

Top Music Hit Classification Using BERT

Sumedh Shah, Natali Ojeda Meneses, Kanika Mahajan
UC Berkeley School of Information
{sumedhshah94, nojedam, kanikamajahan}@berkeley.edu

ABSTRACT

Music evolves over time and identifying features that make a song popular is challenging. However, the music industry can immensely benefit by incorporating specific features like certain lyric phrases and musical attributes to produce more hit songs that will make the Billboard charts. This project explores feeding in chunks of song lyrics and its music metadata into a fine-tuned BERT model to identify hit songs and hones in on the importance of lyrics compared to the song's metadata. Compared with traditional classification models like Naive Bayes and Logistic Regression, BERT achieves higher performance in classifying hits. Training BERT consistently with newer songs would make it possible to understand the ever evolving direction of music and seems like a promising approach to produce more hit songs.

Keywords

BERT, lyrics, non-lexical, metadata

1. INTRODUCTION

The music industry is very competitive and profitable but also risky as it is unclear which aspects of music influence our preference for certain songs. With this project, we want to raise the question: Can someone predict whether a recently produced song will become a hit? We define a hit as a song that has charted on the Billboard Weekly Hot 100 charts for any length of time. This topic is usually referred to as Hit Song Science. Pachet[1] described it in 2012 as “an emerging field of investigation that aims at predicting the success of songs before they are released on the market”.

2. BACKGROUND

In neurology, two aspects of songs are typically tested to determine which aspects of music matter most for preference formation[2]. Inspired by this research, we decided to measure the influence of lyrics and audio features in the popularity of a song. Even though the majority of previous research papers primarily use audio features, lyrics[3] have

been more recently incorporated, as they are more useful than audio features for predicting popularity. Audio features, however, also help to boost the performance of the models[4].

The most frequent techniques utilized to predict song hits have typically been Logistic Regression, Naive Bayes, K-Nearest Neighbors, and SVM. In contrast, our project proposes to use Bidirectional Encoder Representations from Transformers (BERT), a relatively new (Google, 2018) and novel technique since it is pre-trained on a large corpus of unlabelled text. Because of this, the model starts to pick up the deeper and intimate understandings of how the language works. Another advantage of BERT is bidirectionality, meaning that BERT learns information from both the left and the right side of a token's context during the training phase. For those reasons, we hypothesized that BERT would obtain better results than traditional models in predicting a hit song.

3. DATASET OVERVIEW

The songs, lyrics, and their metadata were gathered from Kaggle, where the data was scraped from the Spotify Web API. Our dataset had songs that span from the year 2000 to 2020. To account for time series dependency and leakage that might happen by using future data to train the model, we decided to split our dataset using a cutoff year of 2017 based on the distribution of songs per year. Any songs released prior to 2017 were the training set and the remaining was our test set. To determine which songs were hits from the entire set, we gathered data from the Billboard Weekly Hot 100 website, where they had records dating back to early 1900s, and left merged the Kaggle dataset with this data. *Please refer to Appendix A.1 for more details on Data Extraction and Cleaning.*

3.0.1 Feature Engineering

For additional variables, we included a star variable, indicating that the artist had hits previously, and believe that this will improve the model output. Considering them meaningless, Singh and Brown (2014) [3] decided to eliminate non-lexical vocables (lalala, ooh ohh, etc.); however, after our own exploratory data analysis, we assessed whether this potentially uninvestigated signal could differentiate hit songs from non-hit songs and included this variable into our model, believing that it will help the model output. *Please refer to Appendix A.2 to see the wordclouds analysis for the non-lexical vocables variable.*

3.0.2 Oversampling

Our population contains about 20% of hits songs and is therefore unbalanced. This poses a risk of producing a model with poor predictive performance, especially for the minority class. To solve this problem, we first split the training data into a training set and validation set and then decided to randomly oversample the hits in the training set to have 40:60 proportion of hits and non-hits.

4. BASELINE MODELS

4.0.1 Naive Bayes

For our first baseline model, we decided to choose the Naive Bayes classifier as it has a proven record performance in several text classification applications (e.g. e-mail spam classification) and lyrics classification tasks, while also being computationally efficient in classifying long texts like lyrics [5][6]. Finally, Naive Bayes is a classification algorithm that uses Maximum Likelihood estimates to classify documents and assumes that all features are independent of each other.

The goal of the Naive Bayes model was to estimate the posterior probability $P(A | B)$ of a song belonging to class hit vs. non-hit given the metadata and words present in the lyrics. As part of text pre-processing we did common transformations like removing punctuation and stop words, converting words to lowercase, stemming and lemmatization.

We then used a bag of words TF-IDF vectorization approach, similar to prior works on analysis for classification tasks like mood classification [7][8]. Our basis for selecting TF-IDF was that as song lyrics have a lot of repeated phrases and words, the IDF component will help measure the importance of a word in lyrics with respect to a song rather than focusing on total counts of words. After creating the TF-IDF matrix we combined it with the song metadata and added it to a multinomial Naive Bayes model. As metadata column names were also words in the lyrics, we renamed metadata columns to avoid conflicts while merging with the TF-IDF matrix.

4.0.2 Logistic Regression

Using a similar approach, we also built a logistic regression model to evaluate as another baseline model because it is often used in classification problems and seemed to be a simple, scalable solution to gauge the complexities and performance in conducting this analysis with deep learning models. However, due to the high dimensions of the TF-IDF matrix, the number of columns significantly increased, causing a curse of dimensionality issue for our logistic regression model. To correct this, we performed linear dimensionality reduction using truncated singular value decomposition (SVD), a preferred approach over PCA for sparse matrices like TF-IDF on lyrics[9]. We also used L2 regularization to avoid overfitting and checked for multicollinearity in metadata features using Variance Inflation Factor (VIF). Thankfully, none of the features were severely collinear ($VIF > 10$). **Table 1** shows the results with running both models with lyrics alone and adding metadata to lyrics.

4.0.3 Baseline Results and Analysis

One might expect that the conditional independence assumption on which Naive Bayes is based would not be true

Table 1: Baseline Model Performance

Model	Accuracy	F1-score
Naive Bayes (lyrics)	0.855	0.08
Logistic Regression (lyrics)	0.724	0.28
Naive Bayes (lyrics+metadata)	0.865	0.04
Logistic regression (lyrics+metadata)	0.729	0.34

for lyrics as well. However, Naive Bayes performed surprisingly well for us as a baseline, similar to its performance in other classification problems. The most feasible explanation could be that the dependencies among words in lyrics either evenly distribute among hits and non-hits or cancel each other out [10], as what ultimately affects classification is the combination of dependencies among attributes. However, we would have to look at the theoretical constructs and study the local and global dependency joint probability distributions among attributes to be certain about this.

Even though our Naive Bayes' model accuracy was higher, the F1-score for Logistic Regression actually came out on top. Due to the imbalance of hit songs and non-hit songs in our test set, Naive Bayes tended to guess several true negatives whereas Logistic Regression guessed more true positives, indicating that Logistic Regression might have the edge on correctly distinguishing more hit songs from non-hit songs. Even though regularization was added to our Logistic Regression model, the discriminative nature of Logistic Regression most likely contributed to the higher F1-score. While we are most concerned with accuracy, F1-score is also an important metric to consider for our model in order to predict the optimal number of hits. This will be further examined when we fine-tune our BERT classification model. *Please refer to Appendix B.2 for the Baseline Confusion Matrices for both Naive Bayes and Logistic Regression.*

4.0.4 Feature Importance

In order to understand the importance of each feature in contributing to our prediction of hit, we calculated the feature importance for logistic regression based on order of importance. This refers to the coefficient descending order after taking the maximum value. The results indicate that the top 79 variables correspond to lyrics. The top 3 metadata variables are related to non-lexical vocables, followed by song attributes (loudness) and 'star' variables. However, we concluded that overall, the lyrics variables are more predictive than the metadata variables. *Please refer to Appendix B.1 for the details on our Feature Importance analysis.*

5. FINE-TUNING BERT

The bag of words approach that we used in our baseline models often fails to capture the semantic relationship between words as they are just based on counts. Pre-trained language models offer superior advantages for text analysis as they look at word similarities and contexts rather than just counts. Our goal behind this project was that of transfer learning where we use already pre-trained models for our text of classifying lyrics as a hit or non-hit [11]. For our main model we decided to use BERT for several reasons. First, BERT fuses left and right context's to pre-train a deep bidirectional Transformer which would be beneficial for lyrics

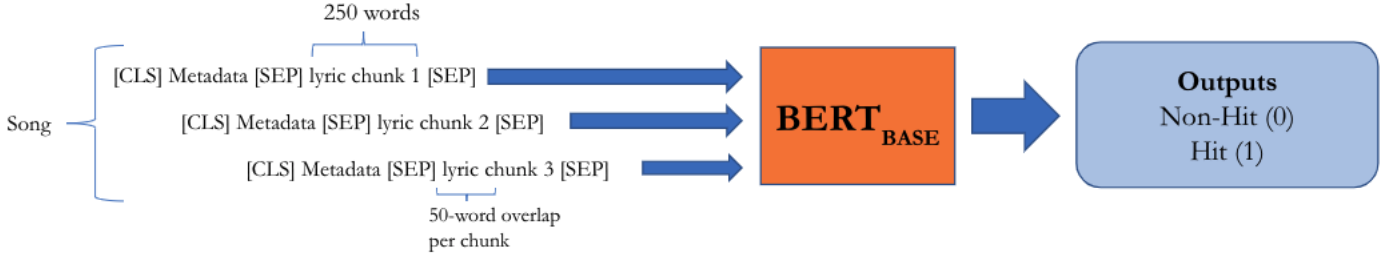


Figure 1: Song Lyric Chunking for BERT

analysis [11]. Next, BERT is trained on various sources like wikipedia and book corpuses, and these corpuses are somewhat similar to our lyrics structure. The [CLS] token whose embedding is a representation of the text sequences can be easily used to fine-tune BERT by adding an additional output classification layer to perform the classification task of predicting hit vs. non-hit[12].

5.0.1 Processing Long Text for BERT

Unlike our baseline models we did not perform any text pre-processing on BERT. BERT is a contextual model and pre-processing steps like removing stopwords that represent negation (e.g. not, never) would prevent BERT from learning the full context of the song. [13] In order to process songs with lengths greater than the maximum token length of 512 that BERT can handle, we created 250-word chunks of the lyrics for each song that would individually be fed into the BERT model. Because we broke our song’s lyrics up into chunks, we decided to focus on BERT_{BASE} as this was less computationally expensive and this allowed for faster training of data and later experimentation. [14] We decided to break up the entire song, as opposed to just the beginning or end, because BERT has the ability to process large amounts of text with a low test error rate. [13] As shown in Figure 1 for each chunk, we left 50 words of overlap between each chunk and for one of our experiments, we prepended song metadata, as text for each song, to the beginning of each lyric chunk. For example, for the song “Cool” by the Jonas Brothers, we prepended “genre rb danceability medium loudness medium speechiness medium acoustictness low valence high tempo low num_nonlexvoc high star yes”, followed by a [SEP] token to distinguish the metadata portion of the song chunk from the actual song chunk to BERT_{BASE}. Similar to the baseline models earlier, we ran another experiment with a similar architecture but just without the metadata and [SEP] token prepended to the lyric chunk.

5.0.2 Model Architecture and Implementation

For our base classification model, we utilized the Tensorflow pre-trained BERT model and attached a neural network layer with a dropout value of 0.1, learning rate of 5e-5, and all of the layers frozen. In addition, [15] recommended using a batch size of 8 and training over 3 epochs for each of our experiments. The earlier layers in BERT help learn some of the general text representation of the lyrics. For fine-tuning, we experimented with unfreezing 1-4 layers [13]. Specifically, we wanted to observe how the metadata portion of our chunk (represented as a sequence of words) interacts

Table 2: BERT Model Performance

Model	Accuracy	F1-score
BERT Base(lyrics)	0.875	0.328
BERT Fine-Tuned(lyrics)	0.878	0.328
BERT Base(lyrics+metadata)	0.875	0.336
BERT Fine-Tuned(lyrics+metadata)	0.879	0.357

with a pre-trained language model which might be expecting sentences. Furthermore, since you are relying on the CLS token, you want it to learn what to pay attention to in the metadata as well as the lyrics and what to ignore in order to make a better prediction. Ultimately, this becomes a balancing act of training over more of our chunks as opposed to using BERT’s existing pre-trained corpus. After experimenting, we unfroze 3 layers for our fine-tuned model as the fourth layer had a small dip in test accuracy. *Please refer to Appendix C.1 for the full BERT classification model summary*

As part of hyperparameter tuning, we primarily decided to look at the learning rate while keeping the number of layers constant. We varied our base learning rate between 5e-6 and 5e-4 and kept it small because BERT is already pre-trained and we didn’t want to override the actual learning rate from the transformer layers and potentially cause catastrophic forgetting [16], [14].

6. RESULTS AND DISCUSSION

6.0.1 Aggregation of results

After the classification model finished training, we ran the predictions for each song chunk in the test set. Recall that each song chunk contains BERT’s wordpieces, that include the metadata and the lyric chunk of one song. Therefore, each song chunk had its own probability and we compiled all of the chunk probabilities for each song. We implemented an OLS regression function that calculates coefficients for each respective chunk for each song and aggregates probabilities for all of the chunks into an aggregated probability for the song. We decided to use a regression to obtain the ‘weights’ of each probability by minimizing the sum of the squares of the differences between the observed dependent variable (hit /no hit variable) in the given dataset and those predicted by the linear function of the independent variables (the probabilities). The smaller the differences, the better the model fits the data. Hence, this method provides us with

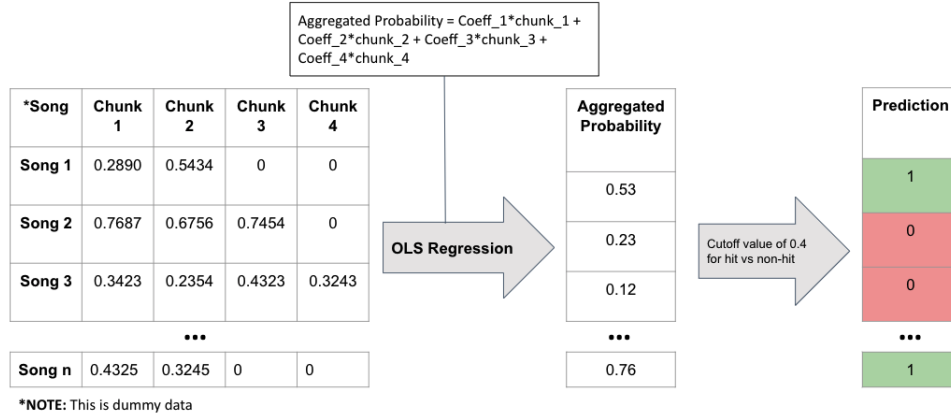


Figure 2: Song Chunk To Prediction Function

the probability weights that better satisfy that condition.

After coming up with the aggregated probabilities, we selected an appropriate cutoff value between 0 and 1 to determine which case would be considered a predicted hit or no-hit, as shown in Figure 2. To do so, we plotted the test accuracies and F1-scores per cutoff value. As seen in Figure 3 for the fine-tuned BERT_{BASE} model, we selected a cutoff value of 0.4. Since it is not possible to maximize two metrics at the same time, we tried to select a cutoff that returns a high enough accuracy without completely sacrificing the F1-score. The reason for the previous is that we are interested in predicting the true positives and true negatives correctly. However, since we have an imbalanced sample (20% of hits), the accuracy will be high just by predicting the non-hits correctly. The F1-score then becomes relevant since it is a better measure of the incorrectly classified metric than accuracy.

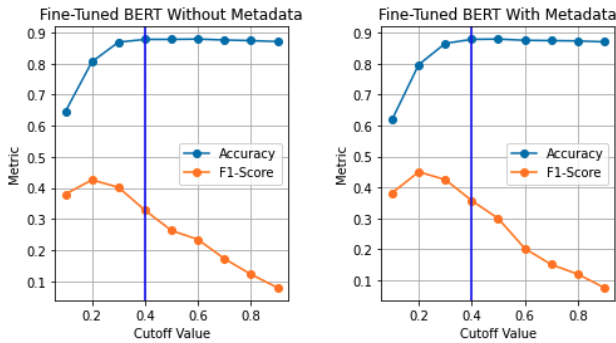


Figure 3: Fine-Tuned BERT Without vs With Metadata

6.0.2 Results

Our results in Table 2 show similar accuracies between our BERT_{BASE} base classification models with and without metadata. However, we noticed a slight difference in F1-scores where the model with metadata performed better by 0.8%.

We observed from hyperparameter tuning that a lower learning rate slightly increased the accuracy of our model (0.1%).

After incorporating the new learning rate of 5e-6, the accuracies of the BERT_{BASE} Fine-Tuned (lyrics only) and BERT_{BASE} Fine-Tuned (lyrics+metadata) models increased slightly. In addition, the F1-score went up by 2.1% for the model with the metadata. All of the BERT model accuracies beat the baseline accuracies, as well as the baseline F1-scores, showing that the BERT model architecture worked well for our purpose of classifying songs as hits or non-hits. We also notice this in Figure 4, where the ROC curve for BERT curves further to the left than both of our baseline models and also has a greater area under the curve. Please refer to Appendix C.2 for the final fine-tuned BERT Confusion Matrix.

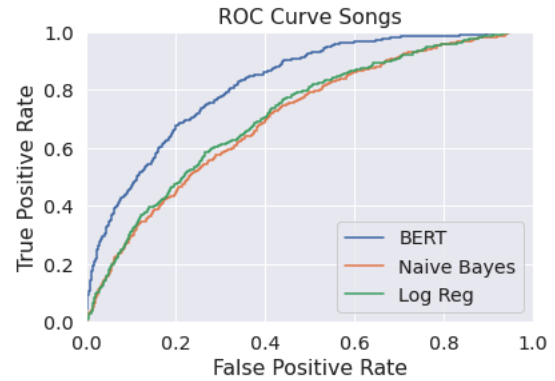


Figure 4: ROC Curve Comparison

6.0.3 Limitations

Our BERT classification model had a few limitations. Firstly, BERT has limited long-term memory (512 tokens) per document and suffers from major limitations in terms of handling long sequences. Furthermore, the self-attention layer has a quadratic complexity in terms of the sequence length n . [17] Secondly, BERT uses a learned positional embeddings scheme, which means that it won't likely be able to generalize to positions beyond those seen in the training data. In order to process each song, we had to chunk our data to feed it into BERT. In addition, BERT is computationally intensive and its feasibility in a production environment is

a limitation for the real world application of this project. Finally, because the Spotify dataset had more songs in the later years, the volume of data was skewed. We need to slide our training set window over to more recent years in order to give BERT the opportunity to learn from newer music data. Also, to further understand errors BERT is making we looked at trends and lyrics in metadata for songs that were wrongly classified. While there were no major trends observed in the metadata we did notice from the lyrics that these songs had negative/denial/sad emotion. On further research on BERT we found that there is some evidence in literature that BERT struggles to show generalizable understanding of negation and this could be one potential reason for us as well [18]. However this was based on manual observation of few songs and warrants further robust analysis.

7. CONCLUSION AND FUTURE STEPS

Based on our analysis, the value added by metadata was negligible to our model performance. On the other hand, lyrics alone are important in determining whether a song will become a hit. Violation of the conditional independence assumption did not appear to affect the performance of Naive Bayes as it performed surprisingly well in terms of accuracy. The limitation of the logistic regression classifier in not capturing complex non-linear decision boundaries within the lyrics might have contributed to its lower performance. Overall, BERT performed better in terms of both accuracy and F1-score compared to our baseline models. This further corroborates that fine-tuned BERT is capable of achieving a high performance in text classification.

For future steps, we will look into training BERT for each genre, gathering predictions per genre, and ensembling them into one final model. Because each genre has its own emphasis (e.g. pop is more about the sound and rap is more about the words), BERT can learn about intrinsic qualities in each genre’s songs and will hopefully make more accurate predictions on whether a song is a hit or not. We can further improve the input into our BERT classification model by distinguishing the verses in each song, like the chorus, with additional [SEP] tokens. This will enable BERT to learn the structure of the songs and which verses are most important. We noticed in recent years that song hits are more profane towards women in certain genres like rap and more songs nowadays discuss emotions and mental health. Societal issues (level of female profanity and discussion about mental health emotions) are prevalent in music and it would be interesting to include them as additional features in distinguishing hits from non-hits. Finally, we noticed a common theme of romance in hit songs in a given year and can encode that as a feature, where we evaluate the number of romantic words as a fraction of the total number of words in the song.

8. ACKNOWLEDGMENTS

We would like to thank our W266 NLP instructor, Mark Butler, for his continued guidance in helping us craft and execute this project. We would also like to thank Sumedh’s friend, Nand Dalal, that works as a Machine Learning engineer at Google, for his expertise and guidance in understanding the ins and outs of the BERT architecture and tying it into our overall goal.

9. REFERENCES

- [1] François Pachet and Pierre Roy. Hit song science is not yet a science. In *ISMIR*, pages 355–360, 2008.
- [2] Psyche Loui. Learning and liking of melody and harmony: Further studies in artificial grammar learning. *Topics in Cognitive Science*, 4(4):554–567, 2012.
- [3] Abhishek Singhi and Daniel G Brown. Hit song detection using lyric features alone. *Proceedings of International Society for Music Information Retrieval*, 30, 2014.
- [4] Abhishek Singhi and Daniel G Brown. Can song lyrics predict hits. In *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research*, pages 457–471, 2015.
- [5] Dalibor Bužić and Jasminka Dobša. Lyrics classification using naive bayes. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1011–1015. IEEE, 2018.
- [6] Shuo Xu. Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.
- [7] Menno Van Zaanen and Pieter Kanters. Automatic mood classification using tf* idf based on lyrics. In *ISMIR*, pages 75–80, 2010.
- [8] Teh Chao Ying, Shyamala Doraisamy, and Lili Nurliyana Abdullah. Lyrics-based genre classification using variant tf-idf weighting schemes. *Journal of Applied Sciences*, 15(2):289–294, 2015.
- [9] Truncated svd. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html>. Accessed: 2022-04-09.
- [10] Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.
- [11] 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.
- [12] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*, 2019.
- [13] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019.
- [14] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE, 2019.
- [15] Googleresearch. <https://github.com/google-research/bert>. Accessed: 2022-04-09.
- [16] Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. What happens to bert embeddings during fine-tuning? *arXiv preprint arXiv:2004.14448*, 2020.

B. BASELINE MODELS

B.1 Feature Importance

There are many types and sources of feature importance scores, although popular examples include statistical correlation scores, coefficients calculated as part of linear models, decision trees, and permutation importance scores. Almost all the algorithms in machine learning assign score or weight to the features. The coefficients calculated for each feature as part of the learning process can be used to score the features which can be accessed through either `coef_` method or `feature_importances_` method. Naïve Bayes classifiers don't offer an intrinsic method to evaluate feature importances. These methods work by determining the conditional and unconditional probabilities associated with the features and predict the class with the highest probability. Thus, there are no coefficients computed or associated with the features you used to train the model. Hence, we calculated the feature importance for logistic regression only by sorting the absolute value of the coefficients estimated for each variable. We are interested in the magnitude of the weight more than the sign. We had a total of 729 variables introduced in the logistic regression. 700 correspond to the SVM output after reducing dimensions. Initially, we had 10000+ variables corresponding to the lyrics TF-IDF. 29 variables correspond to metadata. Table1. below shows a summary of the feature importance.

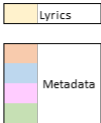
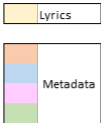
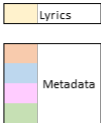
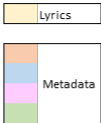
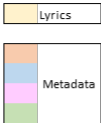
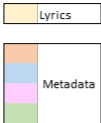
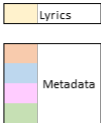
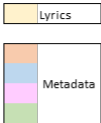
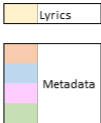
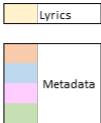
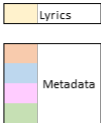
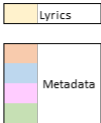
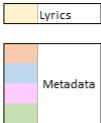
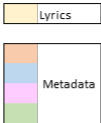
Variable	Order of importance		Variable	Order of importance
Lyrics	1-79		Lyrics	635-637
Non-lexical vocable	80		Genre	638
Lyrics	81-119		Lyrics	638-649
Non-lexical vocable	120		Song_attributes	650
Lyrics	121-338		Genre	651
Non-lexical vocable	339		Lyrics	652-653
Lyrics	340-383		Song_attributes	654
Song_attributes	384		Lyrics	655-665
Lyrics	385-395		Song_attributes	664
Star	396		Lyrics	665
Lyrics	397		Genre	666
Star	398		Lyrics	667-673
Lyrics	399-418		Song_attributes	674
Song_attributes	419		Lyrics	675-679
Lyrics	420-443		Song_attributes	680
Non-lexical vocable	444		Lyrics	681
Lyrics	445-505		Song_attributes	682
Song_attributes	506		Genre	683
Lyrics	507-517		Lyrics	684-685
Song_attributes	518		Song_attributes	686
Lyrics	519-558		Lyrics	686
Song_attributes	559		Song_attributes	688
Lyrics	560-561		Song_attributes	689
Song_attributes	562		Lyrics	690-699
Lyrics	563-581		Song_attributes	700
Genre	582		Lyrics	701-716
Lyrics	583-633		Song_attributes	717
Song_attributes	634		Lyrics	718-729

Figure 6: Feature Importance: Logistic Regression.

B.2 Baseline Confusion Matrixes (Lyrics + Meta-data)

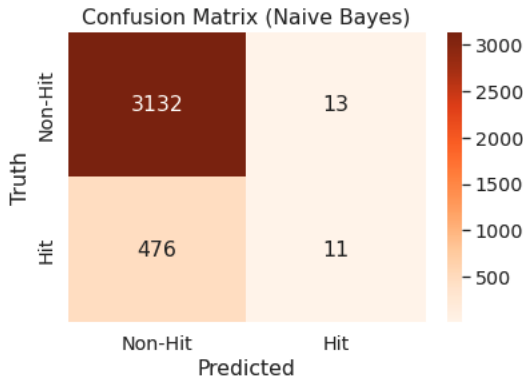


Figure 7: Naive Bayes Confusion Matrix

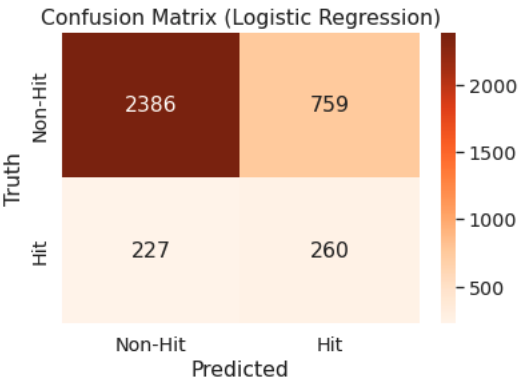


Figure 8: Logistic Regression Confusion Matrix

C. FINE-TUNING BERT

C.1 Classification Model

Layer (type)	Output Shape	Param #	Connected to
attention_mask_layer (InputLayer)	[(None, 512)]	0	[]
input_ids_layer (InputLayer)	[(None, 512)]	0	[]
token_type_ids_layer (InputLayer)	[(None, 512)]	0	[]
bert (TFBertMainLayer)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 512, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['attention_mask_layer[0][0]', 'input_ids_layer[0][0]', 'token_type_ids_layer[0][0]']
get_first_vector (Lambda)	(None, 768)	0	['bert[0][0]']
hidden_layer (Dense)	(None, 200)	153800	['get_first_vector[0][0]']
dropout_75 (Dropout)	(None, 200)	0	['hidden_layer[0][0]']
classification_layer (Dense)	(None, 1)	201	['dropout_75[0][0]']

=====
Total params: 109,636,241
Trainable params: 109,636,241
Non-trainable params: 0

Figure 9: Classification Model Specifications

C.2 BERT Fine-tuned Confusion Matrix (Lyrics + Metadata)

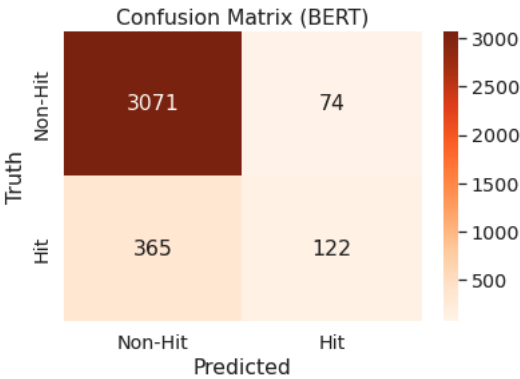


Figure 10: Confusion Matrix - BERT