CS374 Project: Yelp-5 Sentiment Analysis

Eric Zander and Bailey Magelli

November 27, 2022

Due to its wide applicability, sentiment analysis is a natural language processing (NLP) task in which further experimentation is always welcome. However, the dominance of deep learning models in this domain also encourages greater exploration of methods in explainability. Given that attention mechanisms afford both high performance and intuitive visualizations, an architecture making use of multi-head self-attention is leveraged to make and visualize predictions of 5-star user ratings based on review text available in the Yelp-5 dataset.

1 Introduction

Sentiment analysis has maintained its status as a highly relevant intersection of natural language processing (NLP) and data science for good reason. Automatically determining the affective characteristics of social media posts, news articles, historical documents, and a wide variety of other corpora proves indispensable in domains where manual analysis is too inefficient at scale. Given the relevance of such data for research and decision-making, performant sentiment analyzers stand to offer tremendous value.

While the application of novel models and architectures have afforded greater performance and convenience in this domain, there exist two areas in which improvement is always welcome. The first is accuracy; competitive performance in this regard is clearly desirable. However, model explainability constitutes another dimension by which analyzers could benefit from further experimentation. This is especially true in the case of notoriously complex deep learning models capable of offering superior performance.

We set out to develop the foundation for a sentiment analyzer that leverages deep learning methods while offering some mechanisms for explainability. To do this, we used user reviews from the Yelp Dataset (Yelp-5)[1] for training as well as user reviews from IMDb[2] and Steam[3] for additional testing. The model itself makes use of multi-head self-attention to both make predictions and return interpretable attention-scores.

2 Related Work

In 2019, Abreu et al. compared performances of various deep learning architectures on Yelp and IMDb datasets.[4] These include a proposed Hierarchical Attentional Hybrid Neural Network (HAHNN), Convolutional Neural Networks (CNN), and Temporal Convolutional Networks (TCN). The best models achieved 73.28% accuracy for Yelp's 5-star ranking system and 95.17% accuracy for IMDb's binary classification of sentiment.

Other attempts at classification of the Yelp-5 dataset include the employment of the XLNet framework by Yang et al. in 2019[5] as well as the Big Bird Transformer by Zaheer et al. in 2020.[6] These approaches also leverage attention, and afford results comparable to those achieved by Abreu et al. Bidirectional Encoder Representations from Transformers (BERT) were also fine-tuned by Sun et al. in 2019 for the next best performance.[7] These comprise top performances from recent years. The next best result was achieved by Adhikari et al. with a regularized Recurrent Neural Network (RNN) that landed below 70% accuracy.[8] See Table 1 for details.

Authors	Model	Acc. (%)
Abreu et al.	HAHNN	73.3
Yang et al.	XLNet	73.0
Zaheer et al.	Big Bird	72.2
Sun et al.	BERT-ITPT-FiT	70.6
Adhikari et al.	Reg. LSTM	68.7

Table 1: Yelp-5 Performance

It is clear from the described performances that attention and convolutional networks are highly performant. Notably, applying multiple attention layers as with HAHNN or XLNet can help models excel. Consequently, explainability measures should be appropriate for these architectures.

Gholizadeh and Zhou (2021) provide an exploration of some strategies for explainability in NLP using the Yelp dataset as an example.[9] Approaches like Local Interpretable Model-agnostic Explanations (LIME) work by generating understandable inputs and allowing the comparison of model outputs. CNNs could also be supported; Layer-wise Relevance Propagation (LRP) can trace word significance back through convolutional and dense layers to identify the impact of input words on the output. Consequently, CNN-based architecture and LRP could be employed. However, a simpler method of visualization can be applied to models that leverage attention.

In the case of sentiment analysis, attention serves to encode and embed sentences with disproportionate focus on the features that are more relevant to the final prediction. These attentions may then be saved during the prediction process and aggregated to visualize the features considered important by the model. Yang et al. demonstrate this in their paper regarding HAHNN.[5]

3 Approach

Our sentiment analyzer may be split into two components. The first is a preprocessing pipeline for preparing input to our model. The spaCy Python package is used for several components of this.[10] Modelling, the second component, makes use of Tensorflow and Keras.[11]

3.1 Preprocessing

The preprocessing step consists of operations performed on text prior to encoding for attention. We experimented with numerous operations supported by a custom configurable preprocessor.

The following elements were tested in the text normalization step with help from spaCy:

- **Newline Removal**: Removal of explicit newline escape sequences ('\n')
- **Punctuation Removal**: Removal of non-alphanumeric characters.
- **Punctuation Padding**: Adds a space on either side of non-alphanumeric characters.
- **Condense Spacing**: Replaces multiple consecutive spaces with a single space.
- **Lowercase**: Converts all tokens to lowercase.
- **Split Numbers**: Splits numbers into two tokens. The first is up to two leading digits. The second is the component of scientific notation illustrating decimal offset. See Figure 1 for examples.
- **NER Type**: Replaces a token with a named entity type (Ex. 'Peter' becomes 'person')
- **Stopword Removal**: Omits a token if it is determined to be a stopword.
- **Lemmatize**: Replaces a token with a root meaning.

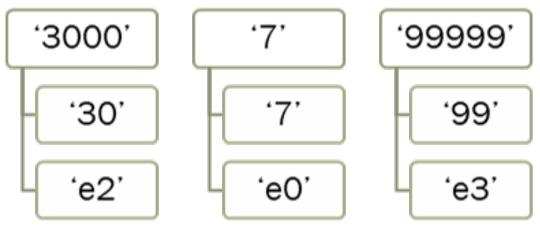

Figure 1: *Examples of the 'split numbers' preprocessing step. This helps limit the representation of numbers in the vocabulary without collapsing them into one named entity type.*

The order of execution of these steps are viewable in Figure 2.

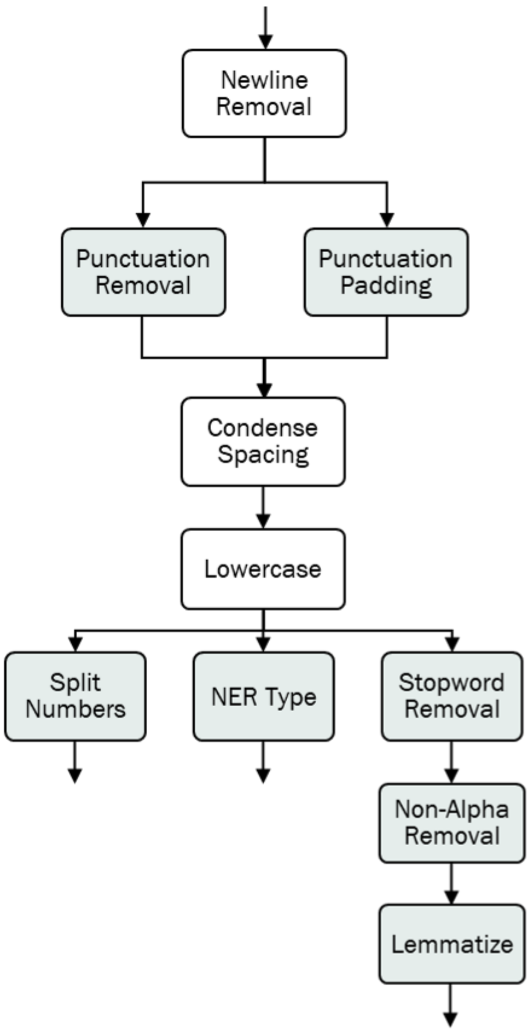

Figure 2: *Preprocessing pipeline diagram. Elements represented in green are optional in our preprocessor, but some are dependent on others. For example, no punctuation can be padded if it is instead removed.*

After this pipeline is applied to each document, sequences are generated with each token represented by an integer indicating its index in a vocabulary of a fixed size. These sequences are then padded to a fixed length. Both of these steps are executed with Tensorflow's tokenizer. Finally, the sequences are converted to tensors to be used as input.

3.2 Modelling

The core of the chosen architecture is a multi-head self-attention block for embedding input sequences. This requires both token and positional encoding of the preprocessed input, but only makes use of a relatively minimal feed-forward network after the fact. A diagram of the architecture may be viewed in Figure 3.

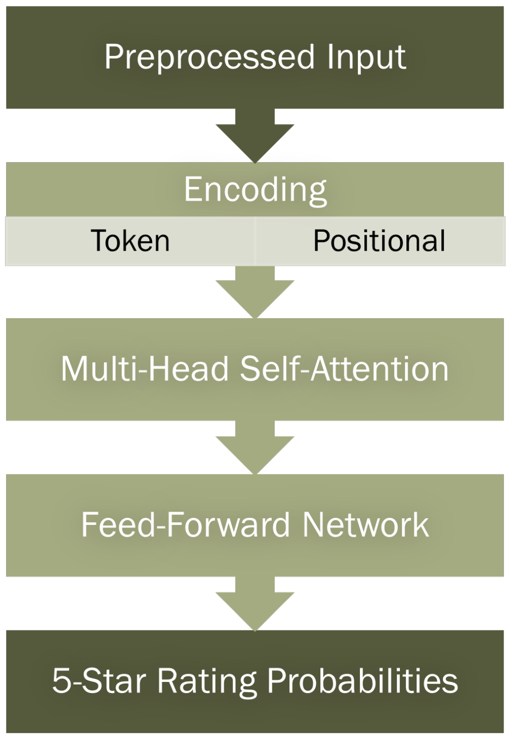

Figure 3: *Overview of model architecture*

The loss function employed in all training is categorical cross-entropy.

Model performance is described in the Experiments section and the Final Modelling subsection in particular. Discussion concerning limitations and potential improvements are highlighted in the Discussion section.

4 Experiments

Two formalized experiments were run with preprocessing pipeline configurations and regularization parameterization. Additionally, a review of the final modelling and validation results are given.

4.1 Preprocessing Comparison

To decide which preprocessing steps to perform on all samples, testing was performed with several pipeline configurations. Models with the same architectures were trained on stratified samples with 250,000 samples for 5 epochs and tested on 50,000 samples.

The components included in each configuration may be seen in Table 2. The base configuration was chosen following earlier testing.

Config.	No Stopwords	Remove Punc.	Pad Punc.	Lemmatize	Alpha. Only	Split Digits	NER Type
base	X		X	X		X	X
no_ner	X		X	X		X	
no_digit	X		X	X			X
no_lemma	X		X			X	X
no_punc	X	X		X		X	X
alpha_only	X		X	X	X		X
with_stop			X	X		X	X
stop_no_ner			X	X		X	
stop_no_lem			X			X	X
min_prep			X			X	

Table 2: Preprocessing experiment configurations.

These tests have several limitations worth remembering. Pervasive overfitting can present a problem even with the regularizing effects of training with more data. This lower number of epochs supports a degree of welcome regularization since training longer tends to exacerbate this issue. While further measures to counteract overfitting are tested and described in the Regularization Comparison section, this issue should be noted alongside the limited number of configurations tested here.

The purpose of this experiment is primarily to select a preferable preprocessing pipeline with which further experimentation may be executed. To this end, the configuration without stopword removal or lemmatization offered the best experimental results. Each of the best validation losses and accuracy scores per epoch may be viewed in Figure 4.

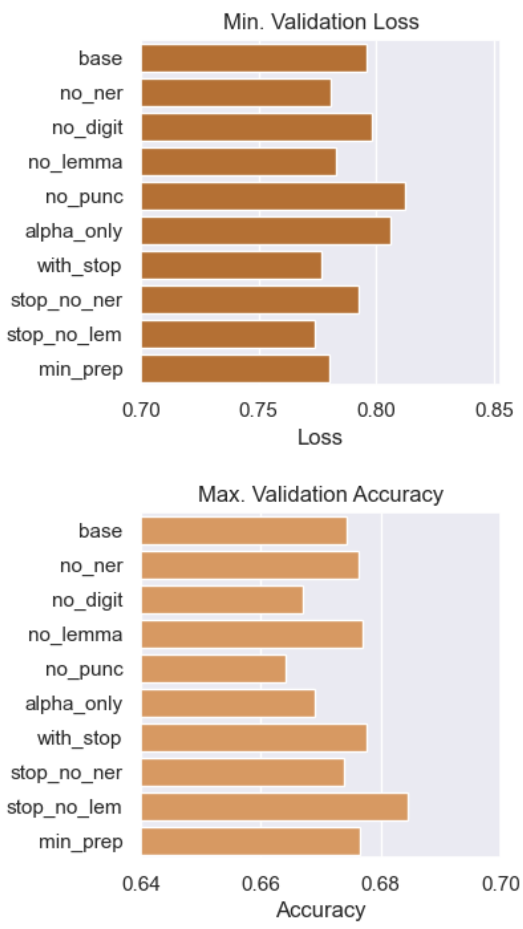

Figure 4: *Best iterations per preprocessing configuration.*

4.2 Regularization Comparison

Several mechanisms can be leveraged to combat overfitting. The first is the use of additional data in training. However, the points of focus in this experiment are regularization terms added to the loss function as well as dropout rates from layer to layer. L2 regularization of varying degrees were applied to address the former while various dropout rates were applied to attention and dense layer output to address the latter. The tested configurations may be seen in Table 3.

Configuration	L2 Reg.	Dropout Rate
none	0.000	0.00
low	0.001	0.10
med	0.010	0.25
high	0.050	0.50

Table 3: Regularization testing parameters.

It is evident from testing that overfitting constitutes a significant problem for this number of samples regardless of the degree of regularization. However, as seen in Figure 5, the best balance is struck via middling values within the confines of the tested configurations. Consequently, such parameters are employed in final modelling.

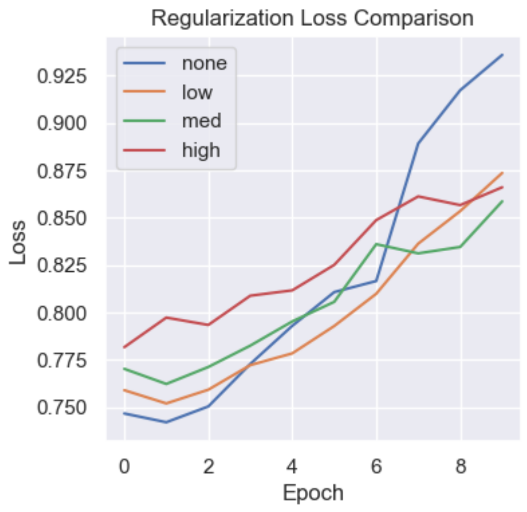

Figure 5: *Comparison of regularization parameter configurations performances for 250,000 samples.*

4.3 Final Modelling

With the preprocessing and degrees of regularization decided, training was performed on 80% of the Yelp dataset. 10% was used for local validation, and the last 10% was set aside for final validation.

The change in accuracy and loss per epoch are viewable in Figure 6.

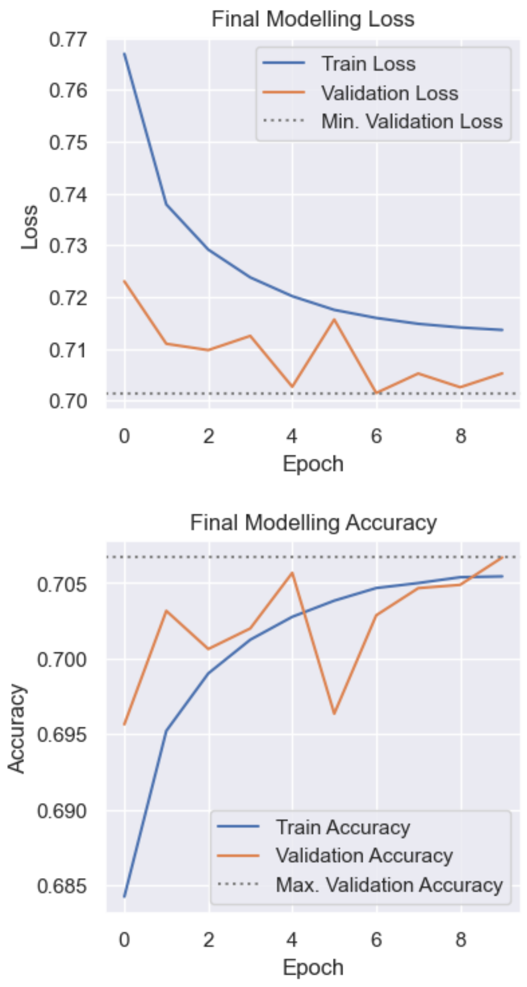

Figure 6: *Overview of final modelling loss and accuracy.*

The final model was saved following the epoch where validation loss was minimal.

4.4 Yelp-5 Validation

Applying the final model to the validation portion of the dataset afforded an accuracy of 70.2%. This improves upon the performance of the regularized LSTM model, but falls just short of the fine-tuned BERT. Consequently, the model is about 2-3% less accurate than the other discussed attention-oriented or convolutional models. These performances are visualized in Figure 7.

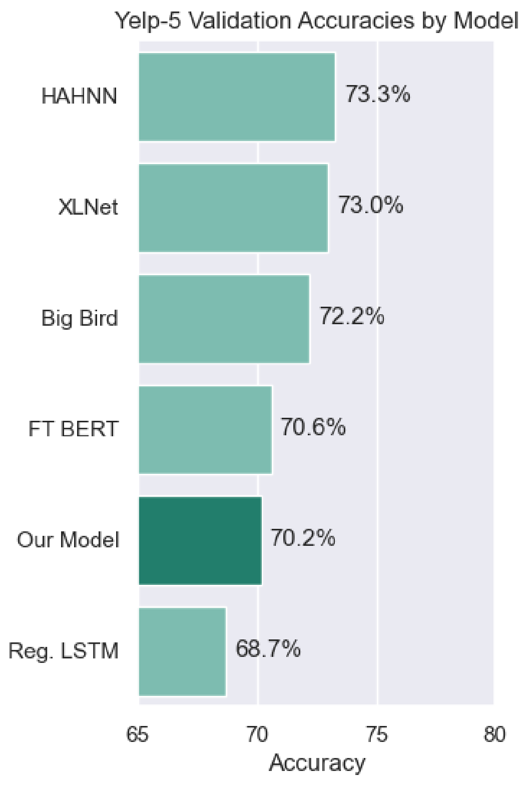

Figure 7: *Comparison of validation performance by model type for 5-star Yelp review datasets.*

Attention visualization was performed for these predictions, and may be seen in Figures 8 and 9. One pattern that becomes clear is the prioritization of earlier words.

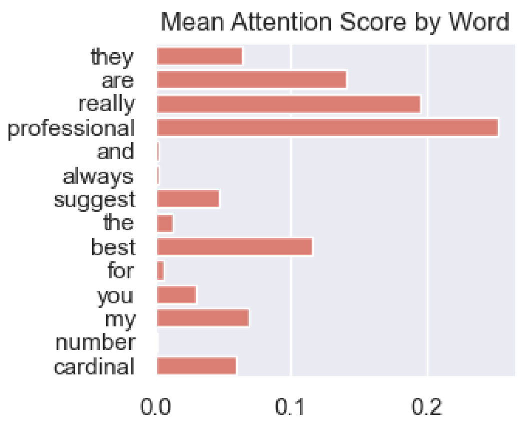

Figure 8: *Example of an attention score visualization in which the words prioritized by the model are made clear.*

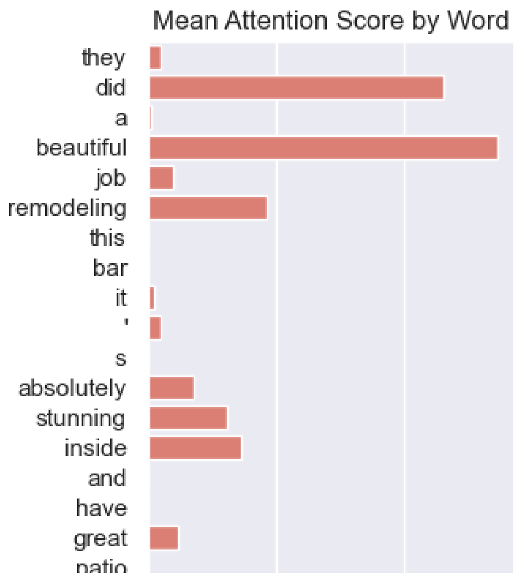

Figure 9: *Another truncated example of an attention score visualization further illustrating the emphasis on earlier words.*

Given that the model outputs the probabilities for each star rating, a precise continuous star rating may be derived by determining the expected value:

$$E[X] = \sum x_i p(x_i)$$

The expected star rating may be used to determine mean absolute error (MAE). During validation, the MAE was 0.3598 stars, or less than half a star off by average.

4.5 Binary Sentiment Classification

The expected star rating also allows the approximation of binary sentiment. This is somewhat imprecise, as a decision must be made concerning whether sentiment is positive or negative based on 5-star rating.

An expected rating of less or more than 2.5 stars could be considered negative and positive respectively, but one could largely express negative sentiment in a review as justification for only 3 stars or higher. Applying statistical methods for determining the point at which sentiment becomes positive is another option, but measures of central tendency may not reflect sentiment if more or less than half of reviews are positive.

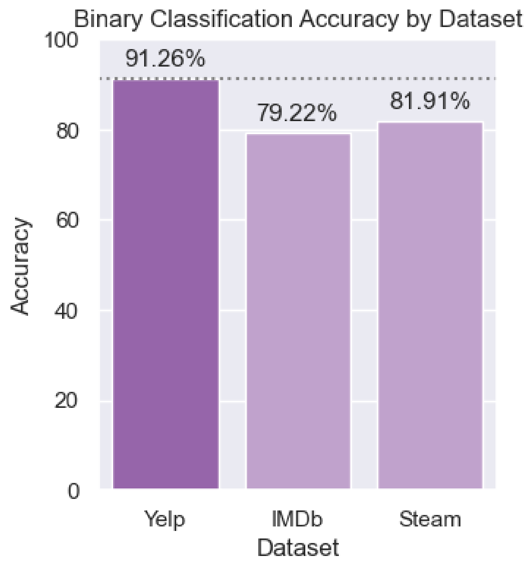

Figure 10: *A comparison of binary classification accuracy for each of the three tested validation datasets.*

In the following experiments, a simple heuristic is employed; the sentiment of a review with an expected rating of less than 3 stars is considered negative. While this allows the reasonable conversion of discrete labels in the original dataset to binary labels, affirming the practical value of these results should be understood to require further experimentation given the discussed complexities.

Binary classification was tested with the Yelp dataset as well as the IMDb and Steam review datasets discussed in the introduction. The results are also visualized in Figure 10.

4.5.1 Yelp

Testing binary classification with the validation portion of the Yelp dataset offered an accuracy of 91.26%.

4.5.2 IMDb

Testing on 50,000 labeled movie reviews from IMDb offered an accuracy of 79.22%. This included the additional removal of HTML tags that were prevalent in this dataset.

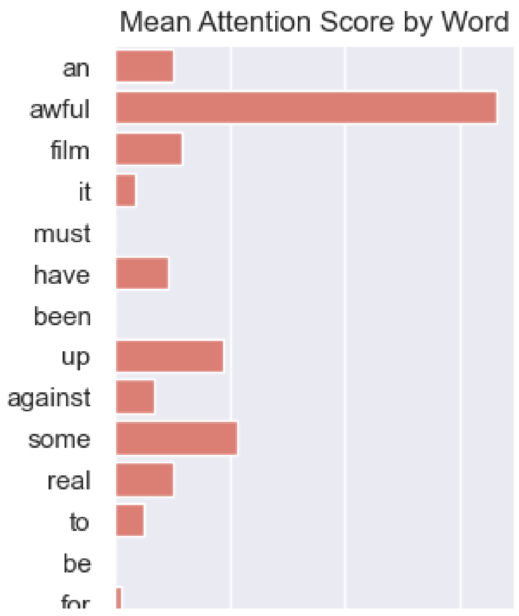

Figure 11: *A truncated example attention score visualization for an IMDb user review.*

4.5.3 Steam

Finally, the model was tested on 50,000 labelled game reviews from Steam. This afforded an accuracy of 81.91%. This is comparable to the results for the IMDb dataset.

5 Conclusion

While the final validation accuracy is certainly comparable to other competitive models, it falls short of outperforming almost all of them. While it is clear that additional architectural complexity may be warranted, one should note that managing additional complexity also requires additional consideration of overfitting; naive addition of layers and neurons with the architecture employed in these experiments only exacerbated this problem.

Hierarchical attention in particular would appear to be a reliable direction going forward for multiple reasons. The first is the clearly superior performance, but the second is explainability. While another attention mechanism certainly increases the complexity at the risk of harming ease of understanding, word and sentence level attention are both intuitive and easy to visualize. The work by Abreau et al. illustrates this well.

However, sentence- and word-level attention visualization alone falls short of full model explainability. Further experimentation with combining attention visualization and LRP may be worthwhile to further contextualize the significance of attention for any one sequence component.

On the subject of binary classification, the results within the Yelp dataset are promising. While there was a noticeable drop-off in performance for the IMDb dataset, the result is still clearly preferable to potential of baselines of 50% for half or 60% for 3 star ratings. Finally, it is worth considering that greater prevalences of misspelling, sarcasm, and special characters indicate that tailoring the preprocessing pipeline to other dataset may be necessary for achieving superior performances. This is especially true for the tested Steam dataset in which the model afforded similar results. More fundamentally, there are other preprocessing steps could be tested prior to encoding.

Overall, this particular set of experiments offers little in the way of challenging the state of the art. However, the groundwork is laid for further experimentations in the domain of deep learning explainability in particular.

References

[1] Yelp. *Yelp Open Dataset*. URL: https://www.yelp.com/dataset.

[2] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[3] Antoni Sobkowicz. *Steam Review Dataset*. 2017. DOI: 10.5281/zenodo.1000885. URL: https://www.kaggle.com/datasets/andrewmvd/steam-reviews.

[4] Jader Abreu et al. "Hierarchical Attentional Hybrid Neural Networks for Document Classification". In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*. Springer International Publishing, 2019, pp. 396–402. DOI: 10.1007/978-3-030-30493-5_39. URL: https://doi.org/10.1007%5C%2F978-3-030-30493-5_39.

[5] Zhilin Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2019. DOI: 10.48550/ARXIV.1906.08237. URL: https://arxiv.org/abs/1906.08237.

[6] Manzil Zaheer et al. "Big Bird: Transformers for Longer Sequences". In: (2020). DOI: 10.48550/ARXIV.2007.14062. URL: https://arxiv.org/abs/2007.14062.

[7] Chi Sun et al. *How to Fine-Tune BERT for Text Classification?* 2019. DOI: 10.48550/ARXIV.1905.05583. URL: https://arxiv.org/abs/1905.05583.

[8] Ashutosh Adhikari et al. "Rethinking Complex Neural Network Architectures for Document Classification". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4046–4051. DOI: 10.18653/v1/N19-1408. URL: https://aclanthology.org/N19-1408.

[9] Shafie Gholizadeh and Nengfeng Zhou. *Model Explainability in Deep Learning Based Natural Language Processing*. 2021. DOI: 10.48550/ARXIV.2106.07410. URL: https://arxiv.org/abs/2106.07410.

[10] Matthew Honnibal and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". 2017.

[11] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.