A Mini Research

ON

Comparative Analysis of Nepali Language Sentiment Classification Using Machine Learning and Deep Learning Approaches

Submitted By

Teksan Gharti Magar

Er. Yuba Raj Devkota

Lecturer, IT Department

Submitted To

Research Management Cell (RMC)

National College of Computer Studies NCCS)

Paknajol, Kathmandu

2083

Acknowledgement

We would like to express our sincere gratitude to everyone who supported and encouraged us during the completion of this mini research entitled “Comparative Analysis of Nepali Language Sentiment Classification Using Machine Learning and Deep Learning Approaches.” We are especially thankful to the Research Management Cell (RMC), National College of Computer Studies (NCCS), Paknajol, Kathmandu, for providing us with the opportunity and necessary support to conduct this research. We would also like to express our heartfelt appreciation to Vice Principal **Mr. Santosh Raj Maskey** and Director **Mr. Prajwal Baniya Chhetri** for their valuable encouragement, guidance, and support throughout this research work.

To conclude, the authors wish to express their gratitude to all those who have directly or indirectly helped in the successful completion of this study. We are very thankful for your valuable advice, motivation and cooperation. We would also like to thank our colleagues, friends and family for their constant encouragement and support throughout the research process. The support and goodwill of all those who participated in this work made it possible to finish it.

Abstract

This study presents a comparative analysis of Machine Learning and Deep Learning approaches for Nepali language sentiment classification. The main objective was to evaluate different classification models and determine their effectiveness in identifying sentiment from Nepali textual data. A dataset of 35789 Nepali text samples was downloaded from Kaggle and after removing the duplicate records 35732 samples were used for the experiment. The dataset was pre-processed and split into training and testing datasets at the ratio of 80:20. We used three machine learning algorithms for the task, Naïve Bayes, Support Vector Machine (SVM) and Logistic Regression, with features based on the text using TF-IDF. For Deep Learning, LSTM, BiLSTM and CNN models were implemented with embedding dimension of 128, batch size of 64 and 10 epochs. The models were evaluated using accuracy, precision, recall and F1-score. The best performance of Machine Learning was obtained by Logistic Regression with accuracy of 67.42% while the best performance of Deep Learning was obtained by CNN with accuracy of 69.06%. In the last comparison CNN had a higher precision and Logistic Regression had a higher accuracy. In the last comparison CNN has better accuracy while Logistic Regression has better precision (70%), recall (73%) and F1-score (71%). The findings demonstrate that both approaches provide competitive results for Nepali sentiment classification and that model selection should consider multiple evaluation metrics rather than accuracy alone.

The findings in this mini research demonstrate that both Machine Learning and Deep Learning approaches can provide competitive results for Nepali sentiment classification. The study also provides a practical reference for National College of Computer Studies (NCCS) in understanding the application of Machine Learning and Deep Learning techniques in Nepali language processing and in supporting future academic and practical work related to Artificial Intelligence and Natural Language Processing.

Keywords: *Nepali Language, Sentiment Analysis, Natural Language Processing, Machine Learning, Deep Learning, TF-IDF, Logistic Regression, Naive Bayes, SVM, Random Forest, LSTM, BiLSTM, CNN.*

Table of Contents

List of Tables

List of Figures

List of Symbols, Abbreviations and Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
F1	F1-Score
LSTM	Long Short-Term Memory
ML	Machine Learning
NB	Naive Bayes
NLP	Natural Language Processing
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
BiLSTM	Bidirectional Long Short-Term Memory
TP	True Positive
FN	False Negative

Chapter I: Introduction

Background

The advent of digital communication and social networking sites has seen an explosion of texts created everyday. Such text data includes opinions, emotions and sentiments expressed by the users. Sentiment analysis is a branch of Natural Language Processing (NLP) which involves automatic extraction of the sentiments from the text data. It is used in many fields such as review systems, social media analysis, feedback collection, public opinion analysis and recommendations. Machine learning methods are normally employed to classify the sentiments expressed in text. Machine learning approaches like Logistic Regression, Naive Bayes, SVM (Support Vector Machine) and Random Forest can classify the text after the textual data is converted into numeric representation. One popular method for representing the importance of the words in a document is TF-IDF.

In recent years, deep learning approaches have provided alternative methods for processing textual information. Useful patterns and representations can be learned from the text data by neural network architectures like Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM) and Convolutional Neural Network (CNN). These approaches have shown promising performance on various NLP tasks. However, Sentiment analysis in Nepali language is still relatively less explored than the widely studied languages like English. Nepali has its own linguistic characteristics, vocabulary, grammatical constructions and variations in digital text. The availability of suitable datasets and computational resources for Nepali NLP is also comparatively limited.

Students in programs of NCCS such as BCA, BITM, and CSIT frequently select Machine Learning and Deep Learning topics for their academic and final-year projects, Therefore, this study focuses on Nepali-language sentiment classification using both traditional machine learning and deep learning approaches. A dataset of 35,789 Nepali text samples obtained from Kaggle will be used. We will compare machine learning algorithms such as Logistic Regression, Naive Bayes, SVM and Random Forest with TF-IDF based feature extraction with deep learning models such as LSTM, BiLSTM and CNN. The models will be evaluated based on Accuracy, Precision, Recall, F1-score and confusion matrices for their relative performance.

Statement of the Problem

The major problems addressed by this study are:

- The amount of Nepali-language content on social media, websites, online reviews, and digital platforms is increasing, but there are comparatively fewer studies focusing on automatic sentiment analysis of Nepali text.
- Most of the current research and tools for sentiment analysis are built for English and other popular languages, but Nepali has received relatively less focus.
- Nepali text might have different writing styles, spelling variations, informal expressions and mixed-language words that may make sentiment classification challenging. It is not clear which approach is more suitable for Nepali sentiment classification: traditional machine learning methods or deep learning methods.
- Traditional machine learning models such as Logistic Regression, Naive Bayes, SVM, and Random Forest can provide good results with TF-IDF features, but their performance needs to be compared with deep learning models.
- Deep learning models such as LSTM, BiLSTM, and CNN can learn patterns from text automatically, but their effectiveness for the selected Nepali dataset needs to be examined.
- There is a need for a common experimental comparison of these models using the same dataset and evaluation measures.
- Therefore, this study attempts to compare the selected machine learning and deep learning models using Accuracy, Precision, Recall, F1-score, and confusion matrices and identify which approaches perform better for the selected Nepali sentiment dataset.

Rationale of the Study

The other language spoken in Nepal apart from English is the Nepali language which has been incorporated in digital communication. The availability of Nepali textual data provides an opportunity to develop NLP-based applications that can automatically understand people's opinions and sentiments. This study is important because it provides a comparative evaluation of different machine learning and deep learning techniques for Nepali sentiment classification. Rather than focusing on only one algorithm, the study evaluates multiple approaches under a common experimental framework.

The findings may help students, researchers, and developers understand the relative effectiveness of traditional machine learning and deep learning models for Nepali text classification. The study may also provide a foundation for future research in Nepali NLP, sentiment analysis, social media analysis, and other language-processing applications. For a college-level mini research project, the study also provides practical exposure to dataset preparation, text preprocessing, feature extraction, model development, model evaluation, and comparative analysis.

Research Objectives

General Objective

- To comparatively evaluate machine learning and deep learning approaches for sentiment classification of Nepali-language text.

Specific Objectives

- To implement Logistic Regression, Naive Bayes, SVM, Random Forest, LSTM, BiLSTM, and CNN models for Nepali sentiment classification.
- To evaluate and compare the performance of the selected models using Accuracy, Precision, Recall, F1-score, and confusion matrices.
- To identify the comparatively best-performing machine learning and deep learning approaches for the selected Nepali sentiment dataset.

Research Questions

Some of the research questions, focusing on this study are as:

- To what extent does a machine learning algorithm predict sentiment classification in Nepali language?
- To what extent does a deep learning algorithm predict sentiment classification in Nepali language?
- Among Logistic Regression, Naive Bayes, SVM, and Random Forest, which machine learning algorithm gives the best sentiment classification?
- Among LSTM, BiLSTM, and CNN, which deep learning algorithm gives the best sentiment classification?

- What is the comparison between traditional machine learning algorithms and deep learning algorithms in Accuracy, Precision, Recall and F1-score?
- Which algorithm performs better among those assessed on the chosen Nepali sentiment dataset?

Hypotheses

If needed, the study can test the hypothesis:

Null Hypothesis (H_0)

There is no significant difference in sentiment classification performance among the selected machine learning and deep learning models when evaluated on the selected Nepali-language dataset.

Alternative Hypothesis (H_1)

There is a significant difference in sentiment classification performance among the selected machine learning and deep learning models when evaluated on the selected Nepali-language dataset.

Limitations

The study suffers from the following limitations:

- There is one set of dataset containing 35,789 examples of texts written in Nepali language available at Kaggle; therefore, the results will not be representative of all types of Nepali texts.
- Performance of the models is dependent on factors like quality, size, distribution and labeling of the chosen dataset.
- The study uses only four machine learning models—Logistic Regression, Naïve Bayes, SVM and Random Forest—and three deep learning models—LSTM, BiLSTM and CNN.
- The focus of the study is on sentiment classification using measures like Accuracy, Precision, Recall, F1-Score and confusion matrix rather than computational cost, performance on deployment and real time application.

- Differences in hyperparameter tuning, text preprocessing and hardware can have impact on the performance of the models.
- Transformer based models like BERT, multilingual BERT or transformer specific for Nepali are not included in this study.
- Spelling mistakes, informal text, code mixing, use of slang language and dialectal variations in Nepali text can affect classification performance.

Definition of Key Terms

Sentiment Analysis:

It is a process of automatic detection and classification of the emotional or opinion oriented nature of a text, whether it is positive, negative or neutral.

Natural Language Processing (NLP):

This is the study of Artificial Intelligence in which computers can interpret the human language.

Machine Learning (ML):

It is a domain of Artificial Intelligence in which computers learn patterns from data and make predictions or classifications.

Deep Learning (DL):

It is the sub-domain of machine learning in which computers learn patterns from data using multi-layered neural networks.

TF-IDF:

A numerical text representation technique that measures the importance of a word in a document relative to a collection of documents.

Logistic Regression:

Supervised machine learning classification algorithm.

Naïve Bayes:

Probabilistic machine learning classification algorithm based on Bayes' theorem and the conditional independence of features.

Support Vector Machine (SVM):

Supervised learning algorithm that tries to find the optimal decision boundary between classes.

Random Forest:

An ensemble machine learning algorithm that combines multiple decision trees to perform classification or prediction.

LSTM:

A type of recurrent neural network designed to learn long-term dependencies in sequential data and reduce problems associated with conventional recurrent networks.

BiLSTM:

A bidirectional version of LSTM that processes sequential information in both forward and backward directions.

CNN:

A neural network architecture capable of learning local patterns from data and commonly used for image and text classification tasks.

Accuracy:

The proportion of correctly classified instances out of the total number of instances.

Precision:

The proportion of correctly predicted positive instances among all instances predicted as positive.

Recall:

The proportion of correctly identified positive instances among all actual positive instances.

F1-Score:

The harmonic mean of Precision and Recall, providing a combined measure of classification performance.

Confusion Matrix:

Table to evaluate the performance of a classification algorithm through numbers of correct and incorrect classifications of instances per class.

Chapter II: Literature Review

The Sentiment Analysis, sometimes called Opinion Mining, is a crucial subfield within NLP that addresses the problem of automatic detection of opinion and attitude along with emotional orientation of the text. Depending on research goals, text can be generally separated into the positive, negative and neutral categories. The rise of social networks and online reviews made Sentiment Analysis increasingly popular as a tool for analyzing users' behavior. Sentiment classification methods used earlier mainly utilized classical machine learning approaches and hand-crafted features. Popular approaches included Naïve Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) with Bag-of-Words and Term Frequency-Inverse Document Frequency (TF-IDF) being generally used to represent text. Such approaches were highly computationally-efficient and still remain relevant especially when dataset size is not too big and computation power is low. Nevertheless, efficiency of such approaches can suffer from inability to capture the order of words in sentences along with contextual meaning and long-range dependencies.

Regarding the Nepali language, an interesting research by Regmi et al. (2017) about the classification of facts and opinions in Nepali subjective texts was performed. The researchers compared several supervised machine learning techniques including Logistic Regression, Multinomial Naïve Bayes, and Support Vector Machine by using the TF-IDF and n-grams as features. The research proved that supervised machine learning techniques can be applied for the Nepali text classification successfully and yielded good results near to 70%. This research is especially important for this research since it used the same three major machine learning techniques that will be analyzed in comparison i.e. Naïve Bayes, SVM, and Logistic Regression.

Furthermore, Tamrakar et al. (2020) examined the aspect-based sentiment analysis of Nepali text using Support Vector Machine and Naïve Bayes. The research included Part-of-Speech tagging and feature extraction to extract aspects and sentiments from Nepali texts. The study highlighted the applicability of traditional machine learning techniques in Nepali sentiment-related tasks, while also showing the challenges associated with feature extraction and linguistic processing in the Nepali language. This work demonstrates that the performance of machine learning models can be strongly influenced by the quality of preprocessing and feature representation.

A more comprehensive study was conducted by Piryani et al. (2020), who explored machine learning, lexicon-based, and deep learning approaches for sentiment analysis of Nepali tweets.

Their study examined conventional machine learning models, including Multinomial Naïve Bayes, Decision Tree, Support Vector Machine, and Logistic Regression. In addition, deep learning models such as CNN, LSTM, and CNN-LSTM were investigated. The researchers also developed Nepali sentiment resources, including Nepali SentiWordNet and Nepali SenticNet. Their findings indicated that deep learning approaches generally performed better than conventional machine learning approaches and the proposed lexicon-based method. This study is highly relevant to the present research because it directly compares traditional and deep learning approaches for Nepali sentiment analysis.

While traditional machine learning approaches depend heavily on manually designed features, recent research has increasingly focused on deep learning methods. Zhang et al. (2018) provided a comprehensive survey of deep learning approaches for sentiment analysis and discussed the application of CNN, RNN, LSTM, GRU, and other neural architectures at document, sentence, and aspect levels. According to their review, it has been found that there are certain models that could learn features from textual data without manually engineered features. The effectiveness of deep learning models in sentiment analysis depends on several issues such as dataset sizes, language properties, and computing power. One of the most popular methods among deep learning methods is the application of Long Short-Term Memory (LSTM) networks which have proved to be efficient for sentiment analysis since they are able to work with sequential information and long-range dependencies. While machine learning methods usually treat words as separate features, LSTM works sequentially and retains previous information. A recent review of sentiment analysis methods has also emphasized the importance of recurrent neural architectures in understanding contextual relationships and long textual sequences.

Dang et al. (2020) conducted a comparative study of deep learning methods for sentiment analysis using different text representation techniques, including TF-IDF and word embeddings. Their research demonstrated the importance of selecting appropriate input representations when evaluating sentiment classification models. The study is relevant to the present research because it supports the comparison of traditional feature extraction methods, such as TF-IDF, with sequence-based representations used in deep learning models.

The development of Bidirectional LSTM (BiLSTM) further improved sequential text analysis by processing information in both forward and backward directions. This enables the model to consider contextual information from both preceding and following words. Zhou and Long (2018)

investigated sentiment analysis using CNN and BiLSTM architectures, demonstrating the usefulness of combining different deep learning methods for capturing both local textual patterns and sequential contextual information. CNN is generally effective in identifying important local features or word patterns, whereas BiLSTM is useful for understanding contextual dependencies in both directions. It is also necessary to mention the use of Convolutional Neural Networks (CNN) for text classification. Despite the fact that CNN was designed for image processing, it can be used for NLP purposes, including sentiment analysis. CNN can identify local patterns in text through convolution filters and pooling operations. Recent reviews have identified CNN and LSTM as among the most widely used deep learning architectures for sentiment classification. The combination of these methods has also been explored to benefit from both local feature extraction and sequential context learning.

For Nepali language sentiment analysis specifically, Sitaula et al. (2021) conducted a significant study on sentiment classification of Nepali COVID-19-related tweets. The researchers created the NepCOV19Tweets dataset containing positive, negative, and neutral sentiment categories. They proposed different feature representation methods and CNN-based models, including an ensemble CNN approach. Their study demonstrated that deep learning could be effectively applied to Nepali social media data and also contributed a benchmark dataset for future research. This research is particularly important because Nepali is considered a comparatively low-resource language, where the availability of large labelled datasets and language-processing resources remains limited. It should be mentioned that low-resource languages face a number of problems in sentiment analysis different from those of English. These difficulties include small amounts of labeled datasets, absence of language lexicons, morphology, orthographic variations, and pre-trained language models. A recent review about low-resource sentiment analysis stated that choosing appropriate datasets and learning models is essential.

A cross-lingual survey by Mäntylä et al. (2023) also highlighted the growing importance of machine learning and deep learning approaches across different languages. The study discussed the transition from traditional feature-based machine learning models toward neural and multilingual approaches. Although modern transformer-based models have achieved strong results in many NLP tasks, traditional machine learning and comparatively simpler deep learning models remain important, especially for educational research and low-resource environments where computational resources may be limited. Recent research has continued to improve Nepali

sentiment analysis through more advanced hybrid models. Sitoula et al. (2025) proposed a hybrid deep learning approach for Nepali social media sentiment analysis using contextual representations from XLM-RoBERTa together with convolutional layers for local feature extraction. The authors tested their approach using different Nepalese sentiment analysis datasets, and achieved better performance than the various existing techniques. These results imply that using contextual language representations and hybrid architectures may be good avenues to investigate in future work on Nepali NLP tasks. At the same time, the study highlights continuing challenges related to Nepali morphology, short social media text, dataset imbalance, and limited language resources.

Recent systematic reviews also confirm the continued importance of comparing traditional and deep learning approaches. Islam et al. (2024) presented a review of advances in sentiment analysis utilizing deep learning and addressed various architectures like CNN, LSTM, BiLSTM, GRU, attention models, and hybrid architectures. The paper provided an overview of important challenges like the quality of datasets, data preprocessing, computational issues, and explainability. In addition, another systematic literature review article written in 2024 highlighted the importance of both traditional and deep learning ML approaches, as per the properties of the specific dataset and research problem.

Generally, the reviewed literature demonstrates how there have been advances in sentiment analysis over time starting from traditional machine learning models to deep learning and contextually aware models. Traditional methods such as Naïve Bayes, SVM, and Logistic Regression are still important owing to their simplicity, computational efficiency, and effectiveness with TF-IDF features. On the contrary, the deep learning models like LSTM, BiLSTM, and CNN can learn from the text patterns and sequential information which may not be efficiently captured by the conventional techniques. But then again, even after several advances in research, there is less literature on the sentiment analysis of the Nepali language. In the existing studies, different data sets, different domains, different pre-processing techniques, and different evaluation processes have been used to carry out research work. Hence, there is an urgent need for comparative research in this field using the same data set and the same evaluation processes for different machine learning and deep learning algorithms. This research fills the gap in the field of literature by carrying out the comparison of Naïve Bayes, Support Vector Machine, and Logistic Regression with LSTM, BiLSTM, and CNN for Nepali language sentiment classification.

Chapter III: Research Methodology

This chapter describes the methodology used to conduct the research entitled *"Comparative Analysis of Nepali Language Sentiment Classification Using Machine Learning and Deep Learning Approaches."* It describes the research design, the data sources, sampling method, data collection technique, data preprocessing technique, experimental method, machine learning/deep learning algorithms, data analysis techniques, and evaluation criteria in this study. This research applies a quantitative and experimental research method, which is suitable for this kind of comparison between different algorithms under similar conditions (Creswell & Creswell, 2018).

The primary objective of this study is to evaluate the efficiency of selected Machine Learning (ML) and Deep Learning (DL) algorithms in sentiment classification in the Nepali language. Sentiment analysis can be defined as the automatic identification of emotional orientation in texts. Therefore, the preprocessing of the text, the selection of the algorithm, and the feature representation play a critical role in this field, especially for the language like Nepali. The study used an experimental and comparative method. A publicly available Nepali sentiment dataset was collected from Kaggle and processed through several stages, including duplicate removal, text preprocessing, feature extraction, model training, testing, and performance comparison. Previous research on Nepali sentiment classification has similarly demonstrated the importance of appropriate text representation and systematic evaluation when applying deep learning models to Nepali textual data (Sitaula et al., 2021).

Research Design

This study adopted a quantitative, experimental, and comparative research design**.**

The research is quantitative because the performance of different sentiment classification models was measured using numerical evaluation metrics. The experimental approach was used because different machine learning and deep learning algorithms were trained and tested on the Nepali language dataset under a defined experimental procedure. The study is comparative in nature because the performance of traditional Machine Learning algorithms was compared with Deep Learning models. The Machine Learning models included:

- *Naive Bayes*
- *Support Vector Machine (SVM)*
- *Logistic Regression*

Similarly, the Deep Learning models included:

- *Long Short-Term Memory (LSTM)*
- *Bidirectional Long Short-Term Memory (BiLSTM)*
- *Convolutional Neural Network (CNN)*

The performances of these models were evaluated and compared to find out which model is more efficient in Nepali language sentiment classification.

Research Approach and Workflow

The research followed a systematic workflow beginning with the collection of the dataset and ending with the final comparison between Machine Learning and Deep Learning approaches. The overall workflow of the research is presented below:

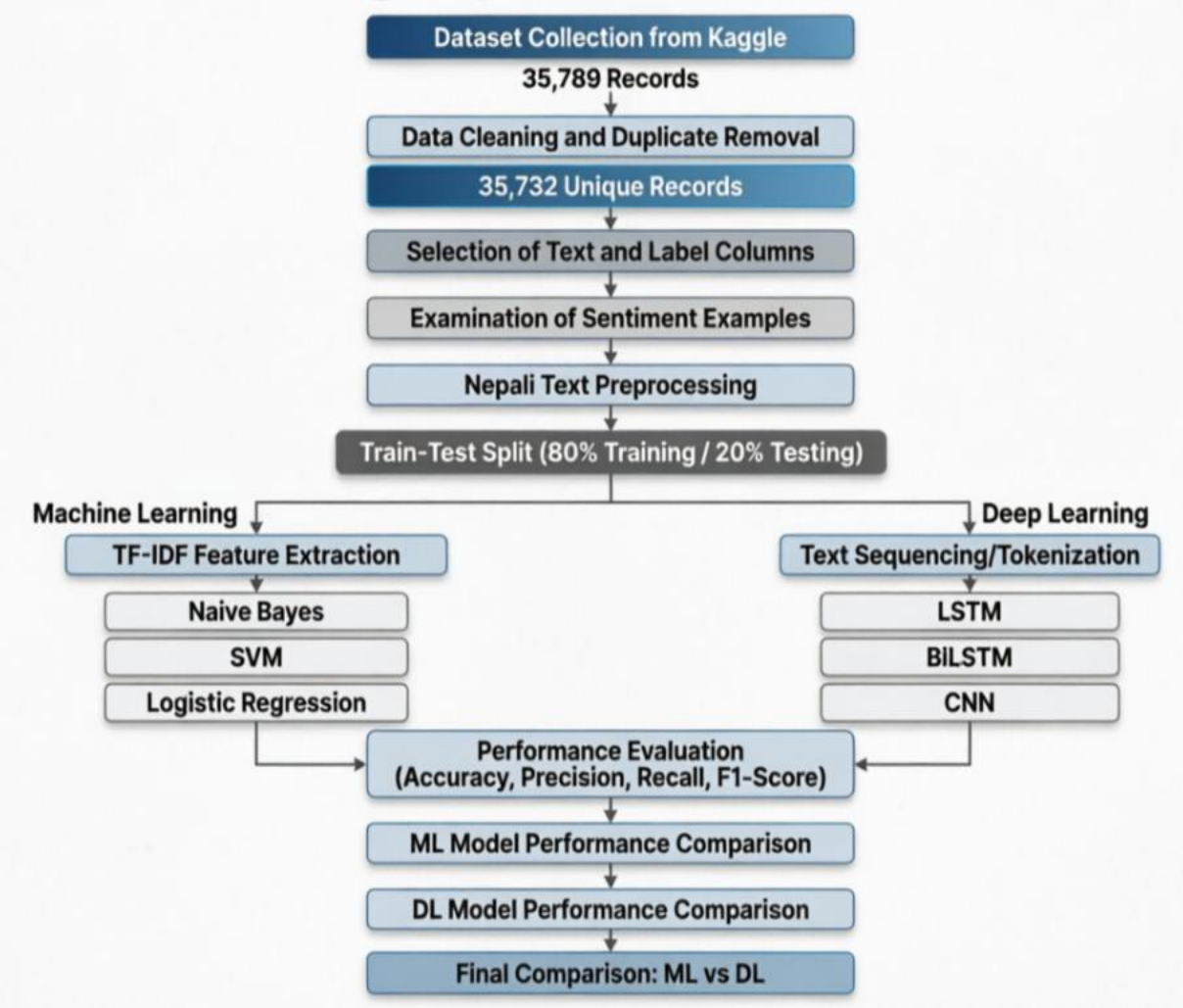


Figure 1: Research Approach and WorkFlow

This workflow ensured that the same dataset was systematically prepared and analyzed to allow a meaningful comparison among different sentiment classification techniques.

Population and Sampling Procedure

The population of this study consisted of Nepali language textual records available in the selected sentiment dataset. The total number of texts that were gathered amounted to 35,789 texts. As for the methodology of the research, since it was based on the secondary data, there was no need to involve people. Thus, no questionnaires, interviews, surveys, or Google Forms were used.

The collected dataset was examined for duplicate records. After removing duplicate entries, 35,732 unique samples remained and were used for further preprocessing, training, testing, and analysis.

Thus, the study used a dataset-based sampling approach**,** where the available valid and unique records were considered for the experimental analysis.

Table 1: Population Sampling

Description	**Number of Records**
Total Records Collected	35789
Duplicate Records Removed	57
Final Unique Records Used	35732

The use of a relatively large dataset helped provide sufficient textual data for training and evaluating both Machine Learning and Deep Learning models.

Source and Collection of Data

The research was conducted with the help of secondary data extracted from a publicly available dataset on Kaggle. It was provided in the CSV (Comma-Separated Values) format. The dataset contained Nepali language textual information along with corresponding sentiment labels. The required data were imported into the research environment and examined to identify the main variables required for sentiment classification. The primary columns used in the study were:

- ***Text:*** *Contains Nepali language textual content.*
- ***Label:*** *The sentiment class corresponding to the text.*

In the course of conducting the research and since it was based on the secondary data, there was no need to use any data collection tools such as questionnaires, interviews, observations, or Google Forms. The gathered CSV file contained 35,789 entries. However, during the data cleaning stage, the duplicates were discovered and eliminated, thus leaving only 35,732 entries for further analysis.

Data Preparation and Cleaning

Data preparation was performed before model development to improve the quality and consistency of the dataset. The following major activities were performed:

- Duplicate records were identified within the collected dataset and removed. Thus, the original dataset contained 35,789 entries while the cleaned dataset contained 35,732

unique entries. The elimination of duplicates was necessary because repeated entries may affect the learning process and produce biased results.

- The dataset was examined to identify the variables required for sentiment classification. The two primary variables used in the research were Nepali text and Corresponding sentiment label. These variables were used as input and output data for developing the sentiment classification models.
- Sample records from different sentiment categories were examined manually to understand the nature and structure of the Nepali text and sentiment labels.This initial examination helped verify whether the dataset was suitable for the sentiment classification task and provided an understanding of the linguistic characteristics of Nepali textual data.

Nepali Text Preprocessing

Text preprocessing is one of the crucial steps in NLP as the raw text data can have some unwanted characters, improper formatting, etc., which may hinder the efficiency of the model. The raw text data was preprocessed and cleaned for the purpose of training the machine learning and deep learning models using the Nepali language text. Depending on the requirements of the models, preprocessing activities included:

- *Removing duplicate records*
- *Handling missing or empty text, where applicable*
- *Removing unnecessary spaces*
- *Cleaning unwanted characters and symbols*
- *Standardizing the textual format*
- *Preparation of the Nepali text for feature extraction and training the model*

As part of the preprocessing step, it was ensured to keep noise as low as possible by improving the input data. Particular emphasis was laid on retaining all the meaningful content of the Nepali language as excessive preprocessing may lead to the removal of all the necessary context for sentiment classification.

Training and Testing Data Split

Upon completion of the preprocessing step, the cleaned data was split in such a way that 80% of the data became the training set while the remaining 20% became the testing set. The training data set was then used to train the selected machine learning and deep learning models, helping them learn patterns related to various sentiment categories. The remaining 20% of the data was kept for testing the models.

The testing dataset contained previously unseen Nepali text and was used to evaluate how effectively the trained models could classify sentiment. Using the same 80/20 train-test split for all the selected models helped ensure a fair and consistent comparison between the Machine Learning and Deep Learning approaches. This type of data splitting is commonly used in machine learning experiments to train models and evaluate their ability to generalize to unseen data (Géron, 2022).

Feature Extraction for Machine Learning Models

However, Traditional Machine Learning models cannot directly process raw text data. Hence, the Nepali text data was transformed into numeric features using the TF-IDF approach. TF-IDF scores numeric values of words depending on their frequency in the document and in the whole dataset. The obtained TF-IDF features are used as the input feature sets for the following Machine Learning models: *Naive Bayes, Support Vector Machine (SVM) and Logistic Regression*

This process transformed the textual data into a numerical format suitable for Machine Learning classification.

Machine Learning Models

Three Machine Learning algorithms were selected for the comparative analysis.

- Naive Bayes is a probabilistic classification algorithm based on probability theory. It is widely used in text classification because of its simplicity and computational efficiency. In this study, Naive Bayes was trained using the TF-IDF representation of the Nepali text and used to predict sentiment categories.
- Support Vector Machine (SVM) is a popular Supervised Machine Learning algorithm used in the context of classification tasks. SVM tries to find an optimal decision

boundary which separates various classes. In our research, SVM is applied to the TF-IDF features extracted from the Nepali text data.

- Logistic Regression is a widely used classification algorithm which predicts the probability of the input belonging to some specific class. The TF-IDF features extracted from the Nepali text data are used as the inputs of the Logistic Regression model.

Deep Learning Models

Deep Learning models were used to investigate whether neural network-based approaches could provide improved performance for Nepali language sentiment classification. The LSTM, BiLSTM, and CNN models were chosen:

- Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). Since textual data consists of sequences of words or tokens, LSTM can capture relationships between different parts of a sentence. In this research, the LSTM model was used to learn patterns associated with sentiment in Nepali text.
- Bi-directional Long Short-Term Memory (BiLSTM) is an advanced version of the traditional LSTM algorithm. Unlike the conventional LSTM algorithm, BiLSTM analyzes the input sequentially in two opposite directions. This allows the model to use contextual information from both earlier and later parts of a text sequence. The BiLSTM model was included to investigate whether bidirectional contextual learning improves Nepali sentiment classification performance.
- Convolutional Neural Network (CNN) CNN is commonly associated with image processing; however, it is also widely used for text classification. In text classification, CNN can identify important local patterns and features within sequences of words. The CNN model was trained and evaluated for Nepali language sentiment classification and compared with LSTM and BiLSTM models.

Experimental Procedure

The experimental procedure was conducted in two major phases.

Phase I: Machine Learning Experiment

- *The cleaned Nepali text dataset was prepared.*

- *The dataset was split into training and testing parts.*
- *TF-IDF was applied for converting textual information to numerical attributes.*
- *Naive Bayes was trained and evaluated.*
- *SVM was trained and evaluated.*
- *Logistic Regression was trained and evaluated.*
- *The performance of the three Machine Learning models was compared.*

Phase II: Deep Learning Experiment

- *The preprocessed Nepali text was prepared for Deep Learning models.*
- *The dataset was split into training and testing parts.*
- *The Deep Learning models were trained.*
- *LSTM was evaluated.*
- *BiLSTM was evaluated.*
- *CNN was evaluated.*
- *The performance of the Deep Learning models was compared.*

Finally, the best-performing Machine Learning model and Deep Learning model were compared to determine the relative effectiveness of traditional and neural network-based approaches for Nepali language sentiment classification.

Data Analysis Techniques

The data analysis was primarily performed using a comparative performance analysis approach. The results generated by different Machine Learning and Deep Learning models were analyzed and compared using classification evaluation metrics. The important performance metrics considered are Accuracy, Precision, Recall, F1-Score and Confusion Matrix.

- Accuracy refers to the percentage of samples which were classified correctly in relation to the total number of samples. The greater the value of accuracy, the better the model classifies the sentiments.
- Precision measures how many of the samples predicted as belonging to a particular sentiment class were actually correct. It is useful for evaluating the reliability of positive predictions made by the model.

- Recall measures the ability of a model to correctly identify relevant samples belonging to a particular class. A higher recall value indicates that fewer relevant sentiment samples were missed by the classification model.
- The F1-score provides a balance between precision and recall. It is particularly useful when comparing classification models because it considers both false positive and false negative predictions.
- A confusion matrix may also be used to examine the classification performance of the models in detail. It provides information about correctly and incorrectly predicted sentiment categories and helps identify patterns of misclassification.

Comparative Analysis

The research conducted comparisons at three levels. Comparison among Machine Learning Models, Comparison among Deep Learning Models and Final Comparison between Machine Learning and Deep Learning.

- In Comparison among Machine Learning Models, the performance of the Naïve Bayes, Support Vector Machine and Logistic Regression models was compared. The model with the best overall evaluation results was identified as the best-performing Machine Learning approach.
- In Comparison among Deep Learning Models, the performance of the LSTM, BiLSTM and CNN models was compared. The best-performing Deep Learning model was identified based on the selected evaluation metrics.
- In Final Comparison between Machine Learning and Deep Learning, the best Machine Learning model and the best Deep Learning model were compared. This final comparison was conducted to answer the main research objective: whether traditional Machine Learning or Deep Learning approaches provide better performance for Nepali language sentiment classification using the selected dataset and experimental procedure.

Research Tools and Technologies

The research was conducted using programming and data analysis tools suitable for Natural Language Processing and Machine Learning experiments. The major technologies used included:

- *Python programming language*
- *CSV dataset*
- *Natural Language Processing techniques*
- *TF-IDF vectorization*
- *Machine Learning algorithms*
- *Deep Learning frameworks and neural network models*

Python was used because it provides a wide range of libraries and tools for data preprocessing, Machine Learning, Deep Learning, and performance evaluation.

Reliability and Validity of the Experiment

To increase the validity of the experiment's results, the same clean data set and preprocessing methodology was applied across the entire experiment. The models' performance was measured by standard classification metrics: accuracy, precision, recall, and F1-score.

Using several machine learning and deep learning models for comparison increased the validity of the research since the conclusions were made according to the performance of different models rather than one. However, the results of the study are dependent on the characteristics, quality, size, and sentiment distribution of the selected Kaggle dataset.

Ethical Considerations

This study used publicly available secondary data obtained from Kaggle. No direct interaction with human participants was conducted. Therefore:

- *No interviews were conducted.*
- *No questionnaires were distributed.*
- *No Google Forms were used.*
- *No personal information was directly collected from participants for this research.*
- *The dataset was used only for academic and research purposes.*

Chapter IV: Implementation and Experimental Setup

This chapter presents the practical implementation of the proposed Nepali language sentiment classification system. The implementation was carried out using Python in a suitable development environment such as Google Colab or Jupyter Notebook. The complete implementation process includes dataset loading, data inspection, data cleaning, duplicate removal, text preprocessing, feature extraction, model training, testing, and comparison. The implementation was divided into three major stages. The first stage involved preparing and preprocessing the dataset. The second stage involved implementing Machine Learning and Deep Learning models. The final stage involved evaluating and comparing the performance of the implemented models.

Implementation Environment

Python, along with common libraries used for data manipulation, Machine Learning, Deep Learning, and visualization, was used to conduct this experiment.

Table 2: Implementation Environment

Component	**Tools / Library**
Programming Language	Python
Environment	Google Colab / Jupyter Notebook
Data Processing	Pandas, NumPy
Machine Learning	Scikit-learn
Deep Learning	TensorFlow / Keras
Visualization	Matplotlib
Dataset Format	CSV

Dataset Loading and Initial Inspection

The following steps were conducted to complete the comparison of data models. It includes steps from uploading csv data to preprocessing to creation of confusion matrix and then result analysis using different parameters such as accuracy, precision and others.

Uploading CSV file and check sentiment labels

The first step of the implementation was uploading the sentiment_analysis_nepali_final.csv file into the working environment. After uploading the dataset, it was loaded using Python and examined to understand its overall structure. The column names were checked to identify the Nepali text data and their corresponding sentiment labels. The number of rows and columns, data types, and general information about the dataset were also examined before proceeding with further implementation.

from google.colab import files

uploaded = files.upload()

```
import pandas as pd

df = pd.read_csv("sentiment_analysis_nepali_final.csv")
```

```
df.head()
```

	Unnamed: 0	Sentences	Sentiment
0	0	म एक शिक्षक , शिक्षा क्षेत्रमा रमाएको मान्छे ।...	1
1	1	म सरकारी स्कूल/कलेजमा पढेर करीब १२ बर्ष भन्दा ...	1
2	2	कति राम्रो शिव मन्दिर देख्न पाइयो कुन ठाउको हो...	1
3	3	मारुनी भन्ने वितिकै सामान्य नाचनीमा आधारित कथा...	1
4	4	यो फ्लिम हेरिसकेपछी थाहा भयो कि किन दर्सकहरुले...	1

```
print(df.columns)
```

```
Index(['Unnamed: 0', 'Sentences', 'Sentiment'], dtype='object')
```

```
df.shape
```

```
(35789, 3)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35789 entries, 0 to 35788
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  35789 non-null  int64
 1   Sentences   35789 non-null  object
 2   Sentiment   35789 non-null  int64
dtypes: int64(2), object(1)
memory usage: 838.9+ KB
```

```
df['Sentiment'].value_counts()
```

Sentiment	count
1	15880
-1	14408
0	5501

dtype: int64

Figure 2: Uploading CSV Files & dataset Entries

Preprocessing dataset

After analyzing the data set, data preprocessing was done to make sure that the data is ready for sentiment classification. It included identifying any missing value in the data set, sentiment labels, removal of duplicate entries, and selection of required columns for conducting the experiment. The text data and sentiment labels were then prepared for further processing and model implementation. These preprocessing steps helped ensure that the dataset was properly organized and suitable for training and testing the Machine Learning and Deep Learning models.

Checking missing values:

```
df.isnull().sum()
```

	0
Unnamed: 0	0
Sentences	0
Sentiment	0

dtype: int64

Checking duplicate values. Total 57 sentences were duplicated.

```
df['Sentences'].duplicated().sum()
```

```
np.int64(57)
```

```
[ ]  df = df.drop_duplicates(subset=['Sentences'])
```

Checking total unique records. Total 35,732 unique records.

```
df.shape
```

```
(35732, 2)
```

After removing unnecessary columns, the total records as seen as:

df = df.drop(columns=['Unnamed: 0'])

```
df.head()
```

	Sentences	Sentiment
0	म एक शिक्षक , शिक्षा क्षेत्रमा रमाएको मान्छे ।...	1
1	म सरकारी स्कूल/कलेजमा पढेर करीब १२ बर्ष भन्दा ...	1
2	कति राम्रो शिव मन्दिर देख्न पाइयो कुन ठाउको हो...	1
3	मारुनी भन्ने वितिकै सामान्य नाचनीमा आधारित कथा...	1
4	यो फ्लिम हेरिसकेपछी थाहा भयो कि किन दर्सकहरुले...	1

After renaming the columns, the dataset is seen as:

```
df = df.rename(columns={
    'Sentences': 'text',
    'Sentiment': 'label'
})
```

```
df[df['label'] == -1]['text'].head(5)
```

	text
5	उपनियाशमा चलचित्र बनाउदा सबैको बुचार गरेर बनाक...
6	नेपालमा बनेको अहिले सम्मको सबै भन्दा खाते फिल...
7	कठै बिचारा कस्तो निर्देशक कस्ता कलाकार, इतिहास...
13	कथा हरु मा केही ठिक भए पनि कलाकार पात्र हरु कु...
14	खै के भनु फिल्म त राम्रो लागेन , अलि राम्रो कल...

dtype: object

```
df[df['label'] == 0]['text'].head(5)
```

	text
574	खरीद गर्नेछु
577	ठिकै उत्पादन हो
587	मेरा बच्चाहरू
603	उहाँसँग यो फोनको तुलनामा ठूलो स्क्रिन छ
610	हल्का वजन

dtype: object

```
df[df['label'] == 1]['text'].head(5)
```

	text
0	म एक शिक्षक , शिक्षा क्षेत्रमा रमाएको मान्छे ।...
1	म सरकारी स्कूल/कलेजमा पढेर करीब १२ बर्ष भन्दा ...
2	कति राम्रो शिव मन्दिर देख्न पाइयो कुन ठाउको हो...
3	मारुनी भन्ने वितिकै सामान्य नाचनीमा आधारित कथा...
4	यो फ्लिम हेरिसकेपछी थाहा भयो कि किन दर्सकहरुले...

dtype: object

Figure 3: Data Preprocessing

After data preprocessing, the cleaned data set of 35,732 rows was split into training and testing sets in the ratio of 80:20. The training set was used for training Machine Learning and Deep Learning models, while the testing set was used for evaluating their performance on previously unseen data. After split, the sentiment label distribution in both training and testing data sets was checked to confirm the appropriate representation of sentiment classes in both the data sets.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.20,
    random_state=42,
    stratify=y
)
```

```
[ ]
print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))

Training samples: 28585
Testing samples: 7147
```

```
[ ]
print("\nTraining labels:")
print(y_train.value_counts())

print("\nTesting labels:")
print(y_test.value_counts())

...
Training labels:
label
 1    12685
-1    11514
 0     4386
Name: count, dtype: int64

Testing labels:
label
 1    3172
-1    2879
 0    1096
Name: count, dtype: int64
```

Figure 4: Separating Testing and Training Dataset

Machine Learning Implementation

TF-IDF vectorization

Next, TF-IDF vectorization was done on the Nepali text data to convert it into numerical data suitable for machine learning models. The maximum number of features was restricted to 10,000 using max_features=10000. The TF-IDF vectorizer was trained using the training data set, while the testing data set was transformed in the same way. The generated TF-IDF features were verified to make sure that the numerical representation of the text data is correct before training the Machine Learning models.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(
    max_features=10000,
    ngram_range=(1, 2)
)
```

```
X_train_tfidf = tfidf.fit_transform(X_train)
```

```
X_test_tfidf = tfidf.transform(X_test)
```

Transforming the test data.

```
print("Training TF-IDF shape:", X_train_tfidf.shape)
print("Testing TF-IDF shape:", X_test_tfidf.shape)
```

```
Training TF-IDF shape: (28585, 10000)
Testing TF-IDF shape: (7147, 10000)
```

Displaying first 50 features learned from the dataset.

```
print(tfidf.get_feature_names_out()[:50])
```

```
['अक' 'अक जन' 'अक जनक' 'अक बर' 'अक सफ' 'अक सफर' 'अकर' 'अकर मण' 'अकल'
 'अकल पन' 'अख' 'अग' 'अग बढ' 'अग रप' 'अग रपङ' 'अग रभ' 'अग रम' 'अग रस'
 'अग रह' 'अगष' 'अगस' 'अघ' 'अघ पछ' 'अघ बढ' 'अघ वर' 'अघ सम' 'अघ सर' 'अङ'
 'अच' 'अच नक' 'अचम' 'अछ' 'अछ मक' 'अछ रह' 'अज' 'अझ' 'अझ एक' 'अझ कड'
 'अझ नभएक' 'अझ पन' 'अझ बढ' 'अझ सक' 'अझ सम' 'अञ' 'अञ चल' 'अट' 'अड' 'अण'
 'अण अध' 'अत']
```

Figure 5: TF-IDF Vectorization from dataset

Implementation of Logistic Regression

The next step is applying the Logistic Regression model to the TF-IDF transformed training data set. The model was trained using the training features and their corresponding sentiment labels, and predictions were then generated using the transformed testing dataset. A confusion matrix was generated to examine the model's correct and incorrect classifications. Finally, different evaluation parameters, including Accuracy, Precision, Recall, and F1-Score, were calculated to analyze the overall performance of the Logistic Regression model.

```
from sklearn.linear_model import LogisticRegression
```

```
model_lr = LogisticRegression(
    max_iter=1000,
    random_state=42
)
```

```
y_pred_lr = model_lr.predict(X_test_tfidf)
```

```
model_lr = LogisticRegression(
    max_iter=1000,
    random_state=42
)
```

```
model_lr.fit(X_train_tfidf, y_train)
```

```
LogisticRegression(max_iter=1000, random_state=42)
```

```
print("Actual labels:")
print(y_test.values[:20])

print("\nPredicted labels:")
print(y_pred_lr[:20])
```

```
Actual labels:
[ 0  0  1  1  0  0 -1 -1  0  1  1  1  0  0  0  1  1  1 -1  1]

Predicted labels:
[ 1  0 -1  1 -1 -1 -1 -1  1 -1 -1  1 -1 -1  1  1 -1  1  1  1]
```

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred_lr)

print("Accuracy:", accuracy)
print("Accuracy (%):", accuracy * 100)
```

```
Accuracy: 0.674268924024066
Accuracy (%): 67.42689240240661
```

Nepali_Sentiment_Analysis.ipynb

File Edit View Insert Runtime Tools Help

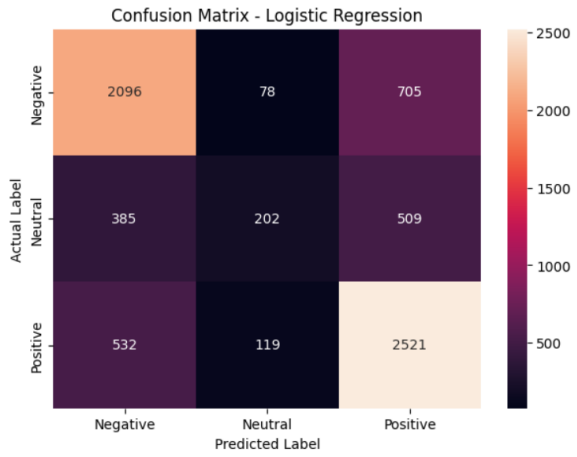
ommands + Code + Text ▷ Run all

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred_lr, labels=[-1, 0, 1])

plt.figure(figsize=(7, 5))
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    xticklabels=['Negative', 'Neutral', 'Positive'],
    yticklabels=['Negative', 'Neutral', 'Positive']
)

plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```



Start coding or generate with AI.

```
from sklearn.metrics import classification_report

print(classification_report(
    y_test,
    y_pred_lr,
    labels=[-1, 0, 1],
    target_names=['Negative', 'Neutral', 'Positive']
))
```

```
              precision    recall  f1-score   support

    Negative       0.70      0.73      0.71      2879
     Neutral       0.51      0.18      0.27      1096
    Positive       0.67      0.79      0.73      3172

    accuracy                           0.67      7147
   macro avg       0.63      0.57      0.57      7147
weighted avg       0.66      0.67      0.65      7147
```

Figure 6: Result Parameters from Logistic Regression

Implement Naïve Bayes

The Naïve Bayes classifier was trained using the extracted TF-IDF features.

```
from sklearn.naive_bayes import MultinomialNB

nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)

nb_prediction = nb_model.predict(X_test_tfidf)
```

Implement Support Vector Machine

```
from sklearn.svm import LinearSVC

svm_model = LinearSVC()
svm_model.fit(X_train_tfidf, y_train)

svm_prediction = svm_model.predict(X_test_tfidf)
```

Deep Learning Implementation

To prepare text for deep learning, a tokenizer was used to convert words into numerical representations.

```
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

Since sentences have different lengths, padding was applied to make the input sequences equal in length.

```
X_train_pad = pad_sequences(
    X_train_seq,
    maxlen=100
)

X_test_pad = pad_sequences(
    X_test_seq,
    maxlen=100
)
```

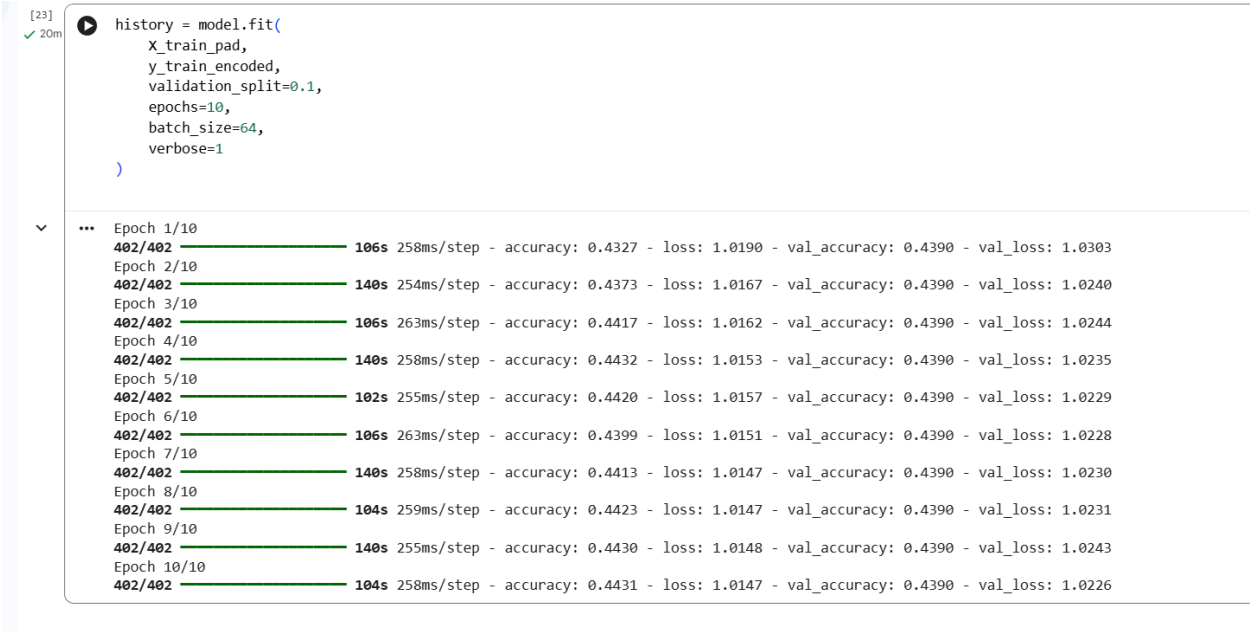
LSTM Implementation

For the Deep Learning approach, an LSTM model was first built to process the sequential Nepali text data. Then, the model is compiled using an appropriate optimizer, loss function, and evaluation metric. Then, the compiled LSTM model was trained using the prepared training data set.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

lstm_model = Sequential([
    Embedding(20000, 128),
    LSTM(64),
    Dense(1, activation='sigmoid')
])
```

```
lstm_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```



```
[23]
20m
history = model.fit(
    X_train_pad,
    y_train_encoded,
    validation_split=0.1,
    epochs=10,
    batch_size=64,
    verbose=1
)

Epoch 1/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 258ms/step - accuracy: 0.4327 - loss: 1.0190 - val_accuracy: 0.4390 - val_loss: 1.0303
Epoch 2/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 254ms/step - accuracy: 0.4373 - loss: 1.0167 - val_accuracy: 0.4390 - val_loss: 1.0240
Epoch 3/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 263ms/step - accuracy: 0.4417 - loss: 1.0162 - val_accuracy: 0.4390 - val_loss: 1.0244
Epoch 4/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 258ms/step - accuracy: 0.4432 - loss: 1.0153 - val_accuracy: 0.4390 - val_loss: 1.0235
Epoch 5/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 102s 255ms/step - accuracy: 0.4420 - loss: 1.0157 - val_accuracy: 0.4390 - val_loss: 1.0229
Epoch 6/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 263ms/step - accuracy: 0.4399 - loss: 1.0151 - val_accuracy: 0.4390 - val_loss: 1.0228
Epoch 7/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 258ms/step - accuracy: 0.4413 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0230
Epoch 8/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 104s 259ms/step - accuracy: 0.4423 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0231
Epoch 9/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 255ms/step - accuracy: 0.4430 - loss: 1.0148 - val_accuracy: 0.4390 - val_loss: 1.0243
Epoch 10/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 104s 258ms/step - accuracy: 0.4431 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0226
```

BiLSTM Implementation

For the Deep Learning approach, a Bidirectional Long Short-Term Memory (BiLSTM) model was implemented to process the Nepali text sequences in both forward and backward directions. The

model was built and compiled by defining the appropriate optimizer, loss function, and evaluation metrics. It was then trained using the prepared training dataset, allowing the model to learn contextual information from the text for sentiment classification and further performance evaluation.

```
from tensorflow.keras.layers import Bidirectional

bilstm_model = Sequential([
    Embedding(20000, 128),
    Bidirectional(LSTM(64)),
    Dense(1, activation='sigmoid')
])
```

[47] ✓ 21s

```
# Step 10 – Evaluate BiLSTM on the test set
# Now use the same 7,147 unseen test samples:
bilstm_test_loss, bilstm_test_accuracy = bilstm_model.evaluate(
    X_test_pad,
    y_test_encoded,
    verbose=1
)
print("BiLSTM Test Loss:", bilstm_test_loss)
print("BiLSTM Test Accuracy:", bilstm_test_accuracy)
print(
    "BiLSTM Test Accuracy (%):",
    bilstm_test_accuracy * 100
)
```

```
224/224 ━━━━━━━━━━━━━━━━━━━━ 21s 96ms/step - accuracy: 0.6736 - loss: 1.6965
BiLSTM Test Loss: 1.6965041160583496
BiLSTM Test Accuracy: 0.6735693216323853
BiLSTM Test Accuracy (%): 67.35693216323853
```

```
[42]
✓ 32m
# Step 7 – Train the BiLSTM

bilstm_history = bilstm_model.fit(
    X_train_pad,
    y_train_encoded,
    validation_split=0.1,
    epochs=10,
    batch_size=64,
    verbose=1
)
```

```
Epoch 1/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 197s 481ms/step - accuracy: 0.6482 - loss: 0.8004 - val_accuracy: 0.6922 - val_loss: 0.7326
Epoch 2/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 188s 468ms/step - accuracy: 0.7787 - loss: 0.5626 - val_accuracy: 0.7097 - val_loss: 0.7346
Epoch 3/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 201s 466ms/step - accuracy: 0.8533 - loss: 0.4025 - val_accuracy: 0.7044 - val_loss: 0.8685
Epoch 4/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 202s 465ms/step - accuracy: 0.9013 - loss: 0.2771 - val_accuracy: 0.6835 - val_loss: 0.9364
Epoch 5/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 187s 466ms/step - accuracy: 0.9326 - loss: 0.1949 - val_accuracy: 0.6866 - val_loss: 1.1110
Epoch 6/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 189s 471ms/step - accuracy: 0.9543 - loss: 0.1383 - val_accuracy: 0.6894 - val_loss: 1.2309
Epoch 7/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 187s 465ms/step - accuracy: 0.9651 - loss: 0.1053 - val_accuracy: 0.6870 - val_loss: 1.3698
Epoch 8/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 184s 459ms/step - accuracy: 0.9730 - loss: 0.0839 - val_accuracy: 0.6772 - val_loss: 1.5778
Epoch 9/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 190s 473ms/step - accuracy: 0.9768 - loss: 0.0689 - val_accuracy: 0.6821 - val_loss: 1.6604
Epoch 10/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 199s 466ms/step - accuracy: 0.9798 - loss: 0.0595 - val_accuracy: 0.6723 - val_loss: 1.7120
```

CNN Implementation

In the case of Deep Learning, the CNN model is used to identify important local features and patterns from the prepared Nepali text sequences. The CNN model consists of four layers – the embedding layer, the convolutional layer, pooling layer, and output layer. After designing the CNN model, it was compiled using an appropriate optimizer, loss function, and evaluation metric. The model was then trained using the prepared training dataset for Nepali sentiment classification, and its performance was recorded for further evaluation and comparison with the LSTM and BiLSTM models.

```
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D

cnn_model = Sequential([
    Embedding(20000, 128),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(1, activation='sigmoid')
])
```

[21] ✓ 10m

```
# Step 7 — Train the CNN
# Now train the model.
# Use the same basic training settings as your LSTM/BiLSTM:
cnn_history = cnn_model.fit(
    X_train_pad,
    y_train_encoded,
    validation_split=0.1,
    epochs=10,
    batch_size=64,
    verbose=1
)
```

```
Epoch 1/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 113ms/step - accuracy: 0.6474 - loss: 0.7950 - val_accuracy: 0.6919 - val_loss: 0.7099
Epoch 2/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 116ms/step - accuracy: 0.7845 - loss: 0.5485 - val_accuracy: 0.7111 - val_loss: 0.7052
Epoch 3/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 112ms/step - accuracy: 0.8719 - loss: 0.3580 - val_accuracy: 0.7065 - val_loss: 0.8094
Epoch 4/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 82s 112ms/step - accuracy: 0.9248 - loss: 0.2214 - val_accuracy: 0.6988 - val_loss: 0.9856
Epoch 5/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 116ms/step - accuracy: 0.9535 - loss: 0.1433 - val_accuracy: 0.6887 - val_loss: 1.1780
Epoch 6/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 111ms/step - accuracy: 0.9677 - loss: 0.1039 - val_accuracy: 0.6908 - val_loss: 1.2882
Epoch 7/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 111ms/step - accuracy: 0.9759 - loss: 0.0763 - val_accuracy: 0.6866 - val_loss: 1.4435
Epoch 8/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 46s 114ms/step - accuracy: 0.9798 - loss: 0.0670 - val_accuracy: 0.6793 - val_loss: 1.5637
Epoch 9/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 81s 111ms/step - accuracy: 0.9848 - loss: 0.0490 - val_accuracy: 0.6814 - val_loss: 1.7104
Epoch 10/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 83s 115ms/step - accuracy: 0.9866 - loss: 0.0450 - val_accuracy: 0.6908 - val_loss: 1.8374
```

[25] ✓ 10s

```
# Step 10 — Evaluate CNN on the Test Set
cnn_test_loss, cnn_test_accuracy = cnn_model.evaluate(
    X_test_pad,
    y_test_encoded,
    verbose=1
)

print("CNN Test Loss:", cnn_test_loss)
print("CNN Test Accuracy:", cnn_test_accuracy)
print(
    "CNN Test Accuracy (%):",
    cnn_test_accuracy * 100
)
```

```
224/224 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - accuracy: 0.6906 - loss: 1.7752
CNN Test Loss: 1.77516508102417
CNN Test Accuracy: 0.6906394362449646
CNN Test Accuracy (%): 69.06394362449646
```

Model Evaluation Implementation

After training the Machine Learning and Deep Learning models, their performance was evaluated using the testing dataset. Predictions generated by each model were compared with the actual

sentiment labels, and the main evaluation parameters i.e. Accuracy, Precision, Recall, and F1-Score were calculated. A confusion matrix was also generated to observe the correct and incorrect classifications made by each model. These evaluation measures were then used to compare the models and identify the best-performing Machine Learning and Deep Learning approaches.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Confusion Matrix Implementation

A confusion matrix was generated for each sentiment classification model to examine the number of correctly and incorrectly classified instances. It presents the relationship between the actual sentiment labels and the predicted sentiment labels**,** making it easier to identify correct predictions and classification errors. The confusion matrices were generated using the testing data and visualized as heat maps for clearer interpretation and comparison of model performance.

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)

plt.imshow(cm)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Figure 7: DL Model Comparison Final Results

Final Machine Learning and Deep Learning Comparison

Finally, the best-performing Machine Learning model and Deep Learning model were compared to determine which approach provided better performance for Nepali language sentiment classification. The comparison was performed using the evaluation results obtained from Accuracy, Precision, Recall, and F1-Score. The results of the selected models were presented in a comparative table and graphical form, allowing the overall performance of traditional Machine Learning and Deep Learning approaches to be clearly examined.

Chapter V: Results and Discussion

This chapter presents the results obtained from the implementation and evaluation of the Machine Learning and Deep Learning models for Nepali language sentiment classification. In this study, the performances of Naïve Bayes, SVM, and Logistic Regression are compared with those of Deep Learning algorithms such as Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), and Convolutional Neural Network (CNN). Various evaluation metrics like accuracy, precision, recall, and F1-score were utilized in order to analyze the effectiveness of each model. Confusion matrices were also generated to provide a clearer understanding of the classification performance across different sentiment categories. The results help identify the strengths and limitations of traditional Machine Learning and Deep Learning approaches when applied to Nepali text data.

The results are analyzed through the comparison of the overall performance of the implemented models and their capability of accurate classification of sentiments. Traditional Machine Learning techniques can work effectively when appropriate feature extraction techniques are utilized, while Deep Learning techniques are capable of learning from data without the need for any additional feature extraction techniques (Goldberg, 2017). For instance, models like LSTM and BiLSTM can be very useful in case of sequential text since they can capture the relationships between words in longer context (Hochreiter & Schmidhuber, 1997). Therefore, comparative analysis provided in this chapter will help to understand better which of the two is more effective for Nepali language sentiment classification.

Comparison among Machine Learning Models

The three Traditional Machine Learning models, namely Naïve Bayes, SVM and Logistic Regression, were used for implementation in the first stage of the experiment. The processed Nepali texts were encoded into feature vectors with the help of an appropriate representation technique like BoW or TF-IDF and these feature vectors were fed to the ML models. The training and testing were done on the same data by all the ML models to ensure a fair comparison.

Table 3: Comparison of Machine Learning Algorithms

ML Model / Metrics	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	64.32%	69%	65%	67%
Support Vector Machine (SVM)	65.9%	68%	71%	70%
Logistic Regression	67.42%	70%	73%	71%

The Table 5.1 below shows the comparative results of the three ML models namely Naïve Bayes, Support Vector Machine (SVM) and Logistic Regression. The performance of these three models has been calculated based on accuracy, precision, recall and F1-score. Of the three models, Logistic Regression performed the best, with an accuracy of 67.42%, precision of 70%, recall of 73% and F1-score of 71%. SVM gave the second-best performance with an accuracy of 65.90%, precision of 68%, recall of 71% and F1-score of 70%. The Naïve Bayes performed comparatively worse than the other two with an accuracy of 64.32%, precision of 69%, recall of 65% and F1-score of 67%.

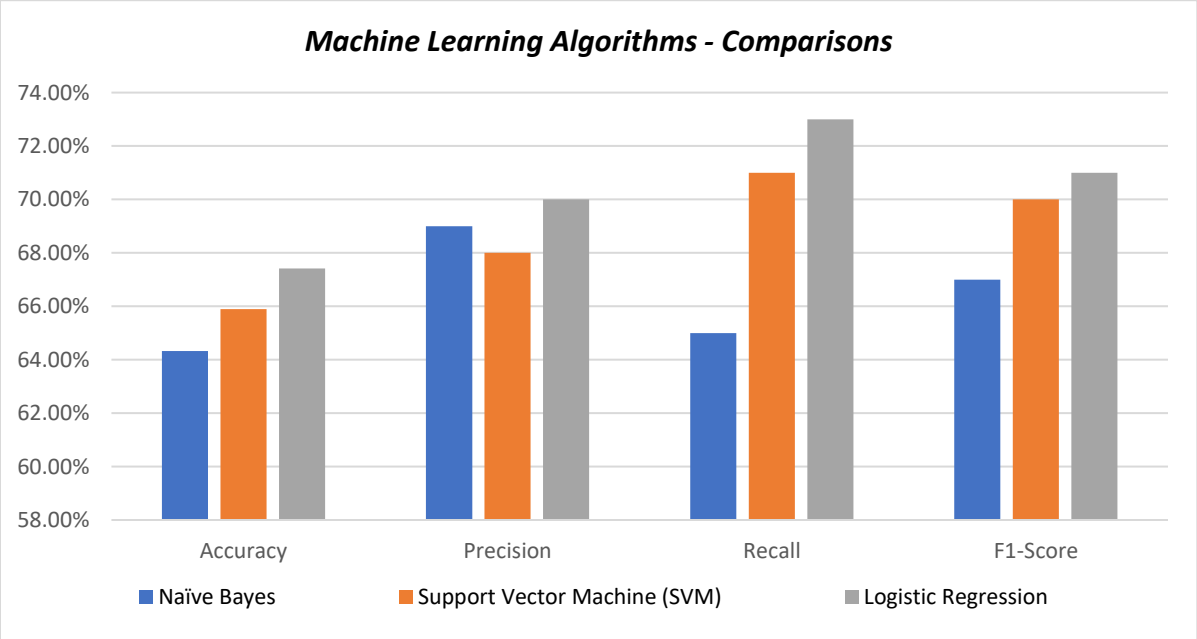


Figure 8: Bar Chart Machine Learning Algorithm Comparisons

From the results, it can be noted that Logistic Regression is the best Machine Learning model that can be used for the selected Nepali language sentiment dataset. Even though the difference in the accuracy rate between the different models was not significant, there is a considerable difference in terms of the general balance between the different metrics, especially recall and F1-score. In addition, SVM is a model that showed good results and outperformed Naïve Bayes in terms of recall and F1-score. Based on these findings, Logistic Regression was selected as the best-performing Machine Learning model and was subsequently used for the final comparison with the best-performing Deep Learning model.

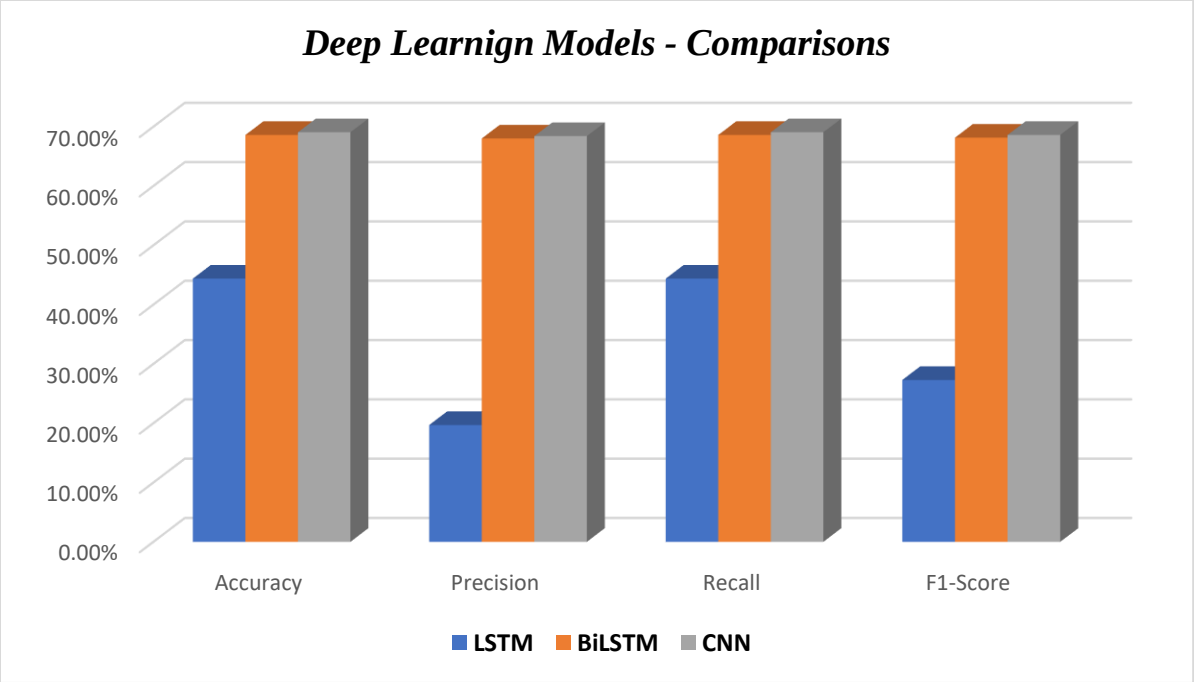
Comparison among Deep Learning Models

The Deep Learning models, namely LSTM, BiLSTM, and CNN, were implemented and evaluated for Nepali language sentiment classification. For consistency in the experimental procedure, the models were trained using a batch size of 64, an embedding dimension of 128, and 10 epochs. Accuracy, precision, recall, and F1-score are some of the metrics that were used for the evaluation of the performance of the models. This was done in order to measure the performance of the models in classifying the sentiment categories

Table 4: Deep Learning Models Comparison

DL Model / Metrics	Accuracy	Precision	Recall	F1-Score
LSTM	44.38%	19.70%	44.38%	27.29%
BiLSTM	68.60%	68.01%	68.60%	68.15%
CNN	69.06%	68.41%	69.06%	68.61%

As seen from the results that are illustrated by the table and bar chart below, there is a notable difference in the performance of the three deep learning models. LSTM model shows the lowest performance with 44.38% of accuracy, 19.70% of precision, 44.38% of recall, and 27.29% F1-score. However, the BiLSTM model showed good results with 68.60% accuracy, 68.01% precision, 68.60% recall, and 68.15% F1-score. The significant increase in the performance of BiLSTM is because of the fact that the bidirectional approach helped in better understanding of the context of the relationship between words in the Nepali language as compared to the conventional LSTM approach.



Out of all the Deep Learning techniques, CNN provided the best performance with an accuracy of 69.06%, precision of 68.41%, recall of 69.06% and F1-score of 68.61%. Although the difference between CNN and BiLSTM was relatively small, CNN achieved the highest values across all the selected evaluation metrics. The bar chart also clearly illustrates the comparatively low performance of LSTM and the closely competitive performance of BiLSTM and CNN. Based on the overall evaluation results, CNN was identified as the best-performing Deep Learning model and was therefore selected for the final comparison with the best-performing Machine Learning model.

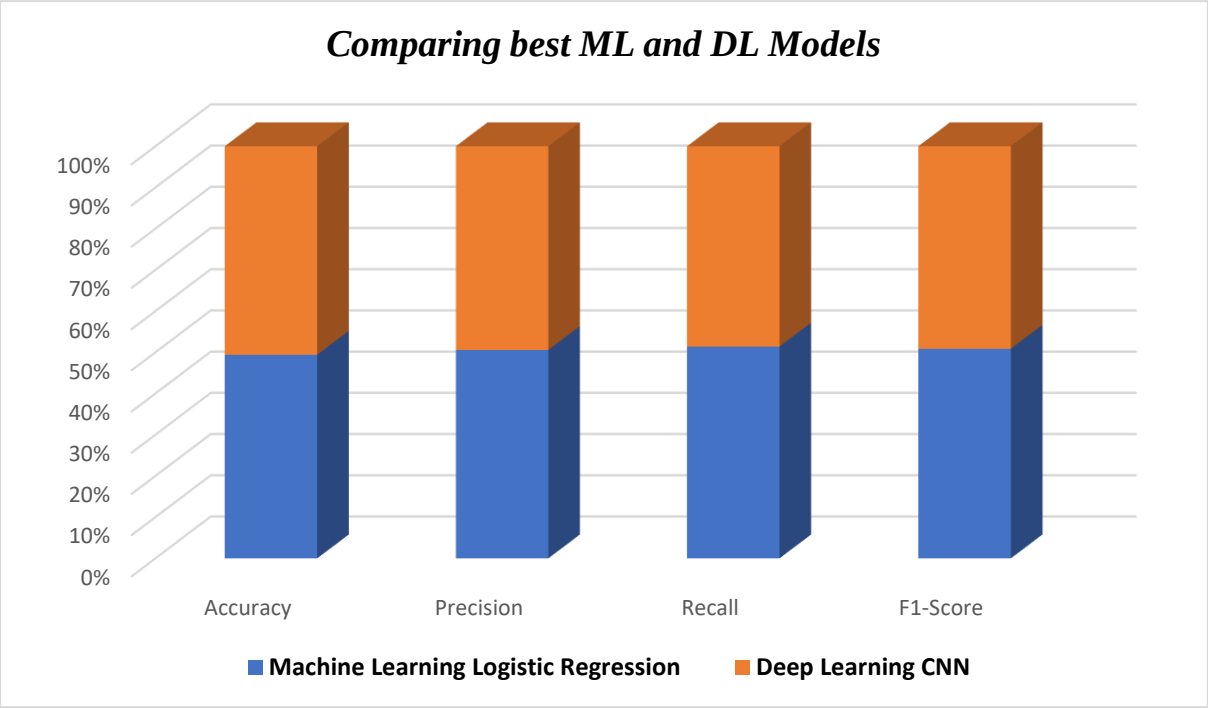
Final Comparison between Machine Learning and Deep Learning

Out of all the Machine Learning techniques considered in the present analysis, Logistic Regression has been found to provide the best performance. The accuracy achieved by it is 67.42% with a precision of 70%, recall of 73% and an F1-score of 71%. From the above results, it can be understood that Logistic Regression has outperformed the Naïve Bayes, Support Vector Machine, and Logistic Regression. On the other hand, CNN outperformed all the other deep learning techniques with an accuracy of 69.06%, precision of 68.41%, recall of 69.06% and an F1-score of 68.61%. Thus, Logistic Regression and CNN were the chosen ones out of machine learning and deep learning respectively.

Table 5: Final Comparison between ML and DL Models

Approach	Best Model	Accuracy	Precision	Recall	F1-Score
Machine Learning	Logistic Regression	67.42%	70%	73%	71%
Deep Learning	CNN	69.06%	68.41%	69.06%	68.61%

From the results shown in the comparison table above, CNN is seen to have performed slightly better than Logistic Regression in accuracy. CNN had an accuracy of 69.06% compared to 67.42% from Logistic Regression, giving a difference of 1.64%. However, in precision, recall, and F1 score, Logistic Regression performed well than CNN, with 70%, 73%, and 71% respectively compared to CNN, which had 68.41% in precision, 69.06% in recall, and 68.61% in F1-score. It can be noted that even though CNN made more correct predictions, Logistic Regression had better performance in other measurements.



The corresponding bar chart further illustrates the competitive performance of the two approaches. From the chart above, one can note that CNN had a slight edge in accuracy while Logistic Regression had better performance in precision, recall, and F1-score. It means that the final result of the comparison shows that both machine learning and deep learning are not better than each

other in all the measurements. The findings suggest that both approaches were effective for Nepali language sentiment classification using the selected dataset and experimental procedure.

Discussions of Findings

The findings of this study show that both traditional Machine Learning and Deep Learning approaches can be effectively applied to Nepali language sentiment classification. In the case of Machine Learning models, the best model in all evaluation measures was the Logistic Regression Model. In the case of Deep Learning models, CNN had the highest accuracy compared to others and was slightly superior to the other Deep Learning models. However, in the final comparison between CNN and Logistic Regression, it was found that CNN was not always superior to Logistic Regression in all evaluation measures. Though CNN had the highest accuracy, the precision, recall, and F1-Score of Logistic Regression were higher than those of CNN. It shows that the performance of a model is not determined by accuracy alone.

The overall findings suggest that Deep Learning does not necessarily guarantee better performance than traditional Machine Learning for every dataset and experimental setting. In this study, the Machine Learning models combined with TF-IDF feature extraction produced highly competitive results, while the Deep Learning models were able to learn patterns directly from the textual data. The performance of Deep Learning models can be influenced by several factors, including model architecture, dataset characteristics, number of training epochs, embedding dimensions, and other hyper parameter settings. Therefore, the results indicate that both approaches have their own strengths and can be useful for Nepali language sentiment classification. The study provides a practical comparison of different techniques and shows that selecting an appropriate model should depend on the evaluation results and requirements of the specific application rather than assuming that one approach will always perform better than the other.

Chapter VI: Conclusion and Implication

Conclusion

This study conducted a comparative analysis of Nepali language sentiment classification using Machine Learning and Deep Learning approaches. Firstly, the dataset consisted of 35,789 Nepali text samples, and after excluding the duplicate entries, 35,732 samples were used in the experiment. The pre-processing of the dataset and the splitting it into training and testing data was done according to the 80:20 split. The evaluation of three Machine Learning models was performed in the context of TF-IDF feature representation. These three models included Naïve Bayes, Support Vector Machine (SVM), and Logistic Regression. Besides, in the context of Deep Learning, three architectures were implemented such as LSTM, BiLSTM, and CNN. The experimental results indicated that the Logistic Regression algorithm outperformed all other Machine Learning models in terms of accuracy, precision, recall, and F1-score being equal to 67.42%, 70%, 73%, and 71% correspondingly. Among Deep Learning architectures, the CNN architecture produced the highest values of accuracy, precision, recall, and F1-score being equal to 69.06%, 68.41%, 69.06%, and 68.61%. Comparing these results, the CNN architecture demonstrated better accuracy than the Logistic Regression algorithm on 1.64 percent points. However, Logistic Regression demonstrated higher precision, recall, and F1-score than CNN based on the evaluated results. Overall, the study demonstrates that both Machine Learning and Deep Learning approaches have potential for Nepali sentiment classification, with CNN providing the highest overall accuracy among the models evaluated in this study. The research also provides a useful comparative baseline for further development and research in Nepali Natural Language Processing.

Implications of the Study

The findings of this study have practical and academic implications for the development of Nepali language-based Natural Language Processing applications. The successful implementation of both Machine Learning and Deep Learning models demonstrates the potential of computational techniques for automatically identifying sentiment from Nepali textual data. Such techniques can be useful in applications such as social media sentiment analysis, customer feedback analysis, opinion mining, educational feedback analysis, and other text-based decision-support systems. The

comparative results also indicate that the choice of classification model can influence the effectiveness of sentiment analysis, and therefore researchers and developers should consider the characteristics of the dataset and the intended application when selecting an appropriate approach. From an academic perspective, the study provides a practical example of applying data preprocessing, feature extraction, model training, evaluation, and comparative analysis to a Nepali NLP problem. The findings may therefore serve as a useful baseline for students, teachers, and researchers interested in Machine Learning, Deep Learning, and Nepali language computing.

Contribution to NCCS

As this mini research was conducted for the National College of Computer Studies (NCCS), Paknajol, Kathmandu, the study also has potential academic and institutional value for the college. In particular, a considerable number of students from programs such as BCA, BITM, and CSIT undertake final-year projects involving Machine Learning, Deep Learning, Artificial Intelligence, and related areas. In this context, the present study can serve as a practical reference for students who are planning or developing projects related to text classification, sentiment analysis, Nepali language processing, and other NLP applications. The methodology followed in this research, including dataset preparation, Nepali text preprocessing, train-test splitting, TF-IDF feature extraction, model implementation, performance evaluation, and comparison of ML and DL approaches, can provide students with a structured example of conducting an applied AI/ML research project. The study may also encourage further exploration of Nepali language technologies among students and faculty members and contribute to strengthening the academic research environment in Artificial Intelligence, Machine Learning, Deep Learning, and Natural Language Processing at NCCS. Furthermore, the findings can provide a foundation for future student projects and institutional research involving Nepali textual data and intelligent information-processing systems.

Recommendations

Based on the findings of this study, future research can explore larger and more diverse Nepali language datasets to improve the generalizability of sentiment classification models. Further experiments could even involve state-of-the-art methods like transformers, word embedding,

ensemble learning, and Hybrid Machine Learning and Deep Learning techniques. Other areas of research could be related to more complicated elements of Nepali language, like sarcasm, code-mixing, emotion recognition, and aspect-based sentiment analysis. Future studies can also develop practical web- or mobile-based applications for real-time Nepali sentiment analysis.

For academic purposes, the methodology and findings of this study can be used as a reference for students and researchers at NCCS who are interested in Machine Learning, Deep Learning, Artificial Intelligence, and Natural Language Processing. Students undertaking final-year projects can further extend this work by using larger datasets, different models, and application oriented implementations. Such extensions can help strengthen practical research and experimentation in Nepali language technologies.

Future Work

The present study provides several opportunities for future research and improvement. Future studies can evaluate larger, more diverse, and domain-specific Nepali datasets collected from sources such as social media, online news, product reviews, and educational feedback. Advanced transformer-based models such as multilingual BERT, Nepali-specific language models, and XLM-R can be investigated to compare their performance with the traditional Machine Learning and Deep Learning approaches evaluated in this study. Future research may also explore ensemble techniques, hyper parameter optimization, cross-validation, word embedding, and hybrid ML-DL architectures to improve classification performance. In addition, sentiment analysis can be extended beyond basic sentiment classification to areas such as aspect-based sentiment analysis, sarcasm detection, emotion classification, and multilingual or code-mixed Nepali text analysis. From an application perspective, the developed approaches could be incorporated into a web- or mobile-based sentiment analysis system capable of processing Nepali text in real time. Such developments could further increase the practical value of Nepali NLP and provide additional opportunities for student projects and institutional research at NCCS.

References

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). SAGE Publications.

Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, *9*(3), 483. https://doi.org/10.3390/electronics9030483

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (3rd ed.). O'Reilly Media.

Goldberg, Y. (2017). *Neural network methods for natural language processing*. Morgan & Claypool Publishers.

Islam, M. S., Kabir, M. N., Ghani, N. A., Zamli, K. Z., Zulkifli, N. S. A., Rahman, M. M., & Moni, M. A. (2024). Challenges and future in deep learning for sentiment analysis: A comprehensive review and a proposed novel hybrid approach. *Artificial Intelligence Review, 57*, Article 62. https://doi.org/10.1007/s10462-023-10651-9

Joseph, J., Vineetha, S., & Sobhana, N. V. (2022). A survey on deep learning based sentiment analysis. *Materials Today: Proceedings, 58*, 456–460. https://doi.org/10.1016/j.matpr.2022.02.483

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2023). Machine learning and deep learning for sentiment analysis across languages: A survey. *Neurocomputing, 531*, 195–216. https://doi.org/10.1016/j.neucom.2023.02.015

Minaee, S., Azimi, E., & Abdolrashidi, A. (2019). Deep-sentiment: Sentiment analysis using ensemble of CNN and Bi-LSTM models. *arXiv*. https://arxiv.org/abs/1904.04206

Piryani, R., Piryani, B., Singh, V. K., & Pinto, D. (2020). Sentiment analysis in Nepali: Exploring machine learning and lexicon-based approaches. *Journal of Intelligent & Fuzzy Systems, 39*(2), 2201–2212. https://doi.org/10.3233/JIFS-179884

Regmi, S., Bal, B. K., & Kultsova, M. (2017). Analyzing facts and opinions in Nepali subjective texts. In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1–4). IEEE. https://doi.org/10.1109/IISA.2017.8316445

Ramaswamy, S. L., & Chinnappan, J. (2023). Review on positional significance of LSTM and CNN in the multilayer deep neural architecture for efficient sentiment classification. *Journal of Intelligent & Fuzzy Systems, 45*(4). https://doi.org/10.3233/JIFS-230917

Sitaula, C., Basnet, A., Mainali, A., & Shahi, T. B. (2021). Deep learning-based methods for sentiment analysis on Nepali COVID-19-related tweets. *Computational Intelligence and Neuroscience, 2021*, Article 2158184. https://doi.org/10.1155/2021/2158184

Sitoula, S., Shahi, T. B., Wibowo, S., & Neupane, A. (2025). Sentiment analysis of Nepali social media text with a hybrid deep learning model. *Social Network Analysis and Mining, 15*, Article 85. https://doi.org/10.1007/s13278-025-01508-w

Tamrakar, S., Bal, B. K., & Thapa, R. B. (2020). Aspect based sentiment analysis of Nepali text using support vector machine and Naïve Bayes. *Technical Journal, 2*(1), 22–29.

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8*(4), e1253. https://doi.org/10.1002/widm.1253

Zhou, K., & Long, F. (2018). Sentiment analysis of text based on CNN and bi-directional LSTM model. In *2018 24th International Conference on Automation and Computing (ICAC)* (pp. 1–5). IEEE.

Appendix I: Data Preprocessing Codes

```
df.head()
```

	Unnamed: 0	Sentences	Sentiment
0	0	म एक शिक्षक , शिक्षा क्षेत्रमा रमाएको मान्छे ।...	1
1	1	म सरकारी स्कूल/कलेजमा पढेर करीब १२ बर्ष भन्दा ...	1
2	2	कति राम्रो शिव मन्दिर देख्न पाइयो कुन ठाउको हो...	1
3	3	मारुनी भन्ने वितिकै सामान्य नाचनीमा आधारित कथा...	1
4	4	यो फ्लिम हेरिसकेपछी थाहा भयो कि किन दर्सकहरुले...	1

```
print("Dataset shape:", df.shape)
print("\nColumn names:")
print(df.columns)

print("\nSentiment distribution:")
print(df['label'].value_counts())
```

```
Dataset shape: (35732, 2)

Column names:
Index(['text', 'label'], dtype='object')

Sentiment distribution:
label
 1    15857
-1    14393
 0     5482
Name: count, dtype: int64
```

Appendix II: TF IDF and Machine Learning Models Code

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(
    max_features=10000,
    ngram_range=(1, 2)
)
```

```
print("Training TF-IDF shape:", X_train_tfidf.shape)
print("Testing TF-IDF shape:", X_test_tfidf.shape)
```

```
Training TF-IDF shape: (28585, 10000)
Testing TF-IDF shape: (7147, 10000)
```

```
print(tfidf.get_feature_names_out()[:50])
```

```
['अक' 'अक जन' 'अक जनक' 'अक बर' 'अक सफ' 'अक सफर' 'अकर' 'अकर मण' 'अकल'
 'अकल पन' 'अख' 'अग' 'अग बढ' 'अग रप' 'अग रपङ' 'अग रभ' 'अग रम' 'अग रस'
 'अग रह' 'अगष' 'अगस' 'अघ' 'अघ पछ' 'अघ बढ' 'अघ वर' 'अघ सम' 'अघ सर' 'अङ'
 'अच' 'अच नक' 'अचम' 'अछ' 'अछ मक' 'अछ रह' 'अज' 'अझ' 'अझ एक' 'अझ कड'
 'अझ नभएक' 'अझ पन' 'अझ बढ' 'अझ सक' 'अझ सम' 'अञ' 'अञ चल' 'अट' 'अड' 'अण'
 'अण अध' 'अत']
```

```
from sklearn.linear_model import LogisticRegression
```

```
model_lr = LogisticRegression(
    max_iter=1000,
    random_state=42
)
```

```
from sklearn.linear_model import LogisticRegression
```

```
model_lr = LogisticRegression(
    max_iter=1000,
    random_state=42
)
```

Appendix III: Deep Learning Models Code

```
[23]
✓ 20m
history = model.fit(
    X_train_pad,
    y_train_encoded,
    validation_split=0.1,
    epochs=10,
    batch_size=64,
    verbose=1
)
```

```
Epoch 1/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 258ms/step - accuracy: 0.4327 - loss: 1.0190 - val_accuracy: 0.4390 - val_loss: 1.0303
Epoch 2/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 254ms/step - accuracy: 0.4373 - loss: 1.0167 - val_accuracy: 0.4390 - val_loss: 1.0240
Epoch 3/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 263ms/step - accuracy: 0.4417 - loss: 1.0162 - val_accuracy: 0.4390 - val_loss: 1.0244
Epoch 4/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 258ms/step - accuracy: 0.4432 - loss: 1.0153 - val_accuracy: 0.4390 - val_loss: 1.0235
Epoch 5/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 102s 255ms/step - accuracy: 0.4420 - loss: 1.0157 - val_accuracy: 0.4390 - val_loss: 1.0229
Epoch 6/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 106s 263ms/step - accuracy: 0.4399 - loss: 1.0151 - val_accuracy: 0.4390 - val_loss: 1.0228
Epoch 7/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 258ms/step - accuracy: 0.4413 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0230
Epoch 8/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 104s 259ms/step - accuracy: 0.4423 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0231
Epoch 9/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 140s 255ms/step - accuracy: 0.4430 - loss: 1.0148 - val_accuracy: 0.4390 - val_loss: 1.0243
Epoch 10/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 104s 258ms/step - accuracy: 0.4431 - loss: 1.0147 - val_accuracy: 0.4390 - val_loss: 1.0226
```

```
[47]
✓ 21s
# Step 10 — Evaluate BiLSTM on the test set
# Now use the same 7,147 unseen test samples:
bilstm_test_loss, bilstm_test_accuracy = bilstm_model.evaluate(
    X_test_pad,
    y_test_encoded,
    verbose=1
)
print("BiLSTM Test Loss:", bilstm_test_loss)
print("BiLSTM Test Accuracy:", bilstm_test_accuracy)
print(
    "BiLSTM Test Accuracy (%):",
    bilstm_test_accuracy * 100
)
```

```
224/224 ━━━━━━━━━━━━━━━━━━━━ 21s 96ms/step - accuracy: 0.6736 - loss: 1.6965
BiLSTM Test Loss: 1.6965041160583496
BiLSTM Test Accuracy: 0.6735693216323853
BiLSTM Test Accuracy (%): 67.35693216323853
```

[21]
✓ 10m

```python
# Step 7 — Train the CNN
# Now train the model.
# Use the same basic training settings as your LSTM/BiLSTM:
cnn_history = cnn_model.fit(
    X_train_pad,
    y_train_encoded,
    validation_split=0.1,
    epochs=10,
    batch_size=64,
    verbose=1
)
```

```
Epoch 1/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 113ms/step - accuracy: 0.6474 - loss: 0.7950 - val_accuracy: 0.6919 - val_loss: 0.7099
Epoch 2/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 116ms/step - accuracy: 0.7845 - loss: 0.5485 - val_accuracy: 0.7111 - val_loss: 0.7052
Epoch 3/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 112ms/step - accuracy: 0.8719 - loss: 0.3580 - val_accuracy: 0.7065 - val_loss: 0.8094
Epoch 4/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 82s 112ms/step - accuracy: 0.9248 - loss: 0.2214 - val_accuracy: 0.6988 - val_loss: 0.9856
Epoch 5/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 47s 116ms/step - accuracy: 0.9535 - loss: 0.1433 - val_accuracy: 0.6887 - val_loss: 1.1780
Epoch 6/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 111ms/step - accuracy: 0.9677 - loss: 0.1039 - val_accuracy: 0.6908 - val_loss: 1.2882
Epoch 7/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 45s 111ms/step - accuracy: 0.9759 - loss: 0.0763 - val_accuracy: 0.6866 - val_loss: 1.4435
Epoch 8/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 46s 114ms/step - accuracy: 0.9798 - loss: 0.0670 - val_accuracy: 0.6793 - val_loss: 1.5637
Epoch 9/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 81s 111ms/step - accuracy: 0.9848 - loss: 0.0490 - val_accuracy: 0.6814 - val_loss: 1.7104
Epoch 10/10
402/402 ━━━━━━━━━━━━━━━━━━━━ 83s 115ms/step - accuracy: 0.9866 - loss: 0.0450 - val_accuracy: 0.6908 - val_loss: 1.8374
```

	Model	Accuracy	Precision	Recall	F1-Score
0	LSTM	44.38	19.70	44.38	27.29
1	BiLSTM	68.60	68.01	68.60	68.15
2	CNN	69.06	68.42	69.06	68.61

Next steps: Generate code with comparison_dl | New interactive sheet

Appendix IV: Confusion Matrix and Result Analysis Code

[29] ✓ 0s

```
# Step 12 – Classification Report

from sklearn.metrics import classification_report
print(
    classification_report(
        y_test_encoded,
        cnn_y_pred,
        target_names=[
            'Negative',
            'Neutral',
            'Positive'
        ]
    )
)
```

```
              precision    recall  f1-score   support

    Negative       0.74      0.72      0.73      2879
     Neutral       0.45      0.37      0.41      1096
    Positive       0.71      0.77      0.74      3172

    accuracy                           0.69      7147
   macro avg       0.64      0.62      0.63      7147
weighted avg       0.68      0.69      0.69      7147
```

[25] ✓ 10s

```
# Step 10 – Evaluate CNN on the Test Set
cnn_test_loss, cnn_test_accuracy = cnn_model.evaluate(
    X_test_pad,
    y_test_encoded,
    verbose=1
)

print("CNN Test Loss:", cnn_test_loss)
print("CNN Test Accuracy:", cnn_test_accuracy)
print(
    "CNN Test Accuracy (%):",
    cnn_test_accuracy * 100
)
```

```
224/224 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - accuracy: 0.6906 - loss: 1.7752
CNN Test Loss: 1.77516508102417
CNN Test Accuracy: 0.6906394362449646
CNN Test Accuracy (%): 69.06394362449646
```