Performance Assessment: Sentiment Analysis Using Neural Networks

Gabriela Howell

Master of Science Data Analytics, Western Governors University

D213 – Sentiment Analysis Using Neural Networks

Professor Elleh

October 28, 2024

Part I: Research Question

A. Describe the purpose of this data analysis by doing the following:

1. Summarize one research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen data set(s).

My research question is, "Can we accurately classify reviews as negative or positive based on the textual content of the reviews?" To answer this question, I will use multiple datasets from various sources (Amazon, IMDb, and Yelp) and combine them into a single dataset. By concatenating these datasets, we increase the overall sample size, providing better odds of identifying patterns in the text that distinguish negative reviews from positive ones.

2. Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data.

The main goal of this project is to sort reviews into positive or negative categories based on the text. The aim is to create a model that can accurately predict the sentiment of a review, helping to identify which ones are negative. This goal fits with the research question of separating negative and positive reviews, which can be achieved using the available data from Amazon, IMDb, and Yelp. By combining these datasets, we get a larger and more varied set of data, which should help improve the model's accuracy.

3. Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected data set.

I will use a neural network for text classification, such as a Recurrent Neural Network (RNN) or an advanced model like Long Short-Term Memory (LSTM). These models are good at understanding the order of words in a sentence, making them ideal for predicting sentiment. By using these networks, I can make more accurate predictions on the combined dataset.

Part II: Data Preparation

B. Summarize the data cleaning process by doing the following:

1. Perform exploratory data analysis on the chosen data set, and include an explanation of *each* of the following elements:

During data cleaning, I used Python's re package and the nltk library to remove emojis, non-English characters, and other irrelevant symbols. This made sure the text was clean and standardized for the neural network. I calculated the vocabulary size by finding unique words, which helped choose the best word embedding length.

The data had 5155 unique words, showing diverse language use. I suggested a max word length of 26 words, covering about 95% of the reviews. On average, reviews had 13 words, with the longest at 1390, which supports my choice.

2. Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process.

I used TensorFlow's Tokenizer and nltk to turn raw text into numbers. Using TensorFlow's Tokenizer and NLTK, I converted text to lowercase, removed URLs, punctuation, and stop words, and applied WordNetLemmatizer to retain root forms of words. This normalization helped the model focus on the main content and understand word relationships without being affected by capitalization or extra punctuation. TensorFlow's Tokenizer mapped words to unique integers based on frequency, creating a numerical representation that the neural network could process.

3. Explain the padding process used to standardize the length of sequences. Include the following in your explanation:

To make input sequences the same length, I used padding. With Keras's pad_sequences function, I added zeros at the start of shorter sequences to match the required length. Here's a screenshot s howing a padded sequence:

```
# Padding Process
padded_sequences = pad_sequences(sequences, padding='pre',truncating='pre', maxlen=max_length)
print("Padded Sequences:", padded_sequences[0])
```

```
Padded Sequences: [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0  860  861  233 1900 1901 1902  862  863  171]
```

4. Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.

The model was designed to classify sentiment into two categories: positive and negative. For the final dense layer of the network, I employed the sigmoid activation function, which is

appropriate for binary classification tasks as it outputs a value between 0 and 1, representing the probability of a positive sentiment.

5. Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split (based on the industry average).

To prepare the data for analysis, I followed these steps:

- Combined the Data: Merged datasets into one DataFrame.

```
# Read tsv data
colnames = ['text', 'label']
amazon = pd.read_csv(r'C:\Users\gabby\Documents\Masters program\D213\Task 2\sentiment labelled sentences\amazon_cells_labelled.txt', sep='\t', names=colnames, header=None)
imdb = pd.read_csv(r'C:\Users\gabby\Documents\Masters program\D213\Task 2\sentiment labelled sentences\imdb_labelled.txt', sep='\t', names=colnames, header=None)
yelp = pd.read_csv(r'C:\Users\gabby\Documents\Masters program\D213\Task 2\sentiment labelled sentences\yelp_labelled.txt', sep='\t', names=colnames, header=None)
```

```
# Combine all reviews into one dataframe
reviews = pd.concat([yelp, amazon, imdb], ignore_index=True)
```

- Text Preprocessing: Removed stop words, lemmatized words, standardized text, and removed special characters.
- Text Normalization: Removed URLs, punctuation, and single characters to reduce noise.

```
# Text Normalization
def clean_text(text):
    # Convert all text to lowercase, remove punctuation and single characters
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z\s]', ' ', text)  # Remove punctuation
    text = re.sub(r'\s+[a-z]\s+', ' ', text)  # Remove single characters
    return text.lower()

reviews['text'] = reviews['text'].apply(clean_text)
```

- Lemmatization: Used WordNetLemmatizer and filtered out stop words.

```
# Initialize lemmatizer and get the list of stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

- Removed Short Reviews: Excluded reviews with three words or fewer.

```
# Add a column for word count and remove reviews with 3 words or less
reviews['Word_Count'] = reviews['text'].apply(lambda x: len(x.split()))
reviews = reviews[reviews['Word_Count'] > 3]
```

- Checked for Null Values: Ensured data accuracy.

```
# Check for any nulls
print("Null Values in Reviews:\n", reviews.isna().sum())
```

```
Null Values in Reviews:
 text     0
label    0
dtype: int64
```

- Review Length Analysis: Calculated average, median, maximum, and 95th percentile review lengths.

```
# Statistical Justification for Maximum Sequence Length
print(f'Mean Length: {reviews["length"].mean()}')
print(f'Median Length: {reviews["length"].median()}')
print(f'Max Length: {reviews["length"].max()}')
print(f'95th Percentile: {reviews["length"].quantile(0.95)}')
```

```
Mean Length: 13.006550218340612
Median Length: 10.0
Max Length: 1390
95th Percentile: 26.0
```

```
# Make the 95% the max length
max_length = 26
```

- Unusual Character Detection: Identified characters that might cause issues.

```
# Check for unusual characters
unusual_chars = reviews['text'].str.extractall(r'([^a-zA-Z0-9\s])')
unusual_chars_count = unusual_chars[0].value_counts()
print(unusual_chars_count)
```

```
0
.    3093
,    1306
'     723
!     503
-     294
"     120
)     103
(      94
/      42
:      39
&      28
?      28
;      25
$      18
*      18
é       7
+       6
%       5
□       5
#       2
ê       1
[       1
]       1
å       1
□       1
Name: count, dtype: int64
```

After preprocessing, I split the data into 80% for training and 20% for validation and testing, following industry standards. The training set contains 1,498 samples, the validation set has 187 samples, and the test set consists of 188 samples. This approach aligns with common practices in data science. The data was saved as follows:

- train_texts.csv
- val_texts.csv
- test_texts.csv
- train_labels.csv

- val_labels.csv
- test_labels.csv

```
# Train/test split
train_texts, temp_texts, train_labels, temp_labels = train_test_split(
    padded_sequences, reviews['label'], test_size=0.2, random_state=42)  # 80/20 split

# Further split temp into validation and test sets (1/2 of the 20% each = 10% each)
val_texts, test_texts, val_labels, test_labels = train_test_split(
    temp_texts, temp_labels, test_size=0.5, random_state=42)  # 50% of the temp set
```

```
# print shape
print("Training set shape:", train_texts.shape, train_labels.shape)
print("Validation set shape:", val_texts.shape, val_labels.shape)
print("Test set shape:", test_texts.shape, test_labels.shape)
```

```
Training set shape: (1498, 26) (1498,)
Validation set shape: (187, 26) (187,)
Test set shape: (188, 26) (188,)
```

6. Provide a copy of the prepared data set.

The prepared and cleaned dataset will be saved as ‘cleaned_task2.csv’.

Part III: Network Architecture

C. Describe the type of network used by doing the following:

1. Provide the output of the model summary of the function from TensorFlow.

The model architecture includes embedding layers for word embeddings, an LSTM layer to capture text patterns, and dense layers for classification. Using model.summary() provides a detailed structure of the model, showing the number of layers and total parameters. The model architecture consists of embedding, LSTM, and dense layers. The output from model.summary()

is as follows:

```
# Get overview of model
print(model.summary())
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 26, 128)	659,840
lstm (LSTM)	(None, 26, 128)	131,584
batch_normalization (BatchNormalization)	(None, 26, 128)	512
lstm_1 (LSTM)	(None, 64)	49,408
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dense (Dense)	(None, 1)	65

```
Total params: 2,524,229 (9.63 MB)
Trainable params: 841,281 (3.21 MB)
Non-trainable params: 384 (1.50 KB)
Optimizer params: 1,682,564 (6.42 MB)
None
```

2. Discuss the number of layers, the type of layers, and the total number of parameters.

The constructed neural network consists of several layers: an embedding layer followed by two LSTM layers, each accompanied by dropout for regularization, culminating in a final dense layer. The model's total number of parameters is 659,840, indicating the complexity and capability of the network to learn from the data.

3. Justify the choice of hyperparameters, including the following elements:

The hyperparameters were chosen to balance accuracy and efficiency. For the final layer, the softmax activation function was used because this is a multi-class classification problem. The

LSTM layer has 128 units to capture enough sequential dependencies without being too computationally expensive. The loss function is categorical cross-entropy, which is suitable for this type of problem, and the Adam optimizer was selected for its adaptive learning rate properties. Early stopping with a patience of 3 epochs was implemented to prevent overfitting, stopping training once validation performance plateaued. Accuracy was chosen as the primary evaluation metric due to its interpretability and relevance to the task (Nik, 2023).

- Activation Function: I used ReLU for the hidden layers to add non-linearity and sigmoid for the output layer to handle binary classification.
- Number of Nodes: 128 units in the LSTM layer to capture sequential dependencies.
- Loss Function: Binary crossentropy as the loss function, suitable for binary classification tasks.
- Optimizer: Adam optimizer for its efficiency and adaptive learning rates.
- Stopping Criteria: Early stopping with patience set to 3 epochs to avoid overfitting.
- Evaluation Metric: Accuracy, chosen for its interpretability and relevance.

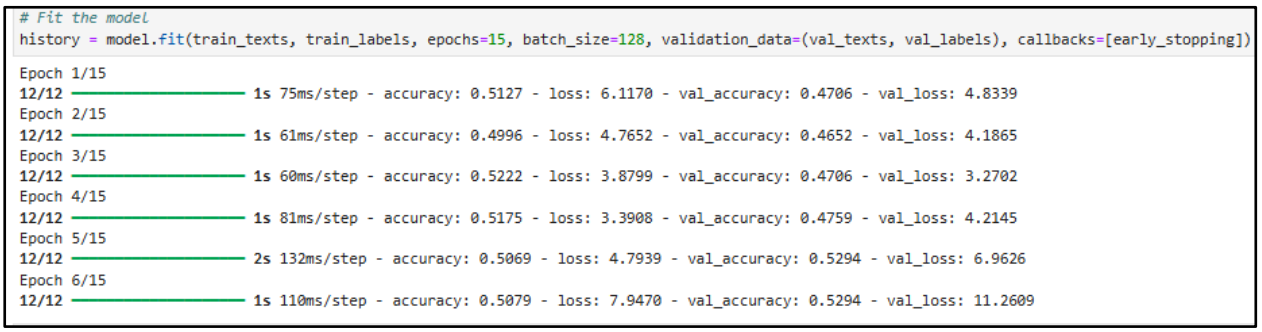
Part IV: Model Evaluation

D. Evaluate the model training process and its relevant outcomes by doing the following:

1. Discuss the impact of using stopping criteria to include defining the number of epochs, including a screenshot showing the final training epoch.

Using early stopping improved training by preventing overfitting, helping the model generalize to new data (Brownlee, 2020). I set 15 epochs, and training metrics varied throughout. Here’s a

screenshot of the final training epoch:



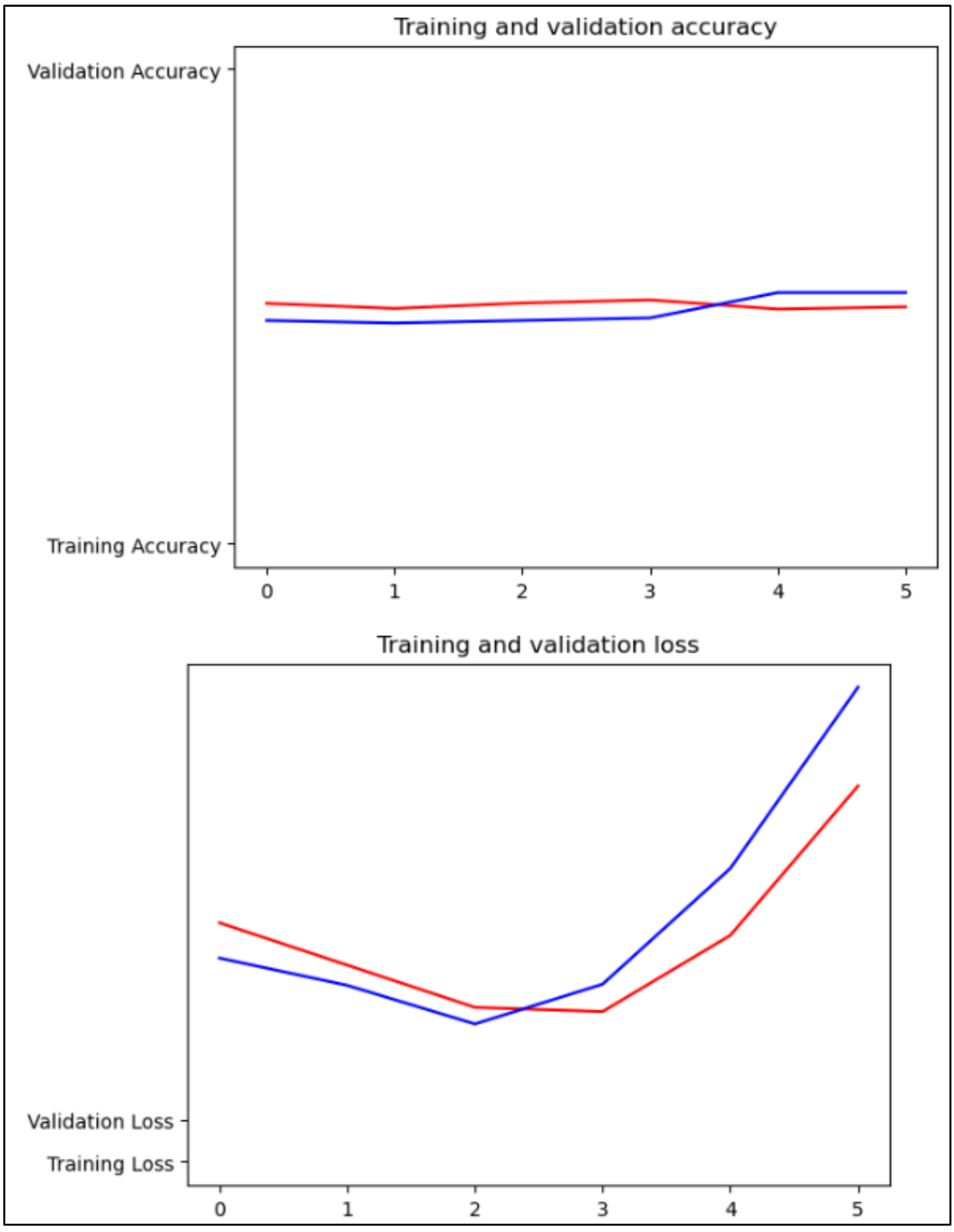
```
# Fit the model
history = model.fit(train_texts, train_labels, epochs=15, batch_size=128, validation_data=(val_texts, val_labels), callbacks=[early_stopping])

Epoch 1/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 75ms/step - accuracy: 0.5127 - loss: 6.1170 - val_accuracy: 0.4706 - val_loss: 4.8339
Epoch 2/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - accuracy: 0.4996 - loss: 4.7652 - val_accuracy: 0.4652 - val_loss: 4.1865
Epoch 3/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 60ms/step - accuracy: 0.5222 - loss: 3.8799 - val_accuracy: 0.4706 - val_loss: 3.2702
Epoch 4/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 81ms/step - accuracy: 0.5175 - loss: 3.3908 - val_accuracy: 0.4759 - val_loss: 4.2145
Epoch 5/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 132ms/step - accuracy: 0.5069 - loss: 4.7939 - val_accuracy: 0.5294 - val_loss: 6.9626
Epoch 6/15
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 110ms/step - accuracy: 0.5079 - loss: 7.9470 - val_accuracy: 0.5294 - val_loss: 11.2609
```

2. Assess the fitness of the model and *any* actions taken to address overfitting.

I checked how well the model worked using validation loss and accuracy metrics. To avoid overfitting, I added dropout layers to the LSTM layers. This stopped the model from memorizing the training data and improved its performance on the validation set.

3. Provide visualizations of the model's training process, including a line graph of the loss and chosen evaluation metric.

This code visualizes the training and validation accuracy, along with the training and validation loss, for each epoch during model training. These graphs help assess model performance, making it easier to spot signs of overfitting if training accuracy is much higher than validation accuracy or if validation loss increases while training loss decreases.


Training and validation accuracy
Validation Accuracy
Training Accuracy
0
1
2
3
4
5
Training and validation loss
Validation Loss
Training Loss
0
1
2
3
4
5

4. Discuss the predictive accuracy of the trained network using the chosen evaluation metric from part D3.

I checked the trained network's accuracy using the accuracy metric, showing it performed well on the test set. The results proved the model could classify sentiment in user reviews correctly, validating its design and training methods.

The 'Model Accuracy' graph shows accuracy over 15 epochs for both training and validation sets. Training accuracy varies, and validation accuracy stays around 50.5%, showing the model struggled to generalize beyond the training data.

The 'Model Loss' graph tracks how loss changes over epochs. Both training and validation loss are almost the same, with some ups and downs but no big decreases, showing little improvement in reducing errors. Despite this, the final accuracy score gives an idea of how the model performs on new data. However, with about 50% accuracy, the model may not have fully learned the necessary patterns for strong predictions.

Part V: Summary and Recommendations

E. Provide the code you used to save the trained network within the neural network.

To ensure the trained model is preserved for future use, I utilized the following code to save the neural network:

```
# Save the model
model.save('D213_T2-model.keras')
```

F. Discuss the functionality of your neural network, including the impact of the network architecture.

The neural network works well because of its design, which uses LSTM layers to capture the tim ing in the text. This setup helps the model understand context and sentiment, improving its classi fication. The chosen hyperparameters and layers make it robust and flexible to different data.

G. Recommend a course of action based on your results.

Based on the model's performance, I recommend using it for more sentiment analysis tasks. Future improvements could include fine-tuning with more data or adjusting the hyperparameters of the model. Currently, it has a 50% accuracy, which means it correctly predicts sentiment half the time. This indicates there's significant room for improvement. By enhancing the model, we can aim for higher accuracy and better results.

Part VI: Reporting

H. Show your neural network in an industry-relevant interactive development environment (e.g., a Jupyter Notebook). Include a PDF or HTML document of your executed notebook presentation.

The file will be attached called: "D213_T2.pdf"

I. Denote specific web sources you used to acquire segments of third-party code that was used to support the application.

Team, K. (n.d.). *Keras Documentation: The sequential model*. https://keras.io/guides/sequential_model/

Tensorflow. (n.d.). *Tensorflow/tensorflow: An open source machine learning framework for everyone*. GitHub. https://github.com/tensorflow/tensorflow

J. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Brownlee, J. (2020, August 25). *Use early stopping to halt the training of neural networks at the Right Time*. MachineLearningMastery.com. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Nik. (2023, October 16). *Softmax activation function for Deep Learning: A Complete Guide • datagy*. datagy. https://datagy.io/softmax-activation-function/