For this project, I will use the Yelp dataset containing restaurant reviews. My research question is, “Can we create a neural network that will predict sentiment accurately?”. This analysis aims to have a neural network that can accurately predict sentiment from restaurant reviews so we can understand how well we have been doing with our customers. The neural network I will be using is the long short-term memory (LSTM) neural network. The LSTM network is more efficient due to its ability to remember long-term dependencies, that a recurrent neural network (RNN) struggles with due to vanishing and exploding gradients. It is crucial to understand the context of the customer’s reviews, and with the advanced LSTM capabilities, this would be the best neural network to use (Luay, 2023).

I started with exploratory data analysis of the yelp dataset. I utilized tensorflow’s regex\_replace function to eliminate the unusual characters like ‘…’ as seen below, so these would not create noise (*Tf.Strings.Regex\_Replace*, n.d.). These unusual characters provide no additional meaning (WGU Learning, n.d.) After cleaning, pre-processing, tokenizing, and padding the text, the vocabulary size was 1842. This is a limited dataset, so I wanted to ensure I had the appropriate amount of vocab size (*How to Find “Num\_Words” or Vocabulary Size of Keras Tokenizer When One Is Not Assigned?*, n.d.). I utilized tensorflow’s layers.Embedding for word embedding based on my tokenizer word index plus 1 for padding, as this utilizes all the vocabulary available (*Why Keras Embedding Layer’s Input\_Dim = Vocab\_Size + 1*, n.d.). I created a distribution visual to see the frequency of different sentence lengths, calculated the 90% percentile, and checked the largest sentence length. While 90% of the sentence lengths were 20 and under, the largest was only 32. I did test both for the modeling, but with the padding being minimal and the sentence length and number of sentences being small, I decided to go with all 32 sentence lengths with padding. This retained more information for the model to train on and did provide better results.

A screenshot of a computer program

AI-generated content may be incorrect.

A graph of a number of words

AI-generated content may be incorrect.

The goal of tokenization is to break down the texts into their own units (WGU Learning, n.d.). These units are represented as numbers so that the computer can understand the numbers and their patterns. Since I utilized TensorFlow for this project, I utilized their tokenizer from tensorflow.keras.preprocessing.text (Deep, 2021). This was of course, occurring after removing stopwords and lemmatization. I utilized the Natural Language Toolkit (NLTK) for stopwords (*Remove Stopwords in Tensorflow Extended*, n.d.) and lemmatization (*How Do I Do Word Stemming or Lemmatization?,* n.d.). Stopwords are words that do not carry significant meaning. Lemmatization reduces words to their base form within the word’s context. Both are used to reduce the complexity of the model and focus on what is important without noise (WGU Learning, n.d.).



A close-up of a computer code

AI-generated content may be incorrect.

The padding process I used to standardize the length of sentences included the tf.keras.utils.pad\_sequences. In this I utilized the 32 max sequence length from our sentence length. The padded shape was (1000, 32). 1000 being the number of sentences and 32 for the sentence length. I set padding as “post”, which would push the padding to the end. I did this as it seemed more reasonable to have the most information first (*tf.keras.utils.pad\_sequences*, n.d.).

A number of numbers on a white background

AI-generated content may be incorrect.

In this sentiment analysis, there were only 2 categories. The two categories are positive or negative, with 500 examples of each. I used bi-directional LSTM. Bi-directional LSTM has improved Natural Language Processing (NLP). It considers both preceding words and the words that follow, to effectively understand the context. The activation function I used in the Bi-Directional LSTM that I used was the sigmoid activation function and the tanh activation function. The input gate utilizes sigmoid activation function when creating the values between 0 and 1, the forget gate utilizes sigmoid activation function when deciding what should be discarded, and the output gate utilizes both the sigmoid and tanh activation function to create the output (Gomede, 2024).

After completing the prior steps, it was time to move forward in preparing the neural network for data analysis. The next step was splitting the dataset into training, testing, and validation sets. I followed the typical industry standard that WGU covered in the learning material. This typical split is 70% for training, 15% for testing, and 15% for validation (WGU Learning, n.d.). This left me with 700 training set sizes and 150 for the validation and test set sizes.

A screenshot of a computer

AI-generated content may be incorrect.

As mentioned before, I utilized TensorFlow for this project. There were 5 layers I used, as seen above. I utilized the embedding, bidirectional LSTM, dropout, dense, and a second dense layer type. These layers covered 277,001 parameters. The embedding layer had 184,200 parameters, the bidirectional layer used 84,480 parameters, the dropout layer utilized 0 parameters, the dense layer used 8,256 parameters, and the last dense layer utilized just 65 parameters. In this model, I used the ReLu in the first dense layer to ensure the model improved over time and sigmoid in the second dense layer as it is used in binary classification to ensure that every value belongs to one of the classes. I tried a few different node amounts, but 64 nodes in the dense layer gave the best results. I chose binary cross entropy as this loss function loss function as it is optimized for binary classification (Elisha, 2022). The optimizer I chose to use for this project was Adam. As we learned in the previous task, Adam is a standard optimizer for neural networks. I once again also used early stopping as this helped ensure the model would not be overfitting and stop after the validation loss did not improve 3 separate times. I also utilized the model checkpointing, so I could save the “best model”, as each time the validation loss decreased, the model was saved. This would ensure the best model according to validation loss, would be available to me for use after each epoch was finished (Keras Team, n.d.). The hyperparameters I used were successful, as we can see from metrics over the epochs. While our model did stop after a few epochs due to validation loss no longer improving, we can see accuracy increasing from 47.60% to 67.39%, loss decreased from 69.47% to 62.13%, validation accuracy increased from 51.33% to 78.67%, and of course, our validation loss started at 69.89% and ended at 52.68%.

The model’s training process can be seen above. While I defined the epochs to be 50, the model stopped at 6 epochs, as the stopping criteria were met for the validation loss not improving 3 times. This was done to improve the accuracy, without overfitting. Our test loss decreased over time, and our accuracy increased. The best Test Loss was 54.42% and the best test accuracy was 73.33%. As seen below, our final predictive accuracy was 73.33% when using our test accuracy. The model can predict whether sentiment is positive or negative with 73.33% accuracy.

The analysis complies with AI global ethical standards. This analysis covers guidelines and principles like mitigating bias, no personal data being used, the model being robust and lawful, and the data being representative of the entire community that has reviewed our restaurant on Yelp. This AI system is transparent in how it works and explains the data, the accountability it has for the laws and ethical standards set, and the fairness in preventing the bias AI systems may have. I am ensuring privacy by ensuring no personally identifiable data, preventing bias by including all present data, and I am holding accountability and responsibility in also discussing the limitations. The limitations to this neural network as it only has 73.33% accuracy, and we truly don’t understand if there was bias among only certain different demographics only reviewing on Yelp (WGU Learning, n.d.).



A graph showing the line of a training

AI-generated content may be incorrect.

My Bidirectional LSTM neural network is functional. The choices I made from early stopping and saving at each check point where the validation loss improved, improved my model and ensured the most accuracy without overfitting. Choosing to use a Bidirectional LSTM was right in line with performing an appropriate and thorough neural network. The model has room for growth, and I recommend gathering more Yelp reviews to train the neural network or training it on other reviews, so the model becomes more complex and accurate as it grows. I saved my model using the checkpoint = ModelCheckpoint('lstm/best\_model.keras', monitor='val\_loss', save\_best\_only=True, mode='min', verbose=1) code. As mentioned before, this improved checkpointed and saved my best model each time the validation loss improved (Keras Team, n.d.).

References:

Deep, A. (2021, December 15). *Understanding NLP Keras Tokenizer Class Arguments with example.* Medium. <https://medium.com/analytics-vidhya/understanding-nlp-keras-tokenizer-class-arguments-with-example-551c100f0cbd>

Elisha, O. (2022, March 30). *Optimizer, losses and activation functions in fully connected neural networks*. Medium. <https://medium.com/@elishatofunmi/optimizer-losses-and-activation-functions-in-fully-connected-neural-networks-e1958bc66121>

Gomede, E., PhD. (2024, March 10). *Bi-Directional Long Short-Term Memory: unveiling the power of recurrent neural networks*. Medium. <https://medium.com/aimonks/bi-directional-long-short-term-memory-unveiling-the-power-of-recurrent-neural-networks-0313219bdf20>

*How do I do word Stemming or Lemmatization?* (n.d.). Stack Overflow. <https://stackoverflow.com/questions/771918/how-do-i-do-word-stemming-or-lemmatization>

*How to find “num\_words” or vocabulary size of Keras tokenizer when one is not assigned?* (n.d.). Stack Overflow. <https://stackoverflow.com/questions/53525994/how-to-find-num-words-or-vocabulary-size-of-keras-tokenizer-when-one-is-not-as>

Keras Team. (n.d.). *Keras documentation: ModelCheckpoint*. <https://keras.io/api/callbacks/model_checkpoint/>

Kotzias, D. (2015). *Sentiment Labelled Sentences [Dataset]*. UCI Machine Learning Repository. <https://doi.org/10.24432/C57604>.

Luay, M. (2023, September 22). *Sentiment Analysis using Recurrent Neural Network(RNN),Long Short Term Memory(LSTM) and Convolutional Neural Network(CNN) with Keras*. Medium. <https://medium.com/@muhammadluay45/sentiment-analysis-using-recurrent-neural-network-rnn-long-short-term-memory-lstm-and-38d6e670173f>

*Remove stopwords in Tensorflow extended*. (n.d.). Stack Overflow. <https://stackoverflow.com/questions/66044999/remove-stopwords-in-tensorflow-extended>

Scribber. (n.d.). *Scribber*. <https://www.scribbr.com/citation/generator>

*tf.keras.utils.pad\_sequences*. (n.d.). TensorFlow. <https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences>

*tf.strings.regex\_replace*. (n.d.). TensorFlow. <http://tensorflow.org/api_docs/python/tf/strings/regex_replace>

WGU Learning. (n.d.). *Advanced Analytics.* [*https://apps.cgp-oex.wgu.edu/wgulearning/course/course-v1:WGUx+OEX0390+v01/block-v1:WGUx+OEX0390+v01+type@sequential+block@521faae8647949a2afaee98052981d37/block-v1:WGUx+OEX0390+v01+type@vertical+block@86229c5272bf45a9a2493cd5e549492e*](https://apps.cgp-oex.wgu.edu/wgulearning/course/course-v1:WGUx+OEX0390+v01/block-v1:WGUx+OEX0390+v01+type@sequential+block@521faae8647949a2afaee98052981d37/block-v1:WGUx+OEX0390+v01+type@vertical+block@86229c5272bf45a9a2493cd5e549492e)

*Why Keras Embedding layer’s input\_dim = vocab\_size + 1*. (n.d.). Stack Overflow. <https://stackoverflow.com/questions/66719029/why-keras-embedding-layers-input-dim-vocab-size-1>