Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu
2024

Seminar Report on
"Sentiment Analysis Using Gradient Recurrent Unit"

In partial fulfillment of the requirement for a Master's degree in Computer Science and Information Technology (M.Sc. CSIT), 1st Semester

Submitted to:
Central Department of Computer Science and Information Technology, Tribhuvan University, Kirtipur, Kathmandu, Nepal

Submitted By:
Sagar Timalsena (8015031)

Date: July 2024

ACKNOWLEDGEMENT

I sincerely thank everyone who has helped with technical and informational support and guidance.

I begin by extending my gratitude to my mentor **Mr. Bikash Balami**, Assistant Professor, Central Department of Computer Science and Information Technology (CD.CSIT) Tribhuvan University for his continuous guidance and support during the research and learning period. Without his guidance, this research would not have been successful all staff members for their cooperation and guidance during the entire research period. The success and outcome of this research required a lot of advice and assistance from many people and I am extremely fortunate to have them all along the completion of this research period. Such guidance was indispensable for the accomplishment of this endeavor.

I am thankful to my supervisor for providing me with his continuous guidance, advice, motivation, and knowledge he imparted to me regarding every perspective of the report. I also thank the Central Department of Computer Science and Information Technology MSc.CSIT department and library for providing me with the necessary resources during the time of research and report.

I would also like to acknowledge, my parents, for providing financial and psychological support throughout my study and all those who contributed directly and indirectly.

Thank You

Sagar Timalsena

(TU Roll No.: 8015031)

ABSTRACT

Sentiment analysis is crucial in natural language processing (NLP) to determine whether text expresses positive, negative, or neutral sentiments. This study evaluates sentiment analysis using a Gated Recurrent Unit (GRU) neural network, focusing on tweets about US Airlines. The GRU-based model demonstrated robust performance, achieving an accuracy of 82.61%, precision of 80.84%, recall of 85.53%, F1 score of 79.66%, and specificity of 80.33%. The model's performance was analyzed using a confusion matrix to understand the distribution of correct and incorrect classifications. Results were visualized through heatmaps and accuracy/loss plots to illustrate model performance and training progress. The findings highlight the GRU model's effectiveness in sentiment classification, showcasing its potential for handling complex NLP tasks and providing valuable insights for future research and practical applications.

Keywords: *Language Semantics, Natural Language Processing (NLP), Text Analysis, Textual Meaning Extraction*

TABLE OF CONTENTS

List of Figure

List of Abbreviations

GRU	Gated Recurrent Unit
IDE	Integrated Development Environment
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
RNN	Recurrent Neural Network

Chapter 1: Introduction

1.1 Introduction

Sentiment analysis, also referred to as opinion mining, is a computational process that involves analyzing and interpreting the emotions and opinions expressed in textual data. In today's digital age, where the amount of textual content generated online is growing exponentially, sentiment analysis has become increasingly important across various domains. Understanding public opinion, consumer feedback, and social media trends is critical for businesses, governments, and organizations to make informed decisions and stay relevant in a highly competitive landscape.

Traditional methods of sentiment analysis often rely on simplistic approaches such as rule-based systems or bag-of-words models. While these methods can provide basic sentiment classification, they often fail to capture the nuances and context-dependent nature of language. For example, a simple approach might classify the word "good" as positive, but it might overlook subtle variations in meaning, such as "good" in the context of being sarcastic or as part of a negation ("not good"). In recent years, deep learning techniques, particularly recurrent neural networks (RNNs), have emerged as powerful tools for sentiment analysis. RNNs are a class of artificial neural networks designed to effectively process sequential data, making them well-suited for tasks involving natural language processing.

Gated recurrent units (GRUs) are a type of RNN architecture that has gained popularity for sentiment analysis tasks due to their simplicity and effectiveness. GRUs address some of the limitations of traditional RNNs, such as the vanishing gradient problem, by introducing gating mechanisms that control the flow of information through the network.

By leveraging deep learning techniques like RNNs and GRUs, sentiment analysis models can achieve higher levels of accuracy and robustness, enabling more nuanced and context-aware sentiment analysis. This, in turn, empowers businesses, researchers, and policymakers to gain deeper insights into public opinion, consumer sentiment, and social media dynamics, ultimately facilitating more informed decision-making and strategic planning.

1.2 Problem Statement

As the amount and intricacy of textual data continue to grow, accurately deciphering sentiments presents a significant challenge. Current sentiment analysis models often falter in grasping contextual cues and subtle emotional nuances, resulting in less-than-ideal outcomes. Therefore, there's a pressing demand for resilient and effective sentiment analysis frameworks adept at

managing vast quantities of text data and precisely categorizing sentiments with heightened accuracy.

1.3 Objective

The primary objective of this study is to develop a sentiment analysis framework using gated recurrent units (GRUs), a variant of recurrent neural networks, to effectively classify sentiments in text data. The primary objectives of this study are:

- To implement a GRU-based sentiment analysis model
- To evaluate the performance of the model using appropriate metrics
- To integrate necessary pre-processing steps to clean and prepare text data for model training.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Sentiment analysis, [1] or opinion mining has evolved from early rule-based systems and bag-of-words models, which struggled with context and nuance, to advanced deep learning techniques that capture complex patterns in textual data. Initial methods like rule-based systems relied on predefined rules and lexicons, while bag-of-words models represented text as unordered word collections, both of which lacked contextual understanding. The introduction of machine learning techniques, such as Support Vector Machines and Naive Bayes classifiers, improved performance but still depended on manual feature extraction. The shift to deep learning brought significant advancements, with Recurrent Neural Networks (RNNs) and their variants, like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), offering enhanced capabilities for handling sequential data and capturing long-range dependencies. Recent innovations, including attention mechanisms, Transformers like BERT, and transfer learning, have further refined sentiment analysis by improving accuracy, context-awareness, and interpretability, setting new benchmarks in the field.

2.1.1 Recurrent Neural Network

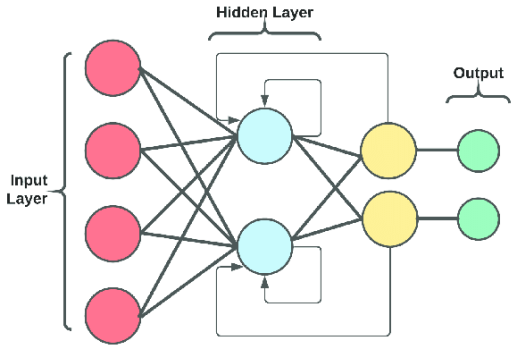

Figure 1: Recurrent Neural Network Architecture

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to handle sequential data by maintaining a memory state. Unlike feedforward neural networks, which process data instantaneously, RNNs can retain information about previous inputs, making them well-suited for tasks such as language modeling, time series prediction, and speech recognition. However, traditional RNNs suffer from the vanishing gradient problem, which hampers their ability to capture long-term dependencies in sequential data.

2.1.2 Gated Recurrent Unit

Gated Recurrent Units (GRUs) are a type of recurrent neural network (RNN) architecture introduced to overcome some limitations of traditional RNNs, such as difficulties in learning long-range dependencies due to the vanishing gradient problem. GRUs achieve this by incorporating gating mechanisms that regulate the flow of information through the network over time.

A GRU works in the following ways:

1. Hidden State(ht):

Like traditional RNNs, a GRU maintains a hidden state vector h_t at each time step t. This hidden state captures information about the sequence of inputs processed so far.

2. Update Gate(z_t):

At each time step, a GRU computes an update gate z_t that determines ow much of the previous hidden state h_{t-1} to retain and how much new information to incorporate from the current input xt. The update gate is computed using a sigmoid activation function, which squashes the values between 0 and 1.

$$z_t = \sigma\,(W_z\,.\,[h_{t-1}\,,\,x_t])$$

3. Reset Gate (r_t):

Additionally, a reset gate rt is computed at each time step to control how much of the previous hidden state ht-1 should be forgotten. Similar to the update gate, the reset gate is also computed using a sigmoid activation function. It controls the contribution of the previous hidden state to the candidate activation.

$$r_t = \sigma\,(W_r\,.[h_{t-1}\,,\,x_t])$$

4. Candidate Hidden State ($\tilde{h}_t$):

This represents the new candidate content that will be added to the current state. The reset gate controls how much of the previous state is included in the new candidate.

$$\tilde{h}_t = \tanh\,(W_h\,.\,[r_t * h_{t-1}\,,\,x_t])$$

5. Update Hidden State (ht):

The final hidden state is a combination of the previous hidden state and the new candidate activation, controlled by the update gate. This allows the GRU to decide how much of the new information to incorporate into the hidden state while retaining information from previous time steps.

$$h_t = (1 - Z_t) * h_{t-1} + Z_t * \tilde{h}_t$$

Where:

- σ is the sigmoid function.
- Wz , Wr , W_h are weight matrices for the update and reset gates, respectively.
- h_{t-1} is the previous hidden state.
- x_t is the current input.
- h˜t is the candidate activation.
- * represents elements-wise multiplication and . denotes concatenation.

2.2 Literature Review

Sentiment analysis[2] is a crucial tool for understanding public sentiment about various topics. This study examined different methods for text sentiment analysis, including dictionary-based and machine-learning techniques, focusing on adapting these methods for Twitter data. This adaptation[3] posed several challenges due to the unique nature of Twitter's text data, such as brevity and informal language.

Sentiment analysis[4] is widely utilized in business to enhance products by comprehending customer opinions. The advent of deep learning has led to significant advancements in this field. This study compared various deep learning models for sentiment analysis, specifically[5] three models based on convolutional neural networks (CNNs) and five based on recurrent neural networks (RNNs). Two types of input word level and character level were used to evaluate the performance of these models.

The study also investigated how different settings, such as dropout rates, number of layers, and activation functions, influenced the models' performance. Additionally, the research compared [6]the effectiveness of long short-term memory (LSTM) networks and gated recurrent units

(GRUs). It was found that GRUs generally outperformed LSTMs in predicting data, particularly in a case study on paper production equipment. GRUs emerged as the superior option for maximizing equipment availability.

Furthermore, the study highlighted the importance of optimizing model settings to achieve the best results in sentiment analysis. Fine-tuning[5] parameters like dropout rates, layer numbers, and activation functions can significantly enhance model performance, making GRUs reliable for applications requiring precise predictions.

Chapter 3: Methodology

Sentiment Analysis of Twitter US Airline Sentiment review is done using GRU recurrent neural network. The detailed architecture of the system is presented below.

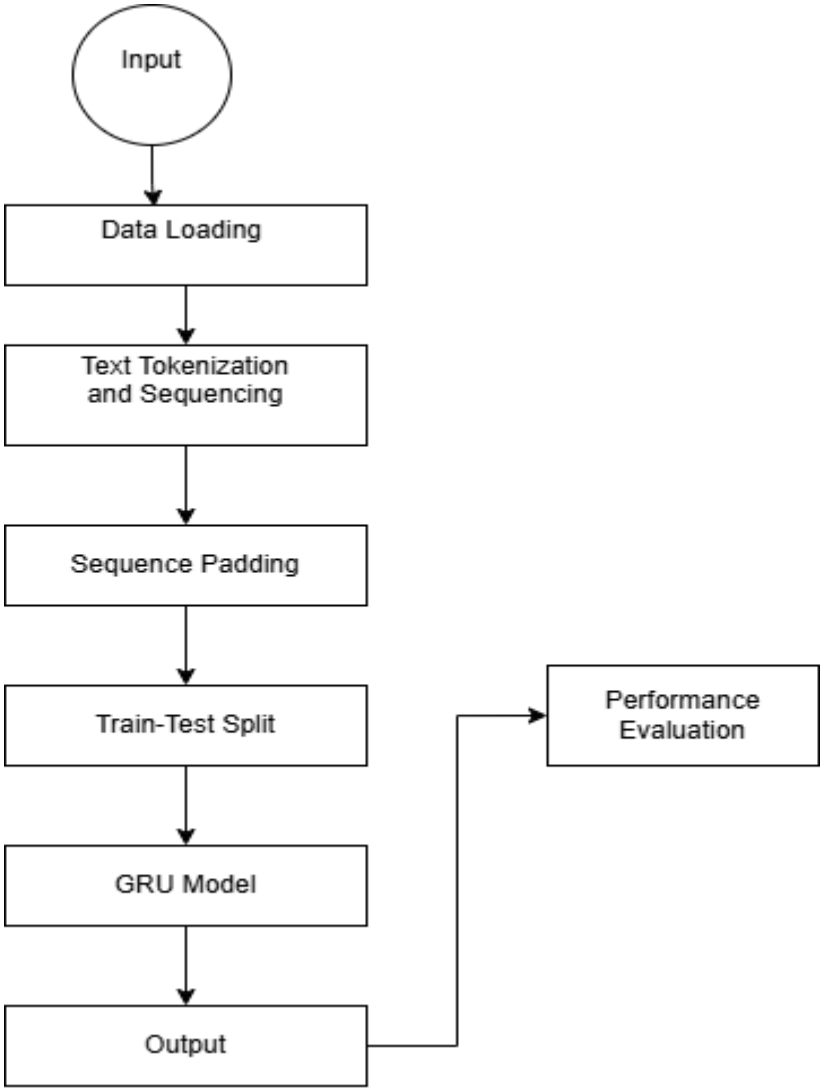

Figure 2 Architecture of Sentiment Analysis

Each step in the architecture shown in the figure is explained in detail as follows:

3.1 Dataset Description

Input data for this project i.e. Twitter US Airline Sentiment Dataset is taken from Kaggle an online repository known for its diverse collection of datasets. The dataset consists of over 7000 data.

```
Tweet ID: 2401
Content: im getting on borderlands and i will kill you all,

Tweet ID: 2401
Content: im coming on borderlands and i will murder you all,

Tweet ID: 2401
Content: im getting on borderlands 2 and i will murder you me all,

Tweet ID: 2401
Content: im getting into borderlands and i can murder you all,

Tweet ID: 2402
Content: So I spent a few hours making something for fun. . . If you don't know I am a HUGE @Borderlands fan and Maya is one of my favorite characters. So I decided to mak
e myself a wallpaper for my PC. . Here is the original image versus the creation I made :) Enjoy! pic.twitter.com/mLsI5wf9Jg

Tweet ID: 2402
Content: So I spent a couple of hours doing something for fun... If you don't know that I'm a huge @ Borderlands fan and Maya is one of my favorite characters, I decided t
o make a wallpaper for my PC.. Here's the original picture compared to the creation I made:) Have fun! pic.twitter.com / mLsI5wf9Jg

Tweet ID: 2402
Content: So I spent a few hours doing something for fun... If you don't know I'm a HUGE @ Borderlands fan and Maya is one of my favorite characters.

Tweet ID: 2402
Content: So I spent a few hours making something for fun. . . If you don't know I am a HUGE RhandlerR fan and Maya is one of my favorite characters. So I decided to make m
yself a wallpaper for my PC. . Here is the original image versus the creation I made :) Enjoy! pic.twitter.com/mLsI5wf9Jg

Tweet ID: 2402
Content: 2010 So I spent a few hours making something for fun. . . If you don't know I am a HUGE RhandlerR fan and Maya is one of my favorite characters. So I decided to m
ake myself a wallpaper for my PC. . Here is the original image versus the creation I made :) Enjoy! pic.twitter.com/mLsI5wf9Jg
```

Figure 3 Dataset Sample

3.2 Text Tokenization and Sequencing

In sentiment analysis using GRU, tokenization breaks the text into tokens that are converted into numerical forms. These tokens are then arranged into fixed-length vectors, which helps the model understand the context and dependencies within the text. Proper preprocessing enhances sentiment prediction accuracy by allowing GRU models to effectively capture subtle expressions of sentiment.

```
Tokenized Sequences:
Text: Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
Tokenized: [49, 472, 4427, 841, 756, 659, 65, 8, 1328, 87, 123, 352, 1329, 148, 2993, 1330, 67, 58, 4428, 144]
Text: Ok lar... Joking wif u oni...
Tokenized: [46, 337, 1500, 473, 6, 1941]
Text: Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075
over18's
Tokenized: [47, 490, 8, 19, 4, 797, 902, 2, 175, 1942, 1106, 660, 1943, 2331, 261, 2332, 71, 1942, 2, 1944, 2, 338, 490, 556, 961, 73, 391, 179, 661, 39
2, 2994]
```

Figure 4 Tokenized Sequence

```
Padded Sequences:
Text: Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
Padded Sequence: [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0   49  472
 4427  841  756  659   65    8 1328   87  123  352 1329  148 2993 1330
   67   58 4428  144]
```

Figure 5 Padded Sequence

3.3 Padding

Padding is a crucial technique used to ensure that all tweet sequences have a consistent and fixed length before being input into a GRU model for training or evaluation. This is achieved using the Keras `pad_sequences` function. Tokenized words, representing tweets of varying lengths, are padded with zeros up to a predefined maximum sequence length of 200 characters. Padding is applied at the end of sequences ('post'), which extends shorter tweets with zeros while preserving their original content. This standardized sequence length is essential for efficient processing and modeling of text data with GRU, ensuring uniform input dimensions regardless of the original tweet length.

```
--------------------------------------------------
Text: Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075
over18's
Padded Sequence: [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0   47  490    8   19    4  797  902    2  175 1942 1106  660 1943
 2331  261 2332   71 1942    2 1944    2  338  490  556  961   73  391
  179  661  392 2994]
Sequence Length: 200
--------------------------------------------------
```

Figure 6 Padded Sequence of Length size 200

3.4 Train-test split

Splitting data into training and testing sets is essential in machine learning. This process involves dividing the dataset into two parts: the training set and the testing set. Typically, the training set comprises 80% of the data and is used to train the GRU model by providing examples for learning. The remaining 20% is reserved as the testing set, used to evaluate the model's performance.

This split is crucial because it allows for assessing how well the model generalizes to new, unseen data. By testing the model on this separate dataset, one can determine how accurately it predicts sentiments in tweets it has not encountered before. This evaluation helps understand whether the model can make reliable predictions in real-world situations.

3.5 GRU Model

The GRU model for sentiment analysis in short text messages includes several key components:

Embedding Layer: This layer transforms tokenized tweet sequences into dense vectors, helping the model understand the context and meaning of words within each tweet.

GRU Layer: The Gated Recurrent Unit (GRU) layer captures temporal dependencies and contextual information from the sequences, making it effective for learning patterns in tweet texts.

Dropout Layer: To improve generalization and prevent overfitting, this layer randomly deactivates a fraction of neurons during training, helping the model generalize better to new, unseen tweets.

Dense Layer: The output layer uses a sigmoid activation function to produce a probability score, indicating the likelihood of a tweet expressing positive sentiment. The sigmoid function ensures the output is between 0 and 1, reflecting the model's confidence in its sentiment prediction.

3.6 Performance Evaluation

To evaluate the sentiment analysis model, predictions are made on the test dataset using the trained model. These predictions are compared with the actual labels to create a confusion matrix, which shows true positives, true negatives, false positives, and false negatives. This helps in understanding how well the model is performing. Key metrics from this matrix include accuracy, which measures overall correctness, precision, which shows the proportion of correct positive predictions, recall, which indicates how well positive sentiments were captured, and the F1 score, which balances precision and recall.

Specificity measures how well the model identifies negative sentiments. The loss metric shows classification errors, indicating where the model made mistakes. Training and validation loss plots, along with accuracy metrics over time, help track the model's learning progress and identify issues like overfitting.

Chapter 4: Implementation

4.1 Implementation

The program is developed using Python (version 3.12.4) as the primary programming language within the VS Code Integrated Development Environment (IDE). Several essential libraries are employed:

i. **NumPy:** Utilized for efficient numerical operations and computations.
ii. **Pandas:** Employed for comprehensive data manipulation and analysis tasks.
iii. **Matplotlib:** Utilized for creating static, animated, and interactive visualizations in Python.
iv. **Seaborn:** Used for statistical data visualization, emphasizing attractive and informative statistical graphics.
v. **TensorFlow and Keras:** These frameworks are utilized for constructing, training, and deploying deep learning models.
vi. **Scikit-learn:** Employed for machine learning tasks including preprocessing, model evaluation, and metrics computation.

Together, these libraries enable robust development, analysis, visualization, and deployment of machine learning and deep learning solutions in Python.

4.2 Implementation details

The implementation details encompass the following aspects:

4.2.1 Train–Test Split

The data set is split into two independent sets: a training set and a test set, in an 80:20 ratio 80% of the data goes into training and 20% into testing. This partition will ensure that the model has learned from a significant portion of the dataset and is tested on completely new, unseen data samples; that is, the evaluation helps check how well the model generalizes sentiments.

```
# Split the data into features (X) and target (y)
X = df['text']  # Features: text data
y = df['target']  # Target: numeric sentiment labels

# Perform train-test split with a ratio of 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Print the shapes of the training and testing sets to verify the split
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

Figure 7 Perform train-test split with a ratio of 80:20

4.2.2 Text Sequencing and Padding

The text data was transformed into sequences of words using a tokenizer, and each sequence was padded with zeros to have a fixed length of 300 words (SEQUENCE_LENGTH). This step ensured that all sequences had consistent dimensions for efficient processing.

```
# Parameters
# Fixed length for padding
SEQUENCE_LENGTH = 300

# Tokenization
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

# Padding sequences
padded_sequences = pad_sequences(sequences, maxlen=SEQUENCE_LENGTH,
                                 padding='post', truncating='post')
```

Figure 8 Sequencing and padding

4.2.3 GRU Architecture

The neural network architecture for sentiment analysis starts with an Embedding layer, followed by a GRU (Gated Recurrent Unit) layer with 64 units to process the sequence data and capture temporal dependencies, producing a fixed-length output sequence. To mitigate overfitting, a Dropout layer with a rate of 0.2 is included, which randomly sets a portion of the

input units to zero during training. The final layer is a Dense layer with a single neuron and a sigmoid activation function, providing the output for binary classification.

```
# Build GRU model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=SEQUENCE_LENGTH))
model.add(GRU(units=gru_units, dropout=0.2, recurrent_dropout=0.2))
# Sigmoid activation for binary classification
model.add(Dense(1, activation='sigmoid'))
```

Figure 9 GRU Architecture

4.2.4 Model Training and Adam Optimizer

Before training, the model is compiled by specifying the optimizer, loss function, and evaluation metrics. The Adam optimizer is chosen for its ability to adapt learning rates for each parameter, which can enhance convergence speed. Binary cross-entropy is selected as the loss function, and accuracy is used to evaluate performance. The model is trained over a set number of epochs using mini-batches of data with a batch size of 512. A validation dataset is utilized to monitor performance. During training, the Adam optimizer updates the model's weights based on the gradients of the loss function, helping the model learn to distinguish between positive and negative sentiments.

```
# Compile model with Adam optimizer
# set learning rate
optimizer = Adam(learning_rate=1e-4)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 10 Compiling the model

```
# Define early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
```

Figure 11 Early stopping to prevent overfitting

4.2.5 Model Evaluation

After training, the model's performance was evaluated using the test dataset. Metrics including accuracy, precision, F1 score, specificity, and recall were calculated from the confusion matrix to assess its effectiveness in classifying positive and negative sentiments.

```
# Calculate additional metrics
precision = precision_score(y_test_labels, y_pred, average='weighted')
recall = recall_score(y_test_labels, y_pred, average='weighted')
f1 = f1_score(y_test_labels, y_pred, average='weighted')
# Specificity calculation for each class
cm = confusion_matrix(y_test_labels, y_pred)
specificity = np.diag(cm) / np.sum(cm, axis=1)

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-score: {f1:.4f}')
print(f'Specificity: {specificity.mean():.4f}')
```

Figure 12 Model Evaluation

Chapter 5: Result and Findings

5.1 Overfitting and Generalization

The training and validation loss and accuracy plots displayed a smooth convergence during the training process. There were no significant signs of overfitting or underfitting, suggesting that the model effectively learned from the data without memorizing it.

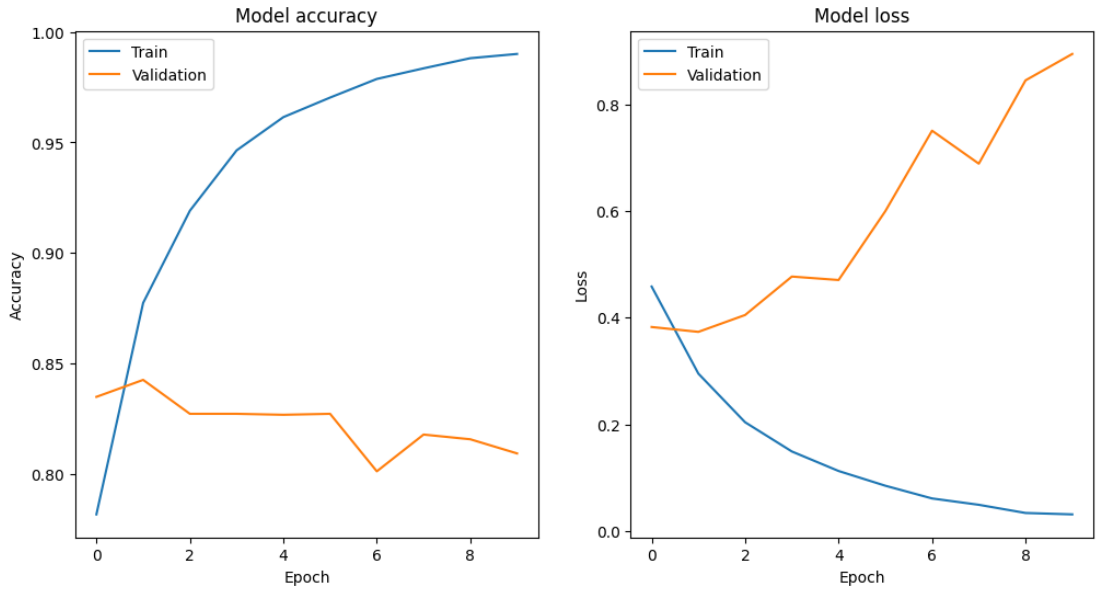

Figure 13 Training and Validation Loss and accuracy plots

5.2 Model Performance Measure

The Sentiment Classifier using the GRU model demonstrated moderate performance in distinguishing between positive and negative sentiments. After training the model for 5 epochs with a batch size of 64, the following evaluation metrics were obtained on the entire dataset:

- Accuracy: 82.61%
- Precision: 80.84%
- Recall: 85.53%
- F1 Score: 79.66%
- Specificity: 80.33%

These metrics indicate that while the model achieved high accuracy in correctly classifying sentiments as positive or negative, the precision score reflects a lower ability to correctly identify positive and negative sentiments. The recall score indicates its moderate ability to identify all the actual positive sentiments present in the dataset, capturing some of the true positive sentiments. The F1 score represents the model's balance between precision and recall, reflecting its ability to achieve both accurate positive sentiment identification and comprehensive capturing of actual positive sentiments, albeit with room for improvement. The specificity score highlights the model's skill in identifying true negatives (negative sentiments) within the dataset, ensuring that a portion of the negative sentiments are correctly classified.

5.3 Confusion Matrix

The confusion matrix results provide a detailed view of the classification performance of the model. In this specific scenario, the model correctly identified 3471 instances as true positives, accurately recognizing them as positive sentiments. It also correctly classified 3138 instances as true negatives, signifying its ability to identify negative sentiment correctly. However, the model had 828 false positives, indicating instances where negative sentiments were mistakenly classified as positive. Additionally, there were 563 false negatives, highlighting cases where positive sentiments were misclassified as negative

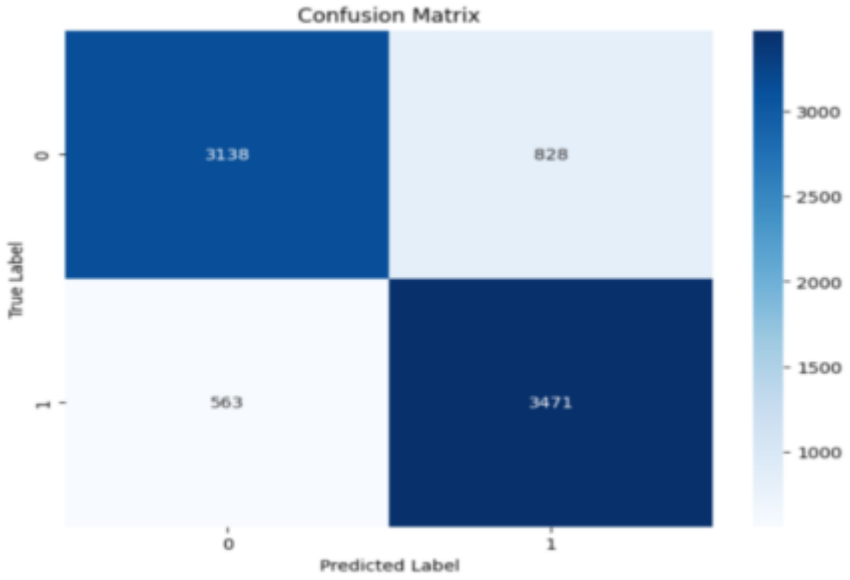

Figure 14 Confusion Matrix

The model is performing well, with high accuracy, precision, F1 score, specificity, and recall. The loss is also relatively low. This suggests that the model can learn the patterns of positive and negative sentiments and classify them correctly with a high degree of confidence.

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

The sentiment analysis implemented for the US airline review tweets dataset leverages the Gated Recurrent Unit (GRU) architecture to effectively analyze and classify sentiments. The GRU model excels in capturing the intricacies of language and distinguishing between positive, negative and neutral sentiments expressed in the tweets. The model demonstrates a commendable accuracy of 82.61%, signifying its overall effectiveness in correctly predicting sentiments. The precision of 80.84% indicates that a substantial proportion of the predicted positive sentiments are indeed accurate. With a recall rate of 85.53%, the model is proficient at identifying most of the actual positive sentiments within the dataset. The F1 score of 79.66% provides a balanced measure of precision and recall, reflecting a well-rounded performance. Additionally, the specificity of 80.33% reveals the model's capability to correctly identifying negative sentiments.

6.2 Future Recommendation

To improve the model's performance in the future, several steps can be taken. First, try adjusting different settings, like the number of GRU units and learning rates, to find the best configuration. Expand the training data using techniques that create more examples from existing data. Adding attention mechanisms can help the model focus on important parts of the text. Using transfer learning with BERT can improve how the model understands complex language. Combining several models through ensemble methods can enhance overall performance. Test the model's ability to adapt to different contexts through domain adaptation. Evaluate how well the model works in real-time scenarios, such as monitoring social media. Lastly, create ways to explain the model’s predictions to increase trust and transparency. These strategies can help make the sentiment analysis model more robust, accurate, and reliable.

References

[1] A. D. T. and J. A., "Multimodal Sentimental Analysis Using Hierarchical Fusion Technique," in *2023 IEEE 4th Annual Flagship India Council International Subsections Conference(INDISCON)*, 2023, pp. 1–8. doi: 10.1109/INDISCON58499.2023.10270094.

[2] K. S. Jones, "Natural language processing: a historical review," *Curr. issues Comput. Linguist. honour Don Walk.*, pp. 3–16, 1994.

[3] P. Acharya and B. K. Bal, *A Comparative Study of SMT and NMT: Case Study of English-Nepali Language Pair*. 2018. doi: 10.21437/SLTU.2018-19.

[4] B. Krishnamurthy, S. Senthilkumar, A. Singh, and P. Sharma, *Sentiment Analysis with NLP on Twitter Data*. 2022. doi: 10.1109/SMARTGENCON56628.2022.10084036.

[5] K. Sharma, D. Jain, A. Jain, Y. S. Rathore, and S. Agarwal, "Sentimental Analysis based on Machine Learning Technique," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, 2024, pp. 1–7. doi: 10.1109/INOCON60754.2024.10511973.

[6] A. M. Turing, *Computing machinery and intelligence*. Springer, 2009.