CS 274C Project:

An Exploration of BERT for Song Classification and Recommendation

Markelle Kelly
Kai Malloy
Mathias Moelgaard
Rohan Mannem
André Rösti

February 2021

Abstract

Music streaming services depend on machine learning for essential tasks such as the classification and recommendation of songs. In this paper, we explore multiple approaches to labeling songs with their sentiments, based on lyrical content and musical features. First, we form a dataset of song metadata, lyrics, and audio features by scraping from Genius, Last.fm, and Spotify. We then compare three styles of BERT models for classification, including one supplemented with topic models, aiming to predict the emotion of each song as extracted from Last.fm tags. Finally, we demonstrate the utility of BERT's hidden states as features in the context of information retrieval. In particular, we reuse our trained model for the application of similarity-based clustering and song recommendation.

1 Introduction

An immensely large music library is available to listeners today, with popular streaming services such as Spotify enabling access to as many as 70 million tracks as of February 2021 (Spotify AB, 2021a). Music information retrieval *(MIR)* involves clustering and extracting information from songs, enabling personalized recommendations for users. Further refining this process will lead to better recommendations and a better overall consumer experience.

Two primary ways to describe a song are the text of its lyrics and characteristics of the audio itself, such as the tempo and tonal scale. Typically, these two types of information are collected separately, but for this project, we combine both data sources. This enables us to predict the mood of songs, in this case simply *happy* or *sad*, using both the lyrical content and musical features.

To predict the songs' emotions, we train and compare three different BERT models. Our first and simplest model is BERT out-of-the-box. We then explore fine-tuning BERT for a more customized model. Lastly, we supplement a fine-tuned BERT model with the output of an LDA model and musical data obtained from Spotify.

After finalizing our model, we apply it to the additional task of music recommendation. We compare two approaches: clustering with BERT hidden layers and a retrieval model based on lyrics. We also combine these concepts to further improve our results.

2 Background and Related Work

Fu et al. (2011) provide an extensive overview of audio-based song classification, pointing to K-Nearest Neighbors and neural networks as popular methods. On the other hand, the utility of lyrical analysis to classify and cluster songs is confirmed by papers such as Fell and Sporleder (2014). However, Laurier, Grivolla, and Herrera (2008) show that combining lyrics and audio features results in overall performance improvements for sentiment classification. Our project aims to affirm the superiority of this multimodal approach, further validating their conclusion.

A popular model for a variety of NLP tasks is BERT (Bidirectional Encoder Representations from Transformers), which achieves state-of-the-art results on many benchmarks (Devlin et al., 2018). After tokenizing and transforming raw text input, the formed word embeddings are passed through attention-based encoders, or Transformers (Vaswani et al., 2017). Each of these encoders uses an attention model which identifies, for each word, which words in the input are contextually relevant. In particular, the BERT Transformers use multi-head attention, so there are multiple attentions generated for each word. BERT attention is also trained bidirectionally, which means both the left and right context of a word are considered jointly. Thus, this process converts a sentence to a sequence of vectors that encapsulate the full context of each word, with multiple "perspectives." After a skip connection and some layer normalization, this output is passed through standard feedforward layers, completing the Transformer. BERT itself is comprised of six of these encoders stacked on top of each other. For more details on the architecture of Transformers and BERT, we refer to their original papers.

Peinelt, Nguyen, and Liakata (2020) introduce an adaptation of the BERT architecture using topic models for semantic similarity detection, tBERT. A pair of documents S_1 and S_2 is passed through a standard BERT model for semantic similarity, as well as a topic model. Then, the [CLS] output of BERT and the topic distributions W_1 and W_2 corresponding to each document are passed through an additional hidden and softmax layer. In effect, this supplements BERT's output with information about the topic distribution of the given documents. Even though tBERT is designed for semantic similarity detection, it makes sense that the topic of a song might be related to its mood, so we consider a modification of this architecture for our classification task.

The use of BERT models for information retrieval has been explored by Yilmaz et al. (2019) as a reranker for improving query results. Embeddings from BERT's hidden layers have also been used for information retrieval with classifiers such as Convolutional Neural Networks (Jeong et al., 2019) and graph based clustering (Das et al., 2020). Finally, Angelov (2020) perform topic modeling by clustering BERT embeddings; we mimic this approach for our song recommendations.

3 Data

There are not any publicly available datasets that contain all the audio and lyrical information required for this project. This shortage is largely due to copyright concerns. Instead, we scrape and combine information from several online sources to form our own dataset.

3.1 Data Collection

Our starting point for data collection is the Wasabi song corpus, which contains 1.7 million songs (Fell, Cabrio, et al., 2019). The dataset includes song titles and artists, along with unique Spotify and Last.fm IDs, which link songs to

the respective online platforms.

Last.fm is our source of "ground-truth": through this socially-based online service, registered users can apply descriptive tags to songs. Accessible through an online API, this gives access to a crowd-sourced directory of song labels, albeit of varying quality. In addition to genre-based information, these labels also describe song sentiment, such as *happy* or *sad*.

Spotify provides audio features for songs through their public API. We refrain from obtaining the actual waveform data of songs, and instead focus on the audio features provided by Spotify (Spotify AB, 2021b). The following features are made publicly available:

- acousticness
- danceability
- energy
- instrumentalness
- key
- liveness
- loudness
- mode
- speechiness
- tempo
- time_signature
- valence

Lastly, we scrape the Genius website for lyrical content of songs. No Genius ID is provided in the Wasabi dataset; instead, we perform a fuzzy search by track title and artist, purging any songs from our data set that do not match closely.

3.2 Data Preprocessing

While the original Wasabi dataset lists 1.7 million songs, it contains missing values and, for our application, some redundancies. Removing all songs that do not have a Spotify and Last.fm ID, and restricting ourselves to songs in English, we are left with only 37,707 rows. Furthermore, a song is represented once for each album it appears in. After removing duplicate songs that are in multiple albums, 24,280 songs remain.

Next, we eliminate songs that did not have any relevant Last.fm tags. Since we aim to classify mood, we are only interested in tags with emotion words, such as *happy*, but not genres, such as *rock*. When restricted to songs that have at least one such emotion tag, our dataset is reduced to 11,207 rows.

Lastly, for a small number of songs, we were unable to retrieve lyrics off of the Genius website. Finally, we arrive at a data set of 11,183 songs.

Labeling Songs The Last.FM tags, which we use as our ground truth for training, are crowd-sourced by an online community, and hence of varying quality. To mitigate this, we remove any songs that have less than 11 tags from our data set, and any tags that have been applied by less than three users. We then classify songs into a set of tags by the following criteria:

Classify a song as its top tag from the set of tags,

1. if that tag is in the song's k most commonly used tags, with $k = 5$,
2. the tag dominates the tags from the other defined sets, i.e. $\frac{\text{tag}_1 - \text{tag}_2}{\text{count}} \leq \epsilon$, with $\epsilon = 0.1$

We chose k and ϵ empirically, using values that kept the dataset sufficiently large without sacrificing label quality. We also experiment with several different groups of synonym sets, but for the purposes of this project, choose to use only two. Our final sets are all synonyms of *happy* (happiness, cheerful, joyful) and *sad* (melancholy, melancholic, sadness, depressing) in the top 100 emotion tags. Note that in these calculations, synonyms can be swapped out, but they are not accumulated, as this would lead to a bias towards songs with many tagged synonyms or moods with more synonyms.

4 Model Architectures

We explore three variations of BERT for classifying the songs in our dataset.

Simple BERT Our first and simplest model uses BERT directly out-of-the-box. Since BERT is pretrained, meaning the weights have already been trained ahead of time on a large corpus, there is no need to train the entire model on our dataset. Instead, we can simply pass our input lyrics through BERT, use the final layer's output as our features, and train an MLP classifier on this output. In effect, this is simply training an additional layer at the end of our network, without backpropagating through BERT. This architecture is shown in figure A.1.

Fine-Tuned BERT As a comparison, our second model fine-tunes BERT itself. This means that we start off with the predefined weights, then tweak them by training with a very small learning rate. After fine-tuning BERT, we can use the final layer's output directly as our prediction, so there is no need to stack on any additional layers. This architecture is shown in figure A.2.

Modified tBERT Our final and most complex model is an adaption of tBERT. We modify the architecture described by Peinelt, Nguyen, and Liakata (2020) for the task of per-document classification. To accomplish this, we replace their flavor of BERT with a BERT model fine-tuned on our dataset and add only one topic model distribution to the final layers. Since we are already supplementing the BERT output, we also add our audio features from Spotify here. We note that adding the topic model features and the audio features individually improved accuracy, but for the sake of brevity we only discuss the full model with both.

For this model, we use the output of LDA, which is a simple bag-of-words style topic model (Blei, Ng, and Jordan, 2003). LDA defines a set of topics from a corpus, which are distributions of words that tend to appear together and are thus likely related. Then, each document is described as a distribution of these topics, based on word counts. After preprocessing the text and running a hyperparameter search, we formed our 7-topic LDA model, which achieved a perplexity score of -6.72 and a coherence score of 0.35.

We then combine the LDA output, which is a distribution over 7 topics, the audio features from Spotify, and the output of a pretrained BERT model into our final hidden and softmax layer. As in our simplest model, this means we train an MLP classifier separately using our output features. This architecture is shown in figure A.3.

5 Results

In general, as our models become more complex, accuracy, precision, and recall improve, while training times get longer. Our exact results are shown in table 1.

	Accuracy	Precision	Recall	Time
Simple BERT	72.7	77.1	81.5	**0:57**
Fine-tuned BERT	73.9	77.6	83.2	6:02
Modified tBERT	**76.4**	**79.2**	**85.6**	8:56

Table 1: Model Results Comparison

While the adapted tBERT model performs the best, it is not by as dramatic of a margin as we expected. However, the original paper also shows mixed results, depending on the dataset used, so it is possible that adding topic models is not especially useful in this domain. This is corroborated by the fact that our LDA model only has a coherence of 0.35, which is quite low. Overall, however, incorporating topics, as well as audio features, does improve model accuracy, so we choose this as our final model.

For this final model, we test a few different training algorithms. For the initial three models, we use PyTorch's AdamW optimizer, which is often recommended as a default algorithm. Here, we compare this to Adagrad, a predecessor of Adam methods, and standard stochastic gradient descent. For each of these methods, we use five epochs, small batches, and hyperparameter values based on a grid search. A comparison of the three methods' descents is shown in figure 1.

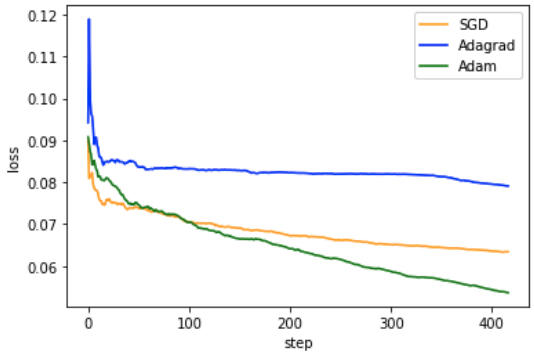

Figure 1: Loss vs. Steps for SGD, Adagrad, and AdamW Algorithms

Overall, the AdamW algorithm performs best, demonstrating the most steady and consistent descent. Even across hyperparameter tuning, the Adagrad algorithm appears to get stuck early on. Adagrad, which updates learning rates

on a per-parameter basis, is often used for problems where gradients can be sparse, such as NLP problems, but it does not do well for our training purposes. While stochastic gradient descent is useful, we choose AdamW as the ideal training algorithm for this problem.

6 Application: Song Recommendation

In this section, we demonstrate a practical application of our trained classifiers. BERT, which is dependent on the concept of transfer learning, is known to be useful for multi-task learning. In particular, BERT can be trained for a specific "proxy" task, then used on a different, related task. While our model is successful at song mood classification, we also explore the utility of its hidden representations in the context of song recommendation. We investigate two main approaches: clustering and a retrieval model based on an inverted index with a ranking system.

Data We use information from the second layer of our final model, Spotify audio features, and lyrics from Genius as input for clustering and the retrieval model. We normalize the BERT embeddings and Spotify audio features, and transform the discrete data using one-hot encoding. This prevents biased weighting for any specific feature data.

Clustering Following the approach of Angelov (2020), where clustering with BERT embeddings was used for topic modeling, we cluster to find groups of related songs. To reduce the high dimensionality of our BERT embeddings, making clustering more feasible, we shrink the input space to 10 dimensions with UMAP, an algorithm that forms low dimensional embeddings while maintaining high level structure (McInnes, Healy, and Melville, 2020).

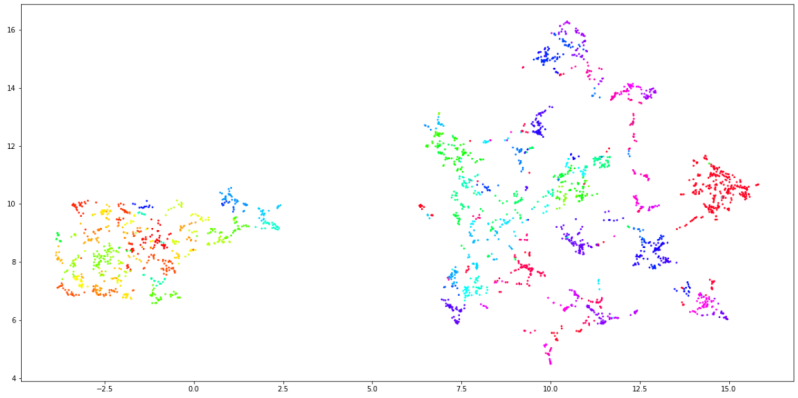

Figure 2: K-Means Clustering of BERT Embeddings Reduced to 2 Dimensions

With the reduced dimensions, we use K-Means to cluster the data as shown in figure 2. We observe two general regions that correspond to the happy and sad classifications, each of which has its own dense spots that can be further split into clusters.

We use K-Means clustering, which allows us to specify the number of clusters to find, while clustering algorithms such as DBSCAN and HDBSCAN have difficulty splitting the data into more than 2 clusters. We then use the number of distortions and the silhouette coefficient to determine the optimal number of clusters. The number of distortions is how far each point in a cluster is from the center point while the silhouette coefficient is how far each cluster center is from the neighboring cluster centers. By minimizing the number of distortions, maximizing the silhouette coefficient, and finding a relatively large number of clusters to split into many groups, we find an optimal cluster count of 122 as shown in figure A.4.

Information Retrieval System Next we build on the BERT embeddings, implementing an information retrieval model with song lyrics as its base. Keywords are extracted through TF-IDF and a distilBERT to retrieve a set of candidate songs. When retrieving a set of recommended songs from a single song input, the scores are ranked through TF-IDF and then reranked using the modified tBERT models final layer of embeddings. The architecture of this system is shown in figure 3.

The reranking of BERT embeddings is paired with cosine similarity between songs and optimized through golden section optimization. We use a categorical entropy loss function, aimed at maximizing the model's precision. The results of this optimization are shown in figure A.5.

After incorporating BERT into an information retrieval model, we see a clear increase in performance, demonstrating that BERT's hidden layers are capturing meaningful information. In fact, our optimized BERT implementation resulted in an increase of six percent precision over the vanilla model. These results are shown in figure 4.

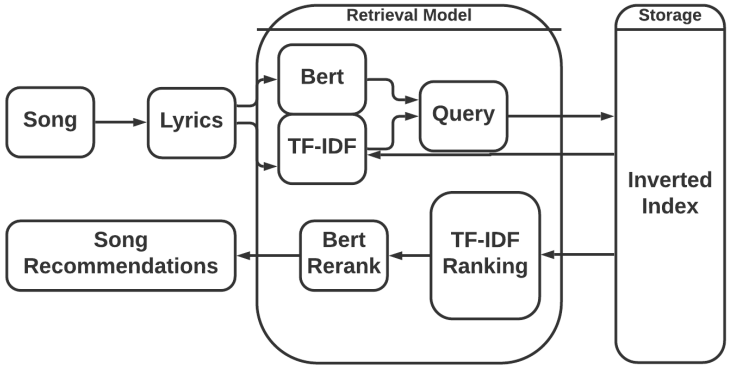

Figure 3: Architecture of Information Retrieval Model

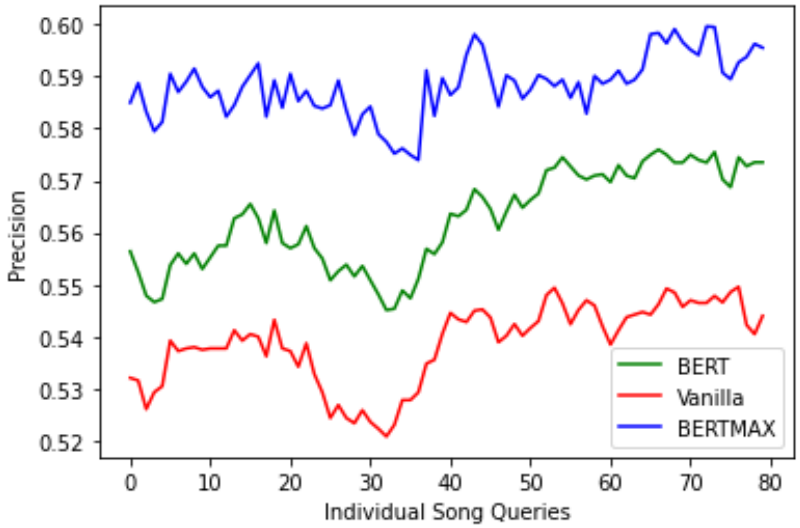

Figure 4: IR Precision across Models

7 Discussion and Conclusion

While our models are overall successful, there are several future directions that should be explored to improve performance. First, after data preprocessing, we have a dataset that is relatively small and imperfectly labeled. If we were to deploy our model, it would be worthwhile to first invest in a large, high-quality dataset to eliminate any problems caused by an insufficient sample size or noisy labels. Second, our LDA approach is largely unable to extract meaningful topics from our song lyrics. One area for future work would be to experiment with other types of topic models, such as GSDMM, which relies on clustering. Lastly, we would like to compare our final model, which combines audio and lyrical features somewhat haphazardly, to a multimodal system where they are more carefully integrated, as done by Laurier, Grivolla, and Herrera (2008).

In this paper, we demonstrate that fine-tuning BERT and supplementing its output with topic model distributions and audio features can improve performance for song mood classification. Further, we show that once trained, this model can be reused in the context of information retrieval to enhance song recommendations.

Member Contributions

All team members were involved in formulating our problem. Everyone also contributed significantly to the presentation and final report.

Markelle Markelle implemented, trained, and compared the three variants of BERT explored. First, she wrote PyTorch programs for each model, adapting the architecture from tBERT for the final model. After preparing the input, and developing an LDA model, she optimized each of the BERT models. She then computed evaluation metrics for each of these models and created convergence plots for the training algorithms considered.

Kai Kai contributed to the song recommendation part of the project. First, he worked on processing the BERT embedding and Spotify audio feature data through normalizing and one-hot encoding. Then, he explored clustering the data with various clustering algorithms. Kai also contributed to working on parts of the retrieval model.

Mathias Mathias created the skeleton and architecture for our information retrieval model as well as the inverted index. He then did keyword extractions from lyrics using distilBERT and TF-IDF scores to pull songs from the inverted index and the TF-IDF and BERT reranking algorithms using clusters and embeddings from BERT. He input our hyperparameter optimization algorithm based on golden search and loss functions to improve performance.

Rohan Rohan created a base k-classification neural network which can classify songs into one of four mood categories. The model was built using a KerasClassifier, a Relu activation function, a logistic loss function, and an Adam gradient descent algorithm.

André André researched the various data sources. Upon decision to create our own data set, he implemented the data scraping scripts using the Spotify, Last.FM and Genius APIs. Lastly, he preprocessed the data. He contributed the introduction, background, related work and data scraping sections of the report and designed the diagrams for the different models.

References

Angelov, Dimo (2020). *Top2Vec: Distributed Representations of Topics*. arXiv: 2008.09470 [cs.CL].

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (Mar. 2003). "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3, pp. 993–1022. ISSN: 1532-4435.

Das, Debasmita et al. (2020). "Information Retrieval and Extraction on COVID-19 Clinical Articles Using Graph Community Detection and Bio-BERT Embeddings". In: *1st Workshop on NLP for COVID-19 at ACL 2020*. ACL.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Fell, Michael, Elena Cabrio, et al. (2019). "Love me, love me, say (and write!) that you love me: Enriching the WASABI song corpus with lyrics annotations". In: *arXiv preprint arXiv:1912.02477*.

Fell, Michael 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*, pp. 620–631.

Fu, Zhouyu et al. (2011). "A survey of audio-based music classification and annotation". In: *IEEE transactions on multimedia* 13.2, pp. 303–319.

Jeong, Chanwoo et al. (2019). "A Context-Aware Citation Recommendation Model with BERT and Graph Convolutional Networks". In: *CoRR* abs/1903.06464. arXiv: 1903.06464. URL: http://arxiv.org/abs/1903.06464.

Laurier, Cyril, Jens Grivolla, and Perfecto Herrera (2008). "Multimodal music mood classification using audio and lyrics". In: *2008 Seventh International Conference on Machine Learning and Applications*. IEEE, pp. 688–693.

McInnes, Leland, John Healy, and James Melville (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv: 1802.03426 [stat.ML].

Peinelt, Nicole, Dong Nguyen, and Maria Liakata (2020). "tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection". In: *58th Annual Meeting of the Association for Computational Linguistics*. ACL, pp. 7047–7055.

Spotify AB (2021a). *Spotify – Company Info*. https://newsroom.spotify.com/company-info/. [Online; accessed 27-February-2021].

— (2021b). *Web API Reference*. https://developer.spotify.com/documentation/web-api/reference/#object-audiofeaturesobject. [Online; accessed 27-February-2021].

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *NIPS*.

Yilmaz, Zeynep Akkalyoncu et al. (2019). "Applying BERT to Document Retrieval with Birch". In: *EMNLP/IJCNLP*.

A Appendix

Figure A.1: Simple BERT Architecture

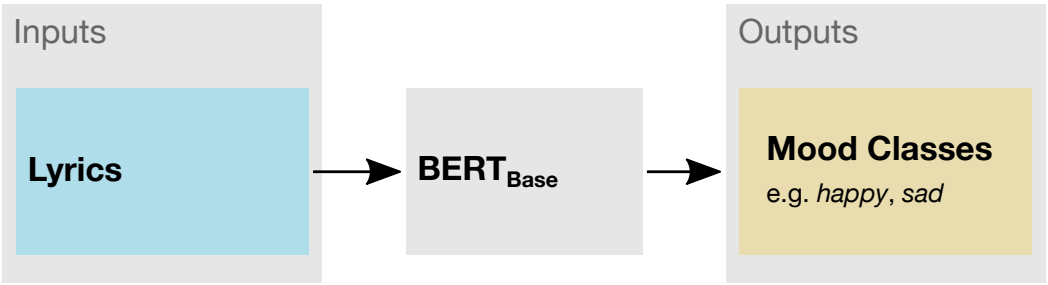

Figure A.2: Fine-tuned BERT Architecture

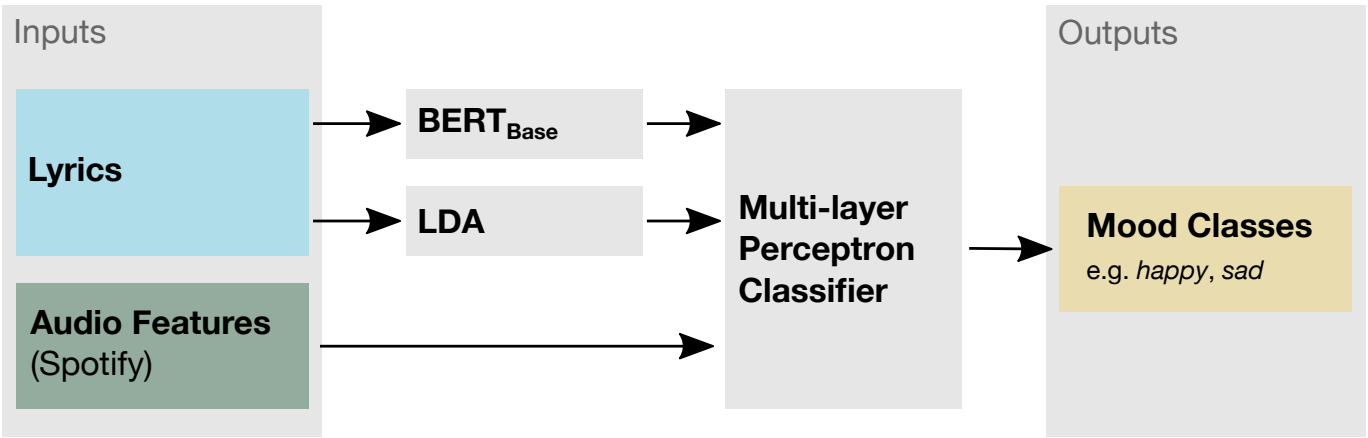

Figure A.3: Fine-tuned BERT + LDA + Spotify Architecture

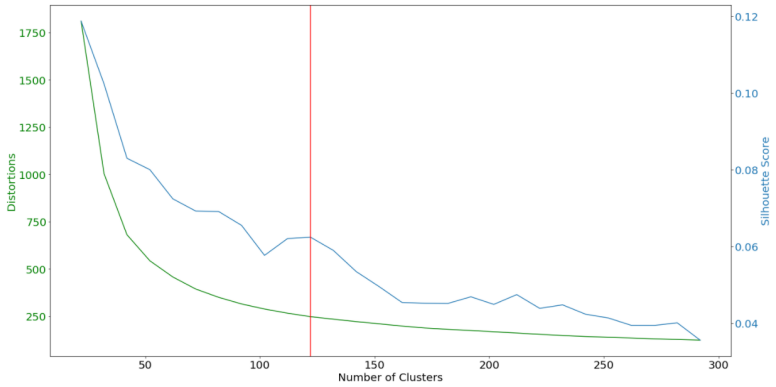

Figure A.4: Distortions & Silhouette Coefficient vs. Number of Clusters

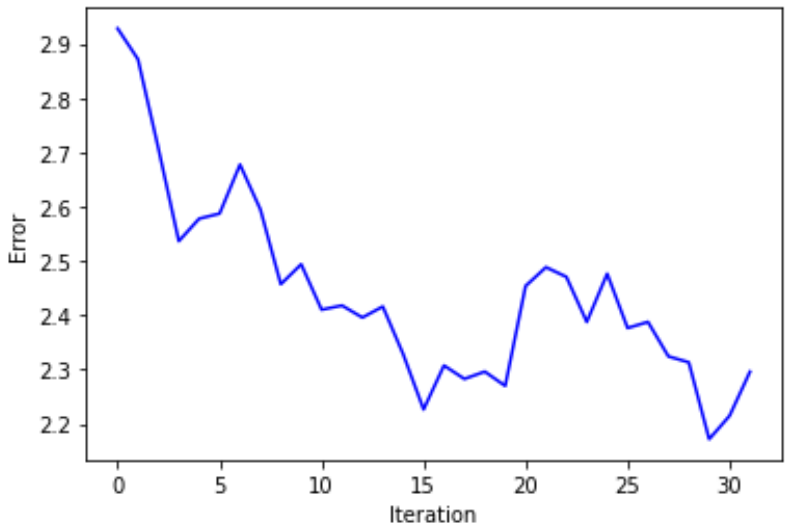

Figure A.5: Hyperparameter Optimization on Retrieval Model