Predicting Yelp Star Ratings: An Analysis of Different Models and Fine-Tuned RoBERTa Model

Stanford CS224N {Custom} Project

Rishi Alluri
Department of Computer Science
Stanford University
`allurir@stanford.edu`

Upamanyu Dass-Vattam
Department of Bioengineering
Stanford University
`udvattam@stanford.edu`

Abstract

This research presents a systematic review of various language models and data preprocessing techniques used for predicting Yelp star ratings from user reviews. Our investigation covers baseline models such as Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM), which leverage sequential data processing and gating units respectively, to handle the information in the review text. We also dive into the analysis of more complex models like BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa. BERT, with its distinctive attention mechanism and bidirectional training, offers a unique approach to our task. RoBERTa, a variant of BERT, but with altered hyperparameters, employs a Transformer-based architecture and self-attention mechanisms. The focus of our research lies in the adaptation and fine-tuning of the RoBERTa model, built on top of the base RoBERTa model, which is our primary contribution to this field of study. The systematic review and comparative analysis of these models, combined with an exploration of different data preprocessing techniques, have been instrumental in developing our fine-tuned RoBERTa model. Our preliminary findings indicate that the LSTM model considerably outperformed the RNN model, demonstrating test accuracies of 50.02% and 55.33% respectively. Results for BERT and RoBERTa are provided here: BERT had a test accuract of 67.18%, and RoBERTa had a test accuracy of 73.41%. The ultimate goal of this systematic review and comparative study is to identify a model, fine-tuned for the specific task of predicting Yelp star ratings, that can provide reliable and actionable insights to businesses aiming to understand and enhance customer satisfaction.

1 Key Information to include

Mentor: Bessie Zhang, External Collaborators (if you have any): NA, Sharing project: NA

2 Introduction

The digital era has resulted in an unprecedented increase of user-generated content on online platforms. This growing digital footprint can provide valuable insights, particularly for businesses seeking to understand and improve customer satisfaction. The ability to accurately predict user sentiment from this online content is therefore highly important.

The restaurant industry, in particular, stands to benefit immensely from accurate sentiment prediction. People often rely on Yelp reviews to decide where to eat, and a difference of even half a star rating can significantly impact a restaurant's profits. A Harvard Business School study found that a one-star increase in Yelp rating leads to a 5-9% increase in revenue for independent restaurants. Hence, an accurate model for predicting Yelp star ratings from user reviews can provide valuable insights to restaurant owners, enabling them to address customer concerns while enhancing their service quality and business performance (Luca, 2016).

Current sentiment prediction methods are primarily based on machine learning and natural language processing techniques. Among these, deep learning models such as Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) have been applied with notable success. These models can process sequential data and maintain a hidden state that holds information about a sequence, making them suitable for handling the temporal dependencies in review text (Sherstinsky, 2018). However, they may suffer from issues like vanishing or exploding gradients, which can limit their predictive accuracy (Pascanu et al., 2012).

More recently, transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) and its variant, RoBERTa, have been introduced. These models use self-attention mechanisms that allow them to capture dependencies between all words in a sentence, regardless of their distance (Devlin et al., 2018) (Liu et al., 2019).

This study aims to tackle these challenges through a systematic review and comparison of various language models, including RNN, LSTM, BERT, and a fine-tuned RoBERTa model. Our research uses the Yelp dataset, providing a rich corpus of user reviews for our analysis (Yelp, 2022).

This research aims to fill the gaps in the current understanding of sentiment prediction from Yelp reviews, providing businesses with a more reliable and accurate tool to gauge customer satisfaction. The insights derived from this study have the potential to significantly impact business strategies, enabling businesses to more effectively address customer needs.

3 Related Work

Sentiment analysis, a critical field in natural language processing, has seen the evolution and development of diverse techniques over the years. According to Wankhade et al. (2022), these techniques can be broadly categorized into machine learning-based approaches and lexicon-based approaches. Lexicon-based approaches quantify sentiment using predefined lexicons, with each word acting as a feature for sentiment classification. This approach is commonly combined with basic feature extraction techniques such as term frequency, parts of speech tagging, and handling of negations. However, these methods struggle with the subtleties and context-dependency of sentiment in natural language according to Wankhade et al. (2022). Machine learning-based techniques utilize supervised learning models trained on labeled data. Traditional methods, such as Naive Bayes, Support Vector Machines, and Decision Trees, are often limited in capturing complex semantic relationships in text. Deep learning methods like CNNs and RNNs overcome this by extracting features from raw data and handling the sequential nature of text. However, they require substantial computational resources and don't perform well with long-term dependencies (Wankhade et al., 2022).

Furthermore, the emergence of transformer-based models like BERT has marked a significant advancement in the field. Dang et al. (2021) in their research study fine-tuned BERT for sentiment analysis on a large dataset of app reviews, reporting that the fine-tuned BERT model outperformed traditional machine learning models, thereby offering more accurate sentiment prediction. Despite the significant advancements brought by BERT in sentiment analysis, the model has limitations. One issue is that BERT struggles with long texts due to its maximum sequence length limit. Another critical issue is that the fine-tuning process may not fully adapt BERT to specific language nuances and styles in different domains, potentially affecting its performance in domain-specific sentiment analysis tasks.

These limitations present opportunities for improvement, leading to the emergence of enhanced models like RoBERTa. RoBERTa addresses BERT's shortcomings by using a longer training time, larger batch sizes, removing the next sentence prediction objective, which leads to improved performance, and dynamically changing the masking pattern applied to the training data. This provides a strong basis for our project, which aims to leverage the improvements of RoBERTa in the field of sentiment analysis (Liu et al., 2019).

4 Approach

We explore four different models for the problem of rating review classification based on restaurant text reviews. Models include an RNN, LSTM, BERT, and RoBERTa. We additionally, modified the BERT and RoBERTa models by testing the addition of a CNN layer and freezing early layers. See appendix for model architectures 8.

4.1 Models

4.1.1 Baseline Models: RNN and LSTM

The RNN processes sequential data by maintaining a hidden state that holds information about a sequence up to the current step. The hidden state is updated with each step through the sequence (Sherstinsky, 2018). The key equation for an RNN is:

$$h_t = \sigma(W * h_{t-1} + U * x_t) \tag{1}$$

where σ is the activation function, h_{t-1} is the previous hidden state, x_t is the input at time t, and W and U are weight matrices (Sherstinsky, 2018). This simple update rule gives RNNs their characteristic ability to model temporal dependencies in data. However, because the effect of a given input on the hidden states decreases exponentially over time, RNNs tend to forget information from earlier in the sequence, leading to the vanishing gradient problem.

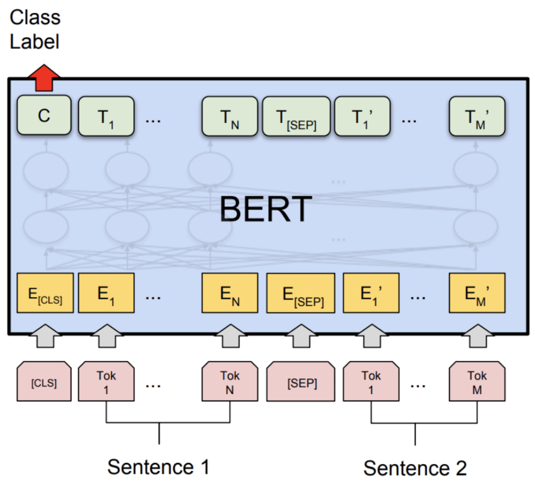
LSTMs are designed to mitigate this issue. They maintain a separate cell state C_t, and use gating units to control the flow of information into and out of this cell state. The equations that determine LSTM's behavior are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$
$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \tag{5}$$
$$o_t = \sigma(W_o \cdot [ht-1, x_t] + b_o) \tag{6}$$
$$h_t = o_t \circ \tanh(C_t) \tag{7}$$

Here, f_t, i_t, o_t, and $\tilde{C}t$ are, respectively, the forget gate's activation vector, the input gate's activation vector, the output gate's activation vector, and a vector of new candidate values for the state, at time t. Each of these vectors is calculated based on the previous hidden state (ht-1) and the current input x_t, with different weight matrices W_f, W_i, W_o, and W_C. The forget gate f_t controls what proportion of the previous cell state C_{t-1} is retained, while the input gate i_t controls what proportion of the new candidate values $\tilde{C}_t$ is added to the cell state. The output gate o_t controls the output h_t which is the final hidden state and is based on the current cell state. Therefore, through this process, an LSTM can control its memory and effectively model long-term dependencies, addressing the key limitations of the simpler RNN. (Sherstinsky, 2018).

4.2 BERT

Google's BERT model diverges from traditional recurrent neural networks (RNNs) by adopting a bidirectional transformer architecture that processes words in relation to all other words in a sequence simultaneously (Devlin et al., 2019). Unlike RNNs, which sequentially process data and maintain hidden states, BERT employs attention mechanisms to capture dependencies between words in both directions within a sequence (Vaswani et al., 2017). The key innovation lies in BERT's transformer blocks, which allow for parallel computation and enable the model to consider the entire context of a word when making predictions (Devlin et al., 2018).



Through self-attention mechanisms, BERT attends to relevant words in a sequence, allowing it to capture long-range dependencies efficiently. Moreover, BERT utilizes multiple layers of transformers, each refining the representation of the input text further. By leveraging this architecture, BERT achieves state-of-the-art performance in various natural language processing tasks, demonstrating its ability to understand nuanced linguistic relationships and contexts effectively (Devlin et al., 2019).

The model utilizes an encoder and decoder stack with scaled multi-head attention:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{8}$$
$$\text{where:} \tag{9}$$
$$head_i = Attention(QW_i^Q, KW_I^K, VW_I^V) \tag{10}$$
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{11}$$
$$W^Q, W^K, W^V, W^O \text{are parameter matrices} \tag{12}$$

Furthermore, each layer in the encoder and decoder contains a fully connected feed-forward network:
$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$ (Vaswani et al., 2017)

4.3 RoBERTa

RoBERTa, an enhanced version of BERT, introduces several key modifications to the original architecture, leading to significant improvements in performance (Liu et al., 2019). Unlike BERT, which masks a portion of the input tokens during pretraining, RoBERTa adopts a more extensive pretraining approach by training on larger datasets and removing the next sentence prediction task (Liu et al., 2019).
Additionally, RoBERTa employs dynamic masking during training, where the masking pattern is randomly sampled for each training instance, allowing the model to adapt better to various input token combinations (Liu et al., 2019). Another crucial difference lies in RoBERTa's training objectives; it utilizes larger batch sizes and longer training sequences, leading to more robust representations (Liu et al., 2019).
Moreover, RoBERTa incorporates Byte-level BPE (Byte Pair Encoding) tokenization, which enhances the model's ability to handle out-of-vocabulary words and improves its generalization capabilities (Liu et al., 2019). These refinements collectively enable RoBERTa to outperform BERT on various downstream tasks, demonstrating its enhanced capacity for capturing complex linguistic patterns and nuances in natural language data (Liu et al., 2019).

4.4 Model Augmentation

4.4.1 CNN Layer

A convolutional neural network (CNN) processes data by applying filters or kernels over input data to extract relevant features. Particularly effective in computer vision tasks, CNNs have also demonstrated utility in natural language processing (NLP), especially when applied to text classification. In a CNN architecture for text, the network typically utilizes one-dimensional convolutions over sequences of words or characters, allowing it to capture local patterns and hierarchical representations within the text (Kim, 2014). Each convolutional layer learns to detect specific features, such as n-grams or semantic structures, by sliding the filter over the input sequence and computing a feature map(Kim, 2014).

The equation describing the convolutional layer is:

$$y_j = \sum_{k=-p}^{p} x_{j-k} w_k \tag{13}$$

where x is our input, w is the kernel, and p is the position in relation to the point of interest.
Integrating a convolutional neural network (CNN) layer before the BERT or RoBERTa models can offer significant enhancements in text classification tasks. Unlike recurrent models like BERT and RoBERTa, which inherently capture sequential dependencies in text data, CNNs excel at capturing local patterns and feature representations through convolutions (Kim, 2014). By incorporating a CNN layer as a preprocessing step, the model can extract relevant features from the input text at different granularities, capturing both low-level and high-level linguistic structures. This feature extraction capability is particularly beneficial for tasks where certain patterns are crucial for classification, such as sentiment analysis or aspect-based sentiment analysis.

4.4.2 Freezing Layers

Freezing layers refers to the process of preventing certain layers within a neural network from being updated during the training process. When you freeze a layer, its weights and biases remain unchanged, and they do not get updated through backpropagation. In BERT, the initial layers capture general linguistic patterns and semantic information from vast pretraining data, which are often transferrable across various downstream tasks (Devlin et al., 2018). By freezing these layers, we essentially maintain the learned representations without allowing them to change during fine-tuning. This strategy reduces the number of parameters that need to be updated during training, thereby accelerating the convergence process and reducing computational resources and time requirements. By freezing layers, we hope to drastically speed up our models' performance in exchange for a small trade off in performance.

4.5 Data Exploration and Preprocessing

As a critical initial step in our analysis, we carried out an extensive exploration of our dataset to gain a comprehensive understanding of its characteristics and underlying patterns. This informed our subsequent data preprocessing and model selection decisions.

- **Text Length Analysis:** Analyzed character length distribution in reviews to inform the determination of sequence lengths in models.
- **Star Rating Distribution:** Assessed the distribution of star ratings to identify potential class imbalance, influencing the choice of evaluation metrics and sampling techniques.
- **Review Length vs Star Rating:** Investigated the relationship between review length and star rating to reveal potential correlations for feature engineering.
- **Named Entity Analysis:** Analyzed the frequency of named entities in the dataset and removed overly dominant ones to prevent the model from overfitting to specific entities.

5 Experiments

5.1 Data

The dataset being used is the Yelp Open Dataset published by Yelp and is a subset of their businesses, reviews, and user data for use in connection with academic research (Yelp, 2022). The dataset contains 6,990,280 reviews, 150,346 businesses, and 11 metropolitan areas. Though the data contains additional information about restaurants, we are only focusing on the language modeling task of analyzing text reviews. To be more specific, the classification task is to predict 1-5 star rating of a specific restaurant given a text review.

To gain insights into the data, we conducted an analysis of the reviews, which included investigating the distribution of star ratings, the length of reviews, and the frequency of common named entities.

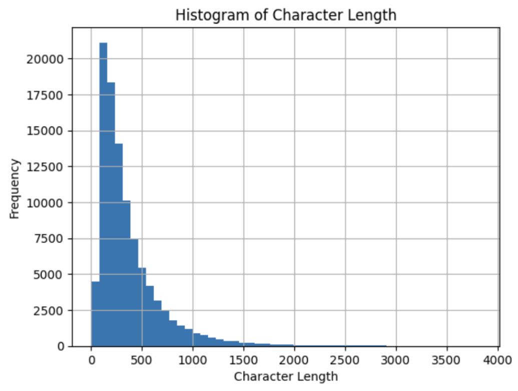


Figure 1: Text Length Analysis

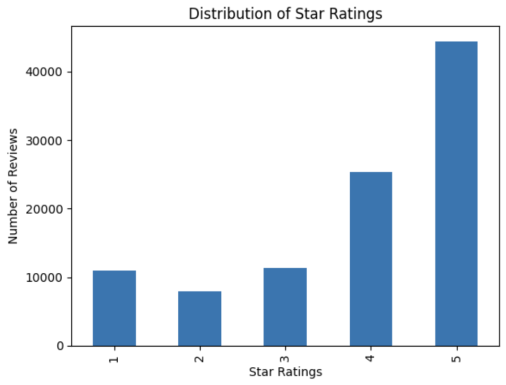


Figure 2: Star Rating Distribution

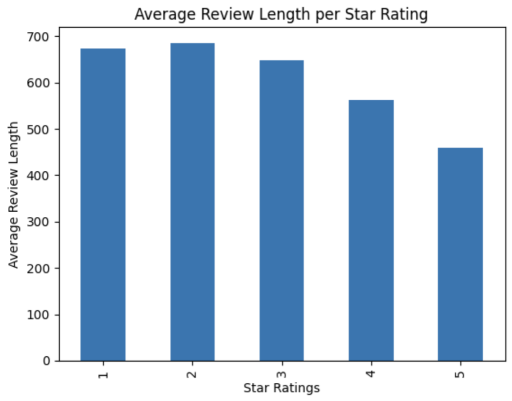


Figure 3: Review Length vs Star Rating

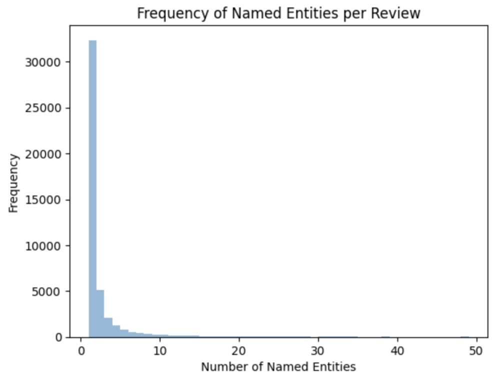


Figure 4: Named Entity Frequency Analysis

5.2 Evaluation method

The primary metrics for evaluation were overall accuracy, per-class accuracy, precision, recall, and F1 score. Overall and per-class accuracy measures the proportion of correct predictions made by the model out of all predictions. Precision refers to the proportion of true positive predictions out of all positive predictions. Recall is the proportion of true positives out of all actual positives. The F1 score is the harmonic mean of precision and recall, offering a balance between the two.

5.3 Experimental details

5.3.1 Data Preprocessing

Through the data exploration process and testing process, we compared average model results without preprocessing, with oversampling to balance class data in training data, named entity removal (NER), and a combination of oversampling and NER. These methods were informed by the data exploration process. After comparing average model performance across all methods, we finalized on the combination of oversampling and NER which we used when training and testing all models listed below.

All of the following experiments were repeated with each of our preprocessing methods:

5.3.2 Baseline RNN

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. The batch size was set to 32, with a learning rate of 0.005 for Adam optimizer. For data embedding, we used learned embeddings that were built into the model with a torch embedding layer with a vocab size of 20,000, embedding dimension of 100, and max length of 256. We trained the model for 100 epochs and saved the first model matching the best results, which took 47 epochs.

5.3.3 LSTM

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. The batch size was set to 32, with a learning rate of 0.005 for Adam optimizer. For data embedding, we used learned embeddings that were built into the model with a torch embedding layer with a vocab size of 20,000, embedding dimension of 100, and max length of 256. We trained the model for 100 epochs and saved the first model matching the best results, which took 32 epochs.

5.3.4 Finetuning BERT

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. We used the pretrained BERT architecture provided by huggingface, loading up the default 'bert-base-cased' model. This model has 12 Bert layers, with 109 million parameters. In the first layer, the embedding size is 768 with a max length of 512 words. We used the AdamW optimizer, with a batch size of 16, and ran for 5 epochs.

5.3.5 CNN-BERT

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. We used the pretrained BERT architecture provided by huggingface, loading up the default 'bert-base-cased' model. this time with a convolutional layer and maxpool layer stacked with it as a Pytorch model. The convolutional 1D layer takes the same 768 input channels, but outputs 256 channels, and has a kernel size of 3. We used the AdamW optimizer, with a batch size of 16, and ran for 5 epochs.

5.3.6 Frozen BERT

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. This model is the same as the finetuned BERT model, except the first six layers are frozen, meaning that they do not change weights during model training. We used the AdamW optimizer, with a batch size of 16, and ran for 5 epochs.

5.3.7 Finetuning RoBERTa

We used a train/val/test split of 0.7, 0.2, and 0.1 respectively. We used the pretrained RobertaForSequenceClassification model, also with AdamW for 5 epochs. All experimental parameters were the same as with finetuning BERT.

5.3.8 CNN-RoBERTa

The same modifications made to BERT were made to RobertaForSequenceClassification. All other experimental procedures were the same.

5.3.9 Frozen RoBERaT

The same first six frozen layers made to BERT were made to RobertaForSequenceClassification. All other experimental procedures were the same.

5.3.10 Hyperparameter Optimization

In our systematic review, each model was subjected to Bayesian hyperparameter optimization for all hyperparameters using the package Optuna. Bayesian hyperparameter optimization starts with defining a search space for the hyperparameters, which includes the range or distribution of possible values for each hyperparameter. Bayesian optimization uses probabilistic models to estimate the performance of different hyperparameter configurations and decides where to explore the search space next using a surrogate model such as a Gaussian model, and stops adjusting the hyperparameters either when reaching a set performance threshold or number of iterations. In this case, for each model we ran it for 10 iterations.

5.4 Results

	Precision	Recall	F1 Score	Accuracy
No-Preprocessing	0.51	0.47	0.47	0.57
Oversampling for Imbalanced Classes	0.59	0.54	0.56	0.63
Named Entity Removal (NER)	0.52	0.49	0.49	0.58
Oversampling and NER	0.61	0.57	0.57	0.65

Table 1: Average Metrics across RNN, LSTM, Finetuned BERT, and Finetuned RoBERTa for Different Preprocessing Techniques

Model	Test Accuracy	Precision	Recall	F1 Score	Accuracy (1-star)	Accuracy (2-star)	Accuracy (3-star)	Accuracy (4-star)	Accuracy (5-star)
Baseline RNN	0.50	0.51	0.43	0.32	0.61	0.36	0.38	0.29	0.49
LSTM	0.55	0.58	0.45	0.31	0.68	0.39	0.40	0.31	0.56
BERT	0.67	0.62	0.61	0.61	0.78	0.49	0.49	0.54	0.80
CNN Bert	0.67	0.62	0.61	0.61	0.78	0.47	0.51	0.55	0.78
Frozen Bert	0.65	0.61	0.59	0.62	0.76	0.48	0.49	0.57	0.76
RoBERTA	0.73	0.66	0.65	0.67	0.86	0.54	0.55	0.58	0.88
CNN RoBERTa	0.73	0.66	0.65	0.67	0.86	0.55	0.53	0.57	0.87
Frozen RoBERTa	0.70	0.63	0.62	0.62	0.82	0.52	0.50	0.55	0.83

Table 2: Model Performance for Different Star Ratings

Our results are as expected. There was a significant increase in all of the models performances as compared to a random guess. BERT outperformed the baselines and then was outperformed by RoBERTa. While expected to see a bigger performance jump between BERT and the LSTM(with primarily accuracy only jumping from 0.55 to 0.67), the jump between BERT and RoBERTa was around the smaller magnitude we were expecting(0.67 to 0.73). Our model performed significantly better on more extreme reviews - 1 starts and 5 starts - than the 'middle' reviews, with an accuracy jump from around 0.54/55 to upwards of 0.86 in our best RoBERTa model.

Based on the results from the preprocessing, NER removal made virtually no difference, with an accuracy jump os just 0.01 from the baseline. However, oversampling for imbalanced classes made a substantial difference, with a jump from 0.57 to 0.63 in accuracy on average.

6 Analysis

6.0.1 Preprocessing

Preprocessing made a small but substantial difference in our results. It was expected that NER would improve test accuracy as the model would be able to better learn textual meaning instead of overfitting to named entities such as restaurant names. Additionally, class oversampling was also expected to increase test accuracy as the model would have equal exposure reviews equally distributed by class (star rating). As seen in the results, when applying both oversample and NER removal, there is an accuracy jump of 0.08. However, it is clear that NER removal did not make a significant impact. From no-preprocessing to NER removal, there is only an increase of 0.01, which is trivial.

6.0.2 Baselines vs BERT vs RoBERTa

BERT(0.67) clearly outperformed the baseline RNN(0.5 test accuracy) and LSTM(0.55) models, and was subsequently outperformed by RoBERTa(0.73). However, the performance jumps were not massive, as detailed in the results. This implies that the task is simple enough such that in a situation where compute resources are limited(for finetuning a large transformers model), the LSTM would perform adequately.

6.0.3 Class Analysis of Accuracy

The model's performance indicates a significant improvement in predicting extreme ratings (1 or 5 stars) compared to moderate ratings (2, 3, or 4 stars). The accuracy increases on average by 0.3 when moving from moderate to extreme rating categories. Extreme ratings (either 1 or 5 stars) generally correspond to very strong positive or negative sentiments towards the product or service. The language used in these reviews is often charged with strong emotions and uses definitive, unambiguous words. For example, a 1-star review might use words like 'terrible', 'worst', or 'disappointing', while a 5-star review might include words like 'excellent', 'perfect', or 'amazing'. On the other hand, moderate ratings (2, 3, or 4 stars) are often associated with more nuanced opinions, where the customer might have liked some aspects of the product or service but disliked others. The language used in these reviews tends to be more ambiguous and balanced, making it harder for the model to correctly predict the sentiment. Words like 'average', 'okay', or 'fair' might be used, which do not convey a strong positive or negative sentiment. The subtleties and complexities in these reviews make them more challenging for the model to accurately classify, resulting in lower accuracy for these rating categories.

6.1 Model Augmentations

The addition of a Convolutional Neural Network (CNN) layer to the BERT and RoBERTa models did not significantly impact their performance. This is indicated by the negligible differences in the key performance metrics - overall accuracy, precision, recall, and F1 score. This suggests that the CNN layer did not contribute any additional predictive power to these models for this specific task.

Additionally, an experiment was conducted to freeze the first six layers of both BERT and RoBERTa models. The idea behind this approach is to preserve the pre-trained patterns in these initial layers while fine-tuning the rest of the model. However, this adjustment resulted in a slight decrease in performance, with a drop of around 0.03 in accuracy for both models. Despite this reduction in model complexity, the computational efficiency improved only slightly with training time reducing from 4 hours and 55 minutes to 4 hours and 37 minutes in the case of BERT as an example. These results suggest that the transformer architecture of BERT and RoBERTa, without additional CNN layers or freezing initial layers, is quite efficient for this particular task. The attempt to increase computational efficiency through model simplification resulted in a marginal decrease in performance but did not significantly reduce the computational time.

7 Conclusion

In this project, we built four different types of models, some augmented with different techniques such as unique architecture choices and frozen weights, for multi-class text classification. All of these models vastly outperformed random chance. As expected, RoBERTa showed why it was considered state of the art and outperformed the baseline and the default BERT model.

One of our biggest accomplishments was experimenting with different types of text preprocessing and dataset manipulation. Oversampling for imbalanced classes resulted in a clear improvement in model performance, and although NER removal did not make a huge impact, it was a challenging accomplishment to perform it in the first place. We also tuned the hyperparameters to ensure we were maximizing model performance

One limitation of the project was the nature of the data itself. The 2/3/4 star reviews were much more ambiguous, and were therefore significantly harder to differentiate than the more extreme reviews. Additionally, even with our sampled dataset, the text data and large models made it difficult to iterate on our models even with additional compute resources. Future work for this project could involve ideas such as:

- Comparing to even newer state of the art models such as GPT
- Correcting spelling mistakes in the Yelp reviews to better align the data to the model's pretrained tokens/embeddings
- Utilizing resources such as Stanford's Sherlock cluster to train on the full 5 million review dataset

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. volume abs/1810.04805.

Y. Kim. 2014. Convolutional neural networks for sentence classification. In *Computation and Language*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*.

Michael Luca. 2016. Reviews, reputation, and revenue: The case of yelp.com. 12-016.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. volume abs/1211.5063.

Alex Sherstinsky. 2018. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. volume abs/1808.03314.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

M. Wankhade, A.C.S. Rao, and C. Kulkarni. 2022. A survey on sentiment analysis methods, applications, and challenges. volume 55, page 5731–5780.

Inc. Yelp. 2022. Yelp dataset. `https://www.yelp.com/dataset`.

8 Appendix

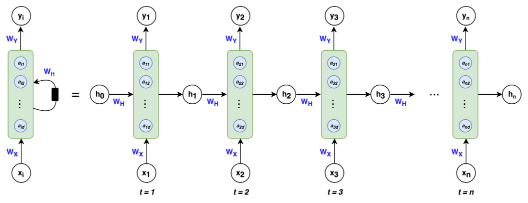


Figure 5: RNN Architecture

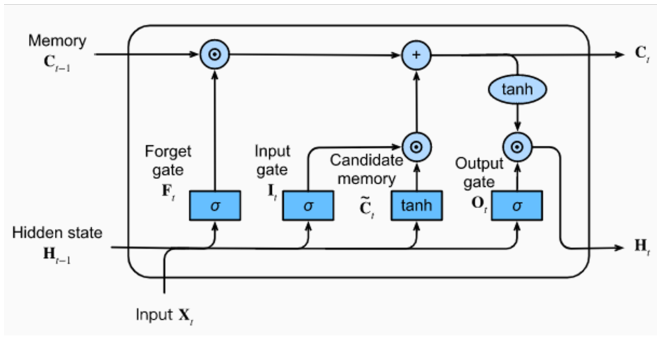


Figure 6: LSTM Architecture

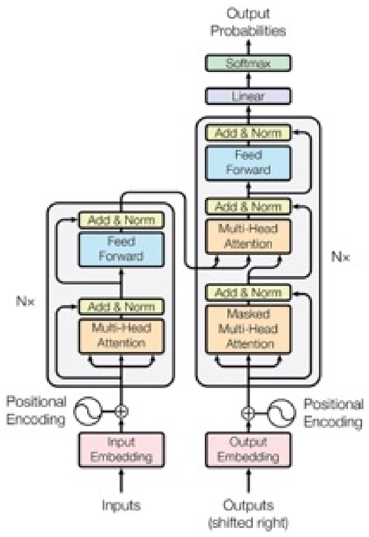


Figure 7: BERT Architecture

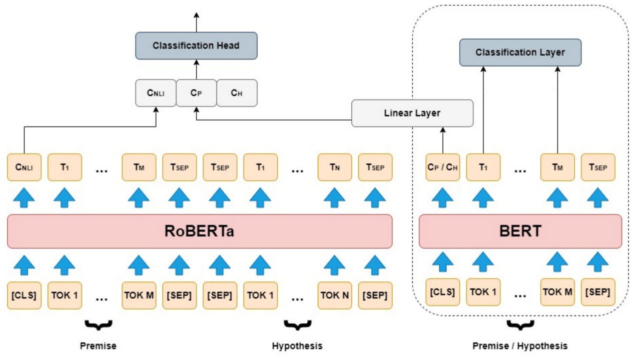


Figure 8: RoBERTa Architecture