Music genre classification based on song lyrics - comparison between different word embedding techniques and classifiers
Project Proposal for NLP Course, Winter 2022

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

Music genre classification (MGC), although a well-known task, still remains challenging in the domain of Music Information Retrieval. We tackle the problem of MGC based solely on lyrics and try to solve it using a solution composed of a state-of-the-art word embedding method and a separate classification model. Our main contribution is the comparison between different word embedding methods and classification techniques, which in the domain of MGC is currently lacking. The novelty comes with an additional approach in the form of testing the impact of enriching the lyrics with the title of the song.

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 [14]. 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.

In this research, we explore different methods for lyrics-based genre classification. Our study includes 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. This kind of analysis is currently lacking in the field of MGC.

We also test if adding additional information in the form of the title of the song may improve the accuracy of the model. This is interesting to see as in many songs the title is often repeated in the song lyrics. That means that the title may be actually redundant.

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. Futhermore, used methods and models are characterized. In section 4 we demonstrate exploratory data analysis, conducted experiments and obtained results, which are discussed in section 5. The whole project is concluded in section 6 and the future work that can be done in this subject is described in section 7.

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 limita-

tions, the main focus of MGC was put on finding the best features for classification purposes. In [16] 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 [8, 13]. 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 [10] 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 [2, 26, 20, 16, 15]. Other less trivial data sources are symbolic [27], culture [16], text reviews [20], and cover art [20]. One could say that at this stage researchers experiment with enriching the pieces of music with any meaningful data possible.

Reading through the papers approaching MGC in different ways, it is striking that in some cases crucial elements of the proposed solution, are presented without proper justification. In the case of [25] a 100-dimensional GloVe model was used. It was stated that it is better than another technique called word2vec, but no proof or reference was provided. No other methods were used therefore it is impossible to say what was the value gained from using the GloVe and not e.g. bag-of-words. In another work [1] authors used word embeddings obtained from BERT and DistilBERT, with build-on classifiers, then compared their accuracy to BILSTM [24], which as input received text embedded in an unspecified manner. Numerous simplifications, lack of details and often incomparable results should raise concerns among researchers. How can one declare improvement over some method or even guarantee the value of the proposed work, when provided context for the work is insufficient? Those concerns motivated us to create such context as a result of this project. We clearly declare how we preprocess data and which exactly embedding methods we use. Then we test which classification method works best on created lyrics representations.

The datasets that we work on are:

- *Song lyrics from 79 musical genres* dataset from Kaggle website [18],
- *MetroLyrics* dataset processed and put in a GitHub repository [6].

We describe in more detail, why we have chosen these datasets and how we preprocess data they contain in section 3.1.

3 Approach and research methodology

3.1 Datasets

While trying to find possible datasets for our project it was important to us for the song lyrics to be in their raw form. That means that they should not be transformed into e.g. bag-of-words model. The reason for this decision was that if we would like to use prepared models in a real-world scenario new song lyrics could be used with minimal preparation. What is more, as the language evolves all the time, with this approach there is still a possibility of further training of the models on song lyrics with the presence of not previously known words. And last but not least, we also wanted to minimize the risk of worse performance of prepared models which could be caused by simplifying the assumptions.

The effect of making this decision is the fact that we could not choose e.g. the musiXmatch dataset (the official lyrics collection of the Million Song Dataset [4]) as the dataset for conducting the experiments. This very well-known dataset consists of an enormous number of song lyrics which are unfortunately kept in a bag-of-words model form. But as we do not use this dataset, we also do not focus on its description.

We have found two datasets that meet our expectations:

- *Song lyrics from 79 musical genres* dataset from Kaggle website [18],
- *MetroLyrics* dataset processed and put in a GitHub repository [6].

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. The main reason for balancing the dataset was a further decrease in training time and therefore a possibility for testing more methods.
- Unbalanced dataset with both lyrics and title from 5 genres, $68,221$ observations in total, created only from *Song lyrics from 79 musical genres* dataset since only this dataset had title available.

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 Embeddings

One of the key problems in the domain of natural language processing has to be the question of how to use words in a model which only understands numbers. This question sparked numerous attempts of representing language in a mathematical way. One can always assign each unique word a different number and in this way encode any language into the computer, but this is insufficient when it comes to using this encoded representation. It was rather clear, that in order for such transformation to be in any way useful, the original meaning of the word should be embedded into this numeric representation itself. This word embedding ought to be treated as a vector in a given, high-dimensional space. For a given model dimensionality is fixed, therefore each word is represented by a vector of a set length, typically a hundred or so numeric values.

There is still a task of creating a model capable of such transformations. It has a couple of possible

approaches, mainly prediction-based and count-based. The second one, although simpler, will not be described here, since all methods described further make use of the first approach. Prediction-based word embedding models share a common trait, which unsurprisingly is that the embedding for a word was learned by performing the task of predicting given word [3]. This definition does not set any requirements for the prediction itself, and, in fact, different approaches have been used successfully, such as the continuous Bag-of-Words Model (CBOW) and continuous Skip-gram Model [17].

There is a single more distinction for the different techniques used, which is significant for this work. The meaning of some words is not dependent on their structure or origin, but on the context in which they are used. In the case of homonyms, it is impossible to state the singular meaning of the word without context, e.g. bank as a financial institution vs. side of a river or play as in theatre vs. as in sport. More traditional embeddings do not incorporate the context of a word in determining its embedding. These models are called static or non-contextualized. The ones that do generate differentiable vectors depending on the context are subcategorized as contextualized word embeddings. Despite the described advantage of the latter method, prior has proven to be successful in multiple cases, such as GloVe [21] or fastText [5].

Other techniques which prove promising are ELMo [22], a deep bidirectional language model, which is pre-trained on a large text corpus and, extracting contextualized word embedding form, pre-trained Google's Bidirectional Encoder Representations from Transformers (BERT) [9].

Little has been said about using word embedding in the context of music lyrics. [10] describes the process of training the word embedding model strictly on music lyrics, but lacks proper evaluation methods to be comparable to other works. This means that in order to reach state-of-the-art we were bound to testing various methods of word embedding. We decided to test GloVe, word2vec and BERT. In order to do that we use the nlu library from John Snow Labs [19].

3.5 Classification models

We test a few varying classification models for this specific task. We consider Naive Bayes classifier, Linear Support Vector Machine, XGBoost, and Convolutional Neural Network.

Naive Bayes is a classifier based on Bayes' theorem known for good performance on real-world tasks despite being a simple model. It is fast and good at dealing with unbalanced data. It has also widespread applications on text classification tasks [23].

Linear Support Vector Machine tries to find a hyperplane that best separates samples of different classes. It is often used for text classification tasks and historically achieved great results [11].

XGBoost [7] is a decision-tree based algorithm that uses a gradient boosting method. It shows great performance on large-scale tasks and is a very flexible and versatile tool.

Convolutional Neural Networks are one of the primarily used types of neural networks used commonly in both image and text classification [12]. Their main feature is using layers with convolution filters that are applied to feature vectors.

3.6 Adding title to song lyrics

When it comes to the additional approach that we tried, i.e. adding title to the song lyrics, it can be described in the following way. Normally we would input the song lyrics into the embedding model and then provide the output to the classifier. Instead, we independently produce the embeddings of the lyrics and the title and combine these two vectors together by putting the title vector first and the lyrics vector right beside. Then, the resulting vector is inserted to the classifier.

4 Experiments and results

4.1 First attempts

At first, we conducted some tests on *MetroLyrics* dataset only, using GloVe embedding. We did not process the dataset or lyrics much back then, therefore we had 11 genres and almost raw lyrics.

Classifier	Accuracy	Bal. acc.	F1-score
Naive Bayes	15.43%	16.48%	9.66%
Linear SVM	46.18%	20.55%	42.78%
XGBoost	30.39%	13.20%	31.17%
CNN	50.91%	25.84%	48.72%

Table 1: Results of the first attempts

As we can see accuracy may not be the best, but seems still decent for 11 genres. Unfortunately, balanced accuracy is much worse. It may be because it was a very highly unbalanced dataset

where the most common *Rock* label had $100,053$ observations and the rarest *Folk* only $1,689$. To eliminate this possibility we performed many previously mentioned operations on datasets in the hope it will improve both accuracy and balanced accuracy.

4.2 Balanced lyrics dataset

Firstly, we performed tests on the balanced lyrics dataset using different classifiers and Smaller BERT embedding. We also used text in two forms: after weak preprocessing and strong preprocessing. We achieved the following results:

Classifier	Accuracy	F1-score
Naive Bayes	43.24%	39.18%
Linear SVM	43.60%	40.03%
XGBoost	42.38%	42.30%
CNN	51.31%	51.05%

Table 2: Results for weak preprocessing

Classifier	Accuracy	F1-score
Naive Bayes	40.33%	33.69%
Linear SVM	50.61%	46.12%
XGBoost	45.47%	45.68%
CNN	53.54%	53.02%

Table 3: Results for strong preprocessing

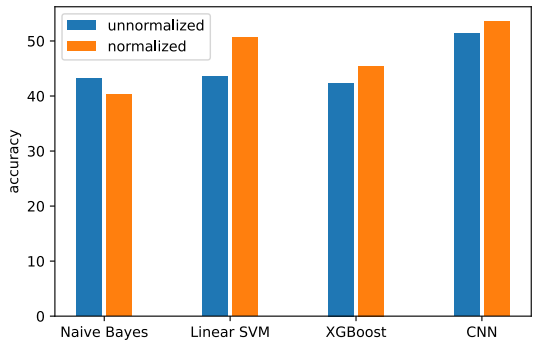

Figure 1: Classifiers and preprocessing comparison

As we can see CNN classifier yields the best results, therefore in the next experiments we limited ourselves to just this classifier.

We can also observe that stronger preprocessing gave better results than weaker preprocessing, achieving the best accuracy of 53.54% for Smaller BERT embedding. Unfortunately, even though we planned to do this and even have the implementation prepared, we didn't manage to conduct tests in time for other experiments with such processed text. In further experiments, we only use weakly preprocessed text.

Next, we tested CNN classifier for different embeddings and achieved the following results. In parenthesis next to the embedding methods' names, we can see the length of the given embedding of a single word.

Embedding	Accuracy	F1-score
GloVe (100)	53.73%	53.25%
Smaller BERT (128)	51.31%	51.05%
Base BERT (768)	56.48%	56.55%
Word2vec (300)	52.61%	52.65%

Table 4: Results for different embeddings

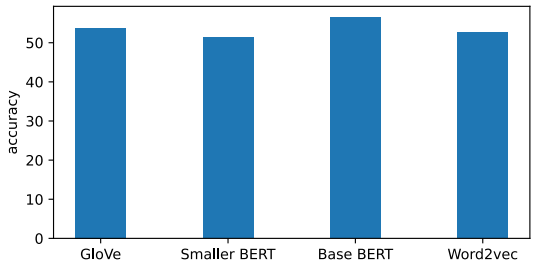

Figure 2: Embeddings comparison

Unsurprisingly, the best results were achieved for the most advanced embedding method, which was Base BERT, with 56.58% accuracy.

What is interesting though, second-best result (53.73%) was achieved by GloVe, whose embedding length is the smallest of all methods.

The plots in figures 3 and 4 present how loss and accuracy were changing while training was conducted for the best method, which is Base BERT with CNN:

Plots suggest overfitting – even though accuracy on the training dataset was very good, above 90%, on the testing dataset it was only 50%. In fact, we tried to help this occurrence by modifying CNN classifier by e.g. changing the dropout layer but unfortunately, it didn't seem to change the results a lot.

4.3 Lyrics with title

The last experiment was conducted for the additional approach and Base BERT + CNN architec-

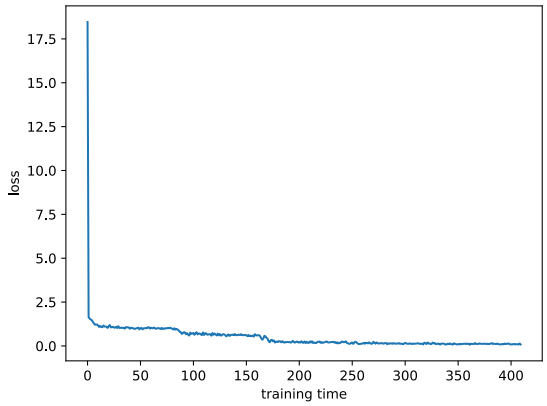

Figure 3: Loss function for Base BERT + CNN

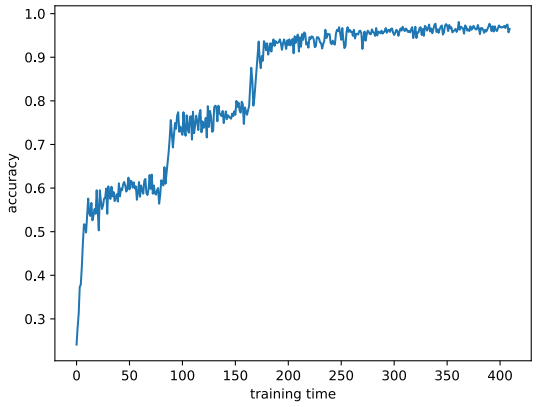

Figure 4: Accuracy to training time for Base BERT + CNN

ture. As mentioned earlier the data came only from the *Song lyrics from 79 musi- cal genres* dataset. The results are visible below.

	Lyrics only	**Lyrics + Title**
Accuracy	54.27%	57.17%
Bal. acc.	55.36%	52.65%
F1-score	54.80%	56.10%

Table 5: Comparison of results of lyrics with and without title

It can be seen that the addition of titles helped to achieve a little bit higher accuracy, but the balanced accuracy was actually lower. It is hard to tell if adding the title is valuable as the training is longer and the results are not significantly better.

5 Discussion

There are several contributions to this work. First of all, we were not able to find a detailed comparison of statistical and deep learning models. Often works focus on a selected group of models like transformers in case of [1], 1D CNNs in [2] or Support Vector Machines explored in [15].

What is more, often different datasets are being used, which overall makes the results difficult to compare. Our work compares older and newer models on different datasets. This creates a common ground and context for other works and researchers. When it comes to the comparison of our results to the current state-of-the-art, again the situation is a bit complex. State-of-the-art solution for different, in our opinion simpler, task of genre classification based on audio stands at 80.93% achieved in [2]. The best lyrics-based prediction accuracy that we were able to find was 77.63%, achieved by the BERT model in [1].

This cast a strong shadow on our best result of 57.17% but again, those results are difficult to compare. Why? This is due to the fact that the authors of mentioned SOTA preprocessed the data in an undocumented way, significantly simplifying the original problem present in the dataset. Naturally, as mentioned numerously in this document we also made changes to the formulation of the original problem, but due to the missing information from other works we are not able to reproduce this process in a similar way, hence making the results non-conclusive. What is worse, the mentioned article makes use of a different dataset, which makes the accuracy even harder to compare.

6 Conclusion

This work explored various approaches to music genre classification based solely on lyrics and was oriented on comparing numerous techniques. This was done in order to create a context for future researchers, which is currently missing in this area of study. In performed experiments, we used two different datasets with two types of inputs for models, four embedding techniques and four vastly different machine learning classifiers.

The final results are the following, 56.48% accuracy on lyrics with BERT embeddings and CNN classifier on our balanced dataset, and 57.17% accuracy on lyrics with titles for BERT embeddings and CNN classifier on our unbalanced dataset. Both of the datasets were described in detail in 3.2.

Those results may seem low, but the reader ought to remember that in the case of this task flawless classification may not be possible and currently the best result of 77.63% accuracy, whereas higher, comes with a set of problems described in 5.

7 Future work

The results of classification could, possibly to a large extent, still be improved and the final methods presented in this document ought to be treated as a decent starting point for future researchers.

Based on all the knowledge gained while working on this project we humbly want to present several ideas, which in our opinion will lead to further improvement in this topic. First of all, we see great potential in more extensive feature extraction, especially in form of adding sentiment vector alongside lyric embedding vector. This sentiment can be extracted directly from the lyrics themselves, hence does not break the assumption of incorporating only the text of the songs.

There can be observed a positive correlation between a model's complexity and performance on a test dataset, which leads to the conclusion that the potential of embedding is still not exhausted and a classification layer with a greater number of parameters could lead to better overall performance. We suspect that densely connected layers of neurons could outperform models presented in this work, but this hypothesis has to be put to the test.

Another matter, which could be addressed, is separating a single classification model into two models: the first one for classes with a disproportional number of observations and the second for a subset of data, which can be treated as a balanced classification task. Music genres present in the dataset from this work were mostly shadowed by one music genre, i.e. *Rock*. We see the potential of firstly classifying genre as rock vs non-rock and if a prediction is made for a non-rock class then using the second model for detailed classification.

Lastly and possibly the most obvious, yet still valuable addition would be to collect more data, especially from the last years and from classes which are underrepresented in current datasets. Most aspects mentioned above are to be addressed in our following work.

References

[1] Hasan Akalp et al. "Language Representation Models for Music Genre Classification Using Lyrics". In: *2021 International Symposium on Electrical, Electronics and Information Engineering*. ISEEIE 2021. Seoul, Republic of Korea: Association for Computing Machinery, 2021, pp. 408–414. ISBN: 9781450389839. DOI: 10.1145/3459104.3459171. URL: https://doi.org/10.1145/3459104.3459171.

[2] 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.

[3] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 238–247. DOI: 10.3115/v1/P14-1023. URL: https://aclanthology.org/P14-1023.

[4] Thierry Bertin-Mahieux et al. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[5] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

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

[7] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: (2016).

[8] 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.

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[10] 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.

[11] Thorsten Joachims. "Text categorization with Support Vector Machines: learning with mny relevant features". In: (1998).

[12] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: (2014).

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

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

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

[16] 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.

[17] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[18] 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).

[19] *NLU: State of the Art Text Mining in Python*. https://nlu.johnsnowlabs.com/. Accessed: 2022-12-20.

[20] 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).

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[22] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[23] Sebastian Raschka. "Naive bayes and text classification i-introduction and theory". In: (2014).

[24] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[25] Alexandros Tsaptsinos. "Lyrics-Based Music Genre Classification Using a Hierarchical Attention Network". In: *CoRR* abs/1707.04678 (2017). arXiv: 1707.04678. URL: http://arxiv.org/abs/1707.04678.

[26] 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.

[27] 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.

8 Appendix

8.1 Team contribution

Self-assessed authors' contribution is in table 6.

Author	Contributions	Workload
Bartłomiej Eljasiak	Exploratory data analysis, part of the report	20%
Aleksandra Nawrocka	Basic data preprocessing, refactorization of training code, adjustment of models and training code and conducting tests for title addition, part of the report, improvements after initial assessment	35%
Dominika Umiastowska	Advanced data preprocessing, preparation of models and training code and conducting tests	45%

Table 6: Authors' contribution

8.2 Some comments regarding presentation reviews

Here are some comments regarding suggestions and questions contained in the presentation's reviews:

1. Barplots were added to the report where it made sense.
2. Our main contribution is stated in the abstract.
3. We have used Smaller BERT and Base BERT because training took a lot of time
4. Title addition is described in section 3.6, there are no additional fields separating lyrics and title.
5. We have tested our models on the balanced dataset, which we have created by dropping redundant observations.
6. We think that adding audio would entirely change the project scope and make it significantly easier. That is not the point of our project.

8.3 Additional comments regarding the assessment of our project

1. We provide some of the models in the `models` subdirectory. Unfortunately, the majority of them are too big to be put into the repository. Moreover, it is possible that because of problems regarding repository and branches a small number of them were permanently deleted by accident.
2. Majority of our code is in notebooks. We believe that that makes the solution quite easily reproducible. The suggested order of code running would be:
   - downloading datasets (links are provided in `data_preprocessing` notebook) and putting them in `data/raw` directory,
   - running `data_preprocessing` notebook,
   - running `basic_classifier` notebook to obtain majority of results,
   - running `title_classifier` notebook to obtain results for title addition.

8.4 Hyperparameters of models

In the tables 7, 8, 9 we provide the hyperparameters of models.

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
Base BERT	John Snow Labs Spark NLP	small_bert_L2_768	768
Word2vec	John Snow Labs Spark NLP	word2vec_gigaword_300	300

Table 7: Hyperparameters of embeddings

Classifier	Library	Class	Optimizer	Epochs
Naive Bayes	sklearn	GaussianNB	-	-
Linear SVM	sklearn	SGDClassifier	-	-
XGBoost	xgboost	xgboost	-	-
CNN	tensorflow.keras	Sequential	Adam	5

Table 8: Hyperparameters of classifiers

Max. number of words in lyrics	Max. number of words in titles
400	15

Table 9: Other hyperparameters

Below there is a summary of the CNN classifier model for GloVe embeddings.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_15 (Conv1D)           (None, 40000, 8)          32
 max_pooling1d_10 (MaxPooling1D)  (None, 20000, 8)      0
 dropout_10 (Dropout)         (None, 20000, 8)          0
 conv1d_16 (Conv1D)           (None, 20000, 16)         400
 max_pooling1d_11 (MaxPooling1D)  (None, 10000, 16)     0
 dropout_11 (Dropout)         (None, 10000, 16)         0
 conv1d_17 (Conv1D)           (None, 10000, 32)         1568
 flatten_5 (Flatten)          (None, 320000)            0
 dense_10 (Dense)             (None, 128)               40960128
 dense_11 (Dense)             (None, 5)                 645
=================================================================
Total params: 40,962,773
Trainable params: 40,962,773
Non-trainable params: 0
_________________________________________________________________
```

Below there is a summary of the CNN classifier model for Smaller BERT embeddings.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_18 (Conv1D)           (None, 51200, 8)          32
 max_pooling1d_12 (MaxPooling1D)  (None, 25600, 8)      0
 dropout_12 (Dropout)         (None, 25600, 8)          0
 conv1d_19 (Conv1D)           (None, 25600, 16)         400
 max_pooling1d_13 (MaxPooling1D)  (None, 12800, 16)     0
 dropout_13 (Dropout)         (None, 12800, 16)         0
```

```
 conv1d_20 (Conv1D)              (None, 12800, 32)     1568
 flatten_6 (Flatten)             (None, 409600)        0
 dense_12 (Dense)                (None, 128)           52428928
 dense_13 (Dense)                (None, 5)             645
=================================================================
Total params: 52,431,573
Trainable params: 52,431,573
Non-trainable params: 0
_________________________________________________________________
```

Below there is a summary of the CNN classifier model for Base BERT embeddings.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_21 (Conv1D)           (None, 307200, 8)         32
 max_pooling1d_14 (MaxPooling1D)  (None, 153600, 8)     0
 dropout_14 (Dropout)         (None, 153600, 8)         0
 conv1d_22 (Conv1D)           (None, 153600, 16)        400
 max_pooling1d_15 (MaxPooling1D)  (None, 76800, 16)     0
 dropout_15 (Dropout)         (None, 76800, 16)         0
 conv1d_23 (Conv1D)           (None, 76800, 32)         1568
 flatten_7 (Flatten)          (None, 2457600)           0
 dense_14 (Dense)             (None, 128)               314572928
 dense_15 (Dense)             (None, 5)                 645
=================================================================
Total params: 314,575,573
Trainable params: 314,575,573
Non-trainable params: 0
_________________________________________________________________
```

Below there is a summary of the CNN classifier model for Word2vec embeddings.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_24 (Conv1D)           (None, 120000, 8)         32
 max_pooling1d_16 (MaxPooling1D)  (None, 60000, 8)      0
 dropout_16 (Dropout)         (None, 60000, 8)          0
 conv1d_25 (Conv1D)           (None, 60000, 16)         400
 max_pooling1d_17 (MaxPooling1D)  (None, 30000, 16)     0
 dropout_17 (Dropout)         (None, 30000, 16)         0
 conv1d_26 (Conv1D)           (None, 30000, 32)         1568
 flatten_8 (Flatten)          (None, 960000)            0
 dense_16 (Dense)             (None, 128)               122880128
 dense_17 (Dense)             (None, 5)                 645
=================================================================
Total params: 122,882,773
Trainable params: 122,882,773
Non-trainable params: 0
_________________________________________________________________
```

Below there is a summary of the CNN title classifier model for Base BERT embeddings.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 318720, 8)	32
max_pooling1d (MaxPooling1D)	(None, 159360, 8)	0
dropout (Dropout)	(None, 159360, 8)	0
conv1d_1 (Conv1D)	(None, 159360, 16)	400
max_pooling1d_1 (MaxPooling1D)	(None, 79680, 16)	0
dropout_1 (Dropout)	(None, 79680, 16)	0
conv1d_2 (Conv1D)	(None, 79680, 32)	1568
flatten (Flatten)	(None, 2549760)	0
dense (Dense)	(None, 128)	326369408
dense_1 (Dense)	(None, 5)	645

Total params: 326,372,053
Trainable params: 326,372,053
Non-trainable params: 0