Lyrical Genre Classification

Master MVA: Deep Learning Final Project Report

Dorian Desblancs
École Normale Supérieure Paris-Saclay
4 Avenue des Sciences, 91190 Gif-sur-Yvette, France
dorian.desblancs@mail.mcgill.ca

Kodjo Mawuena Amekoe
École Normale Supérieure Paris-Saclay
4 Avenue des Sciences, 91190 Gif-sur-Yvette, France
kodjo.amekoe@ens-paris-saclay.fr

Abstract

In this project, we explore the field of musical genre recognition using lyrics. We first introduce a 10000 track dataset that spans 12 genres. The lyrics present in it are then classified using a multitude of popular NLP algorithms such as CNNs, Bidirectional LSTMs, and Bidirectional GRUs superimposed on pre-trained GloVe or fastText word embedding layers. We also report results obtained using a pre-trained BERT model. Our best model obtained an average test set accuracy of 70%. This result is close to those obtained by more elaborate audio-based classifiers on much smaller datasets. It also surpasses past benchmarks on lyrical genre recognition tasks.

1. Introduction

"Jazz has borrowed from other genres of music and also has lent itself to other genres of music." - Herbie Hancock.

In the above quote, the jazz extraordinaire Herbie Hancock expresses one of music's most unique traits: its ability to continuously inspire itself. From jazz's influence on hip-hop to country's evolution into rock and roll, musical genre is an endlessly evolving construct that attempts to categorize songs. Classifying songs using genre can however be extremely tricky, as the latter can be extremely ambiguous.

In the world of technology, musical genre classification has been a popular problem for a long time. It also has numerous applications these days, as more and more companies aim to curate musical playlists and recommendations for their users. This is notably the case for Spotify, Deezer, and Apple Music. These companies track users' tastes so as to recommend new songs to their customers. Hence, the ability to automatically decipher a song's genre is incredibly valuable, and could potentially make these companies' algorithms far more accurate.

In 2002, Tzanetakis and Cook published the seminal paper *Musical Genre Classification of Audio Signals* [30]. This paper was the first notable attempt to classify music based on genre using classic signal processing methods such as Mel-Frequency Cepstral Coefficients [15]. Ever since, a plethora of signal processing-based methods have been developed to improve automatic genre recognition and music information retrieval. These however come with a very heavy downside: computational cost. A standard song is encoded using 44100 samples per second, hence making it impossible to train large models on large datasets. The need for less costly yet viable alternatives is therefore evident.

In this project, we tackle the problem of musical genre recognition from a lyrics-based point of view. We introduce a dataset of our own making containing 10000 song lyrics spanning 12 genres. We then report the classification results obtained using popular deep learning models. These include Convolutional Neural Networks (CNN) [14], Bidirectional Long Short-Term Memory networks (Bidirectional LSTM) [8], and Bidirectional Gated Recurrent Unit networks (Bidirectional GRU) [2] trained upon GloVe (Global Vectors for Word Representation) [24] and fastText [21] word embedding layers. We also report results obtained using the state-of-the-art BERT (Bidirectional Encoder Representations from Transformers) [5] language representation model. Our best model obtained an average test set accuracy of 68.36%. Note that the dataset and code we used throughout the project can be found here [1].

[1] https://github.com/d-dawg78/MVA_DL

2. Related Work

2.1. Musical Genre Classification

In their paper, Tzanetakis and Cook [30] attempted to solve musical genre classification using beat histogram features. These were developed using a series of signal transformations (notably filtering and downsampling). Histograms were extracted from a variety of songs spanning a 10-class dataset. They were then clustered using KNN. Each cluster was then evaluated for its accuracy. Tzanetakis and Cook's approach was the first approach to classify genres well. It achieved an accuracy score of 61%. Tzanetakis and Cook's work also heavily influenced the work of future researchers in the field of musical classification. The dataset they created, now commonly known as GTZAN, is the most heavily used dataset in the field. It contains 1000 30-second song excerpts that span a total of 10 genres.

Over the past few years, Convolutional Neural Networks have taken over the musical genre recognition problem. In 2014, Gwardys and Grzegorz achieved 78% on the GTZAN dataset. They used a CNN to extract spectrogram features. These were then classified using SVM [10]. More recently, Ghosal and Kolekar achieved a mind-boggling 94.2% [7] accuracy score on the dataset using an ensemble of Convolutional Neural Networks applied to a variety of signal features.

The methods outlined above are incredibly impressive, but are unfortunately hard to interpret. Most genre recognition systems are evaluated on the GTZAN dataset. It however contains many flaws and limitations. Most notably, some excerpts are mislabeled and the dataset is too small. Hence, most models trained upon it are massively overfit, and haven't been proven to generalize well on new data. A deeper exploration into the dataset's flaws can be found here [27]. More importantly, signal processing-based methods haven't yet shown an ability to generalize their knowledge to other problems such as musical sentiment analysis.

2.2. Natural Language Processing

The field of natural language processing (NLP), on the other hand, has been extremely active since the early 2000s. Most notably, deep learning methods for document classification have performed extremely well on tasks such as sentiment analysis and language translation. In the fast few years, two main paradigms have emerged for high performance on classification tasks [23].

The first is deep neural networks, such as CNNs, LSTMs, and GRUs, used in conjunction with pre-trained word embeddings. Word embeddings associate each word of a training corpora to a vectorial representation. This representation, usually between 50 and 300 dimensions, models the relationship a word has with its surrounding words. Words with the same meaning have highly similar vectors. The opposite holds true for antonymic words. Note that embeddings for each word can be generated in numerous ways. We will explore two classic embeddings, GloVe [24] and fastText [21] later in this report. Word embedding layers are usually used as the underlying input representation for deep learning algorithms. In 2014, Yoon Kim [12] achieved state-of-art performance on multiple sentence-level classification tasks using a CNN trained on top of pre-trained word2vec vectorial representations [22] (state-of-the-art embeddings at the time). The methodology we use in this paper is very similar to the one he used.

More recently, the transformer architecture [31] has been gaining traction in the NLP community, with models such as Generative Pre-trained Tranformer (GPT) [25] and Bidirectional Encoder Representations from Transformers (BERT) [5] achieving state-of-the-art performance on Natural Language Understanding and Generation tasks. In this project, we focus on BERT for multi-class text classification. BERT is a multi-layer bidirectional Transformer encoder. It encodes all inputs using WordPiece embeddings [33]. The model is then trained in an unsupervised fashion using a masked language model (MLM) [28], during which the model tries to predict masked tokens within each sentence, and Next Sentence Prediction, during which the model tries to determine whether a sentence actually follows the previous one. These two pre-training tasks allow the model to capture unique long-range dependencies between words. More importantly, unlike word embeddings, each word has a different embedding representation that depends on the words that surround it in a particular classification task. BERT can then be fine-tuned for any desired task. In our case we fine tuned BERT on our lyrics data set. We encourage the reader to read the following paper [5] for more information about this exciting model.

2.3. Lyrics-based Musical Genre Classification

In general, research linked to lyric-based music classification is quite sparse. Two papers released in 2017 did however explore the fields in an interesting way. The first can be found here [1], and introduced the MoodyLyrics dataset. This dataset annotates 2500 songs in binary fashion, for positive and negative sentiment. The authors achieved 75% on binary lyrical sentiment analysis using SVM.

Around the same time, Alexandros Tsaptsinos [29] published the state-of-the-art paper for lyrics-based genre classification. He achieved 49.77% on a 20-genre, 449 458-song, dataset using a Hierarchal Attention Model model. His model bases itself on Gated Representation Units [2] trained upon a GloVe [24] embedding layer. Note that the musical genres extracted for each track in his dataset are collected using iTunes' annotations. These base themselves on a unique genre for each artist, which can be highly problematic for artists that span multiple genres.

Finally, in 2007, Mayer and Naeumayer [19] achieved an accuracy of 63.5% on a 10-class dataset using SVM. They trained their model on both lyrical and audio features. Mayer and Rauber [20] further improved upon these results in 2011 using a cartesian ensemble of lyric and audio features. They achieved a top accuracy of 74.08%.

3. Methodology

3.1. Dataset

Our project is based upon a dataset we created ourselves. In order to extract songs by genre, we created a total of 12 playlists on Spotify. Each playlist was created using a subset of Spotify's general public, genre-based playlists. These included playlists such as *Country's Greatest Hits: The 80's* for country music and *Baila Reggaeton* for reggaeton music. A total of 121 playlists curated by Spotify and its users were used in order to create 12 playlists that span 12 genres. Each one of these 12 playlists was checked for duplicates. Note that we scanned the playlists for outlier tracks (tracks that definitely weren't of a said genre) numerous times using our musical knowledge and culture. The Spotify API [26] was then used to retrieve all the songs' names, artists, albums.

We then used the Genius lyrics scraper that can be found here [11]. This web scraper can access the lyrics present on the Genius website based on a song's name and artist. Out of the 130000 tracks we collected at the previous step, a little more than 10500 lyrics were returned. We then further removed all duplicate tracks and lyrics and filtered low-quality lyrics. The dataset was then reduced to 10000 songs for convenience purposes. The number of samples for each musical genre can be found in Table 1.

For each one of our classification models, we removed lyric headings. These contained information like the verse and chorus indices of a song. We also removed line skips and treated the lyrics of each song as one long document to classify.

3.2. Classification Models

3.2.1 Word Embeddings

In this project, we focused our attention on two popular sets of pre-trained word embeddings: GloVe (Global Vectors for Word Represenation) [24] embeddings trained upon the Wikipedia 2014 and Gigaword 5 [2] training corpora and fastText [21] embeddings trained upon the Wikipedia 2017, UBMC web base [17], and the `statmt.org` news dataset. The former contains embeddings for more than 6 billion tokens whilst the latter contains embeddings for more than 16 billion. Note that in both cases, we opted to use the 300-dimension vectors due to their higher performance.

[2]https://catalog.ldc.upenn.edu/LDC2011T07

Table 1. Table Title

Class Name	Number of Samples
Blues	328
Classical	492
Country	726
Dance	728
Disco & Funk	426
Jazz	886
Metal	1579
Pop	1445
Rap	1129
Reggae & Dub	664
Reggaeton	756
Rock	841

While GloVe and fastText embeddings both rely on the idea of encoding words into vectors based upon their surroundings, they are different in some key ways. Throughout training, each word is treated as an individual token to produce GloVe vector representations. These contain meaningful word encodings. However, rare words are often encoded quite poorly, as their vector representation relies solely on a few examples. FastText vectors, on the other hand, are created using the idea that words are composed of character n-grams. These character n-grams are vectorized like GloVe, and then summed to obtain a vectorial representation for each word. Hence, rare words can be encoded in a much more thorough fashion, as their representation is derived from a combination of n-gram vector representations. Recently, models trained upon fastText embeddings have slightly over-performed models trained upon GloVe and word2vec representations.

In this project, the words in each one of our lyrics are mapped to an embedding representation whenever possible. Note that we removed stopwords, punctuation, non-alphabetical tokens, and capitalization in order to map as many words as possible to a pre-trained vectorial representation. The mappings between words in our corpora and their associated embedding were then used to create a word embedding matrix. This matrix was used to create an embedding layer for three of the classification models we used: a CNN, a Bidirectional LSTM, and a Bidirectional GRU. These are covered in the next three sub-sections. We used the Keras [3] API to implement our networks due to its simplicity of use.

3.2.2 Convolutional Neural Network

Our CNN model was built upon the embedding layer described in the previous section. The embedding layer was first fed into a 1-dimension spatial dropout layer of parameter $p = 0.2$. Spatial dropout is a regularization technique that masks entire sequences in the training set at a rate p. In our case, this dropout layer improved our model's future layers' generalization capabilities. Furthermore, since words are highly correlated with their surroundings, spatial dropout allows for better regularization than regular dropout, which only masks individual words in a sequence.

The output of this dropout layer was then fed into a 1-dimensional convolutional layer composed of 256 filters of kernel size 3. We used an ReLU activation function to avoid the vanishing gradients problem. The output of this layer was then used as input for a global max pooling layer of dimension 1 and a sequence of fully connected layers with 48, 24, and 12 units. The first two fully connected layers used ReLU activations whilst the final one used a softmax activation.

During training, we used the Adam optimizer [13] with a learning rate value of 0.0001 and a batch size of 16. Cross-entropy loss was minimized throughout the process. Note that we found lower batch sizes to perform better on our validation set for each epoch. As demonstrated in this paper [18], lower batch sizes tend to regularize CNNs and limit over-fitting. Our CNN was trained for 30 epochs using GloVe embeddings and 40 epochs using fastText embeddings. These values gave the network ample time for convergence during training.

Note that we experimented with deeper CNN architectures. However, these often performed worse due to overfitting on the training set. We therefore stuck to this simple architecture, which only contains 244 468 trainable parameters. Each epoch took between 8 and 25 second to run.

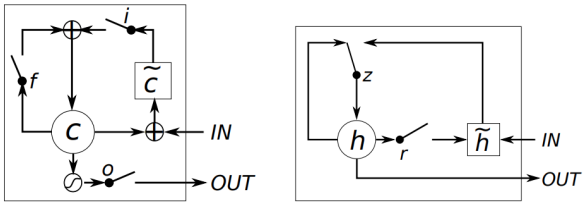
3.2.3 Long Short-Term Memory

Our LSTM network is extremely similar to the CNN network described previously. All the layers are the same except for the convolutional layer, which was replaced by a Bidirectional LSTM layer. This Bidirectional LSTM layer was composed of 100 units. Note that an LSTM unit is composed of four elements: a cell, an input gate, an output gate, and a forget fate. By using multiple units, an LSTM network can model long-range dependencies that occur in text, time series, or speech. A bidirectional LSTM layer reads its inputs in both a forward fashion (from start to finish) and backward fashion (from finish to start). This allows the network to better learn contextual dependencies that occur throughout the input.

Our Bidirectional LSTM network was trained using a batch size of 128 for better training speed using GPUs. We used the Adam optimizer with an initial learning rate of 0.001. The learning rate was then decayed every 10000 steps using exponential decay with a rate of 0.9. The model contains a total of 332 724 trainable parameters. Again, deeper architectures were found to overfit on the training set. Our network was trained using the same number of epochs as our CNN.

3.2.4 Gated Recurrent Unit

Figure 1. Illustration of (a) LSTM and (b) GR units. [4]



(a) Long Short-Term Memory (b) Gated Recurrent Unit

(a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.

Our GRU network replaces the LSTM layer of the previous model with a Bidirectional Gated Recurrent layer with 100 units. As illustarted in Figure 1, GRU units are very similar to LSTM units, the only difference being that they lack an output gate. They are therefore less memory-heavy, and have been found to perform better on smaller datasets due to their ability to better learn less frequent content [9]. Due to the fairly uneven nature of our dataset, we found these models to worth a try. Unsurprisingly, our GRU network only has 252 324 trainable parameters (more than 80 000 less parameters than our LSTM). It was trained in exactly the same fashion as our LSTM network.

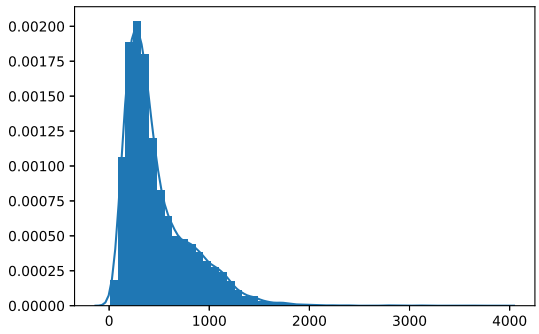
3.2.5 Bidirectional Encoder Representations from Transformers

In this project, we used HuggingFace's [32] Transformer library to implement our BERT model. We opted for the classic cased (includes capitalized letters) BERT model published in 2018 [5]. Our network was pre-trained on the Toronto Book Corpus [34] and English Wikipedia.

BERT comes with its own pre-processing scheme. First, all words and punctuation symbols are tokenized. Each sequence is then padded with a start token, an end token, and extra padding tokens up to a certain maximum length. Note that our BERT model sets the maximum number of tokens per sample to 512. Hence, a majority of our lyrical content was thrown out for the model to work. Figure 2 displays the distribution of our lyrics' number of tokens. Although a sizeable number of lyrics contain less

than 512 tokens before being pre-processed, we believe our model would perform even better by uncapping its input sequences' lengths. Unfortunately, most pre-trained BERT implementations limit the input size to decrease training time. After all, the model does have approximately 110 million parameters.

Figure 2. Number of Tokens per Lyric Distribution



The pre-trained BERT model has 12 layers. The last one is comprised of 768 hidden units. We added a dropout layer for regularization purposes ($p = 0.3$) and a fully-connected layer to map the network to our desired, 12-class output. During training, we used the AdamW optimizer [16] with a learning rate of 0.00002. Note that the learning rate was decreased linearly from its initial value. We used a cross-entropy loss function and only trained the network for 5 epochs. Finally, we were only able to use a batch size of 12 due to computational constraints. Note that for large models like BERT, larger batch sizes decrease training time significantly. We encourage the reader to use a much higher batch size if possible.

4. Results

In order to evaluate each one of our models, we first shuffled the dataset randomly. We then split it into a training set of size 8000, a validation set of size 1000, and a test set of size 1000. For each model, during each training session, the weights that achieved the highest validation accuracy over a pre-determined number of epochs were saved. These model weights were then used to evaluate our network on the test set. We shuffled the dataset five times and report the average optimal validation accuracy and average test set accuracy in Table 2.

When it comes to our word embedding-based models, we found that our Convolutional Neural Network architecture performed slightly better than our LSTM and GRU network architectures, with an average test set accuracy of 61.60% using GloVe embeddings and 62.56% using fastText embeddings. This is most likely due to the fact that lyrics are much less formal than regular language. Word

Table 2. Average Accuracy for Each Model

Model	**Validation Set**	**Test Set**
CNN + GloVe	63.32	61.60
LSTM + GloVe	61.72	60.24
GRU + GloVe	63.90	61.26
CNN + fastText	61.12	62.56
LSTM + fastText	58.24	59.18
GRU + fastText	60.82	62.26
BERT	71.00	68.36

contexts are also much smaller, since a typical sub-sequence within a lyric only lasts between 5 and 8 words, and less reliable, most notably due to the use of interjections. CNNs in NLP are capable of extracting high-level representations within lyrics and combine them for accurate classification. For lyrical genre recognition, these are more helpful than recurrent models.

We also found our GRU networks to perform better than our LSTM networks for musical genre classification. This is most likely due to the small size of our dataset and the prevalence of rare musical genres such as *blues*, *classical*, and *disco & funk* (though our dataset is much more balanced than previous work on the topic). GloVe embedding-based models peformed slightly better than fastText embedding-based models, although the difference is too small to suggest that one embedding model is categorically better than the other for this task.

Table 3. BERT Sample Classification Report

Class	**Precision**	**Recall**	**F1-Score**
Blues	0.39	0.42	0.41
Classical	0.78	0.80	0.79
Country	0.65	0.76	0.70
Dance	0.42	0.42	0.42
Disco & Funk	0.24	0.24	0.24
Jazz	0.92	0.77	0.84
Metal	0.86	0.83	0.84
Pop	0.54	0.56	0.55
Rap	0.89	0.81	0.85
Reggae & Dub	0.81	0.78	0.79
Reggaeton	0.99	0.93	0.96
Rock	0.45	0.54	0.49

Finally, out BERT model vastly outperformed our pre-trained word embedding-based models, with an average test

Table 4. BERT Sample Confusion Matrix

	Blues	*Classical*	*Country*	*Dance*	*Disco & Funk*	*Jazz*	*Metal*	*Pop*	*Rap*	*Reggae & Dub*	*Reggaeton*	*Rock*
Blues	**11**	0	5	0	2	1	2	1	0	0	0	4
Classical	0	**35**	1	0	0	3	1	2	0	0	0	2
Country	3	0	**53**	0	3	0	0	4	0	0	0	7
Dance	0	0	1	**31**	8	1	5	18	2	4	0	4
Disco & Funk	3	2	1	4	**8**	0	0	8	2	1	0	4
Jazz	5	2	8	0	1	**78**	0	2	0	0	0	5
Metal	0	3	1	5	0	0	**130**	4	2	0	0	12
Pop	0	3	6	21	4	1	4	**79**	4	2	1	15
Rap	0	0	0	3	3	0	0	13	**104**	3	0	3
Reggae & Dub	1	0	1	2	1	0	2	1	3	**50**	0	3
Reggaeton	0	0	0	0	0	0	0	4	0	1	**67**	0
Rock	5	0	4	8	3	1	8	11	0	1	0	**49**

set accuracy of 68.36%. It beat our previous best model average by 5.8% on the test set. Tables 3 and 4 display a more detailed report of our BERT results. Unsurprisingly, the genres that are most culturally and linguistically distinct were classified the best. Most notably, *reggaeton*, *rap*, and *reggae & dub* had f1-scores oscillating around 0.96, 0.85, and 0.79. These genres are known to be quite distinct in their wording. Our results with our other models confirmed this idea (results omitted for brevity).

On the contrary, the genres that we deemed as most similar greatly confused our classifiers, including BERT. This was notably the case for *blues*, *rock*, and *disco & funk*, who are each very similar to other genres in the dataset. *Rock* songs were often confused with *metal* songs due to their proximity, as were *disco & funk* tracks with *pop* tracks. We encourage the reader to read tables 3 and 4 carefully as we believe they capture cultural similarities between the musical genres in our dataset very well.

5. Discussion

We believe our results to be very similar, if not slightly better, than the state-of-art results obtained by Tsaptsinos [29] in 2017. He obtained an average accuracy of 49.77% on a 449 458-song, 20-class dataset. Our results are logically higher, since our dataset merely contains 12 classes. However, his dataset is more than 40 times the size of ours. We believe we would have achieved similar results on his dataset using our word embedding models, and probably outperformed his baseline using BERT.

More importantly, our results are in line or better than some of the signal processing methods we described previously. This suggests that genre classification using lyrics can be extremely powerful, and that lyrics carry a lot of information about a song's genre.

Our results could however be bettered. First, all the models we used were pre-trained on English corpora. Although

the majority of the songs in our dataset are indeed in English, a non-negligible amount are in French, Spanish, Italian, or another language. Word embeddings such as fastText and GloVe also exist in other languages, and could be combined with the embeddings we are currently using to encode foreign words. BERT has also already been trained on corpora spanning multiple languages. We hope we can one day use such a model for our lyrics classification task. We also hope to test our classifiers on a dataset that is more representative of the infinite number of musical genres present around the world. Note that such a task is extremely hard. Very recently, Epure et al. [6] conducted a study focusing on the perception of musical genres across culture. They found that genre perception varied greatly from language to language. The musical genre normalization methodology they proposed could however be used to create a more international, high-quality dataset.

Finally, we would like to highlight how light our embedding-based models are. These rarely took more than a few minutes to train on Google Colab or Google Cloud GPUs, and achieved very decent results compared to previous audio-based genre classification methods. Our BERT model took longer (approximately 15 minutes per epoch), but needed very few training epochs to converge towards optimal accuracy. In an era where automatic musical genre classification is increasingly relying on very deep networks trained from scratch, combining these lyrical classifiers with lightweight audio classifiers could potentially provide us with extremely good performance with minimal training time. After all, the field of NLP continuously provides researchers with pre-trained models and word or sentence representations that yield very good performance. These ressources should be used in the field of musical genre recognition, especially since lyrics so often capture a song's genre and character. On the other hand, audio-based classifiers can help address the major flaw of lyrics-based classification: its inability to classify instrumental music (music without lyrics). Audio classifiers also capture songs' structure much better, which is very helpful for genre recognition.

6. Future Work

We would like to outline a future research direction which expands upon the research done in this project. Our dataset currently displays each song's artist, name, album, year of publication, genre, and lyrics. We believe that the lyrical information we provide could also be used for unsupervised learning tasks. Songs could be clustered based on their lyrics. These clusters could then be studied to better study how algorithms separate language samples. Are these clusters associated with songs' year of publication? Their genre? Which genres are most often clustered together? Why?

The answers to these questions would offer unique insights into the evolution of music through time, and how certain genres have influenced each other. Over recent years, for example, *pop* songs have increasingly been inspired by *reggaeton*. Do unsupervised algorithms capture this trend? These questions deserve to be answered. More importantly, any successful unsupervised algorithm could seriously bolster the recommendation engines that suggest music to millions of us every day. The reader is welcome to use our dataset to do so.

7. Conclusion

In this project, we study the musical genre recognition problem through the lens of song lyrics. We propose a dataset that contains 10000 lyrics that span a total of 12 classes. These lyrics are then classified using a Convolutional Neural Network, a Long Short-Term Memory network, and a Gated Recurrent Unit network. All three models are superimposed upon a word embedding layer. This layer is obtained by associating words in our dataset to GloVe and fastText vectors, which have already been trained on large corpora and encode word meanings. We also report results obtained using a pre-trained BERT network. The first three models achieved average accuracies that oscillate around 61%. Our BERT model, however, achieved an average accuracy of 68.36%. These new results are better than previously established baselines using song lyrics. They also approach audio classifier accuracies. In the future, we hope to see our dataset explored through the lens of an unsupervised learning task. We also hope our models can be combined to light audio classifiers for better genre classification using less training time. Finally, we are eager to test our models on larger datasets, that encompass a larger array of musical genres. After all, the diversity that exists within music is what makes the field so fascinating.

Acknowledgements

Thank you to Master MVA's *Deep Learning* team for giving us the opportunity to complete this project.

References

[1] Erion Çano and Maurizio Morisio. Moodylyrics: A sentiment annotated lyrics dataset. In *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, pages 118–124, 2017. 2

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1, 2

[3] Francois Chollet et al. Keras, 2015. 3

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent

neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2, 4

[6] Elena V Epure, Guillaume Salha, Manuel Moussallam, and Romain Hennequin. Modeling the music genre perception across language-bound cultures. *arXiv preprint arXiv:2010.06325*, 2020. 7

[7] Deepanway Ghosal and Maheshkumar H Kolekar. Music genre recognition using deep neural networks and transfer learning. In *Interspeech*, pages 2087–2091, 2018. 2

[8] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005. 1

[9] Nicole Gruber and Alfred Jockisch. Are gru cells more specific and lstm cells more sensitive in motive classification of text? *Frontiers in Artificial Intelligence*, 3(40):1–6, 2020. 4

[10] Grzegorz Gwardys and Daniel Michał Grzywczak. Deep image features in music information retrieval. *International Journal of Electronics and Telecommunications*, 60(4):321–326, 2014. 2

[11] John W. Miller. Python wrapper for downloading lyrics and music metadata from the Genius.com API. https://github.com/johnwmillr/LyricsGenius, 2017. Online; accessed 21 January 2021. 3

[12] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 2

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[14] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 1

[15] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11, 2000. 1

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[17] Tim Finin James Mayfield Lushan Han, Abhay L. Kashyap and Johnathan Weese. UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, June 2013. 3

[18] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018. 4

[19] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Combination of audio and lyrics features for genre classification in digital audio collections. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 159–168, 2008. 3

[20] Rudolf Mayer and Andreas Rauber. Musical genre classification by ensembles of audio and lyrics features. In *Proceedings of International Conference on Music Information Retrieval*, pages 675–680, 2011. 3

[21] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. 1, 2, 3

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013. 2

[23] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 2

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 1, 2, 3

[25] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. 2

[26] Spotify AB. Spotify for Developers: Web API. https://developer.spotify.com/documentation/web-api/, 2021. Online; accessed 21 January 2021. 3

[27] Bob L Sturm. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*, 2013. 2

[28] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953. 2

[29] Alexandros Tsaptsinos. Lyrics-based music genre classification using a hierarchical attention network. *arXiv preprint arXiv:1707.04678*, 2017. 2, 6

[30] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002. 1, 2

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[32] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910, 2019. 4

[33] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 2

[34] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015. 4