Transformer model for sentiment analysis based on the text review and stars

Manoj Kumar Reddy Peta
Email: manoj.peta@tamu.edu
Department of Computer Science and Engineering
Texas A&M University
College Station

***Abstract*—This project explores the effectiveness of Transformer models in the task of sentiment analysis, a key component in interpreting the vast amounts of subjective data generated across digital platforms. In this report, we detail the development and implementation of a Transformer-based model that utilizes Positional Embedding and Transformer Encoder layers, followed by Global Average Pooling and Dropout layers, to accurately classify text into sentiment categories. Experiments involving different hyperparameter settings are carried out to identify the optimal model configuration. The results showcase the model's proficiency in accurately categorizing sentiment.**

I. Introduction

Deep learning has significantly upgraded the tools we use for analyzing emotions in text, such as reviews or tweets. Earlier techniques needed a lot of manual effort and were not very good at picking up on the subtle ways language can vary. Now, with models like LSTMs and CNNs, computers are getting better at recognizing feelings in text just by looking at lots of examples.

The Transformer model has brought a new wave of innovation to the field of natural language processing, thanks to its unique self-attention feature. This model stands out because it can look at all parts of a sentence at the same time, not just one word after another. This not only speeds up how fast it can learn but also helps it understand the role and relevance of each word in a sentence, no matter where the word appears.

This project delves into the exploration and application of Transformer models for sentiment analysis. We examine various configurations of the Transformer architecture, experimenting with different numbers of layers, embedding dimensions, and attention heads to evaluate their impact on the model's performance through rigorous testing across multiple datasets.

II. Related Work and Review of State of Art

The groundbreaking study by Vaswani and colleagues, referred to in their work "Attention Is All You Need," [1] introduced the innovative Transformer architecture, significantly altering the course of natural language processing (NLP). This design moves away from the traditional recurrent layers, adopting instead a self-attention mechanism that assesses the relevance of various segments within the input. It allows simultaneous data processing, the Transformer architecture has greatly improved the ability to perform text classification and sentiment analysis by capturing complex dependencies and subtle context within the text.

Another work, [2], Fundamentals of RNN and LSTM: Recurrent Neural Networks (RNNs), including their Long Short-Term Memory (LSTM) variants, have been foundational in NLP for their ability to process sequential data and capture dependencies. Despite their success, RNNs and LSTMs often struggle with long-range dependencies, a limitation addressed by the Transformer architecture.

Another related work [3], Advancements in Deep Learning Models for Text Classification: A comprehensive review of over 150 deep learning models for text classification reveals the evolution from simple feed-forward networks to more complex architectures like RNNs, CNNs, and hybrid models. This progression underscores the increasing sophistication in handling diverse aspects of text data.

Transformer Variants and Their Applications: The original Transformer model has facilitated numerous variants, each improving or adapting the architecture for specific tasks or efficiency gains. These variants demonstrate the flexibility and adaptability of the Transformer. This is clearly discussed in [4]

The current state of the art in text classification and sentiment analysis is dominated by Transformer-based models. Large-scale pre-trained models, such as GPT-3 and others with billions of parameters, have shown that increasing model size can lead to performance gains across many tasks. These models have been adapted to domain-specific applications, providing significant improvements in fields such as finance, legal, and biomedical. Transformer models now support various adaptation methods like fine-tuning and prompt-tuning, which allow them to achieve state-of-the-art performance on diverse NLP tasks, including sentiment analysis.

Models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) use vast amounts of data and compute power to learn rich representations of language. XLNet further expanded upon this by better learning bidirectional relationships within sequences. The research continues on models like T5, BART, and PEGASUS to set new benchmarks by treating all NLP tasks as text-to-text problems. These models, along with others like RoBERTa and ELECTRA, represent the cutting edge of NLP technology.

III. Dataset

For the sentiment analysis, a subset of the Yelp review dataset is used. The dataset comprises approximately 174,757 reviews for the training data. A validation split of 20% is made. Similarly, a subset of the Yelp review dataset is choosen for testing. The testing set consists of 13,980 instances. Table I encapsulates the dimensions of each subset.

TABLE I
Summary of Dataset Partitions

Partition	**Size**
Training Set	139,806
Validation Set	34,951
Testing Set	13,980

A. Review Text

The dataset contains a 'reviews' column that includes the text of each review. These reviews vary in length and detail, offering a wide range of language use and sentiment expression. The average review is 356.58 tokens long, while the longest review is 3,619 tokens.

B. Star Ratings

Each review is paired with a 'stars' column, which assigns a rating from 1 to 5 stars. These ratings reflect the sentiment of the review, where a 1-star rating indicates a negative experience and a 5-star rating indicates a highly positive experience.

IV. Proposed Methodology

This model is built primarily using resources [1] and [5]. The basic idea regarding the architecture is taken from [1]. The solution involves data preprocessing, input data preparation, transformer implementation, and training.

A. Data Preprocessing

Data preprocessing is a crucial step in any machine learning pipeline. In this study, we focused on two key aspects of preprocessing: converting stars to sentiment and text normalization. Fig 1 and Fig 2 represent the distribution of data before preprocessing for the train and test datasets, respectively.

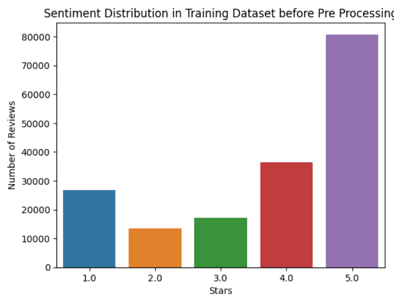

Fig. 1. Sentiment Distribution in Training Dataset before Pre Processing

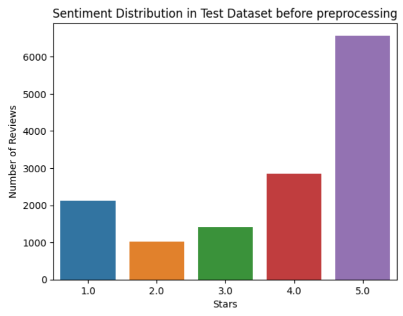

Fig. 2. Sentiment Distribution in Test Dataset before Pre Processing

1) Converting stars to sentiment

The dataset initially contains two features: 'review' (textual reviews) and 'stars' (numerical rating). The 'stars' feature, representing user ratings, was converted into a categorical sentiment label. This conversion is vital for sentiment analysis, allowing the model to categorize reviews more intuitively. The conversion logic is as follows:

- Ratings above 3 stars are labeled as 'Positive'.
- Ratings of 2 stars or less are labeled as 'Negative'.
- Ratings of 3 stars are considered 'Neutral'.

This categorization simplifies the output space of the model, aligning it more closely with typical sentiment analysis tasks.

2) Text Normalization

The 'review' texts underwent several normalization steps:

1) **Punctuation Removal**: Non-word characters (punctuation) were removed.
2) **Lowercasing**: All text was converted to lowercase to ensure uniformity, as the case of letters does not usually affect the sentiment of the text.
3) **Stop Word Removal**: Commonly used words (stop words) that do not contribute significantly to the sentiment of the text were removed. This step was performed using the NLTK library, which provides a comprehensive list of English stop words. The removal of stop words helps in reducing the dimensionality of the data and focuses the analysis on the words that carry the most sentiment.

The result of this preprocessing is a cleaner, more concise dataset that is more suitable for sentiment analysis.

3) Label Encoding

The transformed sentiment labels were then encoded into a numerical format suitable for machine learning algorithms. This encoding was performed using the `LabelEncoder` from Scikit-learn, which converts categorical labels into a numeric array. Label encoding is an essential step in preparing the target variable for classification tasks.

Fig 3 and Fig 4 represent the distribution of data before preprocessing for the train and test datasets, respectively.

In summary, these preprocessing steps collectively refine the dataset into a form that is more amenable for subsequent

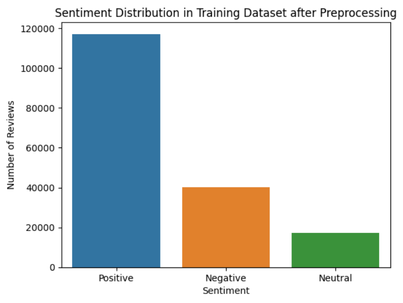

Fig. 3. Sentiment Distribution in Training Dataset after Pre Processing

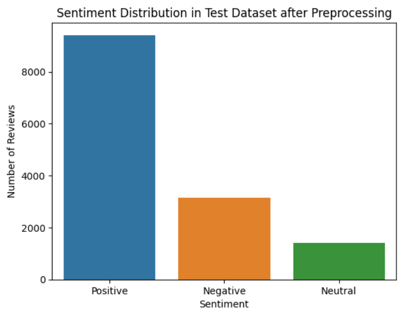

Fig. 4. Sentiment Distribution in Testing Dataset after Pre Processing

analysis, reducing noise and standardizing the input features for effective learning.

B. Input Data Preparation

Following the preprocessing of the dataset, the next critical step was preparing the input data for the machine learning model. This step involved analyzing the text data's characteristics and transforming it into a suitable format for model training.

1) Analysis of Review Lengths

Initially, I analyzed the lengths of the reviews in our dataset. This analysis included calculating the average, minimum, and maximum lengths of the reviews, measured in terms of the number of characters. These metrics provide insights into the variability and distribution of content within our dataset. The average length was found to be 356.48, with a minimum and maximum length of 6 and 3612 respectively. From these, I choose the maximum sequence length to be 500.

2) Word Count Distribution

A key aspect of understanding the textual data involved analyzing the frequency of each word across the dataset. To accomplish this, we implemented a function to count the occurrences of each word. This analysis was crucial not only to identify the most commonly used words but also to inform the creation of a dictionary representing our vocabulary. From this analysis, we identified a total of 148,259 distinct words in the dataset.

Given the extensive size of the vocabulary, we faced a decision regarding the optimal number of words to include in our model's dictionary. To balance the richness of the linguistic information with the computational efficiency, we chose to limit the vocabulary size of the dictionary to 30,000. The dictionary creation thus played a pivotal role in shaping our text vectorization process and the subsequent modeling steps.

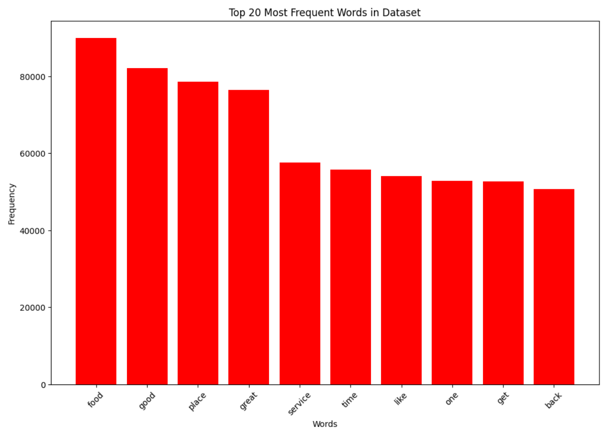

Fig. 5. Top 10 frequent words in the vocabulary dictionary

3) Text Vectorization

The final step in preparing the input data was vectorizing the text. I utilized the `TextVectorization` layer from TensorFlow's Keras API for this purpose. The configuration for the vectorization was as follows:

- **Maximum Tokens**: The maximum size of the vocabulary was set to 30,000.
- **Output Mode**: The output of the vectorization was configured to integer encoding.
- **Output Sequence Length**: Each input text was standardized to a length of 500, truncating or padding as necessary.
- **Standardization**: No extra standardization was applied during vectorization, as the text had already undergone preprocessing.

This vectorization process transformed the textual data into a numerical format, making it suitable for input into our machine learning model.

These preparation steps were crucial for ensuring that the data fed into the model was in an optimal format, balancing the need for information richness and computational efficiency.

C. Transformer implementation

The core of our approach leverages a Transformer architecture, renowned for its effectiveness in handling sequential data, particularly in natural language processing tasks. This subsection details the implementation of our Transformer model, following the input data preparation phase. Fig 6 shows the basic transformer model.

1) Positional Embedding Layer

The first layer in our model is a custom 'PositionalEmbedding' layer. This is a custom layer and it performs word

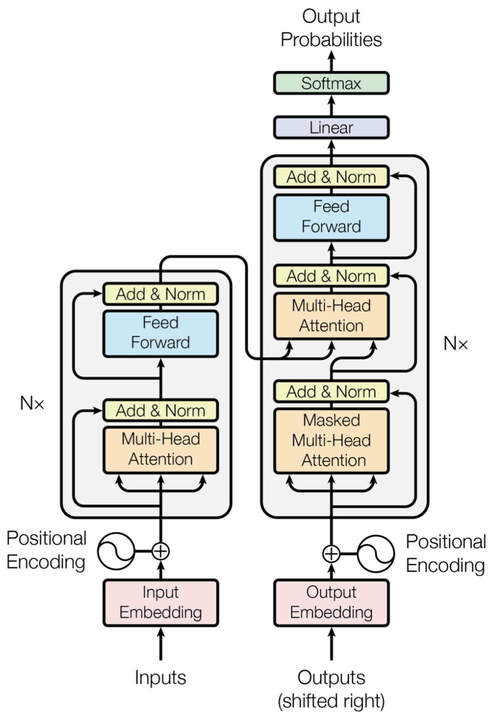

Fig. 6. Transformer Model

embedding and position encoding. This layer serves two primary functions:

- It converts input tokens (words or subwords) into dense vectors of a specified size (‘embed_dim‘), effectively capturing the semantic meaning of each token. This is called input embedding.
- It adds positional information to these token embeddings, essential for maintaining the sequence order, as the Transformer architecture does not inherently capture sequential data’s temporal aspects. This is called positional embedding.

This layer outputs a combination of token and position embeddings, forming the initial representation of the input text.

2) Transformer Encoder Layer

The main building block of our model is a custom ‘TransformerEncoder‘ layer, structured as follows:

- A Multi-Head Attention mechanism, crucial for the Transformer model, allows the model to focus on different parts of the input sequence simultaneously. Fig 9 shows the basic multihead attention mechanism.
  The attention mechanism can be described by the equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

  where Q, K, and V are matrices representing queries, keys, and values, respectively. d_k represents the dimensionality of the keys.

  The multi-head attention mechanism can be described by the equation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \quad (2)$$

  where each head_i is computed as follows:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

  In this context, W_i^Q, W_i^K, and W_i^V are parameter matrices for the i-th attention head, and W^O is the parameter matrix that combines the heads’ outputs.

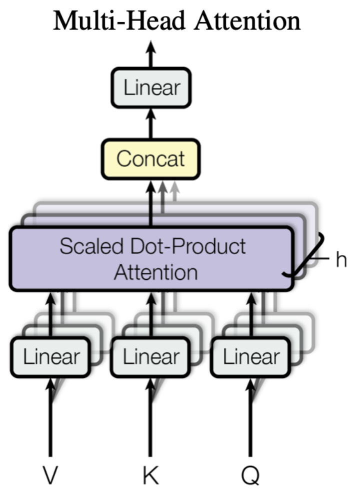

Fig. 7. Multihead Attention Mechanism

- A feed-forward neural network, implemented as a sequential Keras model with two dense layers, enhances the model’s capability to capture complex relationships in the data.
- Layer Normalization steps, applied after both the attention mechanism and the dense layers, aid in stabilizing the learning process and improving convergence.
- The use of residual connections, a standard practice in deep learning architectures, to facilitate the flow of gradients during training and enable deeper models.

3) Model Architecture and Training

The architecture of the model can be found in Fig 8. The overall architecture of our Transformer model is as follows:

- The input text, represented as sequences of integers, passes through the ‘PositionalEmbedding‘ layer.
- Multiple ‘TransformerEncoder‘ layers process the embedded input, allowing the model to capture complex patterns and relationships in the data.
- A Global Average Pooling layer follows, reducing the dimensionality and summarizing the important features.
- Dropout layers are applied for regularization, reducing the risk of overfitting.

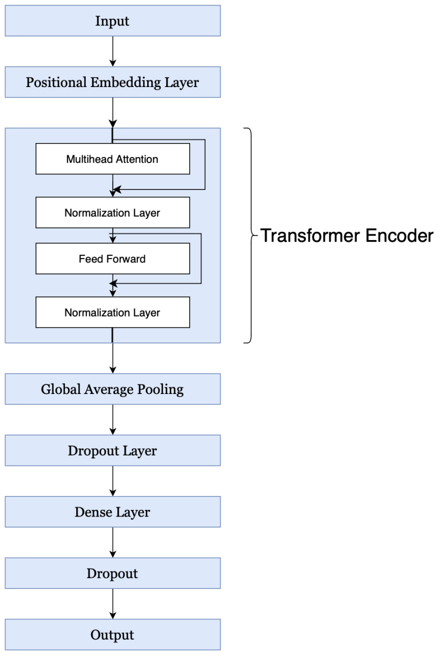

Fig. 8. Transformer architecture

- A final dense layer with a softmax activation function outputs the probabilities for each of the three sentiment classes (Positive, Neutral, Negative).

The model is compiled with the Adam optimizer, using sparse categorical cross-entropy as the loss function, suitable for multi-class classification tasks. The accuracy metric is used for performance evaluation.

This Transformer-based model, with its custom layers and architectural choices, forms the cornerstone of our approach to sentiment analysis, demonstrating robust performance on the tasks at hand.

V. RESULTS

The following section presents the results obtained from the rigorous experimentation of Transformer models using the various configurations detailed earlier. Each configuration is evaluated based on its performance in sentiment analysis on the Yelp dataset.

A. *Configuration 1*

The first configuration, with a single layer, an embedding dimension of 32, two attention heads, a latent dimension of 32, a dropout rate of 0.1, and trained for 20 epochs, represented our baseline model. This setup, while basic, was instrumental in providing a fundamental understanding of the Transformer model's capabilities in sentiment analysis. This basic model achieves an accuracy of 86.66% on the validation data. This model, with the simplest architecture, shows a steady increase in training accuracy over epochs. This shows the power of transformer architecture in sentiment analysis. Results are shown in Fig 9.

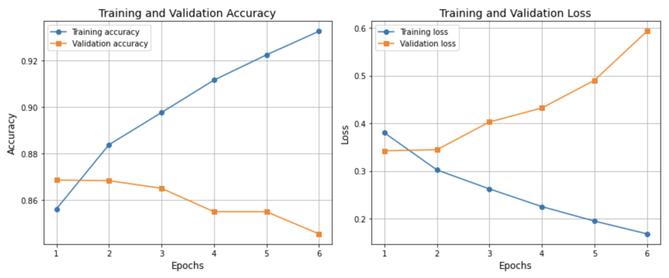

Fig. 9. Accuracy, Loss for Configuration 1

B. *Configuration 2*

The second configuration introduced a moderate increase in complexity with two layers and four attention heads, alongside a latent dimension of 64. This model achieves an accuracy of 86.52% on the validation data. This model too performs well both on training and validation data. However, the training loss decreases more significantly than the validation loss, hinting at the onset of overfitting as the model learns to memorize the training data but does not improve equally on the validation data. It can be seen in Fig 10. But overall best model's accuracy of this configuration is almost similar to that of configuration 1.

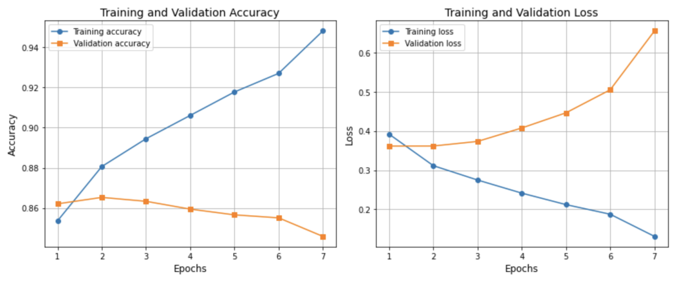

Fig. 10. Accuracy, Loss for Configuration 2

C. *Configuration 3*

With this configuration, the model depth was increased to three layers, embedding dimensions to 128, and attention heads to six. The latent dimension was expanded to 128, with a dropout rate of 0.3. This model achieves an accuracy of 85.39% on the validation data. This configuration shows a similar trend to Configuration 2 but with more fluctuation in the validation metrics. This variability could indicate that the model might benefit from further regularization or that the learning rate needs adjustment. It can be seen in Fig 11.

D. *Configuration 4*

Featuring four layers, eight attention heads, and a latent dimension of 256, this configuration aimed to explore the upper limits of model complexity without extensive computational demands. The embedding dimension was set to 256, and the

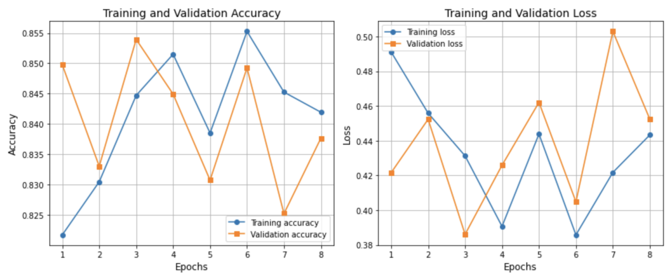

Fig. 11. Accuracy, Loss for Configuration 3

model was trained with a dropout rate of 0.4. This model achieves an accuracy of 67.18% on the validation data. The plots ,in Fig 12, for this configuration show an improvement in training accuracy, but the validation accuracy fluctuates and eventually decreases, which is a strong indication of overfitting. The model's complexity allows it to fit the training data well but not generalize effectively, likely because it starts capturing noise in the training data.

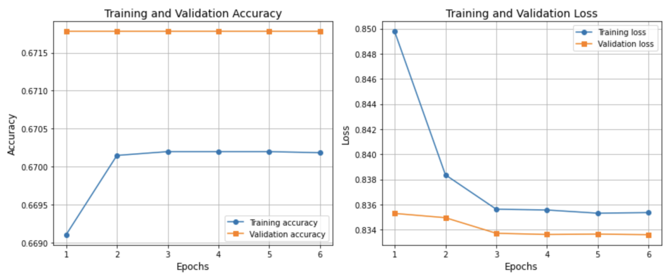

Fig. 12. Accuracy, Loss for Configuration 4

E. Configuration 5

The final configuration pushed the model to its maximum complexity with five layers, ten attention heads, a latent dimension and an embedding dimension of 512, trained for 2 epochs at a dropout rate of 0.5. This model achieves an accuracy of 66.8% on the validation data. Despite being the most complex, does not outperform the others on unseen data. This clearly overfits and the model looks like it is learning unnecessary information and relation between words during the training phase. So, we can observe that as we make the model more complex, i.e. by adding more transformer encoders, multi heads, the model will overfit. That is why the validation accuracy of this model is pretty low.

VI. Results for Test Data

From the results discussed in the previous section, it be seen that configuration 1 performs the best among all the configurations considered. This configuration has 1 transformer encoder layer, an embedding dimension of 32, two attention heads, a latent dimension of 32, a dropout rate of 0.1, and trained for 20 epochs. Upon using this model on testing data, an accuracy of 86.90% is obtained. From the confusion matrix in Fig 14, we can say that,

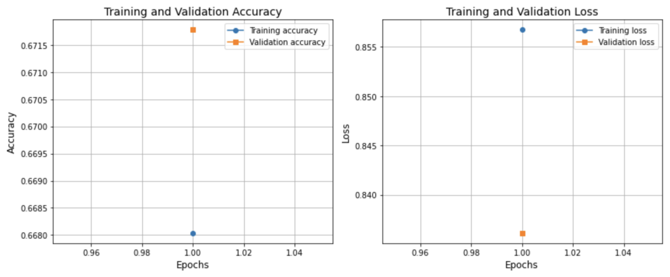

Fig. 13. Accuracy, Loss for Configuration 5

1) Class 2 has a notably high true positive rate, indicating a strong model response to this class.
2) Class 0 and Class 1 with a high frequency of false positives and false negatives, suggesting difficulty in distinguishing between these sentiments.
3) The misclassification between Class 0 and Class 1 suggests that the model may be challenged by the lexical similarities within these classes.

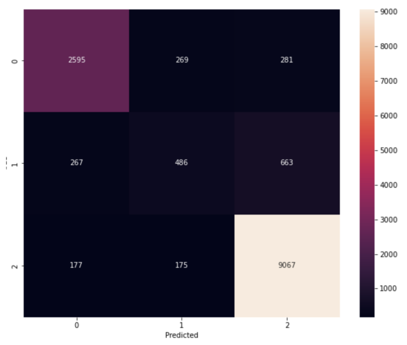

Fig. 14. Confusion Matrix for Test Data

From the confusion matrix in Fig 15, we can say that,

1) Class 0 shows an AUC of 0.97, indicating excellent model discrimination for this class.
2) Class 1 has a lower AUC of 0.87, pointing to a good—but relatively weaker—discrimination ability.
3) Class 2 exhibits a high AUC of 0.96, mirroring the strong performance seen in the confusion matrix.

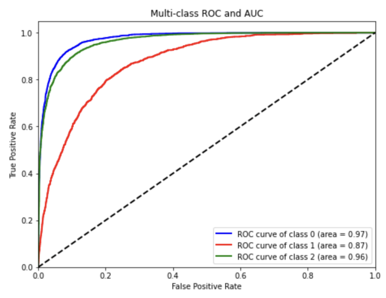

Fig. 15. AUC ROC curve for Test Data

The high AUC values for Classes 0 and 2 reflect strong performance. However, the lower AUC for Class 1 aligns with the confusion matrix, indicating that this class is more challenging for the model to classify correctly.

Overall, the model shows strong discriminative power. The model was able to achieve an overall accuracy of 86.9% on the test data.

VII. Conclusion

The series of experiments conducted with Transformer models on Yelp review data gave a comprehensive understanding of their capabilities for sentiment analysis tasks. The models displayed a solid performance with an overall accuracy of approximately 87% across different sentiment classes. These results validate the effectiveness of Transformer models in interpreting complex language patterns and sentiments in user-generated content. Our findings indicate that while increased model complexity generally leads to improved performance, it also raises the risk of overfitting and necessitates greater computational resources.

References

[1] Attention Is All You Need, arXiv:1706.03762

[2] Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network, arXiv:1808.03314

[3] Deep Learning Based Text Classification: A Comprehensive Review, arXiv:2004.03705

[4] A Survey of Transformers, arXiv:2106.04554

[5] https://www.tensorflow.org/text/tutorials/classify_text_with_bert

[6] The original article on Transformers. https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04.