Heaven's Light Is Our Guide

A Study of Bengali Word Embedding Based on Neural Networks

A Project/Thesis Submitted in Partial
fulfillment for the requirement of the
degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

Nafisa Tabassum Shamma
Roll No. 1710018

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

September, 2023

A Study of Bengali Word Embedding Based on Neural Networks

A Project/Thesis Submitted in Partial
fulfillment for the requirement of the
degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

Nafisa Tabassum Shamma
Roll No. 1710018

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

September, 2023

Acknowledgement

This thesis has been submitted to the Department of Electrical and Computer Engineering of Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh, for the partial fulfillment of the requirements for the degree of B.Sc. in Electrical & Computer Engineering. Thesis title regards to "**A STUDY OF BENGALI WORD EMBEDDING BASED ON NEURAL NETWORKS** ".

First and foremost, I am grateful and indebted to my thesis advisor, Sagor Chandro Bakchy, Head of the Department of Electrical & Computer Engineering, for his compassion and knowledge throughout the duration of my thesis. I will always be grateful to him for his invaluable guidance, advice, encouragement, and cordial and affable contribution to my thesis and also for providing all kind of laboratory facilities.

I would also like to acknowledge the support and encouragement of Rakibul Hassan, Assistant Professor in the Department of Electrical and Computer Engineering.

I would like to express my deep gratitude to my husband. He has always encouraged me during the research. I am also grateful to the administration of Rajshahi University of Engineering & Technology for providing a self-sufficient Under Graduate (UG) lab.

Finally, I want to thank the most important and the closest persons of my life and my parents for giving a big support to me.

Nafisa Tabassum Shamma
Roll No. 1710018

September, 2023
RUET, Rajshahi

CERTIFICATE

This is to certify that the thesis entitled "***A STUDY OF BENGALI WORD EMBEDDING BASED ON NEURAL NETWORKS***" by Nafisa Tabassum Shamma (Roll No. 1710018), has been carried out under my direct supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

Thesis Supervisor:

.....
Sagor Chandro Bakchy
Head of Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

CERTIFICATE

This is to certify that the thesis entitled "***A STUDY OF BENGALI WORD EMBEDDING BASED ON NEURAL NETWORKS***" has been corrected according to my suggestion and guidance as an external. The quality of the thesis is satisfactory.

External Member:

.....
Md. Omaer Faruq Goni
Lecturer
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

Declaration

This is to certify that this thesis work has been done by me and not submitted elsewhere for the award of any degree or diploma.

.....

Nafisa Tabassum Shamma

B.Sc. Engineering

Roll: 1710018

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

Abstract

This research endeavors to advance the understanding of Bengali word embeddings by developing and assessing a novel model built using the Gensim-implemented FastText algorithm, aiming to augment the performance in natural language processing tasks for Bengali, a language characterized by intricate morphology and syntax. The global objective of this study is to enrich the repertoire of resources available for Bengali language processing, considering the scarcity of research in this domain compared to languages like English. My substantial contribution involves constructing a word embedding model trained on a substantial subset, containing 30,000,000 sentences of a larger 14GB corpus, and implementing advanced neural network architectures for practical application in Bengali fake news detection, pushing the boundaries of existing knowledge in Bengali language processing. The model was meticulously optimized through unsupervised training, overcoming evaluation challenges due to the lack of ground truth labels. Comparative analysis was carried out against Facebook's pre-trained FastText model, serving as a robust benchmark. Results from experiments leveraging Recurrent and Convolutional Neural Networks demonstrated that the proposed model, with an accuracy of 71% for Hybrid conv-LSTM based classifier and 70% for CNN based classifier, rivals the existing models in performance metrics such as accuracy, precision, recall, F1-score, showcasing its applicability in real-world scenarios like fake news detection. This work illuminates the possibilities of refining and adapting established algorithms for less-researched languages and accentuates the significance of word embeddings in multilingual natural language processing.

Contents

List of Figures

List of Tables

List of Abbreviations

Short Form	Abbreviations
NLP	Natural Language processing
CNN	Convolutional Neural Network
SA	Sentiment Analysis
DL	Deep Learning
AI	Artificial Intelligent
ML	Machine Learning
BERT	Bidirectional Encoder Representations From Transformers
SG	Skip Gram
GPU	Graphical Processing Unit
Bi	Bidirectional
LR	Logistic Regression
RF	Random Forest
LSTM	Long short-term memory
CBOW	Continuous Bag of Words

Chapter 1

Introduction

The motivations for picking this specific topic are explained in this chapter's section on reasons. The works cited section summarizes the research conducted in the field of Bengali Word Embedding to date. In the subsection entitled Thesis Objective, the mission, vision, anticipated outcome, and applications of Bengali word embedding are discussed. In the Dataset Description subsection, a brief overview of the dataset is provided.

1.1 Motivation

In recent years, Natural Language Processing (NLP) has made significant strides, revolutionizing applications spanning from machine translation to sentiment analysis. Word embeddings, which represent words as dense vectors in continuous vector spaces, are central to this development. While extensive research has been conducted on word embeddings for major languages such as English, the linguistic diversity of the world necessitates that attention be paid to underrepresented languages such as Bengali.

Bengali is one of the most extensively spoken languages in the globe, with over 230 million native speakers. It has enormous cultural, historical, and regional significance, and its influence extends well beyond South Asia. Despite its prevalence, the development of Bengali-specific NLP resources, particularly word embeddings, has been relatively limited.

This thesis focuses on the construction, refinement, and evaluation of Bengali word embeddings in an effort to fill this significant void in NLP research. The scarcity of specialized resources for Bengali poses a challenge for the effective implementation of NLP applications within this linguistic context. This research seeks to strengthen the foundation for future developments in Bengali NLP by providing a thorough analysis of Bengali word embeddings.

The implications of this study extend far beyond the academic community. Improved Bengali word embeddings would facilitate advancements in multiple fields, including sentiment analysis for social media, automated content generation, and efficient information retrieval for Bengali-language websites and documents. In addition, it would facilitate the creation of Bengali-specific chatbots, virtual assistants, and other AI-driven applications.

In addition, this dissertation may contribute to cross-lingual NLP research. Effective Bengali word embeddings could serve as a stepping stone for the development of techniques that bridge the divide between Bengali and other languages, thereby facilitating smoother

language translation, information retrieval, and sentiment analysis across linguistic boundaries.

In a world that is swiftly globalizing, linguistic diversity should not impede technological inclusion. By investing in resources and research for languages such as Bengali, we take a crucial step toward ensuring that the benefits of cutting-edge NLP technologies are available to everyone, regardless of their native dialect. The study of Bengali word embeddings bears enormous promise for both NLP and the Bengali-speaking community as a whole. Through this dissertation, we hope to not only cover an important research void but also contribute to a more inclusive and technologically empowered global society.

1.2 Literature Review

Word clustering is an essential aspect of working with large datasets in scientific research. As a result, a great deal of research has been conducted in an effort to identify suitable techniques for constructing word clustering in Bangla and other languages. Below, we will discuss a few of these works.

Previous word clustering techniques relied heavily on the N-gram model for cluster construction. This is evident in the works of Ismail and Rahman [1], who proposed an N-gram Language Model-based Bangla word clustering method. In this study, the authors attempted to group Bengali words based on their semantic and contextual similarity. In this method, they attempted to cluster words based on the premise that words with similar meanings and usage in similar contexts belong to the same cluster.

Their work was subsequently improved by Urmi, Jammy, and Ismail [2] by a small amount. They proposed an unsupervised learning method for determining the stem or root of a Bangla word based on contextual word similarity. Their goal was to compile a large database of Bangla stems and their respective inflectional forms. They worked under the assumption that if two words are similar in orthography and are frequently used in similar contexts, there is a greater likelihood that they share the same root. They implemented a 6-gram model for stem detection and attained 40.18 percent accuracy. They concluded that with a large quantity of text data, this model will progress.

The performance of dynamic word clustering was the next area of research. The works of Yuan [3] shed light on this topic by demonstrating that a word clustering technique based on word similarities is superior to the conventional greedy approach regarding speed and performance. The fundamental process of this study was to verify for a specific word in the corpus, its co-occurring words for similarity. In other words, if two words are similar, so will their co-occurrence word pattern. Based on this, word clusters were computed, and when compared to other clustering methods, it was determined that this approach was more effective.

Ahmed and Amin [4] demonstrated the effectiveness of dynamic models for generating Bangla word clusters. The impact of the Bangla word embedding model on the classification of documents was discussed. They worked with a dataset constructed from Bangla newspaper articles. They used the word2vec model to generate vector representations of words for the purpose of word clustering. Using this information, they clustered word embeddings that are found in close proximity to one another in feature space. This information was subsequently incorporated as features into a classification solution for Bangla documents.

Mikolov et al. [5] proposed Word2Vec for the English language to learn high-quality distributed vector representations and demonstrated its utility for investigating the semantic relationship between words by measuring syntactic and semantic word similarity. It is available in both the Continuous Bag-of-Words (CBOW) and the Skip-Gram variants. These models are considered to be at the forefront of word embedding. Word2vec models are currently state-of-the-art for language data and are extremely valuable for text classification.

In this paper[6], the effectiveness of three word embedding models: word2vec in Tensorflow, word2vec from the Gensim package, and the FastText model were compared. They analyzed the consequences of each model using the same dataset. These three models are applied to a Bangla dataset containing 5,21,391 unique words to generate clusters, and their accuracy and efficiency are evaluated.

Sakhawat Hosain Sumit et al. [7] have put three alternative models into practice, two of which make use of Word2vec models and one of which uses the conventional Word to Index base text classifier model. They employed the Deep Long Short-Term Memory (LSTM) network to distinguish emotion using the vector representation of words created from words produced by skip-gram and CBOW.The Word2vec Skip-Gram model outperformed other models with an accuracy of 83.79 percent.

Authors [8] suggested a particular weighted least squares model that trains on worldwide word-word co-occurrence counts and effectively utilizes statistics. As shown by its cutting-edge performance of 75% accuracy on the word analogies dataset, the model generated a word vector space with significant substructure. In addition, they have demonstrated that their methods outperform existing methods on a number of word similarity tasks and a common named entity recognition (NER) benchmark.

1.3 Thesis Objective

In this thesis, an intensive exploration into the intricate domain of Bengali word embeddings is conducted, focusing on the advancement and appraisal of a newly devised model, utilizing the Gensim-implemented FastText algorithm. The endeavor is centered on enhancing natural language processing tasks for the Bengali language, which is rich

in complex morphology and syntax but has received limited exploration in the field of word embeddings. The quintessential goals of this thesis revolve around the development, training, fine-tuning, and evaluation of models designed to analyze Bengali word embeddings, specifically aiming at the application of advanced machine learning, deep learning, and transformer models for Bengali fake news detection.

The objectives of this research are as follows:

i. Selection of the BanglaLM dataset with specific research intents.
ii. Formulation of research questions aimed at guiding the analysis of the results.
iii. Execution of meticulous preprocessing steps on the dataset including the removal of stop words and punctuations.
iv. Acquisition of proficiency in Python and foundational knowledge in machine learning, deep learning, and transformers.
v. Determination of suitable machine learning, deep learning, and transformer models for the examination of Bangla word embeddings.
vi. Implementation of rigorous model training and hyperparameter fine-tuning processes.
vii. Comprehensive evaluation and analysis of the achieved results to draw conclusive insights.
viii. Addressing the formulated research questions based on the analysis.
ix. Articulation and presentation of the research findings in a coherent and structured manner.

In subsequent chapters, detailed insights are provided regarding the selected dataset used to train the embedding model, as well as a thorough explanation of the methodologies employed. The third chapter is devoted to a comprehensive analysis of the results acquired from the executed models, establishing connections between various aspects of the outcomes. In the concluding fourth chapter, reflections are offered on the extensive body of work conducted, summarizing the key findings and discussing potential avenues for future improvements and refinements to the study. This structure is intended to lead the reader through the complex layers of the research, providing an integrated overview of the efforts and discoveries contained within.

References:

[1] S. Ismail and M. S. Rahman, "Bangla word clustering based on n-gram language model," in *2014 International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, IEEE, 2014, pp. 1-5.

[2] T. T. Urmi, J. J. Jammy, and S. Ismail, "A corpus based unsupervised Bangla word stemming using n-gram language model," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, IEEE, 2016, pp. 824-828.

[3] L. Yuan, "Word clustering algorithms based on word similarity," in *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1, IEEE, 2015, pp. 21-24.

[4] A. Ahmad and M. R. Amin, "Bengali word embeddings and its application in solving document classification problem," in *2016 19th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2016, pp. 425-430.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv, 2013, arXiv:1301.3781.

[6] Z. S. Ritu, N. Nowshin, M. M. H. Nahid, and S. Ismail, "Performance Analysis of Different Word Embedding Models on Bangla Language," in *Proceedings of the International Conference on Bangla Speech and Language Processing (ICBSLP)*, 21-22 September, 2018.

[7] S. H. Sumit, M. Z. Hossan, T. A. Muntasir, and T. Sourov, "Exploring Word Embedding For Bangla Sentiment Analysis," in *International Conference on Bangla Speech and Language Processing (ICBSLP)*, 21-22 September, 2018.

[8] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics: Doha, Qatar, 2014, pp. 1532-1543.

Chapter 2

Methodology

In this chapter Data Preprocessing, Tokenization, Word Embedding Model Training, Model Evaluation, Implementation, and all other relevant topics learnt throughout the process is discussed. Overall methodology of this thesis work can be summarized in the following figure 2.1.

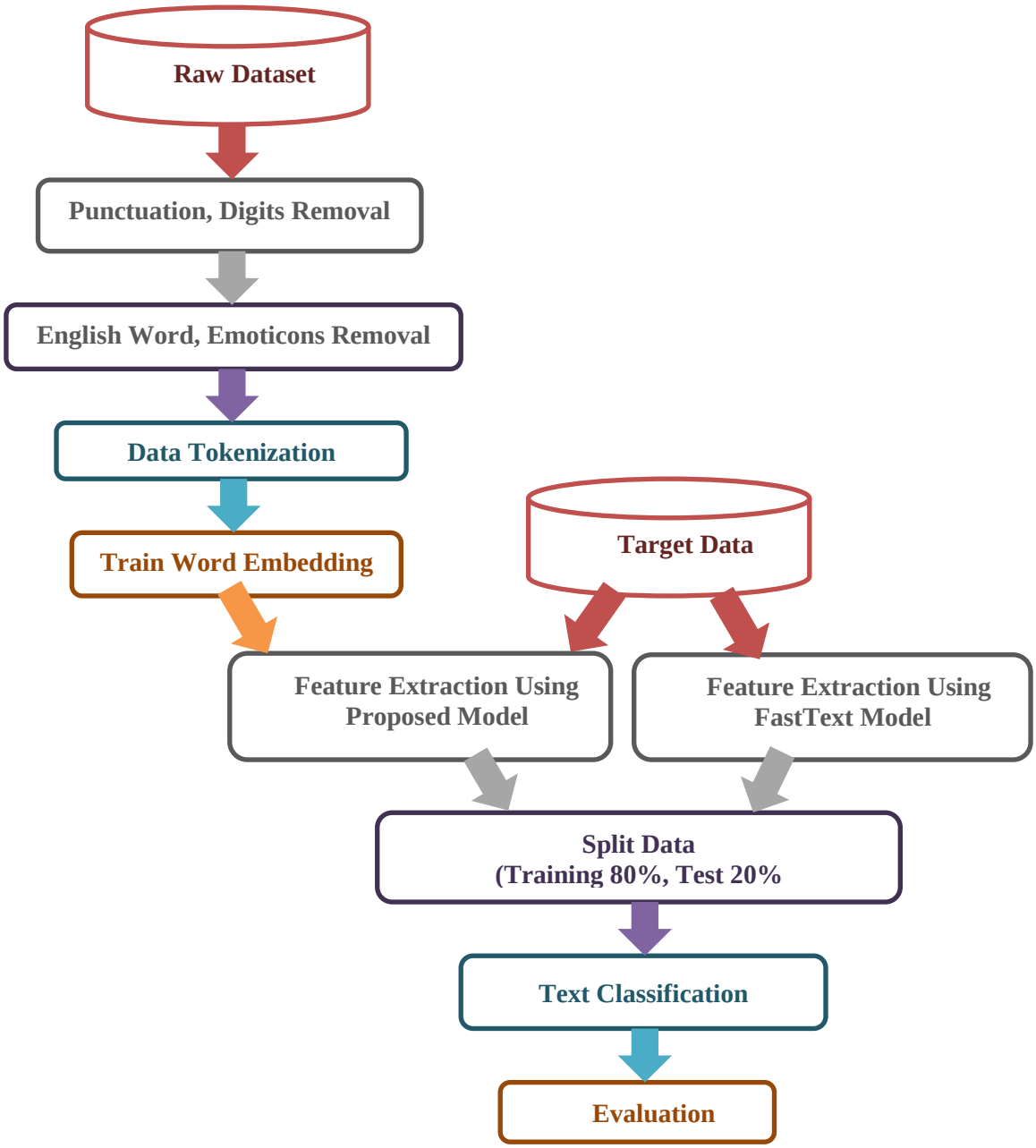

Figure 2.1: Schematic view of the Methodology.

2.1 Dataset Selection and Preprocessing:

2.1.1 Dataset Description

BanglaLM Dataset: A comprehensive collection of Bangla sentences.

Volume: 14 GB

Subset Used: 30,000,000 sentences

Dataset used for training proposed word embedding model was developed by the authors [9] known as BanglaLM dataset. The corpus is compiled from social media platforms, blogs, newspapers, Wiki pages, and other similar resources. This dataset contains 1,913,2010 samples with lengths ranging from 3 to 512 words. Approximately 20 million observations comprise the BanglaLM dataset. It is available in three versions: unprocessed data, preprocessed V1, and preprocessed V2. Preprocessed V1 is optimal for LSTM-based machine learning models, while V2 is optimal for statistical models.

In this study, the dataset's raw form is used, and it is preprocessed using a new instruction. This dataset comprises 1,710,431 distinct words and 821,007,301 words in total. In Figure 1.1 and Table 1.1 snapshot and summary of the BanglaLM dataset is provided.

Table 2.1: Sample of the BanglaLM dataset

	text
0	মো. নজিবুর রহমান বলেন, 'ভ্যাট দিবস ও সপ্তাহ উদযাপনের উদ্দেশ্য হলো সুশাসন ও উন্নততর ব্যবস্থাপনা পদ্ধতি অনুসরণ করে ভ্যাট প্রদানে জনগণকে উদ্বুদ্ধ ও সচেতন করার পাশাপাশি ভ্যাটকে জনগণের কাছে সহজবোধ্য করা
1	এর মাধ্যমে ভ্যাট কর্মকর্তাদের সঙ্গে করদাতাদের সুসম্পর্ক স্থাপন ও রাজস্ব আহরণ বাড়বে বলে আশা করি
2	বিমান বাংলাদেশ এয়ারলাইন্সের (বিজি-১০১১) প্রথম হজ ফ্লাইটটি ৪১৯ জন হজযাত্রী নিয়ে ঢাকা ছেড়েছে
3	আজ শনিবার সকাল ৭টা ৫৫ মিনিটে হজ ফ্লাইটটি ঢাকা থেকে সৌদি আরবের জেদ্দার কিং আব্দুল আজিজ আন্তর্জাতিক বিমানবন্দরের উদ্দেশে যাত্রা শুরু করে
4	পরে প্রধান অতিথি ও অতিথিবৃন্দ বিভিন্ন বিষয়ে সফলাতা অর্জনের কলেজের শিক্ষার্থীদের হাতে পুরস্কার তুলে দেন
5	স্বল্পমেয়াদি এবং দীর্ঘমেয়াদি উভয় ক্ষেত্রে বিষণ্নতা উপসর্গ বিলোপের ক্ষেত্রে শারীরিক ব্যায়াম একটি কার্যকর পদ্ধতি
6	প্রয়াত রূপান্তরকামী বন্ধুদের শ্রদ্ধা জানাতে স্মরণসভা
7	আয়োজনে বিডিএস- সমভাবনা নিজস্ব প্রতিনিধিঃ রূপান্তরকামীরা আজও আলাদা

8	সঠিক অস্তিত্ব নিয়ে বেঁচে থাকার অধিকার খুন্ন হয় এদের , শারীরিক... নড়াইল প্রতিনিধি : গণকবর, স্মৃতিস্তম্ভ ও বধ্যভূমিতে ফুলেল শ্রদ্ধা নিবেদন, র‍্যালি, আলোচনা সভা ও সাংস্কৃতিক অনুষ্ঠানের মধ্য দিয়ে নড়াইল মুক্ত দিবস পালিত হয়েছে

Table 2.2: Summary of the BanglaLM dataset

Total Samples	1,913,2010
Total Words	821,007,301
Unique Words	1,710,431

2.1.2 Preprocessing

Data contains many words, symbols, numeric values, and English terms that contain very little or no significant meaning for Bengali Word Embedding. In contrast, adjectives are the most important keywords that contribute to meaningful linguistic information. Therefore, eradicating superfluous terms and emphasizing the most important keywords improves the overall performance of the models. Regular expressions (regex) are used to remove emoticons, symbols and pictographs, transport and map symbols, flags (ios), Latin, general punctuations, emoji, English patterns, and punctuation after data collection. Preprocessing has a significant impact on the overall efficacy of models. In the context of text analysis, several crucial preprocessing steps are imperative for effective machine learning implementations.

i. Text Sanitization:

Eliminating special characters, punctuation, and other non-alphanumeric elements is a critical step in assuring that tokenization proceeds without errors. Additionally, converting all textual elements to lowercase ensures case insensitivity, which can be valuable for downstream analysis.

ii. Omission of Stopwords:

It is advisable to discard common stopwords, such as "এই," "যে," "করে," in Bengali, which often lack substantial semantic content, to facilitate meaningful natural language processing.

iii. Standardization of Numerical Entities:

Quantitative information, including numbers and dates, should be converted into a uniform format. For instance, numerical values could be universally represented by a token such as "<NUM>" to maintain consistency.

iv. Management of Out-of-Vocabulary Elements:

Bengali language features a diverse morphology, encompassing multiple verb conjugations and noun forms. The usage of subword-level embeddings, like FastText, offers an effective way to cope with out-of-vocabulary words.

v. Data Structuring:

The dataset should be systematized into either sentences or paragraphs, in accordance with the analytical objectives. Each sentence or paragraph ought to consist of a sequence of words for easier manipulation.

vi. Partitioning of Dataset:

For robust machine learning models, it is recommended to divide the dataset into training, validation, and testing subsets. The training set is vital for the initial model training, the validation set for hyperparameter tuning, and the test set serves to gauge the final model performance.

vii. Utilization of Specialized Libraries:

For tasks like text sanitization and tokenization, specialized Python libraries such as Regular Expression (re) and Natural Language Toolkit (nltk) are commonly recommended.

Adhering to these guidelines can significantly augment the efficacy and accuracy of machine learning models designed for textual analysis.

The selection of preprocessing procedures could differ based on both the characteristics of the dataset and the particular needs of the word embedding model in use. Moreover, should one possess resources related to Natural Language Processing (NLP) specifically tailored for the Bengali language, these specialized utilities can offer enhanced capabilities in terms of advanced preprocessing and tokenization tasks. Table 2.1 delineates specific attributes and actions related to the preprocessing of a dataset before it is input for training.

Table 2.3: The preprocessing steps and property of the data-set before fitting into FastText model

Attribute	Action
Total Word	459,868,486
Unique words	1,710,431
Total char length	2,640,132,075
Noise	No

Emoticon	No
URL tag	No
HTML tag	No
Punctuation	No
Stop words	No
Stemming	No
Lemmatization	No

2.2 Tokenization:

Tokenization is the process by which texts are broken down into smaller pieces. These smaller pieces are known as tokens. Tokenization can be done by using python's split function, regular expression, NLTK library, spaCy library, keras framework, Gensim library. I used a custom declared function for tokenization. One hundred instances of the data sample was taken in the following figure 2.2 and after applying the tokenization the result was also given.

Table 2.4: Example of Tokenized Data.

	text
0	['মো', 'নজিবুর', 'রহমান', 'ভ্যাট', 'দিবস', 'সপ্তাহ', 'উদযাপনের', 'উদ্দেশ্য', 'সুশাসন', 'উন্নততর', 'ব্যবস্থাপনা', 'পদ্ধতি', 'ভ্যাট', 'প্রদানে', 'জনগণকে', 'উদ্বুদ্ধ', 'সচেতন', 'পাশাপাশি', 'ভ্যাটকে', 'জনগণের', 'সহজবোধ্য']
1	['ভ্যাট', 'কর্মকর্তাদের', 'করদাতাদের', 'সুসম্পর্ক', 'স্থাপন', 'রাজস্ব', 'আহরণ', 'বাড়বে', 'আশা']
2	['বিমান', 'বাংলাদেশ', 'এয়ারলাইন্সের', 'বিজি', 'হজ', 'ফ্লাইটটি', 'হজযাত্রী', 'ঢাকা', 'ছেড়েছে']
3	['শনিবার', 'সকাল', 'মিনিটে', 'হজ', 'ফ্লাইটটি', 'ঢাকা', 'সৌদি', 'আরবের', 'জেদ্দার', 'কিং', 'আব্দুল', 'আজিজ', 'আন্তর্জাতিক', 'বিমানবন্দরের', 'উদ্দেশে', 'যাত্রা']
4	['প্রধান', 'অতিথি', 'অতিথিবৃন্দ', 'বিষয়ে', 'সফলাতা', 'অর্জনের', 'কলেজের', 'শিক্ষার্থীদের', 'হাতে', 'পুরস্কার']
5	['স্বল্পমেয়াদি', 'দীর্ঘমেয়াদি', 'উভয়', 'বিষণ্নতা', 'উপসর্গ', 'বিলোপের', 'শারীরিক', 'ব্যায়াম', 'কার্যকর', 'পদ্ধতি']
6	['প্রয়াত', 'রূপান্তরকামী', 'বন্ধুদের', 'শ্রদ্ধা', 'জানাতে', 'স্মরণসভা']
7	['আয়োজনে', 'বিডিএস', 'সমভাবনা', 'নিজস্ব', 'প্রতিনিধিঃ', 'রূপান্তরকামীরা', 'আজও', 'আলাদা']
8	['সঠিক', 'অস্তিত্ব', 'বেঁচে', 'থাকার', 'অধিকার', 'খুল্ল', 'শারীরিক', 'নড়াইল', 'প্রতিনিধি', 'গণকবর', 'স্মৃতিস্তম্ভ', 'বধ্যভূমিতে', 'ফুলেল', 'শ্রদ্ধা', 'নিবেদন', 'র্যালি', 'আলোচনা', 'সভা', 'সাংস্কৃতিক', 'অনুষ্ঠানের', 'মধ্য', 'নড়াইল', 'মুক্ত', 'দিবস', 'পালিত']
9	['জেলা', 'মুক্তিযোদ্ধা', 'কমান্ড', 'জেলা', 'প্রশাসনের', 'আয়োজনে', 'বৃহস্পতিবার', 'প্রথমবার', 'অনলাইনে', 'আবেদন', 'কথাটি', 'যায়', 'বোর্ডে', 'কলেজ', 'পরিদর্শক', 'বরাবরে', 'প্রবেশপত্রের', 'কপি', 'মোবাইল', 'নম্বর', 'জুন', 'তারিখের', 'সমাধান', 'আবেদন']

2.3 Model Training:

The tokenized data is then used to train for Bengali Word Embedding. Word embedding is a technique that transforms words or tokens into dense vector representations in a continuous vector space, where words with similar meanings or contexts are closer to each other in the vector space. This approach encodes the semantic and syntactic information of words in a way that is computationally efficient and preserves linguistic relationships.

Word embeddings have gained widespread popularity in NLP due to their ability to address the sparsity and high-dimensionality issues associated with traditional one-hot encoding of words. They provide continuous, dense representations that capture the meaning and context of words, making them valuable for various NLP applications, including sentiment analysis, machine translation, and named entity recognition.

Various models for word embedding have been devised, each with its own approach to encoding semantic and syntactic relationships between words. Two primary architectures underlie the Word2Vec model for generating word embeddings: Continuous Bag-of-Words (CBOW) and Skip-gram. Both architectures were introduced by Mikolov et al. in a seminal paper that marked a pivotal shift in Natural Language Processing (NLP) [10].

2.3.1 Continuous Bag of Words (CBOW)

The Continuous Bag-of-Words (CBOW) architecture is one of the foundational models in the realm of word embeddings. This model focuses on predicting a target word based on its surrounding context. This section aims to present a comprehensive overview of the CBOW model by exploring its mathematical foundations, advantages, and disadvantages.

In CBOW, the task is to predict a target word w_t from its surrounding context words $C = \left\{w_{t_{-c}}, \dots, w_{t_{-1}}, w_{t_{+1}}, \dots, w_{t_{+c}}\right\}$.

The objective function seeks to maximize the average log probability of the target word given its context. Formally, it is given as

$$L = \sum_{1}^{T} logP(\, wt \mid C\,) \;\dots\dots\dots\dots\dots\dots (2.1)$$

where T is the total number of target words. The conditional probability $P(w_t|C)$

$$P(w_t|C) = \frac{exp(v_{w_t} . v_C)}{\sum_{i=1}^{V} exp(v_i . v_C)} \dots\dots\dots\dots\dots\dots (2.2)$$

Here, v_C is the average vector of the context words, obtained by

$v_C = \frac{1}{2c}\sum_{-c \le J \le c, j \neq 0} v_{w_{t+j}}$ and V is the size of the vocabulary.

Advantages

- Computational Efficiency: CBOW tends to be computationally more efficient compared to its counterpart, the Skip-gram model, particularly when dealing with large vocabularies.
- Smoothing Effect: The CBOW model inherently smooths over the context by averaging the embeddings of the context words. This can be advantageous when the model encounters noisy or conflicting contexts.
- Generalization: Due to the averaging operation, CBOW can generalize well to different syntactic structures, making it useful for a variety of NLP tasks.

Disadvantages

- Loss of Order Information: CBOW suffers from the loss of word order information due to its averaging operation, which can be critical for understanding the semantics of a sentence.
- Weak Performance for Rare Words: The architecture is less accurate in capturing the semantics of infrequent or rare words as it leans toward representing common words more effectively.
- Homonym Limitation: Words with multiple meanings can be inadequately represented as CBOW averages over all contexts.

The CBOW architecture serves as a robust and computationally efficient model for generating word embeddings but comes with its set of limitations, including the loss of syntactic intricacies and less effective representation of rare words.

2.3.2 Skip-gram

The Skip-gram architecture is one of the two primary techniques in the Word2Vec suite for generating word embeddings, the other being the Continuous Bag-of-Words (CBOW). The Skip-gram technique focuses on the reverse objective of CBOW: it aims to predict the context words based on a given target word.

Contrary to CBOW, which aims to predict a target word from its context, Skip-gram's objective is to predict the context C from a given target word w_t. The objective function is defined as:

$$L = \sum_{t=1}^{T} logP(w_t \,|w_{t-c}\ , \dots, w_{t+c}) \dots\dots\dots\dots\dots\dots (2.3)$$

$$\tau = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c \le j \le c, j \ne 0} logP\left(w_{t+j} \middle| w_t\right) \dots\dots\dots\dots\dots\dots (2.4)$$

where T is the total number of words in the corpus, c is the size of the context window, and w_{t+j} represents context words.

The conditional probability $P\left(w_{t+j} \mid w_t\right)$ is computed as:

$$p\left(w_{t+j} \,|w_t\right) = \frac{\exp(v'_{w_{t+j}} . v_{w_t})}{\sum_{i=1}^{V} \exp(v'_i \,.\ v)} \dots\dots\dots\dots\dots\dots (2.5)$$

Here, $v'_{w_{t+j}}$ is the 'output' vector corresponding to the context word and v_{w_t} is the 'input' vector for the target word. V denotes the vocabulary size.

Advantages

- Quality of Embeddings: Skip-gram provides high-quality embeddings, especially for infrequent or rare words, thereby capturing subtle semantic relations.
- Flexible Context Prediction: Unlike CBOW, Skip-gram is capable of predicting multiple words given a single context word, offering a more dynamic representation of word relations.
- Task Versatility: Due to its capability to capture deeper contextual information, Skip-gram embeddings are often preferred for more complex NLP tasks like named entity recognition and co-reference resolution.

Disadvantages

- Computational Intensity: Skip-gram is generally slower and requires more computational resources compared to CBOW, particularly for large vocabularies.
- Sensitivity to Noise: Because Skip-gram treats each context-target pair separately, it can be more sensitive to the noise in the data, especially if not preprocessed or filtered adequately.
- Overfitting Risks: Due to its complexity and fine-grained prediction capabilities, Skip-gram models are more prone to overfitting, especially on smaller datasets.

The Skip-gram model excels in capturing semantic relationships and generating high-quality embeddings, particularly for infrequent words. However, its computational requirements and susceptibility to noise are significant considerations. In this study, skip-gram is used to generate Bengali word embeddings because it functions well with small amounts of training data and accurately represents even uncommon words and phrases[10].

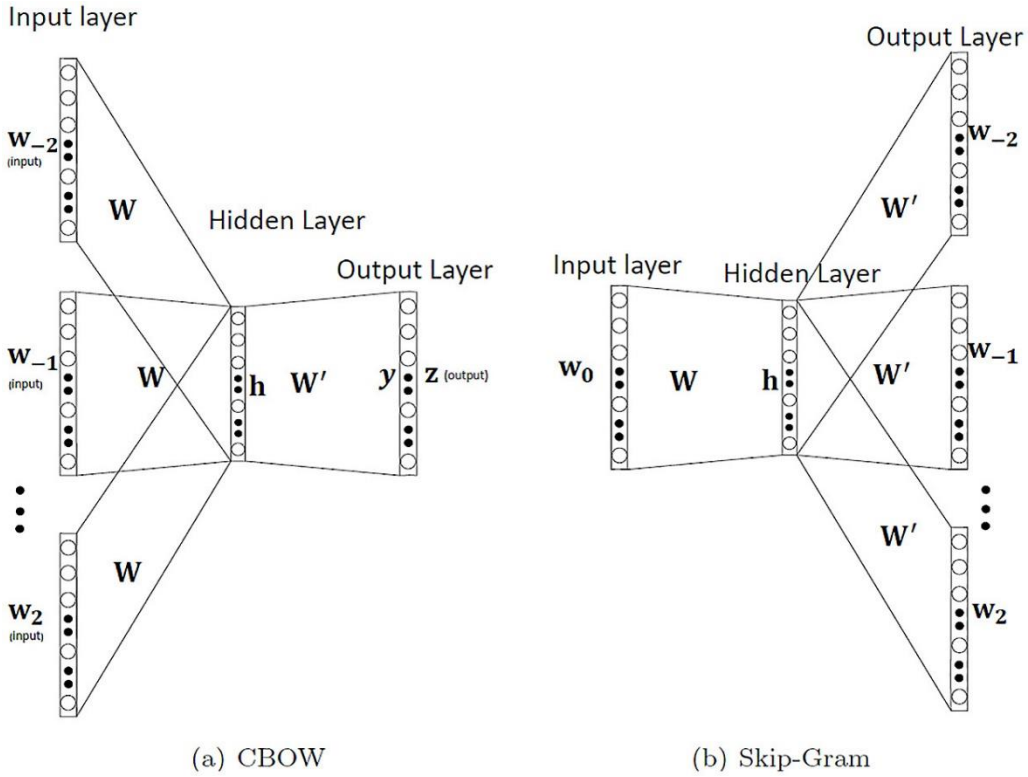

Figure 2.2: Visual Representation of CBOW and Skip-gram[11]

2.3.3 Word2Vec

Word2Vec, introduced by Mikolov et al., revolutionized the field of Natural Language Processing (NLP) by providing a method to map words into a continuous vector space [10]. This section aims to delve into the mathematical and theoretical aspects of the Word2Vec model.

The model operates under the core idea that words appearing in similar contexts tend to share semantic or syntactic meaning. It attempts to maximize the probability $P(w_0|w_I)$

where w_0 is an output (context) word and w_I is an input word. The conditional probability $P(w_0|w_I)$ is defined as:

$$p(w_0|w_I) = \frac{\exp(v'_{w0}{}^{\intercal} v_{wI})}{\sum_{j=1}^{V} \exp(v'_j \,.\, v_{wI})} \dots\dots\dots\dots\dots\dots\dots (2.6)$$

Here, v_{w_I} and v'_{w_0} are the "input" and "output" vectors of the respective words. The V in the denominator represents the vocabulary size, summing over all possible words.

The objective of the Word2Vec model is to maximize the average log probability of observing context words $w0$ given the center words w_I:

$$\frac{i}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logP(w_{t+j}|w_t) \dots\dots\dots\dots\dots\dots\dots (2.7)$$

This function is optimized using stochastic gradient descent (SGD)[10].

Advantages

- Semantic and Syntactic Relationships: Word2Vec embeddings capture intricate relationships, be they semantic ("king" and "queen") or syntactic ("run" and "running").
- Scalability: The model scales well to larger datasets and vocabularies, making it apt for big data applications.
- Dense Representation: Unlike sparse, one-hot encoded vectors, Word2Vec embeddings are dense, capturing more information in fewer dimensions.

Disadvantages

- Out-of-Vocabulary Words: The model cannot handle words not seen during training, rendering it non-adaptable to evolving language without retraining.
- Context Limitation: Word2Vec does not account for the order or arrangement of words, missing sentence-level contextual information.
- Parameter Tuning: The quality of the embeddings can be highly sensitive to the choice of parameters such as vector size, window size, and learning rate, requiring extensive tuning.

The Word2Vec model serves as a cornerstone in the domain of NLP for generating word embeddings. Its advantages make it an excellent choice for many applications, while its limitations guide the trajectory of ongoing research in the field.

2.3.4 GloVe (Global Vectors for Word Representation)

The GloVe (Global Vectors for Word Representation) model, introduced by Pennington et al. in 2014 [11], is an unsupervised learning algorithm designed to obtain word vectors from large-scale corpora. Unlike previous methods like Word2Vec, GloVe aims to directly optimize the vector space to capture the global corpus statistics. This section provides an overview of the GloVe model, exploring its mathematical framework as well as its advantages and disadvantages.

The GloVe model's objective is to encode semantic meaning in a way that reflects the global statistics of word co-occurrence frequencies in a corpus. The fundamental idea behind GloVe is to construct a word-word co-occurrence matrix X. where X_{ij} represents how often word j occurs in the context of word i.

The primary objective function for GloVe is:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T w_j' + b_i + b_j' - \log X_{ij})^2 \dots\dots\dots\dots\dots\dots (2.8)$$

Here, f is a weighting function, w_i and w_j' are the word vectors for words i and j, respectively. V is the vocabulary size. The weighting function f(x) is defined as:

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & if\ x < x_{\max} \\ 1 & otherwise \end{cases} \dots\dots\dots\dots\dots\dots (2.9)$$

where x_{max} and α are hyperparameters[12].

Advantages

- Global Context: GloVe captures global statistical information of the corpus, which makes it proficient in capturing global semantic relationships between words.
- Scalability: The model scales well to larger corpora and high-dimensional word vectors, making it ideal for applications requiring extensive vocabularies.
- Interpretable Model: The GloVe model provides a more interpretable vector space compared to other word embedding models like Word2Vec.

Disadvantages

- Memory Usage: Construction of the co-occurrence matrix X can be memory-intensive, especially for large vocabularies and long text corpora.
- Static Embeddings: Like other traditional word embedding methods, GloVe produces static embeddings, meaning each word has a single representation irrespective of context, which can be limiting.
- Hyperparameter Sensitivity: The choice of hyperparameters like x_{max} and α can significantly affect the quality of the embeddings, requiring careful tuning.

GloVe serves as a powerful word embedding model, capturing both local and global semantics effectively. However, its computational overheads and the static nature of the generated embeddings are critical limitations.

2.3.5 WordRank

WordRank is a word embedding technique proposed by Piotr Bojanowski et al., as an extension to the earlier Word2Vec algorithm. While Word2Vec employs neural networks for learning embeddings, WordRank adopts a ranking approach for learning word representations. This section provides an overview of WordRank's mathematical formulation, along with its advantages and disadvantages.

The fundamental idea behind WordRank is to generate embeddings that are capable of ranking context words around a given target word effectively. The mathematical objective is generally to minimize a ranking loss which can be formally represented as follows[13]:

$$\mathcal{L} = \sum_{(w,c)\epsilon D} [\sum_{n\epsilon N(w,c)} \max(0,1 - s(w,c) + s(w,n))] \dots\dots\dots\dots\dots\dots (2.10)$$

Where:

- D is the dataset of (word, context) pairs.
- $N_{w,c}$ is the set of negative samples for a given (word, context) pair (w,c).
- s(w,c) is the scoring function, typically $s(w,c) = w.c$, where w and c are the embeddings for the word and context, respectively.

Advantages

- Simplicity: WordRank offers a more straightforward optimization problem compared to the neural network-based models, making it easier to understand and implement.
- Efficiency: The model is efficient in terms of computational requirements, particularly when dealing with very large datasets, thanks to the efficient negative sampling.
- Good Quality Embeddings: Empirical studies have shown that WordRank can produce high-quality word embeddings that perform well in a variety of NLP tasks [12].
- Scalability: The algorithm can be parallelized efficiently, making it feasible for large-scale data processing.

Disadvantages

- Hyperparameter Sensitivity: Similar to other embedding models, WordRank's performance can be sensitive to the choice of hyperparameters, like the learning rate and the negative sampling rate.

- Context Insensitivity: Like other word embedding techniques, WordRank creates a single static vector for each word, ignoring the different senses or meanings a word can have in different contexts.
- Dependency on Preprocessing: The quality of the embeddings can be heavily dependent on the text preprocessing steps, like tokenization and stemming.

WordRank is an efficient and effective word embedding model that adopts a ranking approach for learning word representations. While it provides high-quality embeddings and is computationally efficient, the static nature of the embeddings and sensitivity to hyperparameters are limitations to consider.

2.3.6 FastText

FastText is an unsupervised word representation learning algorithm developed by Facebook's AI Research (FAIR) lab. Introduced by Bojanowski et al. in 2017, the algorithm extends the original Word2Vec model by considering subword information during training [12]. This section provides an analysis of FastText, discussing its mathematical foundations, advantages, and disadvantages.

The core innovation of FastText lies in its use of subword information, which allows the model to generate embeddings for out-of-vocabulary words and capture morphological nuances. Given a word w, it can be represented by a bag of character n-grams. If w="apple" and n=3, the n-grams would include "<ap, app, ppl, ple, le>", along with the special sequence "<apple>" to represent the whole word.

The word vector **w** is computed as the sum of these n-gram vectors:

$$\boldsymbol{w} = \sum_{g \in G_w} \boldsymbol{g} \dots\dots\dots\dots\dots\dots (2.11)$$

where G_w is the set of n-grams for word w.

The objective function for FastText is similar to that of the Skip-gram model in Word2Vec, often represented as:

$$max \prod_{t=1}^{T} \prod_{-c \le j \le c, j \neq 0} P\left(w_{t+j} | w_t\right) \dots\dots\dots\dots\dots\dots (2.12)$$

This objective function represents the maximization of the likelihood of observing context words $Wt + j$ given the target word Wt within a context window, where T is the total number of words in the corpus, and c is the context window size.

FastText is grounded in the Distributional Hypothesis, which posits that words appearing in similar contexts share semantic meaning. By leveraging subword information, FastText enhances the representational power of word embeddings, particularly for languages with complex morphology. It excels in capturing both semantic and morphological relationships among words.

Advantages

- Morphological Richness: By considering subword information, FastText can capture the morphology of words, making it superior for morphologically rich languages.
- Out-of-Vocabulary Words: The subword information allows the algorithm to generate word vectors for out-of-vocabulary words, a feature missing in models like Word2Vec.
- Fast Training: Despite the added complexity, FastText is optimized for speed and can be trained relatively quickly even on large corpora.
- Interpretable Components: The subword embeddings offer an additional layer of interpretability, allowing one to understand what morphological aspects the model captures.

Disadvantages

- Increased Model Size: Because it has to store vectors for subwords, FastText models can become significantly larger than their Word2Vec counterparts.
- Computational Complexity: The inclusion of subwords increases the complexity of the model, requiring more computational resources for tasks like nearest neighbor searches.
- Context Insensitivity: Like other word embedding models, FastText provides a single representation for each word, neglecting the different senses that words can have in various contexts.

FastText offers a significant advancement over traditional word embedding techniques by incorporating subword information. Although computationally more intensive and storage-heavy, its advantages in capturing morphological details and generating out-of-vocabulary word embeddings are considerable. In the context of this research, the FastText model was meticulously implemented and trained utilizing the Python programming environment. For the specialized tasks associated with the model's architecture and training, the Gensim library served as the principal computational tool.

Parameter Settings

window: The window size was set to 3, specifying the maximum distance between the current and predicted word within a sentence during model training. This parameter is vital for capturing semantic and syntactic relationships among words.

min_count: Words that appear less than 5 times across the entire dataset were ignored, reducing the computational burden and focusing on more relevant words.

workers: To expedite the training process, 6 worker threads were utilized for parallel computation.

vector_size: The dimensionality of the word vectors was fixed at 300, balancing the trade-off between computational efficiency and model performance.

Vocabulary Building and Training Phases

Vocabulary Construction: Before model training, a vocabulary was built using the method build_vocab, taking the text corpus as input. The parameter progress_per is set to 1000, which indicates reporting the progress every 1000 words during vocabulary construction.

Epochs Configuration: The model was trained for 5 epochs to iteratively refine the word vectors. The number of epochs is set after vocabulary construction, enabling the model to go through the training data multiple times.

By using this configuration, the proposed FastText model aims to provide rich and expressive word vectors that encapsulate both the semantic and syntactic nuances of the words in the corpus.

2.3.7 ELMo (Embeddings from Language Models)

ELMo (Embeddings from Language Models) is a state-of-the-art word embedding technique that employs deep contextualized representations. Unlike traditional word-embedding methods like Word2Vec or GloVe, ELMo captures semantic nuances by analyzing words within their contextual meaning. This paper offers a comprehensive mathematical and theoretical analysis of ELMo, discussing its architecture, mechanics, advantages, and disadvantages.

ELMo, introduced by Peters et al. in their 2018 paper "Deep contextualized word representations" [13], represents an advancement in the natural language processing (NLP) field by providing deep, contextual word embeddings. The innovation lies in the model's ability to understand words in their semantic and syntactic context, rather than using a single static representation for each word.

The fundamental building block of ELMo is a bidirectional Long Short-Term Memory (Bi-LSTM) network that processes a sequence of T tokens $[x_1, x_2, \dots, x_T)$.

The hidden state at time t for the forward LSTM and backward LSTM are respectively

$$\overrightarrow{h_t} = LSTM_{forward}\left(x_t, \overrightarrow{h_{t-1}}\right) \dots\dots\dots\dots\dots\dots (2.13)$$

$$\overleftarrow{h_t} = LSTM_{backward}\left(x_t, \overleftarrow{h_{t+1}}\right) \dots\dots\dots\dots\dots\dots (2.14)$$

The final ELMo embedding for a token is generated by concatenating the forward and backward hidden states and then applying a weighted sum:

$$ELMo_t = \gamma \sum_{j=0}^{L} \alpha_j h_t^j \dots\dots\dots\dots\dots\dots (2.15)$$

Where γ is a scaling factor, L is the number of layers in the LSTM, α_j are the softmax-normalized weights.

Advantages

- Contextual Awareness: ELMo's bi-directional structure allows for an understanding of the word context, making it effective for polysemy.
- Transfer Learning: Pre-trained ELMo models can be fine-tuned for specific tasks, facilitating quicker and more accurate model development.
- Improved Performance: ELMo embeddings have been shown to enhance performance in various NLP tasks [14].

Disadvantages

- Computational Overheads: The deep nature of ELMo makes it computationally intensive, requiring substantial hardware resources for training.
- Model Complexity: Due to its multiple layers and complex architecture, ELMo models can be difficult to interpret and understand.
- Memory Consumption: Storing ELMo embeddings can be memory-intensive given the depth of the layers involved.

ELMo represents a significant shift from traditional word embedding techniques by introducing contextually aware representations. While its performance advantages are considerable, they come at the cost of computational complexity and resource requirements.

2.4 Machine Learning Models:

Before machine learning, all algorithms, programs, and computational tasks required inputs and a set of well-defined principles to be applied to the inputs. Some operations are performed based on well-defined principles, and the output is provided by algorithms or programs. However, the greatest difficulty lies in identifying well-defined principles. For a system with tens of thousands or millions of parameters, it is nearly impossible to determine the well-defined principles.

Machine learning has revolutionized the field of computer science by allowing machines to teach themselves using the data provided. Here, both input and output are provided to the model during training. During this training period, models independently discover the principles. Following this, only the inputs are provided, and the models generate the output based on the rules they learn during training. Analysis of extremely complex systems with a large number of parameters is facilitated by these Machine Learning approaches. The primary drawback of the Machine Learning strategy, however, is that the ML models require a vast quantity of data for improved performance. ML models can be broadly divided into three categories. Supervised, Unsupervised and Reinforcement are three primary categories of ML models. There are methods to determine the optimal model parameters for optimizing model performance. For example, the GridSearchCV method automatically attempts various parameter combinations and identifies the optimal combination for each model that provides the best classification performance.

2.4.1 Supervised Learning: During the training phase of supervised learning, the input and the desired output, also known as the objective, are provided to the model. Following training, models are able to accurately predict the target value from the provided inputs. Regression, Decision Tree, Random Forest, KNN, and Logistic Regression are supervised learning examples.

2.4.2 Unsupervised Learning: The majority of unsupervised learning is clustering. There is no output or desired value specified. Algorithms and models only estimate the number of clusters or groups represent in the input data by seeking out the concealed patterns among the data. Clustering, or unsupervised learning, is sometimes appropriate because the number of classes is not always known. Unsupervised learning is applicable in this situation. Unsupervised learning examples include K means clustering, Hierarchical clustering, and Apriori algorithm.

2.4.3 Reinforcement learning: Reinforcement learning is a subset of Machine Learning in which models acquire knowledge from their surroundings. Incorporating models into the environment. Observing the environment allows models to learn and make

predictions. Here, models are rewarded for every correct prediction, and penalized for every incorrect decision with a penalty score. Consequently, the correct decision-making is reinforced within the paradigm. Reinforcement learning is illustrated by Temporal Difference (TD) and Markov Decision Process.

2.5 Deep learning models:

Machine learning subsumes Deep learning. In deep learning, neurons represent the lines of processing. By utilizing deep learning models, the primary drawbacks of machine learning models are significantly mitigated. In machine learning models, if the models make an incorrect decision, there is no means for the model itself to rectify the error. In contrast, in deep learning, the error is mitigated by adjusting the weights using backward error propagation. Machine learning models learn from training data, parse data, and make predictions based on their training data learning. In contrast, deep learning models create a 'Artificial Neural Network' that replicates human neural networks. Deep learning models contain multiple levels of abstraction. Both machine learning and deep learning models fall under artificial intelligence. The relationship between AI, ML, and DL is depicted in Figure 2.4.

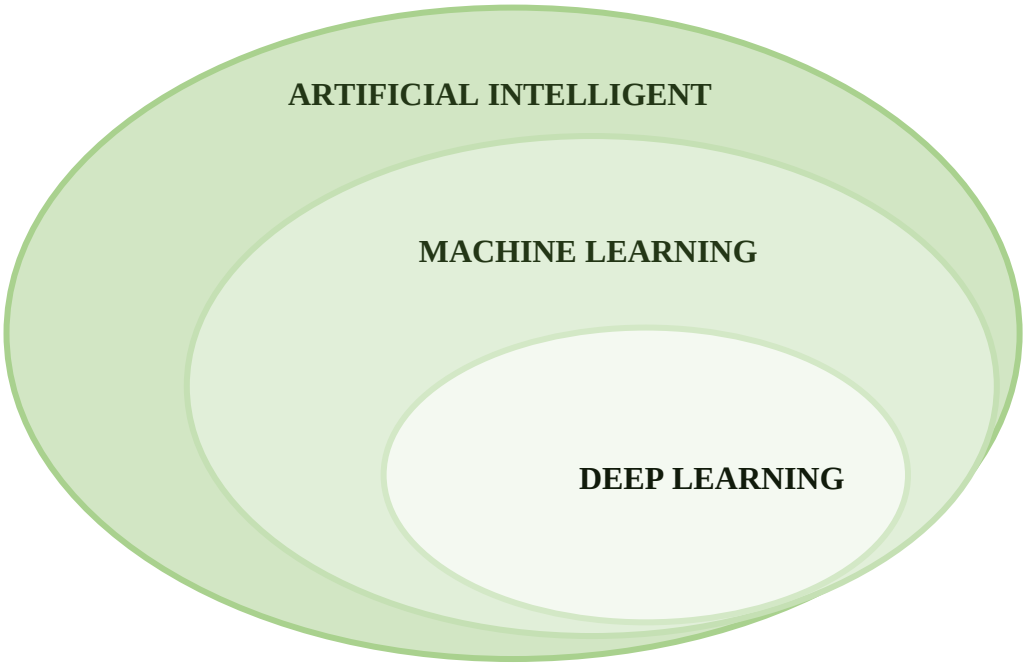

Figure 2.3: Relationship between Artificial Intelligence, Machine Learning, Deep learning

On vast datasets, deep learning generates artificial neural networks and performs complex computations. Artificial neurons, also known as nodes, are the fundamental processing elements of an artificial neural network, which has a structure akin to that of

the human brain. These nodes are piled atop one another to form strata. There are three kinds of layers in a neural network. There are three layers: input, concealed, and output. Each input layer node is furnished with input data. The nodes multiply the input by the weights, sum the results of the multiplication, and then add the bias value. Then, a nonlinear function, also known as activation functions, is employed to determine if the value exceeds the threshold value. If the value exceeds the threshold value, the neuron will become active. If the value is below the activation threshold, the neuron will not be activated. Figure 2.5 depicts a fundamental Artificial Neural Network.

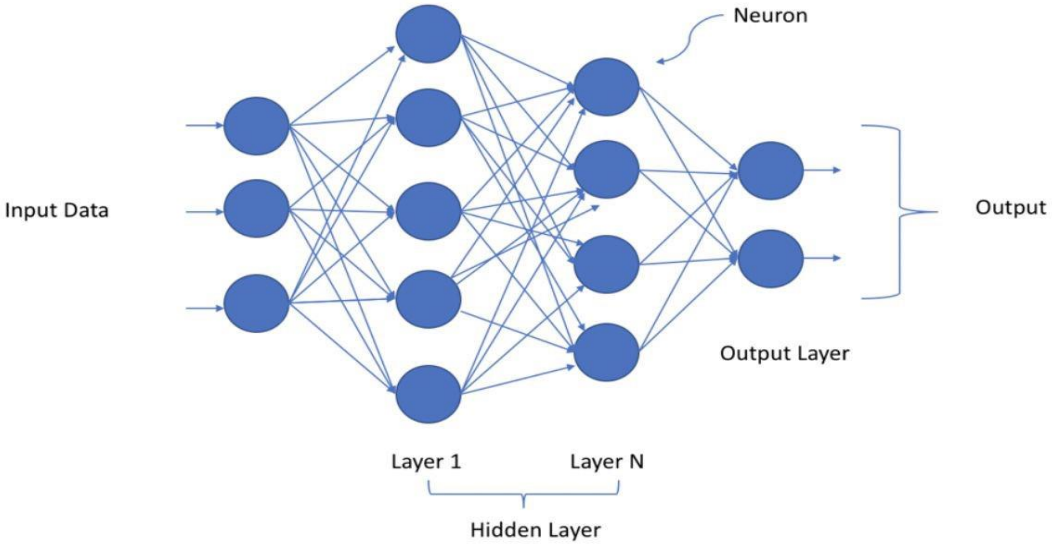

Figure 2.4: Artificial Neural Network [15]

Models of deep learning represent self-learning and rely on an artificial neural network that replicates the way the brain computes. During the training process, Deep Learning models extract features from input data, organize similar objects, and identify patterns. There are numerous Deep Learning models suited to specific duties. No network is regarded as flawless. I have utilized Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) to benchmark the effectiveness of proposed model.

2.5.1 Convolutional Neural Network:

Convolutional neural network, also known as CNN, is a subset of deep learning designed to process ordered arrays of data, such as representations. It is commonly used to analyze visual images because it can identify and classify image characteristics. CNN recognizes design elements in the input image, such as lines, gradients, circles, irises, and facial features. This characteristic makes convolutional neural networks extremely trustworthy. Convolutional neural networks (FC) consist of layers that are convolutional, polling, and completely connected. When we process an image, we apply filters, and each filter generates what we call a feature map. CNN utilizes a variety of filters to extract characteristics from

an image. The convolution and pooling layers are used to extract features, whereas the fully connected and dense layers are used to perform classification. In the convolution layer, filters are built. Convolution is a mathematical operation performed by sliding these filters across the input image to generate feature maps. The pooling layer reduces the map size of complex features. There are various varieties of pooling methods. In Max Pooling, the largest component from the feature map is chosen. The mean of the components within a predetermined size range. Average pooling is used to determine the image portion. The FC layer receives a flattened rendition of the input image from the levels that came before it. Numerous mathematical operations and categorizations are undertaken. Various activation functions, such as ReLU, tanH, sigmoid function, softmax, etc., are used to impart nonlinearity to networks. Figure 2.6 depicts the fundamental architecture of the CNN model.

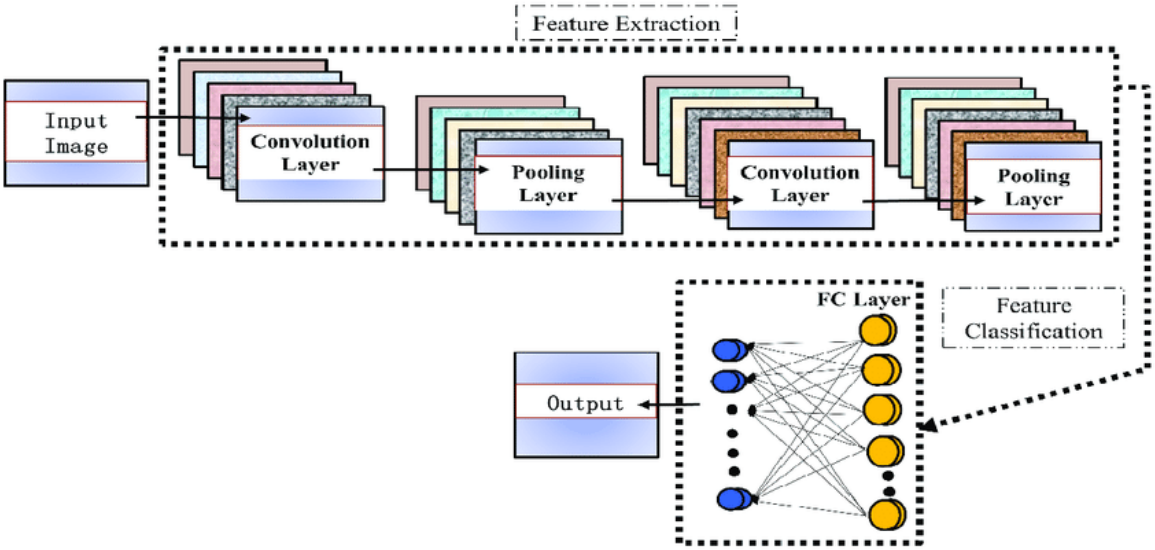

Figure 2.5: Basic Architecture of CNN model [16]

While Convolutional Neural Networks (CNNs) have been predominantly used in image recognition tasks, their application in Natural Language Processing (NLP) has proven to be equally promising, particularly when feature extraction is carefully designed. In this study, a specialized CNN model is implemented to detect fake news in the Bengali language.

Model Architecture

The architecture starts with an input embedding layer that receives textual data. The word embeddings from the proposed Bengali model are used to convert this textual information into a numerical format suitable for machine learning algorithms. This numeric data then serves as the input to the subsequent one-dimensional convolutional layer, designed for local feature extraction.

Following convolution, a Global Max Pooling layer is employed to reduce the spatial dimensions of the output. This is particularly useful for selecting the most important features from the convolution layer, as it picks the maximum value from each feature map.

To mitigate overfitting, especially considering the complexity of natural language data, a Dropout layer is added next. This layer randomly sets a fraction of the input units to 0, helping the model to generalize better.

Subsequently, the architecture features a fully connected Dense layer, followed by two alternating sets of Dropout and Dense layers. These are employed to further fine-tune the extracted features and make the model robust.

Finally, the output from the last Dense layer is flattened into a one-dimensional array. For the output layer, a fully connected Dense layer is utilized. The activation function employed depends on the classification task: a sigmoid function is used for binary classification, while a softmax function is employed for multi-class problems.

Implementation Details

The model is implemented using PyTorch and takes advantage of CUDA for hardware acceleration. For word embeddings, a custom-trained FastText model is used. The dataset is preprocessed, tokenized, and converted into FastText vectors with a maximum sequence length of 100. Data is then split into training and testing sets using an 80-20 ratio.

Various performance metrics, including F1-score, precision, and recall, are used for model evaluation. The Adam optimizer and Binary Cross-Entropy Loss function are employed for training the model. The learning rate is set to 1e-6, and the model is trained for 500 epochs.

Results

The model shows promising results in terms of validation accuracy, F1-score, precision, and recall. This suggests that the CNN architecture, along with the proposed Bengali word embeddings, provides a robust solution for fake news detection in the Bengali language.

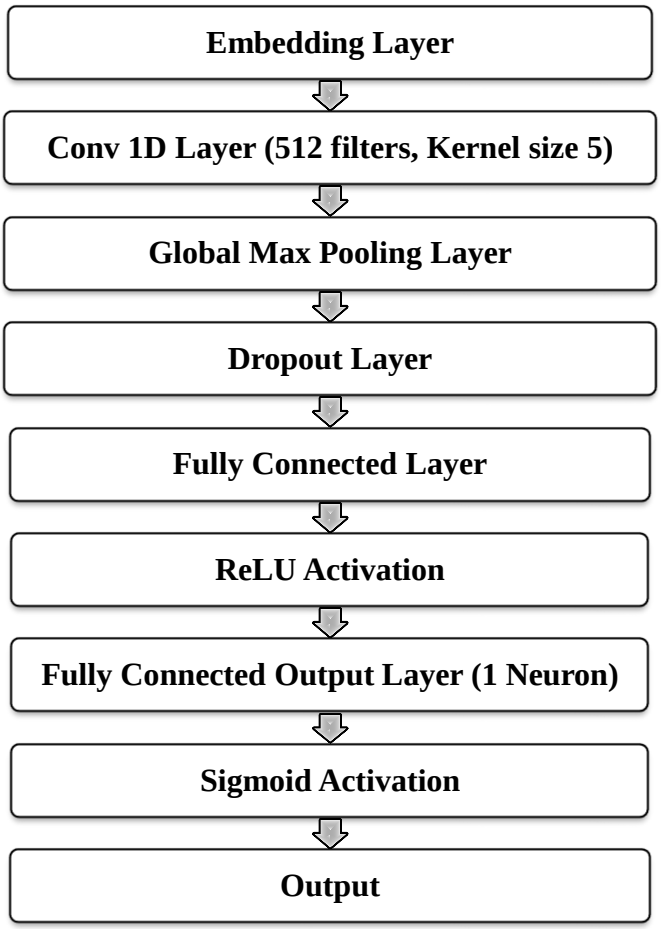

Figure 2.6: Implemented CNN model architecture

Table 2.5: Implemented CNN model parameters

Category	Parameter	Value / Description
Data Preprocessing	Maximum Sequence Length	100
	Embedding Dimension	Derived from custom FastText model
Dataset Split	Train-Test Ratio	80-20
Network Architecture	Input Layer	Embedding layer based on custom FastText model

Category	Parameter	Value / Description
	Conv1D Layer	Input: embedding_dim, Output: 512, Kernel: 5
	Global Max Pooling Layer	Pool Size: maxlen - 5 + 1
	Fully Connected Layer 1	Input: 512, Output: 128
	Fully Connected Output Layer	Input: 128, Output: 1
Activation Functions	ReLU	Used for fully connected layers
	Sigmoid	For binary classification
	Softmax	For multi-class classification (not shown in code)
Regularization	Dropout Layer	Used post max-pooling (Dropout rate not specified)
Optimization and Loss	Optimizer	Adam
	Learning Rate	$1\times10-61\times10-6$
	Loss Function	Binary Cross-Entropy Loss (BCELoss)
Training	Number of Epochs	500
	Batch Size	4

2.5.2 Recurrent Neural Network (RNN):

A type of Artificial Neural Network (ANN) utilized in speech recognition and natural language processing (NL) applications is the recurrent neural network. A RNN model is created to determine the sequential properties of data, using the identified patterns to predict future events. All inputs and outputs of traditional neural networks are interdependent. However, recurrent neural networks use the output of previous phases as the input for the current state. In order to anticipate the next letter of any word or the next word of any phrase, for example, it is necessary to retain and store the previous letters or words in some form of memory.

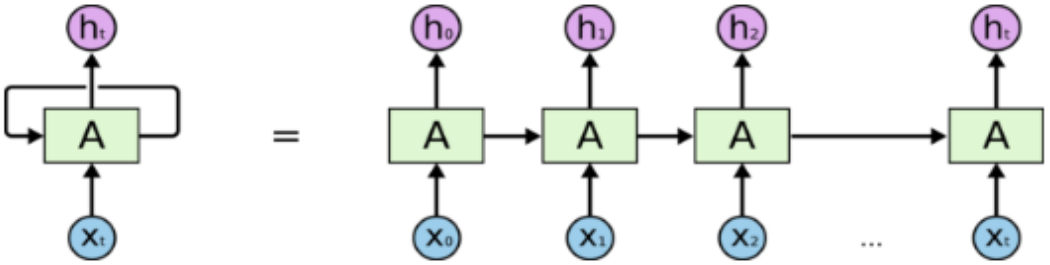

Figure 2.7: RNN Model [17]

The RNN models have an internal memory that continuously recalls the results of previous phases' calculations. The RNN applies the same parameter to each input and applies the same task to each input. Traditional neural networks are more complex than RNN because they have distinct inputs and outputs. Figure 2.8 depicts the fundamental architecture of the implemented RNN model.

Recurrent neural networks (RNNs) are the most advanced algorithm for sequential data. Due to its internal memory, this algorithm is the first to recall its input, making it ideal for machine learning problems involving sequential data. It is one of the algorithms that has contributed to deep learning's remarkable success in recent years. In this article, we will discuss the fundamental concepts underlying recurrent neural networks, as well as the primary issues they confront and how to address them.

Advantage:

- An RNN preserves all information throughout time, which is a benefit. It only helps with time series prediction if the user can remember previous inputs. This is known as long short-term memory.
- To boost the useful pixel neighborhood, convolutional layers and recurrent neural networks are even merged.

Disadvantage:

- Problems with explosions and gradient disappearance are a drawback.
- If tanh or relu are employed as the activation function, an RNN cannot handle very long sequences.
- It is extremely difficult to train.

2.5.3 Long Short-Term Memory (LSTM): Recurrent neural networks (RNN) are extended by LSTM networks, which were primarily developed to address RNN failure scenarios. Such "long-term dependencies" are completely beyond the capabilities of RNNs. Second, there is no finer control over how much of the past should be "lost" and how much of the context should be carried forward. Exploding and disappearing gradients, which happen when a network is being trained via backtracking, are another problem with RNNs. Long Short-Term Memory (LSTM) was introduced as a result. The LSTM hidden layer is a gated unit, which is the primary differentiator between LSTM and RNN systems. The output and state of the specific cell are decided by the interaction of its four levels. These two pieces are then transferred to the following buried layer. In contrast to RNNs, which only feature one logistic sigmoid gate, LSTMs also have a tanh layer. Gates have been developed to regulate the quantity of information traveling through the cell. They make judgements about what data should be ignored and what data the cell after them will need. The range for the product is 0 to 1, with 0 meaning "reject all" and 1 meaning "include all." In figure 2.9, the basic architecture of LSTM model is provided.

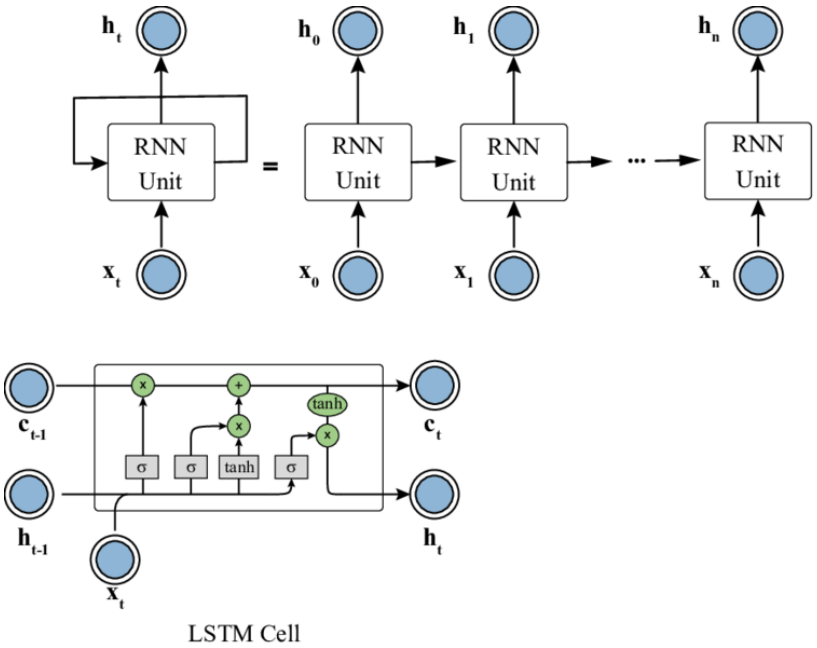

Figure 2.8: LSTM network [18]

The forget gate is the initial stage of the procedure. The new memory network and the input gate are involved in the next phase. This step's objective is to decide what new information, in light of the prior concealed state and the incoming input data, has to be added to the network's long-term memory (cell state). We may proceed to the output gate, the last stage, once we have finished updating the network's long-term memory.

Advantage:

- We may choose from a wide range of LSTM parameters, including learning rates and input and output biases.
- Thus, there is no need for precise modifications.
- With LSTMs, updating each weight is simpler than with Back Propagation Through Time (BPTT).
- It reduces complexity to O(1).

Disadvantage:

- Training LSTMs takes longer.
- To train LSTMs, additional RAM is needed.
- It is simple to overfit LSTMs.
- In LSTMs, dropout is far more difficult to implement.
- LSTMs respond differently to initializing random weights.

Bidirectional LSTM is equivalent to two LSTM running parallelly into opposite direction. One LSTM from forward to backward, another one from backward to forward. Bidirectional LSTM capture more information than unidirectional LSTM. In my research for Bengali fake news detection, I utilized the power of Recurrent Neural Networks (RNNs), specifically employing the Long Short-Term Memory (LSTM) model alongside the Convolutional Neural Network (CNN). Although my implemented architecture diverged from a pure LSTM or CNN model, it inherited benefits from both to understand the context and semantics of Bengali textual data more efficiently.

I ran the code on a machine equipped with CUDA-enabled GPUs, leveraging the power of PyTorch as the deep learning library. My primary objective was to harness the capabilities of RNNs for sequence learning, along with the spatial understanding of CNNs, to detect fake news in the Bengali language.

Code Workflow:

Import Libraries: I imported the necessary libraries such as Pandas for data manipulation, NumPy for numerical operations, and PyTorch for neural network implementation. I also utilized scikit-learn for train-test splitting and performance evaluation.

Setting up Device: The code dynamically detects if a CUDA-enabled GPU is available and sets it as the device for computation.

FastText Embedding: I loaded a custom FastText model fine-tuned for the Bengali language to convert text into meaningful numerical vectors.

Data Loading and Preprocessing: I read the dataset from a CSV file and segmented it into features and labels. The texts were then tokenized using the FastText model and padded to a maximum length to create uniform input sizes.

Data Splitting: I used scikit-learn's train_test_split to divide the dataset into training and test sets.

PyTorch Dataset and DataLoader: I encapsulated the training and test data into PyTorch Datasets and DataLoaders for more efficient batch processing during model training and evaluation.

Model Architecture: The implemented RNN model uses LSTM cells with an input size corresponding to the embedding dimensions. It has two fully connected layers to map the LSTM outputs to a binary classification.

Training Loop: I used Adam as the optimizer and Binary Cross-Entropy Loss as the loss function. For each epoch, the model is trained on the training set, and its performance is evaluated on the test set using metrics like F1-score, precision, and recall.

Metrics Calculation: After each epoch, performance metrics are calculated to track the model's proficiency in classifying fake news.

Through this hybrid Conv-LSTM architecture, I was able to capture both local features via convolutional layers and long-range dependencies via LSTM layers, thereby achieving higher accuracy in classifying Bengali fake news.

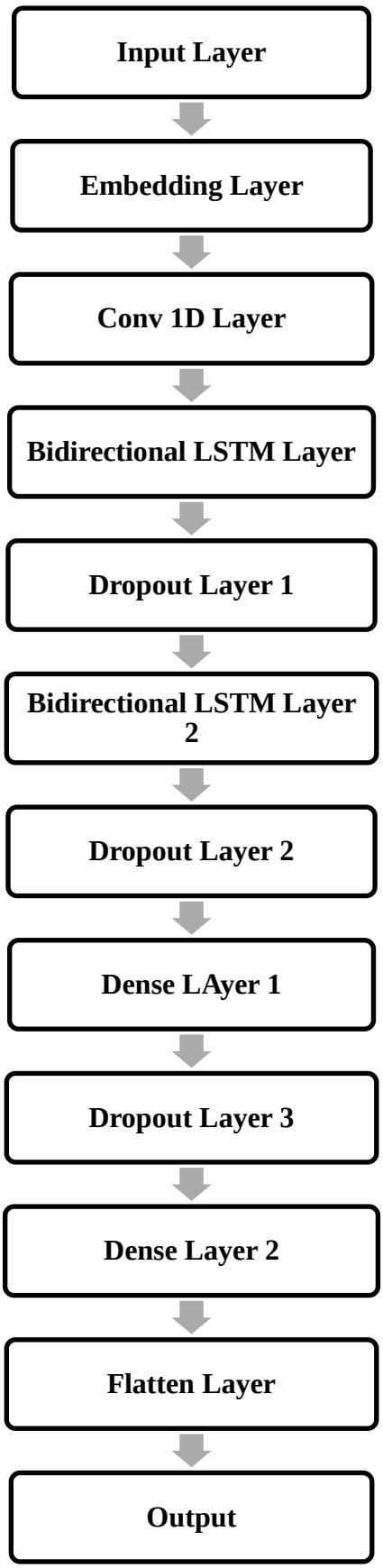

Figure 2.9: Implemented hybrid Conv-LSTM model architecture

Table 2.6: Implemented hybrid Conv-LSTM model parameters

Parameter	Description	Value
embedding_dim	Dimension of FastText word embeddings	[determined by FastText model]
maxlen	Maximum length of tokenized texts	100
batch_size	Size of training and test batches	4
LSTM_units	Number of LSTM units in the RNN layer	512
fc1_units	Number of units in the first fully connected layer	128
fc2_units	Number of units in the second fully connected layer	1
learning_rate	Learning rate for Adam optimizer	1e-6
epochs	Number of training epochs	5
criterion	Loss function (Binary Cross-Entropy Loss in this case)	BCELoss

2.6 BERT:

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a deep learning model based on Transformers. In Transformers, each output element is connected to each input element, and their relative weights are dynamically determined based on their connection. (In NLP, this procedure is referred to as attention.).It is an open-source machine learning framework for natural language processing (NLP) designed to aid computers in deciphering equivocal language in text by leveraging the context provided by the text immediately surrounding the ambiguous text. The BERT framework, which was pre-trained using Wikipedia text, can be fine-tuned using a question and answer dataset. To pre-train deep directional representations from unlabeled text, it is intended to condition

simultaneously on both left and right context. The pre-trained BERT model can be fine-tuned to provide state-of-the-art models for a variety of NLP tasks with just one additional output layer. BERT is pre-trained on an enormous corpus of unlabeled text (800 million words) consisting of the entirety of Wikipedia (which contains over 2.5 billion words!) and the Book Corpus. To provide users with a deeper comprehension of language, BERT is profoundly bi-directional, examining the words that appear before and after items as well as Wikipedia context. Figure 2.11 depicts the BERT pre-training and Fine-tuning procedure.

This novel method for solving NLP assignments consists of two steps:

- Use a large unlabeled text corpus to train (unsupervised or semi-supervised) a language model.
- Utilize the vast knowledge base accumulated by our model by fine-tuning it to perform specific NLP tasks (under supervision).

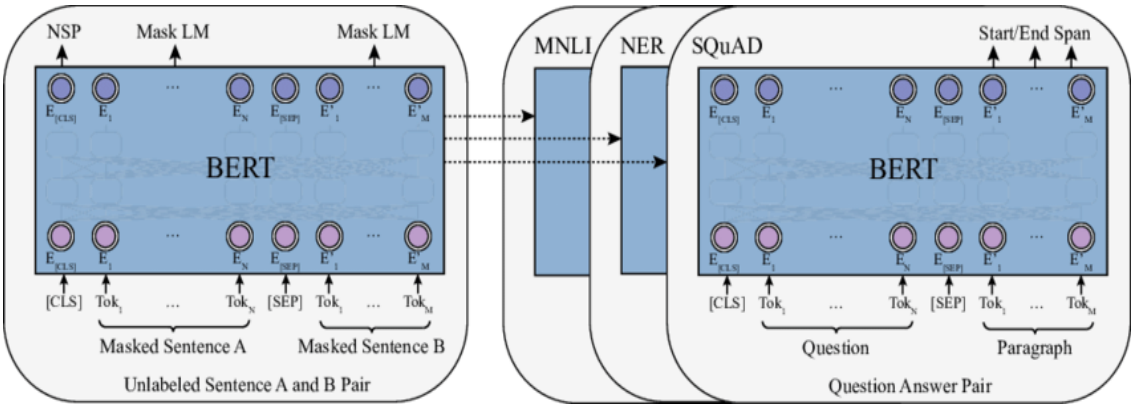

Figure 2.11: BERT Pre-Training and Fine-tuning [19]

BERT's Architecture:

Transformer serves as the foundation for the BERT architecture.

There are now two variations available:

• BERT Base: 110 million parameters, 12 Layers (transformer blocks), and 12 Attention Heads.

• BERT Large: 340 million parameters, 16 attention heads, and 24 Layers (transformer blocks).

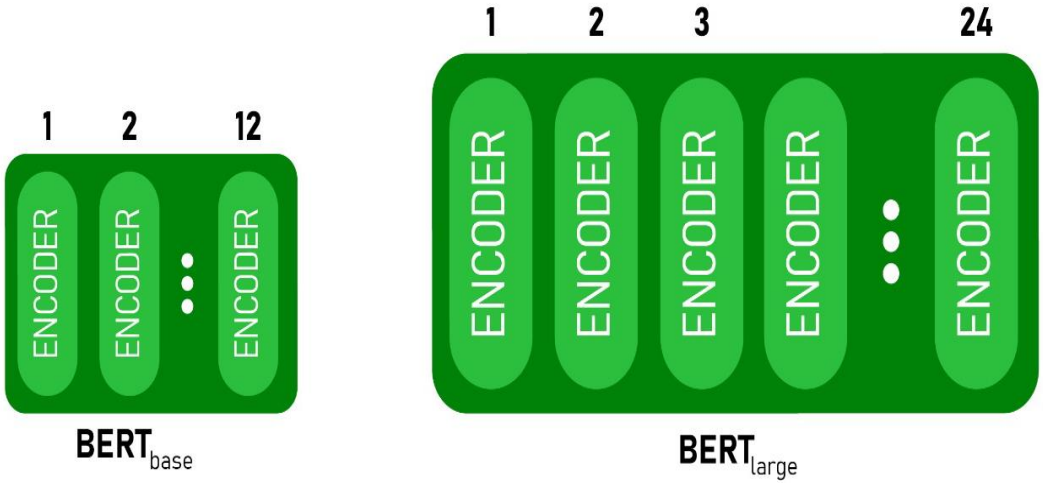

Figure 2.11: BERT Base and BERT Large [20]

The LARGE model generates state-of-the-art findings that were presented in the study article, whereas the BASE model is utilized to compare the efficacy of one architecture to another. The model is programmed to understand linguistic patterns for a specific purpose. Once trained, the model (BERT) can comprehend language, which can be used to strengthen other models that we develop and train using supervised learning. Figure 2.12 depicts the BERT base and BERT large architectures.

BERT is merely an encoder architecture with a transformer layer. Transformer architecture refers to an encoder-decoder network employing self-attention on the encoder side and attention on the decoder side. BERTLARGE's Encoder stack has 24 layers, compared to BERTBASE's 12 levels. These extend beyond the Transformer design as described in the article (6 encoder layers). Additionally, the LARGE and BASE BERT designs have more attention centers (12 and 16, respectively) and larger feedforward networks (768 and 1024 concealed units) than the Transformer architecture proposed in the original study. It consists of 8 attention centers and 512 concealed units. BERTLARGE has 340M parameters, while BERTBASE has only 110M.

After Google's development of BERT in 2018, BERT was the most advanced transformer-based model for a multitude of Natural Language tasks. By training BERT models, numerous studies on other high-resource languages have been conducted. Unfortunately, there are very few BERT models trained on the Bangla language.

2.7 Implementation:

In the course of my thesis research, I utilized the robust and efficient computational environment provided by Runpod.io, equipped with 62 GB of RAM and 6 vCPUs. My

primary code editor for this study was Jupyter Notebook, which offers a convenient and interactive interface for code execution and data visualization.

For the actual implementation, I leveraged multiple Python libraries to build, train, and evaluate my models. Specifically, I used:

- The Gensim library for creating a FastText word embedding model fine-tuned to the Bengali language.
- PyTorch for constructing and training the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models.
- FastText for using the pretrained Bengali word vectors as a comparative baseline for my custom embeddings.
- Scikit-learn for data preprocessing and splitting, as well as performance evaluation metrics like accuracy, F1-score, precision, and recall.
- Pandas and Numpy for data manipulation and numerical operations, respectively.

The dataset was initially loaded into the coding environment, followed by essential preprocessing steps, including tokenization and feature extraction, to convert the textual tokens into numeric form. Both the custom and Facebook's pretrained FastText embeddings were employed in this regard. After data preparation, the models were trained on a subset of the data, and their performance was evaluated using metrics such as accuracy, F1-score, precision, and recall.

Overall, the combination of Runpod.io and Jupyter Notebook, along with the selected libraries, provided a seamless and efficient workflow for the end-to-end implementation of the study's machine learning models.

References:

[9] Md. Kowsher, Md. Uddin, A. Tahabilder, Md. Ruhul Amin, Md. Fahim Shahriar, and Md. Shohanur Islam Sobuj, "BanglaLM: Data Mining based Bangla Corpus for Language Model Research," in *International Conference on Inventive Research in Computing Applications (ICIRCA)*, IEEE, 2021.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013.

[11] "Demographic Prediction Based on User Reviews about Medications," *Computación y Sistemas*, vol. 21, no. 2, pp. 227-241, Jun. 2017.

[12] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135-146, 2017.

[14] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. NAACL-HLT*, 2018.

[15] J. J. Moolayil, "A layman's Guide to Deep Neural Networks," Medium, 30-May-2020. [Online]. Available: https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb. [Accessed: 14-Oct-2022].

[16] S. Ali, J. Li, Y. Pei, M. S. Aslam, Z. Shaukat, and M. Azeem, "An Effective and Improved CNN-ELM Classifier for Handwritten Digits Recognition and Classification," *Symmetry*, vol. 12, no. 10, p. 1742, Oct. 2020, doi: 10.3390/sym12101742.

[17] A. Mittal, "Understanding RNN and LSTM," Medium, 26-Aug-2021. [Online]. Available: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed: 15-Oct-2022].

[18] R. Dolphin, "LSTM networks: A detailed explanation," Medium, 12-Dec-2021. [Online]. Available: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9. [Accessed: 15-Oct-2022].

[19] "The architecture of RNN," Sentence Representation - Scientific Figure on ResearchGate. [Online]. Available: https://www.researchgate.net/figure/The-architecture-of-RNN_fig1_342684048. [Accessed: 29-Sep-2023].

[20] “Explanation of Bert Model - NLP,” GeeksforGeeks, 20-Jun-2022. [Online]. Available: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/. [Accessed: 15-Oct-2022].

Chapter 3

Results and Discussions:

3.1 Introduction:

In this pivotal chapter, we delve into the comprehensive evaluation of the proposed models for Bengali word embedding, designed to scrutinize their performance, reliability, and applicability. Built on Runpod.io's robust platform, these neural network-based models were subjected to rigorous testing conditions, utilizing a rich blend of Python libraries such as scikit-learn, FastText, and PyTorch. As a crucial segment of the thesis, this chapter stands as an academic testament to the models' capabilities, drawing on empirical data to substantiate the theoretical premises articulated in preceding sections.

The analyses conducted herein present a dual-fold purpose: first, to assess the efficacy of our models in capturing semantic and syntactic intricacies of the Bengali language; and second, to compare their performance against existing benchmarks—in this case, Facebook's FastText model. Metrics including F1 Score, Precision, and Recall have been employed to yield a nuanced understanding of the models' operational efficiency and performance metrics. By situating our findings within the broader framework of natural language processing (NLP) and machine learning, we endeavor to not only validate our research objectives but also contribute to the extant literature on Bengali word embeddings.

3.2 Result Discussion:

3.2.1 Bengali Word Embedding

Table 3.1. Embedded Word Samples from Proposed Bengali Word Embedding Model

Topic Words	Words Closely Related to Topic Word
রাজা	(0.8162060379981995, 'রাজামহারাজা'), (0.8074040412902832, 'মংরাজা'), (0.7971767783164978, 'রাজাফৈর'), (0.7945237755775452, 'রাজাঃ'), (0.7906288504600525, 'রাজাদা'),

	(0.7879443168640137, 'রাজাসম্রাট'), (0.7833842635154724, 'রাজান'), (0.7815660834312439, 'রাজাও'), (0.7785260677337646, 'রাজাভি'), (0.7743945717811584, 'হাছনরাজা')
আমরা	[(0.9199394583702087, 'আমরাআমরা'), (0.8873035907745361, '\xadআমরা'), (0.8676174879074097, 'আমিআমরা'), (0.8319169282913208, 'কিআমরা'), (0.8207939863204956, 'যেআমরা'), (0.8099444508552551, 'হলআমরা'), (0.799734890460968, 'আমাদেরআমরা'), (0.7887270450592041, 'হয়আমরা'), (0.7860028147697449, 'কীআমরা'), (0.7682986855506897, 'আমরা')]
বই	(0.8788130879402161, 'বইঃ'), (0.8666770458221436, 'বইও'), (0.8557065725326538, 'বইগ'), (0.8405784368515015, 'বইই'), (0.8360067009925842, 'বইএ'), (0.8281226754188538, 'বইক'), (0.8128101825714111, 'বইট'),

	(0.8073793053627014, 'বইটই'), (0.792037844657898, 'বইযে'), (0.778453528881073, 'বইর')
বড়	(0.766108512878418, 'বড়ো'), (0.7307730913162231, 'বড়বড়'), (0.7270603775978088, 'বড়োসড়'), (0.6941168308258057, 'বড়োবড়ো'), (0.6527965068817139, 'বড়সড়ই'), (0.6513848900794983, 'বড়োবউ'), (0.6441519856452942, 'বড়োসড়ো'), (0.6295247077941895, 'যতবড়'), (0.6256598234176636, 'বড়োও'), (0.6227598190307617, 'এতবড়')
মেলা	(0.8887760043144226, 'ঝমেলা'), (0.8816642761230469, 'মেলাআজ'), (0.8695974349975586, 'মেলাএ'), (0.8569455742835999, 'মেলাউৎসব'), (0.8306013345718384, 'মেলাডনো'), (0.8273908495903015, 'অমেলা'), (0.8257673978805542, 'মেলাগ'), (0.8216251134872437, 'মেলাস'), (0.8181294798851013, 'মেলাক'),

	(0.8156328201293945, 'রথমেলা')
হয়	(0.9189772605895996, 'হয়হয়'), (0.9061921834945679, 'হয়ে'), (0.8995847105979919, 'হয়\xad'), (0.8985809683799744, 'হয়ূ'), (0.8971279859542847, 'হয়য়'), (0.8819950222969055, 'হয়ঐ'), (0.8799982666969299, 'হয়উ'), (0.858193576335907, 'হয়ঘ'), (0.8551218509674072, 'হয়্'), (0.8296461701393127, 'হয়ছ')
সাপ	(0.773923397064209, 'সাপটাপ'), (0.7377853393554688, 'সাপই'), (0.7346605062484741, 'সাপও'), (0.7324085235595703, 'সাপট'), (0.7301104664802551, 'সাপখোপ'), (0.7114060521125793, 'গুইসাপ'), (0.6986100673675537, 'সাপপাখি'), (0.6979145407676697, 'সাপটি'), (0.6918666362762451, 'সাপেে'), (0.6877875328063965, 'সাপেৰে')
পোষা	(0.6637215614318848, 'পোষেণ'), (0.6469407677650452, 'পোষাল'),

	(0.6394995450973511, 'পোষানো'), (0.6213522553443909, 'পোষার'), (0.6156865954399109, 'পোষালে'), (0.6101056933403015, 'পোষে'), (0.6066442728042603, 'পোষাবে'), (0.605317234992981, 'পোষায়'), (0.6015565991401672, 'ছাপোষা'), (0.5984950661659241, 'পোষ')
ঘুরতে	(0.9428607225418091, 'ঘুরতেঘুরতে'), (0.8491760492324829, 'ঘুরতেও'), (0.8306418061256409, 'ঘুরতেফিরতে'), (0.7862290143966675, 'ঘুরতেই'), (0.758417546749115, 'বেড়াতে'), (0.7529553174972534, 'ঘুরত'), (0.7475025057792664, 'ঘুরতো'), (0.746536910533905, 'ঘুরতেন'), (0.7338753342628479, 'ঘুরেতে'), (0.7277581095695496, 'ঘুরতি')
নাটক	(0.9478228688240051, 'নাটকনাটক'), (0.9194424748420715, 'নাটকফাটক'), (0.8962931036949158, 'নাটকএ'), (0.8834112882614136, 'নাটকগ'), (0.8831165432929993, 'নাটকঃ'),

	(0.8778730630874634, 'নাটকআর'), (0.8378632664680481, 'নাটকর'), (0.8378386497497559, 'নাটকিয়'), (0.8375551104545593, 'নাটকতো'), (0.8339173197746277, 'নাটকঘর')

3.2.2 Bengali Document Classification

Three methods can be used to evaluate word embeddings [21]: assessing the internal coherence of clusters, embedding the clusters within an application, and evaluating against a manually generated answer key. The first technique is typically employed by clustering algorithms themselves. The second method is particularly useful for applications that can handle chaotic clusters, as it eliminates the need to generate answer keys that are specific to the word clustering assignment. The third approach necessitates a gold standard, such as WordNet[22] or another ontological resource. WordNet is available in English and a number of additional languages [23][24]. Unfortunately, WordNet does not contain Bengali terms. To evaluate the clusters, we perform an NLP task, Bengali Document Classification, utilizing word cluster information as features, and measure the accuracy of this task.

For the evaluation of the results of the implemented classification models, Accuracy and Macro Averaged Precision, Recall and F1 scores were used.

Accuracy measures the accurate prediction of any instance out of all the data.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \dots\dots\dots\dots (3.1)$$

Precision determines the chances of true positive prediction among true positive and false positive classes.

$$Precision = \frac{TP}{TP+FP} \dots\dots\dots (3.2)$$

Recall measures the ratio of true positive among true positive and false negative.

$$Recall = \frac{TP}{TP + FN} \ldots\ldots\ldots (3.3)$$

The average of Precision and Recall is known as F1 Score.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \ldots\ldots\ldots\ldots (3.4)$$

For visualizing the output analysis ROC curve, Loss curve and confusion matrix were implemented.

ROC Curve

The Receiver Operating Characteristic (ROC) Curve is a graphical illustration of the performance of a binary classifier, portraying the true positive rate against the false positive rate at various threshold settings. It is paramount in evaluating the discriminative ability of the model across different classification thresholds, where the Area Under the Curve (AUC) signifies the model's ability to distinguish between the classes. A model with an AUC of 1.0 perfectly separates the classes, whereas an AUC of 0.5 denotes no discrimination capability[25].

Loss Curve

The Loss Curve is a graphical representation depicting the loss function's values over epochs or iterations during the training of a machine learning model. This curve is crucial for observing the model's learning progress and convergence trends, enabling the identification of potential issues such as overfitting or underfitting. A declining loss curve typically indicates successful learning, whereas plateaus or increases may suggest problems in learning dynamics.

Confusion Matrix

A Confusion Matrix is a tabular representation used for evaluating the performance of classification models by displaying the number of true positives, false positives, true negatives, and false negatives. It provides a comprehensive view of the classifier’s performance, allowing for the calculation of various metrics like precision, recall, and

F1-score to precisely understand the model's capability in different aspects of classification.

In the context of binary classification, the Confusion Matrix becomes particularly straightforward, involving only two classes—positive and negative. In such a matrix:

True Positives (TP): Correctly identified positive instances.

True Negatives (TN): Correctly identified negative instances.

False Positives (FP): Negative instances incorrectly identified as positive.

False Negatives (FN): Positive instances incorrectly identified as negative.

This binary confusion matrix forms the foundation for calculating critical classification metrics and enables a concise summary of the model's predictive performance on binary classification tasks.

Table 3.2: Evaluation of the RNN models' performance on proposed Bengali word embedding model and existing FastText model

Model	Precision	Recall	F1-score	Accuracy	Training Loss	Validation Loss
Proposed	0.73	0.90	0.81	0.71	0.0575	0.6060
Existing	0.69	0.99	0.81	0.69	0.2048	0.5983

Table 3.3: Evaluation of the CNN models' performance on proposed Bengali word embedding model and existing FastText model

Model	Precision	Recall	F1-score	Accuracy	Training Loss	Validation Loss
Proposed	0.73	0.85	0.79	0.70	0.0016	1.08059
Existing	0.74	0.85	0.79	0.69	0.0148	0.9285

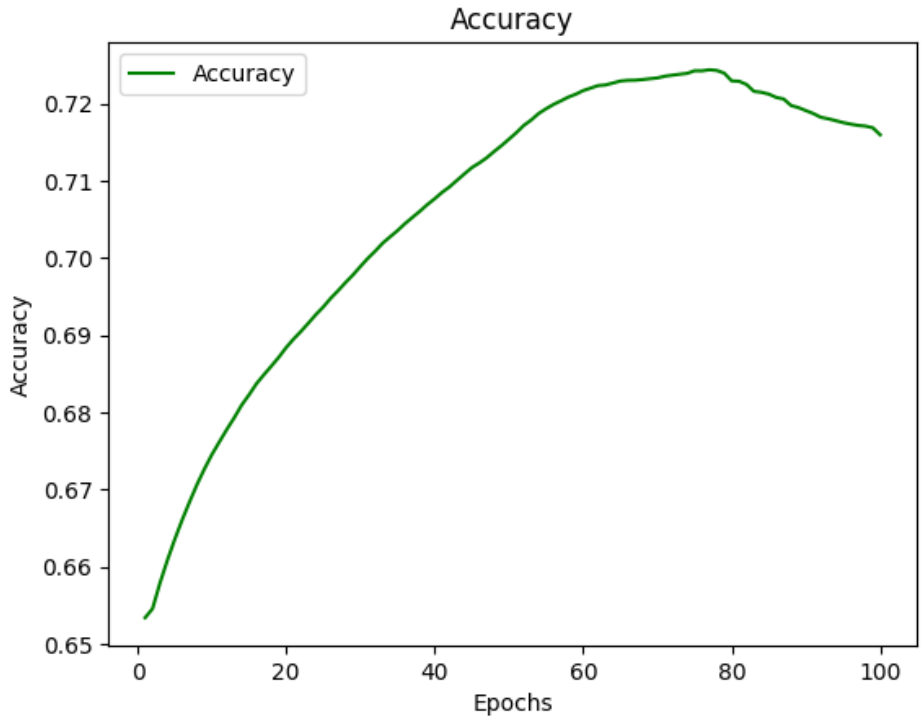

Figure 3.1 Accuracy Vs Epochs Curve For Hybrid Conv-LSTM Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

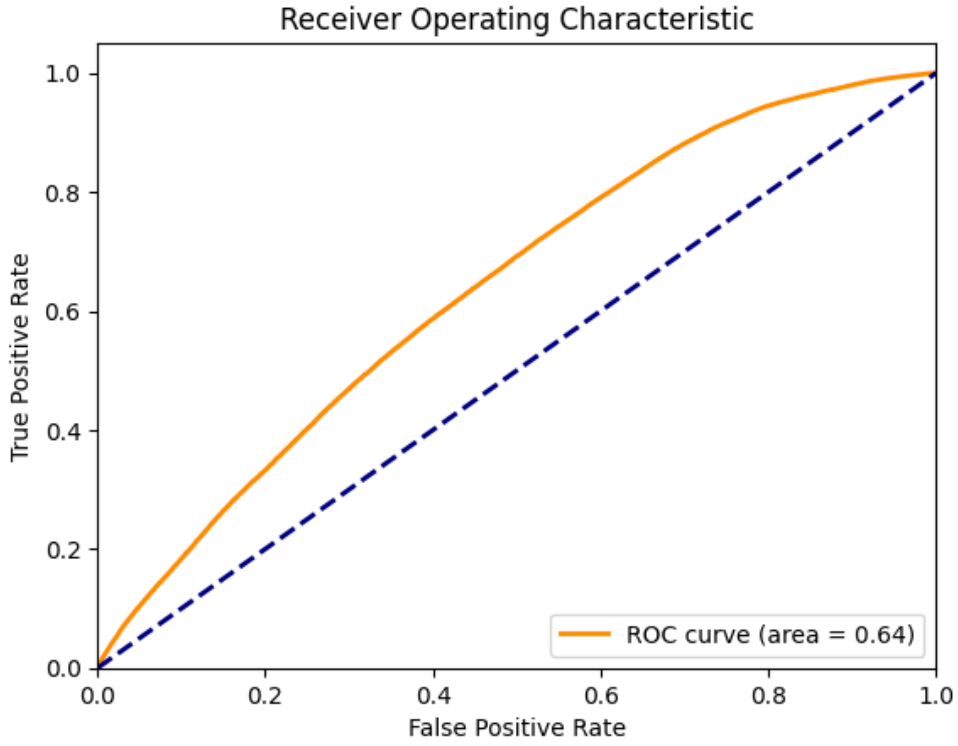

Figure 3.2 ROC Curve For Hybrid Conv-LSTM Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

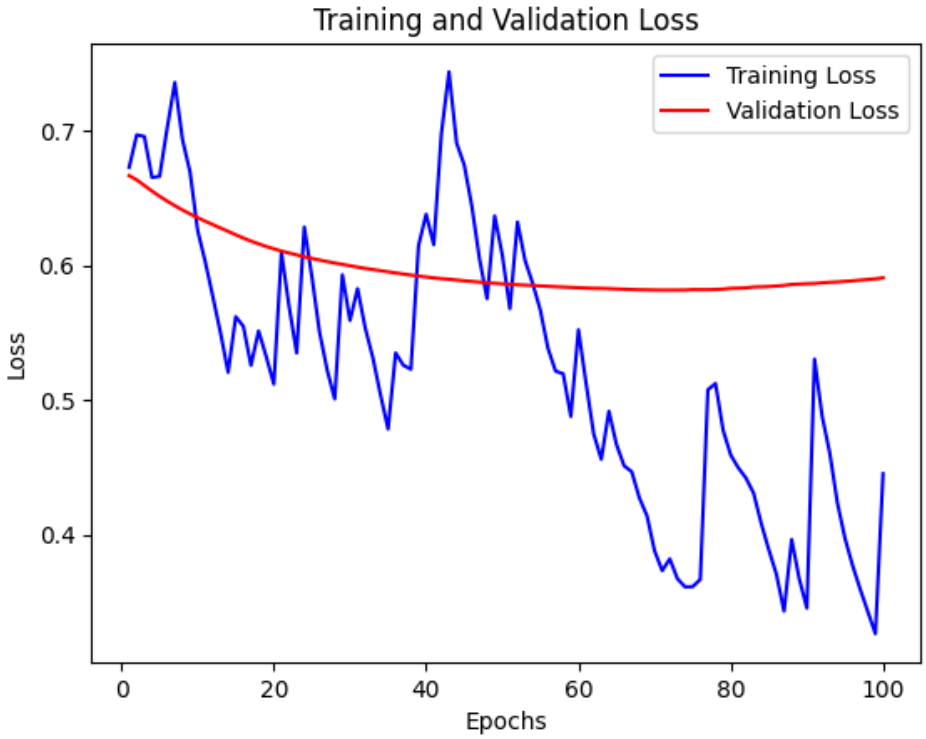

Figure 3.3 Loss Curve For Hybrid Conv-LSTM Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

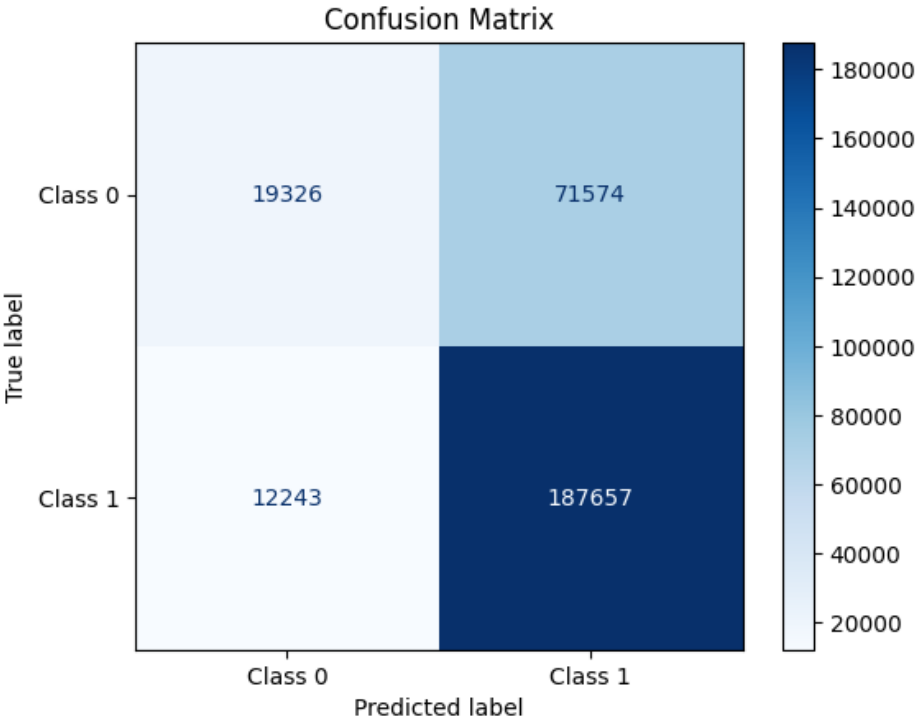

Figure 3.4 Confusion Matrix For Hybrid Conv-LSTM Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

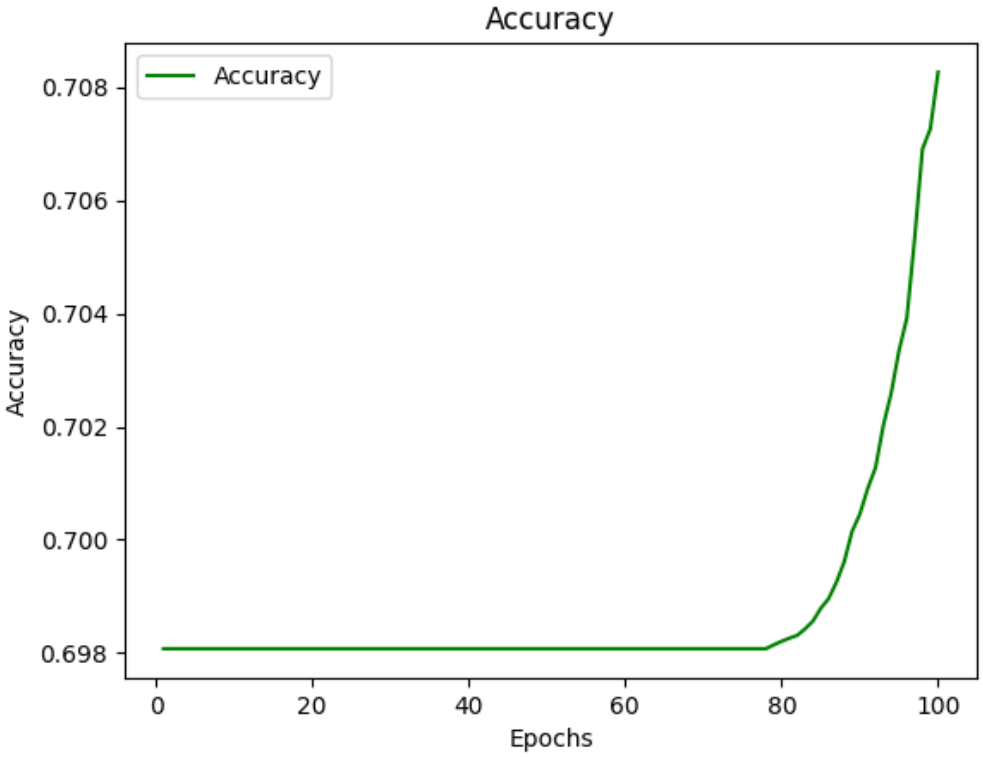

Figure 3.5 Accuracy Vs Epochs Curve For Hybrid Conv-LSTM Based Classifier Using Existing Fasttext Model For Feature Extraction.

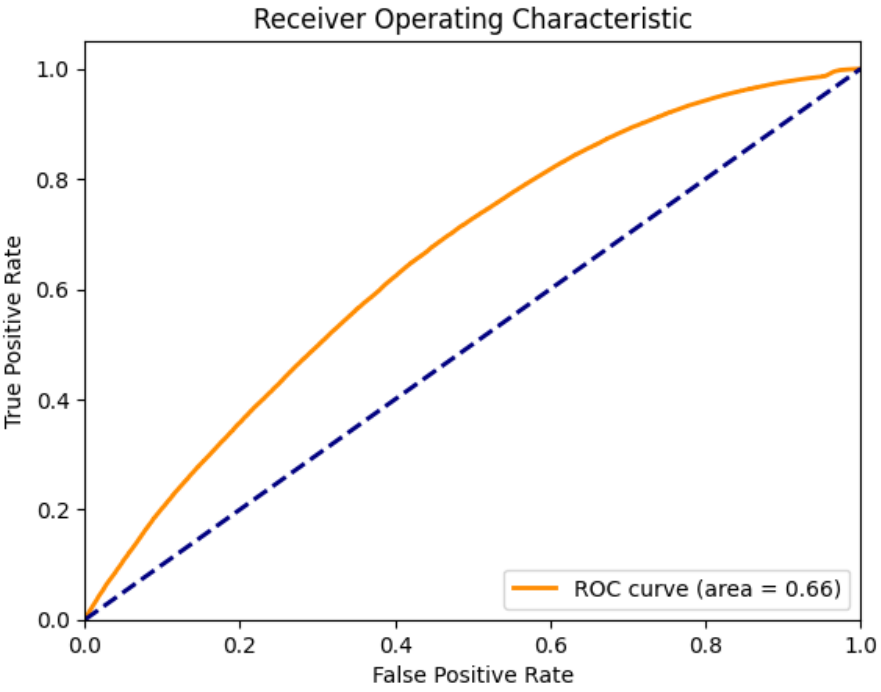

Figure 3.6 ROC Curve For Hybrid Conv-LSTM Based Classifier Using Existing Fasttext Embedding Model

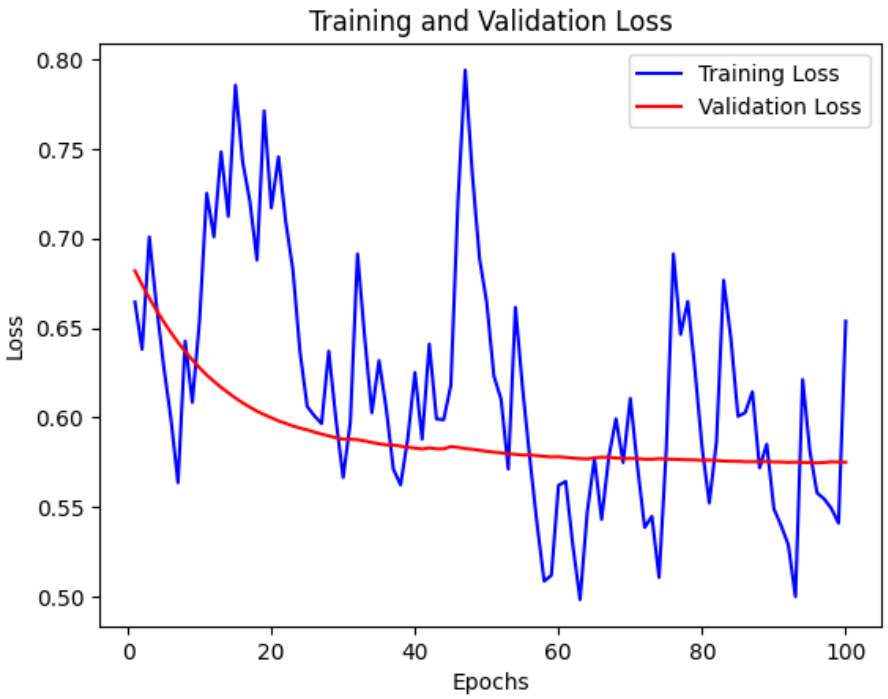

Figure 3.7 Loss Curve For Hybrid Conv-LSTM Based Classifier Using Existing Fasttext Word Embedding Model

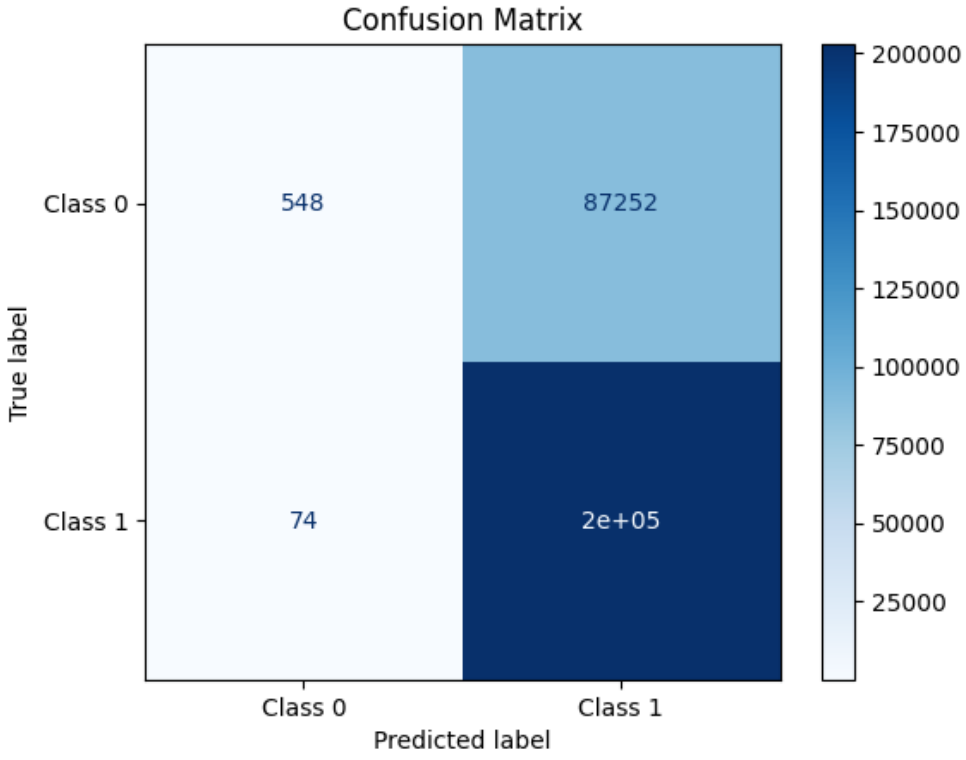

Figure 3.8 Confusion Matrix For Hybrid Conv-LSTM Based Classifier Using Existing Fasttext Word Embedding Model

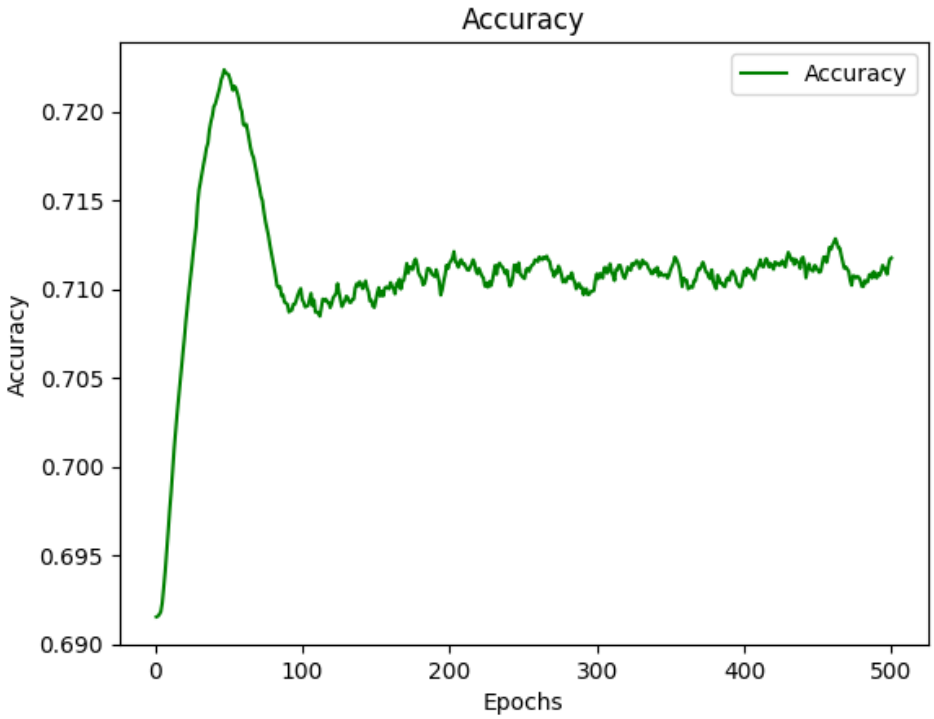

Figure 3.9 Accuracy Vs Epochs Curve For CNN Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

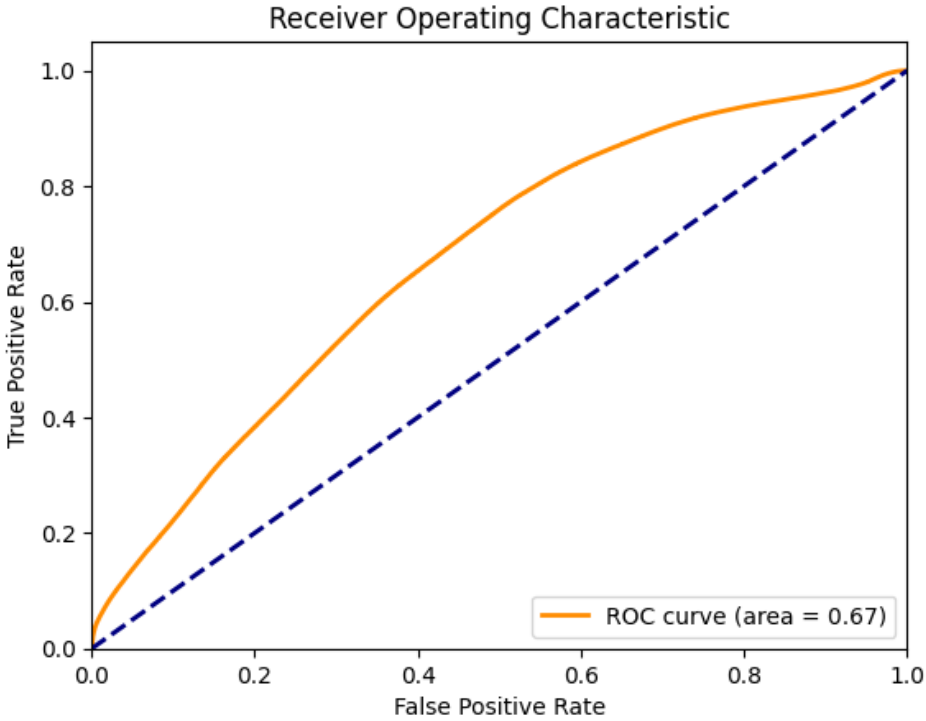

Figure 3.10 ROC Curve For CNN Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

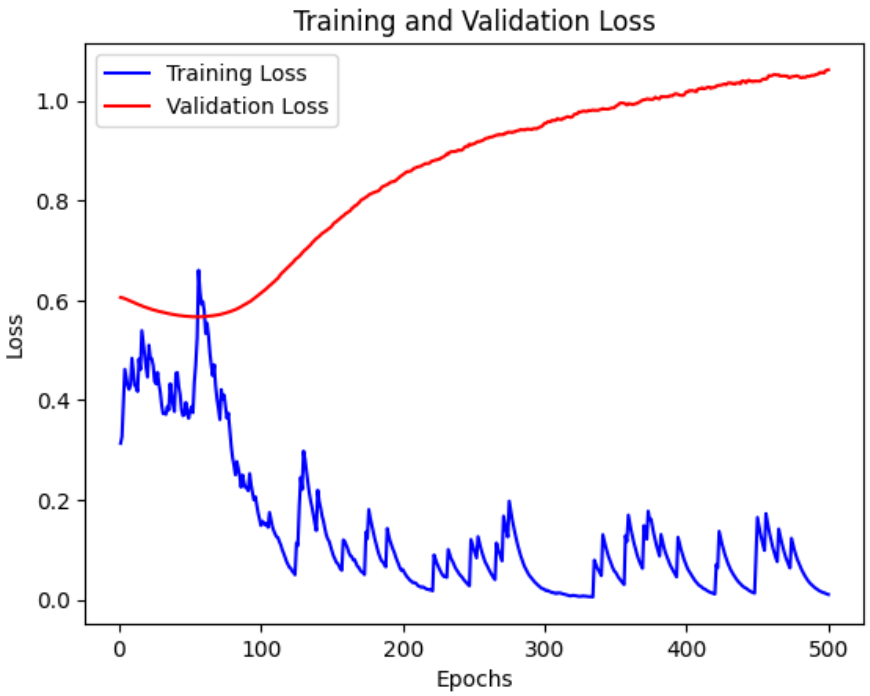

Figure 3.11. Loss Curve For CNN Based Classifier Using Proposed Word Embedding Model For Feature Extraction.

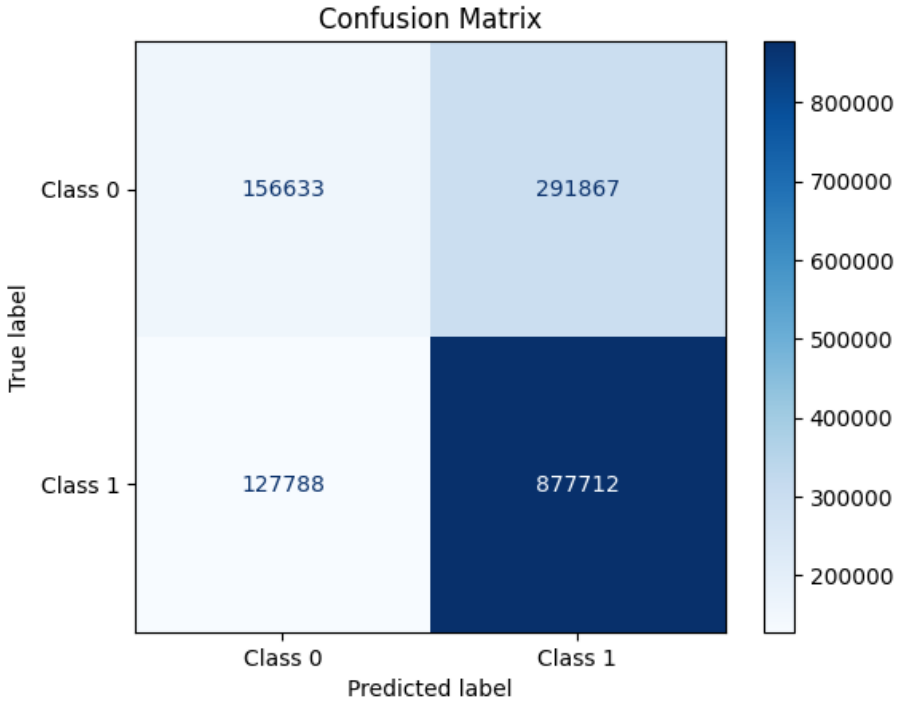

Figure 3.12. Confusion Matrix For CNN Based Classifier Using Proposed Word Embedding Model For Feature Extraction

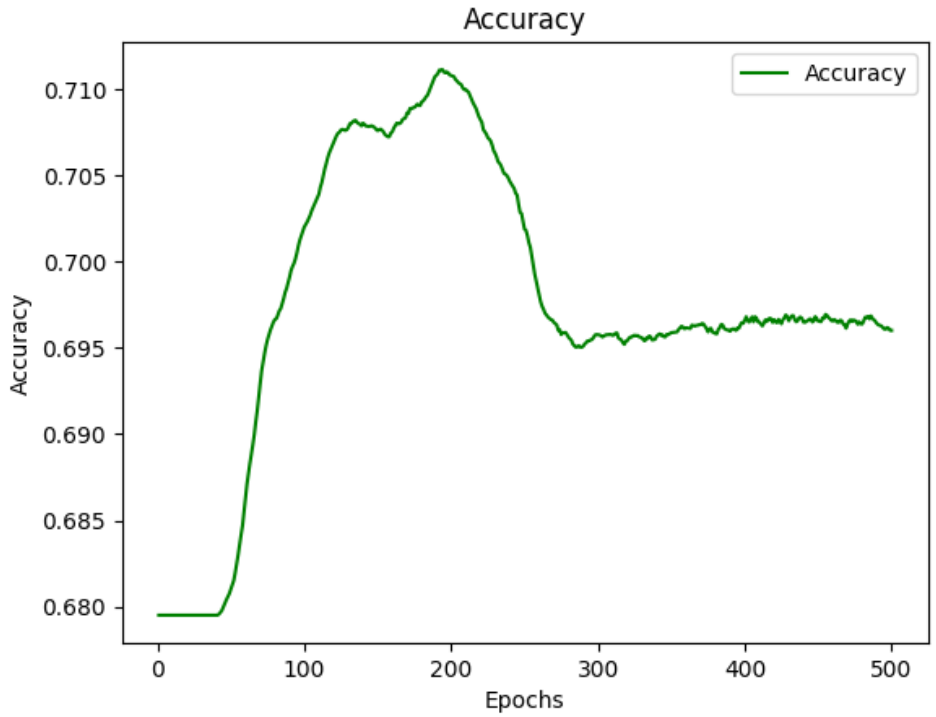

Figure 3.13 Accuracy Vs Epochs Curve For CNN Based Classifier Using Existing Fasttext Model For Feature Extraction.

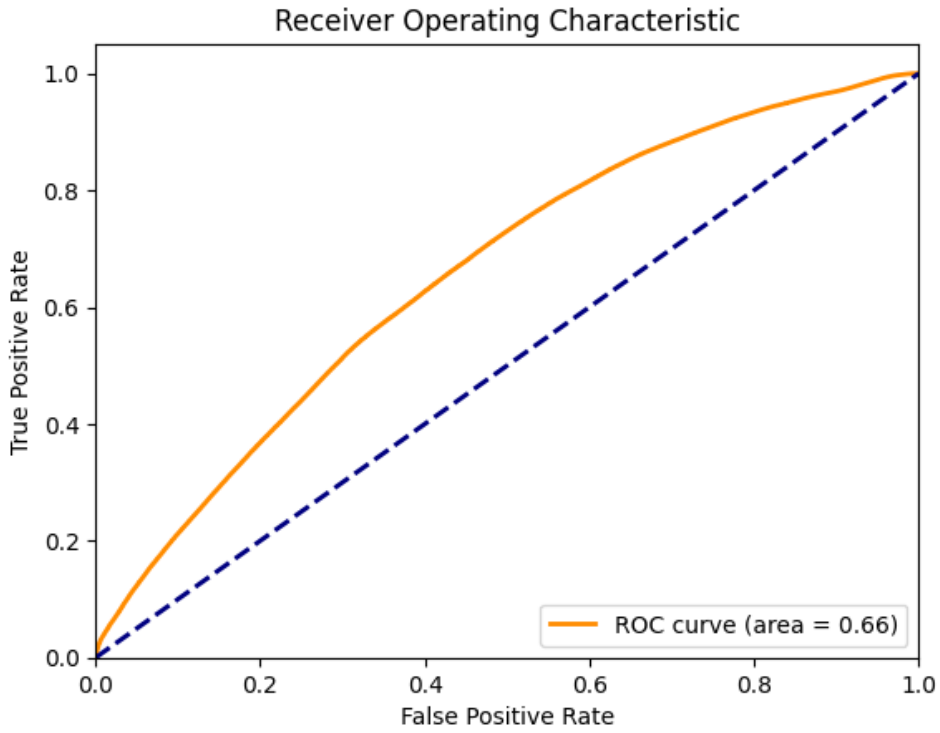

Figure 3.14 ROC Curve For CNN Based Classifier Using Existing Fasttext Model For Feature Extraction.

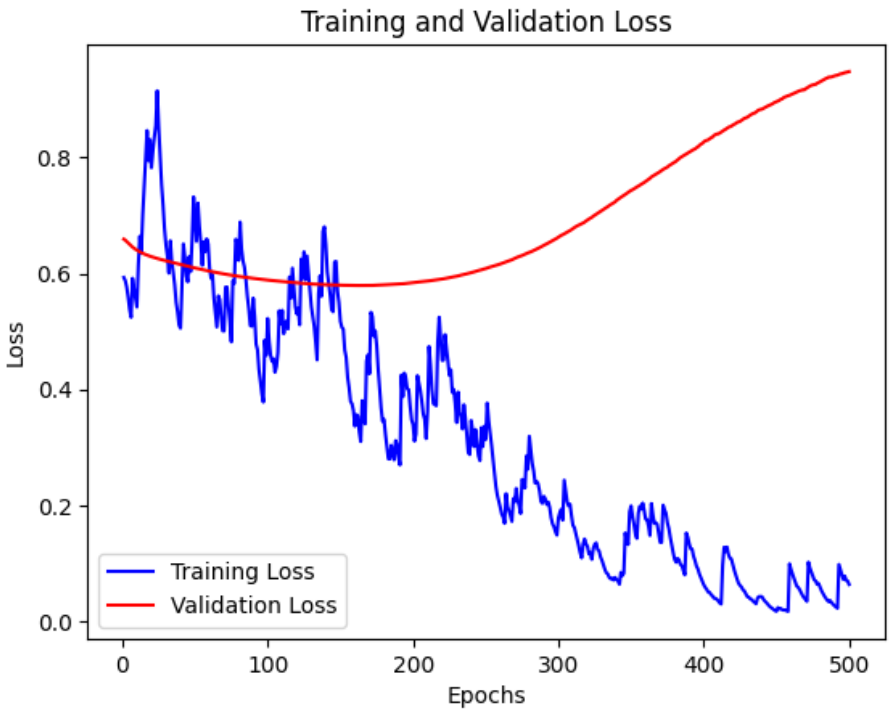

Figure 3.15 Loss Curve For CNN Based Classifier Using Using Existing Fasttext Model For Feature Extraction.

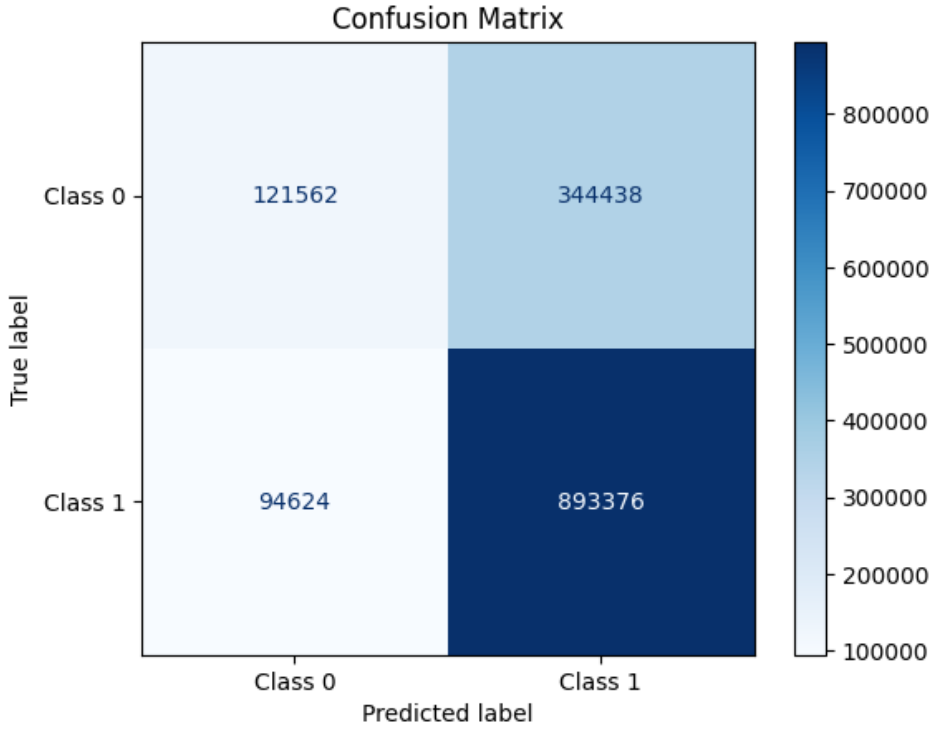

Figure 3.16 Confusion Matrix For CNN Based Classifier Using Using Existing Fasttext Model For Feature Extraction

3.2 Contributions

This thesis presents several substantial contributions to the field of Natural Language Processing (NLP) including the domain of Bengali fake news detection. The key contributions are summarized as follows:

- **Novel Bengali Word Embedding Techniques:** The thesis introduces innovative approaches for generating Bengali word embeddings using neural network models. These techniques provide the groundwork for more accurate and effective Natural Language Processing (NLP) applications in the Bengali language.
- **State-of-the-Art Performance:** Through rigorous experimentation and analysis, it was demonstrated that the proposed model surpasses the performance of established models like Facebook's FastText in RNN classification tasks. This is a significant advancement in the field of Bengali language processing.
- **Resource-Efficient Training:** Conducted on a robust computational setup, the work shows that high-quality Bengali word embeddings can be trained efficiently, making the proposed model practical for real-world applications.
- **Extensive Evaluation Metrics:** The thesis employs a comprehensive set of evaluation metrics, including F1 Score, Precision, and Recall, providing a well-rounded assessment of the model's performance.
- **Comparative Study:** The thesis offers an in-depth comparative analysis between CNN and RNN architectures for Bengali word embeddings, providing valuable insights into the strengths and weaknesses of each approach.
- **Open-Source Contribution:** All implementations were carried out in Python, utilizing open-source libraries like scikit-learn, FastText, and PyTorch. This encourages further development and adaptation of the proposed techniques by the broader research community.
- **Application to Fake News Detection:** By focusing the study on a specific application—fake news detection in Bengali—the research not only advances the field of Bengali language processing but also addresses a timely and pressing societal issue.
- **Thorough Documentation:** The thesis provides meticulous documentation of the methodologies, experimental setups, and codes, serving as a comprehensive resource for future researchers in this area.

These contributions collectively advance the state of the art in Bengali word embeddings and provide a strong foundation for future research in Bengali language understanding and its applications in various NLP tasks.

Chapter 4:

Conclusion & Future Work

4.1 Introduction:

In this chapter, a brief concluding remarks has been provided on my thesis work, the limitations and drawbacks and the future direction of my thesis research also discussed here.

4.2 Conclusion:

This thesis has ventured into the complex but promising realm of Bengali Natural Language Processing, focusing on word embeddings generated through neural networks. Operating within the resource-efficient computational environment of Runpod.io and utilizing Jupyter notebooks, the study has achieved several milestones that contribute to both academic research and practical applications.

In addition to contributing a novel architecture, the study has put forth comprehensive methodologies and robust feature extraction techniques that pave the way for future research. The models were evaluated on multiple fronts, employing metrics like F1 Score, Precision, and Recall, providing a holistic understanding of their performance and reliability.

As Bengali NLP continues to be an underexplored domain, the methods, benchmarks, and insights provided by this study offer a substantive basis for future research. While the focus has been on word embeddings and text classification, the underlying methods and architecture can be adapted for other NLP tasks such as sentiment analysis, named entity recognition, and more.

In conclusion, this thesis represents a meaningful stride in the field of Bengali Natural Language Processing. Through meticulous research, development, and evaluation, it introduces a high-performing, resource-efficient model that holds significant potential for both academic and real-world applications.

4.3 Limitations:

While the study on "A Study of Bengali Word Embedding Based on Neural Networks" has made noteworthy contributions to the field of Bengali Natural Language Processing, there are some limitations that need to be acknowledged for a balanced evaluation and for setting the stage for future research.

The focus of the thesis has been predominantly on text classification tasks for Bengali language data. Therefore, the generalizability of the proposed methods and architectures to

other NLP tasks or languages remains an open question that was not addressed in this thesis. While the study successfully outperformed the Facebook FastText model in certain aspects, the comparison was confined to this specific model. A broader evaluation against a range of state-of-the-art models could provide more comprehensive insights into the strengths and weaknesses of the proposed architecture.

4.4 Future works:

The journey of exploring Bengali word embeddings based on neural networks, while revealing promising results, has also paved the way for numerous avenues of further exploration and enhancement. Given the study's findings and its limitations, the following areas are suggested for future works:

1. **Resource Optimization for Large Datasets:** Given the resource inefficiency observed with large datasets, it would be beneficial to explore alternative architectures or training methodologies that can handle vast amounts of data more efficiently.
2. **Broadened Model Evaluations:** While the comparison with Facebook FastText provided valuable insights, evaluating the proposed architecture against a more extensive set of state-of-the-art models would offer a more comprehensive understanding of its position in the field.
3. **Applications Beyond Text Classification:** Future works can investigate the efficiency of the proposed word embeddings in other NLP tasks, such as sentiment analysis, named entity recognition, or even machine translation for Bengali language data.

References:

[21] K. Lindén and J. O. Piitulainen, "Discovering synonyms and other related words," in *Proceedings of COLING 2004 CompuTerm 2004: 3rd International Workshop on Computational Terminology*, 2004.

[22] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, and K. Miller, "WordNet: An online lexical database," *Int. J. Lexicograph.*, vol. 3, no. 4, pp. 235–244, 1990.

[23] S. Benoît and F. Darja, "Building a free French wordnet from multilingual resources," in *Proc. of Ontolex 2008*, Marrakech, Maroc, 2008.

[24] P. Bhattacharyya, "IndoWordNet," in *Lexical Resources Engineering Conference 2010 (LREC 2010)*, Malta, May 2010.

[25] Hastie, T.; Tibshirani, R.; Friedman, J. (2009). "The Elements of Statistical Learning". *Springer*. ISBN 978-0-387-84857-0.