Artist Classification using Bert Pre-Trained

Anat Cohen[1], **Shir Friedman**[2], **Amit Shreiber**[2]

[1]Department of Electrical Engineering, Tel-Aviv University
[2]Department of Industrial Engineering, Tel-Aviv University

Abstract

Text classification is an important and common task in supervised machine learning [Bužić and Dobša, 2018]. The goal of our research was prediction of song performer based solely on lyrics. For the embedding part we used Bert model and compared it with naive method: TF-IDF.
Language model pre-training has proven to be useful in learning universal language representations [Sun et al., 2020]. As a state-of-the-art language model pre-training model, BERT (Bidirectional Encoder Representations from Transformers) has achieved amazing results in many language understanding tasks. A dataset that has been created consists of lyrics performed by 19 different artists, 8255 songs in total. Best results were achieved by fine-tuning (unfreezing entire model) Bert For Sequence Classification model: accuracy of 0.45.

1 Introduction

Text classification is an important and common task in supervised machine learning. Its application is in email spam detection, sentiment analysis, language detection of written text, classification etc. Many classifiers can be used for document classification. Some of them are neural networks, support vector machines, Naive Bayes classifier and k-nearest neighbours, etc. [Bužić and Dobša, 2018].

The quantity of music, especially on the internet, is growing rapidly and its organizing is a challenging task. Therefore, successfully modelling artist classification could yield benefits for automation in the music industry. For example, such a model could be used to detect copyright violations or identify songs that sound stylistically similar to another artist as part of a recommendation system. [Nasrullah and Zhao, 2019]

Classification can be made according to genre, mood, performer, geographical region, etc. To make classification successful, one can rely on audio features such as tempo, rhythm, timbre, pitch, loudness or lyric features such as word and sentence length, word frequencies, word n-grams, sentence and phrase structure, errors, synonyms, rhyme patterns etc. According to [Hu and Downie, 2010] most existing work on automatic music mood classification is based on audio features (spectral and rhythmic features are the most popular).
This fact was a motivation to classify artists based only by song lyrics.
The paper is organized as follows. Section 2 presents related methods including Bert and TF-IDF. In Section 3 we design experiments and constructed the dataset. In Section 4, the proposed methodology for artist classification is given. In Section 5, we present the results and compare between different models.

2 Methods

2.1 Bert

Language model pre-training has been shown to be effective for improving many natural language processing tasks [Radford et al., 2018]. These include sentence-level tasks such as natural language inference and paraphrasing, which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level [Devlin et al., 2018]. There are two existing strate-

gies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based approach, uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pretrained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations. Current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. BERT: Bidirectional Encoder Representations from Transformers, alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task [Taylor, 1953].

2.1.1 Bert for Text Classification

BERT-base model contains an encoder with 12 Transformer blocks, 12 self-attention heads, and the hidden size of 768. BERT takes an input of a sequence of no more than 512 tokens and outputs the representation of the sequence. The sequence has one or two segments that the first token of the sequence is always [CLS] which contains the special classification embedding and another special token [SEP] is used for separating segments. For text classification tasks, BERT takes the final hidden state h of the first token [CLS] as the representation of the whole sequence. A simple softmax classifier is added to the top of BERT to predict the probability of the label.

2.2 TF-IDF

Textual data needs to be converted to numbers and the most widely used method to process textual data into numbers is TF-IDF. In TF-IDF, text data is transformed into vectors without compelling the exact sequence of word order into consideration. Each word in the corpus is correlated with a number by TF-IDF that shows how significant each word to the corpus. Once the words are converted into numbers, the numerical values of TF-IDF are fed to supervised learning classifiers in a context that machine learning methods can interpret. Essentially, TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF-IDF numbers than common words. Given a document collection D, a word w, and an individual document $d \subseteq D$, we calculate:

$$w_d = f_{w,d} \cdot log(|D|/f_{w,D})$$

where $f_{w,d}$ equals the number of times w appears in d, $|D|$ is the size of the corpus, and $f_{w,D}$ equals the number of documents in which w appears in D [Ramos, 2003].

2.3 Tokenizing

Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. The aim of the tokenization is the exploration of the words in a sentence. The list of tokens becomes input for further processing such as parsing or text mining. Stemming is the process of conflating the variant forms of a word into a common representation, the stem. For example, the words: "presentation", "presented", "presenting" could all be reduced to a common representation "present". This is a widely used procedure in text processing [Kannan et al., 2014].

2.4 Neural Network for Classification

Neural networks have emerged as an important tool for classification. The recent vast research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods. The advantage of neural networks lies in the following theoretical aspects. Neural networks are data driven

self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. In addition, neural networks are nonlinear models, which makes them flexible in modeling real world complex relationships. Finally, neural networks are able to estimate the posterior probabilities, which provides the basis for establishing classification rule and performing statistical analysis [Richard and Lippmann, 1991].

2.4.1 Dropout

The term "dropout" refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random. Dropout prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently [Srivastava et al., 2014].

2.4.2 Batch Normaliztion

Batch Normalization is a technique that enables faster and more stable training of deep neural networks, by stabilizing the distributions of layer inputs. This is achieved by augmenting the network with additional layers that set the first two moments (mean and variance) of the distribution of each activation to be zero and one respectively. Then, the batch normalized inputs are also typically scaled and shifted based on trainable parameters to preserve model expressivity. This normalization is applied before the non-linearity of the previous layer [Santurkar et al., 2019].

2.4.3 Weight Decay

Weight decay is a regularization technique by adding a small penalty, usually the L2 norm of the weights (all the weights of the model), to the loss function. It is widely interpreted as a form of L2 regularization because it can be derived from the gradient of the L2 norm of the weights in the gradient descent setting. The purpose of weight decay is to improve the generalization performance of neural networks by encouraging the weights to be small in magnitude [Zhang et al., 2019].

2.4.4 Early Stopping

A problem with training neural networks is in the choice of the number of training epochs to use. Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that stops training once the model performance stops improving on the validation dataset.

3 Experiments

The goal of this research was to find out if the classifier can correctly identify the performer only by lyrics. A set of data was made for the purpose of research, and the data was prepared for processing. Subsequently, the model was trained and evaluated. In addition, we have tested how a more naive embedding method like TF-IDF will affect the prediction results. Another objective of our study was to test if fine-tuning the Bert model will improve the model compared to using a Pre-trained Bert model.

3.1 Dataset

There appears to be no reliable large dataset of lyrics with author attribution, so we have constructed our own dataset. Song lyrics were obtained via the Genius API using Lyricsgenius package in python. For each artist dataset, we downloaded the lyrics to all available songs by each artist, and removed duplicates. We obtained a 3-artist dataset (1485 Songs), a 5-artist dataset (2174 Songs), and a 19-artist dataset (8255 Songs). Table 1 represents the song count for each artist.

4 Methodology

4.1 Preprocessing

When the BERT model was trained, each token was given a unique ID [Devlin et al., 2018]. Hence, when we want to use a pre-trained BERT model, we will first need to convert each token in the input sentence into its corresponding unique IDs. There is an important point to note when we use a pre-trained

Artist	Song Count
Alice Cooper	331
Beck	411
Bee Gees	403
Bob Dylan	726
Bon Jovi	303
Britney Spears	192
Bruce Springsteen	481
David Bowie	418
Elvis Costello	520
Elvis Presley	707
Eric Clapton	432
Madonna	332
Neil Young	339
Rod Stewart	456
The Beach Boys	448
The Beatles	378
The Rolling Stones	445
U2	341
Van Morrison	592

Table 1. The artists and song counts from the 19-artist dataset

model. Since the model is pre-trained on a certain corpus, the vocabulary was also fixed (30,000 token vocabulary). In other words, when we apply a pre-trained model to some other data, it is possible that some tokens in the new data might not appear in the fixed vocabulary of the pre-trained model. This is commonly known as the out-of-vocabulary (OOV) problem. For tokens not appearing in the original vocabulary, it is designed that they should be replaced with a special token [UNK], which stands for unknown token. Hence, BERT makes use of a WordPiece algorithm that breaks a word into several subwords, such that commonly seen subwords can also be represented by the model. In order to inform the model where the start and end of each sentence, we will use two special tokens:
[CLS] token at the beginning, and the [SEP] token at the end of each input text. The BERT model receives a fixed length of sentence as input. For sentences that are shorter than this maximum length, we will have to add paddings (empty tokens) to the sentences to make up the length. The token [PAD] is used to represent paddings to the sentence. In our project, we have tokenized our input data using the the transformers Package [Wolf et al., 2020]. In particular, we used the function *encode plus*, which does the following in one go:

1. Tokenize the input sentence
2. Add the [CLS] and [SEP] tokens.
3. Pad or truncate the sentence to the maximum length allowed
4. Encode the tokens into their corresponding IDs Pad or truncate all sentences to the same length.
5. Create the attention masks which explicitly differentiate real tokens from [PAD] tokens

The "attention mask" tells the model which tokens should be attended to and which (the [PAD] tokens) should not.

4.2 Bert Model

we have tested two models using Bert and trained them in two different ways:

- Bert Model - we used this model for embedding the data, and used the embeddings as the input to a classifier model that was trained on this specific task.
- Bert For Sequence Classification - a Bert model that includes a classifier. we have fine-tuned all the layers in the model (the Bert layers and the classifier layers).

we wanted to represent the differences between the two approaches. There are some advantages for fine-tuning the model:

- The pre-trained BERT model weights already encode a lot of information about the language. As a result, it takes much less time to fine-tune the model [Mohanty et al., 2016].
- Because we are using a pretraind model, allows us to fine-tune our task on a much smaller dataset than would be required in a model that is built from scratch [Too et al., 2019].

- Finetuned models are more accurate compared to models trained from scratch.

4.3 Bert For Sequence Classification

One method we examined is Bert For Sequence Classification. BertForSequenceClassification is the normal BERT model (12 hidden layers) with an added single linear layer on top for classification that we used as a song classifier. The entire pre-trained BERT model and the additional untrained classification layer was fine-tuned for our specific task.
For BertForSequenceClassification we tested different values of the following hyper-parameters: batch size, weight decay, and learning rate. Number of epochs was defined according to early stopping criterion. The parameters that yield the best results were:
batch size: 8
learning rate: 2e-5
weight decay: 0.001

4.4 Classification Model

We build from scratch a classifier network. The input for the model was embedding data. We have used Bert for embedding the tokenized data. Each layer of BERT captures the different features of the input text and the last layer gives the best performance [Sun et al., 2020]. We have tried other layers and combining some layers outputs. For example, we used the eleventh layers, we summed the last four layers results, etc. The model achieved the best results using just the last layer. In order to decide about the number of epochs, we used early stopping in which we stopped training as soon as the accuracy on the validation set after 3 epochs was not improved. The network is made of one hidden layer with batch normalization and dropout following the input and hidden layers. We used cross validation to determine the layer sizes and dropout probability. The following parameters were also determined using cross validation: learning rate, batch size and weight decay.

4.5 TF-IDF

TF-IDF method was used for embedding each song. First we removed punctuation from the lyrics, lemmatized using WordNetLemmatizer, and lower cased. afterwords all stop words removed from the data. Then each word was computed with TF-IDF formula and resulted with a features TF-IDF matrix. Each row represents a particular song lyric and each column is a unique word and its corresponding TF-IDF value. The embeddings were inserted as input to the classification model for artist prediction.

5 Results

For consistency, we report all results as accuracy and Top-k accuracy (higher is better). Top-N accuracy means that the correct class gets to be in the Top-N probabilities for it to count as "correct". Table 2 compares between the three different datasets and three different models. According to our expectations, we can see from the results that the smaller number of artists, the model performs with higher accuracy rate. The Fine-tuned Bert for sequence classification model outperform all other models, both in respect to test accuracy and top-k accuracy. Second best is the pre-trained bert model.

6 Conclusion

Artist prediction using only lyrics is a complicated task. Unlike author identification tasks, the artist

Dataset					
	19-artist		**5-artist**		**3-artist**
Model	**Accuracy**	**Top-5 accuracy**	**Accuracy**	**Top-3 accuracy**	**Accuracy**
Pre-trained Bert + classifier NN	0.3906	0.731	0.6572	0.921	0.7631
Bert For Sequence Classification	0.4468	0.791	0.6619	0.924	0.7700
TF-IDF + classifier NN	0.3373	0.683	0.6436	0.9057	0.7037

Table 2. Test results

performing the song is not necessarily the one who wrote the lyrics. This complicates the prediction task because lyricists with different styles can write for the same artist. The best results were achieved using fine-tuning Bert For Sequence Classification model. In addition using the pre-trained BERT model just for embedding the data outperformed the TF-IDF embeddings. We can see that even for a task that doesn't require a sensitive analysis of each word in the sentence and the connections between the words, there is a big advantage for using a state of the art language model like Bert. From the results of the models, we can learn that fine-tuning Bert For Sequence Classification model makes it more accurate.

References

[Bužić and Dobša, 2018] Bužić, D. and Dobša, J. (2018). Lyrics classification using naive bayes. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1011–1015.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Hu and Downie, 2010] Hu, X. and Downie, J. S. (2010). When lyrics outperform audio for music mood classification: A feature analysis. In *ISMIR*.

[Kannan et al., 2014] Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., and Nithya, M. (2014). Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16.

[Mohanty et al., 2016] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7:1419.

[Nasrullah and Zhao, 2019] Nasrullah, Z. and Zhao, Y. (2019). Music artist classification with convolutional recurrent neural networks. *CoRR*, abs/1901.04555.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Ramos, 2003] Ramos, J. (2003). Using tf-idf to determine word relevance in document queries.

[Richard and Lippmann, 1991] Richard, M. D. and Lippmann, R. P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483.

[Santurkar et al., 2019] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2019). How does batch normalization help optimization?

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[Sun et al., 2020] Sun, C., Qiu, X., Xu, Y., and Huang, X. (2020). How to fine-tune bert for text classification?

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

[Too et al., 2019] Too, E. C., Yujian, L., Njuki, S., and Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279. BigData and DSS in Agriculture.

[Wolf et al., 2020] Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

[Zhang et al., 2019] Zhang, G., Wang, C., Xu, B., and Grosse, R. (2019). Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*.