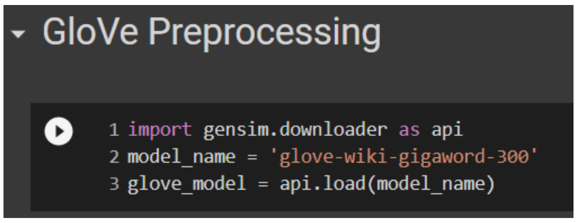
<u>We balanced the genres – 2557 each genre.</u>

Approaches

We have used four models: SVM model, GloVe model, Word2Vec model and Bert model. The W2V model that we used was the most effective of all, we used a LSTM Network, with W2V word embeddings.

Glove Model:

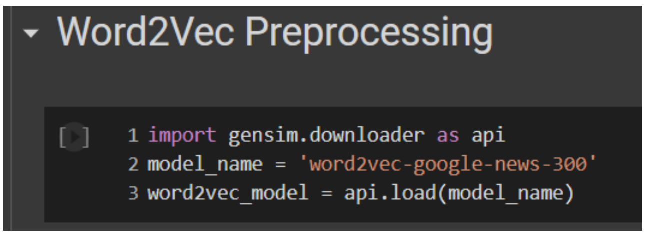
The GloVe model is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. The GloVe model we used was 'glove-wiki-gigaword-300'. We trained Glove vectors using python's genism library. We preprocessed the data with the spacy library. It uses a LSTM network we have built.


GloVe Preprocessing

```
import gensim.downloader as api
model_name = 'glove-wiki-gigaword-300'
glove_model = api.load(model_name)
```

Word2Vec model

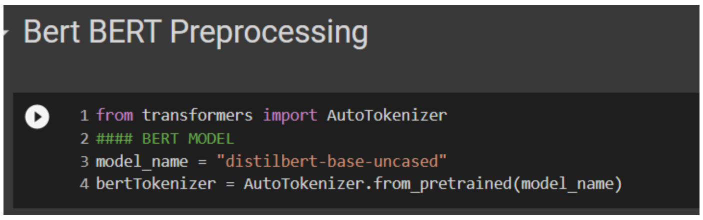
We used the word vectors (word2vec) to represent our lyrical text. These semantic vectors preserve most of the relevant information in a text while having relatively low dimensionality. Word2Vec is an algorithm that takes every word in your vocabulary that is, the text that needs to be classified is turned into a unique vector that can be added, subtracted, and manipulated in other ways just like a vector in space. The Word2Vec model we used was: 'word2vec-google-news-300'. We trained word vectors using python's genism library. We preprocessed the data with the spacy library. It uses a LSTM network we have built.


Word2Vec Preprocessing

```
import gensim.downloader as api
model_name = 'word2vec-google-news-300'
word2vec_model = api.load(model_name)
```

Bert Model

BERT is based on the transformer architecture. Specifically, BERT is composed of Transformer encoder layers. We preprocessed the data with the spacy library It uses a LSTM network that we have built.

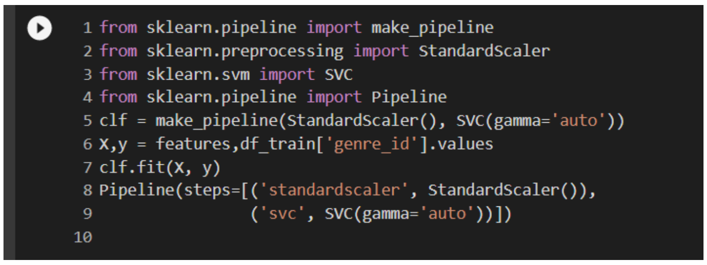


Bert BERT Preprocessing

```
from transformers import AutoTokenizer
#### BERT MODEL
model_name = "distilbert-base-uncased"
bertTokenizer = AutoTokenizer.from_pretrained(model_name)
```

SVM model

With our features and labels ready we fed them into a classier and trained it. We used 4:1 split of the dataset for training and testing. We preprocessed the data with NLTK library. We used python's scikit learn library to implement the algorithms:



```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
X,y = features,df_train['genre_id'].values
clf.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])

```

Note: everything is shown in the notebook

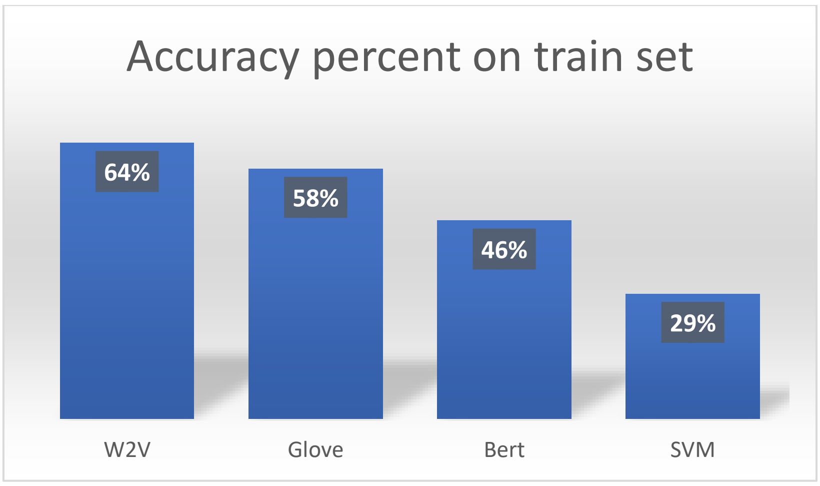
Results

Now we report the results of experiments on these models on a dataset of 10,228 (2557*4) songs equally distributed among all the genres.

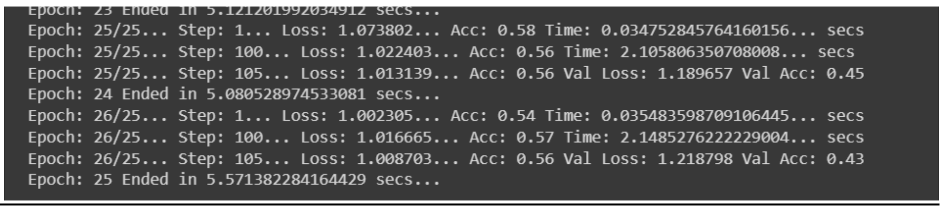
We ran the models (except SVM) with the following parameters:

- learning rate 10^{-5}
- dropout probability: 0.2
- The number of epochs: Providing enough time for the model to learn about 25 epochs.
- batch size: Tried batch sizes of 96.
- LSTM layers: 2
- hidden dim: 128

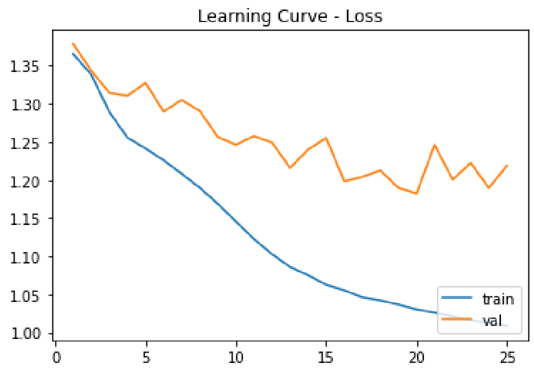
In the Glove model we could achieve an accuracy of 58%.
In the W2V model we could achieve an accuracy of 64%.
In the Bert model we could achieve an accuracy of only 46%.
In the SVM model we could achieve an accuracy of only 29%.

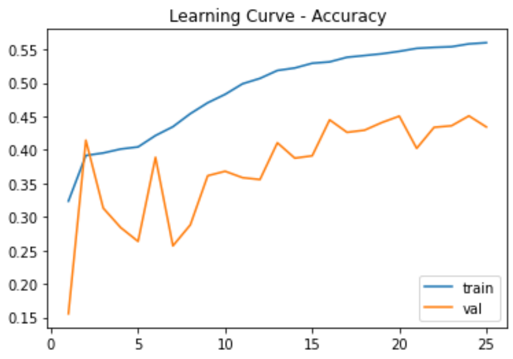


Loss function and accuracy Glove model:

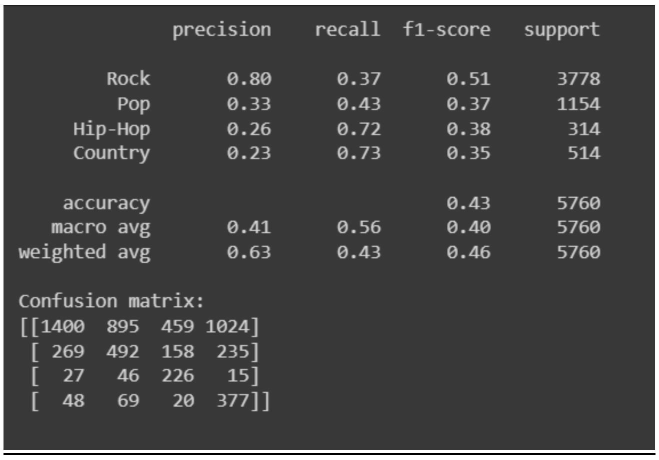


```
Epoch: 23 Ended in 5.121201992034912 secs...
Epoch: 25/25... Step: 1... Loss: 1.073802... Acc: 0.58 Time: 0.034752845764160156... secs
Epoch: 25/25... Step: 100... Loss: 1.022403... Acc: 0.56 Time: 2.105806350708008... secs
Epoch: 25/25... Step: 105... Loss: 1.013139... Acc: 0.56 Val Loss: 1.189657 Val Acc: 0.45
Epoch: 24 Ended in 5.080528974533081 secs...
Epoch: 26/25... Step: 1... Loss: 1.002305... Acc: 0.54 Time: 0.035483598709106445... secs
Epoch: 26/25... Step: 100... Loss: 1.016665... Acc: 0.57 Time: 2.1485276222229004... secs
Epoch: 26/25... Step: 105... Loss: 1.008703... Acc: 0.56 Val Loss: 1.218798 Val Acc: 0.43
Epoch: 25 Ended in 5.571382284164429 secs...
```



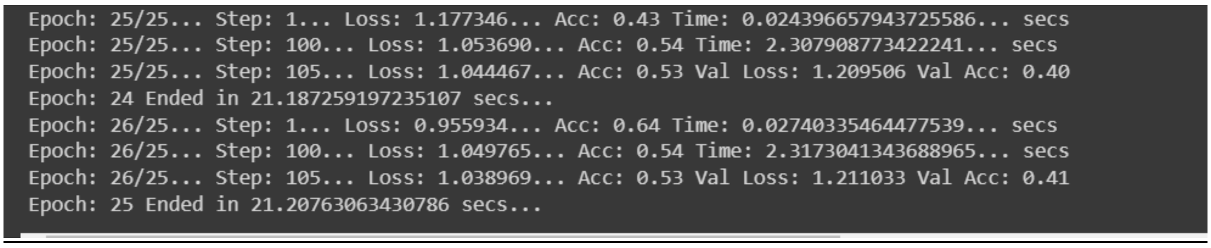


Evaluation Glove on test set:

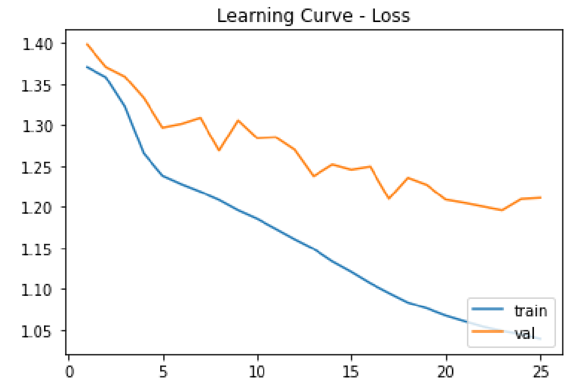


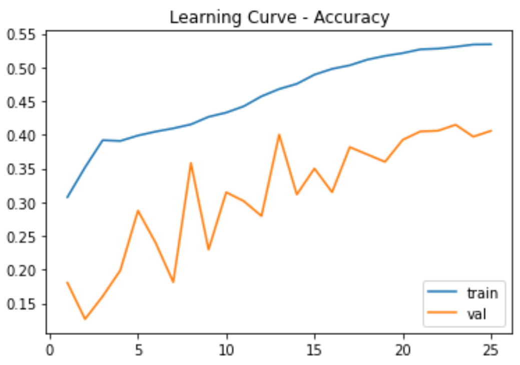
```
              precision    recall  f1-score   support

        Rock       0.80      0.37      0.51      3778
         Pop       0.33      0.43      0.37      1154
     Hip-Hop       0.26      0.72      0.38       314
     Country       0.23      0.73      0.35       514

    accuracy                           0.43      5760
   macro avg       0.41      0.56      0.40      5760
weighted avg       0.63      0.43      0.46      5760

Confusion matrix:
[[1400  895  459 1024]
 [ 269  492  158  235]
 [  27   46  226   15]
 [  48   69   20  377]]
```

Loss function and accuracy Word2Vec model:

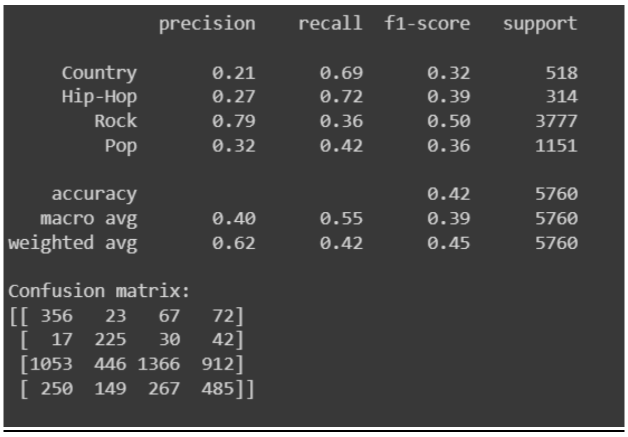


```
Epoch: 25/25... Step: 1... Loss: 1.177346... Acc: 0.43 Time: 0.024396657943725586... secs
Epoch: 25/25... Step: 100... Loss: 1.053690... Acc: 0.54 Time: 2.307908773422241... secs
Epoch: 25/25... Step: 105... Loss: 1.044467... Acc: 0.53 Val Loss: 1.209506 Val Acc: 0.40
Epoch: 24 Ended in 21.187259197235107 secs...
Epoch: 26/25... Step: 1... Loss: 0.955934... Acc: 0.64 Time: 0.027403354644775539... secs
Epoch: 26/25... Step: 100... Loss: 1.049765... Acc: 0.54 Time: 2.3173041343688965... secs
Epoch: 26/25... Step: 105... Loss: 1.038969... Acc: 0.53 Val Loss: 1.211033 Val Acc: 0.41
Epoch: 25 Ended in 21.20763063430786 secs...
```



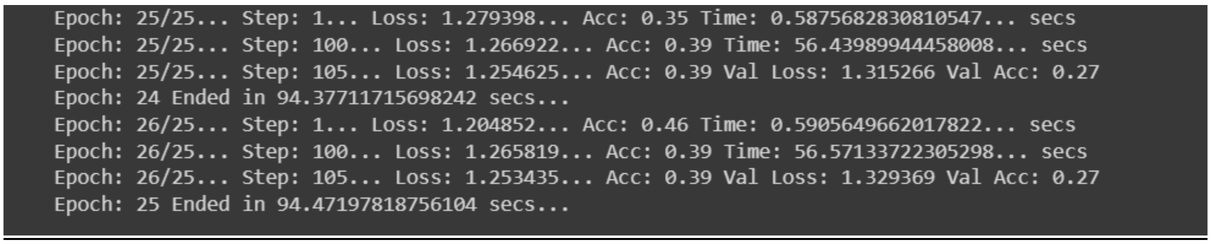


Evaluation Word2Vec on test set:

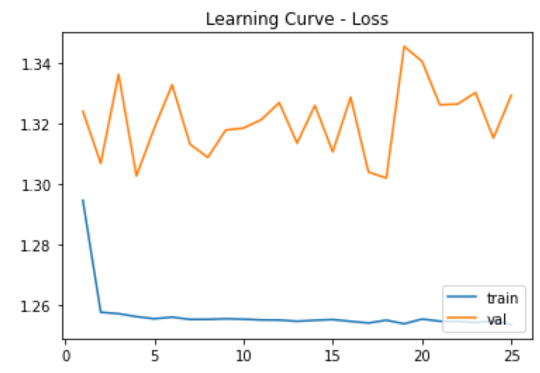


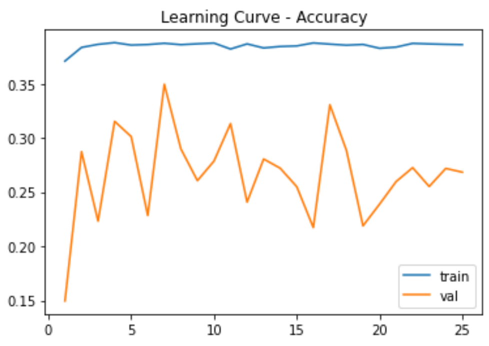
```
             precision    recall  f1-score   support

     Country       0.21      0.69      0.32       518
     Hip-Hop       0.27      0.72      0.39       314
        Rock       0.79      0.36      0.50      3777
         Pop       0.32      0.42      0.36      1151

    accuracy                           0.42      5760
   macro avg       0.40      0.55      0.39      5760
weighted avg       0.62      0.42      0.45      5760

Confusion matrix:
[[ 356   23   67   72]
 [  17  225   30   42]
 [1053  446 1366  912]
 [ 250  149  267  485]]
```

Loss function Bert:

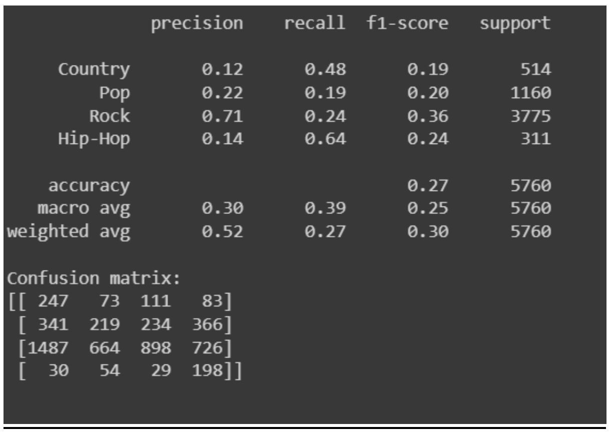


```
    Epoch: 25/25... Step: 1... Loss: 1.279398... Acc: 0.35 Time: 0.5875682830810547... secs
    Epoch: 25/25... Step: 100... Loss: 1.266922... Acc: 0.39 Time: 56.43989944458008... secs
    Epoch: 25/25... Step: 105... Loss: 1.254625... Acc: 0.39 Val Loss: 1.315266 Val Acc: 0.27
    Epoch: 24 Ended in 94.37711715698242 secs...
    Epoch: 26/25... Step: 1... Loss: 1.204852... Acc: 0.46 Time: 0.5905649662017822... secs
    Epoch: 26/25... Step: 100... Loss: 1.265819... Acc: 0.39 Time: 56.57133722305298... secs
    Epoch: 26/25... Step: 105... Loss: 1.253435... Acc: 0.39 Val Loss: 1.329369 Val Acc: 0.27
    Epoch: 25 Ended in 94.47197818756104 secs...
```



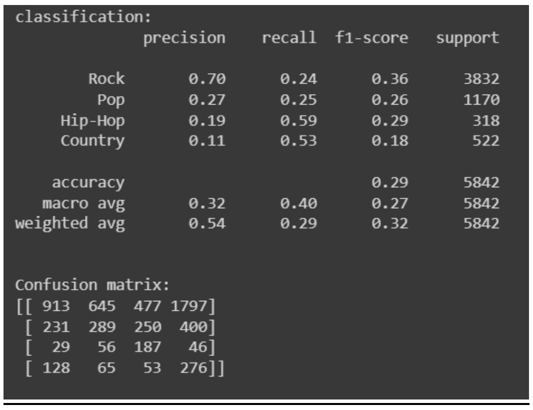


Evaluation Bert on test set:



```
               precision    recall  f1-score   support

     Country       0.12      0.48      0.19       514
         Pop       0.22      0.19      0.20      1160
        Rock       0.71      0.24      0.36      3775
     Hip-Hop       0.14      0.64      0.24       311

    accuracy                           0.27      5760
   macro avg       0.30      0.39      0.25      5760
weighted avg       0.52      0.27      0.30      5760

Confusion matrix:
[[ 247   73  111   83]
 [ 341  219  234  366]
 [1487  664  898  726]
 [  30   54   29  198]]
```

Evaluation SVM:



```
classification:
               precision    recall  f1-score   support

        Rock       0.70      0.24      0.36      3832
         Pop       0.27      0.25      0.26      1170
     Hip-Hop       0.19      0.59      0.29       318
     Country       0.11      0.53      0.18       522

    accuracy                           0.29      5842
   macro avg       0.32      0.40      0.27      5842
weighted avg       0.54      0.29      0.32      5842


Confusion matrix:
[[ 913  645  477 1797]
 [ 231  289  250  400]
 [  29   56  187   46]
 [ 128   65   53  276]]
```

Conclusions and Future Work

From the models that we developed and the experiments that we conducted we can say that the Word2Vec Model performed well compared to the other models. In that respect Glove model perform well as well. Apart from Rock (as seen from the confusion matrixes) other genres might be mislabelled at times. However, limited by time/RAM/GPU, we could produce some significant results in the field of music genre classification based on lyrics. There is a lot that can be done like better pre-processing of data. Adding more data for each of genre classes. We might train a model with lyrics as well as audio features and it is expected that we can get better results. Also, we might train a more complex model which would remember order of words, and we can experiment on our training data. Classification by lyrics will always be inherently awed by vague genre boundaries with many genres borrowing lyrics and styles from one another. For example, one merely need consider cover songs which utilise the same lyrics but produce songs in vastly different genres, songs which have no lyrical content. To produce a state of the art classifier is evident that this classifier must take into account more than just the lyrical content of the song. Audio data typically performs the strongest and further research could look into employing these models to the audio and symbolic data and combining with the lyrics to build a stronger classifier.