Song Genre Classification Using Lyrics

Michael Alvin
mchlalvn@umich.edu

Ramgopal Chandrasekaran
ramgopal@umich.edu

Joshua Fitzpatrick
jtfitzpa@umich.edu

Keerthana Kolisetty
kkoliset@umich.edu

Abstract

Music is nearly ubiquitous in modern society, present in nearly all forms of media and playing a role in the daily lives of nearly everyone. Music contains information in a wide variety of types of information that can be used to classify the music by genre, sentiment, or any other number of categorization. This includes information from the raw audio signal, the songs metadata (e.g. artist, title, length, etc.), or the song's lyrics. This project attempts to evaluate classification of songs by genre using only the song lyrics. Various models for this task are evaluated and resulting classification accuracies are compared. Code has been made available at: `https://bitbucket.org/jtfitzpa/eecs595_project/`.

Keywords: classification, music, song lyrics, neural networks, pretrained models, bag of words, finetuned models

1 Introduction

1.1 Motivation

Music is "the art of combining tones to form expressive composition; any rhythmic sequence of pleasing sounds" (Merriam-Webster, 2019). Music can be found everywhere in the world, helps connect cultures and removes boundaries. It helps people find themselves and gives us a way to show how we feel inside.

There is an enormous volume of music available online and the ability to distinguish music is crucial and interesting. Songs can be classified into categories such as genre, mood, tempo, key etc.

People use genre keywords to search for songs and to find similarities between other songs. Classification of the songs into music genres is very useful and has a wide range of applications. This project restricts its focus to genre classification using lyrics, which requires significantly less compute than raw audio features.

1.2 This Project's Contributions

The problem that will be addressed is music genre classification using lyrics as features. This is a multi-class classification problem and the music will be classified into 5 genres: Rock, metal, pop, hip-hop and country. The following models have been implemented: a majority classifier, various bag of word models, and neural network models on the data. The models will be compared to the base model and vis-á-vis to evaluate the various methods used.

2 Prior Work

Most research explored genre classification using audio or a combination of audio and text features. There are a few relevant papers that have previously explored genre classification using only text lyrics.

(Teh Chao Ying et al., 2012) experimented with using lyric words and their part-of-speech tags to predict genre. K-Nearest Neighbors (KNN), Naive Bayes (NB), and Support Vector Machines (SVM) were used, with SVM achieving the highest accuracy of 39.94% over 10 genres.

(Fell and Sporleder, 2014) explored classification with n-gram models as well as more sophisticated features. This paper takes a more classical NLP approach to try to parse each song's lyrics rather than relying on deep learning models. Their models achieved an F-score for genre classification of 52.5 using a collection of 8 hand selected features for classification including length, slang usage, and echoisms.

More recently, (Tsaptsinos, 2017) experimented with using lyric words, their word embeddings, and hierarchical structure of songs to predict genre. Logistic Regression (LR), Long Short-Term Memory (LSTM), and Hierarchical Network (HN) were used in this paper, with LSTM achieving the highest accuracy of 49.77 over 20 genres and HN achieving the highest accuracy of 46.42 over 117 genres.

3 Methods

3.1 Data

The data that was used was obtained from a Kaggle data set from (Mishra, 2017) and it contains over 380,000 songs with lyrics, genre, year, song name and artist data from MetroLyrics. This dataset has 12 different genres but some of the genres don't have enough examples, so only the 5 genres with the most examples were used. Since the project focused on lyrics all the attributes were removed except for genre and song lyrics.

3.2 Preprocessing

In order to get the data ready for the models, a lot of preprocessing was done. The following was done to preprocess the data:

- Non-English songs were removed
- Capitalization and punctuation was removed
- Noise characters (html tags) were removed
- Lyrics with less than 100 words were removed

3.3 Majority Classifier

The majority classifier model simply predicts the most common genre in the dataset. Rock is the most common genre, so this model simply predicts Rock. Because Rock makes up about half the dataset, this simple model achieves 50% accuracy and serves as the baseline model.

3.4 Bag of Word Models

Several bag of word models were used to classify the songs lyrics into genres. The bag of words model is a way of representing text data when modeling text with machine learning algorithms. In bag of words models a sentence or a document is considered as a 'bag' containing words, disregarding grammar and even word order but keeping multiplicity. The models used cannot take in text directly as input; instead, the text needs to be converted to numbers. Three different ways of making this conversion were tested: CountVectorizer, TfidfVectorizer, and HashingVectorizer. The CountVectorizer encodes using length of the entire vocabulary and an integer count for the number of times each word appears in the document. The TfidfVectorizer method uses word frequency scores that try to highlight words that are more interesting. This method will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow the encoding of new documents. The HashVectorizer uses a hash function to encode the word to an integer. Simple test runs were done using all 3 encoding methods and TfidfVectorizer performed the best. The results displayed later in this paper were done using the TfidfVectorizer (Brownlee, 2019).

The bag of word models used were K-Nearest Neighbors (KNN), Decision Tree, Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machine (SVM). These models were built using the scikit-learn Python library. Different parameters were set in running the model to optimize performance. The KNN algorithm takes a bunch of labeled points and uses them to learn how to label other points. Decision Tree builds classification models in the form of a tree structure. It uses an if-then rule set which is mutually exclusive and exhaustive for classification. Random Forest is like decision trees but with using multiple decision trees. Logistical Regression is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. Naive Bayes is a probabilistic classifier based on Bayes Rule under an assumption which is the attributes are conditionally independent. In the SVM algorithm, we plot each data item as a point in n-dimensional space with the value of each feature being the value

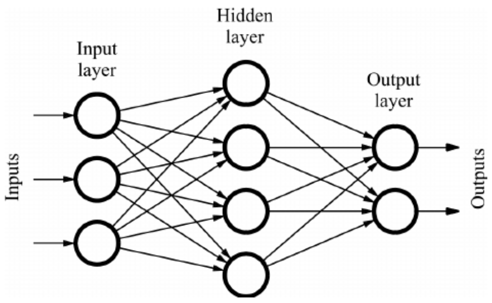

Figure 1: Feed-forward neural network with one hidden layer

of a coordinate (Sidana, 2019).

3.5 Neural Networks

3.5.1 Feed-Forward Network

The first neural network model tested on the song genre classification task was a basic feed-forward network. The feed-forward neural network is the first and simplest type of artificial neural network (ANN) devised. It is an ANN where connections between the nodes do not cycle or loop. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes, and then to the output nodes. An example of simple feed-forward network can be seen in Figure 1.

The feed-forward network was constructed using 2 hidden layers and 256 hidden dimensions. Different parameters were set in running the model to optimize performance and minimize overfitting, including: word embedding size (100, 300), dropout rate (0.3, 0.4, 0.5, 0.6, 0.7), optimizer (Adam, SGD).

3.5.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are types of neural networks that were originally developed for computer vision. These models use layers with convolving filters that are applied to local features. CNNs offer us a powerful feature function that is applied to constituting word-grams to extract high level features. CNNs have been applied in Natural Language Processing since 2008 (Young et al., 2017). An example of a CNN for text data can be seen in Figure 2. A disadvantage of using CNNs for sentiment analysis is that unlike LSTMs they do not

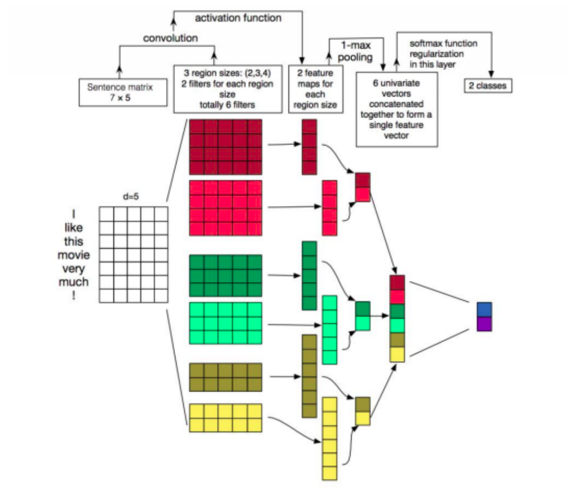
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 2: Example CNN for text data (Figure taken from (Young et al., 2017))

preserve sequential order or long distance contextual information (Young et al., 2017).

In this project, each song's lyrics are taken in and transformed into a word embedding matrix. A feature map was then generated by applying convolutional filters on the input embedding layer. Max-pooling is then done on each filter to reduce the dimensionality and produce a final sentence representation.

3.5.3 Long Short-Term Memory

Recurrent neural networks (RNNs) are special types of neural networks with loops in them that allow information to persist. The Long Short-Term Memory (LSTM) is a special kind of RNN, first introduced by (Hochreiter and Schmidhuber, 1997), which for most tasks works much better than the standard RNN architecture. The basic RNN structure can be seen in Figure 3. LSTMs were designed to avoid long-term dependency problem, and the model addresses the issue by integrating additional neural network layers that learn what types of information to pass through. An LSTM block can be seen in Figure 4.

In this project, the LSTM takes for input each song's lyrics as a single sequence of words and uses max-pooling of the hidden states for classification. GloVe word embeddings (Pennington et al., 2014) were used to represent the lyrics.

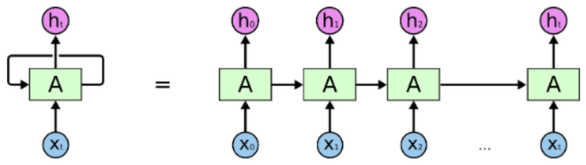
Figure 3: An unrolled recurrent neural network (Figure taken from (Olah, 2015))

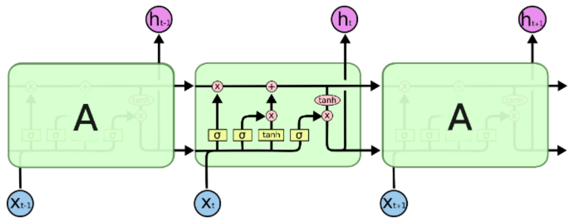
Figure 4: The repeating module in an LSTM contains four interacting layers (Figure taken from (Olah, 2015))

The bidirectional LSTM model was constructed using 2 hidden layers and 256 hidden dimensions. Different parameters were set in running the LSTM model to optimize performance and minimize overfitting, including: maximum vocab size (25,000, 50,000), word embedding size (100, 300), dropout rate (0.3, 0.5), optimizer (Adam, SGD), and learning rate (0.01, 0.1). The model was trained on Google Colab GPUs for 10 epochs each time, with training time that ranged from 1 hour (Adam) to 2 hours (SGD).

3.6 Fine-Tuning Pretrained Models

With the many recent advancements in NLP over the past couple years, the common theme among almost all of them has been the use of huge numbers of parameters and volumes of training data. With the introduction of models with larger numbers of parameters showing dramatic gains in performance, the obvious solution if sufficient compute is available is to make use of these models for whatever specific NLP task is of interest. However, the biggest impediment to this is the huge volumes of data and hundreds of GPU hours needed to train most of these models to obtain the best performance. This makes it largely infeasible for anyone without access to large corpora of data and plenty of money to spend on server time for GPUs to make use of these models. Thankfully, there are solutions to this using transfer learning. As shown in (Devlin et al., 2018), by simply fine-tuning a pretrained model with an added layer for the final task of interest, state of the art performance can be achieved using a fairly small amount of data and compute. To explore these techniques, a classifier was trained for the music genre classification task using 3 different base architectures: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and GPT-2 (Radford et al., 2019). These architectures are currently the state of the art for a wide variety of tasks and largely rely on the massive number of learnable parameters in their models to out-perform previous models on NLP tasks.

For the setup of the fine-tuning experiments, code from (McCormick and Ryan, 2019) was used as a starting point. The HuggingFace Transformers library (Wolf et al., 2019) was used to implement each of these pretrained models, which were then fine-tuned for 5-20 epochs on the training set.

For classification, the final hidden state of the transformer was taken and fed into a two stage feed-forward network with dropout and a ReLU nonlinearity between the feed-forward layers. Cross entropy loss was used to compare the predicted labels to the ground truth labels.

Weight freezing was also explored, i.e. leaving the weights of the pretrained classifier untouched and training only the weights of the final fully connected classification layer. This led to a decrease in both the training and text accuracies, so it was not explored further.

3.6.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a model introduced by (Devlin et al., 2018). BERT was designed as a language model which can be pretrained and subsequently fine-tuned for a variety of downstream tasks. BERT is an attention-based transformer drawing inspiration from (Vaswani et al., 2017), which was designed to be bidirectional, meaning it captures relationships among words in both the forward and backward direction, as opposed to many transformer models before it, which the authors of (Devlin et al., 2018) point out were limited in their representational capabilities since they were only unidirectional. Figure 5 shows how BERT can be used for sequence classification as in

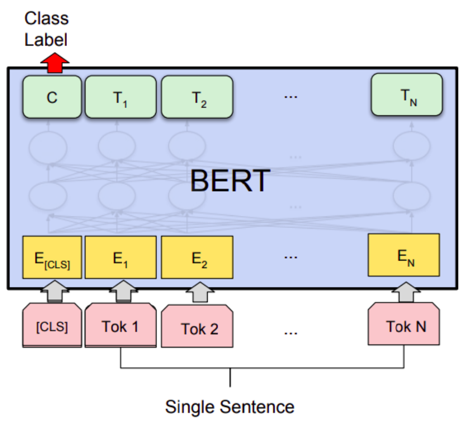

Figure 5: BERT architecture for sequence classification (Figure taken from (Devlin et al., 2018))

this project.

3.6.2 RoBERTa

A Robustly Optimized BERT Pretraining Approach (RoBERTa) is a language model introduced by (Liu et al., 2019). RoBERTa an extension of the original BERT model from (Devlin et al., 2018), which uses the same architecture as BERT, but uses more principled training strategies to produce a model which outperforms BERT on nearly all downstream tasks. The main contribution of (Liu et al., 2019) is to provide a better starting point for fine-tuning, along with training strategies that tend to lead to convergence to better models.

3.6.3 GPT-2

Generalized Pretrained Transformer 2 (GPT-2) (Radford et al., 2019) is OpenAI's second language model, released as a follow up to the original GPT model (Radford, 2018).

The core idea behind GPT-2 is that it is a language model, capable of generating new text by understanding the relationship among words across a wide variety of contexts. This is achieved primarily through massive amounts of data and tunable parameters. The models were trained to predict the next word in 40GB of internet text (Radford et al., 2019). An example of the architecture for GPT and GPT-2 can be seen in Figure 6. To date, OpenAI has released 4 different GPT-2 models with different numbers of parameters: Small (124M), Medium (355M), Large (774M), and XL (1.5B).

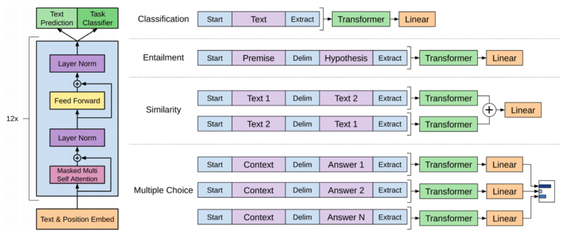

Figure 6: GPT architecture as presented in the original GPT paper (Radford, 2018). The architechure used in GPT-2 is the same, simply scaled up in size (Radford et al., 2019). (Figure taken from (Radford, 2018))

For this project, the Medium model was initially used to test the effect of using more a model with more parameters, but because of memory constraints only a small subset of the training data could be used because of the limited memory available on the Colab GPUs. For this reason, the Small model was used for larger training runs. It is worth noting that the small model is smaller than BERT, taking away GPT-2's size advantage.

4 Results

Results for each of the trained models are shown in Table 1. For the neural network models, the number of learnable parameters is listed, which gives an estimate of the representational capacity of the model, as well as a rough comparison of training time among the different models. Additionally, accuracy on the training set of approximately 146,000 examples is given for the neural network models. Test accuracy is given for all trained models, which is the classification accuracy on the held-out test set of approximately 18,000 examples.

4.1 Model Parameters

In this section are listed the parameters for each neural network model which gave the best performance.

- **Feed-Forward Network:** hidden layers: 2, hidden dimension: 256, dropout rate: 0.7, Adam optimizer, learning rate: 0.001

- **CNN:** word embedding dimension: 100, filters: 10, dropout rate: 0.5, Adam optimizer

- **LSTM:** hidden layers: 2, hidden dimensions: 256, maximum vocab size: 25,000, word embeddings dimension: 300, dropout rate: 0.5, Adam optimizer, learning rate: 0.001

- **BERT:** epochs: 10, maximum sequence length: 256, Adam optimizer, learning rate: 2e-5, batch size: 32, warmup rate: 0.1
- **RoBERTa:** epochs: 4, maximum sequence length: 128, Adam optimizer, learning rate: 2e-5, batch size 32, warmup rate 0.1
- **GPT-2:** epochs: 20, maximum sequence length: 128, Adam optimizer, learning rate: 5e-5, batch size: 16, warmup rate: 0.1

5 Discussion

5.1 Fine-Tuning

One of the primary goals of this project was to explore whether fine-tuning pretrained models could yield classifiers with better performance than prior attempts at music genre classification. To this end, the fine-tuned BERT model did perform about 2.5% higher than the rest of the models, but considering the number of parameters in the BERT model is over 17 times as many as the next highest performing model, the LSTM. In many other domains of NLP, fine-tuning models has led to significant improvement over the previous state of the art.

On the song genre classification task evaluated here, the fine-tuned BERT model performed the best out of all models, including the fine-tuned RoBERTa model. This is interesting because RoBERTa and BERT share the same architecture, with their only difference being that RoBERTa is designed to be more optimized for fine-tuning, which ended up being the opposite of the results seen here. This indicates that perhaps there are tasks that the original pretrained BERT model from (Devlin et al., 2018) is a better starting point. Specifically this could be the case when the data for the downstream task is not necessarily very similar to the original data used for pretraining, in which case a less optimized model might allow more freedom for the downstream task to fit the new data.

One of the more surprising results was the performance of the fine-tuned GPT-2 model, which only achieved a test accuracy of 55.22%. This is surprising because GPT-2 is currently considered state of the art for most NLP tasks. In addition to the concerns with lack of domain overlap between the dataset used here and the training data for these pretrained model, another factor in the performance of the GPT-2 model is that the strength of GPT-2 is in its ability to generate new text and to perform well on natural language understanding tasks such as SQuAD and CoQA. Thus, it is very possible that the GPT-2 model is not trained to pay attention to features which are more helpful for classification, a problem which a few epochs of fine-tuning may not fix.

Notably, while the models were being fine-tuned, one of the limitations found in the pretrained models was the inability to overfit a small training set. Clearly, each of these models has more than enough representational capability to overfit a small set of only a couple hundred examples, but in practice the model was only able to correctly predict around 70% of the training images it had seen. This could point to a fundamental difference in how the pretrained models expect inputs to be structured compared to the training data specific to this project. During training, a decrease in the loss of 2-3 orders of magnitude was observed, which should generally coincide with an increase in at least training accuracy, and hopefully test accuracy as well. Interestingly though, in the case of the fine-tuned models, often the opposite was the case; the training accuracy would increase until it reached somewhere between 60-70% and then from there it would decrease back to around 55%. This phenomenon occurred repeatedly, across multiple hyperparameter combinations, despite the fact that the cross entropy loss continued to decrease consistently, meaning it was not an issue with the learning rate or other hyperparameter settings.

5.2 Noisy Data

One thing that was observed when looking through the lyric data was how noisy it was. A significant portion of the data was not in English, and for the lyrics which were in English, many of the words were misspelled or slang terms. Correction was attempted by removing non-English songs from the dataset, but this does not completely solve the issue of slang terms. These could be as simple as ”runnin” versus ”running”. Without stemming or similar

Model	**Learnable Parameters**	**Training Accuracy**	**Test Accuracy**
Base Model	N/A	N/A	50.06%
KNN	N/A	N/A	62.00%
Decision Tree	N/A	N/A	50.4%
Logistic Regression	N/A	N/A	67.1%
Naive Bayes	N/A	N/A	54.5%
Random Forest	N/A	N/A	60.00%
SVM	N/A	N/A	65.30%
Feed-Forward Network	2,627,333	73.05%	67.40%
CNN	50,576,322	76.34%	67.80%
LSTM	10,222,909	**85.99%**	68.02%
Fine-Tuned BERT	**177,952,517**	72.20%	**70.50%**
Fine-Tuned RoBERTa	125,240,069	67.87%	65.86%
Fine-Tuned GPT-2 (Small)	125,129,477	56.45%	55.22%

Table 1: Results for each model trained. N/A indicates models which did not have any learnable parameters.

processing, these words will be treated differently, even though they have the same meaning. This could have been an issue in particular with the fine-trained models, as the pretrained models were trained for more well structured text inputs. Since the lyrics are not very well structured in most cases, lacking a lot of punctuation, this likely was at least part of the reason the fine-trained models didn't improve much compared to other models trained. A possible solution to this would be to train these large models (e.g. BERT, RoBERTa, GPT-2) from scratch, but that is well beyond current computational capabilities, as these models all take hundreds, if not thousands of GPU-hours to train. Another consideration to make is that music is not always easy to divide into clearly defined categories. Many songs are a hybrid of two or even many genres, so choosing a single class may not be the most meaningful way to represent a song. One possible way to improve this classification would be to provide multiple more specific tags for each song and predict multiple tags at inference time. However, this would significantly increases the work required in collecting data, since labeling would require more expertise on the subgenres a given song might belong to.

5.3 Multi-Label Classification

Another consideration to make is that music is not always easy to divide into clearly defined categories. Many songs are a hybrid of two or even many genres, so choosing a single class may not be the most meaningful way to represent a song. One possible way to improve this classification would be to provide multiple more specific tags for each song and predict multiple tags at inference time. However, this would significantly increases the work required in collecting data, since labeling would require more expertise on the subgenres a given song might belong to.

5.4 Data Distribution

In the training dataset of size approximately 146,000 songs, the distribution is as follows: Rock (74,000), Pop (29,000), Hip-Hop (19,000), Metal (14,000), and Country (10,000). There is a possibility that the non-uniformity of the dataset have compromised the models' performances, i.e. the models may tune their parameters mostly on the Rock songs, which comprised of about half the dataset. Given more time, this could be explored more by obtaining a more representative dataset or simply by trimming the dataset to achieve uniformity of genres.

5.5 Song Components

Most songs are comprised of several components, including, but not limited to, intro, verse, chorus, hooks, etc. It is possible that certain components of a song provide more hints for genre classification than other components. Given more time, this project could be extended by exploring the amount of information that is conveyed by different components of songs.

6 Future Work

Extending the work on fine-tuning some of the larger models used would require access to significantly more compute, but with access to this, one possibility would be to train a transformer from scratch and evaluate its performance on the dataset used for this project. Taking another step backwards, collecting a new dataset would perhaps be a way to improve the quality of the models' predictions by providing the pretrained models with lyrics that more closely align with the data they were trained on.

7 Conclusion

Recent research explorations have demonstrated that pretrained generalized language models are powerful systems that can be applied for many language tasks. In particular, these systems have been utilized for classification, question answering, and entity recognition tasks. Our goal for this project is to tackle the task of classifying music genre and experiment with using pretrained models for the task.

(Tsaptsinos, 2017) showed the effectiveness of neural network models for this task. In their research, LSTM performed the best in classifying 20 genres with 49.77% accuracy. In our experiments, the LSTM classifies 5 genres with 68.02% accuracy, second to Google's BERT which classifies 5 genres with 70.5% accuracy. This suggests that pretrained models can be used to improve results in classifying music genres.

In (Tsaptsinos, 2017), the neural network models performs substantially better than bag of words models. In their research, the Logistic Regression classified 20 genres with 38.13% accuracy compared to LSTM's 49.77% accuracy. However, in our project, BERT's performance is only 3% higher than that of logistic regression. From the results, we gather two explanations: (1) the dataset quality may not have been optimal and/or (2) the lyrics/text feature is simply not enough to predict a song's genre.

8 Acknowledgements

We are grateful to Professor Joyce Chai and Cristian-Paul Bara for their guidance and inspiration.

References

Jason Brownlee. 2019. How to prepare text data for machine learning with scikit-learn, Aug.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Michael Fell and Caroline Sporleder. 2014. Lyrics-based analysis and classification of music. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 620–631, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Chris McCormick and Nick Ryan. 2019. Bert fine-tuning tutorial with pytorch, Jul.

Merriam-Webster. 2019. music.

Gyanendra Mishra. 2017. 380,000+ lyrics from metrolyrics. https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics.

Chris Olah. 2015. Understanding lstm networks, Aug.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019. Better language models and their implications. *OpenAI blog*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Mandy Sidana. 2019. Types of classification algorithms in machine learning, Jul.

Teh Chao Ying, S. Doraisamy, and Lili Nurliyana Abdullah. 2012. Genre and mood classification using lyric features. In *2012 International Conference on Information Retrieval Knowledge Management*, pages 260–263, March.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. *CoRR*, abs/1707.04678.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709.