Machine Learning Final Project

Shashank Santosh Jagtap
UIN: 534002338

Abstract

The project focesed on sentiment analysis using a subset of the Yelp review dataset, comprising around 174,000 reviews. The primary objective was to implement a Transformer model, a state-of-the-art architecture, for sentiment analysis based on textual reviews and their corresponding star ratings. The project encompassed crucial stages, including data pre-processing, input data preparation, and Transformer implementation. We used pre-trained BERT Model with some modifications in its architecture to fulfill our objective. The model was trained using stochastic gradient descent, and the training curve was analyzed to understand the learning dynamics. We achieved the best model with validation accuracy of 85.84%, which was saved for subsequent evaluation on the test set. After model evaluation on testing data we got testing accuracy of 82%. Through these steps, we developed an effective sentiment analysis solution with practical insights into model training and evaluation nuances.

1 Introduction

1.1 Given Problem

We were provided with two essential CSV files: one for training data and another for testing data. The objective was to construct a robust classification model leveraging transformer architecture for effective sentiment analysis.

1.2 Theory

1.2.1 Stop-words

Stop words refer to common words that are often considered to be of little value in text analysis due to their high frequency of occurrence across various documents. These words, such as articles, prepositions, and conjunctions, are generally necessary for constructing grammatically correct sentences but often do not carry significant meaning on their own in the context of document analysis or information retrieval. Common examples: ’a’, ’an’, ’the’, ’and’, ’or’ and so on.

1.2.2 Vocabulary

The vocabulary refers to the set of unique words that occurs in a given corpus or dataset. It plays a crucial role in understanding and processing textual data. The vocabulary is essentially a collection of all distinct words present in the text, and it is important for various NLP tasks, including sentiment analysis.

The vocabulary serves as the foundation for feature extraction, allowing the model to convert raw text into meaningful numerical representations that capture semantic information.

1.2.3 Attention

Attention is a mechanism in machine learning that allows models to selectively focus on different parts of input data when making predictions or processing information. In the context of natural language processing (NLP), attention enables models to assign varying levels of importance to different words in a sequence, allowing the model to weigh the relevance of each word concerning a specific task.

1.2.4 Self Attention

Self-attention is a specific type of attention mechanism where the input sequence is processed against itself. In other words, for each element in the sequence, the model computes attention scores with respect to all other elements in the same sequence. This allows the model to capture relationships and dependencies within the sequence, considering the importance of each word in the context of the entire sequence.[1]

1.2.5 Transformer

Transformers leverage self-attention mechanisms to process input sequences efficiently. Instead of relying on recurrent or convolutional layers, transformers process the entire input sequence simultaneously, making them highly parallelizable and suitable for capturing long-range dependencies in data.[3]

The key components of transformers include self-attention mechanisms, multi-head attention, positional embeddings, and feedforward neural networks. By stacking multiple layers of these components, transformers can learn complex patterns and relationships within sequences, making them particularly effective for natural language processing tasks. The architecture's scalability has contributed to its widespread adoption in various state-of-the-art models for machine translation, sentiment analysis, text generation, and other NLP tasks. The use of self-attention allows transformers to efficiently capture contextual information and dependencies across different positions in the input sequence, making them versatile and powerful in handling sequential data.[6]

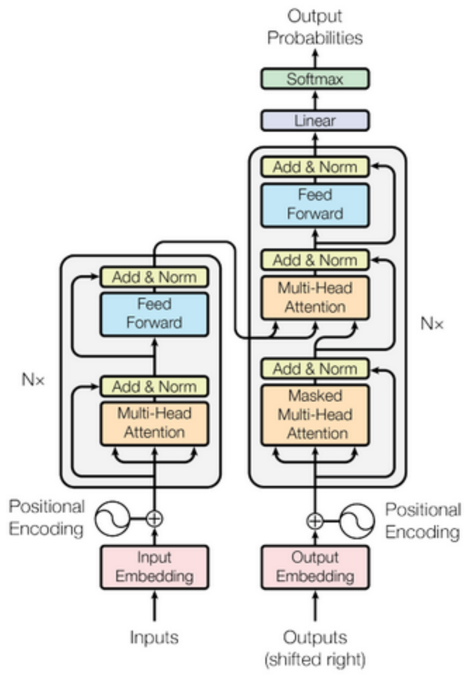


Figure 1: Transformer Architecture

1.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) [5] is a state-of-the-art natural language processing (NLP) model introduced by Google in 2018. What sets BERT apart is its ability to capture contextual information from both directions (left-to-right and right-to-left) in a given sequence of words. Unlike traditional language models that process text in a unidirectional manner, BERT utilizes a transformer-based architecture that facilitates bidirectional learning, enabling it to grasp the intricate relationships and nuances between words in a sentence. The pre-training process involves exposure to massive amounts of textual data, allowing BERT to learn robust contextual representations, which are then fine-tuned for specific downstream NLP tasks, such as sentiment analysis [4], named entity recognition, and question answering.

BERT's architecture consists of multiple layers of self-attention mechanisms, a key component of transformers.[2] The model comprises an encoder stack, where each layer contains a multi-head self-attention mechanism and feedforward neural networks. BERT employs attention masks to focus on relevant parts of the input sequence, capturing dependencies and relationships between words. Additionally, positional embeddings are incorporated to consider the order of words in a sequence. BERT's bidirectional training approach and transformer architecture contribute to its exceptional performance on a wide range of NLP benchmarks, making it a pivotal advancement in natural language understanding.

2 Methodology

2.1 Data importing

In this phase, I imported the training and testing datasets, both provided in CSV format. Each dataset consists of two columns: 'text' and 'stars.' The 'text' column contains the textual content, representing the reviews, while the 'stars' column holds the corresponding star ratings associated with each review. These datasets serve as the foundation for constructing and training a sentiment analysis model using the Transformer architecture.

2.2 Data Pre-processing

I conducted three key preprocessing steps on the training data. Initially, I transformed all text to lowercase to ensure uniformity in the dataset. Subsequently, I removed punctuations to streamline the text for further analysis. Leveraging the 'nltk' library, I also implemented a stopwords removal process as part of the preprocessing steps. Additionally, I introduced a new column labeled 'sentiment' wherein each data point is categorized as positive if the associated star rating exceeds 3, neutral if it equals 3, and negative if the rating is less than or equal to 2. These preprocessing measures are pivotal in refining the quality of the training data and establishing a solid foundation for subsequent tasks, such as the implementation of a sentiment analysis model using the Transformer architecture.

2.3 Input data preparation

In this section, several crucial steps are undertaken to prepare the text data from the 'text' column of the training dataset for machine learning model input. Initially, a process known as tokenization is applied, wherein each text undergoes transformation into a list of its constituent words. This step is fundamental for breaking down the textual information into manageable units, facilitating subsequent analysis.

Following tokenization, a vocabulary is meticulously constructed from the tokenized texts. This vocabulary plays a pivotal role in the conversion of words to integers, as each unique word is assigned a distinct integer index.

Subsequently, the tokenized texts undergo conversion into sequences of integer indices utilizing the established vocabulary. In this process, every word within the tokenized texts is substituted with its corresponding integer index, aligning the textual information with numerical values essential for model comprehension.

To address the varying lengths of sequences, a further step involves padding the sequences of integer indices to a fixed length. This is a critical measure, ensuring uniformity across all sequences and meeting the length criteria necessary for many machine learning models. Padding involves appending zeros to the end of shorter sequences, aligning them with the desired fixed length.

Overall, these pre-processing steps collectively shape the raw text data into a structured and standardized format suitable for input into machine learning models, particularly those based on transformer architectures.

Additionally, I created a column containing integer representations of sentiments ('Positive' as 2, 'Negative' as 0, 'Neutral' as 1) for each corresponding text entry in the training data.

2.4 Transformer Implementation

Firstly, we defined a PyTorch dataset (ReviewDataset) and data loaders for a sentiment analysis task using the BERT (Bidirectional Encoder Representations from Transformers) model. It employs the 'bert-base-cased' pre-trained model and tokenizer. The dataset is structured to include tokenized and encoded reviews along with their corresponding sentiment labels. The dataset is 80-20 split into training and validation sets using the training DataFrame. The data loaders are created for training, validation, and testing sets, enabling efficient batch processing during model training and evaluation. Each data batch consists of tokenized input reviews, attention masks, and the corresponding sentiment labels encoded as tensors. This setup establishes a streamlined pipeline for preparing and loading the sentiment analysis data, facilitating the training and evaluation of a BERT-based model on the sentiment task. We had to repeat this process as BERT requires data preprocessed according to its standards using its own functions.[2]

Following it we defined a sentiment classification model using the BERT architecture for a specific sentiment task. The BertModel.from pretrained function initializes the BERT model with pre-trained weights, and the sentiment classifier is implemented as a PyTorch neural network module (SentimentClassifier). The first three layers of the BERT model are frozen (their parameters are set to not require gradients), and the remaining layers are used for fine-tuning on the sentiment task.

The model consists of a dropout layer (Dropout), followed by three linear layers (Linear). The first linear layer reduces the hidden size of the BERT output to 128, the second linear layer further reduces it to 64, and the final linear layer outputs the predicted sentiment classes. Rectified Linear Unit (ReLU) activation functions are used between the linear layers to introduce non-linearity. The forward method defines the forward pass of the model, taking input IDs and attention masks as inputs and returning the model's output.

This architecture is designed to extract features from BERT's contextualized embeddings and map them to the desired sentiment classes. The model is instantiated with three output classes (for positive, negative, and neutral sentiments) and is moved to the specified device (e.g., GPU) for training and evaluation. The intention is to fine-tune the BERT model for sentiment analysis using the specified architecture and frozen layers.

2.5 Model Training

In this section, we conducted the training of the sentiment classification model over a span of 5 epochs. The optimization process employed stochastic gradient descent (SGD) with a mini-batch strategy. For the SGD, a learning rate of 0.01 and a momentum value of 0.9 were set, influencing the update mechanism of the model parameters during training. Throughout each epoch, the validation accuracy was monitored, and the model achieving the highest validation accuracy was saved in a .bin file, ensuring the preservation of the best-performing model.

To comprehend the evolution of model accuracy across epochs, a graph was generated with the x-axis representing the epochs and the y-axis illustrating both training and validation accuracies. This visualization serves as a valuable tool for assessing the model's learning progress over successive epochs. The meticulous tracking of validation accuracy aids in capturing the model's generalization performance, ultimately guiding the selection of the most optimal model for subsequent evaluation on unseen data.

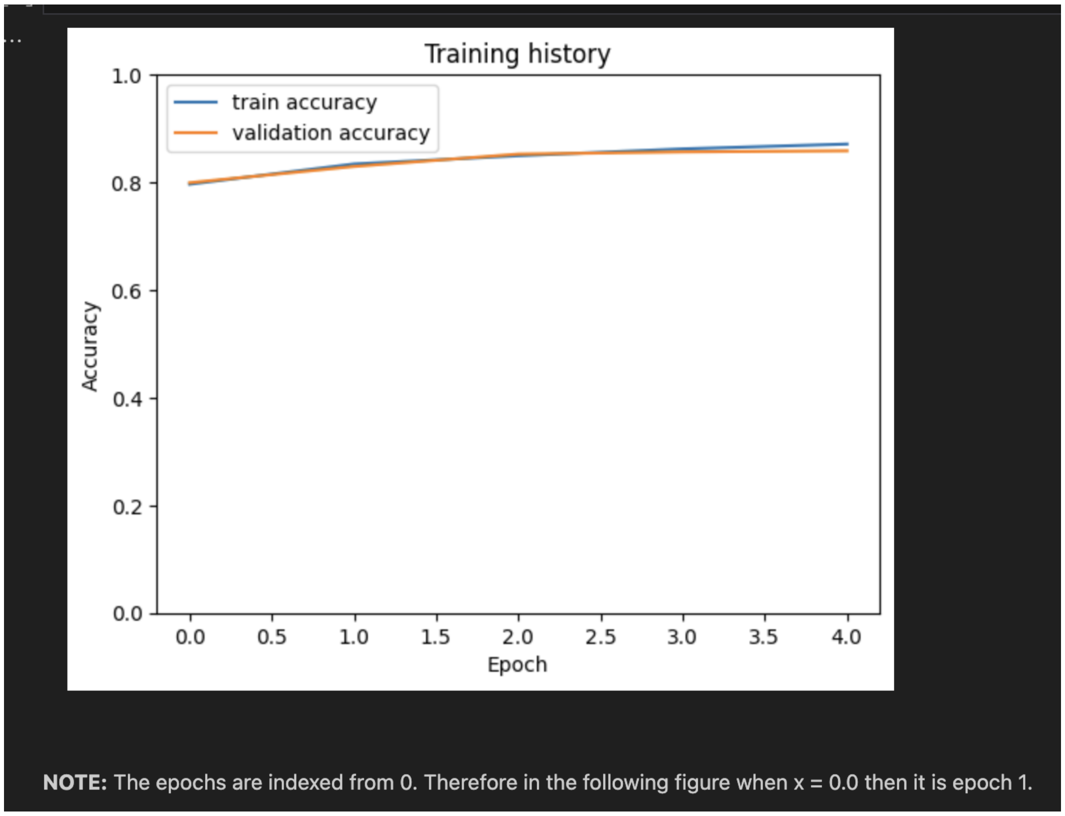
2.6 Result Analysis

In the concluding phase, the previously saved best-performing model was loaded, and the testing data was subjected to the model for evaluation. The accuracy of the model on the testing dataset was calculated, providing insights into its performance on previously unseen examples. Furthermore, to gain a comprehensive understanding of the model's efficacy, a classification report and a confusion matrix were generated.

3 Results

3.1 Model Training

The following graph showcase the training and validation accuracy during the model training process over 5 epochs.

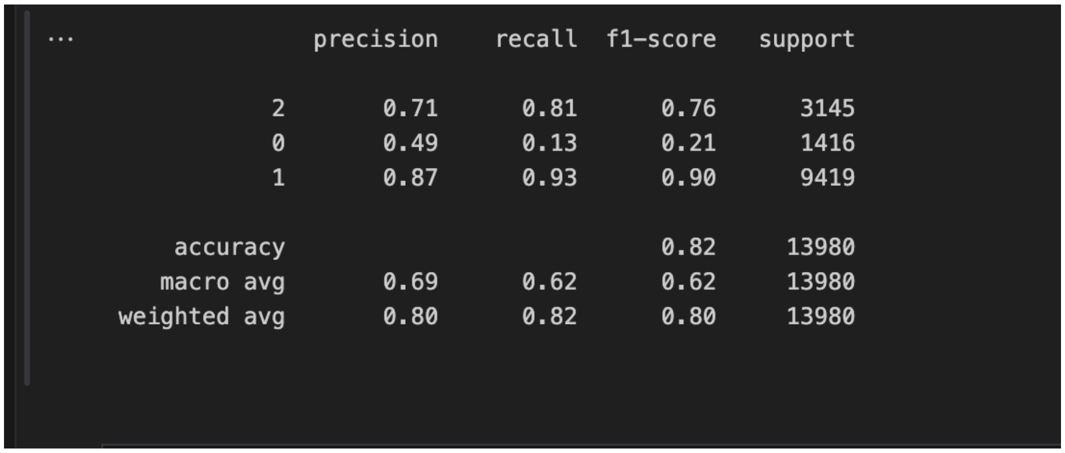


NOTE: The epochs are indexed from 0. Therefore in the following figure when x = 0.0 then it is epoch 1.

Figure 2: Training and Validation Accuracies

The observation reveals that the validation accuracy stabilizes around the 4th epoch, while the training accuracy continues to rise. This indicates a phenomenon known as overfitting, where the model becomes excessively tailored to the training data, hindering its generalization to new examples. Consequently, it is prudent to cease model training after the 5th epoch, striking a balance between capturing relevant patterns in the training data and avoiding overfitting, thus optimizing the model's performance on unseen datasets.

3.2 Model Evaluation



```
...              precision    recall  f1-score   support

           2       0.71      0.81      0.76      3145
           0       0.49      0.13      0.21      1416
           1       0.87      0.93      0.90      9419

    accuracy                           0.82     13980
   macro avg       0.69      0.62      0.62     13980
weighted avg       0.80      0.82      0.80     13980
```

Figure 3: Classification Report

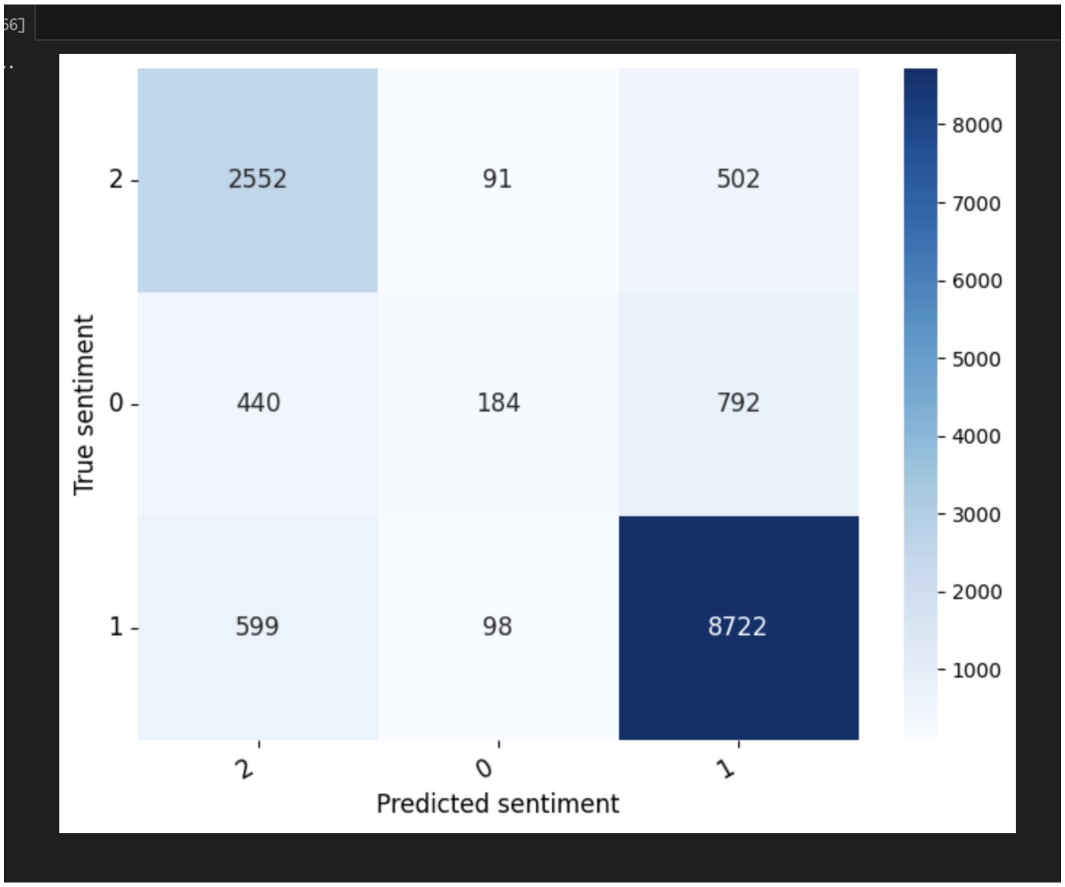


Figure 4: Confusion Matrix

Once the model is trained we can look at the following images to analyse the model performance on testing data.

The model achieved a testing accuracy of 82%, indicating that it correctly predicted the sentiment labels for a significant portion of the testing dataset. This metric is a measure of the model's overall correctness in classifying sentiments—ranging from 'Positive' to 'Negative' and 'Neutral'.

We know that the sentiment labels are mapped as follows: 'Positive' - 2, 'Negative' - 0, 'Neutral' - 1. Notably, our model exhibits strong performance in predicting 'Neutral' and 'Positive' sentiments. However, it struggles with accurate predictions for the 'Negative' sentiment class.

Despite achieving a respectable validation accuracy during model construction, the notable discrepancy in the 'Negative' sentiment predictions suggests a potential imbalance in the label distribution within the training data. It's plausible that the training dataset may have contained a higher proportion of 'Positive' and 'Neutral' labels compared to 'Negative' labels.

As a consequence, this imbalance is reflected in the lower f1-scores for 'Negative' sentiments compared to 'Positive' and 'Neutral' sentiments.

4 Conclusion

In summary, we successfully implemented a Transformer-based sentiment analysis model, specifically leveraging the BERT architecture, on a subset of the Yelp review dataset. Through meticulous data preprocessing, the training data was refined to enhance model performance. The Transformer model, trained over five epochs using stochastic gradient descent, demonstrated strong generalization capabilities with a validation accuracy of 85.84%. However, challenges were identified in accurately predicting 'Negative' sentiments, attributed to potential class imbalances in the training data. Finally we achieved 82% of testing accuracy.

The utilization of advanced Transformer architectures, exemplified by BERT, showcased the model's proficiency in capturing contextual information for sentiment analysis tasks. The findings underscore the significance of balanced datasets and thoughtful preprocessing in achieving optimal performance. Overall, this project contributes valuable insights into the intricacies of sentiment analysis model development and highlights the transformative potential of state-of-the-art Transformer architectures in natural language processing applications.

References

[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[2] Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. Transformer models for text-based emotion detection: a review of bert-based approaches. *Artificial Intelligence Review*, pages 1–41, 2021.

[3] Adrian MP Braşoveanu and Răzvan Andonie. Visualizing transformers for nlp: a brief survey. In *2020 24th International Conference Information Visualisation (IV)*, pages 270–279. IEEE, 2020.

[4] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. Aspect-based sentiment analysis using bert. In *Proceedings of the 22nd nordic conference on computational linguistics*, pages 187–196, 2019.

[5] Sudharsan Ravichandiran. *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing Ltd, 2021.

[6] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.