Universiteit Leiden

Master Computer Science

Music Album Review Rating Prediction Using Transformers

Name:	Arvindeva Wibisono
Student ID:	S3084736
Date:	12/07/2023
Specialisation:	Computer Science: Data Science
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof. Dr. M. S. K. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgement

I would like to extend my gratitude to my thesis supervisor, Dr. E.M. Bakker, for his guidance in assisting the writing process of my master's thesis. I would also like to thank my father Ario Wibisono, my mother Burnelly Putnam, my brother Maztra Perdana, and my sister-in-law Tania Hazzriandiba for their support and encouragement. Last, but certainly-not-least, I would like to extend the biggest gratitude to my girlfriend Jana Rumak for sticking with me through my effort in writing this thesis. It would not have been possible without your support and encouragement.

Thank You.

Abstract

This research paper explores the task of predicting music album ratings based on their corresponding text reviews using machine learning algorithms. Two experiments are conducted to evaluate the performance of Support Vector Machine, Random Forest, Recurrent Neural Network with Long Short-Term Memory, BERT, and Longformer models in review rating prediction tasks. The first experiment is done using a benchmark dataset for review rating prediction. The second experiment is done using a novel dataset, the Pitchfork Review Rating dataset, consisting of review-rating pairs from music album reviews published by the online music publication Pitchfork. The results from the experiments show that Longformer outperformed other algorithms in both experiments. The results also indicate a high degree of correlation between text reviews and their numerical ratings.

Contents

1 Introduction

Countless pieces of music are released every day. The average music listeners only listen to music that they are interested in. They discover new music in a variety of ways, such as by word of mouth, hearing it in a public place, and getting recommendations from applications like Spotify. For many music fans, reviews and ratings can be used to filter out new releases. Music critics affect album sales (and streaming plays), as people are more likely to listen to a particular song or album, if it has a good rating [1]. Critic ratings affect music artists, especially smaller or upcoming artists, as their livelihood can depend on the sales numbers of their musical output. With the availability of music critics on the internet, music reviews are now more common than they used to be, yet there have been limited studies in this area. In this paper, we present research on music album review rating prediction.

Review rating prediction is a Natural Language Processing (NLP) task that aims to predict the rating of a review based on its corresponding textual content. Review rating prediction can be seen as a part of sentiment analysis. Sentiment analysis, also called "opinion mining", is a task that aims to figure out how people feel about something by analyzing and extracting subjective information from text. NLP techniques and machine learning algorithms are used in sentiment analysis to figure out what words, phrases, and the context of a text mean. Review rating prediction comes in a variety of forms, one active research area is customer online review rating prediction, such as for Yelp. Yelp, founded in 2004, is an online directory of business reviews where users can submit reviews of businesses, such as restaurants. A review consists of a text review and a star rating. Yelp review rating prediction is a task with the goal of predicting star ratings (1–5 stars) based on their corresponding text reviews. Yelp is often used as a Benchmark dataset. We use several machine learning algorithms and compare their performances by training them on the Yelp dataset, a dataset published by the Yelp company as part of the Yelp Dataset Challenge. We include traditional machine learning algorithms such as SVM and Random Forest, a deep Recurrent Neural Network with LSTM, and two recent transformer based models, BERT and Longformer.

While there has been research on online customer review rating prediction, there is limited research on music review rating prediction. With the algorithms used in the Yelp experiment, we will retrain them for music review rating prediction purposes. To do so, we train the algorithms on a dataset consisting of music reviews and their corresponding rat-

ings from Pitchfork. Pitchfork, founded in 1995, is an influential music publication and has been described as "the most influential music publication to emerge in the internet age"[2]. They are known for their incisive reviews and ratings. The publication rates albums on a scale from 0.0 to 10.0 with a precision up to a tenth of a point. As of 2021, they have published over 27,000 reviews [3], and all of their reviews are archived online on their website.

Utilizing their archive, we collected more than 18,000 review and rating pairs. This new dataset is then used to train machine learning algorithms, aiming to see how well these algorithms can predict the rating of a music album based on its text review. While this is a sentiment analysis task, there are minor differences from the Yelp experiment. Pitchfork use different scales for numerical ratings. Yelp uses a star rating of 1 star to 5 stars, while Pitchfork uses a numerical rating of 0.0 to 10.0 (up to one decimal point). Another difference is in the length of the text reviews. On Yelp, the review lengths vary greatly as they are user-generated and users are allowed to submit reviews of any length. On Pitchfork, the reviews are written by professional critics and are generally longer than Yelp user reviews. Due to the length of the reviews, we included Longformer in our experiment. Longformer is a transformer-based neural network architecture that scales linearly with sequence length, making it capable of processing review texts of thousands of tokens. This capability makes Longformer a suitable algorithm for the Pitchfork dataset.

In this paper, the following **research questions** will be addressed:

1. To what extent are machine learning models able to predict a music album rating based on its text review?
2. Is there a correlation between an album's text review and its corresponding numerical rating?

Contributions For this study, we constructed a new dataset, the Pitchfork Review Rating dataset. This new dataset is built upon an existing Pitchfork dataset on Kaggle by Conaway [4]. Conaway's dataset provides us with Pitchfork review URLs, which are then used to extract the review's textual content. Furthermore, we present a novel experiment on a music review rating prediction task, using Pitchfork's reviews to evaluate the performance of machine learning algorithms and see if the algorithms are able to predict Pitchfork's ratings. With this, we are able to compare the performances of state-of-the-art transformer-based

models such as BERT and Longformer, fine-tuned on a dataset of longer documents.

Structure The rest of the paper is organized as follows: In Section 2, we discuss previous studies on review rating prediction. In Section 3, we provide background information on the machine learning algorithms used in our experiments, as well as the definitions of the measures we use to evaluate the performance of the algorithms. Section 4 describes the methodology of our experiments, including the setup, hyperparameter tuning, and training process. In Section 5, we describe the Yelp and Pitchfork datasets and provide information regarding data collection, analysis, and preprocessing. Section 6 describes the two experiments—the Yelp and Pitchfork experiments and their results, while also providing notable observations from the results. In Section 7, the results will be discussed, answering our research questions. And finally, in Section 8, the conclusions are given, summarizing this paper and listing improvements that can be made for possible future work.

2 Related Work

This section discusses relevant previous research. There has been extensive research on sentiment analysis and review rating prediction, such as online retail product ratings prediction, the mood of Twitter tweets prediction, movie recommendations, and many more. Nevertheless, research on music review rating prediction has been relatively sparse.

Sentiment analysis is one of the most extensively researched areas of NLP. Early approaches involved rule-based methods and traditional machine learning techniques. Turney [5] introduced the idea of using unsupervised learning techniques to classify sentiment in text reviews. Pang and Lee [6] used support vector machines (SVM) to predict the sentiment of movie reviews. In 2013, Yelp officially released the Yelp Dataset [7]. The Yelp Dataset has since become one of the benchmark datasets for various NLP tasks, including review rating prediction. Other benchmark datasets include the IMDb movie reviews dataset [8] and the Amazon online customer product reviews dataset [9]. The Yelp review rating prediction task aims to predict the star rating of a Yelp review based on its text review. Zhang [10] introduced a character-level convolutional neural network (CNN) and achieved 62% accuracy in predicting the star ratings of Yelp reviews. Zhang showed that character-level convolutional networks are capable of achieving comparable results to traditional models such as bag of words, n-grams, and their TF-IDF variants. Johnson & Zhang [11] proposed the use of LSTM units on top of CNN and achieved state-of-the-art results on four benchmark datasets, including the Yelp Dataset. Johnson & Zhang [12] then proposed another method, deep pyramid CNN, which outperformed their own previous work on benchmark datasets, achieving 69.42% accuracy on the Yelp dataset. In 2018, Howard & Ruder [13] published a paper in which they proposed Universal Language Model Fine-Tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduced techniques that are key for fine-tuning a language. At the time, the method outperformed state-of-the-art methods on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Multiple other studies have been done on Yelp review rating prediction [14][15].

In recent years, the transformer architecture has emerged as a groundbreaking approach to NLP tasks. Vaswani et al. [16] introduced the original transformer model in the seminal paper "Attention Is All You Need", which outperforms previous state-of-the-art models in machine translation tasks at a fraction of the training cost. The transformer model em-

ploys a self-attention mechanism to capture dependencies between tokens, allowing parallel computation for training. Since then, different variants and improvements have been proposed. Devlin et al. [17] introduced Bidirectional Encoder Representations from Transformers (BERT), a large language model that is able to capture both the left and right contexts of a token. BERT is pre-trained using a large amount of text data from the internet on two tasks: masked language modeling and next sentence prediction. In 2019, Sun et al. [18] conducted exhaustive experiments to investigate different fine-tuning methods of BERT on text classification tasks and provide a general solution for BERT fine-tuning that achieves new state-of-the-art results on eight widely studied text classification datasets, including 70.58% accuracy on the Yelp 5-class review rating prediction.

One limitation of BERT is that it can only take input sequences up to 512 tokens in length. To process a sequence of more than 512 tokens, BERT truncates the input, which may lead to information loss. Studies have been done to solve this limitation, and one of them is Longformer, the long-document transformer. Proposed by Beltagy et al. [19] in 2020, The Longformer employs a sliding window attention mechanism. This mechanism is used to handle longer sequences, up to a length of 4096 tokens, by approximating standard self-attention with a computational complexity of $O(n)$, compared to BERT's $O(n^2)$, making it more efficient for long documents. In 2022, Lyu et al. [20] used the Longformer in conjunction with user and product information on the Yelp dataset, achieving state-of-the-art results.

While benchmark datasets have been extensively used for review rating prediction, there is limited research being done on music review rating prediction. Conaway [4] published the Pitchfork dataset on Kaggle, consisting of more than 17,000 Pitchfork reviews metadata. The dataset has since been extended by Pinter et al. [21] as P4KxSpotify, adding Spotify audio features such as genre, energy, and tempo for each album. This paper builds upon Conaway's Pitchfork dataset by iterating through all review URLs and extracting the review textual content.

3 Fundamentals

In this section, we give background information on deep learning and the attention mechanism in transformers, as well as the definitions of the measures we use to evaluate the performance of our proposed methods.

3.1 Evaluation Metrics

Here we introduce and define the metrics that are used to evaluate the performance of the machine learning models present in our experiments.

Accuracy

Classification accuracy is a metric that measures the ratio of correct rating predictions to the total number of predictions. One disadvantage of accuracy is that it does not take into account how close the prediction is to the actual value in the case of a wrong prediction. Equation 1 shows the formula to calculate the accuracy for all classification models used.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{1}$$

Precision & Recall

Precision and recall are evaluation metrics commonly used in machine learning for assessing the performance of classification models. They can provide more insights compared to accuracy to evaluate a model's performance, especially if there is a data imbalance.

Precision measures the proportion of true positive predictions (correctly identified positive instances) out of all positive predictions made by the model. Precision is important in scenarios where false positives are costly or have serious consequences. Equation 2 shows the formula to calculate the precision of a class.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2}$$

Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It assesses the completeness of the model's predictions and indicates how well it avoids false negatives. Recall is important in scenarios where false negatives are costly or could lead to missed opportunities or risks. Equation 3 shows the formula to calculate the recall of a class.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3}$$

F-score

The F-score, also known as the F_1 score, is another metric used to evaluate the performance of a classification model. It combines both precision and recall into a single value, providing a balanced measure of a model's accuracy. It is the harmonic mean of precision and recall and is useful due to its interpretability as a single metric. Equation 4 shows the formula to calculate the F-score of a model's performance.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Mean Absolute Error

Mean Absolute Error (MAE) is a metric to measure the average magnitude of errors between predicted and actual values of a regression (or ordinal classification) model. It provides a numerical value that represents the average absolute difference between the predicted and actual values of a set of observations. Equation 5 shows the formula to calculate the MAE of a model's performance.

$$\text{Mean Absolute Error} = \frac{\sum |\text{Prediction} - \text{Actual}|}{\text{Number of data points}} \tag{5}$$

Mean Squared Error & Root Mean Squared Error

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are two of the most widely used metrics in machine learning to measure regression model performance. They are the average squared difference between predicted and actual values, which is similar to MAE except that all the errors are squared. RMSE is the square root of the MSE. MSE and RMSE penalize bigger errors by squaring the errors and they are continuous and differentiable. Equations 6 and 7 show the formulas to calculate MSE and RMSE, respectively.

$$\text{Mean Squared Error} = \frac{\sum(\text{Prediction} - \text{Actual})^2}{\text{Number of data points}} \tag{6}$$

$$\text{Root Mean Squared Error} = \sqrt{\text{MSE}} \tag{7}$$

Pearson Correlation Coefficient (r)

Pearson's correlation coefficient, or Pearson's r, measures the strength and direction of the linear relationship between two variables. It ranges from -1 to +1, where a value of +1 indicates a perfect positive linear relationship, a value of -1 indicates a perfect negative linear relationship, and a value of 0 indicates no linear relationship. Equation 8 shows the formula to calculate Pearson's correlation coefficient between two variables , where in the context of this experiment, x_i is the predicted value, $\hat{x}$ is the predicted mean, y_i is the actual value, and $\hat{y}$ is the actual mean.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{8}$$

R-squared (R^2)

R^2, or R-squared, also known as the coefficient of determination, is a statistical measure that assesses the goodness-of-fit of a regression model. It provides information about the proportion of the variance in the dependent variable (the variable being predicted) that can be explained by the independent variables (the predictors) included in the model. The R-squared value ranges between 0 and 1, where 0 indicates that the independent variables do not explain any of the variability in the dependent variable, while 1 indicates that the

independent variables perfectly explain the variability in the dependent variable. Equation 9 shows the formula to calculate R-squared in the context of this experiment.

$$R^2 = \frac{\sum(\text{Prediction} - \text{Actual})^2}{\sum(\text{Prediction } - \text{Mean of data points})^2} \tag{9}$$

Confusion Matrix

A confusion matrix is a useful tool for evaluating the performance of a classification model. It provides a summary of the model's predictions by comparing them to the actual classes of the data. The matrix is typically represented as a square table with rows and columns corresponding to the true and predicted classes, respectively. Each cell in the matrix represents the count or proportion of data points that fall into a particular combination of true and predicted classes. The diagonal cells represent the correctly classified instances, while the off-diagonal cells represent the misclassified instances.

3.2 Traditional Machine Learning

Traditional machine learning is a branch of artificial intelligence that focuses on algorithms and statistical models to help computers learn and make guesses or decisions without being explicitly programmed. It uses labeled training data to train models, which can then be used to examine new data that has never been seen before. Traditional machine learning methods usually require manual feature engineering, in which people pick out relevant features from the data for the models to learn from, as not all data is useful. Naive Bayes, Logistic Regression, SVM, and Random Forests are all examples of traditional machine learning algorithms. [22]

3.3 Deep Learning

Deep learning is a field of study about deep neural networks. A neural network is a computational model that is inspired by biological brains and how they work. It is made up of interconnected nodes called neurons that are linked together and arranged in layers. A neural network is considered 'deep' when it has more than one hidden layer. Each neuron takes in information, runs it through a mathematical function, and sends the result to another neuron. By changing the weights and biases of the connections between neurons during a

process called training, neural networks are capable of learning and identifying patterns in data. Most of the time, this training is done with the help of big data sets and optimization tools. Neural networks can generalize from the training data, which lets them make predictions or organize data they have never seen before. They have been used successfully in many areas, such as speech recognition, image classification, natural language processing, and self-driving systems. Neural networks can range in depth and complexity, from simple feedforward networks to more complicated ones like Generative Adversarial Networks (GANs) and transformers. [22]

3.4 Attention Mechanism and Transformers

The attention mechanism is a key component in modern neural network architectures. It lets the model focus on the parts of the data that are most important for the task at hand. The basic idea behind attention is to give different parts of the input chain different weights. This lets the model focus on or pay more attention to certain parts while making an output. In the field of NLP, attention mechanisms have significantly improved the performance of deep neural networks on machine translation, summarizing text, answering questions, and other tasks. Attention weights are worked out by comparing how similar the current state of the decoder is to each state of the encoder in the input sequence. The attention weights are then used to figure out a weighted sum of the states of the encoder, which is sent to the decoder so that it can make the output.

Transformer is a deep learning model architecture that revolutionized many NLP tasks (and other fields like computer vision). Transformers use the attention mechanism to figure out how words or tokens in an input sequence are related to each other. Transformers are different from traditional recurrent neural networks (RNNs) because they can handle all of the data at once. The encoder and the decoder are the two main parts of a transformer's design. The encoder takes the input order and makes a set of representations that show what the words mean and how they relate to each other. With the aid of these representations and the attention system, the decoder then creates the output sequence. A typical transformer architecture is shown in Figure 1.

Transformer's main innovation is the self-attention mechanism, which, for example, lets NLP models figure out how words depend on each other no matter where they are in the order. This makes it easier for transformers than RNNs to describe long-range dependencies.

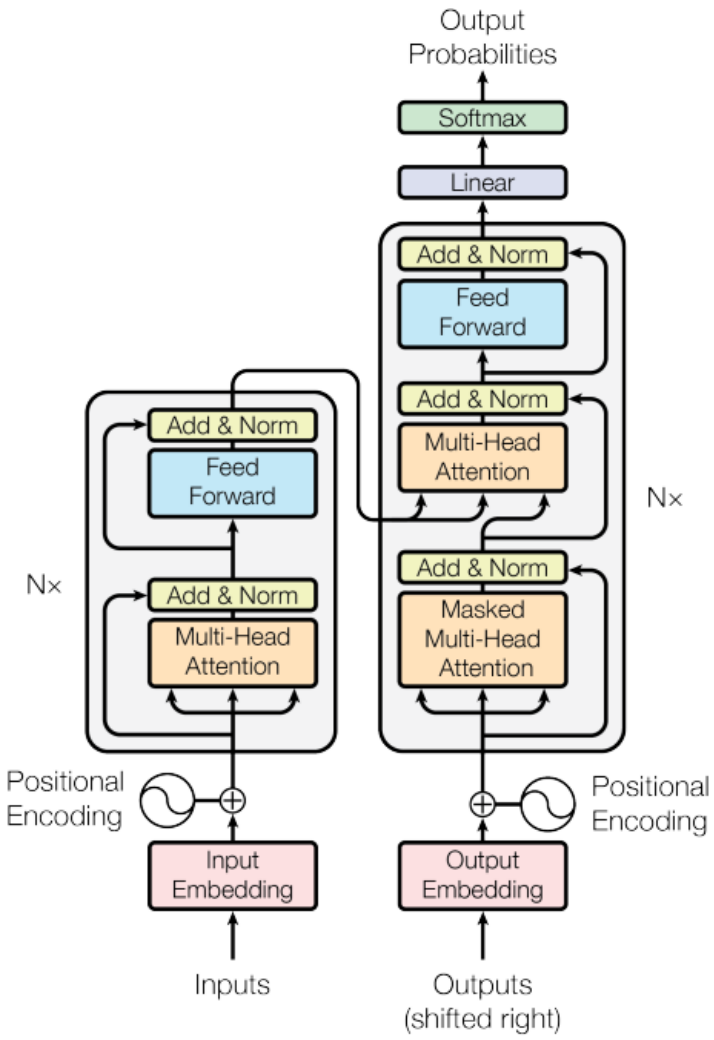


Figure 1: The transformer model architecture. Source: Vaswani et al. [16]

Transformers have achieved state-of-the-art results in various NLP tasks, including machine translation, text generation, sentiment analysis, and named entity recognition. One limitation of the transformer architecture is its maximum input sequence token length (e.g. 512 for BERT), which often leads to information loss when the input is longer than the maximum number of tokens. A few workarounds have been developed, such as different truncation techniques and the use of sliding attention windows, which is the main idea behind Longformer [16] [17].

4 Methodology

This section discusses the methods that are used in the experiments, including Support Vector Machine (SVM), Random Forest, Recurrent Neural Network (RNN), BERT, and Longformer. All implementations used in our experiments are available on GitHub[1].

4.1 Support Vector Machine

SVM is a supervised machine learning algorithm used for classification and regression tasks. It creates a hyperplane that separates data points into different classes by maximizing the margin between them. The algorithm finds the best hyperplane by identifying support vectors, which are data points closest to the decision boundary. SVM can handle high-dimensional data and works well with both linearly separable and non-linearly separable datasets through the use of kernel functions. SVM is known for its effectiveness in handling small to medium-sized datasets and its ability to generalize well to unseen data [23]. For this research, the Python library scikit-learn is used, as it is currently the most widely used library for traditional machine learning algorithms. Since the number of features is large, we chose the linear kernel as our kernel function because it is the fastest to train and the most suitable for text data [24]. Other functions, such as RBF and polynomial, are also considered, but linear outperformed both. RBF might be able to perform better than linear with the right values of C and γ, which we leave for future work.

4.2 Random Forest

Random Forest is an ensemble learning algorithm used for both classification and regression tasks. It constructs multiple decision trees and combines their predictions to make a final prediction. Each tree is built using a random subset of the training data and features, ensuring diversity. During prediction, each tree's output is aggregated to determine the majority or average outcome. This method improves accuracy, handles missing data, and reduces overfitting. Random Forest also provides feature importance, allowing identification of influential variables [25]. We use the scikit-learn implementation of random forest. We use the default parameter for the number of trees (100), as increasing it further will increase training time, and there is no significant increase in performance when we increase the number [26].

[1]Link to source code of implementations: https://github.com/arvindeva/p4k-rating-prediction

4.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural network specifically designed to process sequenced data, such as time series or text, by allowing information to persist and flow from one step to the next. Long short-term memory (LSTM), a variant of RNNs, addresses the limitation of traditional RNNs in capturing long-term dependencies. LSTMs incorporate memory cells and gates that regulate the flow of information, enabling the network to selectively remember or forget information over time. This makes LSTMs particularly effective in tasks involving long sequences and complex dependencies, and they were the de facto state-of-the-art neural networks for NLP tasks until the transformers were introduced [22].

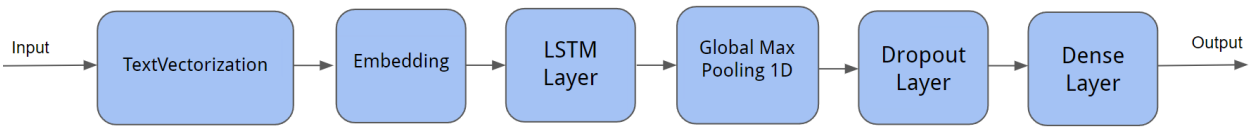


Figure 2: Architecture of our recurrent neural network with LSTM layer.

For the implementation, we use a modified RNN architecture from the official TensorFlow documentation [27]. This architecture consists of an embedding layer, a LSTM layer with 128 nodes, a 1D pooling layer, a dropout layer with a rate of 0.5, a dense layer with 50 nodes, another dropout layer with a 0.5 dropout rate, another denser layer with 50 nodes, and finally an output layer. For the Yelp experiment, the output layer has 5 nodes as there are 5 different classes using the softmax activation function. Since the Pitchfork experiment is a regression task, a minor adjustment was made to the loss function, instead of cross entropy, mean squared error was used. Figure 2 depicts the architecture of our recurrent neural network.

4.4 BERT

In 2018, Google introduced BERT (Bidirectional Encoder Representations from Transformers), a pre-trained language model. BERT revolutionized the field of NLP by achieving state-of-the-art performance on a range of tasks, such as 80.5% GLUE score, 86.7% accuracy in MultiNLI, and 93.2 F1 score in SQuAD [17]. The BERT architecture is based on the transformer model; however, BERT uses a bidirectional approach for training, meaning that BERT considers both left and right contexts of a given token during training and inference.

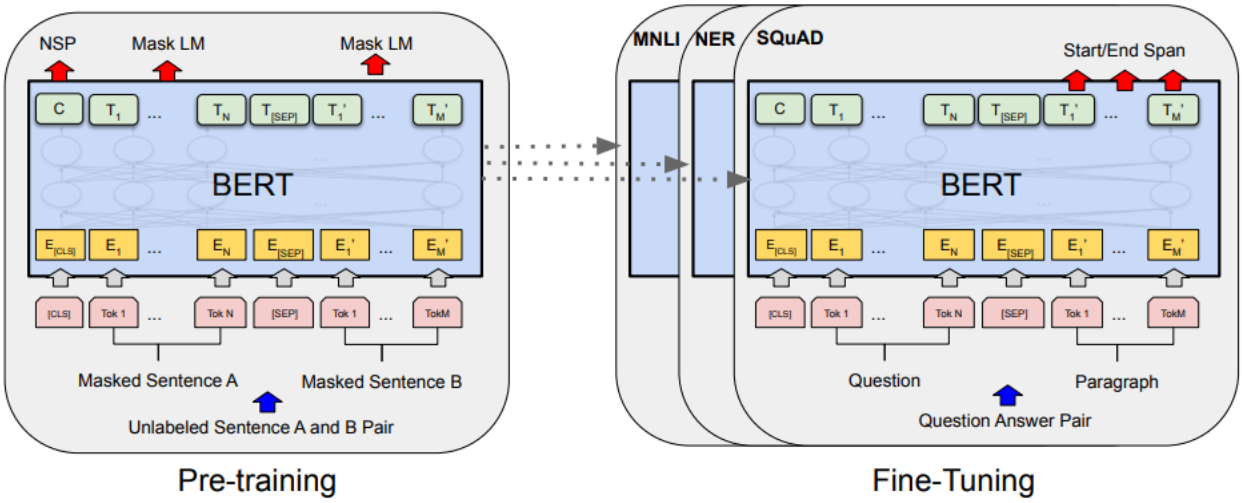


Figure 3: Pre-training and fine-tuning procedures for BERT. The same architecutres are used in both pre-training and fine-tuning except for the output layers. Source: Devlin et al. [17]

BERT has a few key components and steps to note, such as:

- Tokenization: The input text is split into individual tokens, such as words or subwords. BERT uses WordPiece tokenization, which further breaks down words into subword units. BERT has a maximum sequence length of 512 tokens.
- Input Representation: Each token is represented as the sum of three embeddings:
  - Token Embeddings: These are the initial word representations that are learned during training.
  - Segment Embeddings: BERT incorporates sentence-level information by assigning different segment embeddings to tokens from different sentences.
  - Position Embeddings: These embeddings encode the position of each token in the input sequence.
- Transformer Encoder: BERT utilizes a stack of 12 transformer encoder layers. Each encoder layer consists of a self-attention mechanism and a feed-forward neural network.
- Self-Attention: This mechanism allows each token to attend to other tokens in the input sequence, capturing contextual relationships. BERT uses multi-head self-attention, enabling the model to attend to different parts of the input simultaneously.

- Feed-Forward Neural Network: Following the self-attention layer, a feed-forward neural network is applied to each token independently, transforming its representation.
- Pre-training: BERT is pre-trained on large amounts of unlabeled text data using two main objectives:
  - Masked Language Modeling (MLM): Randomly selected tokens in the input are masked, and BERT aims to predict the original words based on their context.
  - Next Sentence Prediction (NSP): BERT is trained to predict whether two input sentences appear consecutively in the original text or are randomly chosen.
- Fine-tuning: After pre-training, BERT can be fine-tuned on specific downstream tasks such as text classification, named entity recognition, question answering, etc. The model is further trained on labeled task-specific data with a task-specific objective.

BERT's bidirectional nature and pre-training on large amounts of data, such as the entirety of the English Wikipedia and the Brown Corpus, enable BERT to capture deep contextual information, making it effective for a wide range of NLP tasks [28]. Figure 3 shows the pre-training and fine-tuning processes of BERT. By fine-tuning BERT on specific tasks, it can adapt and provide state-of-the-art performance on various natural language understanding and generation tasks, including sentiment analysis such as review rating prediction.

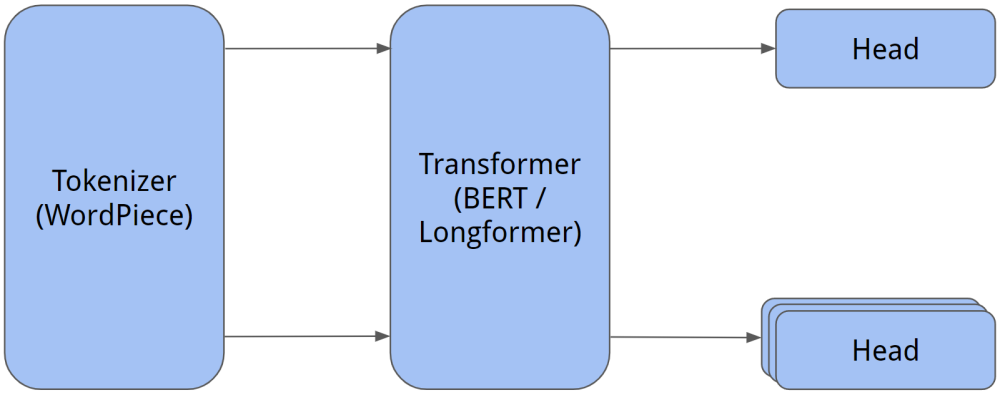


Figure 4: The Hugging Face transformer model. Each model is made up of a Tokenizer, Transformer, and Head. The model is pre-trained with a fixed head and can be further fine-tuned with alternate heads for different tasks. [29]

We use the BERT model implementation by Hugging Face, `bert-base-uncased`. Hugging Face is a company that specializes in NLP, and they provide valuable tools to make

developing NLP models more accessible [29]. Their library, often called the "Huggingface Transformers", allows anyone to easily access and use pre-trained models. We fine-tuned the model using Hugging Face's trainer interface, which works seamlessly with their model by freezing the first layers of BERT and training a fully connected layer attached at the end. Figure 4 depicts the architecture of a Huggingface transformer model. Each model is made up of a Tokenizer, Transformer, and Head. The model is pre-trained with a fixed head and can be further fine-tuned with alternate heads for different tasks. For our BERT model, we use the `BertForSequenceClassification` head.

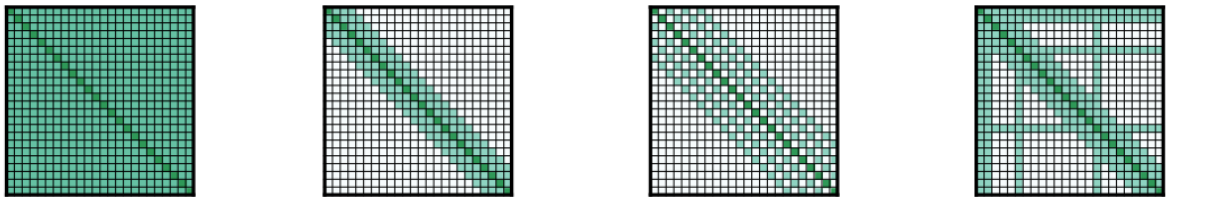
4.5 Longformer

Introduced by Beltagy et al. [19] in 2020, the Longformer architecture is built upon the foundations of the transformer architecture. Transformers rely on self-attention mechanisms to capture the relationships between tokens in a sequence. This comes with a limitation: the standard transformer's computational complexity grows quadratically with sequence length. Therefore, the required computational complexity might be too much for longer sequences. The Longformer addresses this challenge by introducing a sliding window attention mechanism, resulting in linear computational complexity.

4.5.1 Architecture

The sliding window attention mechanism in the Longformer operates in two key steps. Firstly, the input sequence is divided into overlapping chunks of fixed length. For example, if the sequence length is 1,000 tokens and the chunk length is 512, there would be three chunks: [1–512], [256–768], and [512–1,000]. The overlapping regions allow tokens at the boundaries of the chunks to be attended to by multiple chunks, ensuring comprehensive information flow across the entire sequence.

Within each chunk, a modified self-attention mechanism is applied. In traditional transformers, attention is calculated between all pairs of tokens in a sequence, resulting in a quadratic number of computations. However, in Longformer, attention is only computed within each chunk, reducing the complexity to a linear number of computations. This local attention mechanism allows the model to capture dependencies within a limited context window, enabling efficient computation while retaining the ability to understand local relationships between tokens.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 5: The full self-attention pattern and the configuration of attention patterns in Longformer. Source: Beltagy et al. [19]

To incorporate global information from the entire sequence, Longformer includes a global attention head. This head attends to all tokens in the input sequence, allowing the model to capture long-range dependencies that extend beyond the scope of the sliding window. By combining local and global attention patterns, Longformer can effectively model both local context and distant dependencies, providing a comprehensive understanding of the input sequence.

During training, Longformer can be pre-trained in a similar manner to other transformer models, such as BERT. Large-scale language modeling objectives, such as masked language modeling or next sentence prediction, can be employed to learn general language representations. Once pre-trained, Longformer can be fine-tuned for specific downstream tasks using task-specific labeled data. This fine-tuning process adapts the model to the specific requirements of the task at hand.

4.5.2 Implementations

We used the Longformer model from Hugging Face, `longformer-base-4096`, which is then fine-tuned on the Yelp dataset and the Pitchfork dataset. Fine-tuning Longformer was done the same way as fine-tuning BERT. See Figure 4 for a diagram depicting the Hugging Face Longformer model. We use the `LongformerForSequenceClassification` head (provided by the Hugging Face library) to fine-tune the pre-trained Longformer. For the Yelp experiment, cross-entropy is chosen as the loss function as it is a classification task, while for the Pitchfork experiment, mean squared error is chosen as it is a regression task.

4.5.3 Hyperparameter Tuning

The Hugging Face implementation of Longformer provides a set of recommended hyperparameters, which were provided by Devlin et al. [17] in the original BERT paper. This provides a good starting points but it is optimized for BERT's maximum sequence length of 512. As longformer accepts longer sequences of up to 4096, an adjustment to the batch size needs to be made to accommodate this. The recommended batch size values in the recommended set have a minimum value of 16, which is not ideal for Longformer due to memory limitations. Using a GPU card, a batch size of 16 with a 2048 input length would require more than 40GB of GPU RAM. Hence, the candidates for batch sizes are adjusted to a range of 1 to 8.

Parameter	Values
Batch size	1, 2, **4**, 8
Learning rate	5e-5, 3e-5, **2e-5**
No. of epochs	2, 3, 4, **5**

Table 1: Hyperparameter candidate values for grid search. Best results are shown in **bold**.

With the established candidates, as shown in Table 1, a simple grid search is performed with a fixed random seed, and the best results are shown in **bold**. With the selected hyperparameters (in **bold**) and using Google Colab's A100 GPU, the training for the Yelp dataset takes 61 minutes, 73 minutes, and 145 minutes for Yelp-2013, Yelp-2014, and Yelp-all, respectively. For the Pitchfork dataset, the training process takes 3 hours and 41 minutes. Since there are five epochs, the average running time for a single epoch is about 44 minutes. Table 2 shows the training duration of the Longformer.

Dataset	Training Time
Yelp-2013	61 minutes
Yelp-2014	73 minutes
Yelp-All	145 minutes
Pitchfork	221 minutes

Table 2: Training duration of the fine-tuning process of the Longformer on all datasets.

During training, we set the random seed to a fixed value for reproducibility. Fine-tuning Longformer (or any other neural network) requires weights to be initialized randomly. This

causes neural networks to produce different results every time they are trained, especially for transformers, where the results may vary greatly with different random seeds as different seeds have different effects on the behavior of attention [30]. As this is not the focus of this research, five randomly selected seeds were used, and the one that produced the best performance was picked.

5 Dataset

This section discusses the two datasets used in our experiments: the Yelp dataset and the new Pitchfork Review Rating dataset.

5.1 Yelp Dataset

Yelp is an online platform for users to write reviews and give star-ratings to businesses such as restaurants, retail stores, or any other form of business. The platform was primarily used for restaurants before it expanded to include other categories, and it was one of the first user-reviewed online platforms. The Yelp dataset [7], published by the company itself, is a large collection of data consisting of around 150,000 businesses, almost 7 million reviews, 200,000 pictures, and user information from 11 metropolitan areas. It contains information about businesses in various categories, such as restaurants, shopping, entertainment, and more. The dataset includes details about the businesses, such as their names, locations, operating hours, and other attributes. The dataset also includes a substantial number of user reviews, which provide insights and opinions about the businesses. Each review contains information about the user who wrote it, the business being reviewed, the review text itself, and various ratings, such as the overall rating, food quality, service, and more. The dataset is frequently used for various research and analysis purposes, such as studying consumer behavior, sentiment analysis, recommendation systems, and understanding the dynamics of online reviews [31]. It offers a rich resource for exploring relationships between businesses, users, and reviews and for developing machine learning models and algorithms in the domain of online reviews, such as review rating prediction.

5.1.1 Yelp Data Preprocessing

We use three different subsets of the whole dataset: reviews from 2013, 2014, and a random undersampling of the whole dataset. The 2013 and 2014 subsets of reviews are also used by Lyu et al. [20] since Yelp used to organize the Yelp Dataset Challenge in those years. The average word count of the review column is 60 words, substantially smaller than the Pitchfork dataset's average word count. While the dataset provides a lot of information about a business, not all of it is useful for the purpose of this research; only the review texts and the star ratings are used in our experiments. This dataset suffers from data imbalance; as observed in Figure 6, 5-star ratings dominate the distribution with more than

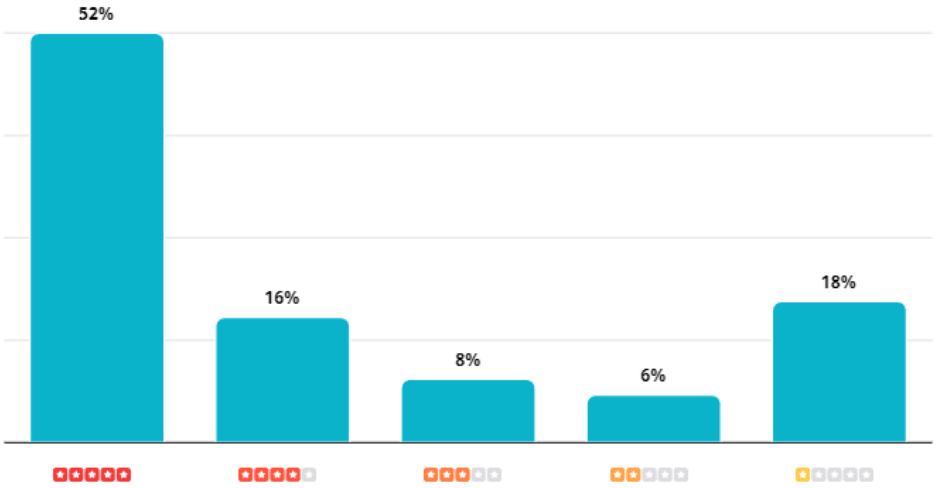


Figure 6: Star distribution of all Yelp reviews. Source: Yelp [32]

50%. A random undersampling process was done to aid with the imbalance. The process is done individually for each subset. Table 3 shows the size and star-rating distributions of the subsets of the Yelp dataset used in our experiment before the train, test, and validation split.

Subset	⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	Total
Yelp-2013	2500	2500	2500	2500	2500	12500
Yelp-2014	2500	2500	2500	2500	2500	12500
Yelp-All	5000	5000	5000	5000	5000	25000

Table 3: Size and star-rating distribution of all subsets of the Yelp dataset before train, test, and validation split

For experiments using traditional machine learning, the preprocessing is done using the Natural Language Toolkit (NLTK) [33]. NLTK is a widely used library for text preprocessing in Python. The NLTK tokenizer and lemmatizer are used to retrieve the list of tokens for each text review in the dataset. The review texts are then converted into numerical feature vectors. While there are many methods to do this, the most popular one is the "term frequency-inverse document frequency", also known as TF-IDF. TF-IDF gives a score to every token, and it aims to highlight words that are more interesting. For the RNN experiment, we used the Keras tokenizer to tokenize the texts. For the experiments that use the Hugging Face library (BERT and Longformer), we use the Hugging Face WordPiece

tokenizers. The tokenization and lemmatization steps are the same for both the Yelp and Pitchfork experiments. Lastly, we split the dataset into train, test, and validation sets, with a split of 80% for training, 10% for test, and 10% for the validation set.

5.2 Pitchfork Dataset

Pitchfork, formerly known as Pitchfork Media, is one of the most influential online music publications that came out in 1995. Since then, it has become a major influence in the music business, focusing on independent and alternative music. Pitchfork's reviews and scores can boost or hinder an album's sales numbers [1]. Beyond music, they have had an effect on movies, literature, and society. It is a place where new artists can be found and promoted. One thing to note is that the word 'album' is used loosely here; it can mean any kind of musical project that Pitchfork chooses to review, such as a full-length LP, EP, LP reissue, CD, or other formats.

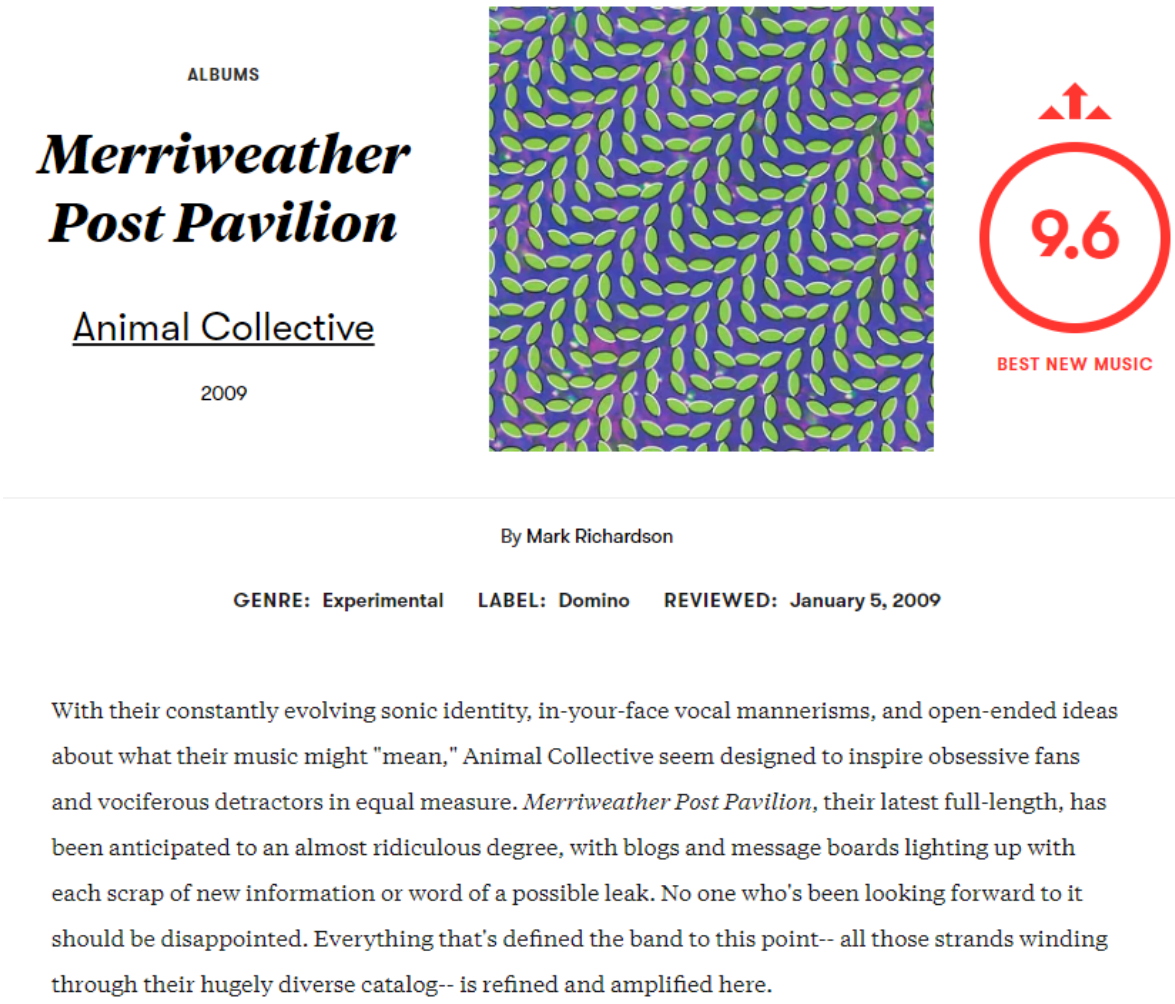


Figure 7: A screenshot of Pitchfork Album Review page of the album "Merriweather Post Pavilion" by the American experimental pop band Animal Collective, only the first paragraph of the text review is shown due to space constraints.

5.2.1 Pitchfork Dataset Collection

Ever since its inception in 1995, Pitchfork has reviewed more than 27,000 albums, and all reviews are archived on the publication's website. As with the Yelp dataset, for our research, two things are needed from each review: the review text and its corresponding numerical rating. Conaway created a Pitchfork dataset on Kaggle in 2018 [4]. Conaway's dataset, collected by means of webscraping, consists of multiple tables, and the 'reviews' table is where all the reviews (from 2000–2017) are stored. This table has useful information such as the album name, the review date of publication, the URL, and the numerical rating (0.0

– 10.0). While this information is useful, it is missing the text review itself. Using the URL column from Conaway's dataset, we iterate over all URLs using a crawler script written with Beautiful Soup, a Python package for parsing HTML documents [34]. As there are more than 18,000 reviews, the script takes more than 5 hours to complete as it has to visit each of the 18,000 review pages. The review text is then saved in a Python list, which then gets stored as a pickle file. This Pitchfork Review Rating dataset has been made available on GitHub[2].

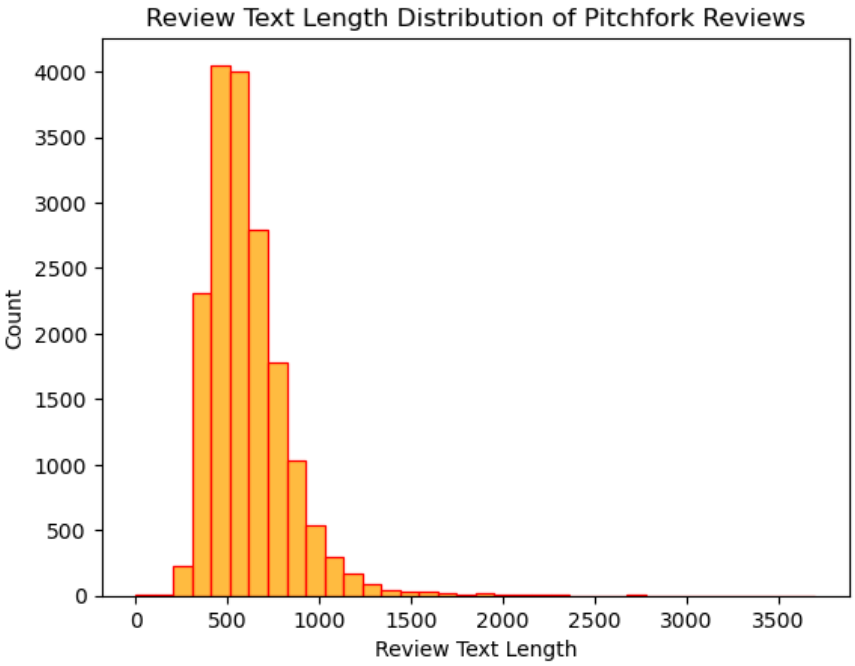


Figure 8: Review text length (in word count) distribution of the Pitchfork Review Rating dataset.

5.2.2 Pitchfork Data Analysis

The Pitchfork Review Rating dataset consists of two columns: the review text and its corresponding numerical rating. There are more than 18,000 reviews in the dataset, and the total word count of the whole dataset (corpus) is around 12 million, where the average length (mean) of a review (document) is 710 words, almost 10 times longer than Yelp's dataset average review text length. As BERT can only accept up to 512 sequence lengths, which

[2]Link to dataset: https://github.com/arvindeva/p4k-rating-prediction

is less than the length of an average review, it is expected that in many cases the review text has to be truncated before BERT is applied, which might lead to information loss [35]. Longformer can handle longer texts and thus does not have to cope with any such information loss. Figure 8 shows the distribution of word counts for all reviews. We observe that some of the reviews go beyond 2048 words, which is more than half of the Longformer limit, before they get tokenized into subwords. Compared to other big music publications, Pitchfork review lengths are longer than average. For example, the average word count of music album reviews in Rolling Stone and NME is 312 and 403, respectively.

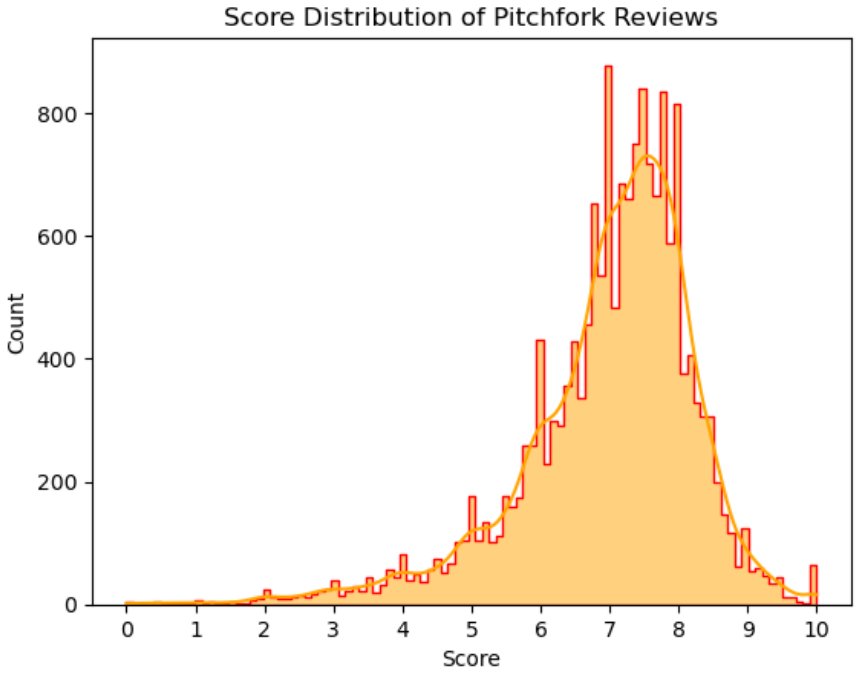


Figure 9: Score distribution of the Pitchfork Review Rating dataset.

Pitchfork is known for their opinionated rating scores, so it is worth looking at how their ratings are distributed. Figure 9 shows the distribution of scores for all Pitchfork reviews in the dataset. We can observe that the majority of their reviews fall between 6.5 and 8.3. It is interesting to see that there are reviews below 3.0, albeit not that many, because this rarely happens in other publications. Other major music publications rarely rate an album a 5.0 or less (on a 0.0 - 10.0 scale). The mean rating of all reviews is 7.0. The most frequent score is 7.0, and the median score is 7.2. Their status as "opinionated" is also why Pitchfork is one of the most influential review publications, and why they have an effect on their readers' decisions to listen to a certain music album. As there is so much music being released every day, it

is impossible for Pitchfork to review every album in existence, so they need to have a certain degree of selectiveness in choosing which album to review. They tend to gravitate toward certain genres. Figure 10 depicts the distribution of the album genres that they reviewed. Rock is the most represented genre, and this is because originally Pitchfork was more focused on reviewing alternative rock albums before branching out and reviewing other genres.

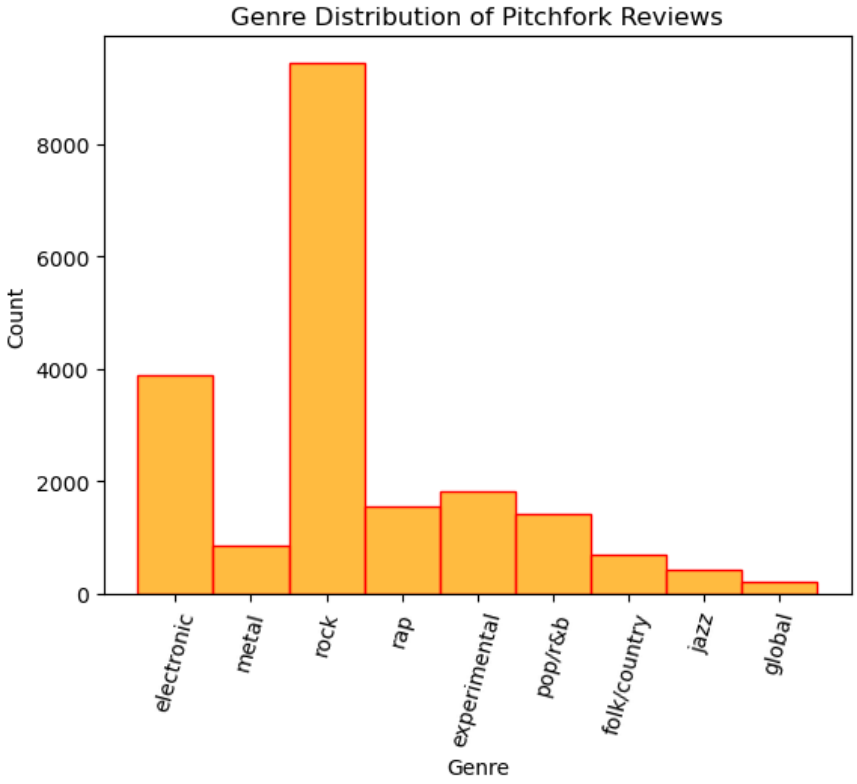


Figure 10: Album genre distribution of Pitchfork reviews dataset.

5.3 Data Preprocessing

After scraping the Pitchfork dataset, we removed any null values, such as empty strings. A total of 323 empty strings were removed. This was caused by our scraper occasionally failing to find and extract the right HTML tags of the review texts from the review web page. Secondly, we handled the data imbalance issue present in the dataset. As observed in Figure 9, the majority classes far outnumber the minority classes. This imbalance negatively affects the performance of some of the models, especially the RNN, BERT, and Longformer, while the effect on traditional machine learning methods is negligible. To remedy the imbalance,

the same method as the Yelp dataset, random undersampling, is used. All the majority classes are randomly undersampled until they have a maximum frequency of 60. 60 is obtained by running training runs with candidates from the set $\{40, 50, 60, 70, 80\}$, where 60 comes out with the best result. The distribution of ratings in the undersampled dataset is shown in Figure 11. This undersampling process reduces the number of reviews from 17,420 to 3,869. Finally, the 3,869 reviews are then split into three categories, the training set, the validation set, and the test set, with a ratio of 80:10:10, respectively, amounting to 3095 data points for the training set, 387 for the validation set, and 387 for the test set.

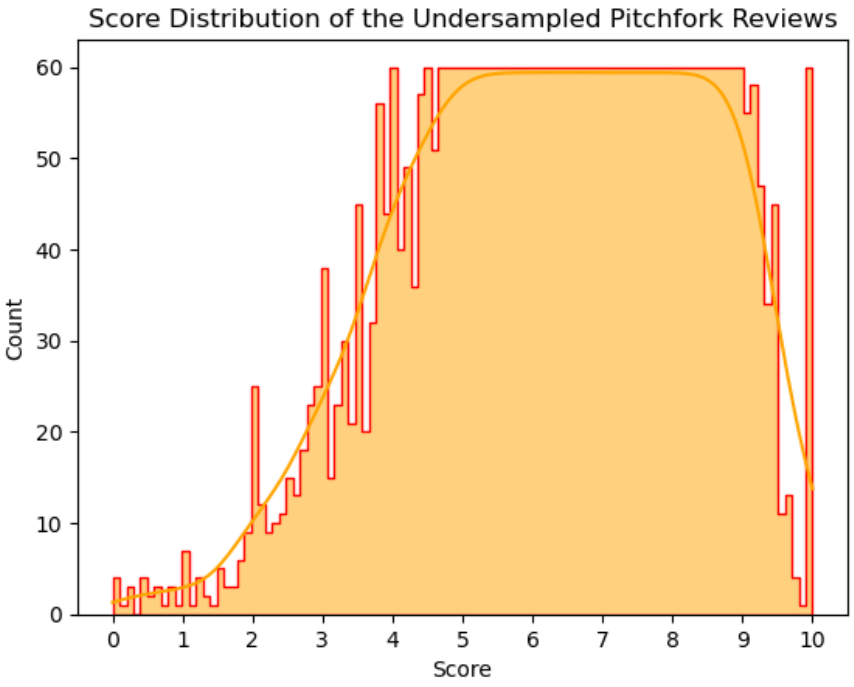


Figure 11: Score distribution of Pitchfork reviews after undersampling the majority classes to be inline with other classes.

6 Experiments and Results

In this section, we present our experiments and their results, followed by notable observations. The most relevant information is displayed in the figures and tables.

6.1 Experiments

Two different experiments are conducted: one for the Yelp dataset and one for the Pitchfork dataset. For both experiments, the same set of algorithms are used: SVM, Random Forest, RNN + LSTM, BERT, and Longformer. The Yelp experiment is a multiclass ordinal classification task where the classes are the number of assigned stars, ranging from 1 star to 5 stars. The Pitchfork experiment is a regression task where the output is a numerical rating between 0.0 and 10.0.

6.2 Results: Yelp

The results of the Yelp experiment are displayed in Table 4. It is observed that Longformer outperforms other methods in every subset of the Yelp dataset. For the Yelp-All dataset, Longformer outperformed the next best model, BERT, by 3.5% accuracy. Our results are comparable to state-of-the-art results from Sun et al. [18] and Lyu et al. [20].

Method	Yelp-2013		Yelp-2014		Yelp-All		
	Acc. (%)	RMSE	Acc. (%)	RMSE	Acc. (%)	MAE	RMSE
Random Forest	52.2	0.902	54.5	0.898	55.6	0.622	0.893
SVM	56.4	0.914	56.7	0.926	60.2	0.618	0.885
RNN+LSTM	63.6	0.675	62.3	0.678	62.3	0.426	0.677
BERT	68.7	0.619	69.1	0.622	69.3	0.371	0.614
Longformer	**71.0**	**0.578**	**72.4**	**0.575**	**72.8**	**0.269**	**0.570**

Table 4: Results of the test set of Yelp-2013, Yelp-2014, and Yelp-All. Experiments were run once to obtain Accuracy (↑) and RMSE (↓). MAE (↓) is shown for Yelp-All for the purpose of comparison to the result of the Pitchfork experiment. Best values are shown in **bold**.

.

The results show that neural network models perform better than traditional machine learning methods and that attention mechanisms improve the quality of a neural network for a review rating prediction task. As the performance gets better, the accuracy increases while the RMSE decreases.

	Precision	Recall	F_1	support
⋆	0.76	0.67	0.71	432
⋆⋆	0.66	**0.77**	0.71	526
⋆⋆⋆	0.72	0.73	0.73	555
⋆⋆⋆⋆	0.73	0.69	0.71	518
⋆⋆⋆⋆⋆	**0.81**	**0.77**	**0.79**	469
Macro Avg.	0.73	0.73	0.73	2500
Weighted Avg.	0.73	0.73	0.73	2500
Accuracy		0.73		2500

Table 5: Classification results of the fine-tuned Longformer model trained on the Yelp-All Dataset. Best values are shown in **bold**.

While accuracy is a sufficient metric to evaluate a classifier's performance, it does not provide information about the model's performance for each class present in the data (star ratings). Taking a closer look at the best-performing model, Table 5 shows a more detailed information to analyze the performance of the fine-tuned Longformer. The results show that Longformer has the best performance when predicting 5-star reviews, reaching **0.81** precision, **0.77** recall (tied with 2-star), and **0.79** F_1. 1-star, 2-star, and 4-star reviews are harder to classify correctly as they have the lowest F_1 score.

We observed that 1-star reviews have the second highest precision while having the lowest recall, in contrast to 2-star reviews, which have the highest recall while having the lowest precision, which means the model is most likely to get confused when predicting between 1-star and 2-star reviews. Figure 12 shows the confusion matrix of the predictions made by the fine-tuned Longformer. The highest confusion rate occurs when the model predicts a 1-star review as 2-stars. The second most confused prediction is when the model predicts a 5-star review as a 4-star.

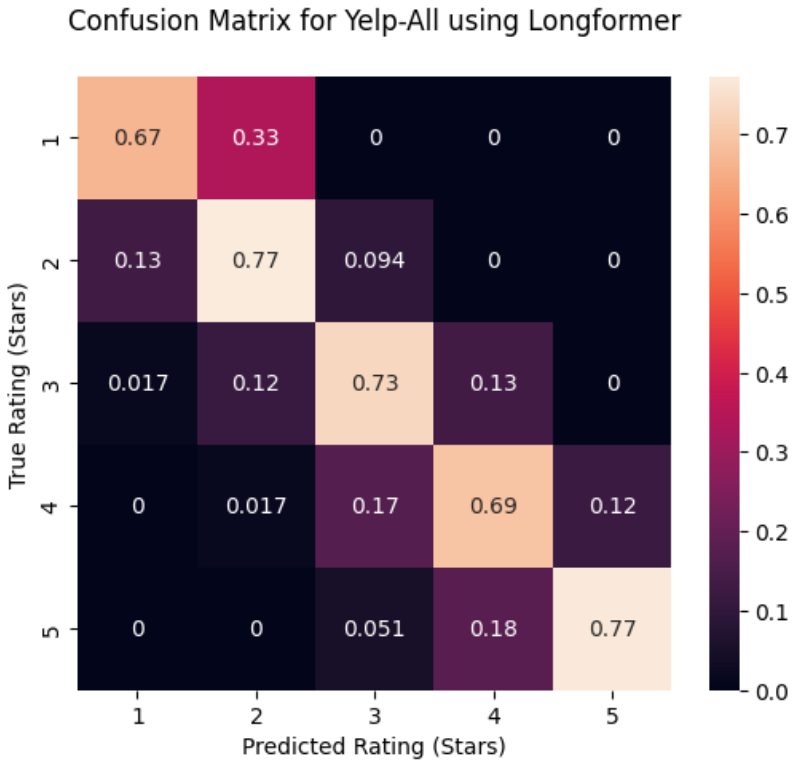


Figure 12: Confusion matrix of the predicted and actual stars by the Longformer model trained on the Yelp-All dataset

6.3 Results: Pitchfork

Method	MAE	MSE	RMSE	r	R^2
Random Forest	0.947	1.734	1.317	0.775	0.601
SVM	0.959	1.756	1.325	0.772	0.599
RNN+LSTM	0.887	1.556	1.247	0.801	0.643
BERT	0.823	1.237	1.097	0.842	0.711
Longformer	**0.689**	**0.829**	**0.910**	**0.902**	**0.799**

Table 6: Performance of machine learning models on the test set of the balanced Pitchfork Review Rating dataset. Experiments were run once to obtain MAE (↓), MSE (↓), RMSE (↓), and R^2 (↑). Best values are shown in **bold**.

The results of the Pitchfork experiment are shown in Table 6. It can be observed that transformers perform better than RNNs and that neural networks in general perform better than traditional machine learning algorithms. Longformer is the best-performing model, outperforming BERT by about 0.134 in MAE. It also has the best R^2 value, meaning Long-

former is able to account for a significant amount of the variation in the data and provides a good fit. Longformer is also the best performing model in capturing the correlation between predicted rating (based on review text) and actual rating, since it has the highest value of the Pearson correlation coefficient. Pearson's r does not show how good a model is at predicting, as it only shows the linear relationships between two variables (in this case, the review text and its corresponding rating).

The result of training for the Longformer model is shown in Figure 13, where it can be observed that the validation loss follows the training loss' trend, indicating that there is no overfitting during training. The best-performing model is deployed and has been made available for inference on Hugging Face [3] for anyone to try out.

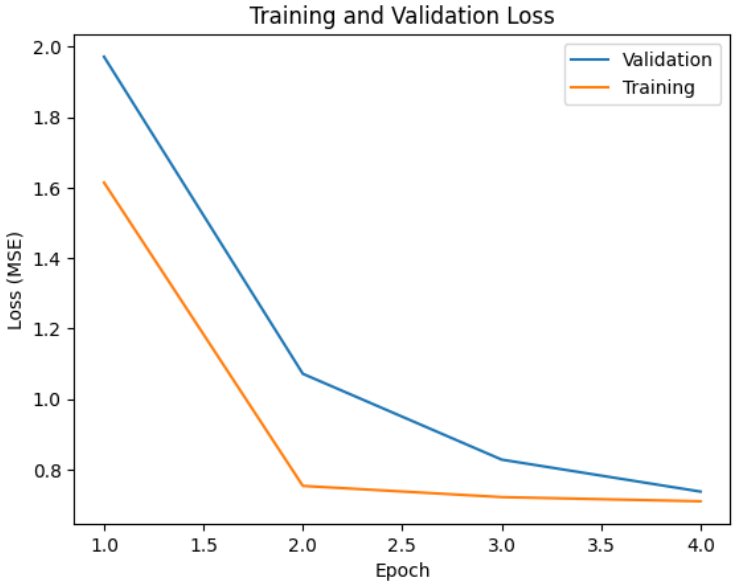


Figure 13: Validation and training loss (MSE) for Longformer trained on the balanced Pitchfork dataset.

A confusion matrix of the predicted and actual values could establish more information; however, as this is a regression problem where the ratings are continuous, it is not straightforward to form a confusion matrix. To create a corresponding confusion matrix, the results could be transformed into a kind of classification problem. To do so, all the prediction and real rating values will be rounded to the nearest integer, between 0-10, and as such, the

[3]Inference link: https://huggingface.co/spaces/Rveen/p4k-longformer

results can be treated as a classification problem with 10 different classes. The resulting confusion matrix can be seen in Figure 14. It is crucial to note that this confusion matrix is not a good representation of the model's performance due to the rounding method used, and is only generated to provide more insights on which ratings raise any confusion for the model. The matrix shows that the model never predicted a 1, 2, or 10, meaning that our fine-tuned Longformer never learned to give an album a rating of less than 2.5. The confusion matrix could also be formed by fine-tuning the Longformer neural network for a classification problem of 5 classes, which we leave for future work.

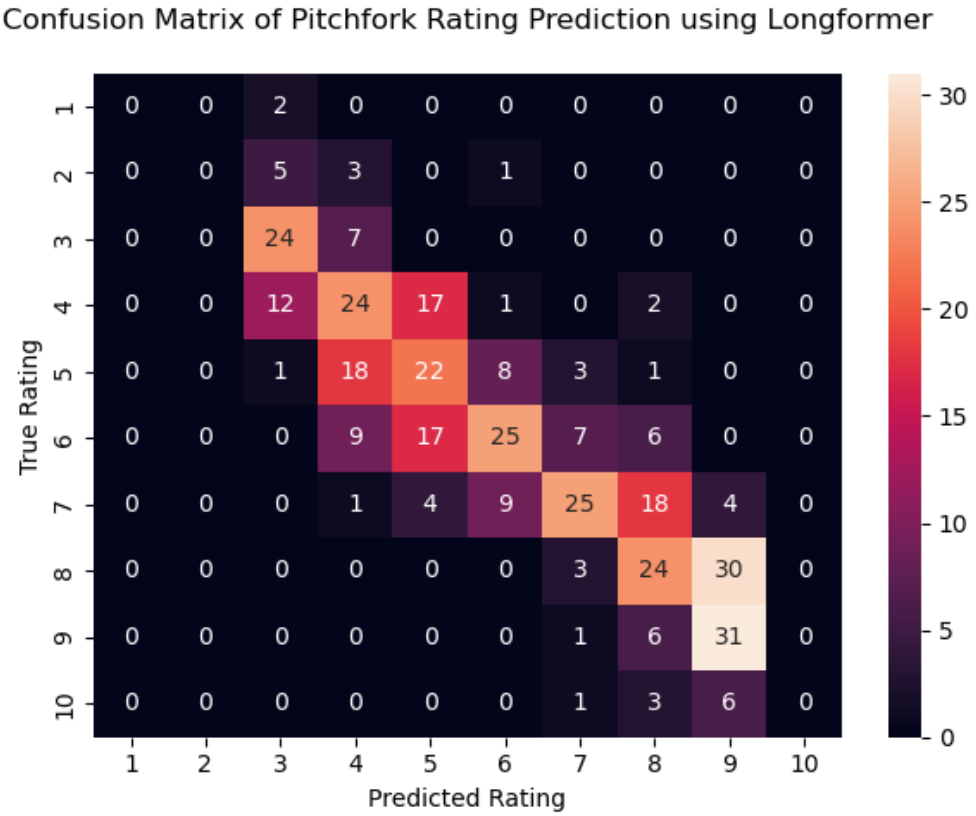


Figure 14: Confusion matrix of the predicted and actual stars by the fine-tuned Longformer model trained on the Pitchfork dataset.

7 Discussion

This section provides a discussion of the results obtained from the Yelp and Pitchfork experiments.

With the results obtained from the Yelp and Pitchfork datasets, we answer the first research question: "To what extent are machine learning algorithms able to predict a music album rating based on its text review?". As observed in the results, Longformer is the best-performing model for predicting music album ratings, outperforming all other methods in the Pitchfork experiment. Furthermore, it achieves 0.799 R^2, indicating that the model seems to be able to explain a significant amount of the variation in the data.

While Longformer achieves the highest performance for both datasets, one thing worth noting is how much it outperforms the other methods. In the Yelp experiment, Longformer outperforms the second best performing method, BERT, by 0.102 MAE. In the Pitchfork experiment, it outperformed BERT by 0.134. MAE is used here due to its simplicity of interpretation. While 0.102 is close to 0.134, the two datasets have different scales for the ratings. In the Yelp experiment, the range of ratings is between 1 and 5, while in the Pitchfork experiment, the range of ratings is between 0.0 and 10.0. This means that the difference in MAE is more significant in the Pitchfork dataset. We expect that the reason for this is that the two datasets have different review text lengths. As mentioned in Section 5, the Yelp dataset review texts have an average length of 60 words, whereas the Pitchfork reviews have an average length of 700 words. With the Pitchfork reviews having longer reviews, the performance of Longformer is expected to outperform BERT by a greater margin than on a dataset with a shorter average length, such as Yelp.

Looking at the confusion matrix of the Pitchfork dataset (See Figure 14), it can be observed that the model never made a 1-star or 2-star prediction. The lowest prediction the model made was 2.6, which got rounded to 3 in the confusion matrix. This means that the model never learned the capability of rating an album higher than a 2.6. Either this is caused by a data imbalance (not enough data points with lower ratings) or it is because these areas of low ratings are difficult to distinguish between each other; hence, the model failed to learn to differentiate between albums with low ratings.

With the results from our experiments, we are able to answer our second research ques-

tion: "Is there a correlation between an album's text review and its corresponding numerical rating?". The results of the Pitchfork experiment show that models with better performance also have a higher Pearson's r, with Longformer having an r value of 0.9, indicating that there is a high degree of correlation between the predicted and actual ratings (as the predicted rating increases, so does the actual). Another important observation can be made by looking at the R^2 scores. Longformer achieved an R^2 score of 0.799, indicating that a large proportion of the variability in the actual score can be explained by the prediction score, which is obtained from the text review. In other words, while the coefficient of determination is not a metric of accuracy, it tells us that our Longformer model is able to show a high degree of correlation between text review and numerical rating since almost 80% of the variation in the dependent variable (ratings) can be attributed to the independent variable (reviews).

The result of the Pitchfork experiment tells us that well-performing models are able to 'understand' the sentiments of music reviewers, but only to a certain degree. As we can see, the best MAE the model achieved is 0.689, which means, on average, the model's prediction is off by 0.7 (on Pitchfork's rating scale). The quality of the prediction is highly subjective, as there is no quantifiable way to tell whether missing a prediction by 0.7 is good or not. This comes back to the concept of music criticism itself, which by itself is also highly subjective. The numerical ratings are something that reviewers come up with, including their personal biases, and a computer may not be able to replicate them. Even humans might have a hard time predicting a numerical rating based on its review text. For example, how can one distinguish between a 4.0 and 4.5 rating when it is highly subjective? This experiment tells us that while there is a positive correlation, it is only up to a certain point. With this knowledge, people should not blindly trust music critics, as reviews and ratings are just opinions and highly biased.

8 Conclusion and Future Work

In this paper, we present two experiments in review rating prediction, a sentiment analysis task of predicting a numerical rating based on its text review. For our experiments, two datasets are used: the Yelp dataset, a benchmark dataset for review rating prediction, and the Pitchfork Review Rating dataset, a new dataset consisting of text reviews and numerical ratings of Pitchfork's album reviews.

Our first experiment uses several machine learning algorithms, SVM, Random Forest, RNN with LSTM, BERT, and Longformer, to predict the rating of a Yelp review based on its text review. The second experiment is a novel experiment in music review rating prediction, utilizing our new Pitchfork Review Rating dataset. The first research question, "To what extent are machine learning algorithms able to predict a music album rating based on its text review?", can be answered from this experiment, as the results show that Longformer achieved a mean absolute error of 0.689.

The results of our Pitchfork dataset experiment can also answer the other research question of whether there is a correlation between an album's text review and its numerical rating. The results show that Longformer achieved 0.902 Pearson's r and 0.799 R^2 values. This indicates that while a correlation does exist, it is only up to a certain degree, and that 80% of the variability in the actual score can be explained by the prediction score that is obtained from the text reviews.

This study extends the research on NLP tasks in the field of review rating prediction by presenting a novel experiment using a music review dataset. With the results of our experiments, we hope to give music listeners new insights on how they perceive music reviews; for example, they should not treat the ratings given by music critics as a definite value, as there are always personal biases behind a rating, since neither computers nor humans can predict numerical ratings based on the text reviews perfectly.

There are improvements that can be made to this study. Since we chose Longformer as our main model, it would be better to compare it to other models that were designed specifically for longer documents, as BERT does not accept longer sequences. This made Longformer the obvious winner since that's what Longformer is for: longer documents. Another model that we considered was GPT, another large language model, but due to time

constraints, we opted not to use the model.

Another improvement that could be made is in how we evaluate the performance of our models. It was hard to evaluate the performance of our model due to the lack of comparable studies in the domain of music album review-rating prediction. One way to make this better is to incorporate human evaluations. For example, we can have human predictors so that a comparison between the model and humans can be observed, giving further insights on the performance of the model. We could also do the Pitchfork review rating prediction experiment as a 5-class classification task to make it more similar to our Yelp review rating prediction experiment.

References

[1] M. B. Briskin, "Quantifying the pitchfork effect." [Online]. Available: https://blogs.brown.edu/econ-1400-s01/files/2015/01/ECON1400_MichaelBriskin.pdf

[2] S. Kornhaber, "Pitchfork, the reluctant men's magazine," 2015. [Online]. Available: https://www.theatlantic.com/entertainment/archive/2015/10/conde-nast-buys-pitchfork-for-the-millennial-men/410341/

[3] K. Borovinsky, "Pitchfork's reviews section by the numbers." [Online]. Available: https://pitchfork.com/features/lists-and-guides/25-years-of-pitchfork-reviews-by-the-numbers/

[4] N. Conaway, "18,393 pitchfork reviews," 2017. [Online]. Available: https://www.kaggle.com/datasets/nolanbconaway/pitchfork-data

[5] P. D. Turney, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 417–424. [Online]. Available: https://doi.org/10.3115/1073083.1073153

[6] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '04. USA: Association for Computational Linguistics, 2004, p. 271–es. [Online]. Available: https://doi.org/10.3115/1218955.1218990

[7] Yelp, "Yelp dataset," 2016. [Online]. Available: https://www.yelp.com/dataset

[8] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: https://aclanthology.org/P11-1015

[9] A. Bhatt, A. Patel, H. Chheda, and K. Gawande, "Amazon review classification and sentiment analysis," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 6, pp. 5107–5110, 2015. [Online]. Available: https://ijcsit.com/docs/Volume%206/vol6issue06/ijcsit2015060652.pdf

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

[11] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using lstm for region embeddings," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 526–534. [Online]. Available: https://proceedings.mlr.press/v48/johnson16.html

[12] ——, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 562–570. [Online]. Available: https://aclanthology.org/P17-1052

[13] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 328–339. [Online]. Available: https://aclanthology.org/P18-1031

[14] N. Asghar, "Yelp dataset challenge: Review rating prediction," 2016. [Online]. Available: https://arxiv.org/abs/1605.05362

[15] Z. Liu, "Yelp review rating prediction: Machine learning and deep learning models," 2020. [Online]. Available: https://arxiv.org/abs/2012.06690

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[18] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" in *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*. Springer, 2019, pp. 194–206. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-32381-3_16

[19] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020. [Online]. Available: https://arxiv.org/abs/2004.05150

[20] C. Lyu, L. Yang, Y. Zhang, Y. Graham, and J. Foster, "Exploiting rich textual user-product context for improving sentiment analysis," 2022. [Online]. Available: https://arxiv.org/abs/2212.08888

[21] A. T. Pinter, J. M. Paul, J. Smith, and J. R. Brubaker, "P4kxspotify: A dataset of pitchfork music reviews and spotify musical features," 2020. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/7355

[22] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.

[23] S. Guido, *Introduction to Machine Learning with Python*. O'Reilly Media, 2016.

[24] C.-J. L. C.-W. Hsu, C.-C. Chang, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Tech. Rep., 2003.

[25] S. C. Gopinath Rebala, Ajay Ravi, *An Introduction to Machine Learning*. Springer, 2019.

[26] E. S. Gérard Biau, "A random forest guided tour," *Springer*, 2016. [Online]. Available: https://doi.org/10.1007/s11749-016-0481-7

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner,

I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[28] A. Merchant, E. Rahimtoroghi, E. Pavlick, and I. Tenney, "What happens to BERT embeddings during fine-tuning?" in *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Online: Association for Computational Linguistics, Nov. 2020, pp. 33–44. [Online]. Available: https://aclanthology.org/2020.blackboxnlp-1.4

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[30] P. Madhyastha and R. Jain, "On model stability as a function of random seed," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 929–939. [Online]. Available: https://aclanthology.org/K19-1087

[31] X. Lei, X. Qian, and G. Zhao, "Rating prediction based on social sentiment from textual reviews," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1910–1921, 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7484319

[32] Yelp, "Fast facts." [Online]. Available: https://www.yelp-press.com/company/fast-facts/default.aspx

[33] E. Loper and S. Bird, "Nltk: The natural language toolkit," 2002.

[34] L. Richardson, "Beautiful soup documentation," *April*, 2007. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[35] M. A. Mutasodirin and R. E. Prasojo, "Investigating text shortening strategy in bert: Truncation vs summarization," in *2021 International Conference on Advanced*

Computer Science and Information Systems (ICACSIS), 2021, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9631364