LyricsBERT: Musical Recommendation on a Transformer-based Neural Embedding Backbone

Dong Young Kim
MIT
77 Massachusetts Ave.
dyk0518@mit.edu

Joey Zheng
MIT
77 Massachusetts Ave.
joeyz@mit.edu

Kevin Wen
MIT
77 Massachusetts Ave.
kwen1@mit.edu

Ronald Xu
MIT
77 Massachusetts Ave.
ronaldxu@mit.edu

Abstract

We introduce LyricsBERT, a transformer-based music recommendation system that emphasizes lyrics in hopes of enhancing user experiences on popular music platforms. Through fine-tuning on Masked Language Modeling (MLM) task, LyricsBERT produces representative song embeddings from lyrics. Experiments reveal that for the original DistilBERT model, higher learning rates diminish embedding quality, while a novel architecture with an added fully connected layer performs optimally at a learning rate of 5e-5. Attention maps highlight an improved attention distribution with the fully connected layer, enhancing semantic representation. A rigorous attempt at objective evaluation of song recommendations demonstrates LyricsBERT's superior recommendation quality compared to the unmodified DistilBERT, emphasizing the impact of architectural modifications. The recommendation system employs K-Means clustering for efficient vector similarity measures, contributing. Results indicate that the model excels in capturing diverse semantic relationships within lyrics, showcasing its potential for fairer and more contextually aware music suggestions.

Introduction

In the past few decades, the music industry has seen one of its most prolific eras with novel technologies that have accelerated music production and distribution. However, the way listeners mainly encounter and consume new pieces of music has raised concerns about fairness (Dinnissen and Bauer, 2022). Song recommendation algorithms employed by YouTube, Spotify, and other music streaming services often incorporate artist popularity and the users' social connections, unfairly overemphasizing the recommendation likelihood of mainstream music (Airoldi et al., 2016). These algorithms act as a perpetual feedback loop, reinforcing existing preferences and limiting user exposure to other music. While it is good to give users a custom listening experience, a music recommendation should also avoid making biased recommendations against less popular artists. A recommendation system that does not bias its decisions against less popular artists provides an opportunity for the listeners to discover their new favorite artists more easily, and it also provides a more equal playing ground for the smaller artists in a highly competitive music industry. To do so, we would like to develop a music recommendation model whose primary measure of decisions is based on the quality of the music rather than the artist's popularity and social connections. Therefore, in this project, we propose a model that gives song recommendation that focuses on features that are inherent to a song: its lyrics.

Related Work

Many works in the academic field have focused on developing music recommendation algorithms that incorporate non-musical features. For example, Gatta et al. proposed Hypergraph Embeddings for Music Recommendation (HEMR), which incorporates all the possible and complex interactions between users and songs with related characteristics in their recommendation algorithm (Gatta et al., 2023). Similarly, Wang et al. proposed Content- and Context-Aware Music Embedding (CAME), which obtains behavior information, including users' listening behaviors, music-playing sequences, and sessions, to generate music recommendations (Wang et al., 2020). While these techniques may help maximize the sequential listening/viewing time by focusing on analyzing the users' listening behaviors and past sessions, they fail to incorporate the inherent musical features of the songs in their recommendations and undermine less mainstream music. Hence, there is a need for a song recommendation model that focuses on the features of a piece of music itself, not the users' listening behaviors, to generate its recommendations, which provides fairer opportunities for smaller artists and a higher quality musical experience for the listeners.

Among different musical features, such as genre, beats, etc., this project focuses on generating song recommendations based on lyrics. The attempt to generate music recommendations based on lyrics, however, is not new. These attempts have often tried to extract the semantics and emotions from the lyrics, which were then used to make recommendations to other songs with similar features. In 2021, there was a study that extracted contextual embeddings from the lyrics with Google's Universal Sentence Encoder and used semantic similarity to generate song recommendations. This proposed model achieved an F1 score of 0.7700, which was higher than other lyrics-based song recommendation models at that time (Gupta et al., 2021). Moreover, Revathy et al. proposed LyEmoBERT, which uses emotional classification (happy, angry, relaxed, and sad) of lyrics and the Sentence Transformer model to generate song recommendations (Revathy et al., 2023). However, Gupta et al., when evaluating the quality of their recommendations, relied on search engine recommendations as ground truth, which may be biased towards using popularity metrics to generate results to queries. Moreover, Revathy et al. relied solely on the emotional labels predicted for lyrics as the primary criteria for song recommendations, neglecting to consider the intrinsic content of the lyrics.

While these works have made significant advancements in analyzing lyrics for generating music recommendations, we believe there are two main ways in which our work can improve on these past works.

First, we aim to incorporate more recent encoding techniques to generate richer semantic and syntactical embeddings. To do this, we will add further improvements to the existing DistilBERT for the encoding mechanism (Sanh et al., 2019) with additional. This way, we can allow the model to fine-tune and learn a better way to generate embeddings for our project's more specific domain: song recommendation through lyrical analysis. Second, we will conduct more extensive hyperparameter searches of an existing high-performing encoder. This includes, but is not limited to, testing different optimizers during the embedding model training. Furthermore, in the recommendation model in the final layer, we can test different hyperparameters such as sizes of the k-neighbors and randomization factors for picking the song within the clusters.

Past work has also been done in the evaluation of recommendation systems.

Firstly, a simple yet popular method is to manually annotate how similar a query song is to the recommended songs. This could entail labels by experts or surveys from laymen. Gupta et al. utilized manual labor to compare queries and recommended lyrics on testing sets of various sizes(Gupta et al., 2021). Using these annotations, we can calculate accuracy, precision, recall, and F1-score as metrics of evaluation for a recommendation system. The benefit of this evaluation method is its simplicity and straightforwardness in calculation. However, manual annotations require substantial resources in labor and time, which is unsuitable for the timeline of our project. Additionally, human biases and error pose a large issue as external factors such as the mood or background of the annotator could affect their answer and the task is, to a certain degree, subjective in nature.

Another existing evaluation method is to compare with existing song recommendation systems. In particular, given a query, comparing the output lyrics of the existing recommendation systems with that of ours. However, it is problematic if we treat existing systems as ground truth since, as mentioned before, existing systems introduce biases by generating recommendations based largely on users' behavior and activities. At best, we can conclude if our recommendations are reasonable according to industry standards. Thus, this evaluation method does not align with our focus on the intrinsic properties of the lyrics of songs.

Methodology

The overall method is to train and fine-tune embedding models on downstream Natural Language Processing (NLP) tasks. In brief, we utilize transformer-based embedding models similar to Distilbert, as well as an added fully connected layer (Sanh et al., 2019). We then train them on the downstream tasks of Masked Language Modeling (MLM) (Salazar et al., 2019). Finally, we integrate the embedding model into a recommendation system and evaluate its abilities.

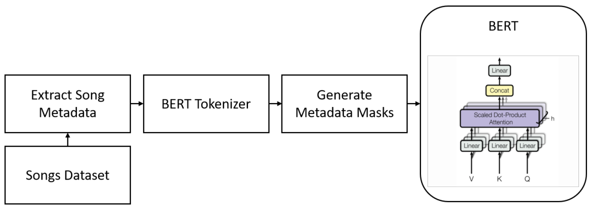

Figure 1: LyricsBERT Training Method

Dataset and Preprocessing

We utilize two datasets from Kaggle (Song Dataset, SHRIRANG MAHAJAN) (Song Dataset, EDENBD). Combined, they have more than 200,000 songs. Due to limitations in computation, we randomly selected 12,000 songs as our raw dataset. We partition this dataset 70/15/15 for training, validation, and testing, respectively.

The key downstream tasks we will train on are Masked Language Modeling (MLM). To generate training data for MLM, we will modify 15% of the tokens in a data point. 80% of these tokens are replaced with the [MASK] tokens. Then, 10% are corrupted, where the tokens are replaced by another random token. The remaining 10% of the tokens remain the same. The ground truth will simply be the original text.

Embedding Models

BERT is the cornerstone of our embedding model. It is extremely capable for its ability to capture semantic and syntactical information from sequences. Moreover, we introduce a fully connected layer for LyricsBERT. Unlike the natural English language that BERT is trained in, the lyrics of songs inherit more specific features and structures. By adding a fully connected layer that is then trained on the downstream tasks for our specific domain of musical lyrics, LyricsBERT's embedding models capture key features that are inherent to musical lyrics.

The training method for LyricsBERT is shown in Figure 1. We extract the metadata from songs, tokenize them, generate augmented input for model training, and then send the masked inputs to a BERT architecture to produce embeddings.

Downstream Tasks

Once the embedding model has processed the input tokens, these embeddings are fed to a downstream task.

The main downstream task we used was MLM. This requires the model to predict what the original words of the chosen masked, corrupted, and protected tokens were. Although the original BERT is trained on the MLM task itself, we introduced MLM as our downstream task because, with our specific focus on musical lyrics, LyricsBERT may learn more nuanced structures and patterns that are specific to the musical lyrics. The loss criterion used is Cross-Entropy Loss (Zhang and Sabuncu, 2018), and this is used for backpropagation through the model.

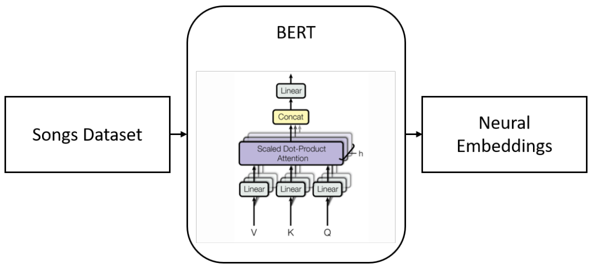

Figure 2: Preprocessing for LyricsBERT Inference

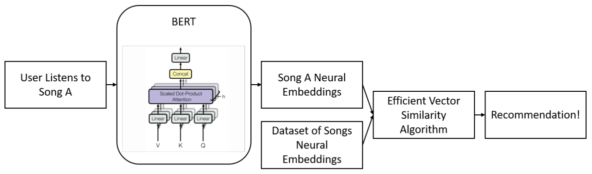
User Listens to Song A
BERT
Song A Neural Embeddings
Dataset of Songs Neural Embeddings
Efficient Vector Similarity Algorithm
Recommendation!

Figure 3: LyricsBERT Inference

Recommendation System

To have a fast recommendation system, we will need to preprocess the song dataset.

We pass the songs' lyrics through LyricsBERT. We save the embedding representations in a data structure so we don't have to reproduce them each time. This process is displayed in Figure 2.

Finally, for inference, we follow the algorithm in Figure 3. When a user listens to or interacts with a song, we generate embeddings for it using the song's metadata and then compare these embeddings with the embeddings found in our dataset of songs. We then use an efficient vector similarity measure to produce some relevant songs. The main technique we use for finding similar vectors in our embedding space for songs is K-Means clustering (Makwana et al., 2013) followed by random sampling from the cluster that is selected. K-Means clustering allows us to decrease the search space by eliminating the need for having to go through every song in our dataset linearly to generate the recommendation. Moreover, it introduces random, not deterministic, sampling, which is a desired aspect of a music recommendation system.

Evaluation

We will evaluate the recommendation system by comparing the cosine similarity between the embeddings of songs returned. We utilize the key idea that a song should be similar to itself. With two songs A and B that are not similar, if we pass the lyrics of both songs into our recommendation system, we get the corresponding recommendations for A and B, calling them A_r and B_r, respectively. In this case, we would hope that the songs returned by the recommendation system are different. Now, assume that we cut A in halves to arrive at A_1 and A_2. If we pass the lyrics of A_1 into the recommendation system, we get out a song A_{1r}. Similarly, we pass the lyrics of A_2 into the recommendation system and get song A_{2r}. We compare the cosine similarity between the embeddings of A_r and B_r, specifically $e(A_r)$ and $e(B_r)$ as well as the cosine similarity between the embedding of A_{1r} and A_{2r}, namely $e(A_{1r})$ and $e(A_{2r})$. Because the former lyrics, A and B, were from differing songs, it makes sense that they should have a smaller cosine similarity to the latter, A_1 and A_2. Based on this observation, we compute these scores over our test dataset and find the average difference between cosine similarities. Hence, in our model, we desire the average cosine similarity between recommended songs generated from the same song to be greater than the average cosine similarity between recommended songs generated from different songs. Thus, as a point of evaluation, we try to maximize the score in the formula below:

$$\textbf{Score} = |e(A_r) - e(B_r)| - |e(A_{1r}) - e(A_{2r})|$$

We believe that this method of evaluation reduces human bias and is efficient while preserving the intrinsic properties of a song's lyrics—satisfying the goals of our system.

Experimental Results and Discussion

Masked Language Modeling

To fine-tune our embedding model for downstream tasks, we experimented with different learning rates and architectures in the Masked Language Modeling (MLM) task. The goal was to assess the effect of these variations on the quality of our recommendation system and hence the embeddings generated by the model, as quantified by the evaluation metric discussed in the previous section. We compare all performances to that of an unmodified, un-fine-tuned DistilBert model using our custom evaluation metric. A comparison of all of model performances can be seen in Figure 4, where we interpret a higher evaluation score to signify higher quality recommendations.

We conducted experiments with different learning rates to understand their effect on the performances of our different architectures. For the unmodified DistilBERT model, we observed that using a higher learning rate during fine-tuning led to a deterioration in embedding quality. This phenomenon can be attributed to the model being already close to a local optimum, and further training at higher rates results in bouncing around without meaningful improvement. In contrast, when introducing an additional fully connected layer to the architecture, we found that a learning rate of 5e-5 yielded the best performance. However, as we increased the learning rate beyond this point, the performance of the modified architecture started to degrade. This drop in performance may be attributed to the risk of overshooting optimal parameter values. It is important to note that at a learning rate of 5e-5, there is still room for convergence as the newly added fully connected layer introduces additional parameters.

Now, in our investigation of architectural variations, we found that the model with an added fully connected layer showed the most promising results. Fine-tuning this architecture at a learning rate of 5e-5 yielded the highest, suggesting that incorporating task-specific modifications can enhance the model's ability to generate high-quality embeddings. However, it is crucial to strike a balance in model complexity, as overly large architectures may hinder convergence and result in sub-optimal performance.

Overall, these experiments provide insights into the sensitivity of our embedding model to learning rates and architectural modifications. The findings guide our choices in fine-tuning strategies for optimal downstream task performance.

Attention Maps

To gain insights into how our model processes and attends to different parts of the input sequence, we analyzed attention maps for both the original DistilBERT architecture and a modified version with an additional fully connected layer. Attention maps provide a visualization of the self-attention

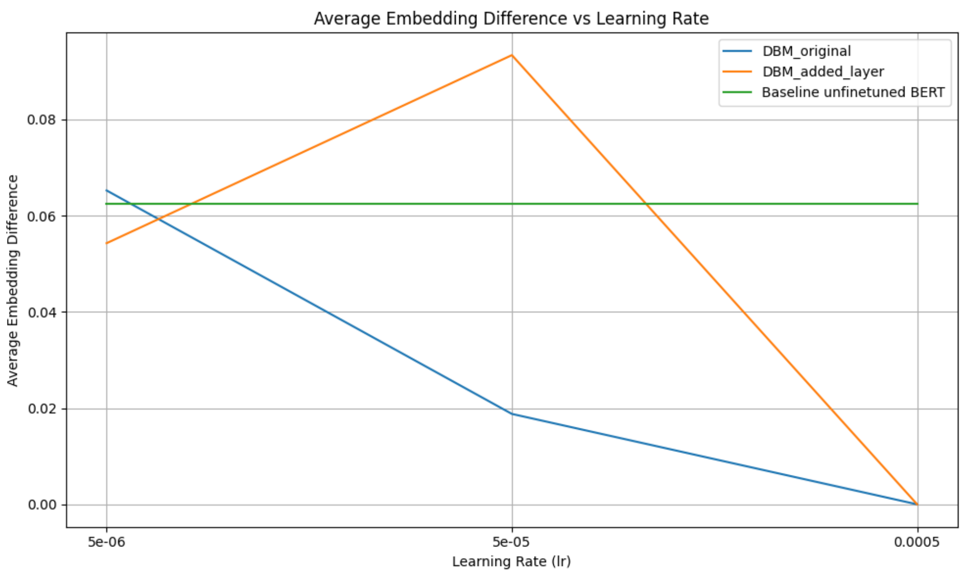

Figure 4: LyricsBERT Evaluation Performance: Average Embeddings Difference Scores are Computed Over Different Learning Rates

mechanism, showcasing which tokens in the input sequence receive more focus during the embedding generation process.

In the unmodified DistilBERT architecture (Figure 5), we observed that certain attention heads displayed a tendency to concentrate heavily on specific tokens, particularly the [SEP] (separator) token. This behavior raised concerns about the potential dominance of non-informative tokens in the attention process, possibly hindering the model's ability to capture meaningful semantic relationships within the sequence.

Upon introducing a fully connected layer to the architecture (Figure 6), we noted a significant improvement in the attention maps. The attention heads exhibited a more uniform distribution of attention across various tokens in the input sequence. Notably, there was a reduction in the pronounced focus on the [SEP] token observed in the unmodified architecture.

This shift towards a more balanced distribution of attention suggests that the model with the added fully connected layer may be capturing better semantic relationships within the sequence. The enhanced attention diversity indicates that the model is not overly fixated on specific tokens and is better equipped to consider the contextual nuances of the input.

In summary, the analysis of attention maps supports the hypothesis that the introduction of a fully connected layer contributes to improved semantic representation in the embeddings. The modified architecture demonstrates a more nuanced and contextually aware attention mechanism, potentially enhancing the model's ability to capture meaningful information from input lyric sequences.

Limitations

While our proposed approach shows promise in addressing the challenges of biased song recommendations, several limitations should be considered.

Limited Diversity in Training Data

The datasets used for training and evaluation are sourced from Kaggle, potentially limiting the diversity of musical genres and styles represented. The model's performance may be biased towards the characteristics of the songs present in the training data, impacting its effectiveness for underrepresented genres. The distribution of musical style of this dataset was not investigated, so that may have contributed to a source of bias in our embeddings and hence our recommendations.

Lyric-Only Focus

Our recommendation system primarily relies on the analysis of lyrics for generating embeddings. While lyrics are essential, neglecting other musical features, such as melody and rhythm, may limit the system's ability to provide holistic recommendations. Incorporating additional audio features could improve the model's capability to capture diverse musical preferences.

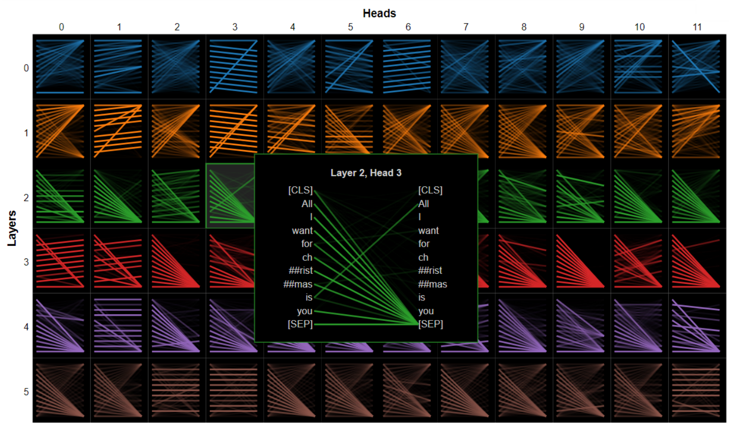

Figure 5: Attention map for unmodified DistilBERT architecture

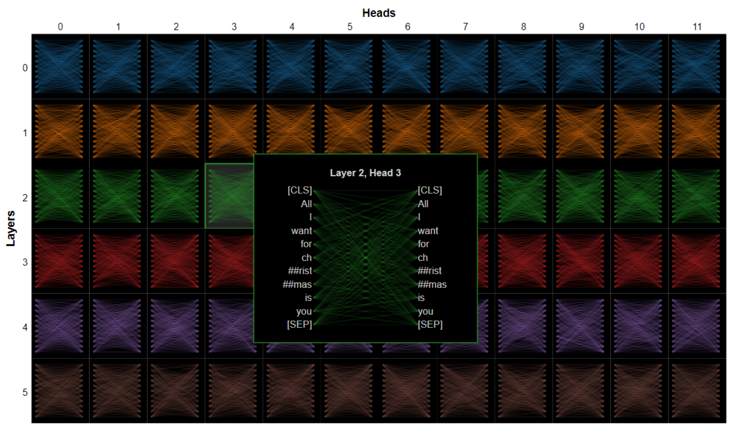

Figure 6: Attention map for modified architecture

Evaluation Metrics and Subjectivity

The evaluation method, based on cosine similarity between embeddings, may not fully capture the subjective nature of music preferences. User-specific factors, emotional context, and personal tastes are challenging to quantify accurately. Different users may have diverse expectations from a recommendation system, and a more user-centric evaluation strategy could provide a more nuanced understanding of the model's effectiveness in future works.

Model Interpretability

While attention maps provide insights into the model's attention mechanism, interpreting these visualizations may be challenging. Understanding the exact semantic meaning behind specific attention patterns requires careful analysis and might be subjective. The interpretability of the model's decisions remains an open challenge.

Scalability and Real-time Recommendations

Efficiency in handling a large number of songs and providing real-time recommendations is not explicitly addressed. Scalability concerns, particularly when dealing with extensive music libraries, could impact the practical applicability of the recommendation system in real-world scenarios.

These limitations highlight areas for future exploration and refinement in our approach to ensure a more comprehensive and robust music recommendation system.

Conclusion and Future Works

In this work, we introduced LyricsBERT, a revised approach to music recommendation focusing on generating song embeddings from lyrics. The transformer-based embedding model demonstrated promising results in capturing semantic relationships within lyrics. The addition of a fully connected layer improved attention mechanisms, contributing to more meaningful embeddings.

Despite these advancements, there are notable areas for future exploration. A more diverse training dataset is needed to enhance generalization, considering various musical genres, languages, and artist profiles. The inclusion of multimodal features, such as melody and rhythm, can provide a more comprehensive understanding of songs. Additionally, refining evaluation metrics to align with user preferences and exploring real-time scalability are crucial for practical applications.

In conclusion, LyricsBERT represents progress towards fairer and contextually aware music recommendations. Ongoing research is essential to address current limitations and evolve towards a more comprehensive and user-centric music recommendation paradigm.

Impact Statement

In our project, we introduced LyricsBERT, a musical recommendation system that recommends songs based on their inherent feature—lyrics. As outlined in the Introduction section, we developed this system with the hope that it would enhance the enrichment of the listeners' experience by focusing on music's inherent quality rather than social or popularity features. Moreover, we believe this recommendation system would help to serve as a tool to reduce the ever-increasing gap between popular artists and not-as-popular artists by avoiding artist popularity and the users' social connections to impact the recommended music.

Furthermore, we believe accessibility and transparency of our project are highly important, so we have conducted our project in a way that it can be reproduced easily. Specifically, we used two publicly available datasets on Kaggle for our study (Song Dataset, SHRIRANG MAHAJAN) (Song Dataset, EDENBD). Moreover, by incorporating open-source libraries and DistilBERT, a distilled version of BERT, in our model architecture, we strategically constructed our project so that the code base could be run and the results could be reproduced with no extensive computing resources. The exact code base of the project is also available to be shared upon request with any of the authors of this paper.

Acknowledgments

We would like to extend our gratitude to Professor Kim, Professor Andreas, and Professor Tanner for their dedicated teaching during the course, their support for this project, and their expertise and passion for natural language processing. We would also like to give a special thanks to Thomas Pickering, our recitation instructor, and our TAs for giving invaluable feedback and guidance in our project deliverables.

References

Massimo Airoldi, Davide Beraldo, and Alessandro Gandini. 2016. Follow the algorithm: An exploratory investigation of music on youtube. *Poetics*, 57:1–13.

Karlijn Dinnissen and Christine Bauer. 2022. Fairness in music recommender systems: A stakeholder-centered mini review. *Frontiers in big Data*, 5:913608.

Valerio La Gatta, Vincenzo Moscato, Mirko Pennone, Marco Postiglione, and Giancarlo Sperlí. 2023. Music recommendation via hypergraph embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7887–7899.

Vidit Gupta, S Jeevaraj, and Somesh Kumar. 2021. Songs recommendation using context-based semantic similarity between lyrics. In *2021 IEEE India Council International Subsections Conference (INDISCON)*, pages 1–6. IEEE.

P Makwana, TM Kodinariya, and PR Makwana. 2013. Review on determining of cluster in k-means clustering review on determining number of cluster in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6):90–95.

V. R Revathy, Anitha S. Pillai, and Fatemah Daneshfar. 2023. Lyemobert: Classification of lyrics' emotion and recommendation using a pre-trained model. *Procedia Computer Science*, 218:1196–1208. International Conference on Machine Learning and Data Engineering.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2019. Pseudolikelihood reranking with masked language models. *CoRR*, abs/1910.14659.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Song Dataset, EDENBD. Https://www.kaggle.com/datasets/edenbd/150k-lyrics-labeled-with-spotify-valence.

Song Dataset, SHRIRANG MAHAJAN. Https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset.

Dongjing Wang, Xin Zhang, Dongjin Yu, Guandong Xu, and Shuiguang Deng. 2020. Came: Content-and context-aware music embedding for recommendation.

IEEE Transactions on Neural Networks and Learning Systems, 32(3):1375–1388.

Zhilu Zhang and Mert R. Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. *arXiv e-prints*, page arXiv:1805.07836.