Project Title

Sentiment Analysis on IMDb Movie Reviews Using Deep Learning Techniques

F70DB Statistics Dissertation B

Kwan Weng Thong
H00379717

Heriot-Watt University
Department of Actuarial Mathematics and Statistics

Semester 2
2023/2024

Supervised by
Dr. Abdullah Almasri

Abstract

Deep learning neural networks have made a great breakthrough in the field of text classification in today's world, where 328.77 million terabytes of data are created each day. An excessive amount of blogs, comments, and product and movie reviews will be difficult to analyze if there is no machine to help us automatically perform text classification. Sentiment analysis plays a crucial role in extracting meaningful information from a large amount of text data (i.e., comments, reviews, etc.). It is one of the major natural language processing (NLP) tasks, which mainly focuses on classifying sentiment polarity given a large amount of textual data. This report extensively explores the implementation and comparative evaluation of four deep learning models: CNN, BERT, BiLSTM, and CNN-BiLSTM for the classification of movie reviews. As a result, the CNN model demonstrated superior performance, and obtained a competitive result among the implemented deep learning models with state-of-the-art techniques on the IMDb movie review dataset.

Acknowlegdement

I would like to express my thankfulness to all who have given me the motivation to complete this report throughout the two semesters. I am grateful to my machine learning teacher and the TensorFlow tutor for imparting their knowledge and guidance, which has been invaluable in understanding the concepts and techniques used in this project. I would also like to express my gratitude to my family and friends for their constant supports and motivation throughout the whole process of completing this report. A million thanks to everyone who has been helping me during this period of my life.

Table of Contents

List of Figures

List of Tables

Chapter 1 **Introduction**

1.1 Overview

In today's internet-driven world, the digital landscape has witnessed a profound shift in how people express their thoughts and emotions. The rise of social media platforms like X (previously Twitter), Facebook, and online message boards has given individuals unprecedented channels to share their opinions. This transformation in information sharing has led to a new era of understanding public sentiment and emotions, moving away from traditional data collection methods (Lee et al., 2020). Therefore, there is a dire need to develop a methodology to extract valuable insights from the extensive amount of data, classify it into different result categories, and predict the sentiments of end users. At the core of this evolution lies sentiment analysis, a field within Natural Language Processing (NLP) dedicated to interpreting and categorizing emotions from textual data.

Sentiment analysis (also called opinion mining) is the field of study that used to analyse people's opinions, emotions, and attitudes from text expression (Liu, 2022). Sentiment analysis is particularly important in the context of reviews, where short texts capture how people feel about movies, products, and services (Tripathy et al., 2016). By leveraging sentiment analysis systems, unstructured information spanning various subjects, such as products, businesses, films, or topics, where people openly convey their sentiments, can be transformed into well-organized and structured data (Sinha et al., 2022). This capability empowers businesses to comprehend market responses, meet consumer demands, and stimulate growth. The study has noted that significant research has been conducted on various topics like News, X (previously Twitter) data, and Amazon. Nevertheless, sentiment analysis has broadened its area of focus to include the analysis of movie reviews (Nanda et al., 2018). Movie viewers express their opinions through comments, offering valuable insights to the movie viewers. These reviews provide essential information about the movie's strengths, weaknesses, quality, and other relevant factors (Nanda et al., 2018). If an individual is opting to watch a movie, they often base their choices on feelings. When considering which movie to watch, users frequently turn to reviews and ratings. Positive feedback and high ratings instil a sense of excellence, influencing their decision to watch a particular film.

IMDb movie reviews datasets from https://ai.stanford.edu/~amaas/data/sentiment/ is used to train and test the proposed model. IMDb, which stands for Internet Movie Database, is a well-established platform for movie enthusiasts to express their opinions and emotions. The dataset contains a wide range of reviews that reflect diverse sentiments, while it also offers large-scale high-quality data. It consists of 50,000 reviews, and they were evenly divided into 25,000 positive reviews and negative reviews respectively, which is balanced. This dataset is applied in the proposed model because it is a foundation dataset that has been widely used in many previous works on sentiment analysis. This makes it a suitable benchmark for comparing the effectiveness of the deep learning models and will allow us to make a feasible comparison to previous models that have also used this dataset. Hence, this project aims to employ deep

learning techniques to conduct sentiment analysis on movie reviews obtained from the IMDb dataset.

1.2 Problem Statement

In today's digital age, the volume of online movie reviews is overwhelming. Analysing the language spoken by a different kinds of humans is challenging due to the complexity and intricacies of human languages, mainly because that the textual data doesn't have a proper structure (Amulya et al., 2022). Understanding the sentiments embedded in these reviews is crucial for all users who watch movies. Sentiment analysis, which is one of a technique used in NLP, plays a vital role in analysing and classify the polarity of textual data.

To perform sentiment analysis, machine learning models have drawn lots of attention in recent years. The features in most classical machine learning–based models came with several limitations, including the need for labour-intensive feature engineering, a strong dependency on domain knowledge, and an inability to efficiently leverage large training datasets due to predefined features (i.e., the feature space that the model should be trained on is sparse and high-dimensional which reduces the model's performance)(Basiri et al., 2021). To address these limitations, neural network approaches have emerged, mitigating the need for hand-crafted features (Minaee et al., 2021). Since the traditional machine learning techniques could not effectively handle such extensive data, and deep learning became the key of the big data era (Kim et al., 2018). While previous approaches, such as machine learning and lexicon-based methods, have been used to analyse sentiments, recent advancements in deep neural networks, deep learning-based methods are gaining popularity due to their accuracy enhancement in recent years (Kadu & Joshi, 2022). Hence, this research seeks to employ deep learning models that can analyse large volumes of movie reviews effectively.

Deep learning algorithms automatically learn patterns in data and create their own features to represent it (Chitkara et al., 2019; Gasparetto et al., 2022; Li, 2022; Nguyen et al., 2022; Ranjbarzadeh et al., 2022; Rehman et al., 2019; Xu et al., 2022). They then combine these different representations of the dataset, each one identifying a specific pattern or characteristic, to form a more abstract and high-level view of the data (LeCun et al., 2015). This hands-off approach, with less human intervention in feature design and extraction, allows the algorithms to quickly adapt to new data (Heaton, 2018). Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two of the many deep network types that are more frequently used in text processing-related studies (Zhang et al., 2018). Moreover, despite the success of certain deep learning models, there remains a gap in the exploration of various CNN models with different parameter configurations in the context of sentiment analysis on IMDb movie reviews. Besides, Sinha *et al.* (2022) and Arora *et al.* (2023) applied Bidirectional Encoder Representations from Transformers (BERT) to classify sentiment polarity using the IMDb movie reviews dataset, which achieved better accuracy compared to other deep learning models such as CNN and LSTM. A lighter version of BERT, while still retaining 97% of the original BERT's language understanding capabilities, that is DistilBERT, this model has not been explored using IMDb movie reviews dataset yet. Moreover, although there are a lot of study that shows the unidirectional LSTM model demonstrated a superior performance, there

is not much of research investigating the Bidirectional LSTM model for sentiment analysis using the IMDb movie reviews dataset.

Hence, this project aims to implement four deep neural networks: Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), Bidirectional Encoder Transformer (BERT), and a hybrid model, CNN-BiLSTM to conduct the experiment of sentiment analysis on the IMDb movie reviews dataset. Additionally, these models were compared to deep learning models from literature reviews. By employing these four deep learning models on the IMDb movie reviews dataset, we aim to compare the results of the implemented deep learning models.

1.3 Research Questions

1. What are the key characteristics and features of deep Convolutional Neural Network (CNN) models, Bidirectional LSTM (BiLSTM) models, and BERT models in the context of sentiment analysis on IMDb movie reviews?
2. What are the advantages of CNN, BiLSTM, and BERT models in sentiment analysis, and how do their disadvantages impact their overall performance?
3. How does the performance of sentiment analysis on IMDb movie reviews compare among CNN, BiLSTM, BERT, proposed model CNN-BiLSTM and deep learning models in previous works?

1.4 Objectives

The main objectives are:

1. To use the state-of-the-art deep CNN models, BiLSTM model, BERT model for sentiment analysis of IMDb movie reviews.
2. To implement a deep model, that is CNN-BiLSTM model for sentiment analysis.
3. To evaluate deep learning techniques (CNN, BiLSTM, BERT, CNN-BiLSTM) for sentiment analysis on IMDb movie reviews.

1.5 Gantt chart

The Gantt Chart below illustrates the timeline and progression of tasks completed throughout the duration of the project in this semester. It provides a visual representation of the project's workflow, indicating the allocation of time and resources to each task over the course of 13 weeks.

Table 1.1 Gantt chart

No.	Task	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13
1	Problem Statement													
2	Literature Review Discussion													

3	Coding													
4	Methodology													
5	Result and Discussion													
6	Conclusion													
7	Appendix													

1.6 Report Outline

The project paper follows this organisational structure: Chapter (2) reviews and discusses the related work on sentiment analysis that used the same dataset, and the deep learning model structure with respect to the research. Chapter (3) describes the methodology of preprocessing steps and model implementation. Chapter (4) shows the results of the implemented model. Chapter (5) is the results and conclusion of the overall report.

Chapter 2 **Literature Review**

2.1 Introduction

The roots of sentiment analysis at the beginning of the 20th century are focused on public opinion analysis and text subjectivity analysis performed by the computational linguistic community in the 1990's (Mäntylä *et al.*, 2018). After year 2004, sentiment analysis became one of the fastest-growing research areas in computer science as more and more papers related to sentiment analysis have been published (Mäntylä *et al.*, 2018). In the next section, the papers were published in recent years would be summarized, discussed, and criticised, particularly from the year 2017 to the latest year 2023.

2.2 Related Work

Yenter and Verma (2017) proposed a novel approach to sentiment analysis using a combined kernel from multiple branches of Convolutional Neural Network (CNN) and Long Short-term Memory (LSTM) layers. Their proposed model, the combination of CNN and LSTM schemes acquired produced a model with the highest accuracy on the IMDb review sentiment dataset, surpassing prior models. The network architecture incorporates Fully Connected Layers, Convolutional Layers, and LSTM layers. The use of a kernel size of 2, instead of large pooling sizes, resulted in better outcomes. To tackle overfitting, the dropout layers with a rate of 0.5 were employed, in addition, batch normalization is included in the LSTM model. While most models faced the challenges of overfitting, the paper outlines adjustments to layers and parameters that addressed this issue and led to improved accuracy with the dataset. Furthermore, the report also emphasizes the possibility of conducting additional research to increase the efficacy of the model by examining different word representations and preprocessing methods. Nevertheless, it could be beneficial to use different types of word representations such as Word2Vec or Glove pre-trained word embedding model to improve the model efficacy.

Rehman *et al.* (2019) proposed a hybrid CNN-LSTM Model to boost sentiment analysis accuracy in movie reviews. This method merges Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models, showing promise in two popular movie review datasets: IMDB Movie Reviews and Amazon Movie Reviews. They employ Word2Vec embeddings to efficiently capture word semantics, which makes the model more effective. The hybrid CNN-LSTM model with global max-pooling benefits in accuracy and efficiency with reduced memory usage compared to models using more parameters. It outperforms traditional machine learning techniques like Naïve Bayes (NB), Support Vector Machines (SVM), and hybrid models like NB + SVM, as well as deep learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models, achieving an impressive 91% accuracy. But rather of focusing only on machine learning models, a more thorough assessment would compare the proposed hybrid model with other deep learning techniques.

Haque *et al.* (2019) identified the most suitable architecture among Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM), and LSTM-CNN hybrids for classifying sentiment in IMDb movie reviews. They opted for word embedding

over the Bag of Words model to improve memory efficiency by representing words as compact n-dimensional vectors. CNN with two convolutional and pooling layers for feature extraction is explored whereas their LSTM model employed a single LSTM layer after the Embedding Layer to maintain simplicity. The LSTM-CNN hybrid combined LSTM and CNN elements, using the LSTM layer to analyse text order before processing by five Convolutional Networks for identifying sentiment catchphrases. Accuracy metrics based on the confusion matrix and ROC curves are used for assessing the performance. The result shows that the CNN model outperforms LSTM, LSTM-CNN, and other approaches, achieving a remarkable 91% F-score. The main reason for the outperformance of CNN was attributed to its ability to prioritize identifying positive and negative catchphrases over syntactic and semantic sentence structures, which are more crucial for sentiment analysis. Nonetheless, to investigate their performance across a wider range of datasets to provide a more comprehensive understanding of their effectiveness in sentiment analysis tasks.

Ali *et al.* (2019) proposed that the implementation of sentiment analysis classifier for IMDb dataset of 50k movies review using 3 deep learning networks which are Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long short-term memory (LSTM) recurrent neural network in addition to a hybrid network (CNN+LSTM), and compared them to traditional Machine Learning architectures, including Support Vector Machine (SVM), Naïve Bayes, and Recursive Neural Tensor Network (RNTN). The findings reveal that Machine Learning models achieved accuracy rates ranging from 80.70% to 82.90%, while the Deep Learning models significantly surpassed them, boasting accuracy rates between 86.64% and 89.20%. Among the various deep learning classifiers tested, the hybrid model, CNN+LSTM, resulted as the top performer, producing an accuracy rate of 89.20%. In the proposed model, CNN+LSTM, word2vec embeddings from the Keras Library played a pivotal role in the preprocessing stage. Additionally, the Rectified Linear Unit (ReLU) served as the activation function within the CNN layer, while a max-pooling layer was strategically applied to reduce input dimensions for subsequent layers, particularly the convolutional layer. While the summary briefly mentions the use of word2vec embeddings, it doesn't delve into the impact of hyperparameter tuning. However, it would be beneficial if the study discusses how different hyperparameters (e.g., learning rates, batch sizes, network architectures) affect model performance as investigating the optimal hyperparameters for deep learning models can provide valuable insights for reproducibility and generalization.

Qaisar (2020) proposes the Long Short-Term Memory (LSTM) classifier for analysing sentiments of the IMDb movie reviews, which is based on the Recurrent Neural Network (RNN) algorithm. The classification performance is studied in terms of accuracy. Results show a best classification accuracy of 89.9%. This approach they developed has the capability to be incorporated into modern text-based sentiment analysis systems. The two main limitations of sentiment analysis addressed in the study are: the ambiguity of keywords and the incapability to classify sentences without clear emotional keywords. In other words, sentiment analysis can be complicated by keywords that have multiple meanings depending on their context. Also, the difficulty in classifying sentences that do not contain explicit emotional keywords will cause the system to struggle to detect emotions in sentences that lack obvious emotional indicators. In contrast to alternative methods, the Doc2Vec model has proven superior accuracy outcomes while requiring less computational resources (Hoque *et al.*, 2019). To conclude, they utilize a

Long Short-Term Memory classifier with the Adam optimizer to classify IMDb movie reviews and achieve an accuracy of 89.9%. Anyhow, this paper lacks a comparative analysis as there is no comparison in accuracy with other models. Also, the limited use of evaluation metrics and there is only one evaluation method which is "Accuracy".

Cen *et al.* (2020) proposed a sentiment analysis approach employing three deep learning models: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) Neural Network, both commonly used in NLP, and Convolutional Neural Networks (CNN), traditionally associated with image recognition. The results indicated that the CNN model outperformed the other two models, achieving a classification accuracy of 88.22%, while RNN and LSTM reached 68.64% and 85.32%, respectively. In addition, a comparison with previously published works that utilized various machine learning techniques has shown the outperformance of the proposed deep learning methods (CNN, RNN, LSTM) over SVM and RNTN. In future work, the authors aim to develop a superposition model and refine their experimental approach to improve the effectiveness of sentiment analysis, particularly in movie reviews and emotional analysis. However, it is important to recognize the significant impact of data preprocessing on model recognition and feature extraction. Therefore, they should focus more on improving data preprocessing methods.

Minaee *et al.* (2021) present an extensive exploration of the advancements in deep learning-based text classification, with a particular focus on critical tasks like sentiment analysis, news categorization, question answering, and natural language inference. The authors review over 150 deep learning models developed in recent years, highlighting their technical contributions, similarities, and strengths. When implementing sentiment analysis datasets like Yelp, IMDB, SST, and Amazon reviews, the findings clearly show that deep learning outperforms traditional machine learning techniques. For the IMDb dataset, the most effective model is XLNet-Large (Yang *et al.*, 2019) achieving an impressive accuracy of 96.21%. In the end, the article also delves into the challenges faced by deep learning models. One notable challenge is the lack of interpretability in these models. While they do well on some datasets, they may not perform as strongly on others. Moreover, memory-efficient models are essential, as many modern neural language models demand large memory for training. These models have to be compressed to meet the computation and storage constraints of edge applications. This can be done either by using model compression techniques. Developing a task-agnostic model compression method is an active research topic (W. Wang et al., 2020).

Sinha *et al.* (2022) explored sentiment analysis using various deep learning techniques, including CNN, LSTM, LSTM-CNN, GRU, BERT, BERT-CNN, and BERT-LSTM, to assess their effectiveness on the IMDb Movie review dataset. While sentiment analysis has seen significant advancements, there remains a scarcity of comprehensive comparisons to determine the most efficient algorithm. Particularly in the context of IMDb movie reviews, where such research is limited, their study aims to address this gap by conducting sentiment analysis with a wide array of deep learning models, such as CNN, LSTM, and BERT, among others. Their results revealed that CNN exhibited the weakest performance among the models considered. Meanwhile, all three BERT-based models displayed promising performance, but there exist marginal differences. One of the factors potentially contributing to this similarity is that the reduced length of comments simplifies pre-processing. The special CLS token at the beginning

of each sentence in the BERT-CNN model enables the sequential fine-tuning of BERT and achieves higher accuracy. All the same, their study evaluated model performance based on "Precision," "Recall," and "F1" rather than "Accuracy," which makes it challenging to compare with other sentiment analysis studies that utilise "accuracy" as the primary assessment parameter.

Amulya *et al.* (2022) a study focusing on sentiment analysis for IMDb movie reviews, employing both machine learning (ML) and deep learning (DL) techniques to detect and categorize positive and negative sentiments. For the ML approaches, various algorithms such as logistic regression, support vector machine (SVM), Multinomial Naïve Bayes, and XGBoost are utilized. These algorithms were applied to two different vectorization techniques: TF-IDF and count vectorization. The highest accuracy achieved among ML algorithms was approximately 89%, using SVM with TF-IDF. On the other hand, DL models, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM), were implemented on the IMDb Movie Review dataset. DL models inherently perform automatic feature extraction, simplifying complex problem-solving. RNN, among the DL algorithms, achieved the highest accuracy, approximately 88%. Comparing ML and DL approaches, ML requires manual feature extraction, while DL automatically handles this step. Consequently, DL models demonstrated higher accuracy and efficiency than ML. However, for the deep learning models, it could be beneficial if the authors utilize pre-trained word embedding models before implementing the DL models.

Arora *et al.* (2023) implemented the Bidirectional Encoder Transformer (BERT) model to perform sentiment analysis on the IMDb Movie Reviews dataset. In the process of feature extraction, Word to Vector (Word2Vec) is used in Word Embedding. The rise of Neuro-evolution in sentiment analysis has been discovered to enhance classification performance and neural network structures, particularly those combined with BERT. Based on this idea, the author addressed a research question on how to apply the NLP with BERT to evaluate the sentiment analysis model using the IMDb movie reviews dataset. BERT, a transformer model with bidirectional layers, utilizes the Masked Language Model (MLM) for training, significantly improving language comprehension. The study's results indicate that the pre-trained BERT-base model achieved the highest accuracy at 92.4%. Although the model achieved relatively higher result in terms of accuracy, it would be beneficial for the study to apply the model in different datasets to test the adaptability of the model.

2.3.1 Convolutional Neural Network (CNN)

CNN is generally known as a feed-forward neural network because it uses layer-to-layer convolution to create a mechanism of numerous filters, which allows it to extract important hidden features from datasets (Ju et al., 2019; Wang et al., 2019). These networks consist of neurons that own learnable weights and biases. This method is known for its ability to discover and interpret patterns (Salehi *et al.*, 2023). CNN can extract an area of features from global information, and it is able to consider the relationship among these features. This will lead to a higher accuracy in analysis and classification. For natural language processing, text data features also can be extracted piece by piece and to consider the relationship among these features, but without the consideration of context or whole sentence, the sentiment might be understood wrong (Liao *et al.*, 2017). CNNs have a relatively simple structure compared to

other classifiers. The transformation of the input data through the layers of a CNN is simpler than other classifiers. There are very few distinct layers, and each layer transforms the input into output through a differentiable function (Jogin *et al.*, 2018). This simplicity can make CNNs easier to work with and understand. The network structure consists of three layers, a convolutional layer, a pooling layer, and a fully connected layer as shown in Figure 2.1.

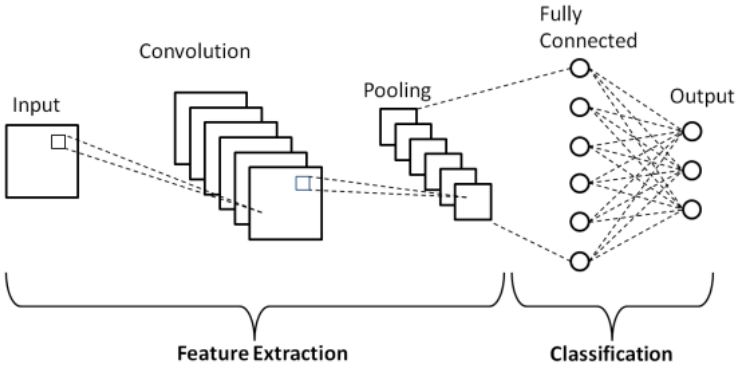

Figure 2.1 CNN Network Architecture

Source: (Phung & Rhee, 2019)

The first layer is a convolutional layer. This is the first layer that is used to extract the various features from the input word vector. The second layer is the pooling layer. This layer is used to reduce the dimension of the feature maps and summarize all the features present in a region of the feature map generated by a convolution layer. The final layer is a fully connected layer. This layer is used for classification. By connecting the data extracted from the previous steps (i.e., convolutional layer and pooling layers) to the output layer, this layer then classifies the input text into the desired label.

2.3.2 Bidirectional-LSTM (Bi-LSTM)

Bidirectional-LSTM (Bi-LSTM) is one of RNN models which recently achieved impressive results in text sentiment analysis. Traditional RNNs encounter challenges such as vanishing and exploding gradients when processing long data sequences. However, these issues are effectively addressed by Long Short-Term Memory (LSTM) units, which are a type of RNN architecture incorporating specialized memory units (Hochreiter & Schmidhuber, 1997). BiLSTM takes this a step further by introducing bidirectionality, allowing the network to gather contextual information from both past (backward) and future (forward) states simultaneously (Graves & Schmidhuber, 2005; Schuster & Paliwal, 1997). This bidirectional approach enhances the model's ability to understand context in text processing, thereby improving accuracy (Li et al., 2020).

The BiLSTM architecture comprises two layers: the first layer processes input sequences from left to right conventionally, while the second layer operates in the reverse direction. This dual-layer structure enables efficient sequential learning tailored for natural language processing tasks (Muthusankar et al., 2023). In comparison to LSTM, which only considers historical context, BiLSTM excels in capturing sequential dependencies, making it particularly effective for tasks requiring nuanced understanding of context. Previous research has demonstrated the effectiveness of both LSTM and BiLSTM in text classification (Chen et al., 2017; Liu et al., 2017; Niu et al., 2017; Nowak et al., 2017). Figure 2.2 illustrates the basic architecture of BiLSTM.

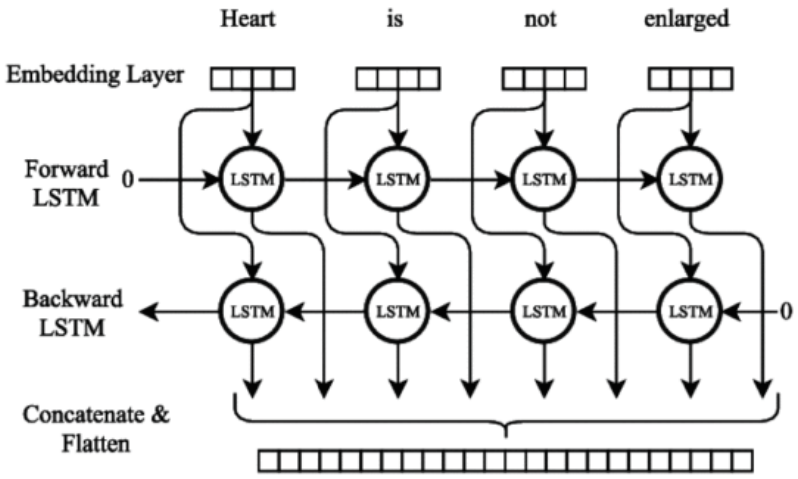

Figure 2.2 Basic BiLSTM Network Architecture

Source: (Tam et al., 2021)

2.3.2 Bidirectional Encoding Representation for a Transformer (BERT)

The BERT model, trained by GOOGLE, is a multi-layered network, that uses multiple cooperative attention feedback to gather informational fusion and generate representations (Sur, 2020). These are referred to as Transformer units. In order to understand a word's context, bidirectional encoder representations (BERTs) take into account both the left and right sides of the word, and hence are capable of multitasking learning and performing different NLP tasks simultaneously (Joloudari et al., 2023). BERT is a bidirectional layer model that solely uses the encoder part, and the Masked Language Model (MLM) is its biggest novelty (Arora et al., 2023). MLM is a way of training a self-supervised model that randomly masks a portion of the words in the text. This improves the model's overall comprehension of the language by teaching it not only what comes before a word but also what comes after it (Arora et al., 2023). Below Figure 2.3 shows the basic transformers architecture.

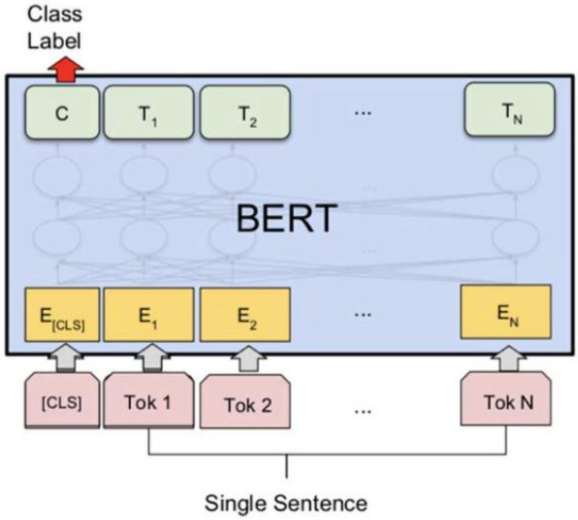

Figure 2.3 Basic transformer architecture.

Source: (Hettiarachchi et al., 2021)

2.4 Literature Review Discussion

From the related work studies, it can be deduced that using CNN in sentiment analysis on IMDb movie reviews mostly achieved high accuracy. However, no work has been done using different parameter settings of configurations in CNN models. Therefore, CNN with different parameters was implemented to analyse the performances of the models. Moreover, hybrid models such as CNN-LSTM which were frequently used for sentiment analysis in recent years, and another hybrid model BERT-CNN both showcased high accuracy and outperformed the previous CNN models. There is no work investigating in hybrid CNN-BiLSTM model in sentiment analysis using the same dataset (i.e., IMDb movie reviews datasets). In addition, Bidirectional LSTM (BiLSTM) allows the network to capture both past and future context that has not been used in sentiment analysis for the same dataset. Therefore, BiLSTM and a hybrid CNN- BiLSTM were explored and implemented to test the hybrid model's performance. Another model that has been rapidly used in recent years due to its advantage of learning what comes before and after words, is Bidirectional Encoder Transformers (BERT). This model has also achieved a high accuracy in sentiment analysis using the IMDb movie review dataset. As BERT model has several versions, the model that mostly utilized was BERT-base model, which is very large. Since there is no work that utilizes DistilledBERT model on this dataset, hence a lighter version of BERT model which is smaller, cheaper and faster was applied. All deep learning models were implemented and compared to other deep learning models that reported in literature reviews.

Table 2.1 Summary of Literature Reviews

Author year	Dataset	Proposed Model	Classifier	Evaluation Metrics	Result	Limitations
(Arora *et al.*, 2023)	IMDb Movie Reviews	BERT (Bidirectional Encoder Transformers)	• BERT	• Accuracy • Recall • Precision • F1-score	• Overall Accuracy: 92.40% Negative Class • Precision: 96.10% • Recall: 88.40% • F1-score: 92.10% Positive Class • Precision: 89.20% • Recall: 96.40% • F1-score: 92.60%	
(Charitha *et al.*, 2023)	IMDb Movie Reviews	Machine Learning	• Logistic Regression • SVM • Multinomial NB • Decision Trees • KNN • Xgboost	• Accuracy	BOW feature extraction technique: • Accuracy: 90% (LogisticRegression) • Accuracy: 90% (SVM) TF-IDF feature extraction technique: • Accuracy: 89% (LR, SVM, MNB)	
(Amulya *et al.*, 2022)	IMDb Movie Reviews	Machine Learning and Deep Learning	• Logistic Regression • Naïve Bayes • SVM	• Recall • F1 Score • Accuracy • Precision	• Accuracy: RNN (0.88) • F1-Score: RNN (0.88)	To identify better data-preprocessing methods to

			• XGBOOST • CNN • RNN • LSTM •	• Accuracy Curves • Loss Curves	• Recall: RNN (0.86) • Precision: RNN (0.95)	achieve improved accuracy for movie review sentiment analysis.
(Sinha *et al.*, 2022)	IMDb Movie Reviews	Deep Learning Techniques and Hybrid Model	• CNN • LSTM • CNN-LSTM • LSTM-CNN • GRU • BERT • BERT-CNN • BERT-LSTM	• Precision • Recall • F1	• Precision: BERT-CNN (89.61%) • Recall: BERT-LSTM (89.32%) • F1: BERT-LSTM (89.22%)	
(Minaee *et al.*, 2021)	IMDb Movie Reviews, SST-2, Amazon-2, Amazon-5, Yelp-2, Yelp-5	A Comprehensive Review of more than 40 popular datasets and 150 deep learning-based models for text classification	• LDA • BoW+SVM • TF-IDF • ULMFiT • BCN+Char+CoVe • Virtual adversarial training • Block-sparse LSTM • BERT-base • BERT-large • Multi-Task DNN • BERT Finetune + UDA • XLNet-Large	• Accuracy	• Accuracy: XLNet-Large (96.21%), BERT model (95.63%~95.8%)	• Most DL models are not interpretable. • Most modern neural language models require a significant amount of memory for training and inference.
(Cen *et al.*, 2020)	IMDb Movie Reviews	Deep Learning Model	• CNN • RNN • LSTM • SVM	• Accuracy	• Accuracy: CNN (88.22%)	

			• RNTN			
(Qaisar, 2020)	IMDb Movie Reviews	Long Short-Term Memory (RNN algorithm)	• LSTM with Adam optimizer	• Accuracy • Confusion Matrix	• Accuracy: 89.9%	
(Ali *et al.*, 2019)	IMDb Movie Reviews	Deep Learning Models	• MLP • CNN • LSTM • CNN-LSTM Compare with • SVM • NB • RNTN	• Accuracy • Accuracy curve • Loss curve	• Accuracy: CNN-LSTM (89.20%) • DL models: 86% ~ 89% • ML models: 80% ~ 82.9%	
(Haque *et al.*, 2019)	IMDb Movie Reviews	Various neural network architectures	• Random Forest • SVM • Logistic Regression • LSTM • LSTM-CNN • CNN	• Accuracy • Recall • Precision • F-Score • Training Loss • ROC	• Accuracy: CNN (0.9) • Recall: CNN (0.95) • Specifity: LSTM (0.9) • Precision: LSTM (0.9) • F-Score CNN (0.91) • Training Loss: CNN • ROC: CNN	
(Rehman *et al.*, 2019)	IMDb Movie Reviews, Amazon Movie Review	Hybrid CNN-LSTM Model	• NB • SVM • GA • CNN • LSTM • CNN-LSTM	• Precision • Recall • F-measure • Accuracy	• Precision: Hybird CNN-LSTM (91%) • Recall: Hybird CNN-LSTM (86%) • F-measure: Hybird CNN-LSTM (88%) • Accuracy: NB, NB+SVM, Hybird CNN-LSTM (91%)	

(Yenter & Verma, 2017)	IMDb Movie Reviews	Combined kernel from multiple branches of CNN with LSTM	• CNN-LSTM	• Training Loss • Validation Loss • Training Accuracy • Validation Accuracy	• Accuracy: Proposed CNN based LSTM network (0.895)	

Chapter 3 **Methodology**

This section specifies the methodology for the project. The procedure of the project follows the steps of problem identification, data acquisition, data preprocessing, exploratory data analysis, deep learning model implementation and evaluation for sentiment classification. IMDb movie reviews dataset acquired from https://ai.stanford.edu/~amaas/data/sentiment/ was used to perform sentiment analysis with the proposed deep learning models. The results of all model's implementation will be discussed in the next chapter, Chapter 4. Figure 3.1 below shows the flow chart of methodology in this project.

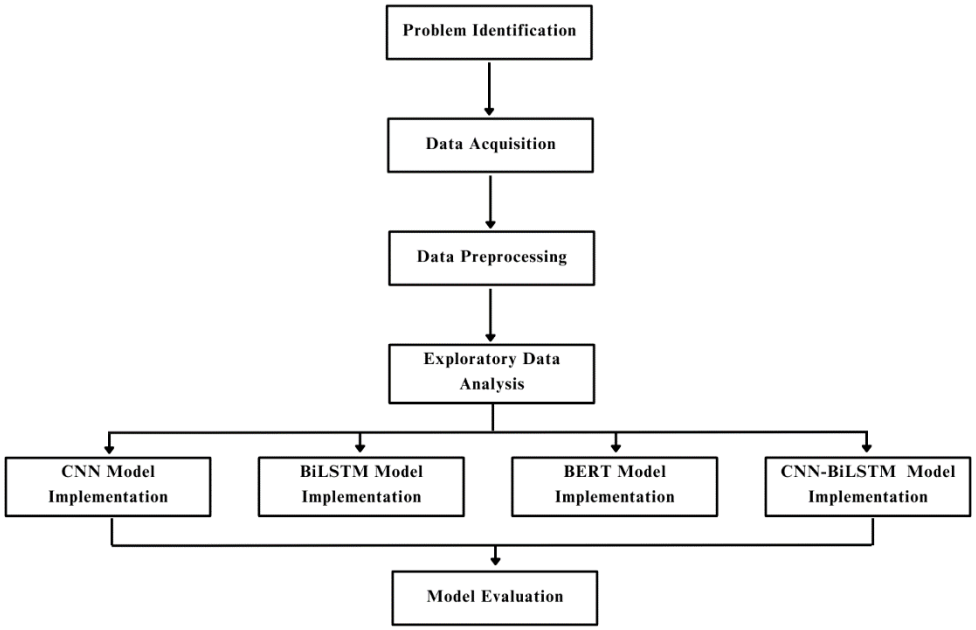

Figure 3.1 Methodology used in the research.

3.1 Step 1: Problem Identification

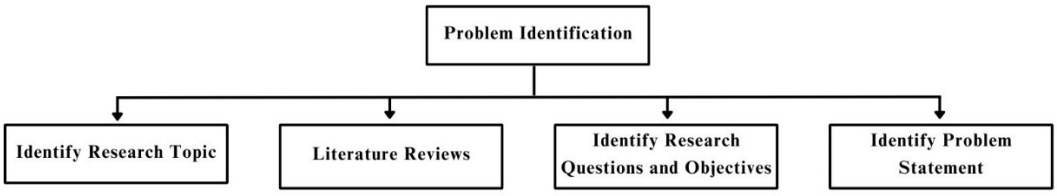

Figure 3.2 Problem Identification

At the beginning of the setting of the research topic, identifying a research problem is the first and most crucial step in any research project. A research problem is a specific issue, gap, or challenge that needs to be addressed through the investigation. It directs the questions, objectives, techniques, and conclusions of the research. Before issuing the gap or challenge of the research, literature reviews are explored to collect the similarities and differences of the related works, and lastly limitations and gaps were addressed. Since the subject area was

chosen to focus on deep learning techniques for sentiment analysis, the sentiment classification-related works were explored and summarized. After discussing literature reviews, problem statement, research questions and objectives are clearly claimed in Chapter 1, also, the four deep learning models were chosen to implement.

3.2 Step 2: Data Acquisition

The IMDb Movie Reviews dataset represents a group of movie reviews that contains 50,000 movie reviews along with their binary classification (positive or negative). This dataset is intended to serve as a benchmark for sentiment classification. The dataset is balanced as it contains the same number of positive and negative reviews, i.e. 12500 reviews for two classes. Figure 3.3 below shows the number of reviews for each class in IMDB movie reviews dataset.

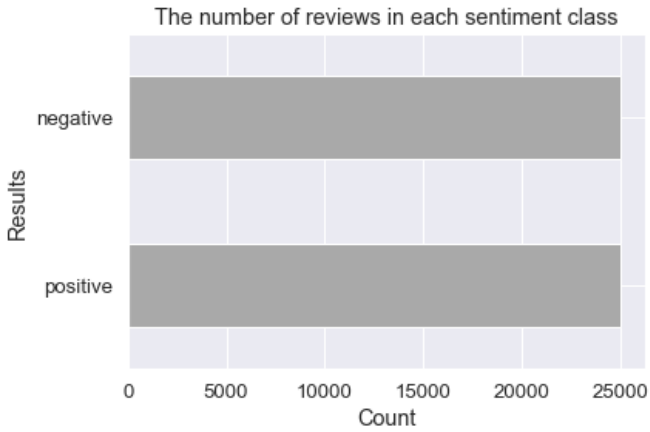

Figure 3.3 The number of reviews for each class in IMDb movie reviews dataset.

3.3 Step 3: Data Preprocessing

It is crucial to preprocess the text data before fitting them into the model. Preprocessing is the process of formatting and organizing the valid information from unstructured data using methods like text mining. Cleaning the data allows for the removal for unnecessary content from the corpus texts, such as the removal of stop words, symbols, URLs as well as punctuation, etc. Below figure shows the steps of data preprocessing for the textual data of movie reviews before training the model.

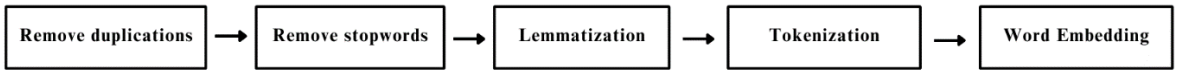

Figure 3.4 Steps of data preprocessing.

3.3.1 Remove Duplication

Removing duplicate data reduces the noise and redundancy in the data and hence improve the efficiency of the model. The dataset has a total of 50,000 reviews, i.e. 25k reviews for each class. The first step is to identify the dataset where the duplicate exists. After identifying that

there are 418 duplicated reviews, then the next part is the removal of duplications. Below figure shows the number of reviews in each class after removing the duplications.

```
positive    24884
negative    24698
Name: sentiment, dtype: int64
```

Figure 3.5 Number of reviews in each class after removing duplications.

3.3.2 Remove Stop Words

A critical preprocessing step is the removal of stop words, which are words like "the," "a," "and," and so on that are overly common yet add little semantic meaning to the text. This procedure is essential for reducing background noise in the text and maximizing attention to the important words. The Natural Language Toolkit (NLTK) was utilized to methodically eliminate stop words from the corpus. Additionally, NLTK was used to remove other unnecessary components such as punctuation, HTML tags, emojis, and URLs, which resulting in a text dataset that is cleaner and more refined.

3.3.3 Tokenization

Tokenization helps to divide and break down the textual information into individual separated words (Vijayarani & Janani, 2016). There are two types of tokenizers: word tokenizer which tokenizes the sentence into words and sentence tokenizer tokenizes the paragraph into sentences (Anees *et al.*, 2020). For this project, word tokenization is executed using the NLTK tool as it is a widely recognized Python Natural Language Processing Toolkit.

```
word_tokenize("The sole meaning of life is to serve humanity")
['The', 'sole', 'meaning', 'of', 'life', 'is', 'to', 'serve', 'humanity']
```

Figure 3.6 Example of word tokenizer.

```
sent_tokenize('Life is a matter of choices, and every choice you make makes you.')
['Life is a matter of choices, and every choice you make makes you.']
```

Figure 3.7 Example of sentence tokenizer.

3.3.4 Lemmatization

Lemmatization is an essential processes of preprocessing for feature extraction (Jeong et al., 2011; Kobayashi et al., 2007; Stymne, 2011). The lemmatization takes the context into account and reduces the word to its meaningful base form, which is called lemma. For example, lemmatizing the words “caring” and return “care’, which is appropriate.

3.3.5 Word Embedding

Word embeddings act as an crucial component in deep models by providing input features for various language tasks like sequence labelling and text classification. (C. Wang et al., 2020). Compare to earlier methods, Word2vec eliminates the need for a nonlinear hidden layer in the forward feedback neural network and directly connects the intermediate layer to the SoftMax layer (C. Wang et al., 2020). Word2Vec, abbreviated as "Word to Vector," stands out for its efficiency and high effectiveness tool in learning word representations from a given text corpus (Mikolov, Chen, *et al.*, 2013; Mikolov, Sutskever, *et al.*, 2013), which comprises two models: CBOW and Skip-gram. These two models can effectively capture the semantics of words and easily transfer them into other downstream tasks. However, an embedding layer that Keras provided was made used, using a vocabulary consisting of 143380 unique words, each of which is embedded in a vector space with 4000 dimensions. We trained our embedding layer from the training samples from the IMDb movie review dataset, instead of utilizing a pre-trained embedding word model. This decision was made due to lower accuracy observed when testing with CNN and CNN-BiLSTM models.

3.4 Step 4: Exploratory Data Analysis

Once the data is acquired and preprocessed, the next step is to go with an exploratory data analysis (EDA) to get a complete grasp of the data by analyzing, investigating the data sets and summarizing their main characteristics with graphical techniques. Below figure 3.8 shows the methodology of exploratory data analysis. The results of EDA were shown in the next chapter.

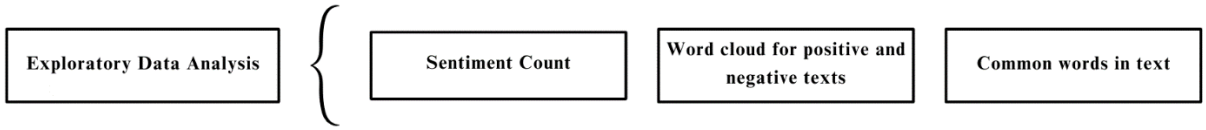

Figure 3.8 Exploratory Data Analysis.

3.5 Step 5: Training and Testing

Before implementing the models, the dataset was divided into training and testing sets to facilitate model training and evaluation. The dataset was split into training and testing sets using an 80:20 ratio, with 80% of the data allocated for training and 20% for testing. Specifically, the training set comprised 39665 reviews used for model training, while the testing set consisted of 9917 reviews utilized to assess the model's performance. In this context, X represented the reviews, while Y denoted the corresponding sentiment labels. Consequently, the dimensions of the training set were represented by $X\_train.shape = (39665, 4000)$, and those of the testing set were denoted by $X\_test.shape = (9917, 4000)$. It is important to note that the maximum number of words in a sentence was set to 4000 during the tokenization and padding process.

3.6 Step 6: Model Implementation

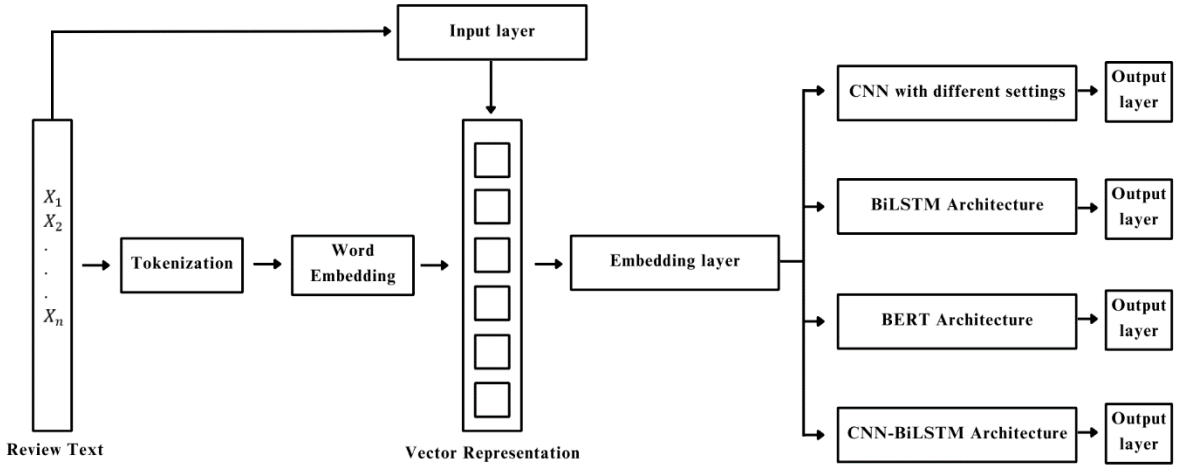

Figure 3.9 Flow diagram of model implementation.

The proposed model CNN has been implemented according to the workflow illustrated in Figure 3.9 in addition to BiLSTM, BERT and CNN-BiLSTM. Tensorflow.keras is applied and used for building the layers of deep learning models. Each model was trained on 80% of the dataset while the remaining 20% were used for testing the model. The performances of the four deep learning models were evaluated and compared using the evaluation metrics such as accuracy, precision, recall and F1-score. Highest accuracy reported of each model was recorded accordingly, as elaborated in the next chapter (Result and Discussion).

3.6.1 Convolutional Neural Network (CNN)

The CNN models were developed utilizing TensorFlow, which is a core open-source library, to enhance the efficiency of deep learning models, implemented through the Python programming language. The CNN model comprises the input layer, convolution layer, max-pooling layer, and a fully connected layer with a sigmoid activation function. Figure 3.10 illustrates the CNN architecture that is implemented in this project.

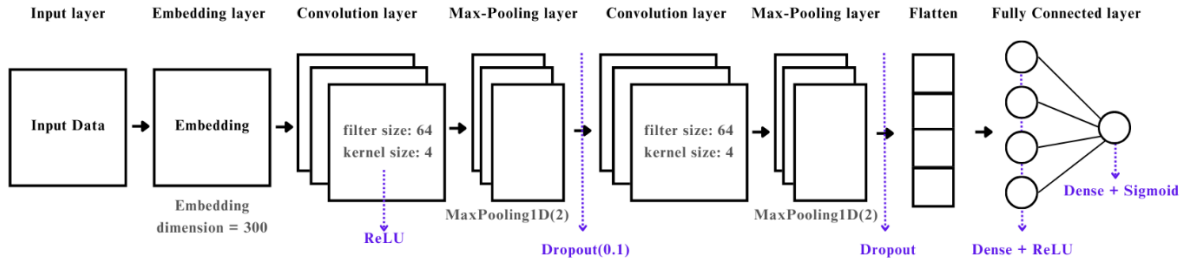

Figure 3.10 Adapted CNN architecture for sentiment analysis.

Source: (Lim et al., 2020)

In the input layer, the neural network utilizes word embeddings as word vector representations, where words sharing similar meanings produce vectors of identical dimensions. An embedding

layer with 300 vector dimensions was utilized. The sequences are padded with zero vectors to ensure dataset lengths are the same.

After the convolutional operation, the output for each filter is a feature map. The max-pooling layer then operates on each feature map separately by calculating the maximum value in each patch of each feature map. Max-pooling layer is used to reduce the spatial dimensions of the input volume (Wang et al., 2018). It applies a sliding window to a set of filters m with a length h for each sentence. The result of max-pooling for each filter is a single value (the maximum value) extracted from that feature map. After the max-pooling layer, a dropout layer which is a regularization technique helps to prevent overfitting by promoting redundancy is added with a dropout rate of 0.1 and hence the collection of these maximum values from each filter, after dropout, forms a vector. The dimensions of this vector are equal to the number of filters used in the max-pooling layer, fed into the fully connected layer for classification tasks. The fully connected layer performs calculations given by $y = \alpha(W \cdot X + b)$ where $\alpha(z) = \max(0, z)$ is the activation function that utilizes Rectified Linear Unit (ReLU). The weight matrix W is in $R^{m \times m}$, the bias b is in R^m, and the feature map is represented by C.

In this study, the various CNN models were developed with different parameters in each layer. The configurations of these parameters were determined through several rounds of trial-and-error experimentation, including the number of convolution layers ranging from 2 to 3, and filters varying from 64 to 256. Table 3.1 presents the detailed parameter configurations of the proposed CNN models. The results of various CNN models were reported in next chapter.

Table 3.1 The configuration of parameters of the proposed CNN models

Parameter/CNN Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
No. of Convolutional Layers	2					3				
Vocabulary Size	143380									
Embedding Size	300									
Activation Function	ReLu, Sigmoid									
Number of Filters	64		64,32	128	256	64		128	256,256,128	256
Filters Size	4	8	3	5	5	4	8	8,5,5	8,5,3	8
Input Vector Size	4000									
No. of Fully Connected Layer	2									
Dense	256, 1									
Dropout	0.1									
Batch Size	32									
Epochs	10									

3.6.2 Bidirectional Long Short-Term Memory (BiLSTM)

The Bidirectional Long Short-Term Memory (BiLSTM) model is a variant of the Long Short-Term Memory (LSTM) network (Zhou et al., 2016), designed to capture sequential dependencies and patterns in sequential data, such as text. The bidirectional feature improves the ability of the model to understand the context by enabling it to simultaneously take into

account data from both past and future time steps. Figure 3.11 illustrates the CNN architecture that is implemented in this project.

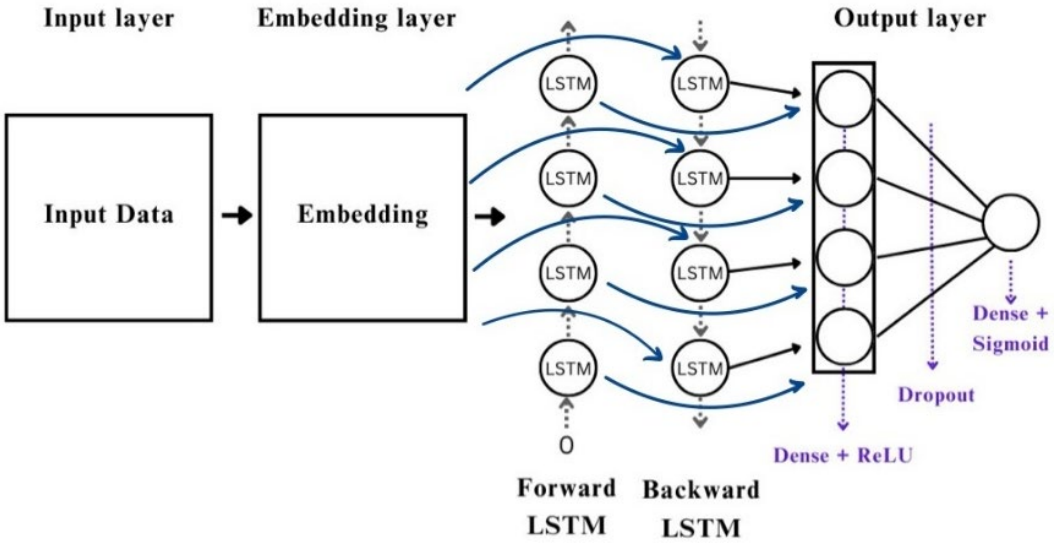

Figure 3.11 Adapted BiLSTM architecture for sentiment analysis.

Source: (Zhou & Wu, 2018)

The model starts with an embedding layer, which is responsible for converting the input sequences into dense vectors of fixed dimensions. In this case, the embedding dimension is set to 300. Subsequently, two Bidirectional LSTM layers follow the embedding layer. Each LSTM layer has 256 units, and these layers allow the model to consider information from both the forward and backward directions in the input sequence. Following the LSTM layers, there are two Dense layers that help to capture high-level features from the LSTM output with 250 units and utilizes ReLU activation function. Next, a Dropout layer with a dropout rate of 0.1 is added after the first Dense layer. Lastly, the final Dense layer has a single unit with a sigmoid activation function. This layer produces the model's output, which represents the predicted sentiment (binary classification: positive or negative).

3.6.3 CNN-BiLSTM Model

The CNN-BiLSTM hybrid model aims to combine the strengths of both Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory networks (BiLSTMs) for improved sentiment analysis on IMDb movie reviews. The best CNN model from the previous experiment which displayed in figure 3.10 is selected and combined with BiLSTM model which displayed in figure 3.12. Below figure shows the simple architecture for the combine model of CNN-BiLSTM.

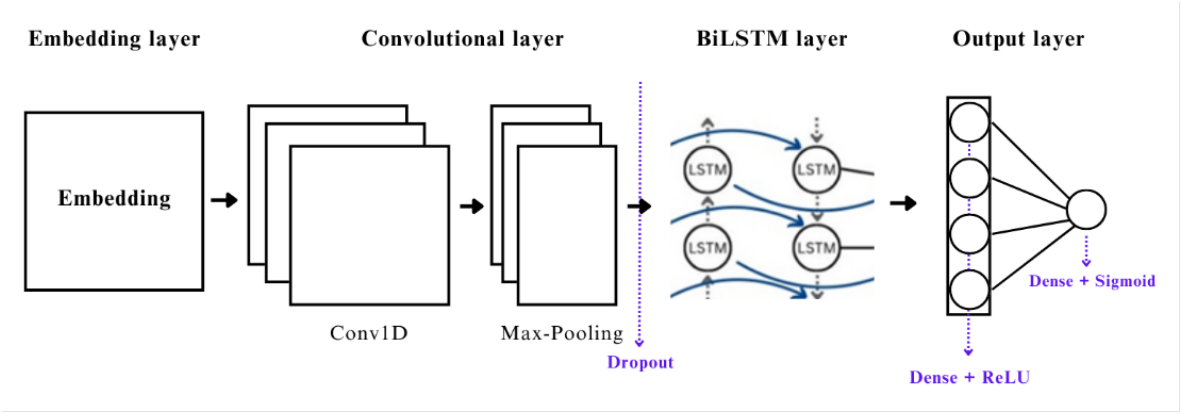

Figure 3.12 Adapted CNN-BiLSTM architecture for sentiment analysis.

Source: (Shahriar & Bashar, 2021)

The embedding layer and the convolutional layer are described the same as the previous model. While max-pooling effectively addressed the primary concern of overfitting, it was observed that a larger pooling size resulted in decreased accuracy. Through experimentation, it was determined that the optimum kernel size for the experiment was 2. This choice contributed to reducing the height of the input by half, maintaining a balance between capturing essential features and preventing overfitting.

The dropout layer was recognized as the optimal choice for reducing overfitting. Setting the dropout rate at 0.1 proved effective in encouraging the network to rely on a diverse set of weights and provides better generalization. This approach not only ensured convergence but also resulted in higher accuracy and a better understanding of the underlying data.

A bidirectional LSTM layer with 256 memory units is then introduced to capture both past and future context in the sequential data. It is designed to understand dependencies and relationships between words bidirectionally. Lastly, the dense layers were added to complete the output layer, the setting of this layer is same as previous model. For all applied neural network models, training was conducted with 10 epochs, a batch size of 32, and the Adam optimizer.

3.6.4 Bidirectional Encoder Representations from Transformers (BERT)

Figure 3.13 illustrates the BERT architecture for sentiment analysis used in this project. The implementation utilizes the DistilBERT variant, a lightweight version of BERT.

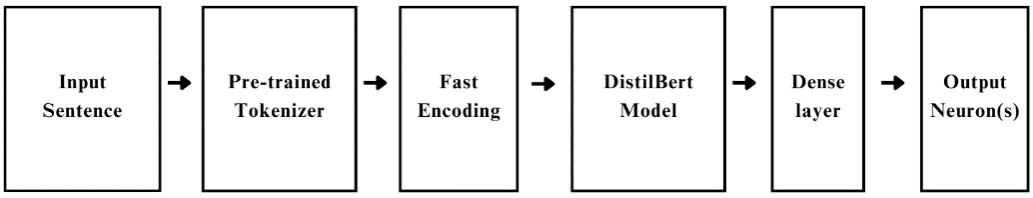

Figure 3.13 Adapted simple BERT architecture for sentiment analysis.

Source: (Farzana & Parde, 2020)

In BERT model, there are several versions. Here, the DistilBERT which is a distilled version of the original BERT model, is utilized for sentiment classification. It is proven that the model is possible to reduce the size of a BERT model by 40%, while still understanding 97% of the language and operating 60% faster (Sanh et al., 2019). Additionally, It runs 60% faster with 40% less parameters than google-bert/bert-base-uncased, while maintaining over 95% of BERT's performance on the GLUE language understanding benchmark. (Sanh et al., 2019).

To prepare the textual data for BERT, the DistilBERT tokenizer provided by the Hugging Face Transformers library has been utilized. This tokenizer was configured with the 'distilbert-base-uncased' pre-trained model and customized to ensure lowercase representation, which consistent with the model's requirements. Subsequently, the tokenizer was saved locally for further use. The fast-encoding process involves converting the textual data into numerical representations suitable for BERT. A chunking strategy is applied to efficiently manage the large datasets. To put it another way, the fast-encoding function leveraged the tokenized representations generated by the tokenizer and segmented the input text into manageable chunks to alleviate computational overhead. The encoded sequences were constrained to a maximum length of 400 tokens to maintain computational efficiency.

BERT is a network architecture was trained on a variety of datasets from a variety of articles written in a variety of languages. Therefore, training and fine-tuning the BERT layer for specific natural language processing tasks becomes unnecessary. Here, the model incorporated the DistilBERT transformer as the primary encoder. The model received word token IDs for input, allowing it to process text efficiently. Its output was obtained using the '[CLS]' token, capturing the summarized representation of the input. This representation then went through a dense layer with a sigmoid activation for binary sentiment classification. The overall model was compiled using the Adam optimizer and optimized for binary cross-entropy loss. It was trained on the IMDb movie reviews dataset for 10 epochs with utilizing a batch size of 32.

3.7 Step 7: Evaluation Metrics

The performance of classification model is verified using standard evaluation metrics, which are Accuracy, Precision, Recall and F1-score as to evaluate the testing dataset. Adam optimizer is used to calculate the accuracy of the experimented models.

3.7.1 Accuracy

Accuracy is used to measure that how well the devised model can automatically identify the data. It shows the percentage of labels that have been correctly classified. The mathematical formulation for accuracy is given in equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \times 100\%$$

where TP is the true positive, TN is the true negative, FN is the false negative and FP is the false positive. True positives are the number of instances that matched the target correctly, false positives are the number of instances that matched the target incorrectly but were counted as true positives, and true negatives are the number of instances that did not match the target correctly. To compute the calculation of Accuracy, confusion matrix is provided as below.

Table 3.2 Confusion matrix principle

	Positive	Negative
Positive	True Positive	False Positive
Negative	True Negative	False Negative

3.7.2 Precision

Precision (also called positive predictive value) measures the percentage of predictions made by the model that are correct. Precision is one of the indicators to evaluate machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions as equation:

$$Precision = \frac{TP}{TP + FP}$$

3.7.3 Recall

Recall (also known as sensitivity or the true positive rate) measures the percentage of relevant data points that were correctly identified by the model. It is the percentage of data samples that a machine learning model correctly identifies as belonging to a class of interest—the "positive class"—out of the total samples for that class. The mathematical formulation for recall is given in equation:

$$Recall = \frac{TP}{TP + FN}$$

3.7.4 F1-score

F1-score (or F-measure) is the harmonic mean (weighted average) of precision and recall. It is an overall measure of a model's accuracy that combines precision and recall. An F1-score is considered perfect when it is 1, while the model is a total failure when it is 0. The mathematical formulation for F1-score is given in equation:

$$F1\ score = \frac{2\ Prec\ Rec}{Prec + Rec}$$

where "Prec" is Precision and "Rec" is Recall.

Chapter 4 **Results and Discussions**

An in-depth analysis of the IMDb Movie Reviews dataset is explored. EDA is a primary step to begin with analyzing our data. Since the features of the dataset are limited, a variety of techniques to gain the maximum insights from this dataset was applied. Figure 4.1 shows the summary of the dataset.

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

Figure 4.1 Summary of the dataset with feature of 'Review' and target of 'Sentiment'

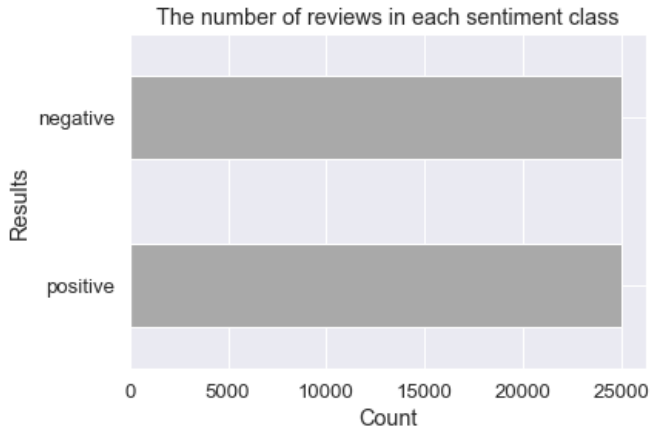

Figure 4.2 Sentiment count of the dataset

From the above figures, it is observed that there are no missing values in the entire dataset. The distribution of our data is good. In our entire dataset, there are 25,000 observation units in total (i.e., sentiment distribution: positive 50% and negative 50%) and 2 features. However, there is a need to check whether the dataset contains duplicated value.

Below figure shows sentiment count after removing the duplications.

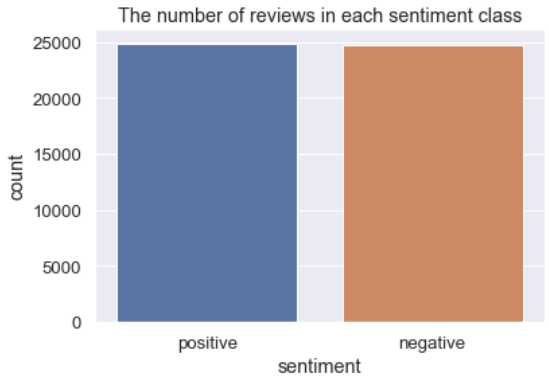

Figure 4.3 Bar chart of sentiment count after removing duplications.

Figure 4.9 shows the bar chart of sentiment count after removing the duplications. The data seems balanced after removing the duplications. Therefore, the next step is to present the percentage of samples in each class using a pie chart.

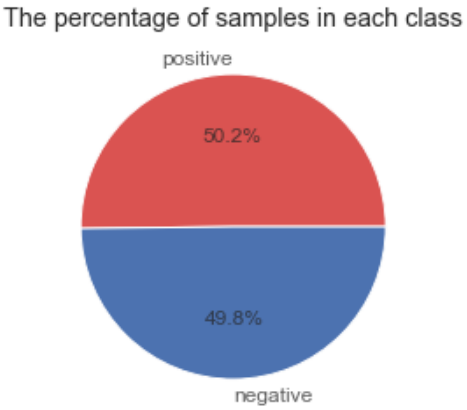

Figure 4.4 Pie chart of percentage of sentiment count after removing duplications.

From Figure 4.4, it is shown that the positive reviews have 50.2% of samples which is slightly higher than negative reviews that has a percentage of 49.8% of samples in each class. However, we can still consider that the data is balanced as the difference is mildly 0.4%.

The summary of the entire dataset will now look like:

	review	sentiment
count	49582	49582
unique	49582	2
top	From the acting, direction, scriptwriting and ...	positive
freq	1	24884

Figure 4.5 Summary of entire dataset after removing duplications.

Next, the word cloud is a visualization technique to represent the frequency of words in a text where the size of the word represents its frequency. Word cloud for positive words and the word cloud for negative words are shown in below figures.

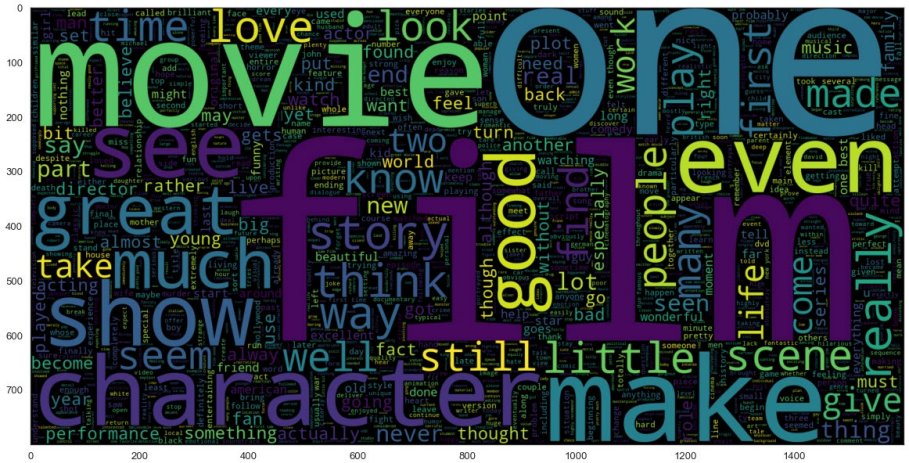

Figure 4.6 Word cloud for positive words.

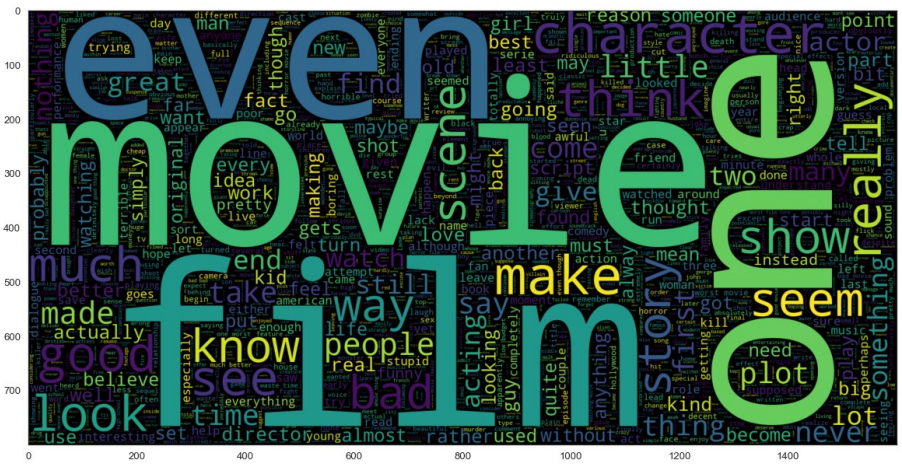

Figure 4.7 Word cloud for negative words.

The word cloud for positive and negative words have been generated separately, where the relevance of sentiment class (positive and or negative) is mapped (highest count words are represented in larger font size and lower frequent words are smaller). Ignoring the neutral words like “movie, film, even, etc.”, the words “great”, “good”, “love”, “well” have higher word count in positive class set which is showed in Figure 4.6. While the words “bad”, “stupid”, “never”, “little”, “awful” are frequent used in negative words which is showed in Figure 4.7.

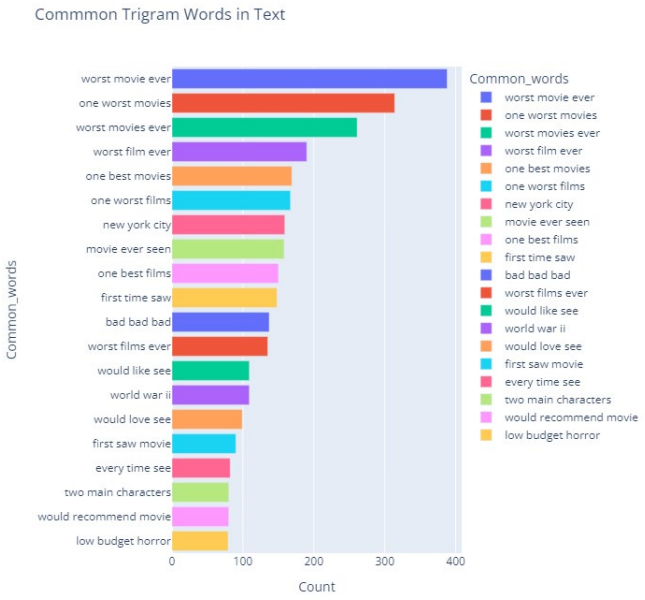

Figure 4.8 The common words in text by trigram analysis.

Figure 4.8 shows the common words in text by trigram analysis. By trigram analysis, which take three words into account, the most common words are surprisingly “worst movie ever”, “one worst movies”, “worst film ever”, “one best movies”, which give sentimental words as common words. So, the negative words are more common in the reviews than positive words by trigram analysis.

After conducting several experiments using different parameters, the result of the CNN models' accuracy was obtained. The best CNN model is selected and chosen to compare to other models. In this experiment, CNN 1 model obtained the highest accuracy (88.75%) among the 10 various CNN models. The experiment setting and results of various CNN models is shown in Table 4.1 while the scores comparison between the 10 various CNN models is shown in Table 4.2.

Table 4.1 The experiment setting and results of various CNN models.

Various CNN Models	Convolutional Layers	No. of Filters	Filter Size	Result (Accuracy %)
CNN1	2	64	4	88.75
CNN2	2	64	8	87.57
CNN3	2	64,32	3	87.31
CNN4	2	128	5	87.55
CNN5	2	256	5	87.53
CNN6	3	64	4	87.92
CNN7	3	64	8	87.74
CNN8	3	128	8, 5, 5	86.55
CNN9	3	256, 256, 128	8, 5, 3	87.79
CNN10	3	256	8	87.60

Table 4.1 shows the results of the experiment conducted with different settings of CNN models while fixing the number of epochs, batch size, vocab size, input vector size and increasing the number of convolutional layers from 2 to 3, number of filters from 32 to 256 and the filter size from 3 to 8. Overall, the CNN models have an average accuracy of 87.63% which is considered good.

Table 4.2 The scores comparison between the 10 various CNN models.

CNN Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CNN1	88.75	89.05	88.75	88.73
CNN2	87.57	87.64	87.57	87.56
CNN3	87.31	87.54	87.31	87.30
CNN4	87.55	87.81	87.55	87.52
CNN5	87.53	87.63	87.53	87.52
CNN6	87.92	87.94	87.18	87.18
CNN7	87.74	87.74	87.67	87.66
CNN8	86.55	86.76	86.59	86.53
CNN9	87.79	87.88	87.79	87.78
CNN10	87.60	87.66	87.60	87.59

From Table 4.2, CNN 1 model achieves better results than other CNN models with different parameters. Hence, it was chosen and used to combine with BiLSTM for another model –

CNN-BiLSTM. Additionally, this model is selected to be compared with other deep learning models.

In this project, deep learning algorithms are applied to the IMDB movie reviews dataset to analyse positive and negative sentiment by detecting the emotion of the reviewer through text that includes some emotional key words which determine the emotion of the reviewer. Some positive emotions represented by “good”, “like”, “best”, “great” and negative emotions include “worst”, “sadly”, “bad”, “awful”. To analyse these positive and negative emotions, deep learning algorithms and transformer model are applied. The different approaches are compared with the standard evaluation metrics (Accuracy, Precision, Recall, F1-Score) as shown in Table 4.3. The accuracy comparison of the implemented models is shown in Figure 4.9, while the validation accuracy of the implemented models is shown in Figure 4.10.

Table 4.3 The comparison scores of deep learning models on IMDB dataset.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
BERT	85.84	86.02	85.84	85.81
BiLSTM	86.34	86.34	86.32	86.33
CNN	88.75	89.05	88.75	88.73
CNN-BiLSTM	87.67	87.74	87.67	87.66

From the observations, it is found that the CNN model performs better than other deep learning algorithms (BERT, BiLSTM, CNN-BiLSTM), and gives the highest accuracy score among the implemented deep learning models, that is 88.75%. In terms of other evaluation metrics (Precision, Recall and F1-Score), CNN models give the highest score as well, which may suggest its capability in capturing the essential features for sentiment analysis in movie reviews. But in terms of recall and F1-score, the BiLSTM model are almost as good as the CNN, this shows that it is quite effective at weighing both correctness and completeness when identifying sentiments. Although the accuracy of the BERT model is marginally lower, it still performs well overall, this suggests that the pre-trained language model is reliable when handling natural language data. Overall, a competitive performance is shown by the implemented deep learning models on the IMDb dataset. Interestingly, the combined CNN-BiLSTM model does not significantly outperform the single CNN and BiLSTM models in terms of accuracy, indicating that the combination of these architectures may not always provide compounded gains for this particular task.

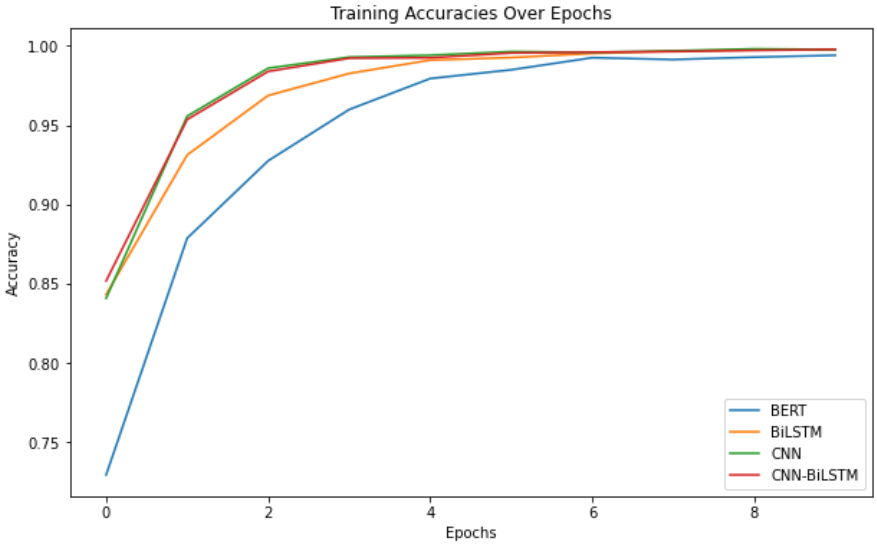

Figure 4.9 The accuracy comparison of the implemented models.

From Figure 4.9, it is observed that the CNN model and CNN-LSTM starts converging earlier than BiLSTM and BERT model on IMDb movie reviews dataset. As we can see that the BERT model exhibits rapid learning in the initial stages, performing better than the other models early on. The CNN model, while starting lower, shows a consistent upward trajectory, so this indicate that this model has a steady learning rate. BiLSTM model has stable performance during the training process as well. Overall, the average performance of all the models remains high, especially towards the end of the training period.

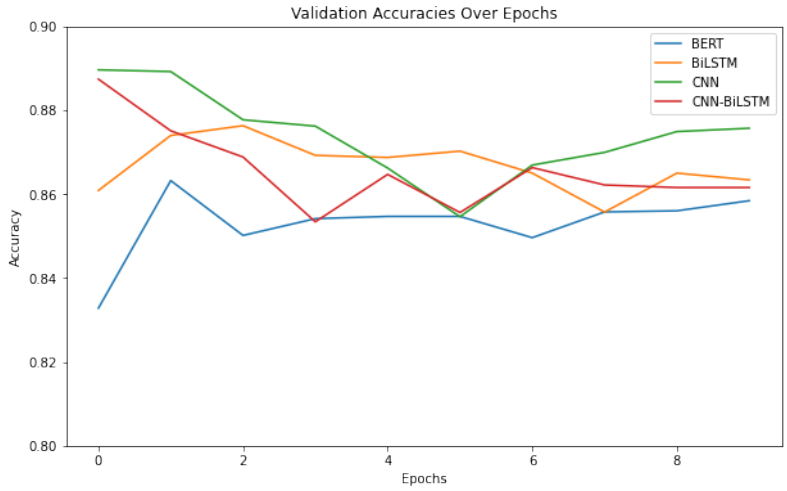

Figure 4.10 The validation accuracy of the implemented models.

Figure 4.10 shows the validation accuracy of the implemented models, it illustrates the variability in the models' performance. The BERT model shows noticeable fluctuations, which may indicate overfitting or sensitivity to dataset nuances. Although BiLSTM's performance varies too, it shows a general downward trend, which may cause one to wonder if it is stable over various datasets. The most inconsistent performance is seen in the CNN model, this indicates potential issues with generalization. Interestingly, the CNN-BiLSTM model appears to balance adaptability and stability from epoch 6 to the end.

Table 4.4 The comparison of deep learning models in the literature review.

DL Models	Accuracy (%)
BERT	85.84
BiLSTM	86.34
CNN	88.75
CNN-BiLSTM	87.67
BERT (Arora et al., 2023)	92.40
LSTM (Amulya et al., 2022)	71
LSTM (Qaisar, 2020)	89.9
RNN (Cen et al., 2020)	68.64
CNN (Cen et al., 2020)	88.22
CNN-LSTM (Hoque et al., 2019)	89.20
MLP (Hoque et al., 2019)	86.74

Table 4.4 compares the results of the implemented DL models with those reported in the literature (Amulya et al., 2022; Arora et al., 2023; Cen et al., 2020; Hoque et al., 2019; Qaisar, 2020). When comparing the results to those found in the literature, our models perform well in the context of current research. The CNN model implemented in this study aligns closely with accuracy levels reported by (Cen et al., 2020), which suggests consistency with current findings. The CNN model outperforms the LSTM and RNN models reported by (Amulya et al., 2022) and (Cen et al., 2020), respectively, which may highlight the effectiveness of convolutional approaches in text classification tasks. In addition, the CNN model's performance is comparable to the CNN-LSTM hybrid reported by (Hoque et al., 2019), precisely, the CNN-LSTM model obtained the accuracy slightly higher (0.45%) than our CNN model. This may raise questions about the added complexity of combined models since our combined CNN-BiLSTM model did not perform better than the single CNN. Hence, this suggests a potential area for further investigation, particularly examining the conditions under which model ensembles may offer distinct advantages. On the other hand, the BERT model falls short of the performance achieved by (Arora et al., 2023), which could be attributed to differences in the version of model architecture used. So, it could be said that the architecture of DistilBERT is very similar to that of BERT, in which managing to retain 97% performance of the BERT-base, but there exists a noticeable discrepancy in accuracy when applied to large datasets. However, DistilBERT is much more efficient than the original BERT model, despite having slightly worse performance. With 40% fewer parameters and 60% faster operation, this model is an excellent choice for applications where efficiency is crucial.

Chapter 5 **Results and Conclusion**

This final chapter presents a comprehensive summary of the project and summarizes the findings from the comprehensive analysis conducted on sentiment analysis of IMDb movie reviews using deep learning techniques. The objective was to explore the effectiveness of four distinct models: CNN, BERT, BiLSTM, and CNN-BiLSTM in classifying sentiments expressed in movie reviews as either positive or negative. The chapter summarizes the results obtained from the experiments, discusses the implications of these findings, and concludes with reflections on what the project has contributed to the field of sentiment analysis.

This project begins with problem identification, which focuses on understanding the importance of sentiment analysis in today's data-driven world. Since the subject area was chosen to focus on deep learning techniques for sentiment analysis, the sentiment classification-related works were explored and summarized in literature review, Chapter 2. In the discussion of literature reviews, it is concluded that deep learning models outperform the machine learning model, especially the CNN model. Previous studies demonstrated the effectiveness of CNN and LSTM models in capturing textual features and sentiments. Furthermore, the research also highlighted BERT's ability to understand the context of words in a sentence significantly, improving sentiment analysis accuracy. Moreover, the exploration of hybrid models, such as CNN-LSTM, has shown potential improvements in accuracy through the utilization of various architectures' advantages. After discussing literature reviews, the four models (CNN, BiLSTM, CNN-BiLSTM, BERT) were decided to implement this project.

Before the implementation of the models, this project utilized the IMDb movie reviews dataset from https://ai.stanford.edu/~amaas/data/sentiment/ that contains 50,000 reviews categorized into positive and negative sentiments. The dataset is balanced since the number of each class is the same. Followed by data preprocessing that involved cleaning the dataset by removing duplicates, removing stopwords, lowercasing, tokenization, lemmatization, and word embeddings was employed by using an embedding layer that Keras provided with a vocabulary consisting of 143380 unique words and vector space with 4000 dimensions. The word embedding layer was used to train the training samples for numerical representation of text. The Exploratory Data Analysis was conducted during semester 1 to gain insights into the dataset, including sentiment distribution and common words analysis. From trigram analysis, it was found that the common words are most likely from the negative reviews, such as "worst film ever", "worst movie ever", etc.

Using the IMDb movie reviews dataset, the four deep learning models (CNN, CNN-BiLSTM, BiLSTM and BERT) were implemented and evaluated based on accuracy, precision, recall, and F1-score. After testing, it was found that the Convolutional Neural Network (CNN) model performed better than the other models, obtaining an accuracy rate of 88.75%. This result emphasizes how well the CNN model can extract important features for sentiment classification from textual input. The BERT model, while not achieving the highest accuracy when applying its distilled version - DistilBERT, still demonstrated substantial effectiveness with an accuracy of 85.84%. This has demonstrated its robustness in understanding language nuances. The BiLSTM model followed closely, reflecting its strength in capturing the information from both directions within text data through its accuracy of 86.34%. The hybrid CNN-BiLSTM model achieved an accuracy of 87.67% by utilizing the advantages of both

CNN and BiLSTM architectures. While it fell short of the CNN model's performance, this demonstrated how hybrid models can improve sentiment analysis tasks. The outcomes of the experiment highlight how important model architecture is to sentiment analysis. The CNN model's superior performance can be due to its ability to process local features such as n-grams in text in data, allowing for better feature extraction from the text. On the other hand, the BERT model's pre-trained nature allows it to bring a broad understanding of language context into the analysis, which is crucial for detecting sentiment nuances. The BiLSTM model's comparative performance demonstrates the significance of sequential data processing in sentiment analysis since it effectively collects contextual information from the past and the future. The CNN-BiLSTM's performance, meanwhile, indicates that although integrating various model strengths can be advantageous, the integration must be adjusted for the specifics of the task and dataset.

This research project has contributed to the understanding of how different deep learning techniques can be applied to sentiment analysis of IMDb movie reviews. By comparing the performance of CNN, BERT, BiLSTM, and CNN-BiLSTM models, the CNN model, with its high accuracy, emerged as the most effective approach for this specific dataset, indicating its potential for similar sentiment analysis tasks. The findings also demonstrate the continued need for research in this area, especially when it comes to maximizing the potential of hybrid models like CNN-BiLSTM. It's noteworthy to mention that while CNN models in existing literature demonstrate high accuracy, there exists considerable variance in their reported accuracy values, this can be due to differences in IMDb dataset importation methods. some prior works importing the TensorFlow module (e.g., tf.keras.datasets.imdb) tend to yield better accuracy compared to those applying original text files or CSV files. Additionally, the variability in accuracy values could be attributed to differences in data preprocessing techniques, impacting model training outcomes. Therefore, future research could explore not only the impact of further tuning, but also the data preprocessing method, the integration of additional model architectures, and extend the application of these techniques across various datasets and domains.

References

Ali, N. M., Abd El Hamid, M. M., & Youssif, A. (2019). Sentiment analysis for movies reviews dataset using deep learning models. *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol*, *9*.

Amulya, K., Swathi, S., Kamakshi, P., & Bhavani, Y. (2022). Sentiment analysis on IMDB movie reviews using machine learning and deep learning algorithms. 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT),

Anees, A. F., Shaikh, A., Shaikh, A., & Shaikh, S. (2020). Survey paper on sentiment analysis: Techniques and challenges. *EasyChair2516-2314*.

Arora, K., Gupta, N., & Pathak, S. (2023). Sentimental Analysis on IMDb Movies Review using BERT. 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC),

Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., & Acharya, U. R. (2021). ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis. *Future Generation Computer Systems*, *115*, 279-294.

Cen, P., Zhang, K., & Zheng, D. (2020). Sentiment analysis using deep learning approach. *J. Artif. Intell*, *2*(1), 17-27.

Charitha, N. S. L. S., Yasaswi, K., Rakesh, V., Varun, M., Yeswanth, M., & Kiran, J. S. (2023). Comparative Study of Algorithms for Sentiment Analysis on IMDB Movie Reviews. 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS),

Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Systems with Applications*, *72*, 221-230.

Chitkara, P., Modi, A., Avvaru, P., Janghorbani, S., & Kapadia, M. (2019). Topic spotting using hierarchical networks with self attention. *arXiv preprint arXiv:1904.02815*.

Farzana, S., & Parde, N. (2020). Exploring MMSE Score Prediction Using Verbal and Non-Verbal Cues. Interspeech,

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A survey on text classification algorithms: From text to predictions. *Information*, *13*(2), 83.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), 602-610.

Haque, M. R., Lima, S. A., & Mishu, S. Z. (2019). Performance analysis of different neural networks for sentiment analysis on IMDb movie reviews. 2019 3rd International conference on electrical, computer & telecommunication engineering (ICECTE),

Heaton, J. (2018). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618. *Genetic programming and evolvable machines*, *19*(1-2), 305-307.

Hettiarachchi, H., Adedoyin-Olowe, M., Bhogal, J., & Gaber, M. M. (2021). DAAI at CASE 2021 task 1: Transformer-based multilingual socio-political and crisis event detection. Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021),

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hoque, M. T., Islam, A., Ahmed, E., Mamun, K. A., & Huda, M. N. (2019). Analyzing performance of different machine learning approaches with doc2vec for classifying sentiment of bengali natural language. 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE),

Jeong, H., Shin, D., & Choi, J. (2011). Ferom: Feature extraction and refinement for opinion mining. *Etri Journal*, *33*(5), 720-730.

Jogin, M., Madhulika, M., Divya, G., Meghana, R., & Apoorva, S. (2018). Feature extraction using convolution neural networks (CNN) and deep learning. 2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT),

Joloudari, J. H., Hussain, S., Nematollahi, M. A., Bagheri, R., Fazl, F., Alizadehsani, R., Lashgari, R., & Talukder, A. (2023). BERT-deep CNN: State of the art for sentiment analysis of COVID-19 tweets. *Social Network Analysis and Mining*, *13*(1), 99.

Ju, Y., Sun, G., Chen, Q., Zhang, M., Zhu, H., & Rehman, M. U. (2019). A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting. *Ieee Access*, *7*, 28309-28318.

Kadu, S., & Joshi, B. (2022). Text-Based Sentiment Analysis Using Deep Learning Techniques. In *Deep Learning for Social Media Data Analytics* (pp. 81-100). Springer.

Kim, K., Aminanto, M. E., Tanuwidjaja, H. C., Kim, K., Aminanto, M. E., & Tanuwidjaja, H. C. (2018). Classical machine learning and its applications to IDS. *Network Intrusion Detection using Deep Learning: A Feature Learning Approach*, 13-26.

Kobayashi, N., Inui, K., & Matsumoto, Y. (2007). Extracting aspect-evaluation and aspect-of relations in opinion mining. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL),

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

Lee, C.-C., Gao, Z., & Tsai, C.-L. (2020). BERT-Based stock market sentiment analysis. 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan),

Li, J., Zhao, S., Yang, J., Huang, Z., Liu, B., Chen, S., Pan, H., & Wang, Q. (2020). WCP-RNN: a novel RNN-based approach for Bio-NER in Chinese EMRs: Paper ID: FC_17_25. *The journal of supercomputing*, *76*, 1450-1467.

Li, Y. (2022). Research and application of deep learning in image recognition. 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA),

Liao, S., Wang, J., Yu, R., Sato, K., & Cheng, Z. (2017). CNN for situations understanding based on sentiment analysis of twitter data. *Procedia computer science*, *111*, 376-381.

Lim, W. L., Ho, C. C., & Ting, C.-Y. (2020). Tweet sentiment analysis using deep learning with nearby locations as features. Computational Science and Technology: 6th ICCST 2019, Kota Kinabalu, Malaysia, 29-30 August 2019,

Liu, B. (2022). *Sentiment analysis and opinion mining*. Springer Nature.

Liu, W., Liu, P., Yang, Y., Gao, Y., & Yi, J. (2017). An attention-based syntax-tree and tree-LSTM model for sentence summarization. *International Journal of Performability Engineering*, *13*(5), 775.

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16-32.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning--based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, *54*(3), 1-40.

Muthusankar, D. D., Kaladevi, D. P., Sadasivam, D. V., & Praveen, R. (2023). BIDRN: A Method of Bidirectional Recurrent Neural Network for Sentiment Analysis. *arXiv preprint arXiv:2311.07296*.

Nanda, C., Dua, M., & Nanda, G. (2018). Sentiment analysis of movie reviews in hindi language using machine learning. 2018 International Conference on Communication and Signal Processing (ICCSP),

Nguyen, T.-T., Phan, T. T. V., Ho, D.-D., Pradhan, A. M. S., & Huynh, T.-C. (2022). Deep learning-based autonomous damage-sensitive feature extraction for impedance-based prestress monitoring. *Engineering Structures*, *259*, 114172.

Niu, X., Hou, Y., & Wang, P. (2017). Bi-directional LSTM with quantum attention mechanism for sentence modeling. Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24,

Nowak, J., Taspinar, A., & Scherer, R. (2017). LSTM recurrent neural networks for short text and sentiment classification. Artificial Intelligence and Soft Computing: 16th International Conference, ICAISC 2017, Zakopane, Poland, June 11-15, 2017, Proceedings, Part II 16,

Phung, V. H., & Rhee, E. J. (2019). A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, *9*(21), 4500.

Qaisar, S. M. (2020). Sentiment analysis of IMDb movie reviews using long short-term memory. 2020 2nd International Conference on Computer and Information Sciences (ICCIS),

Ranjbarzadeh, R., Jafarzadeh Ghoushchi, S., Anari, S., Safavi, S., Tataei Sarshar, N., Babaee Tirkolaee, E., & Bendechache, M. (2022). A deep learning approach for robust, multi-oriented, and curved text detection. *Cognitive computation*, 1-13.

Rehman, A. U., Malik, A. K., Raza, B., & Ali, W. (2019). A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis. *Multimedia Tools and Applications*, *78*, 26597-26613.

Salehi, A. W., Khan, S., Gupta, G., Alabduallah, B. I., Almjally, A., Alsolai, H., Siddiqui, T., & Mellit, A. (2023). A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope. *Sustainability*, *15*(7), 5930.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673-2681.

Shahriar, F., & Bashar, M. A. (2021). Automatic Monitoring Social Dynamics During Big Incidences: A Case Study of COVID-19 in Bangladesh. *arXiv preprint arXiv:2101.09667*.

Sinha, S., Jayan, A., & Kumar, R. (2022). An Analysis and Comparison Of Deep-Learning Techniques and Hybrid Model for Sentiment Analysis for Movie Review. 2022 3rd International Conference for Emerging Technology (INCET),

Stymne, S. (2011). Pre-and postprocessing for statistical machine translation into germanic languages. Proceedings of the ACL 2011 Student Session,

Sur, C. (2020). RBN: enhancement in language attribute prediction using global representation of natural language transfer learning technology like Google BERT. *SN Applied Sciences*, *2*(1), 22.

Tam, S., Said, R. B., & Tanriöver, Ö. Ö. (2021). A ConvBiLSTM deep learning model-based approach for Twitter sentiment classification. *Ieee Access*, *9*, 41283-41293.

Tripathy, A., Agrawal, A., & Rath, S. K. (2016). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, *57*, 117-126.

Vijayarani, S., & Janani, R. (2016). Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, *3*(1), 37-47.

Wang, C., Nulty, P., & Lillis, D. (2020). A comparative study on word embeddings in deep learning for text classification. Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval,

Wang, N., He, M., Sun, J., Wang, H., Zhou, L., Chu, C., & Chen, L. (2019). ia-PNCC: Noise Processing Method for Underwater Target Recognition Convolutional Neural Network. *Computers, Materials & Continua*, *58*(1).

Wang, S., Huang, M., & Deng, Z. (2018). Densely connected CNN with multi-scale feature attention for text classification. IJCAI,

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, *33*, 5776-5788.

Xu, H., Liu, Y., Shu, C.-M., Bai, M., Motalifu, M., He, Z., Wu, S., Zhou, P., & Li, B. (2022). Cause analysis of hot work accidents based on text mining and deep learning. *Journal of Loss Prevention in the Process Industries*, *76*, 104747.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Yenter, A., & Verma, A. (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. 2017 IEEE 8th annual ubiquitous computing, electronics and mobile communication conference (UEMCON),

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers),

Zhou, Q., & Wu, H. (2018). NLP at IEST 2018: BiLSTM-attention and LSTM-attention via soft voting in emotion classification. Proceedings of the 9th workshop on computational approaches to subjectivity, sentiment and social media analysis,

appendix-maincode

March 28, 2024

0.0.1 Data Cleaning

[ ]:
```
# remove duplicated reviews
df.drop_duplicates(keep='first', inplace=True)
print(df.shape)

# download the stopwords corpus
import nltk
nltk.download('stopwords')
stop = stopwords.words('english')
wl = WordNetLemmatizer()

mapping = {"ain't": "is not", "aren't": "are not","can't": "cannot",
           "'cause": "because", "could've": "could have", "couldn't": "could␣
 ↪not",
           "didn't": "did not",  "doesn't": "does not", "don't": "do not",␣
 ↪"hadn't": "had not",
           "hasn't": "has not", "haven't": "have not", "he'd": "he␣
 ↪would","he'll": "he will",
           "he's": "he is", "how'd": "how did", "how'd'y": "how do you",␣
 ↪"how'll": "how will",
           "how's": "how is",  "I'd": "I would", "I'd've": "I would have",␣
 ↪"I'll": "I will",
           "I'll've": "I will have","I'm": "I am", "I've": "I have", "i'd": "i␣
 ↪would",
           "i'd've": "i would have", "i'll": "i will",  "i'll've": "i will␣
 ↪have",
           "i'm": "i am", "i've": "i have", "isn't": "is not", "it'd": "it␣
 ↪would",
           "it'd've": "it would have", "it'll": "it will", "it'll've": "it will␣
 ↪have",
           "it's": "it is", "let's": "let us", "ma'am": "madam", "mayn't": "may␣
 ↪not",
           "might've": "might have","mightn't": "might not","mightn't've":␣
 ↪"might not have",
           "must've": "must have", "mustn't": "must not", "mustn't've": "must␣
 ↪not have",
```

```
          "needn't": "need not", "needn't've": "need not have","o'clock": "of␣
↪the clock",
          "oughtn't": "ought not", "oughtn't've": "ought not have", "shan't":␣
↪"shall not",
          "sha'n't": "shall not", "shan't've": "shall not have", "she'd": "she␣
↪would",
          "she'd've": "she would have", "she'll": "she will", "she'll've":␣
↪"she will have",
          "she's": "she is", "should've": "should have", "shouldn't": "should␣
↪not",
          "shouldn't've": "should not have", "so've": "so have","so's": "so␣
↪as", "this's": "this is",
          "that'd": "that would", "that'd've": "that would have", "that's":␣
↪"that is",
          "there'd": "there would", "there'd've": "there would have",␣
↪"there's": "there is",
          "here's": "here is","they'd": "they would", "they'd've": "they would␣
↪have",
          "they'll": "they will", "they'll've": "they will have", "they're":␣
↪"they are",
          "they've": "they have", "to've": "to have", "wasn't": "was not",␣
↪"we'd": "we would",
          "we'd've": "we would have", "we'll": "we will", "we'll've": "we will␣
↪have",
          "we're": "we are", "we've": "we have", "weren't": "were not",
          "what'll": "what will", "what'll've": "what will have","what're":␣
↪"what are",
          "what's": "what is", "what've": "what have", "when's": "when is",␣
↪"when've": "when have",
          "where'd": "where did", "where's": "where is", "where've": "where␣
↪have", "who'll": "who will",
          "who'll've": "who will have", "who's": "who is", "who've": "who␣
↪have", "why's": "why is",
          "why've": "why have", "will've": "will have", "won't": "will not",␣
↪"won't've": "will not have",
          "would've": "would have", "wouldn't": "would not", "wouldn't've":␣
↪"would not have",
          "y'all": "you all", "y'all'd": "you all would","y'all'd've": "you␣
↪all would have",
          "y'all're": "you all are","y'all've": "you all have","you'd": "you␣
↪would",
          "you'd've": "you would have", "you'll": "you will", "you'll've":␣
↪"you will have",
          "you're": "you are", "you've": "you have" }
```

```
#function to clean data
import nltk
nltk.download('wordnet')
def clean_text(text,lemmatize = True):
    soup = BeautifulSoup(text, "html.parser") #remove html tags
    text = soup.get_text()
    text = ' '.join([mapping[t] if t in mapping else t for t in text.split("␣
 ↪")]) #expanding chatwords and contracts clearing contractions
    emoji_clean= re.compile("["
                            u"\U0001F600-\U0001F64F"  # emoticons
                            u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                            u"\U0001F680-\U0001F6FF"  # transport & map symbols
                            u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                            u"\U00002702-\U000027B0"
                            u"\U000024C2-\U0001F251"
                            "]+", flags=re.UNICODE)
    text = emoji_clean.sub(r'',text)
    text = re.sub(r'\.(?=\S)', '. ',text) #add space after full stop
    text = re.sub(r'http\S+', '', text) #remove urls
    text = "".join([word.lower() for word in text if word not in string.
 ↪punctuation]) #remove punctuation
    #tokens = re.split('\W+', text) #create tokens
    if lemmatize:
        text = " ".join([wl.lemmatize(word) for word in text.split() if word␣
 ↪not in stop and word.isalpha()]) #lemmatize
    else:
        text = " ".join([word for word in text.split() if word not in stop and␣
 ↪word.isalpha()])
    return text

df['review']=df['review'].apply(clean_text,lemmatize = True)
```

[ ]:
```
#converting target variable to numeric labels
df.sentiment = [ 1 if each == "positive" else 0 for each in df.sentiment]
df.head()
```

0.0.2 Converting sentences into tokens

[ ]:
```
# Tonenizer
token = Tokenizer()
token.fit_on_texts(df['review'].values)

#padding
max_len = 4000
X = token.texts_to_### Train test split
```

```
label = df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, label, test_size=0.2,␣
 ↪random_state=42)
X_train.shape, X_test.shapesequences(df['review'].values)
X = pad_sequences(X, maxlen=max_len)
```

0.0.3 Train test split

[ ]:
```
label = df['sentiment']
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, label, test_size=0.2,␣
 ↪random_state=42)
X_train.shape, X_test.shape
```

0.0.4 CNN Model

[ ]:
```
# Define CNN1 model
embedding_dim = 300

model = Sequential()
model.add(Embedding(vocab, embedding_dim, input_length=X.shape[1]))
model.add(Conv1D(64, 4, activation="relu", padding='same'))
model.add(MaxPooling1D(2))
model.add(Dropout(0.1))

model.add(Conv1D(64, 4, activation='relu', padding='same'))
model.add(MaxPooling1D(2))
model.add(Dropout(0.1))

model.add(Flatten())
model.add(Dropout(0.1))
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()

# compile and train the model
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪epochs=10, batch_size=32, validation_split=0.1)
```

[ ]:
```
# Evaluate the model on the test set
results = model.evaluate(X_test, y_test, verbose=0)
```

0.0.5 BiLSTM

[ ]:
```
embedding_dim = 300

model = Sequential()
model.add(Embedding(vocab, embedding_dim, input_length=X.shape[1]))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Bidirectional(LSTM(256)))
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.summary()

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪epochs=10, batch_size=32, validation_split=0.1)
result = model.evaluate(X_test,y_test)
```

0.0.6 CNN-BiLSTM

[ ]:
```
model = Sequential()
model.add(Embedding(vocab, embedding_dim, input_length=X.shape[1]))

# Convolutional layer
model.add(Conv1D(64, 4, activation='relu'))
model.add(MaxPooling1D(2))

model.add(Conv1D(64, 4, activation='relu'))
model.add(MaxPooling1D(2))
model.add(Dropout(0.1))

model.add(Conv1D(64, 4, activation='relu'))
model.add(MaxPooling1D(2))
model.add(Dropout(0.1))

# Bidirectional LSTM layer
model.add(Bidirectional(LSTM(256)))

# Dense layers
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()

# Compile the model
```

```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
# Train the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪epochs=10, batch_size=32, validation_split=0.1)
```

0.0.7 Bert Model

Fast encoding

[ ]:
```
from tokenizers import BertWordPieceTokenizer

# First load the real tokenizer
tokenizer = transformers.DistilBertTokenizer.
 ↪from_pretrained('distilbert-base-uncased' , lower = True)

# Save the loaded tokenizer locally
tokenizer.save_pretrained('.')

# Reload it with the huggingface tokenizers library
fast_tokenizer = BertWordPieceTokenizer('vocab.txt', lowercase=True)
fast_tokenizer
```

[ ]:
```
def fast_encode(texts, tokenizer, chunk_size=256, maxlen=400):

    tokenizer.enable_truncation(max_length=maxlen)
    tokenizer.enable_padding(length=maxlen)
    all_ids = []

    for i in range(0, len(texts), chunk_size):
        text_chunk = texts[i:i+chunk_size].tolist()
        encs = tokenizer.encode_batch(text_chunk)
        all_ids.extend([enc.ids for enc in encs])

    return np.array(all_ids)
```

[ ]:
```
x_train = fast_encode(x_train.values, fast_tokenizer, maxlen=400)
x_test = fast_encode(x_test.values, fast_tokenizer, maxlen=400)
```

[ ]:
```
def build_model(transformer, max_len=400):

    input_word_ids = Input(shape=(max_len,), dtype=tf.int32,␣
 ↪name="input_word_ids")
    sequence_output = transformer(input_word_ids)[0]
    cls_token = sequence_output[:, 0, :]
    out = Dense(1, activation='sigmoid')(cls_token)
```

```
    model = Model(inputs=input_word_ids, outputs=out)
    model.compile(Adam(learning_rate=2e-5), loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

    return model
```

[ ]:
```
bert_model = transformers.TFDistilBertModel.
 ↪from_pretrained('distilbert-base-uncased')
```

[ ]:
```
model = build_model(bert_model, max_len=400)
model.summary()
```

[ ]:
```
history = model.fit(x_train,y_train,batch_size = 32␣
 ↪,validation_data=(x_test,y_test),epochs = 10)
```