COMPARATIVE STUDY OF DEEP LEARNING MODELS FOR NATURAL LANGUAGE PROCESSING

A Major Project Report

Submitted in partial fulfillment of the requirements

for the award of the degree of

MASTER OF SCIENCE
(Computer Science)

Submitted by

RAMGANESH P

[Reg.No:20224012506118]

Under the Guidance of

Dr. D. MURUGAN M.E., Ph.D.,

Professor

Department of Computer Science and Engineering

Manonmaniam Sundaranar University

Tirunelveli-627012

April-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MANONMANIAM SUNDARANAR UNIVERSITY

TIRUNELVELI-12

BONAFIDE CERTIFICATE

This is to certify that the Major Project entitled **"COMPARATIVE STUDY OF DEEP LEARNING MODELS FOR NATURAL LANGUAGE PROCESSING"** submitted in partial fulfillment of the degree in MASTER OF SCIENCE (Computer Science) to the Department of Computer Science & Engineering, Manonmaniam Sundaranar University, done by **RAMGANESH P** Register No. **20224012506118** is an authentic work carried outduring the academic year 2023-2024.

GUIDE **HEAD OF THE DEPARTMENT**

Submitted for the Viva-Voce Examination on:_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I thank Lord Almighty, who blessed me with wonderful faculties, friends, and family members whose love and encouragement have given new significance to my accomplishment.

I express my sincere thanks with love to my parents, who are ardentcheer leader and support throughout the project. Their dedication helped me in many subtle ways which are indeed instrumental in achieving the final outcome.

I whole-heartedly thank **Dr. A. SURULIANDI M.E., Ph.D.,** the Head of the Department of Computer Science and Engineering, Manonmaniam Sundaranar University for providing us with constructive criticism and suggestions for the successful completion of this project.

I thank **Dr.V. SUBHA B.E., M.E., Ph.D.,** Project coordinator, Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli for her consistent encouragement throughout the entire project period.

I extend my gratitude to my guide **Dr.D.MURUGAN M.E.,Ph.D.,** Professor, Department of Computer Science and Engineering, Manonmaniam Sundaranar University for his valuable guidance and encouragement throughout my project.

I thank all staff- members of department for their kind advice and pleasing co-ordination throughout my project.

I also thank my friends who supported me with real encouragement and suggestions in each and every phase of my project.

DECLARATION

I hereby declare that the project work entitled "**COMPARATIVE STUDY OF DEEP LEARNING MODELS FOR NATURAL LANGUAGE PROCESSING"** submitted to Manomaniam Sundaranar University, Tirunelveli in partial fulfillment of the requirement for the award of the degree of Master of Science (Computer Science) is a record of original project work done during the academic year 2023 - 2024 under the supervision and guidance of **Dr.D.MURUGAN M.E.,Ph.D.,** Professor, Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli and same work has not been submitted elsewhere for the award of any degree.

[RAMGANESH P]

ABSTRACT

Deep learning (DL) models have significantly advanced natural language processing (NLP) tasks, prompting a comparative analysis in this study focusing on sentiment analysis and textual entailment. Sentiment analysis categorizes text into positive, negative sentiments, while Natural Language Inference assesses logical inference between premises and hypotheses, leveraging DL techniques including attention mechanisms and transformers. DL models such as CNNs, RNNs, LSTMs, GRUs, Bi-LSTMs, and Bi-GRUs excel due to their sequential data analysis capabilities. A systematic process of data preprocessing is followed to ensure consistency and comparability across models. Each model is trained on relevant datasets for sentiment analysis and Natural Language Inference. Trained models are then evaluated on the test set using the evaluate method to calculate test accuracy and loss. Additional metrics such as precision, recall, and F1-score are also calculated. Models are compared to determine the most effective one for each task. The assessment, considering computational efficiency, interpretability, task complexity, and model suitability, aids in selecting the most appropriate DL model for NLP tasks, providing insights into their strengths, limitations, and application suitability.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Natural language processing (NLP) has become a vital tool in many fields, including healthcare, finance, customer service, and marketing, since it allows computers to analyze, grasp, and generate human language. Deep learning is a subfield of machine learning that involves teaching complex data patterns to artificial neural networks with several layers. Networks are trained by continuously giving them new information. In natural language processing (NLP), deep learning models are gaining traction due to their ability to efficiently learn from and accurately anticipate large amounts of text. Natural Language Processing (NLP) has seen significant advancements in recent years, especially with the rise of deep learning techniques. In this paper, a comparative study of deep learning models for two fundamental NLP tasks: sentiment analysis and Natural Language Inference(NLI). Sentiment analysis involves determining the sentiment expressed in text, whether it's positive, negative. On the other hand, Natural Language Inference(NLI) deals with understanding the logical relationships between text fragments, such as whether one sentence logically follows from another or contradicts it.

The deep learning models focus on in this study are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) with Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), as well as their bidirectional variants, Bi-LSTMs and Bi-GRUs. These models have shown prowess in handling various NLP tasks and are particularly relevant for sentiment analysis and Natural Language Inference(NLI) due to their ability to capture complex patterns and sequential dependencies in text data.

To evaluate these models, we use two well-known datasets: the Internet Movie Database (IMDb) dataset for sentiment analysis and the Stanford Natural Language Inference (SNLI) dataset for Natural Language Inference(NLI). The IMDb dataset consists of movie reviews labeled with sentiment polarity, making it ideal for sentiment analysis benchmarking. On the other hand, the SNLI dataset contains pairs of sentences labeled with textual relationship categories such as entailment, contradiction, and neutral, providing a rich source for evaluating models' understanding of logical relationships in text.

The goal of study is to compare the performance of these deep learning models across both sentiment analysis and Natural Language Inference(NLI) tasks. By analyzing metrics such as accuracy, precision, recall, and F1 score, aim to identify the strengths and weaknesses of each model in handling these NLP tasks. Additionally, we seek to provide insights into the suitability of these models for real-world applications, considering factors such as computational efficiency and interpretability.

1.1 Motivation and Justification:

The rapid advancements in natural language processing (NLP) have increased the demand for effective NLP solutions, particularly in tasks like sentiment analysis and named entity identification. Deep learning models offer diverse architectures, such as CNNs, RNNs (LSTMs and GRUs), and bidirectional variants, each with unique strengths in processing text data. A comparative study is essential to identify the most suitable models for these tasks.

Using well-known datasets like IMDb for sentiment analysis and SNLI for Natural Language Inference(NLI) ensures the study's relevance and validity. Evaluating metrics like accuracy, precision, and computational efficiency will provide insights into model performance and practical applicability in real-world scenarios. Ultimately, this study aims to guide NLP practitioners and researchers in selecting optimal deep learning models for specific NLP tasks, driving advancements in NLP technology.

1.2 OUTLINE OF RESEARCH WORK

1. **Datasets:** Utilize IMDb dataset, containing movie reviews with sentiment labels, and SNLI dataset, comprising sentence pairs with logical relationship annotations, for comprehensive analysis in sentiment analysis and Natural Language Inference(NLI) tasks.
2. **Preprocessing:** Apply tokenization, lowercasing, punctuation removal, and word embeddings (e.g., GloVe) to prepare data for DL model training, ensuring standardized input across models.
3. **DL Model Selection:** Choose from CNNs, RNNs with LSTMs and GRUs, and their bidirectional variants, considering their ability to capture complex patterns and sequential dependencies in text data.
4. **Trained Models:** Train selected DL models on IMDb and SNLI datasets separately, optimizing hyperparameters and using techniques like dropout to prevent overfitting.
5. **Evaluate Models:** Evaluate model performance using accuracy, precision, recall, F1 score, and confusion matrix analysis to gain insights into strengths and weaknesses.
6. **Performance Metrics:** Analyze computational efficiency and interpretability alongside performance metrics to assess the trade-offs and practical applicability of each model.
7. **Make Predictions:** Utilize trained models to make predictions on unseen test data, validating their generalization ability beyond the training set.
8. **Select Best Model:** Select the best-performing DL model for each task based on a holistic assessment of performance metrics, computational efficiency, and interpretability, ensuring suitability for real-world applications.

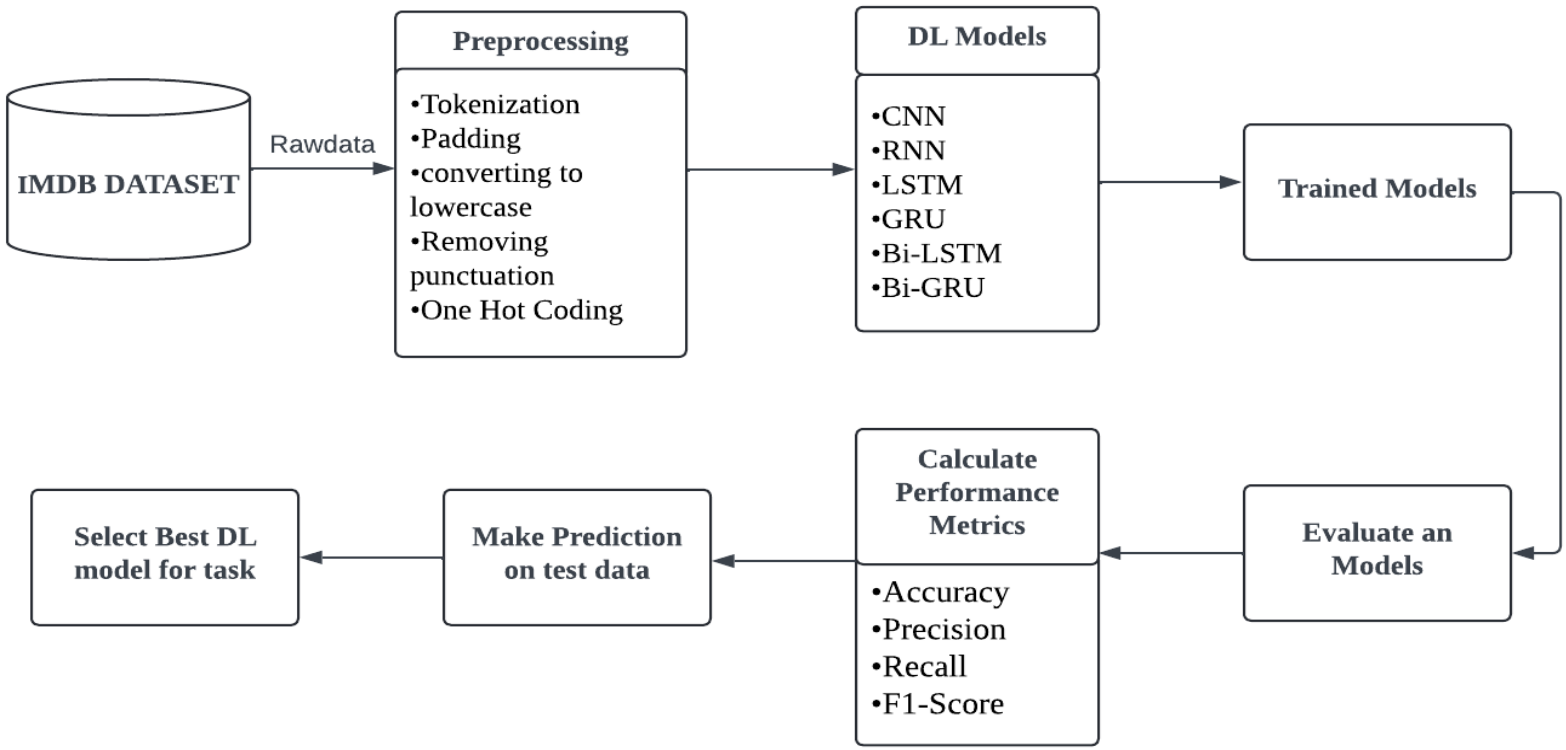

Figure 1: Work Flow of Sentimental Analysis

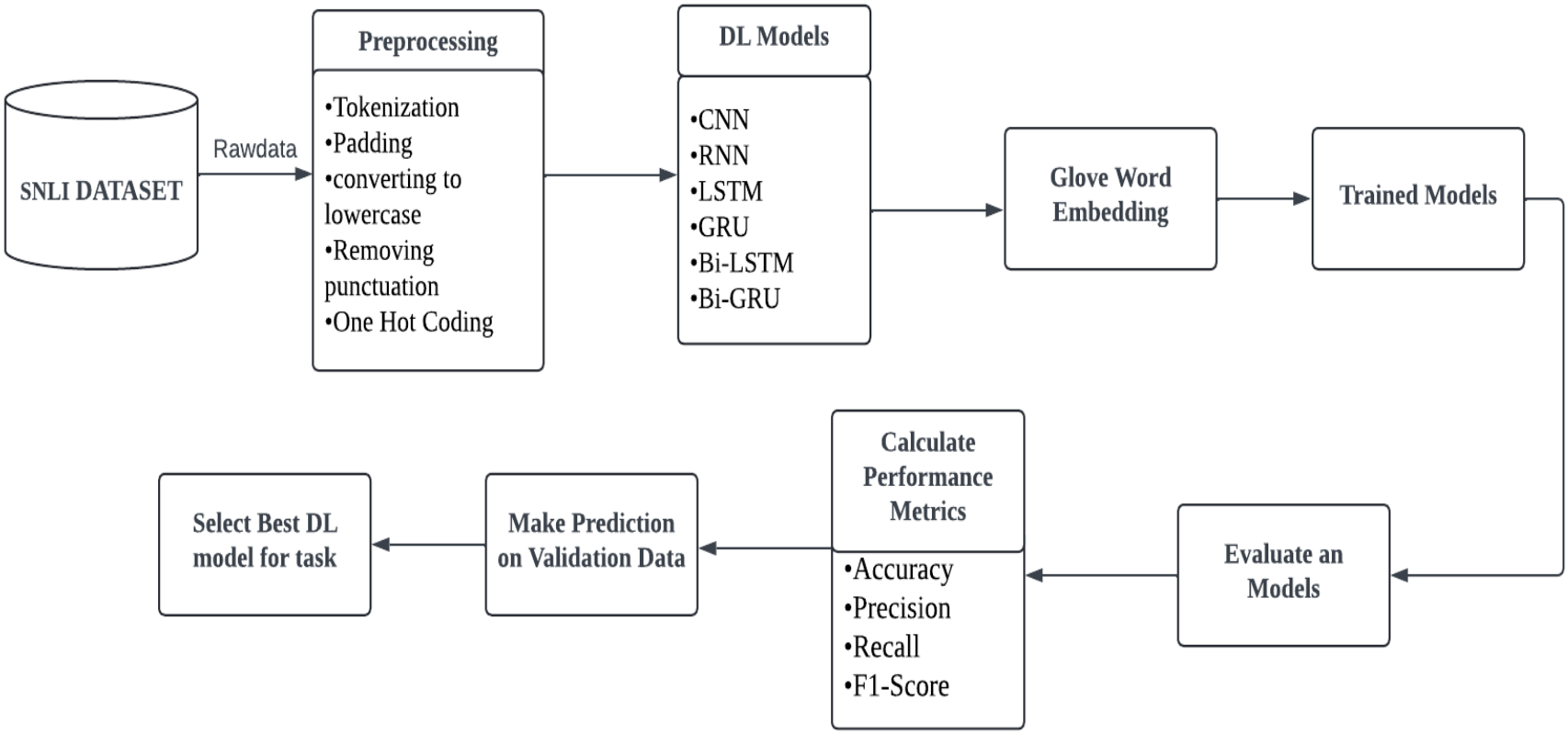

Figure 2: Work Flow of Natural Language Inference

CHAPTER 3

LITERATURE SURVEY

Muhammad Zulqarnain, Rozaida Ghazali, Yana Mazwin, Mohmad Hassim, and Muhammad Rehan, and uploaded on 13 April 2020 by Muhammad Zulqarnain [1], aims to provide a comparative review of deep learning models for text classification. Specifically, it focuses on Deep Belief Neural (DBN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) architectures [1]. The paper investigates these architectures and compares their performance on text classification tasks, with the goal of guiding which deep learning model is best suited for such tasks.

Oumaima Hourrane, El Habib Benlahmar, and Ahmed Zellou [2] conducts an empirical study on sentiment analysis using machine learning techniques on IMDB movie reviews and tweets datasets. It compares nine supervised learning models and implements a voting ensemble classifier with the top four models, showing the ensemble's superiority. The paper suggests the need for more sophisticated sentiment analysis to capture a broader range of human emotions . One limitation mentioned is the challenge of speculating on the evolution of sentiment analysis. Overall, the study concludes that ensembles improve classification, with logistic regression, SVM, and Ridge classifier showing higher accuracy, precision, and recall.

Lisa Gopal [3] conducts a comparative study of deep learning models for Natural Language Processing (NLP) across tasks like sentiment analysis, named entity recognition, and machine translation, while discussing recent advancements [1][2]. Future work includes building interpretable and multilingual models, exploring cross-modal learning, and addressing domain-specific challenges [1]. Limitations include dataset biases and hyperparameter fine-tuning time [3]. The study emphasizes comparing NLP models to identify effective ones for specific tasks and highlights advancements enhancing NLP model performance

Rajesh Das, Mirajul Islam, Mahmudul Hasan, Mocksidul Hassan, and Sharun Khushbu [4] aims to conduct a comparative analysis of machine learning and deep learning models for sentiment analysis in English and Bangla text [1]. It focuses on sentiment analysis of comments from the Bengali e-commerce site 'DARAZ' using various models.Future research could expand the dataset, explore alternative techniques, extend analysis to different domains, focus on aspect-based sentiment analysis, and explore practical applications . Limitations include dataset size and focusing primarily on specific algorithms and models.

Hongxia Lu, Louis Ehwerhemuepha, and Cyril Rakovski [5] compares deep learning models for text classification of unstructured medical notes with varying class imbalances . Seven AI models were employed to classify the presence or absence of 16 disease conditions in patients' discharge summary notes .The study does not mention future work or specific limitations. The Transformer encoder model generally performed best, with the CNN model showing comparable performance in some cases.

Lisa Gopal [6] conducted a comparative study of deep learning models for Natural Language Processing (NLP) . The study compares NLP tasks, discusses recent advancements like pretrained language models, and outlines future work . Limitations include dataset biases and hyperparameter fine-tuning time . The study concludes that comparing NLP models can identify effective ones for specific tasks and highlight advancements enhancing NLP model performance.

Farhad Shiri's paper [7] provides a comprehensive comparative analysis of deep learning models, including CNN, RNN, LSTM, and GRU. The study evaluates these models using IMDB, ARAS, and Fruit-360 datasets. It emphasizes the importance of selecting the appropriate model based on data characteristics and tasks. A limitation mentioned is the short-term memory of simple RNNs. The experiments demonstrate specific model suitability for different data types, showcasing their effectiveness across domains.

Muhammad Zulqarnain, Rozaida Ghazali, Yana Mazwin, Mohmad Hassim, and Muhammad Rehan [8] published a paper on 13 April 2020 providing a comparative review of DBN, CNN, and RNN architectures for text classification tasks. The paper aims to guide the selection of the best-suited deep learning model for text classification.

Abdullah Sharfuddin, Md Tihami, and Md Islam [9] introduces a new method for sentiment classification of Bengali text using a deep recurrent neural network with BiLSTM, achieving an accuracy of 85.67%. The research focuses on sentiment analysis in Bengali, highlighting the need for benchmark datasets and well-furnished models in the language. Future work may involve stemming the dataset and exploring hybrid deep learning models for sentiment classification in Bengali text. Despite advancements, Bengali sentiment analysis still lacks benchmark datasets and well-developed models, posing challenges for researchers.

Muhammad Zulqarnain, Rozaida Ghazali, Yana Mazwin, Mohmad Hassim, and Muhammad Rehan [10] published a paper proposing a text classification model based on an improved approach using GRU and SVM . The research focuses on combining GRU and SVM for text classification, highlighting GRU's advantages in handling long text sequences and suggesting potential future work to enhance the model's performance . The proposed GRU-SVM model outperformed baseline approaches in text classification tasks

Table 1: Literature Survey

S.No	Title	Models Compared	Datasets	Tasks	Performance Metrics	Conclusion
1	A comparative review on deep learning models for text classification	CNN, RNN, DBN	SentiTC,CNAE-9,Textual Entailment,20News Group, Reuters (R-21578)	Text Classification	Learning Rate, Batch size, Hidden Size	Results indicate that CNN is preferred for challenging NLP tasks like text classification, DBN excels in learning multiplex features, and RNN is suitable for sequential modeling tasks across domains.
2	Comparative study of deep learning models for sentiment analysis	Logistic Regression, SVM ,Ridge Classifier, and Perceptron	IMDB, Twitter Review	Sentiment Analysis	Accuracy, Precision	The results of the study indicated that ensembles improve classification, with logistic regression, SVM, and Ridge classifier showing higher accuracy, precision, and recall.
3	A Comparative Study of Deep Learning Models for Natural Language Processing (NLP)	CNN RNN LSTM	SemEval 2014 Task 4, i2b2 2010	Sentiment Analysis, Name Entity Recognition, Machine Translation	Accuracy F1 Score	The comparison of deep learning models for NLP can reveal the merits and weaknesses of different models, identify the most effective models for specific NLP tasks, and highlight recent advancements that enhance NLP model performance

4	Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models	BI-LSTM LSTM	Dataset collected from the Daraz E-commerce site, consisting of 2577 reviews in Bengali along with their English translations.	Sentimental Analysis	Accuracy, Precision, Recall, F1 score	The study contributed significantly to sentiment analysis in English and Bangla languages, highlighting the superiority of SVM models and Bi-LSTM-based models. It offered valuable insights for researchers and practitioners in sentiment analysis, especially in multilingual contexts.
5	Deep Learning Models for Text Classification of Medical Notes	CNN, Transformer encoder, BERT, RNN, GRU, LSTM, and Bi-LSTM,	The dataset used in the study is de-identified discharge summary data provided by Harvard University in 2008, containing information on 16 disease conditions from patients' summaries	Text Classification	AUC-ROC, AUC-PR, F1 Score, and Balanced Accuracy.	The Transformer encoder model performed the best in most scenarios, with the CNN model showing comparable performance in certain cases.

6	Natural Language Processing Applications	Quasi-Recurrent Neural Network and LSTM	Stanford Question Answering Dataset (SQuAD), Penn Treebank dataset, Salesforce's WikiText-103 dataset	Machine Translation, Text Summarization, Named Entity Recognition, Optical Character Recognition, and Part of Speech Tagging	BLUE, GLUE	The paper discusses important terminologies, history, applications, recent developments, datasets, approaches, and evaluation metrics in NLP.It highlights the need for more research on regional languages in the future.
7	A comprehensive Overview and comparative analysis on Deep Learning models	CNN, RNN, LSTM, GRU, Bi-LSTM Bi-GRU	IMDB, ARAS, Fruit-360	Analysis of models	Accuracy, Recall, Precesion,F1-score	The article provides a comprehensive overview of deep learning applications, emphasizing the model-specific suitability for tasks such as image classification and time series analysis across diverse datasets.
8	Comparative Study of CNN and RNN for NLP	CNN RNN	Stanford Sentiment Treebank (SST), WebQSP, and WikiQA	Sentiment Classification, Question Relation Match, and Answer Selection	Accuracy, MAP MRR, Hit 10 metric, Ranking Loss	RNNs excel in various tasks but struggle with keyphrase recognition like sentiment detection and question-answer matching. Optimizing hidden size and batch size is crucial for CNNs and RNNs to perform well

9	A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification	BI-LSTM RNN	Social Media Bengali Text Dataset	Sentimental Analysis	Accuracy	Deep learning models, particularly a deep recurrent neural network with BiLSTM, have shown promising results in sentiment classification of Bengali text, achieving an accuracy of 85.67%
10	Text classification based on gated recurrent unit combines with support vector machine	GRU with SVM	Chinese corpus collected by Fudan University Dr. Li Rongla.	Text Classification	Accuracy Precision Recall F1 Sore	The empirical results demonstrate that the proposed GRU-SVM model outperformed baseline approaches in text classification tasks.

CHAPTER 3

METHODOLOGY

3.1 Preprocessing

3.1.1 Tokenization:

Tokenization is the process of splitting a piece of text into smaller units, typically words or sub words, which are referred to as tokens.

3.1.2 Padding:

Adding zeros to sequences to ensure they are of uniform length, often necessary for training models.

3.1.3 Converting to lowercase:

Standardizing text by converting all characters to lowercase to avoid case sensitivity.

3.1.4 Stop words Removal:

Removing common words (e.g., "the," "is," "and") that do not add significant meaning to the text.

3.1.5 Removing punctuation:

Eliminating punctuation marks from text to focus on content words.

3.1.6 One-Hot Coding:

Encoding words as binary vectors, where each word has a unique index and is represented as a vector with all zeros except for a one at its index position.

3.1.7 GloVe Word Embedding:

Using pre-trained word embeddings like GloVe (Global Vectors for Word Representation) to represent words as dense vectors based on their context and semantic meaning, capturing relationships between words in a vector space.

3.2 Deep Learning Models for NLP Tasks

Natural Language Processing (NLP) is a field of artificial intelligence that empowers computers to understand, interpret, and interact with human language in a way that is both meaningful and useful.

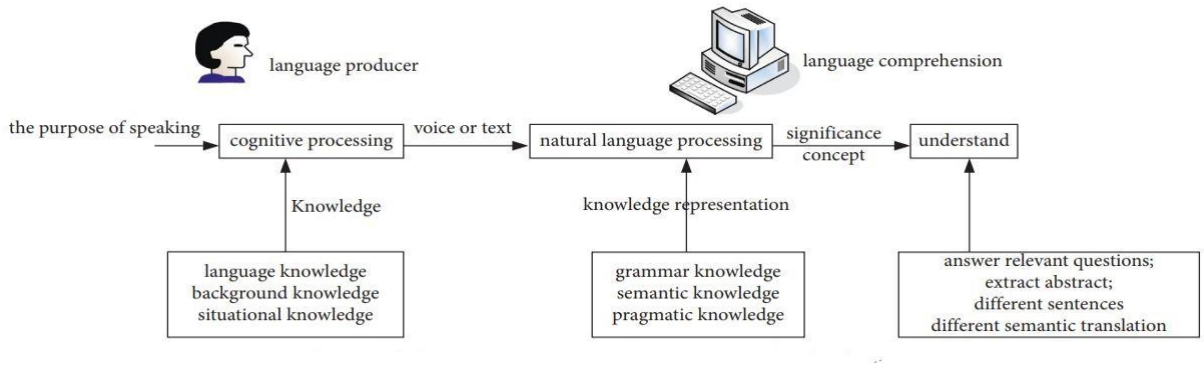

Figure 3: Architecture of NLP

In Deep Learning (DL), Natural Language Processing (NLP) tasks like sentiment analysis and Natural Language Inference (NLI) can be tackled using various neural network architectures. Here's how these tasks can be approached using architectures like CNNs, RNNs (including LSTM, BiLSTM, GRU, and BiGRU):

Task 1: Sentiment Analysis:

1. **CNNs:** Convolutional Neural Networks can capture local patterns and features in text, useful for sentiment classification tasks where important features may be present in specific word sequences.
2. **RNNs (LSTM, GRU):** Recurrent Neural Networks, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), can model sequential dependencies and capture long-range dependencies in text data, aiding in sentiment analysis tasks.
3. **BiLSTM:** Bidirectional LSTMs process sequences in both forward and backward directions, capturing contextual information effectively and enhancing sentiment analysis models.
4. **BiGRU:** Similar to BiLSTM, Bidirectional GRUs can also capture bidirectional context information, beneficial for sentiment analysis tasks involving understanding context and dependencies in text.

Task 2: Natural Language Inference (NLI):

1. **CNNs:** CNNs can be used in NLI tasks for sentence pair classification, where they can capture semantic relationships and features between premise and hypothesis sentences.
2. **RNNs (LSTM, GRU):** RNNs are suitable for NLI tasks as they can process sequential data and capture dependencies between words in premise and hypothesis sentences.
3. **BiLSTM:** Bidirectional LSTMs are effective in NLI tasks as they can capture bidirectional context information and understand the relationships between sentences more comprehensively.
4. Bidirectional GRUs can also be employed in NLI tasks to capture bidirectional context and dependencies between premise and hypothesis sentences.

3.2.1 Convolutional Neural Network(CNN)

Conv1D (or Convolutional 1D) is a type of convolutional neural network (CNN) layer used for processing one-dimensional sequences or signals. While traditional CNNs are commonly used for two-dimensional data such as images, Conv1D layers are specifically designed for processing sequences of data, such as time series data, audio signals, and text.

CNN Architecture

Embedding Layer:

Adds an Embedding layer to convert integer-encoded input sequences into dense vectors of fixed size, capturing semantic relationships between words. The vocabulary + 1 parameter specifies the size of the vocabulary plus one for handling out-of-vocabulary words, while 50 determines the dimensionality of the embedding vectors, affecting the richness of word representations. The input_shape=(all_sentences.shape[1], all_sentences.shape[2]) parameter defines the shape of input sequences, indicating their length and dimensionality.

Reshape Layer:

Reshapes the output of the Embedding layer to match the input shape required by the Conv1D layer. It reshapes the output to the desired shape for the Conv1D layer, ensuring compatibility.

Convolutional Layers (Conv1D):

Adds a Conv1D layer with 128 filters and a kernel size of 5, applying a rectified linear unit (ReLU) activation function for non-linearity. The Conv1D layer performs convolution across the temporal dimension of the input sequences, capturing local patterns.

MaxPooling1D Layer:

Adds a MaxPooling1D layer to perform max pooling along the temporal dimension, reducing the spatial dimensions of the feature maps. The pool_size is determined dynamically based on the output shape after convolution to ensure compatibility.

Dropout Layers:

Adds Dropout layers with a dropout rate of 0.1 after each Conv1D layer and MaxPooling1D layer to introduce regularization, preventing overfitting.

Flatten Layer:

Flattens the output tensor from the previous layers into a 1D vector, preparing it for the final classification layer.

Dense Layer:

Adds a Dense layer with 2 units and softmax activation for the binary classification task, generating class probabilities. Utilizes softmax activation to ensure the output represents probability distributions over the two classes.

It initializes the Embedding layer with pre-trained GloVe word embeddings, freezes its weights to retain these embeddings during training, and sets the trainable parameter to False. This helps leverage semantic knowledge and improve model generalization.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 2, 33, 50)         550850

 reshape_1 (Reshape)         (None, 33, 100)           0

 conv1d_2 (Conv1D)           (None, 29, 128)           64128

 max_pooling1d_2 (MaxPoolin  (None, 7, 128)            0
 g1D)

 dropout_6 (Dropout)         (None, 7, 128)            0

 conv1d_3 (Conv1D)           (None, 3, 128)            82048

 dropout_7 (Dropout)         (None, 3, 128)            0

 flatten_3 (Flatten)         (None, 384)               0

 dense_3 (Dense)             (None, 2)                 770

=================================================================
Total params: 697796 (2.66 MB)
Trainable params: 146946 (574.01 KB)
Non-trainable params: 550850 (2.10 MB)
_________________________________________________________________
```

Figure 4: CNN Model Summary

3.2.2 Recurrent Neural Network(RNN)

A recurrent neural network (RNN) is a deep learning model that is trained to process and convert a sequential data input into a specific sequential data output.

RNN Architecture

Embedding Layer:

This layer converts integer-encoded input sequences into dense vectors of fixed size, capturing semantic relationships between words. The vocabulary + 1 parameter specifies the size of the vocabulary plus one for handling out-of-vocabulary words, while 50 determines the dimensionality of the embedding vectors, affecting the richness of word representations. The input_shape=(all_sentences.shape[1], all_sentences.shape[2],) parameter defines the shape of input sequences, indicating their length and dimensionality.

TimeDistributed(SimpleRNN) Layer (1st):

This layer applies a SimpleRNN unit independently to each time step of the input sequences, capturing sequential dependencies. It is configured with 128 units to control the complexity of learned patterns, and return_sequences=True ensures that this layer returns sequences instead of a single output, which is crucial for sequence-to-sequence modeling.

Dropout Layer (1st):

This layer introduces regularization by randomly setting a fraction (0.10) of input units to zero during training, reducing overfitting.

TimeDistributed(SimpleRNN) Layer (2nd):

This layer adds another SimpleRNN unit to capture deeper contextual information in the sequences. It also has 128 units and return_sequences=True to maintain consistency with the previous layer.

Dropout Layer (2nd):

This layer further regularizes the model by randomly dropping input units during training (0.10 dropout rate), enhancing generalization.

TimeDistributed(GlobalMaxPooling1D) Layer:

This layer performs global max pooling along the time dimension, extracting the most salient features from each sequence. No specific parameters are mentioned in the code, as the default behavior of GlobalMaxPooling1D is utilized.

Flatten Layer:

This layer flattens the output tensor from the previous layers into a 1D vector, preparing it for the final classification layer. No specific parameters are mentioned in the code, as the Flatten layer operates on default settings.

Dense Layer:

This layer concludes the model with a dense layer for classification, applying softmax activation to generate class probabilities. It has 2 units for the binary classification task and uses activation='softmax' to ensure the output represents probability distributions over the two classes.

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_14 (Embedding)    (None, 2, 33, 50)         550850

 time_distributed_18 (TimeD  (None, 2, 33, 128)        22912
 istributed)

 dropout_26 (Dropout)        (None, 2, 33, 128)        0

 time_distributed_19 (TimeD  (None, 2, 33, 128)        32896
 istributed)

 dropout_27 (Dropout)        (None, 2, 33, 128)        0

 time_distributed_20 (TimeD  (None, 2, 128)            0
 istributed)

 flatten_12 (Flatten)        (None, 256)               0

 dense_17 (Dense)            (None, 2)                 514

=================================================================
Total params: 607172 (2.32 MB)
Trainable params: 56322 (220.01 KB)
Non-trainable params: 550850 (2.10 MB)
_________________________________________________________________
```

Figure 5: RNN Model Summary

3.2.3 Long short-term memory (LSTM)

Long short-term memory (LSTM) network is a recurrent neural network (RNN), aimed to deal with the vanishing gradient problem present in traditional RNNs.

LSTM Architecture

Embedding Layer:

This layer converts input words into dense vectors of a fixed size (in your case, 50 dimensions). It essentially maps each word in your vocabulary to a unique vector representation. By setting GloVe word embeddings as weights for this layer, you're leveraging pre-trained word embeddings to initialize the embedding matrix. This helps capture semantic relationships between words.

Reshape Layer:

After the embedding layer, you reshape the input to meet the requirements of the LSTM layer. The reshaping here transforms the output of the embedding layer into a shape that the LSTM layer can process effectively. It ensures that the LSTM layer receives input in the expected format, which is crucial for the model's functionality.

LSTM Layer 1:

This is the first LSTM (Long Short-Term Memory) layer in your model. LSTMs are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data like text. The layer has 64 units, which represent the number of memory cells or nodes in the LSTM. Setting return_sequences=True means this layer outputs sequences rather than just the final output of the sequence, which is useful when stacking multiple LSTM layers.

Dropout Layer 1:

Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to 0 during training. In your model, you've set a dropout rate of 0.2, meaning 20% of the input units to this layer will be randomly dropped during each training iteration, forcing the model to learn more robust features.

LSTM Layer 2:

This is the second LSTM layer in your model, also with 64 units and return_sequences=True. Adding another LSTM layer allows the model to learn more complex patterns and relationships in the input data, especially useful for tasks involving intricate dependencies like text classification.

Dropout Layer 2:

Similar to the first dropout layer, this dropout layer applies regularization to the output of the second LSTM layer, further enhancing the model's generalization ability by reducing overfitting.

Flatten Layer:

After the LSTM layers, the flatten layer reshapes the output from the LSTM layers into a 1D array, preparing it to be fed into the subsequent dense layer.

Dense Layer:

This fully connected dense layer consists of 64 units with a rectified linear unit (ReLU) activation function. The ReLU activation introduces non-linearity into the model, allowing it to learn complex relationships between the features extracted by the LSTM layers.

Output Layer:

The final dense layer with 2 units and softmax activation function is the output layer of your model. It performs binary classification by predicting the probabilities for the two classes (assuming a binary classification task). Softmax activation ensures that the output probabilities sum to 1, making them interpretable as class probabilities.

```
LSTM Model Summary:
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 2, 33, 50)         550850

 reshape_4 (Reshape)         (None, 2, 1650)           0

 lstm (LSTM)                 (None, 2, 64)             439040

 dropout_8 (Dropout)         (None, 2, 64)             0

 lstm_1 (LSTM)               (None, 2, 64)             33024

 dropout_9 (Dropout)         (None, 2, 64)             0

 flatten_4 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 64)                8256

 dense_5 (Dense)             (None, 2)                 130

=================================================================
Total params: 1031300 (3.93 MB)
Trainable params: 1031300 (3.93 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 6: LSTM Model Summary

3.2.4 Bidirectional Long short-term memory (Bi-LSTM)

Bidirectional LSTM (BiLSTM) is a recurrent neural network used primarily on natural language processing. Unlike standard LSTM, the input flows in both directions, and it's capable of utilizing information from both sides.

Bidirectional LSTM Architecture

Embedding Layer:

Converts input words into 50-dimensional dense vectors using GloVe word embeddings. Each word in the input vocabulary is represented by a dense vector capturing semantic relationships.

Reshape Layer:

Adjusts the input shape to match the Bidirectional LSTM's input requirements. It reshapes the output from the embedding layer into a format suitable for processing by the Bidirectional LSTM layers.

Bidirectional LSTM Layer 1:

This layer is a Bidirectional LSTM with 64 units and return_sequences=True, meaning it returns sequences of hidden states rather than just the final output. Bidirectional LSTM processes input sequences in both forward and backward directions, capturing context from both past and future information.

Dropout Layer 1:

Adds regularization by randomly dropping 30% of input units during training, helping prevent overfitting and improving model generalization.

Bidirectional LSTM Layer 2:

Another Bidirectional LSTM layer with 64 units and return_sequences=True is added, further enhancing the model's ability to capture complex patterns and dependencies in the input data.

Dropout Layer 2:

Another dropout layer with a dropout rate of 0.3 is applied after the second Bidirectional LSTM layer to further regularize the model and improve its robustness.

Bidirectional LSTM Layer 3 :

An additional Bidirectional LSTM layer with 64 units and return_sequences=True is included in the model. This layer increases the model's capacity to capture intricate sequential patterns.

Dropout Layer 3:

Another dropout layer with a dropout rate of 0.3 is applied after the third Bidirectional LSTM layer for regularization.

Flatten Layer:

Converts the output from the Bidirectional LSTM layers (which are sequences) into a 1D array, preparing it for input into the subsequent dense layers.

First Dense Layer:

A dense layer with 64 units and ReLU activation function is added to introduce non-linearity and learn complex patterns in the flattened data.

Second Dense Layer (Output Layer):

The final dense layer with softmax activation (2 units) serves as the output layer for binary classification, computing class probabilities for the two classes.

```
Bidirectional LSTM Model Summary:
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 2, 33, 50)         550850

 reshape_7 (Reshape)         (None, 2, 1650)           0

 bidirectional_3 (Bidirecti  (None, 2, 128)            878080
 onal)

 dropout_15 (Dropout)        (None, 2, 128)            0

 bidirectional_4 (Bidirecti  (None, 2, 128)            98816
 onal)

 dropout_16 (Dropout)        (None, 2, 128)            0

 bidirectional_5 (Bidirecti  (None, 2, 128)            98816
 onal)

 dropout_17 (Dropout)        (None, 2, 128)            0

 flatten_7 (Flatten)         (None, 256)               0

 dense_10 (Dense)            (None, 64)                16448

 dense_11 (Dense)            (None, 2)                 130

=================================================================
Total params: 1643140 (6.27 MB)
Trainable params: 1643140 (6.27 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 7: Bi-LSTM Model Summary

3.2.5 Gated Recurrent Unit(GRU)

A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture that was designed to address certain limitations of traditional RNNs, particularly the vanishing gradient problem.

GRU Architecture

Embedding Layer:

Converts input words into 50-dimensional dense vectors using GloVe word embeddings. This layer learns the semantic relationships between words based on the pre-trained word embeddings.

Reshape Layer:

Reshapes the input to match the GRU's input requirements. The reshaping transforms the output from the embedding layer into a format suitable for the GRU layers.

First GRU Layer:

This GRU layer has 64 units and is configured to return sequences (return_sequences=True). It processes the sequential input data and outputs sequences of hidden states, capturing temporal dependencies in the data.

First Dropout Layer:

After the first GRU layer, a dropout layer with a dropout rate of 0.3 is applied. This dropout layer helps prevent overfitting by randomly dropping 30% of the input units during training, promoting better generalization.

Second GRU Layer:

Another GRU layer with 64 units and return_sequences=True follows the first dropout layer. It further processes the sequential information, extracting more complex patterns and features from the data.

Second Dropout Layer:

Similar to the first dropout layer, this dropout layer with a rate of 0.3 adds regularization by randomly dropping a fraction of input units during training, aiding in reducing overfitting and improving model robustness.

Flatten Layer:

The flatten layer converts the output from the GRU layers, which are sequences, into a 1D array. This step prepares the data for feeding into the subsequent dense layers.

First Dense Layer:

This dense layer consists of 64 units with a rectified linear unit (ReLU) activation function. It introduces non-linearity to the model and learns complex patterns in the flattened data from the GRU layers.

Second Dense Layer (Output Layer):

The final dense layer with softmax activation (2 units) serves as the output layer for binary classification. It computes the class probabilities for the two classes, making it suitable for binary classification tasks like sentiment analysis or spam detection.

```
GRU Model Summary:
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 2, 33, 50)         550850

 reshape_5 (Reshape)         (None, 2, 1650)           0

 gru (GRU)                   (None, 2, 64)             329472

 dropout_10 (Dropout)        (None, 2, 64)             0

 gru_1 (GRU)                 (None, 2, 64)             24960

 dropout_11 (Dropout)        (None, 2, 64)             0

 flatten_5 (Flatten)         (None, 128)               0

 dense_6 (Dense)             (None, 64)                8256

 dense_7 (Dense)             (None, 2)                 130

=================================================================
Total params: 913668 (3.49 MB)
Trainable params: 913668 (3.49 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 8: GRU Model Summary

3.2.6 Bidirectional Gated Recurrent Network(Bi-GRU)

A Bidirectional GRU, or BiGRU, is a sequence processing model that consists of two GRUs. one taking the input in a forward direction, and the other in a backwards direction.

Bidirectional GRU Architecture

Embedding Layer:

Converts input words into 50-dimensional dense vectors using GloVe word embeddings. Each word in the input vocabulary is represented by a dense vector capturing semantic relationships.

Reshape Layer:

Adjusts the input shape to match the Bidirectional GRU's input requirements. It reshapes the output from the embedding layer into a format suitable for processing by the Bidirectional GRU layers.

Bidirectional GRU Layer 1:

This layer is a Bidirectional GRU with 64 units and return_sequences=True, meaning it returns sequences of hidden states rather than just the final output. Bidirectional GRU processes input sequences in both forward and backward directions, capturing context from both past and future information.

Dropout Layer 1:

Adds regularization by randomly dropping 30% of input units during training, helping prevent overfitting and improving model generalization.

Bidirectional GRU Layer 2:

Another Bidirectional GRU layer with 64 units and return_sequences=True is added, further enhancing the model's ability to capture complex patterns and dependencies in the input data.

Dropout Layer 2:

Another dropout layer with a dropout rate of 0.3 is applied after the second Bidirectional GRU layer to further regularize the model and improve its robustness.

Bidirectional GRU Layer 3 (Extra):

An additional Bidirectional GRU layer with 64 units and return_sequences=True is included in the model. This layer increases the model's capacity to capture intricate sequential patterns.

Dropout Layer 3:

Another dropout layer with a dropout rate of 0.3 is applied after the third Bidirectional GRU layer for regularization.

Flatten Layer:

Converts the output from the Bidirectional GRU layers (which are sequences) into a 1D array, preparing it for input into the subsequent dense layers.

First Dense Layer:

A dense layer with 64 units and ReLU activation function is added to introduce non-linearity and learn complex patterns in the flattened data.

Second Dense Layer (Output Layer):

The final dense layer with softmax activation (2 units) serves as the output layer for binary classification, computing class probabilities for the two classes.

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 2, 33, 50)         550850

 reshape_6 (Reshape)         (None, 2, 1650)           0

 bidirectional (Bidirection  (None, 2, 128)            658944
 al)

 dropout_12 (Dropout)        (None, 2, 128)            0

 bidirectional_1 (Bidirecti  (None, 2, 128)            74496
 onal)

 dropout_13 (Dropout)        (None, 2, 128)            0

 bidirectional_2 (Bidirecti  (None, 2, 128)            74496
 onal)

 dropout_14 (Dropout)        (None, 2, 128)            0

 flatten_6 (Flatten)         (None, 256)               0

 dense_8 (Dense)             (None, 64)                16448

 dense_9 (Dense)             (None, 2)                 130

=================================================================
Total params: 1375364 (5.25 MB)
Trainable params: 1375364 (5.25 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 9: Bi-GRU Model Summary

3.3 Performance Metrics

1. Accuracy:

Accuracy measures the proportion of correctly classified instances out of all instances. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP+TN+FP+TN} \quad (1)$$

Higher accuracy indicates a better-performing model in terms of overall correctness.

2. Precision:

Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Precision is a measure of how many of the predicted positive instances are actually positive, indicating the model's ability to avoid false positives.

3. Recall (Sensitivity):

Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

Recall is a measure of the model's ability to identify all positive instances without missing any (avoiding false negatives).

4. **F1 Score:**

The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance by considering both precision and recall. The formula for F1 score is:

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Recall} + \text{Precision}} \quad (4)$$

The F1 score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates poor performance in both metrics.

True Positives (TP):

- In sentiment analysis on the IMDb dataset, TP would refer to instances where the model correctly predicts a positive sentiment for a positive review (i.e., the review is actually positive, and the model predicts it as positive).
- In natural language inference on the SNLI dataset, TP would indicate cases where the model correctly identifies entailment or contradiction between sentence pairs.

True Negatives (TN):

- In sentiment analysis, TN would represent instances where the model correctly predicts a negative sentiment for a negative review (i.e., the review is actually negative, and the model predicts it as negative).
- In natural language inference, TN would indicate cases where the model correctly identifies neutral relationships between sentence pairs.

False Positives (FP):

- In sentiment analysis, FP would occur when the model incorrectly predicts a positive sentiment for a negative review (i.e., the review is actually negative, but the model predicts it as positive).
- In natural language inference, FP would indicate cases where the model incorrectly identifies entailment or contradiction when there is none.

False Negatives (FN):

- In sentiment analysis, FN would occur when the model incorrectly predicts a negative sentiment for a positive review (i.e., the review is actually positive, but the model predicts it as negative).
- In natural language inference, FN would indicate cases where the model fails to identify entailment or contradiction when it exists.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1 Task 1: Sentimental Analysis

Sentiment analysis, also known as opinion mining, is the process of using natural language processing (NLP) and machine learning techniques to analyze and determine the sentiment expressed in text data. In the context of the IMDb dataset, sentiment analysis involves classifying movie reviews as positive or negative based on the sentiment expressed in the text.

IMDb Dataset for Sentiment Analysis:

The IMDb dataset is a popular choice for sentiment analysis tasks, particularly for binary classification (positive or negative sentiment). It contains a large collection of movie reviews along with their corresponding sentiment labels.

Example:

-Review: "This movie was absolutely fantastic! The acting was superb, and the plot kept me engaged throughout."

Sentiment Label: Positive

In this example, the sentiment analysis model would classify the review as positive due to the positive language used ("fantastic," "superb acting," "engaged throughout"). The IMDb dataset provides a rich source of labeled data for training and evaluating sentiment analysis models on movie reviews.

4.1.1 Dataset Description:

IMDB dataset having 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets.

Total Positive Sentiment: 25,000

Total Negative Sentiment: 25,000

The dataset was taken from https://www.kaggle.com/code/sajjadhadi/imdb-reviews-sentiment-analysis-deep-learning.

Training	35000
Testing	15000

Table 2: Overview of IMDB Dataset

This dataset have 2 Attributes.

No	Attribute Name	Description
1	Review	This attribute contains the text of the review.
2	Sentiment	This attribute represents the sentiment or opinion expressed in the review. It could be binary (e.g., positive/negative)

Table 3: Description of IMDB Dataset

]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Figure 10: Example of IMDB Dataset

4.1.2 Steps for Sentimental Analysis:

Step 1: Load and Preprocess Data:

- Load the IMDB dataset.
- Clean the data by removing unnecessary columns and handling missing values.
- Encode categorical labels using LabelEncoder.

Step 2: Tokenization and Padding:

- Tokenize text data using Tokenizer with specified vocabulary size and sequence length.
- Pad sequences to a fixed length.

Step 3: Define DL Models:

- Create functions for RNN, LSTM, GRU, CNN, Bidirectional LSTM, and Bidirectional GRU models.
- Include an Embedding layer, specific RNN/GRU/LSTM layers, dropout, and Dense layer with activation.

Step 4: Compile Models:

- Compile each model with an optimizer, loss function (binary_crossentropy), and metrics (accuracy).

Step 5: Train Models:

- Train each model using the fit method with training data, validation split, batch size, and epochs.
- Implement early stopping using EarlyStopping callback.

Step 6: Evaluate and Calculate Metrics:

- Evaluate trained models on the test set using the evaluate method to calculate test accuracy and loss.
- Make predictions on test data and calculate additional metrics (precision, recall, F1-score.

Step 7: Load the Six (CNN,RNN, LSTM, Bi-LSTM,GRU, Bi-GRU) trained model.

Step 8: Make predictions on the test data using the loaded model.

Step 9: Calculate accuracy by comparing predicted labels with true labels.

4.1.3 Experimental Results of different deep learning models on sentiment analysis tasks with NLP.

Model	Accuracy	Precision	Recall	F1-Score
CNN	86.06	86	87	86
RNN	83.40	83	84	83
LSTM	85.62	86	85	86
GRU	**86.59**	**86**	**88**	**87**
Bi-LSTM	85.93	87	85	86
Bi-GRU	86.37	87	86	86

Table 4: Experimental Results of Sentimental Analysis

```
RNN Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      7474
           1       0.83      0.84      0.83      7526

    accuracy                           0.83     15000
   macro avg       0.83      0.83      0.83     15000
weighted avg       0.83      0.83      0.83     15000

LSTM Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.86      0.86      7474
           1       0.86      0.85      0.86      7526

    accuracy                           0.86     15000
   macro avg       0.86      0.86      0.86     15000
weighted avg       0.86      0.86      0.86     15000

GRU Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.85      0.86      7474
           1       0.86      0.88      0.87      7526

    accuracy                           0.87     15000
   macro avg       0.87      0.87      0.87     15000
weighted avg       0.87      0.87      0.87     15000
```

```
CNN Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      7474
           1       0.86      0.87      0.86      7526

    accuracy                           0.86     15000
   macro avg       0.86      0.86      0.86     15000
weighted avg       0.86      0.86      0.86     15000

Bidirectional LSTM Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.87      0.86      7474
           1       0.87      0.85      0.86      7526

    accuracy                           0.86     15000
   macro avg       0.86      0.86      0.86     15000
weighted avg       0.86      0.86      0.86     15000

Bidirectional GRU Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86      7474
           1       0.87      0.86      0.86      7526

    accuracy                           0.86     15000
   macro avg       0.86      0.86      0.86     15000
weighted avg       0.86      0.86      0.86     15000
```

Performance Metrics were evaluated using six deep learning models (CNN, RNN, LSTM, GRU, Bi-LSTM, Bi-GRU) and the best performing model was identified as **Gated Recurrent Unit(GRU).**

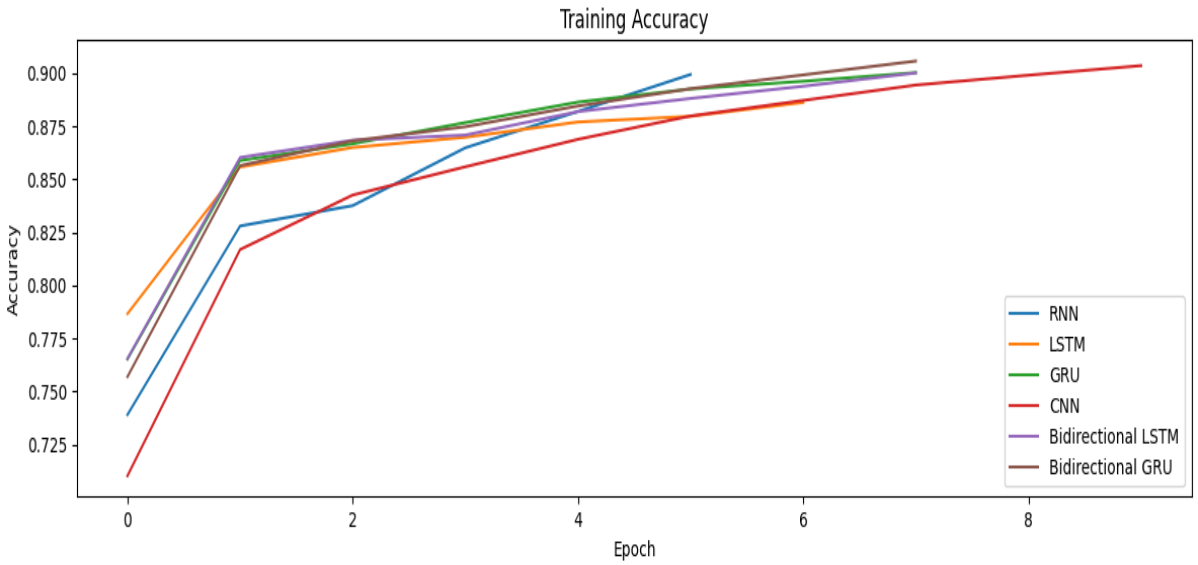

Figure 11: Training Accuracy of Sentimental Analysis Task using six DL models on IMDB Dataset.

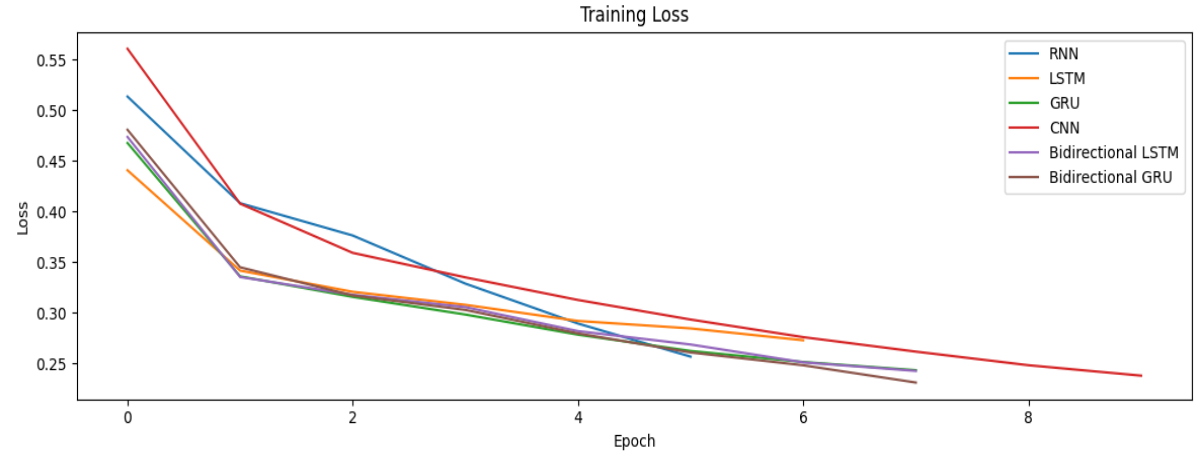

Figure 12: Training Loss of Sentimental Analysis Task using six DL models on IMDB Dataset

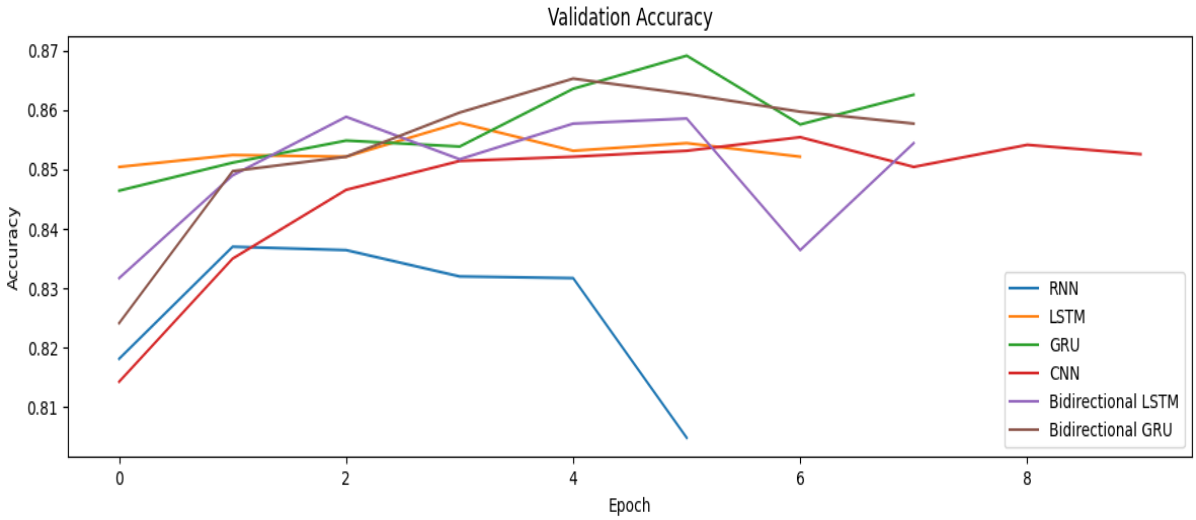

Figure 13: Validation Accuracy of Sentimental Analysis Task using six DL models on IMDB Dataset.

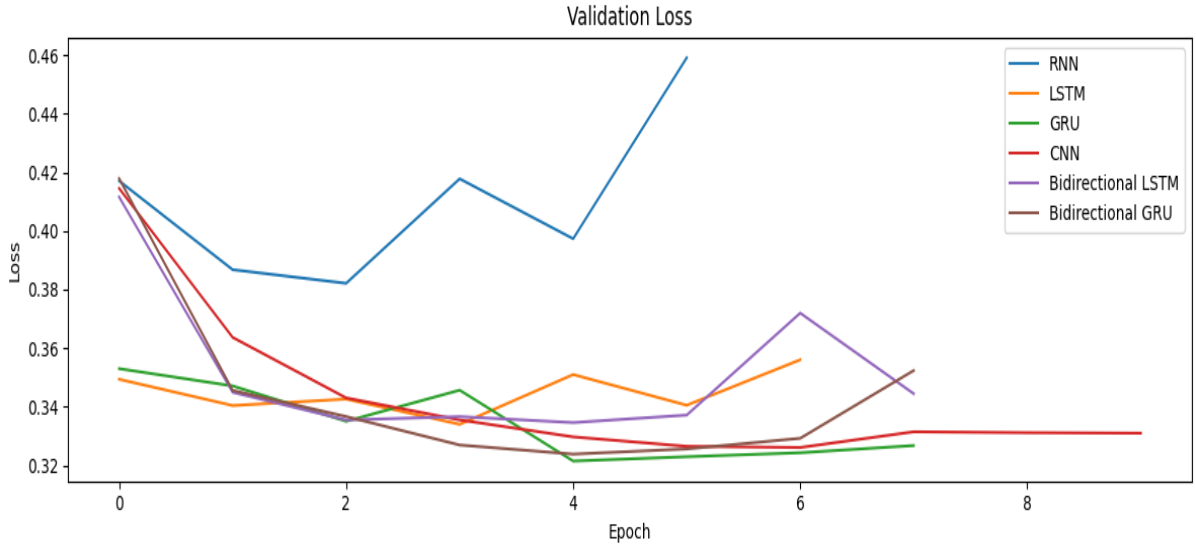

Figure 14: Validation Loss of Sentimental Analysis Task using six DL models on IMDB Dataset

4.1.4 Evaluate On Test Data

Test Data = 10000

Model	Correct Prediction	Wrong Prediction	Accuracy
CNN	12893	2107	86
RNN	12568	2432	83.79
LSTM	12840	2160	85.60
GRU	**12990**	**2010**	**86.60**
Bi-LSTM	12845	2155	85.63
Bi-GRU	12970	2030	86.47

Table 5: Calculate the correct and wrong prediction on test data

If the confidence score is close to 0, then the statement is negative. On the other hand, if the confidence score is close to 1, then the statement is positive. I use a threshold of 0.7 to determine which confidence score is positive and negative, so if it is equal or greater than 0.7, it is positive and if it is less than 0.7, it is negative.

```
Filtered_text: one reviewers mentioned watching  oz episode youll hooked right exactly happened mebr br first thing struck oz b
rutality unflinching scenes violence set right word go trust show faint hearted timid show pulls punches regards drugs sex viol
ence hardcore classic use wordbr br called oz nickname given oswald maximum security state penitentary focuses mainly emerald c
ity experimental section prison cells glass fronts face inwards privacy high agenda em city home manyaryans muslims gangstas la
tinos christians italians irish moreso scuffles death stares dodgy dealings shady agreements never far awaybr br would say main
appeal show due fact goes shows wouldnt dare forget pretty pictures painted mainstream audiences forget charm forget romanceoz
doesnt mess around first episode ever saw struck nasty surreal couldnt say ready watched developed taste oz got accustomed high
levels graphic violence violence injustice crooked guards wholl sold nickel inmates wholl kill order get away well mannered mid
dle class inmates turned prison bitches due lack street skills prison experience watching oz may become comfortable uncomfortab
le viewingthats get touch darker side
[[ 19  20  21   7   1   8  22  23   9  24  25  26   3  10  27  11   1  28
   29  30   2  31   9  32  33  34   4  35  36  37   4  38  39  40  41  42
    2  43  44  45  46   3  47   1  48  49  50  51  52  53  54  55  56  57
   12  58  59   5  60  61  62  63  64  65  13  66  67  12  68  69  70  71
   72  73  74  75  76  77  78  79  80  81  82  83  84  85  86   3  87  14
   88  89   4  15  90  91  92  93  94   6  95  96  97  98  99   6 100   6
  101 102 103 104  10   8 105 106  11 107 108 109  14 110 111 112 113   1
  114 115  13 116 117   2   2 118 119 120  16 121 122  17  16 123 124  18
  125 126 127 128 129  17]]
1/1 [==============================] - 0s 33ms/step
[[0.8368327]]
```

```
[nltk_data]   Package stopwords is already up-to-date!
```

In [9]:
```
print('Text:', filtered_text)
if result >= 0.7:
    print('\033[1mThe Review is positive\033[0m')  # Highlight positive review
else:
    print('\033[1mThe Review is negative\033[0m')  # Highlight negative review
```

```
Text: one reviewers mentioned watching  oz episode youll hooked right exactly happened mebr br first thing struck oz brutality
unflinching scenes violence set right word go trust show faint hearted timid show pulls punches regards drugs sex violence hard
core classic use wordbr br called oz nickname given oswald maximum security state penitentary focuses mainly emerald city exper
imental section prison cells glass fronts face inwards privacy high agenda em city home manyaryans muslims gangstas latinos chr
istians italians irish moreso scuffles death stares dodgy dealings shady agreements never far awaybr br would say main appeal s
how due fact goes shows wouldnt dare forget pretty pictures painted mainstream audiences forget charm forget romanceoz doesnt m
ess around first episode ever saw struck nasty surreal couldnt say ready watched developed taste oz got accustomed high levels
graphic violence violence injustice crooked guards wholl sold nickel inmates wholl kill order get away well mannered middle cla
ss inmates turned prison bitches due lack street skills prison experience watching oz may become comfortable uncomfortable view
ingthats get touch darker side
The Review is positive
```

Figure 15: Prediction on Test Data

4.2 Task 2: Natural Language Inference(NLI)

NLI stands for Natural Language Inference, a task in natural language processing (NLP) where the goal is to determine the relationship between two given text segments: a premise and a hypothesis. The SNLI dataset (Stanford Natural Language Inference) is a widely used benchmark dataset for NLI tasks. It consists of pairs of sentences along with labels indicating the relationship between them, such as entailment, contradiction, or neutral.

Premise: The first sentence, providing context or information.

Hypothesis: The second sentence, which may be related to the premise in various ways.

Labels:

Entailment: The hypothesis can be inferred from the premise.

Contradiction: The hypothesis contradicts the premise.

Neutral: There is no clear inference or contradiction between the premise and the hypothesis.

Example:

Premise: "A person is riding a bicycle on a trail in the forest."

Hypothesis 1: "Someone is outdoors, enjoying nature."

Label: Entailment (The hypothesis can be inferred from the premise as it describes a person outdoors, which aligns with riding a bicycle in the forest.)

Hypothesis 2: "The person is sitting indoors, watching TV."

Label: Contradiction (The hypothesis contradicts the premise as it describes the person as indoors watching TV, which contradicts riding a bicycle in the forest.)

Hypothesis 3: "The person is wearing a helmet."

Label: Neutral (There is no clear inference or contradiction between the premise and this hypothesis, as wearing a helmet is not directly related to riding a bicycle in the forest.)

NLI tasks, such as those in the SNLI dataset, are essential for evaluating models' ability to understand semantic relationships, infer information, and reason about textual entailment and contradiction.

4.2.1 Dataset Description

The SNLI corpus (version 1.0) comprises 570k human-written English sentence pairs labeled for natural language inference (NLI) tasks: entailment, contradiction, and neutral. It serves as a benchmark for evaluating representational systems and as a resource for NLP model development. NLI involves determining the truth of a hypothesis given a true premise. In this task, two sentences, called premise and hypothesis, are provided. The goal is to determine whether the hypothesis is true (entailment), false (contradiction), or undetermined (neutral), given that the premise is true.

The dataset was taken from https://www.kaggle.com/datasets/stanfordu/stanford-natural-language-inference-corpus.

Training	Entailment = 10,006 Contradiction = 9,982 Neutral = 9,974 - = 38 Total = 30,000
Testing	Entailment = 3,368 Contradiction = 3.237 Neutral = 3,219 - = 176 Total = 10,000
Validation	Entailment = 3,329 Contradiction = 3,378 Neutral = 3,235 - = 158 Total = 10,000

Table 6: Overview of SNLI Dataset

This dataset have 6 Attributes.

No	Attribute Name	Description
1	gold_label	Indicates the gold-standard label for the relationship between the premise and hypothesis sentences.
2	sentence1_binary_parse	Represents the binary parse tree structure of the premise sentence.
3	sentence2_binary_parse	Represents the binary parse tree structure of the hypothesis sentence.
4	sentence1_parse	Represents the parse tree structure of the premise sentence.
5	sentence2_parse	Represents the parse tree structure of the hypothesis sentence.
6	sentence1	Contains the premise sentence.
7	sentence2	Contains the hypothesis sentence.
8	pairID	Identifier for the pair of sentences.
9	label1, label2, label3, label4, label5	Alternative annotations or judgments provided by human annotators for the relationship between the sentences.

Table 7: Description of SNLI Dataset

```
      gold_label                          sentence1_binary_parse  \
0        neutral  ( ( ( A person ) ( on ( a horse ) ) ) ( ( jump...
1  contradiction  ( ( ( A person ) ( on ( a horse ) ) ) ( ( jump...
2     entailment  ( ( ( A person ) ( on ( a horse ) ) ) ( ( jump...
3        neutral  ( Children ( ( ( smiling and ) waving ) ) ( at c...
4     entailment  ( Children ( ( ( smiling and ) waving ) ) ( at c...

                              sentence2_binary_parse  \
0  ( ( A person ) ( ( is ( ( training ( his horse...
1  ( ( A person ) ( ( ( ( is ( at ( a diner ) ) )...
2  ( ( A person ) ( ( ( ( is outdoors ) , ) ( on ...
3  ( They ( are ( smiling ( at ( their parents ) ...
4            ( There ( ( are children ) present ) )

                                    sentence1_parse  \
0  (ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...
1  (ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...
2  (ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...
3  (ROOT (NP (S (NP (NNP Children)) (VP (VBG smil...
4  (ROOT (NP (S (NP (NNP Children)) (VP (VBG smil...

                                    sentence2_parse  \
0  (ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...
1  (ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...
2  (ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...
3  (ROOT (S (NP (PRP They)) (VP (VBP are) (VP (VB...
4  (ROOT (S (NP (EX There)) (VP (VBP are) (NP (NN...

                                   sentence1  \
0  A person on a horse jumps over a broken down a...
1  A person on a horse jumps over a broken down a...
2  A person on a horse jumps over a broken down a...
3            Children smiling and waving at camera
4            Children smiling and waving at camera

                                   sentence2        captionID  \
0  A person is training his horse for a competition.  3416050480.jpg#4
1     A person is at a diner, ordering an omelette.  3416050480.jpg#4
2                 A person is outdoors, on a horse.  3416050480.jpg#4
3                 They are smiling at their parents  2267923837.jpg#2
4                        There are children present  2267923837.jpg#2

              pairID         label1 label2 label3 label4 label5
0  3416050480.jpg#4r1n        neutral    NaN    NaN    NaN    NaN
1  3416050480.jpg#4r1c  contradiction    NaN    NaN    NaN    NaN
2  3416050480.jpg#4r1e     entailment    NaN    NaN    NaN    NaN
3  2267923837.jpg#2r1n        neutral    NaN    NaN    NaN    NaN
4  2267923837.jpg#2r1e     entailment    NaN    NaN    NaN    NaN
```

Figure 16: Example of SNLI Dataset

4.2.3 Steps for NLI Task

Step 1: Load and Preprocess Data:

- Load the SNLI dataset.
- It preprocesses the data by removing missing values, selecting relevant columns, and cleaning text data by converting to lowercase, removing punctuation, and filtering out stop words using NLTK.

Step 2: Tokenization and Padding:

- Tokenize text data using Tokenizer with specified vocabulary size and sequence length.
- Pad sequences to a fixed length.
- Encode target labels and convert them to one-hot encoded format.

Step 3: GloVe Word Embedding:

- Load GloVe word embeddings to create an embedding matrix for initializing embedding layers in the neural network models.

Step 4: Model Architectures:

- Define and compile neural network models for sentence pair classification using various architectures like RNN, CNN, LSTM, GRU, Bidirectional GRU, and Bidirectional LSTM.

Step 5: Train Models:

- Train each model using the fit method with training data, validation split, batch size, and epochs.
- Implement early stopping using EarlyStopping callback.

Step 6: Evaluate and Calculate Metrics:

- Evaluate trained models on the test set using the evaluate method to calculate test accuracy and loss.
- Make predictions on test data and calculate additional metrics (precision, recall, F1-score).

Step 7: Prediction on Validation data.

- Calculate total correct and wrong predictions for each class ('contradiction' and 'entailment').
- Print validation set metrics including total sentences, correctly predicted sentences for each class, and overall accuracy.

Results of different deep learning models on Natural Language Inference tasks with NLP.

Model	Accuracy	Precision	Recall	F1-Score
CNN	85.20	85.43	85.20	85.17
RNN	64.67	64.70	64.67	64.67
LSTM	91.57	91.61	91.57	91.57
GRU	86.99	87.43	86.99	86.94
Bi-LSTM	**92.42**	**92.74**	**92.42**	**92.40**
Bi-GRU	89.79	90.21	89.79	89.77

Table 8: Experimental Results of SNLI Dataset

Classification Report for RNN:

	precision	recall	f1-score	support
0	0.64	0.66	0.65	16497
1	0.65	0.63	0.64	16703
accuracy			0.65	33200
macro avg	0.65	0.65	0.65	33200
weighted avg	0.65	0.65	0.65	33200

Classification Report for CNN:

	precision	recall	f1-score	support
0	0.88	0.81	0.84	16497
1	0.83	0.89	0.86	16703
accuracy			0.85	33200
macro avg	0.85	0.85	0.85	33200
weighted avg	0.85	0.85	0.85	33200

Classification Report for LSTM:

	precision	recall	f1-score	support
0	0.93	0.90	0.91	16497
1	0.90	0.93	0.92	16703
accuracy			0.92	33200
macro avg	0.92	0.92	0.92	33200
weighted avg	0.92	0.92	0.92	33200

Classification Report for GRU:

	precision	recall	f1-score	support
0	0.92	0.81	0.86	16497
1	0.83	0.93	0.88	16703
accuracy			0.87	33200
macro avg	0.87	0.87	0.87	33200
weighted avg	0.87	0.87	0.87	33200

Classification Report for Bidirectional GRU:

	precision	recall	f1-score	support
0	0.86	0.95	0.90	16497
1	0.94	0.85	0.89	16703
accuracy			0.90	33200
macro avg	0.90	0.90	0.90	33200
weighted avg	0.90	0.90	0.90	33200

Classification Report for Bidirectional LSTM:

	precision	recall	f1-score	support
0	0.96	0.88	0.92	16497
1	0.89	0.97	0.93	16703
accuracy			0.92	33200
macro avg	0.93	0.92	0.92	33200
weighted avg	0.93	0.92	0.92	33200

Performance Metrics were evaluated using six deep learning models (CNN, RNN, LSTM, GRU, Bi-LSTM, Bi-GRU) and the best performing model was identified as **Bidirectional Long Short Term Memory(Bi-LSTM).**

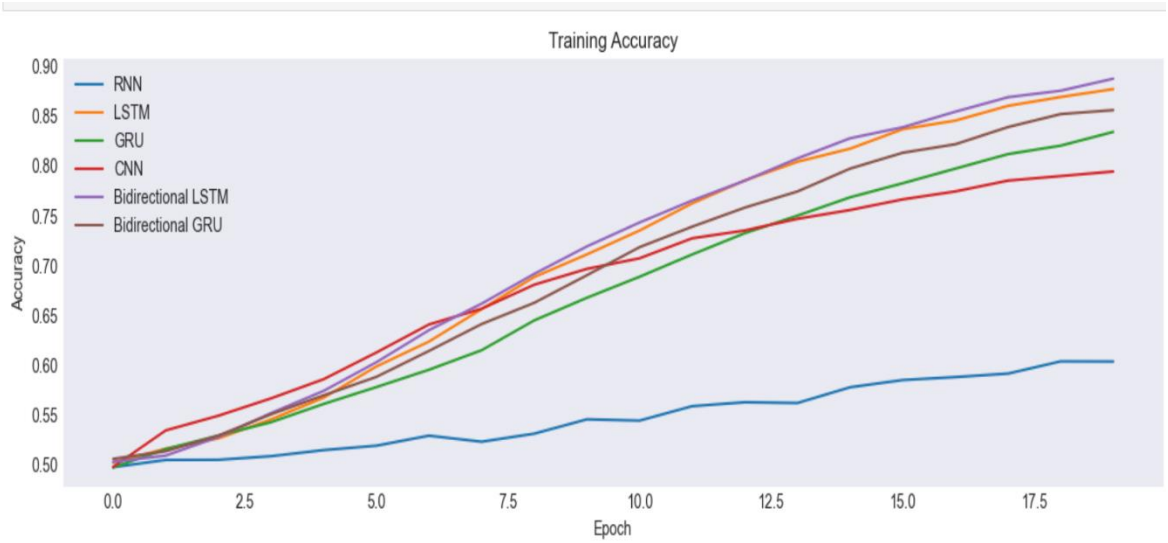

Figure 17: Training Accuracy of Natural Language Inference Task using six DL models on SNLI Dataset

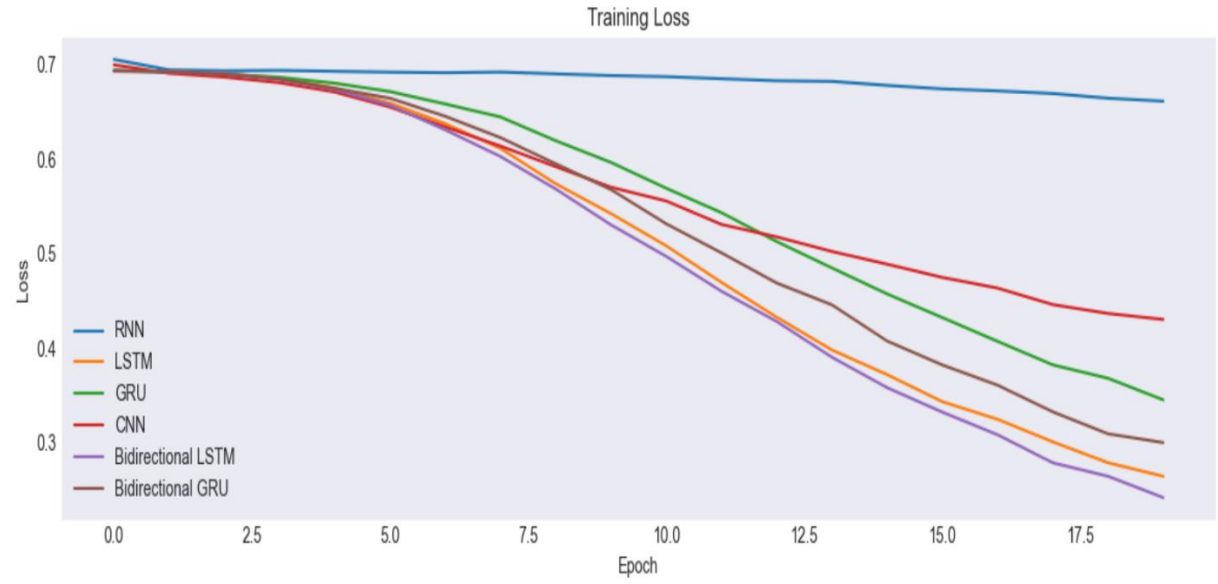

Figure 18: Training Loss of Natural Language Inference Task using six DL models on SNLI Dataset.

Generate predictions on the validation data using the trained model, and then calculate the total correct and wrong predictions for each class ('contradiction' and 'entailment').

Total Contradiction Sentences: 3278

Total Entailment Sentences: 3329

Model	Entailment(%)		Contradiction(%)	
	Correct Prediction	Wrong Prediction	Correct Prediction	Wrong Prediction
RNN	2689	640	2408	870
CNN	2685	644	2699	579
LSTM	2997	332	2958	320
GRU	2270	1059	2586	692
Bi-LSTM	2755	574	3006	272
Bi-GRU	2581	748	2634	644

Table 9: Prediction on Validation Data

CHAPTER 5

CONCLUSION

The comparative study of deep learning models for NLP tasks, including sentiment analysis and natural language inference (NLI), revealed insights into their performance and capabilities. Among the six models (CNN, RNN, LSTM, BiLSTM, GRU, BiGRU), the GRU model showed superior performance for sentimental Analysis and Bi-LSTM model gives best result for Natural Language Inference task. Future work involves exploring ensemble methods, hyperparameter tuning, transfer learning, data augmentation, domain adaptation, and additional NLP tasks such as named entity recognition (NER), relation classification, question answering (QA), and paraphrase identification (PQAP). This comprehensive approach aims to enhance model performance, address specific task requirements, and contribute to the advancement of NLP techniques.

CHAPTER 6

FUTURE ENHANCEMENT

1. **Ensemble Methods:**

   Investigate ensemble methods such as model averaging or stacking to combine predictions from multiple models, potentially improving overall performance and robustness.

2. **Hyperparameter Tuning:**

   Perform comprehensive hyperparameter tuning to optimize model performance further, including tuning learning rates, batch sizes, and regularization parameters.

3. **Transfer Learning:**

   Explore the use of pre-trained language models like BERT or GPT for transfer learning tasks, which could enhance model understanding and generalization.

4. **Data Augmentation:**

   Experiment with data augmentation techniques, such as adding noise or generating synthetic data, to increase the diversity and size of the training dataset, potentially improving model generalization.

5. **Domain Adaptation:**

   Investigate techniques for domain adaptation to fine-tune models specifically for sentiment analysis or natural language inference tasks in specific domains, improving task-specific performance.

6. **Comparative Advanced NLP Tasks:**

   Comparing NER, relation classification, QA, sentiment analysis, and text summarization to evaluate their effectiveness and applicability across various NLP domains.

REFERENCES

[1] Zulqarnain, M., Ghazali, R., Mazwin, Y., Hassim, M., & Rehan, M. (2020). A comparative review on deep learning models for text classification. Indonesian Journal of Electrical Engineering and Computer Science, 19(1), 325-335. DOI: 10.11591/ijeecs.v19.i1.pp325-335.

[2] Hourrane, O., Benlahmar, E. H., & Zellou, A. (2018). Sentiment Analysis: An Empirical Study of Machine Learning Techniques. International Journal of Engineering & Technology, 7(4), 5726-5731. DOI: 10.14419/ijet.v7i4.24459.

[3] Gopal, L. (2020). A Comparative Study of Deep Learning Models for Natural Language Processing (NLP). Journal of Algebraic Statistics, 11(1), 59-70. ISSN: 1309-3452. DOI: https://doi.org/10.52783/jas.v11i1.1432.

[4] Das, R. K., Islam, M., Hasan, M. M., Razia, S., Hassan, M., & Khushbu, S. A. . Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models. Journal Name, Volume(Issue), Page numbers.

[5] Lu, H., Ehwerhemuepha, L., & Rakovski, C. (2022). A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. BMC Medical Research Methodology, 22(181). DOI: 10.1186/s12874-022-01665.

[6] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. Corrected publication. Springer Science+Business Media, LLC, part of Springer Nature.

[7] Mortezapour Shiri, F., Perumal, T., Mustapha, N., & Mohamed, R.. A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. University Putra Malaysia (UPM), Malaysia.

[8] Yin, W., Kanny, K., Yu, M., & Schütze, H. (Year). Comparative Study of CNN and RNN for Natural Language Processing. CIS, LMU Munich, Germany; IBM Research, USA. Contact: wenpeng@cis.lmu.de, kanng@cis.lmu.de, yum@us.ibm.com.

[9] Sharfuddin, A. A., Tihami, M. N., & Islam, M. S. (2018). A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification. Presented at the International Conference on Bangla Speech and Language Processing (ICBSLP), 21-22 September 2018, Shahjalal University of Science and Technology, Sylhet, Bangladesh.

[10] Zulqarnain, M., Ghazali, R., Mazwin, Y., Hassim, M. M., & Rehan, M. (2020). Text classification based on gated recurrent unit combines with support vector machine. International Journal of Electrical and Computer Engineering (IJECE), 10(4), 3734-3742. DOI: 10.11591/ijece.v10i4.pp3734-3742.