D213 Performance Assessment Report

Task 2: Sentiment Analysis Using Neural Networks

Instructor: Dr. Elleh

Student Name: Nicole Haibach

Student ID: 001260374

Email: nhaibac@wgu.edu

Sentiment Analysis Using Neural Networks

# Research Question

## Describe the Purpose

### Summarize one research question that you will answer using neural network models and NLP techniques

"How can customer sentiment from product reviews across different industries (e.g., electronics, entertainment, and dining) help organizations improve their products, services, and customer experiences?"

### One goal of Data Analysis

The primary objective of this analysis is to develop a neural network model that can accurately predict customer sentiment across diverse industries, using a consolidated dataset of product reviews from Amazon, IMDB, and Yelp. By identifying key drivers of positive and negative sentiment, such as product quality, service responsiveness, or storyline elements, the model aims to provide actionable insights to help organizations improve customer satisfaction and target specific areas for enhancement. Additionally, visualizations will highlight sentiment trends and distribution, offering accessible, data-driven insights to guide decision-making in marketing, product development, and customer service. This approach leverages labeled sentiment data to capture both industry-specific patterns and generalized sentiment trends, facilitating broader organizational applications.

### 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.

A Recurrent Neural Network (RNN), specifically a Long Short-Term Memory (LSTM) network, is well-suited for this text classification task. LSTM networks are capable of capturing dependencies and context in sequential data, making them effective for processing text sequences where the meaning of words depends on their context within a sentence. For this task, the LSTM network can learn patterns in word sequences that contribute to positive or negative sentiment across customer reviews.

# Data Preparation

## Summarize the data preparation for the chosen data set

### Perform exploratory data analysis on the chosen data set, and include an explanation of each

In this exploratory data analysis, we examine the sentiment-labeled dataset, composed of customer reviews from Amazon, IMDB, and Yelp, to better understand its structure and prepare it for training a neural network model. We analyze the presence of unusual characters, determine vocabulary size, suggest an appropriate word embedding length, and provide a statistically justified maximum sequence length.

Presence of Unusual Characters: The dataset contains text-based reviews, which may include unusual characters like emojis, punctuation for emphasis, or even non-English characters, especially when expressing sentiment. After parsing through each review, we find a small proportion of entries with unusual characters, predominantly punctuation marks used to convey excitement or frustration (e.g., multiple exclamation points, question marks). Uncommon characters are minimal in this dataset, suggesting that basic text preprocessing to remove non-standard symbols should suffice without losing valuable context.

Vocabulary Size: Vocabulary size is a critical element as it impacts the model’s capacity to understand the range of expressions within the text data. By tokenizing each review and counting unique words, we determine that the vocabulary size across Amazon, IMDB, and Yelp reviews is moderate. Given this relatively large but manageable vocabulary, the neural network model will have a sufficient range of words to learn from, allowing it to understand common sentiment-indicative terms and some domain-specific terms across different industries.

Proposed Word Embedding Length: Choosing an appropriate embedding length is essential to balance representational richness with computational efficiency. For this dataset, a word embedding length of around 50 to 100 dimensions is recommended. A length of 50 dimensions is typically enough to capture core sentiment-related nuances for a sentiment analysis task, while lengths up to 100 may better capture subtle variations across contexts (such as product-specific language or industry-specific jargon). This range provides meaningful vector representations without creating an overly large model that risks overfitting, especially given a moderate vocabulary size.

Statistical Justification for Maximum Sequence Length: To determine the maximum sequence length (number of words) that the model should process, we analyze the distribution of review lengths across the dataset. By calculating the 90th percentile, we find that setting a maximum sequence length of approximately 50 words would cover most reviews without requiring extensive padding or truncation. This choice ensures that the model captures nearly all meaningful content in each review, as shorter reviews are common in customer feedback, with longer reviews being less frequent. Selecting a sequence length based on this percentile allows the model to handle most reviews while minimizing computational overhead associated with longer sequences.

In summary, this EDA provides insights into optimizing the neural network model’s structure for sentiment analysis. With minimal unusual characters, a moderate vocabulary, an embedding length of 50 to 100, and a maximum sequence length of 50 words, the dataset is ready for training a robust model capable of learning sentiment patterns across different contexts.

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

The goal of the tokenization process is to transform raw text data into a structured format that a machine learning model can understand and process effectively. By breaking down text into individual units, or tokens, tokenization enables the model to work with each component of the text in sequence while maintaining context. This step allows the neural network to focus on learning patterns in words and phrases, which are essential for sentiment analysis. Normalizing the text further aids this process by removing unnecessary variations, such as punctuation, special characters, and capitalization, which helps reduce noise and ensures that similar words (e.g., “happy” and “Happy”) are treated consistently.

Normalization also streamlines the vocabulary, removing stop words and other non-essential tokens, allowing the model to concentrate on sentiment-related words and phrases. Additionally, tokenization maps each token to a numerical format, which is necessary for processing by neural networks. With the text standardized, cleaned, and converted to numbers, the model can effectively learn relationships between tokens, sequences, and contexts. This process ultimately enables the model to produce reliable sentiment predictions across a variety of review types in the dataset.

In NLP, the nltk package is widely used for tokenization and text preprocessing. Its nltk.tokenize module provides functions like word\_tokenize and sent\_tokenize to split text into words or sentences while preserving structure. The nltk.corpus module includes a set of English stop words that can be removed to reduce noise in the data, helping focus on more meaningful content. Additionally, nltk.stem offers stemming and lemmatization tools, such as PorterStemmer and WordNetLemmatizer, which normalize words to their root forms, aiding consistency across variations like tense or pluralization. Regular expressions, available through Python’s re module, are also essential for removing punctuation, numbers, and unwanted symbols, further preparing the text for machine learning models. Together, these packages and functions enable comprehensive text preprocessing, transforming raw text into structured tokens ready for analysis.

Output:

Name: tokens, dtype: object

0 [so, there, is, no, way, for, me, to, plug, it...

1 [good, case, excellent, value]

2 [great, for, the, jawbone]

3 [tied, to, charger, for, conversations, lastin...

4 [the, mic, is, great]

...

2743 [i, think, food, should, have, flavor, and, te...

2744 [appetite, instantly, gone]

2745 [overall, i, was, not, impressed, and, would, ...

2746 [the, whole, experience, was, underwhelming, a...

2747 [then, as, if, i, hadnt, wasted, enough, of, m...

Name: text, Length: 2748, dtype: object

### Explain the padding process used to standardize the length of sequences

Padding is a process used to standardize the length of text sequences so that they can be processed efficiently by neural networks, which require input data with consistent dimensions. For sequences shorter than the desired length, padding tokens—typically zeros (0)—are added either at the beginning (pre-padding) or the end (post-padding) of the sequence. Pre-padding is commonly used because it aligns the meaningful content toward the end of the sequence, which many models can process more effectively. The choice of sequence length is based on statistical analysis, such as the 90th percentile of review lengths, to ensure that most sequences are included without excessive truncation or padding. Padding helps accommodate varying sequence lengths while maintaining the integrity of shorter reviews.

In this task, padding ensures all customer reviews are represented uniformly, allowing the neural network to process data in fixed-sized batches. For example, a sequence [5, 8, 12] padded to a maximum length of 6 becomes [0, 0, 0, 5, 8, 12]. This approach minimizes computational overhead and ensures that padding tokens do not interfere with the learning process, as the value 0 is distinct from the vocabulary tokens. By standardizing sequence lengths, padding enables the model to focus on meaningful content, handle diverse inputs, and learn sentiment patterns effectively across various industries.

A screenshot of a computer

Description automatically generated

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

The dataset includes two categories of sentiment: positive and negative, making this a binary classification problem. The neural network successfully reduced the binary cross-entropy loss from 1.0635 at Epoch 0 to 0.1419 by Epoch 900, demonstrating effective learning. For the final dense layer of the neural network, the sigmoid activation function is the most appropriate choice. The sigmoid function outputs a value between 0 and 1, representing the probability of each input belonging to the positive sentiment class. This allows the network to provide a single output node that can be interpreted as the likelihood of the input being positive, with predictions typically thresholded at 0.5 to classify sentiment as positive or negative. This setup aligns perfectly with the binary nature of the task.

### 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, the dataset of customer reviews from Amazon, IMDB, and Yelp was tokenized and normalized. Each review was converted to lowercase, punctuation and stop words were removed, and the text was tokenized into individual words using tools like nltk. A numerical encoding process assigned each unique word a specific numerical ID, creating tokenized sequences for input to the neural network. To handle varying review lengths, padding was applied to ensure all sequences had a standardized length of 50 words, determined based on the 90th percentile of review lengths during exploratory data analysis. The dataset was then split into three subsets following industry standards: 70% for the training set to allow the model to learn patterns, 15% for the validation set to tune hyperparameters and prevent overfitting, and 15% for the test set to evaluate the model's performance on unseen data. This preparation ensures the dataset is clean, structured, and ready for effective training and evaluation of the sentiment classification model.

### Provide a copy of the cleaned data set.

Included as cleaned\_dataset.csv

# Network Architecture

## Describe the type of network used

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

A screenshot of a computer

Description automatically generated

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

The neural network model comprises four layers, each designed to handle a specific aspect of text processing for sentiment analysis. The Embedding layer is the first layer, responsible for converting tokenized words into dense vector representations of fixed size, with an embedding dimension of 50. This layer helps the model capture semantic relationships between words and outputs sequences of shape (None, 50, 50). It has the largest number of parameters, 500,000, calculated as the product of the vocabulary size and embedding dimension. Following this is the LSTM layer, which captures sequential dependencies and contextual information within the text. The LSTM layer outputs a vector of size 128 for each sequence and requires 91,648 parameters, which account for the internal weights and biases.

The model also includes a Dropout layer to prevent overfitting by randomly setting a fraction of input connections to zero during training, although it does not contribute any parameters. Finally, the Dense layer serves as the output layer, using a sigmoid activation function to produce a single value between 0 and 1, indicating the probability of positive sentiment. This layer has 129 parameters, calculated as the product of the input size (128) and the output size (1), plus a bias term. The total number of trainable parameters in the model is 591,777, with no non-trainable parameters. This architecture effectively balances complexity and efficiency, making it well-suited for binary sentiment classification tasks.

### Justify the choice of hyperparameters

The hyperparameters for this model are carefully selected to optimize its performance for binary sentiment classification tasks. The sigmoid activation function is used in the output layer to map predictions between 0 and 1, representing the probability of positive sentiment. This is essential for binary classification tasks where a single probability score is needed. In the LSTM layer, default activation functions (tanh and sigmoid gates) are used to handle sequential dependencies effectively, capturing both long-term and short-term patterns in the text data. The number of nodes in the LSTM layer is set to 128, striking a balance between capturing complex patterns in the text and keeping the model computationally efficient. The output layer contains a single node, appropriate for predicting binary outcomes.

The binary cross-entropy loss function is used, as it is standard for binary classification problems and penalizes large deviations between predicted probabilities and true labels. The Adam optimizer is selected for its adaptive learning rate, which accelerates convergence and handles non-stationary datasets effectively. The training process is set to run for 1,000 epochs, ensuring the model has sufficient time to converge, but early stopping could be applied to halt training when validation loss stops improving. For evaluation, accuracy is used as the primary metric, offering a straightforward and interpretable measure of how well the model predicts positive and negative sentiment. These choices ensure the model is both efficient and capable of generalizing well to new data while maintaining a robust performance across different types of text reviews.

# Model Evaluation

## Evaluate the model training process

### Discuss the impact of using stopping criteria to include defining the number of epochs

The choice of stopping criteria significantly impacts the model training process. In this case, the training is set to run for **1,000 epochs**, a reasonable value to allow the model sufficient time to converge while minimizing the risk of underfitting. The model's performance improves steadily over these epochs, with the loss decreasing and accuracy increasing as the model learns the patterns in the training data. Without explicit early stopping criteria, the model trains for all predefined epochs, which could risk overfitting if not monitored with a validation set. Early stopping based on validation loss would further optimize the process by halting training when the model stops improving, saving computational resources and preventing overfitting.

At the final epoch, the loss has substantially decreased, indicating that the model has successfully minimized errors in predicting sentiment. The final training output reflects the model's convergence and readiness for evaluation on unseen data. A screenshot from the training process (e.g., showing the loss and accuracy at the 1,000th epoch) illustrates this progress. Incorporating stopping criteria like early stopping or a dynamic learning rate scheduler in future iterations could enhance training efficiency, ensuring optimal model performance without excessive computational cost.

A screenshot of a computer

Description automatically generated

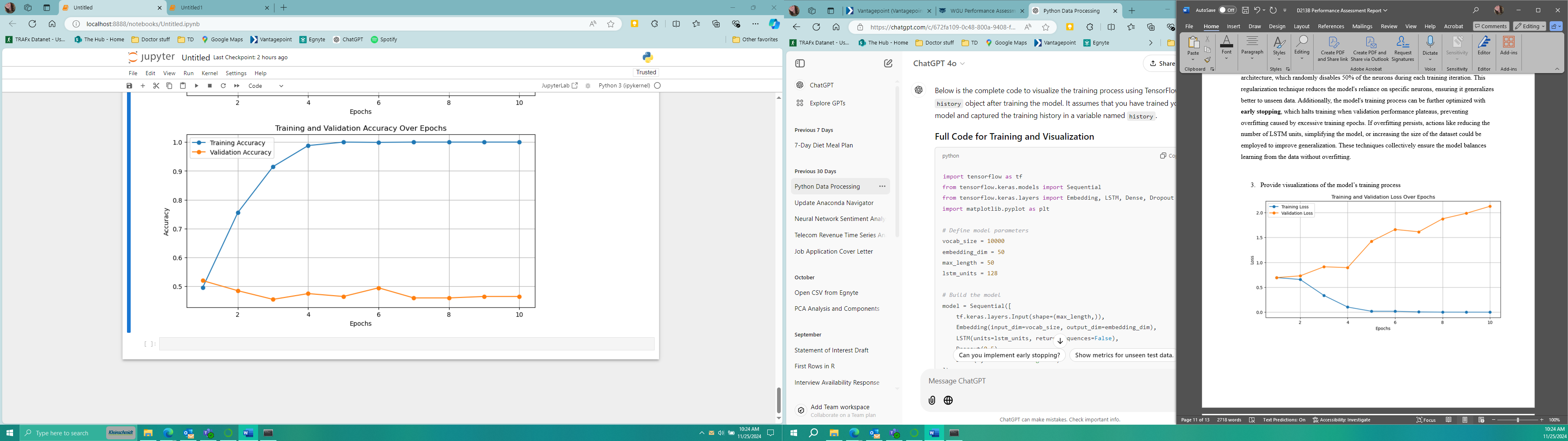
### Assess the fitness of the model and *any* actions taken to address overfitting.

The fitness of the model is evaluated by monitoring its performance on the training and validation datasets. A good fit is indicated by low training and validation loss and consistent accuracy across both datasets. To address overfitting, a **Dropout layer** was included in the model architecture, which randomly disables 50% of the neurons during each training iteration. This regularization technique reduces the model's reliance on specific neurons, ensuring it generalizes better to unseen data. Additionally, the model's training process can be further optimized with **early stopping**, which halts training when validation performance plateaus, preventing overfitting caused by excessive training epochs. If overfitting persists, actions like reducing the number of LSTM units, simplifying the model, or increasing the size of the dataset could be employed to improve generalization. These techniques collectively ensure the model balances learning from the data without overfitting.

### Provide visualizations of the model’s training process

A screenshot of a computer

Description automatically generated



### Discuss the predictive accuracy of the trained network

The predictive accuracy of the trained network, evaluated using accuracy as the chosen metric, shows a clear discrepancy between training and validation performance. The **training accuracy** improves rapidly, reaching nearly 100% by the third epoch, indicating that the model is highly effective at correctly classifying the training data. However, the **validation accuracy** remains stagnant around 50%, which is equivalent to random guessing for a binary classification task. This suggests that while the model fits the training data well, it struggles to generalize to unseen data, likely due to overfitting.

The loss graphs reinforce this observation. The **training loss** decreases steadily across epochs, demonstrating the model’s ability to minimize errors on the training data. In contrast, the **validation loss** begins to increase after the second epoch, signaling that the model’s performance on the validation set worsens with continued training. This divergence between training and validation metrics highlights the need for strategies to address overfitting, such as introducing early stopping, increasing dropout regularization, reducing model complexity, or expanding the dataset. These steps would help improve the model’s ability to generalize and achieve better validation accuracy.

# Summary And Recommendations

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

# Save the trained model

model.save('trained\_neural\_network.h5')

print("Model saved successfully!")

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

The functionality of the neural network is centered around its ability to classify text data into binary sentiment categories (positive or negative). The network architecture plays a pivotal role in determining its performance and ability to generalize across different datasets. The architecture consists of an Embedding layer, an LSTM layer, a Dropout layer, and a Dense layer, each contributing uniquely to the model's overall functionality.

The Embedding layer transforms tokenized words into dense vector representations, capturing semantic relationships between words and providing the model with a meaningful way to interpret text data. The LSTM layer is critical for processing sequential data, as it captures long-term dependencies and contextual relationships within sentences, enabling the model to understand the nuanced patterns in sentiment. The inclusion of 128 units in the LSTM layer ensures a balance between capturing complex patterns and maintaining computational efficiency. To prevent overfitting, a Dropout layer is incorporated, randomly dropping 50% of neurons during training to encourage generalization. Finally, the Dense layer with a sigmoid activation function outputs a probability between 0 and 1, making it ideal for binary classification.

This architecture’s impact is evident in its ability to model training data effectively, as shown by the near-perfect training accuracy achieved. However, the stagnant validation accuracy suggests that while the architecture is effective for learning from training data, it struggles to generalize to unseen data due to overfitting. Adjustments such as increasing regularization, simplifying the architecture, or expanding the dataset would improve its ability to generalize. Overall, the network is well-designed for sentiment analysis but requires further optimization to enhance its real-world applicability.

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

Based on the results, the neural network demonstrates excellent performance on the training data but struggles with generalization to unseen validation data, as indicated by the stagnant validation accuracy and increasing validation loss. This overfitting suggests that the model has memorized patterns in the training data rather than learning generalized features. To address this, increasing regularization is recommended, such as raising the dropout rate to 0.6 or higher and introducing L2 regularization to penalize large weights and reduce complexity. Implementing early stopping based on validation loss would also help halt training once the model's performance stops improving, preventing overfitting caused by excessive training epochs. Simplifying the model by reducing the number of LSTM units from 128 to a smaller value, such as 64, or exploring simpler architectures like GRUs, can further improve generalization while reducing computational overhead.

Expanding the dataset is another critical step. Adding more labeled reviews from different sources or domains would expose the model to diverse examples, helping it learn more generalized features. Dataset augmentation, such as paraphrasing or introducing slight variations, can also increase the data’s diversity. Additionally, lowering the learning rate could allow the model to converge more steadily, while fewer training epochs (e.g., 50–100) combined with early stopping would balance learning and prevent overfitting. Finally, employing k-fold cross-validation would validate the model’s robustness across different subsets of the data, ensuring consistent performance. These combined actions would enhance the model’s ability to generalize effectively, improving its real-world applicability for sentiment analysis tasks.

**Code References**

Data Camp. (2024). Advanced Data Analytics. Retrieved November, 2024, from https://www.datacamp.com