Music genre classification based on song lyrics
Final report for NLP Course, Winter 2022-2023

Bartłomiej Eljasiak
Warsaw University of Technology
bartlomiej.eljasiak.stud@pw.edu.pl

Aleksandra Nawrocka
Warsaw University of Technology
aleksandra.nawrocka.stud@pw.edu.pl

Dominika Umiastowska
Warsaw University of Technology
dominika.umiastowska.stud@pw.edu.pl

supervisor: Anna Wróblewska
Warsaw University of Technology
anna.wroblewska1@pw.edu.pl

Abstract

Although a well-known task, music genre classification (MGC) remains challenging in the domain of Music Information Retrieval. We tackle the problem of MGC based solely on lyrics. Our main contribution is trying to improve the results we obtained during the first project by testing two new approaches: the addition of sentiment analysis and using two classifiers to address the imbalance of the dataset. Additionally, we prepared a new dataset and test some of the methods on it.

1 Introduction

A music genre is a conventional label on the musical piece which characterizes it as having certain features, conventions, or characteristics. It is quite a complicated problem to say precisely how genres are distinguished. The genre often dictates the style and rhythm of the audio of the song. It seems much harder to define the music genre by lyrics alone, even from a human perspective. Therefore, it is quite an interesting topic to try making such a distinction based on song text. Similar research has already been conducted, but this topic is yet to be fully explored.

The song's lyrics are often related to its melody and rhythm. It is also common for different genres to raise different topics. It was already shown that a combination of audio and text features gets better results than using only audio features [7]. Furthermore, lyrics may be more accessible and easier to process than audio. Therefore, lyrics classification seems to be an interesting field of study both for its own and for its potential connection with audio features.

We have previously explored different methods for lyrics-based genre classification. Our study included testing different methods of obtaining text embeddings, such as GloVe, word2vec, BERT, and varying classification models, such as Naive Bayes, Linear Support Vector Machine, XGBoost, and Convolutional Neural Network. In this research, we focus on improving obtained results. We add sentiment analysis for the classifier to see the lyrics from a different perspective. We build a model containing two separate models, where the first one decides whether the song belongs to the *Rock* genre or not and the second one classifies samples into remaining genres.

We also create a small dataset to explore using available APIs. Then we test some of the models on it.

The research paper is divided into multiple sections. In section 2 we describe related works in the domain of MGC. Section 3 presents used datasets and the preprocessing done. Furthermore, used methods and models are characterized. In section 4 we show the hyperparameters of the models used. In section 5 we demonstrate conducted experiments and obtained results. The whole project is concluded in section 6.

2 Related works

Music genre classification (MGC) is at this point a well-known research problem and a subdomain of Music Information Retrieval (MIR). Culture and therefore music avoids strict barriers and definitions, nevertheless, each piece of music is usually categorized into one or more genres. MGC enables us to study this categorization, explore similarities and differences between various genres or even construct a taxonomy.

In the past, due to heavy computational limitations, the main focus of MGC was put on finding the best features for classification purposes. In [9] such features were e.g. *AverageSyllablesPerWord* or *SentenceLengthAverage*. Naturally, word embedding played an important role in extracting information from lyrics and the use of simple methods like *bag-of-words* can be found in various papers [3, 6]. With time, an increasing amount of focus was put strictly on embeddings themselves, developing novel and improved representations.

Currently, all state-of-the-art approaches for MGC utilizing lyrics rely heavily on word embeddings. In a recent publication [4] an attempt was made to train the embedding model strictly on lyrics. Unfortunately, the significance of the work is hard to assess due to the lack of usage of this model.

It is also rather common to approach MGC in a multi-modal manner. Usage of the audio itself has to be second if not the most popular source of information with many published articles [1, 13, 11, 9, 8]. Other less trivial data sources are symbolic [14], culture [9], text reviews [11], and cover art [11]. One could say that at this stage researchers experiment with enriching the pieces of music with any meaningful data possible.

The datasets that we work on are:

- *Song lyrics from 79 musical genres* dataset from Kaggle website [10],
- *MetroLyrics* dataset processed and put in a GitHub repository [2],
- our own dataset created using Spotify API [12] and Genius API [5].

3 Approach and research methodology

3.1 Datasets

We have found two datasets which we decided to work on:

- *Song lyrics from 79 musical genres* dataset from Kaggle website [10],
- *MetroLyrics* dataset processed and put in a GitHub repository [2].

In the description of the first dataset, we can find the information that the dataset consists of 379 893 song lyrics from 4239 artists. Around 50% of the song lyrics are in English and we test our models on them. Information about the artists is kept in a separate file and contains a list of music genres each artist is connected with. As we predict only one music genre for each song we preprocess this dataset by reducing these lists to individual genres (we take the first one from the list) and assigning them to song lyrics of appropriate artists. Furthermore, we preprocess song lyrics as they contain punctuation and span across multiple lines.

By contrast, the second dataset required minimal work on our site. It was initially published on Kaggle website and consisted of 362 237 song lyrics from 18231 artists. The majority of song lyrics (probably around 60%) were in English. Unfortunately, this dataset was removed from Kaggle website and we were not able to find it in its original form anywhere else. We have found a preprocessed version of it in a GitHub repository of a students' project performed by University of California students in 2018. This version's song lyrics have punctuation removed and contain only one genre for each entry.

3.2 Datasets preparation

First of all, we conducted some basic preparations of datasets. For *MetroLyrics* we decided to remove genre *Other* and merge genres *Country* and *Folk* since our research showed they are very similar, and additionally the second was significantly smaller in samples. As for *Song lyrics from 79 musical genres*, we filtered songs to only English ones and omitted the genre *Pop/Rock* since it is ambiguous.

After conducting a few simple tests on the whole dataset we decided to limit ourselves to a much smaller part of the observations. There were two main reasons for that: limited resources and time - we would not manage to conduct all desired tests on the whole dataset, and second - the dataset was very strongly unbalanced and therefore accuracy was often significantly bigger than balanced accuracy. To solve both problems we decided to limit ourselves to only five genres with the biggest number of samples: *Rock*, *Pop*, *Metal*, *Hip-hop* and *Country*.

In the end, we conducted tests on two datasets:

- Balanced dataset with lyrics from 5 genres, $116,120$ observations in total, created from both *MetroLyrics* and *Song lyrics from 79 musical genres* datasets.
- Unbalanced dataset with lyrics from 5 genres, $262,122$ observations in total, created from both *MetroLyrics* and *Song lyrics from 79 musical genres* datasets.

3.3 Text preprocessing

Since in our project, we use embeddings that may be or even should be used on the raw text we decided to test two approaches: one with strong and the second with weak preprocessing of text.

Weak preprocessing consists of deleting numbers in the text and some words characteristic of lyrics notation (e.g. ”VERSE”, ”CHORUS”, ”2x”). We decided to also expand contractions since they were proven to be troublesome in further preprocessing such as tokenization or removing stopwords. We deleted all special characters since interpunction was used inconsistently in most songs (e.g. lyrics usually didn’t have a division for sentences). We also lowered the whole text since every verse began with a capital letter which usually had nothing to do with starting a sentence. Thanks to that we could also use uncased versions of embeddings.

Strong preprocessing consists of, besides the above, tokenization, lemmatization, and deleting stop words. For these steps, we used *nltk* package.

3.4 Division into two models

Since one of the datasets is very unbalanced (the amount of Rock genre observations is two times more numerous than all other genres together) we decided to try an approach that will classify the largest genre class separately from the rest. Therefore we prepared a 2-step CNN classifier that consists of two sub-classifiers: a BinaryCNN classifier which only recognizes if lyrics are of the one selected genre or not, and a multiclass CNN classifier (the same is used as a normal classifier for all genres) which classifies all other genres.

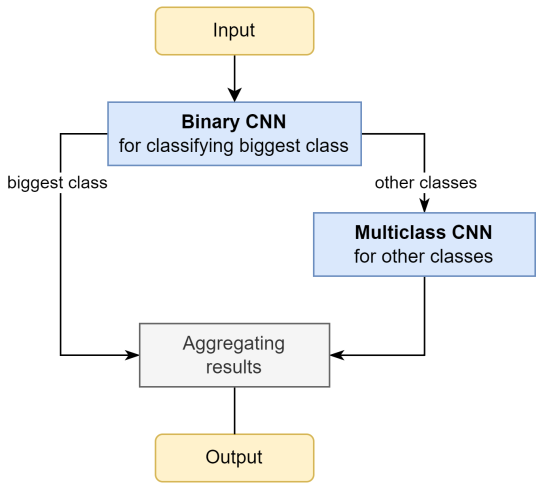

Figure 1: 2-step CNN architecture

In this way, we hope to better recognize and separate the Rock genre and influence significantly balanced accuracy.

3.5 Creation of the dataset

We decided to create our own dataset using Spotify API [12] and Genius API [5]. We wanted to use Spotify to get tracks from chosen genres and Genius to get lyrics of returned songs. At first, we utilized the *Get Recommendations* endpoint. Unfortunately, the recommended songs repeated a lot between different API calls. That meant that after dropping duplicates we were left with a few times less observations than desired. To deal with this issue, we decided to use the *Search for Item* endpoint. This endpoint allows specifying genre but limits the possible output to the first 1000 tracks. Another problem that we faced was that Genius did not always have the lyrics to the songs provided by Spotify. And even if it did sometimes instead of proper lyrics text of books or different lists of artists and the titles of their songs were returned. Because of that the created dataset needed also manual data cleaning. At last, in that way, we created the dataset containing 5 genres and 4,092 observations. Table 1 presents the number of observations in each genre.

Genre	**Number of observations**
country	896
metal	887
pop	815
rock	767
hip-hop	727

Table 1: The number of observations in each genre in created dataset

4 Hyperparameters of models

In the tables 2, 3, 4 we provide the hyperparameters of models. For our dataset and CNN and Dense classifiers the number of epochs was equal to 60 but for other datasets was equal to 5.

Embedding	**Library**	**Model name**	**Dimension**
GloVe	John Snow Labs Spark NLP	glove_100d	100
Smaller BERT	John Snow Labs Spark NLP	small_bert_L2_128	128

Table 2: Hyperparameters of embeddings

Classifier	**Library**	**Class**	**Optimizer**
Naive Bayes	sklearn	GaussianNB	-
Linear SVM	sklearn	SGDClassifier	-
XGBoost	xgboost	xgboost	-
CNN	tensorflow.keras	Sequential	Adam
2-step CNN	tensorflow.keras	Sequential	Adam
DenseNet	tensorflow.keras	Sequential	Adam

Table 3: Hyperparameters of classifiers

Max. number of words in lyrics
400

Table 4: Other hyperparameters

5 Experiments and results

5.1 New models and embeddings

In this project, several embedding models were tested, but it was soon rather clear that the best results will be obtained by the most complex embedding models. Additionally, we wanted to decrease the input size of classifiers to reduce the training time and complexity of several models. Described goals were achieved by introducing new embedding models with aggregated word embeddings. Said models are `DistilBERT` and `SentenceTransformerMPNET` with word embeddings transformed to singular sentence embedding, by aggregating them. This seemingly minor change greatly increased accuracy scores across all classifiers and set a new standard for future experiments.

We also used two new classifiers: DenseNet which consists of a few fully connected dense layers, and the 2-step CNN closer described before.

Results of our experiments on a balanced dataset are visible in 5 and 6 tables. Δ Acc. signify the difference between results acquired in this project for this dataset with the new embedding and the best results obtained in the previous project.

Classifier	Accuracy	F1-score	Δ Acc.
NB	54.40%	53.16%	+ 11.16
SVM	62.06%	61.07%	+ 11.45
XGBoost	63.93%	63.78%	**+ 18.46**
CNN	**65.47%**	**65.53%**	+ 8.99
DenseNet	64.96%	64.99%	-
2StepCNN	62.21%	57.97%	-

Table 5: Results for the DistilBERT on balanced dataset

Classifier	Accuracy	F1-score	Δ Acc.
NB	56.23%	55.97%	+ 12.99
SVM	62.83%	61.66%	+ 12.22
XGBoost	62.45%	62.31%	**+ 16.98**
CNN	65.06%	64.30%	+ 8.58
DenseNet	**66.42%**	**66.08%**	-
2StepCNN	63.34%	60.44%	-

Table 6: Results for the SentenceTransformerMPNET on balanced dataset

As we can see, all results are significantly better than in the previous project which shows how important embedding is in text classification. Also, the new classifier DenseNet gives very good results, even given its quite simple structure. The comparison of the results is shown in Figure 2.

5.2 Fine-tunning embedding models

The natural next step was to improve the information contained in the embedding model by combining it with the classification layer and fine-tuning the combined architecture. Since a DistilBERT was pre-trained on a broad corpus and a wide range of texts, we hoped that by partially training it on lyrics it will be able to improve the quality of the created embeddings. This task was realized on `small_balanced` version of the available dataset. Unfortunately, repeated experiments did not bring the expected results and the model performance was not significantly improved. A comparison of both fine-tuned and pretrained models is presented in Table 7. Lack of improvement may indicate problems in the training process but repeated experiments and tests on different models indicate that this was not the case.

Embedding	Classifier	Accuracy	F1 score
pretrained	Dense	0.6455	0.6431
fine-tuned	Dense	**0.65**	**0.6682**

Table 7: Results after fine-tuning

5.3 Addition of sentiment analysis

One of the main things that we wanted to test, was the impact of additional information added to the embeddings. At the same time since one of the assumptions of the project was working with only the lyrics, features that we could add to the embedding of the lyrics are all lyrics based. Our idea was to incorporate sentiment obtained by a separate model. Unfortunately, since we focused on using as hardware demanding embedding model there were little to no resources left for this sentiment model. After minor research we settled for `BERT-tiny-emotion-intent` model. It offers sentiment prediction in form of a six-dimensional vector measuring the sentiment as happy, sad, neutral, angry, excited, or frustrated. This is to some extent a redundancy since we expect the sentiment to be present in embeddings, at least to some degree, but this information may not be easy for the model to extract especially since our classification models are relatively simple.

We tested the influence of sentiment in two experiments. The first one was oriented on the changes in the performance of classifiers. As it can be seen in 8 results for feature vector enriched with embeddings did not really change, when compared to 5. The only exception here is the XGBoost model which improved significantly, but this may be partially explained by the instability of the XGBoost which was observed.

Classifier	Accuracy	Bal. Acc.	F1 score
Naive Bayes	0.5272	0.5272	0.5095
Linear SVM	0.6054	0.6054	0.5998
XGBoost	0.6300	0.6300	0.6279
CNN	0.6433	0.6433	**0.6433**
Dense	**0.6449**	**0.6449**	0.6341

Table 8: Classification results for better embeddings

The second experiment concerning sentiment included fine-tuning the combination of the embedding model, sentiment model and dense classifier. Same as with the previous first fine-tuning the results did not really change 9, and actually the model without sentiment achieves the best score, but only by a small margin of less than 1%.

sentiment	Accuracy	F1 score
yes	0.6412	0.6638
no	**0.65**	**0.6682**

Table 9: Results after fine-tuning with or without sentiment

5.4 Division into two models

There were already shown some results for the 2-step CNN classifier on the balanced dataset in the previous section in tables 5 and 6. In general, achieved by 2-step CNN accuracies were slightly worse than those of normal CNN and DenseNet.

But this classifier was created with the thought of an unbalanced dataset and therefore results on this dataset are the most important to us. Therefore, we tested 2-step CNN and two other classifiers which gave the best results in previous tests on the unbalanced dataset (which is also much bigger than balanced) to see if separate recognition of the biggest class gives better results. Results are in Table 10 and the comparison is shown in Figure 3.

Even though neither accuracy nor F1-score was better for 2-Step CNN, it proved to give higher balanced accuracy results. Therefore it may be better than other classifiers depending on the task at hand.

CLassifier	Accuracy	Bal. acc.	F1 score
CNN	64.74%	50.71%	60.88%
NN	**65.23%**	53.08%	**62.14%**
2-step CNN	52.94%	**59.88%**	**54.11%**

Table 10: Results for 2-step CNN, DenseNet, and CNN for DistilBERT on unbalanced dataset

5.5 Creation of the dataset

We conducted multiple tests for our dataset as it was a lot smaller than the other ones. The results are presented in Tables 11, 12 and Figures 4, 5, 6, 7. The best embedding turned out to be DistilBERT. But when it comes to classifiers the results were quite similar so it makes more sense to use the most basic classifier to limit the computations.

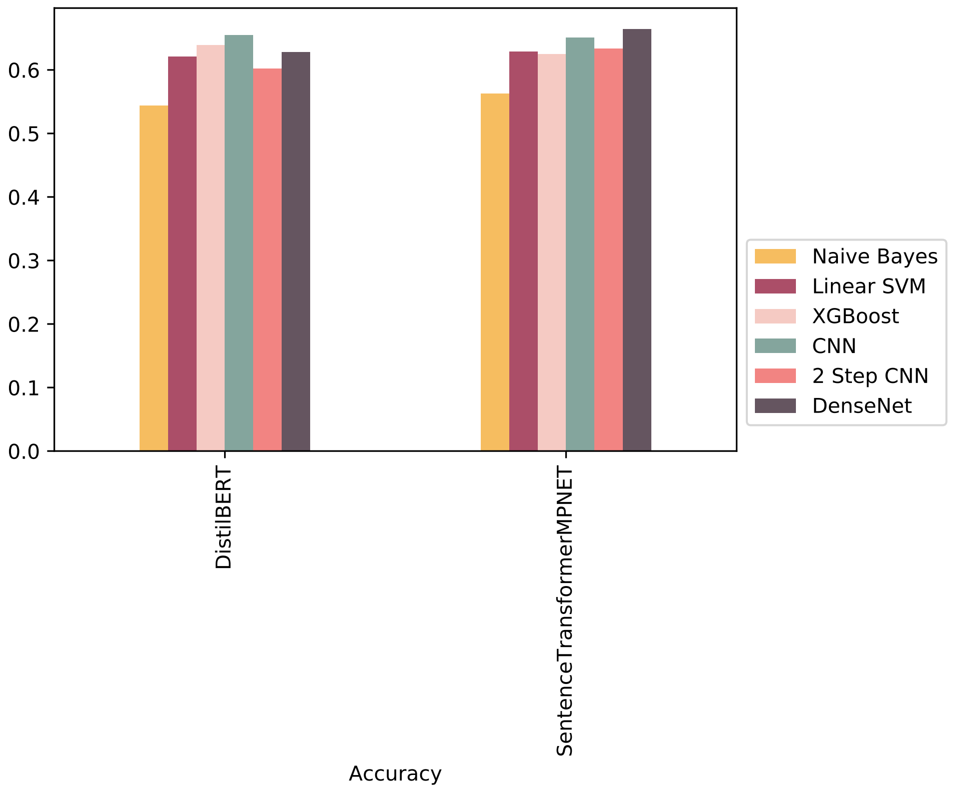

Figure 2: Comparison of DistilBERT and SentenceTransformerMPNET on balanced dataset

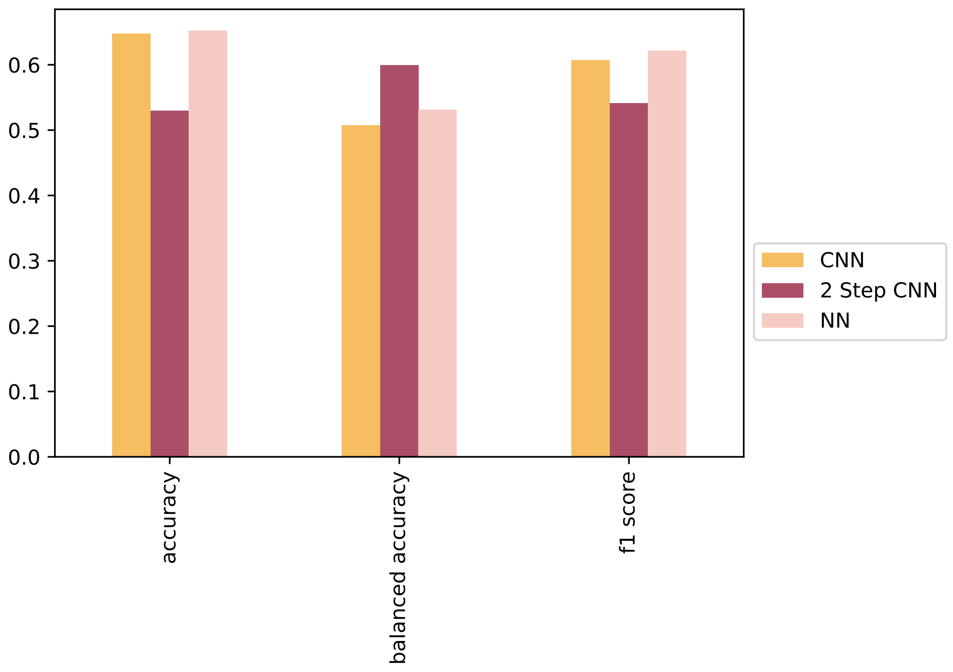

Figure 3: Comparison of different metrics for different classifiers on unbalanced dataset

Embedding	**Classifier**	**Accuracy**	**F1 score**
Smaller BERT	Naive Bayes	0.379479	0.328668
Smaller BERT	Linear SVM	0.399837	0.353035
Smaller BERT	XGBoost	0.375407	0.374183
Smaller BERT	CNN	0.415309	0.401595
Glove	Naive Bayes	0.333876	0.259667
Glove	Linear SVM	0.320033	0.295821
Glove	XGBoost	0.343648	0.336383
Glove	CNN	0.374593	0.382742
DistilBERT	**Naive Bayes**	**0.547231**	**0.550679**
DistilBERT	Linear SVM	0.477199	0.373867
DistilBERT	XGBoost	0.518730	0.519839
DistilBERT	CNN	0.535016	0.530048
DistilBERT	DenseNet	0.516287	0.516995
SentenceTransformerMPNET	CNN	0.495928	0.499407
SentenceTransformerMPNET	DenseNet	0.517915	0.517484

Table 11: Results for the created dataset and unnormalized lyrics

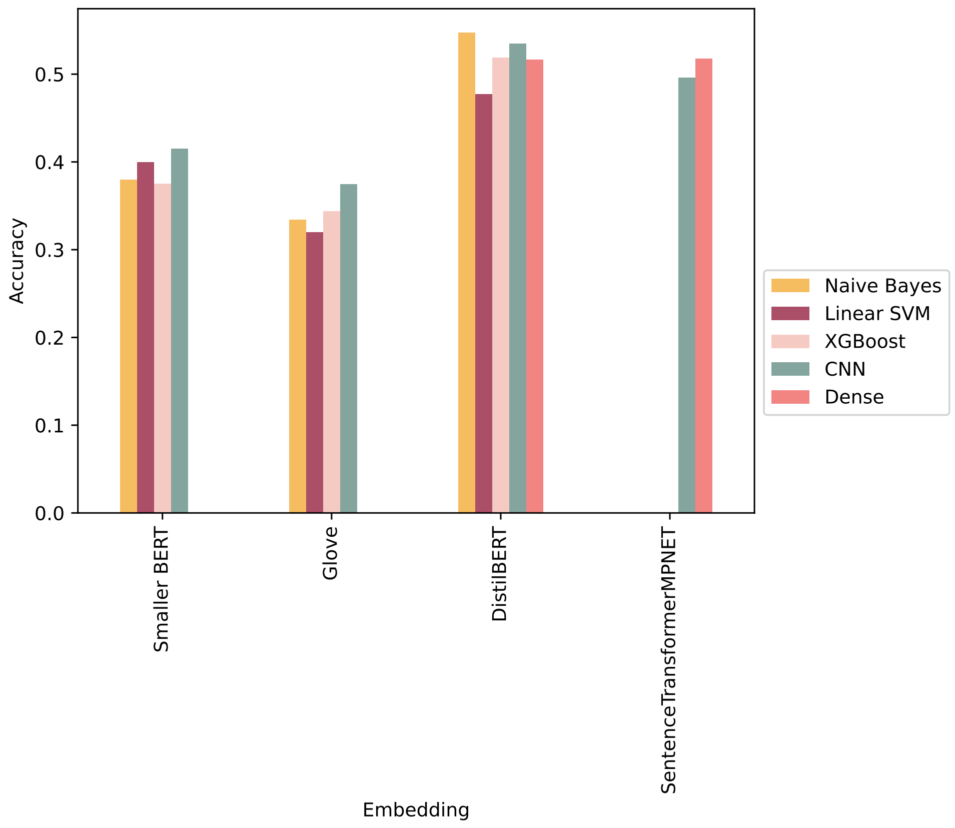

Figure 4: Accuracy for the created dataset and unnormalized lyrics

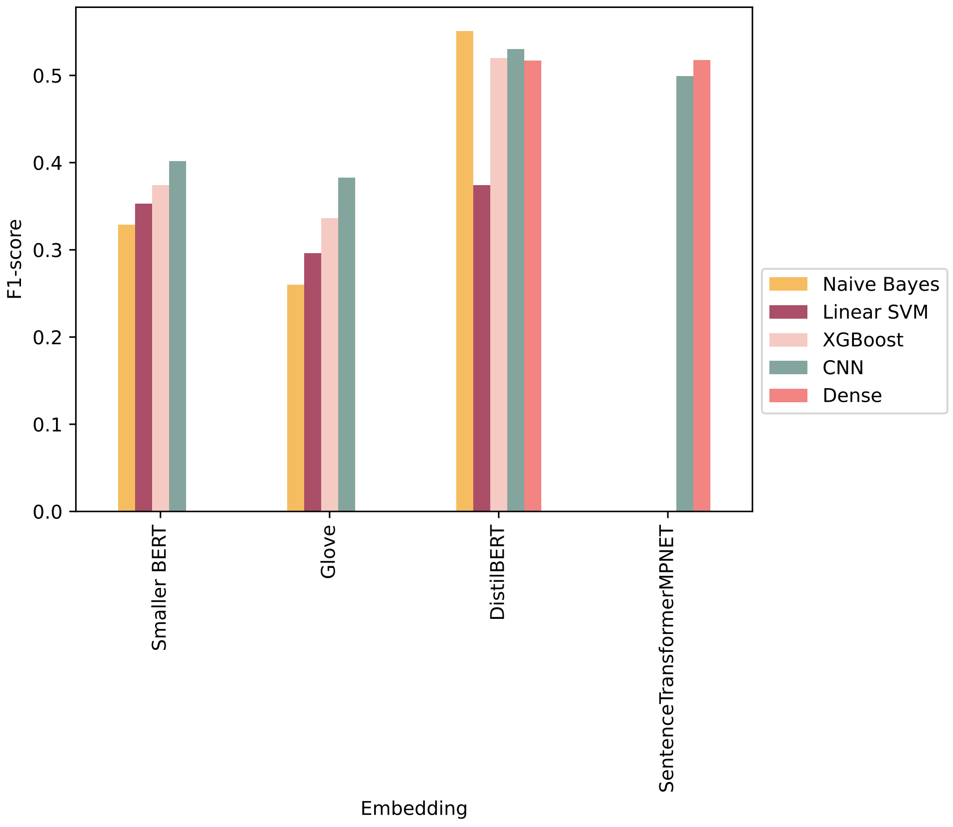

Figure 5: F1-score for the created dataset and unnormalized lyrics

Embedding	**Classifier**	**Accuracy**	**F1 score**
Smaller BERT	Naive Bayes	0.310261	0.247440
Smaller BERT	Linear SVM	0.368078	0.352763
Smaller BERT	XGBoost	0.400651	0.395027
Smaller BERT	CNN	0.439739	0.439537
Glove	Naive Bayes	0.328176	0.236727
Glove	Linear SVM	0.383550	0.376734
Glove	XGBoost	0.372964	0.369424
Glove	CNN	0.389251	0.395433
DistilBERT	Naive Bayes	0.515472	0.495294
DistilBERT	Linear SVM	**0.541531**	0.492267
DistilBERT	XGBoost	0.503257	0.500318
DistilBERT	CNN	0.494300	0.494112
DistilBERT	DenseNet	0.518730	**0.517378**
SentenceTransformerMPNET	CNN	0.514658	0.509839
SentenceTransformerMPNET	DenseNet	0.500000	0.501151

Table 12: Results for the created dataset and normalized lyrics

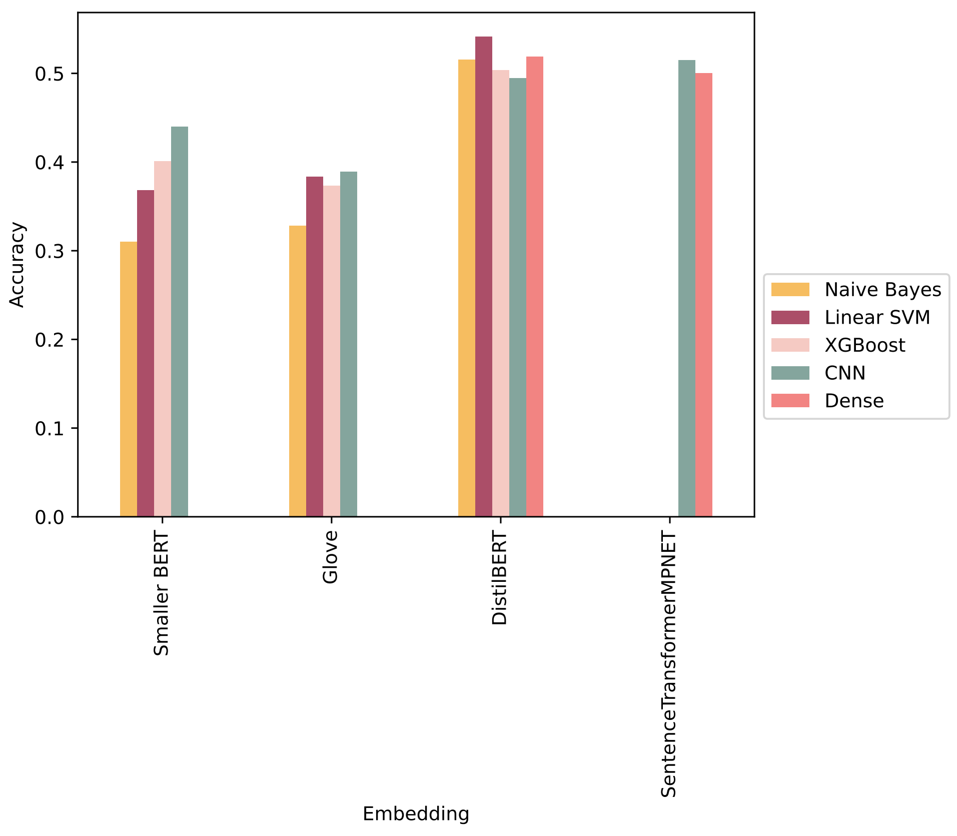

Figure 6: Accuracy for the created dataset and normalized lyrics

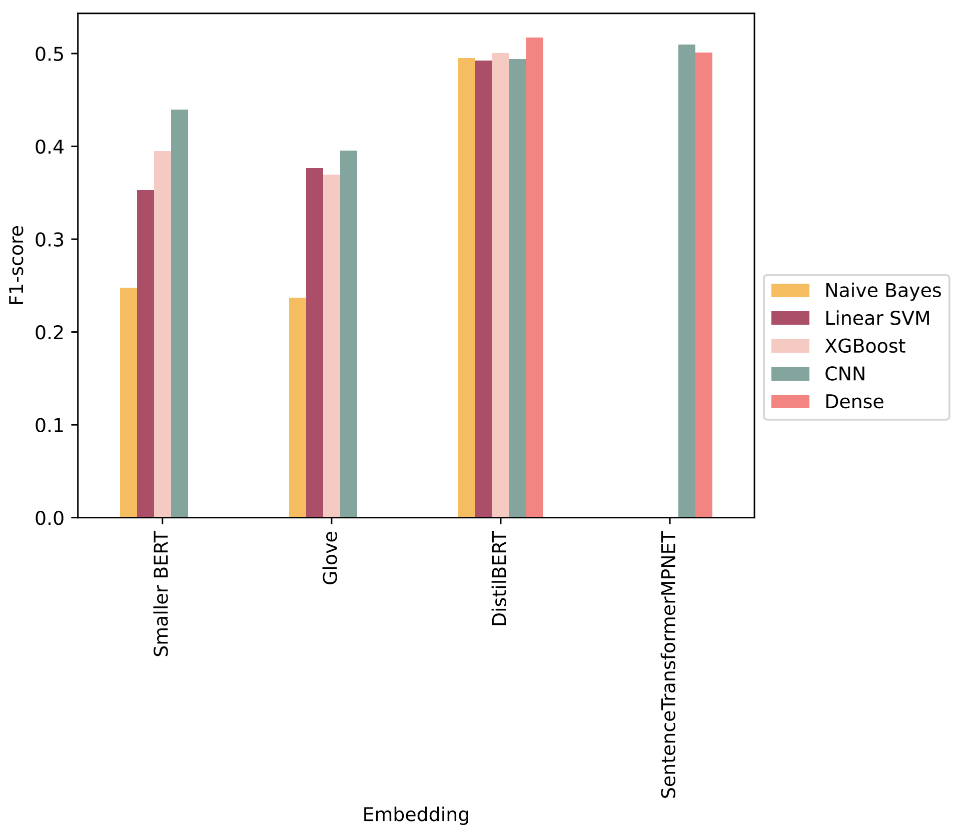

Figure 7: F1-score for the created dataset and normalized lyrics

6 Conclusion

Music genre classification performance seems to be limited strictly by the lyrics embedding quality. The most significant improvements were observed when switching to embedding models with substantially different complexity. This suggests that further improvements in this field may be achieved just by scaling up the embedding models. On the other hand, the typical procedure of fine-tuning pre-trained general language models, done on presented music genre data, did not bring significant improvements when compared to using just pre-trained models.

Fusing lyrics data, in the form of embeddings, with the sentiment of the lyrics did not bring significant improvements when tested in different settings.

The fusion of two models in the form of a 2-step classifier gives better results when it comes to balanced accuracy but worse overall. The problem may be that errors from the two models accumulate when they are used together. However, this approach can still be useful depending on the task at hand.

In the case of small datasets, simpler classifiers may give better results and are faster than training e.g. CNNs for multiple epochs.

7 Team contribution

Self-assessed authors' contribution is in table 13.

Author	Contributions	Workload
Bartłomiej Eljasiak	Changes to the repository, new embedding models, new embedding models fine-tuning, dense model addition, sentiment addition, sentiment influence analysis, part of the report and presentation	40% (20h)
Aleksandra Nawrocka	Dataset creation, performing tests on the created dataset, part of the report and presentation	30% (15h)
Dominika Umiastowska	Creating 2-step model, performing tests with this and other classifiers, part of report and presentation.	30% (15h)

Table 13: Authors' contribution

References

[1] Safaa Allamy and Alessandro Lameiras Koerich. "1D CNN Architectures for Music Genre Classification". In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2021, pp. 01–07. DOI: 10.1109/SSCI50451.2021.9659979.

[2] Connor Brennan et al. *SongGenreClassification*. 2018. URL: https://github.com/hiteshyalamanchili/SongGenreClassification/tree/master/dataset (visited on 10/31/2022).

[3] Önder Çoban and Işıl Karabey. "Music genre classification with word and document vectors". In: *2017 25th Signal Processing and Communications Applications Conference (SIU)*. 2017, pp. 1–4. DOI: 10.1109/SIU.2017.7960145.

[4] Seungheon Doh et al. "Musical Word Embedding: Bridging the Gap between Listening Contexts and Music". In: *CoRR* abs/2008.01190 (2020). arXiv: 2008.01190. URL: https://arxiv.org/abs/2008.01190.

[5] Genius. *Genius API*. URL: https://docs.genius.com/ (visited on 10/31/2022).

[6] Dawen Liang, Haijie Gu, and Brendan O'Connor. "Music genre classification with the million song dataset". In: *Machine Learning Department, CMU* (2011).

[7] Rudolf Mayer and Andreas Rauber. "Music genre classification by ensembles of audio and lyrics features". In: (2011).

[8] Rudolf Mayer and Andreas Rauber. "Music Genre Classification by Ensembles of Audio and Lyrics Features." In: Jan. 2011, pp. 675–680.

[9] Cory McKay et al. "Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features." In: *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010* (Jan. 2010), pp. 213–218.

[10] Anderson Neisse. *Song lyrics from 79 musical genres*. 2022. URL: https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres/versions/3 (visited on 10/31/2022).

[11] Sergio Oramas et al. "Multimodal deep learning for music genre classification". In: *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.* (2018).

[12] Spotify. *Spotify Web API*. URL: https://developer.spotify.com/documentation/web-api/ (visited on 10/31/2022).

[13] Yang Yu et al. "Deep attention based music genre classification". In: *Neurocomputing* 372 (2020), pp. 84–91. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.09.054. URL: https://www.sciencedirect.com/science/article/pii/S0925231219313220.

[14] Mingliang Zeng et al. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training". In: *CoRR* abs/2106.05630 (2021). arXiv: 2106.05630. URL: https://arxiv.org/abs/2106.05630.