Finetuning Transformer Encoders for Classifying Movie Reviews

Moses Odei Addai, Jakob Simmons, Emma Vejcik

Abstract

Predicting movie ratings based on the text of the review provides useful information for sentiment analysis and can be used to improve recommendation systems. We have fine-tuned a BERT model to generate embeddings of movie reviews for classification based on the score the movie was given. We trained a multilayer perceptron head to classify these reviews based on the extracted embedding from the BERT model. Our methods make use of a strong, pretrained BERT model as our starting point from the Hugging Face library and a small, fully connected, neural network for classification. To evaluate our model we used a labeled IMDB dataset containing the raw score of the movie review, not a sentiment analysis of the review text.

1 Introduction

The dataset we decided to analyze is Kaggle IMDB Movie Reviews for 2021, a set of IMDB reviews from 2021 containing the following data for 4000+ reviews. Unlike other movie review datasets that provide pre-annotated sentiments, this dataset allows direct mapping to numerical ratings, offering a richer training signal for classification tasks.

- **id**: A unique key for reviews in the dataset
- **review**: The text of the review
- **rating**: The given score between 1-10
- **author**: The author of the review
- **title**: The title of the review (not the movie)

An example tuple from the dataset can be found in table 1

ID	Review	Rating	author	Title
2	I cannot believe anyone could give this film less than a 6, I gave it an 8 because I thought it was well acted, great filming, and an exciting story. C'mon man, what's with all the negativity?	8	joemay-2	What are all the bad reviews about? Is it a woke thing?

Table 1: An example tuple from the Kaggle dataset

Our model will be trained using the **review** as our input feature and **score** as a label for a multiclass classification problem. This is an interesting problem because the model can be used to assign scores to reviews left without scores, improving the websites recommendation algorithm. In order to create

Preprint. Under review.

a usable feature from the review string, we decided to fine-tune a transformer encoder to generate embeddings of the review which could be classified using a MLP head.

Transformer encoders are an important part of large language models because of their ability to create comprehensive, context-aware embeddings of text and other media. Compared to methods such as Doc2Vec, transformer encoders offer two main advantages: contextualized word embeddings, and the ability to model relationships between all words in the text.

Doc2Vec and other simple text vectorizers generate static word representations, meaning words with multiple meanings lose contextual significance. For example Doc2Vec won't distinguish between the word bank referring to a river and bank referring to a financial institution. On the other hand, transformer encoders can understand these differences, and provide embeddings that account for context.

Another important limitation of vectorizers like Doc2Vec are their fixed context window, this means that if two words which provide important context together do not fall within the window, their relationship is not captured. Transformers leverage a method called self-attention, which allows them to model connections between words across the entire text, no matter how far apart they are. This helps transformers capture the full meaning of text, including long-range dependencies between words [**?**].

These two benefits are what motivated our use of transformer encoders to generate embeddings. However, training a transformer encoder from scratch takes a lot of compute power so we determined that fine-tuning a pretrained encoder to our classification task would be a more efficient approach.

Fine-tuning is a method for tuning a pretrained model for a specific task to improve the embeddings create. Fine-tuning consists of freezing a number of layers at the beginning of a model so that their weights are not changed, and setting a lower than normal learning rate for the remaining weight layers. This allows the pretrained encoder to be optimized for a specific class, while still maintaining the general features that the encoder learned during it's initial training. In this case we are finetuning a generalized text encoder for the purposes of classifying movie Reviews.

Methodology

Our model consists of two main components: a pretrained transformer encoder, which we fine-tune, and a multilayer perceptron (MLP) classification head, which is attached to the encoder output. We formally define our model as:

$$F(x; \theta, \theta', \varphi) = h(\phi(x; \theta, \theta'), \varphi)$$

where:

- θ represents the frozen parameters of the pretrained transformer encoder.
- θ' denotes the fine-tuned parameters of the encoder.
- $\phi(x; \theta, \theta')$ is the transformer model that generates an embedding for input text $\mathbf{x}$.

The number of transformer layers fine-tuned to update θ' is controlled by the

As discussed in the Introduction, the transformer encoder extracts contextualized feature representations from movie reviews, denoted as:

The number of hidden layers that are being finetuned to create θ' is determined by the hyperparameter **% Finetuned** found in table 2.

As discussed in the Introduction, the transformer encoder extracts contextualized feature representations from movie reviews, denoted as:

$$\mathbf{E} = \phi(x; \theta, \theta')$$

These embeddings $\mathbf{E}$ are then passed to the classifier:

$$\tilde{\mathbf{y}} = h(\mathbf{E}, \varphi)$$

where $\tilde{\mathbf{y}}$ is the predicted review score class. The model's performance is optimized by minimizing the classification loss between the predicted labels $\tilde{\mathbf{y}}$ and the ground truth labels $\mathbf{y}$.

2 Model Implementation and Hyperparameter Tuning

To efficiently implement our approach, we utilized the Hugging Face Library to load a pretrained BERT model , which we then fine-tuned for multi-class classification.

For classification, we employed the cross-entropy loss function , defined as:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log \tilde{y}_i$$

where:

- C is the number of classes,
- y_i is the true one-hot encoded label, and
- $\tilde{y}_i$ is the predicted probability for class i.

2.1 Model Architecture

As discussed in the Introduction , the transformer encoder extracts key textual features from movie reviews, producing an embedding:

$$\mathbf{E} = \phi(x; \theta, \theta')$$

where:

- x is the input text,
- θ represents the frozen parameters of the pretrained transformer,
- θ' represents the fine-tuned parameters .

These embeddings serve as input to a classification model :

$$\tilde{\mathbf{y}} = h(\mathbf{E}, \varphi)$$

where h is the MLP classifier with parameters φ, which assigns a predicted review score. The model is optimized by minimizing the loss between the predicted labels $\tilde{\mathbf{y}}$ and the true labels $\mathbf{y}$.

2.2 Model Selection and Computational Constraints

Initially, we experimented with the Gemini 5T encoder , which contains approximately 770 million parameters . However, fine-tuning and hyperparameter optimization proved computationally infeasible. As a more practical alternative, we opted for BERT , which has 110 million parameters , making fine-tuning significantly more manageable.

We leveraged the Hugging Face Library to load a pretrained BERT-base-uncased model for fine-tuning.

2.3 Hyperparameter Search

We first evaluated the hyperparameter space for the Gemini model, shown in Table 2.

The fixed hyperparameters for the Gemini model are listed in Table 3.

% Finetuned	Hidden Size	Number of Hidden Layers
0, 25, 50, 75, 100	256, 512, 1024	1, 3

Table 2: Hyperparameter space tested for the Gemini model

Dropout	Learning Rate	Epochs	Weight Decay
0.1	2×10^{-5}	5	0.01

Table 3: Fixed hyperparameters for the Gemini model

2.4 Alternative Model Approaches

Due to the computational inefficiency of Gemini, we proceeded with three alternative models :

1. **Gemini Agent Model** Instead of fine-tuning, we utilized a rule-based Gemini agent that classifies reviews directly based on textual input using predefined instructions.

   ```
   You are a highly accurate movie review classification service. Your task
   is to analyze a given movie review and assign it a rating based on the
   following system:

   - Ratings range from  0 to 9 , where:
     -  0  represents the worst possible review.
     -  9  represents an excellent review.

    Instructions:
   - You must  only  return a  single integer  from this set:
      0, 1, 2, 3, 4, 5, 6, 7, 8, 9 .
   -  Do not  include any additional text, explanations, or symbols.
   - If sentiment is unclear, select the rating that best reflects the overall
   tone.
   ```

2. **Fine-Tuned BERT Model** - We transitioned from a 10-class to a 3-class classification task to simplify training. - The hyperparameter space explored for BERT is detailed in Table 10.

% Finetuned	Epochs
0, 25, 50, 75, 100	1-10

Table 4: Hyperparameter space tested for the BERT model

3. **MLP with TF-IDF Vectorization** - As a non-transformer baseline, we trained an MLP classifier using TF-IDF embeddings of reviews. - Table 9 lists the hyperparameters used for this approach.

MLP with TF-IDF Vectorization: As a non-transformer baseline, we trained an MLP classifier using TF-IDF embeddings of reviews. Table 9 lists the hyperparameters used for this approach.

The second was the main model that we focused on. We tested the hyperparameter space found in table 10 on the Bert encoder. From this point on we also changed from training on 10 labels to training on 3 labels

- 0 - rating is <= 3
- 1 - rating is >3 and <=6
- 2 - rating is >6

using the fixed hyperparameters found in table 11

Max Features	Hidden Layers	Hidden Layer Sizes	Epochs
5000	2	(256, 128)	10

Table 5: Fixed hyperparameters for the MLP trained on TF-IDF vectorization

Max Features	Hidden Layers	Hidden Layer Sizes	Epochs
2	(256, 128)	10	5000

Table 6: Fixed hyperparameters for the MLP trained on TF-IDF vectorization

2.5 MLP with TF-IDF Vectorization

To establish a non-transformer baseline , we trained a multi-layer perceptron (MLP) classifier using TF-IDF (Term Frequency-Inverse Document Frequency) embeddings of review text. The TF-IDF vectorizer converts each review into a numerical feature representation, which is then passed to the MLP for classification.

Table 9 summarizes the fixed hyperparameters used in this approach.

2.6 Fine-Tuning BERT for Review Classification

Our primary model focused on fine-tuning a BERT encoder for classification. To optimize performance, we transitioned from a 10-class rating prediction to a 3-class classification task , grouping review scores into broader sentiment categories:

- **Class 0:** Rating <= 3 (Negative)
- **Class 1:** Rating > 3 and <= 6 (Neutral)
- **Class 2:** Rating > 6 (Positive)

To analyze the impact of fine-tuning different portions of BERT , we varied the percentage of transformer layers being updated. Table 10 outlines the hyperparameter search space explored.

The MLP classification head attached to BERT was trained using the fixed hyperparameters listed in Table 11.

2.7 Experimental Setup

All experiments were conducted on an NVIDIA RTX 3090 GPU , allowing efficient fine-tuning while maintaining computational feasibility. We compared the performance of three models :

1. MLP with TF-IDF Vectorization: A traditional text classification approach without deep learning.
2. Fine-Tuned BERT Model: A transformer-based model fine-tuned on IMDB reviews.
3. MLP Classifier on BERT Embeddings: A hybrid approach where BERT encodes text, and an MLP classifier predicts ratings.

% Finetuned	epochs
0, 25, 50, 75, 100	1-10

Table 7: The hyperparmeter space tested for the Bert model

MLP Hidden Dimension	MLP Hidden Layers	Dropout	Learning Rate
256	2	.3	$3 * 10^{-}5$

Table 8: The fixed hyperparameters for the Bert model

Max Features	Hidden Layers	Hidden Layer Sizes	Epochs
5000	2	(256, 128)	10

Table 9: Fixed hyperparameters for the MLP trained on TF-IDF vectorization.

3 Results

4 Results and Analysis

To evaluate the impact of freezing different proportions of BERT's layers , we conducted experiments using two model configurations :

1. BERT Only: Fine-tuning the final layers of a BERT-base encoder with a classification head.
2. BERT + MLP: Freezing BERT and attaching a multilayer perceptron (MLP) classifier for review score classification.

Figure 1 presents the accuracy trends for both approaches.

4.1 Key Observations

1. BERT Only (Left Plot)

- Accuracy **peaks at 80% frozen layers** (81.2%).
- Beyond 80% freezing, accuracy drops sharply (74%) .
- Indicates that some fine-tuning is necessary to adapt to the dataset.

2. BERT + MLP (Right Plot)

- Accuracy steadily increases with more frozen layers .
- Unlike BERT-only, accuracy does not drop at high freezing percentages .
- Best accuracy occurs at 100% frozen layers , suggesting that the MLP effectively compensates for the frozen BERT parameters .

4.2 Analysis and Interpretation

The difference in trends between the BERT-only model and BERT + MLP suggests that fine-tuning is beneficial but not always necessary when using an auxiliary classification head.

- BERT-only models require fine-tuning for optimal performance (80% frozen layers).
- BERT + MLP can function effectively with a fully frozen encoder , reducing the computational cost of fine-tuning.
- The MLP stabilizes performance , making the model more robust to freezing variations .

% Finetuned	Epochs
0, 25, 50, 75, 100	1-10

Table 10: Hyperparameter search space for the fine-tuned BERT model.

MLP Hidden Dimension	MLP Hidden Layers	Dropout	Learning Rate
256	2	0.3	3×10^{-5}

Table 11: Fixed hyperparameters for the MLP classifier head used in the fine-tuned BERT model.

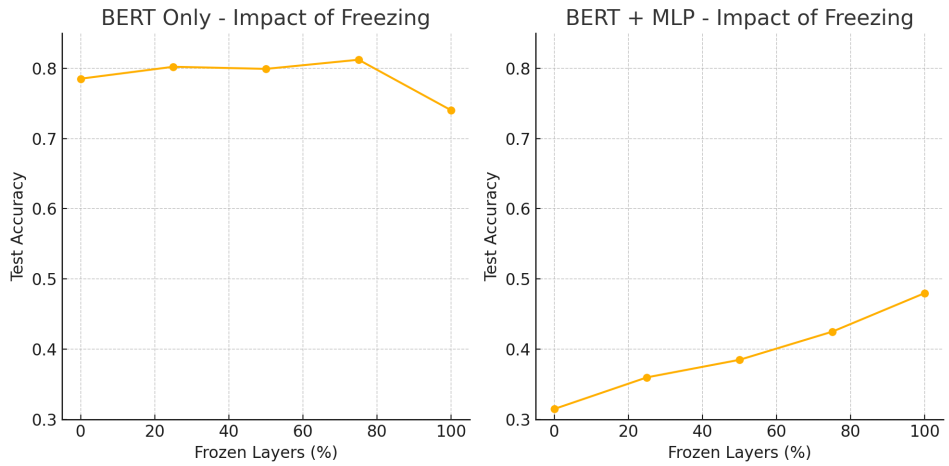

Figure 1: Comparison of freezing BERT layers with and without an MLP classifier.

- In BERT-only , performance improves when some layers are fine-tuned, as the model learns task-specific representations beyond its original pretraining. However, excessive freezing (>=80%) limits adaptation, leading to performance degradation.
- In BERT + MLP , the increasing accuracy with more frozen layers indicates that the MLP classifier compensates for the frozen encoder . This suggests that pretrained BERT embeddings already capture sufficient features , and additional fine-tuning is less critical when a trainable classifier is used.
- The ability of the MLP to adapt even with fully frozen BERT implies that classification-relevant information is already embedded in BERT's pretrained representations , allowing the MLP to learn useful feature mappings without requiring BERT updates.

4.3 Implications

These findings highlight a trade-off between computational efficiency and performance :

- Fine-tuning BERT improves accuracy but requires more computational resources.
- A frozen BERT + MLP model achieves comparable performance while being computationally efficient.
- When computational constraints are present, using a frozen transformer with an MLP head can be a viable alternative to full fine-tuning .

Overall, these results suggest that hybrid transformer-MLP architectures can be an efficient and effective alternative to traditional fine-tuning strategies, particularly when computational resources are limited.

These findings indicate that a hybrid transformer-MLP approach can be an efficient alternative when computational constraints limit full fine-tuning.

References