TikNep: Content Analysis of Nepali TikTok Users using Natural Language Processing

by

Aakar Dhakal
20201203
Ashok Lamichhane
21201785
Aatish Kumar Jha
20201206
Mukund Prasad Singh
20201202

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

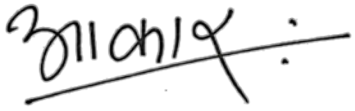
Department of Computer Science and Engineering
Brac University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Aakar Dhakal
20201203

Ashok Lamichhane
21201785

Aatish Kumar Jha
20201206

Mukund Prasad Singh
20201202

Approval

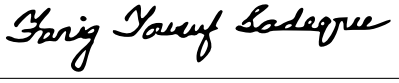
The thesis titled "TikNep: Content Analysis of Nepali TikTok Users using Natural Language Processing" submitted by

1. Aakar Dhakal (20201203)
2. Ashok Lamichhane (21201785)
3. Aatish Kumar Jha (20201206)
4. Mukund Prasad Singh (20201202)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May, 2024.

Examining Committee:

Supervisor:
(Member)



Dr. Farig Yousuf Sadeque
Associate Professor
Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Nepal, with an approximate population of 29 million, has over 2.2 million active TikTok users, TikTok has gained attention as a platform for self-expression and social connection among diverse age groups. Users in Nepal are using TikTok to share their opinions on matters related to politics, social issues, pop culture, lifestyle and beauty, sports, etc. While the content on platforms like Facebook and Twitter has been studied and evaluated thoroughly, the impact and influence of TikTok's content on Nepali society have not been assessed yet. In this study, we propose to analyze content on Nepal's TikTok using Natural Language Processing (NLP) tools to draw conclusions regarding where the conversation is being shifted towards. To meet this objective, we will focus on the comments posted by users on popular TikTok videos in Nepal and conduct Sentiment Analysis, Hate-Offense Detection, Political Stance Detection, and Multi-label Topic Classification.

Keywords: TikTok; Nepal; Social Influence; Natural Language Processing; Machine Learning; Deep Learning; Sentiment Analysis; Hate-Offense Detection; Political Stance Detection; Multi-label Topic Classification

Acknowledgement

We would like to express our sincere gratitude to our supervisor, Dr. Farig Yousuf Sadeque, for his invaluable support and guidance throughout our thesis. His assistance was crucial at every step of our thesis.

We are also deeply appreciative of consultations provided by Suraj Maharjan, Prasha Shrestha, and Dr. Bal Krishna Bal. Their combined expertise and pioneering work set foundation for Nepali Natural Language Processing, driving forward our research and curiosity in this field.

And finally we are grateful for immense love and support we received from our friends and family during our research.

Table of Contents

List of Figures

List of Tables

Nomenclature

$ALBERT$	A Lite BERT
$BERT$	Bidirectional Encoder Representations from Transformers
$Bi-GRU$	Bidirectional Gated Recurrent Unit
$Bi-LSTM$	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Network
DL	Deep Learning
GNB	Gaussian Naïve Bayes
LDA	Latent Dirichlet Allocation
ML	Machine Learning
MNB	Multinomial Naïve Bayes
NB	Naïve Bayes
NLP	Natural Language Processing
RF	Random Forest
RNN	Recurrent Neural Network
$SANDS$	Self-Attention Network with Distance-aware Self-attention
SVM	Support Vector Machine
$tALBERT-CNN$	task-specific ALBERT with Convolutional Neural Networks
$TF-IDF$	Term Frequency Inverse Document Frequency

Chapter 1

Introduction

In recent years, TikTok has emerged as one of the most dynamic and popular social media platforms, with over 3.5 billion downloads in the App Store and Play Store combined. It was the most downloaded app in 2022, with 879.2 million downloads. TikTok has over 1 billion monthly active users and is available in more than 135 countries and over 35 languages. TikTok's popularity can be attributed to its ability to allow users to create, view, and share short videos ranging from 15 seconds to 3 minutes. TikTok users can add effects like background music, stickers, and filters to create engaging content and even collaborate with other users to create content. TikTok's algorithm maps out user personality and interest based on their engagement with the existing content and their simple data like language, location, and device used. Based on this analysis, TikTok offers highly relevant content to its users. According to Statista's report on social media usage in 2022, the average user spends 53.8 minutes on TikTok daily, which is the highest among all social media platforms [35].

TikTok has established itself as one of the most prominent social media platforms in Nepal. With an approximate population of 29 million, Nepal has over 2.2 million active users, but regulators and digital activists suggest the number could be much higher. In Nepal, TikTok has gained attention as a platform for self-expression and discussion on contemporary issues among diverse age groups and backgrounds. Unlike platforms like YouTube, Facebook, Twitter, etc. the contents of TikTok have not been analyzed, and the impact and influence of TikTok have not been assessed yet. This presents a significant gap in understanding where the conversation is headed in the virtual world of TikTok and what it means for overall society. On 13th November 2023, the government has insisted that it made the decision to outlaw the social media app due to the widespread distribution of content that it saw as endangering public order, inciting religious intolerance and violence, fracturing families, and promoting misogyny and casteism [39]. Due to this negative effect of Tiktok, we decided to conduct sentiment analysis, hate and offense detection, political stance detection and multi-label topic classification to verify the facts of how it affects Nepali people along with its causes with consequences.

The goal of this thesis is to develop methods that will help us understand the contents available on TikTok. We aim to achieve this goal by leveraging Natural Language Processing (NLP) and Machine Learning (ML) tools. We will be focusing

on the comments posted by Nepali TikTok users on popular videos on diverse topics.

First, we will conduct sentiment analysis to determine whether the comments are positive, negative, or neutral. Second, we plan to detect whether the comments contain any targeted hate speech or offensive language. Third, we will detect if there is any political stance made in the comment section. Finally, we will conduct multi-label topic classification to classify the types of user interests in the TikTok posts. This holistic method will enable us to completely evaluate the multidimensionality of TikTok content in Nepal.

Finally, it is also crucial for us to protect the user's privacy and follow ethical standards during the study. By the completion of this study, we want to present a detailed and data-driven understanding of TikTok's content in Nepal, unwrapping its impact on society, and delivering important insights for users, policymakers, and platform administrators. Moreover, our findings will contribute to the broader academic discussion on the influence of dynamic social media platforms on modern cultures, laying the stage for further research on this subject.

1.1 Problem Statement

TikTok is an entertaining platform for some users, and it can also be used to promote one's business and talent. The tremendous growth and influence of TikTok has certainly raised important questions regarding contents produced and consumed in the platform. To address these concerns, a study of TikTok's content is crucial as it deals with various societal elements.

Cyberbullying, hate and offensive speech dispersion, racism, age inappropriate content, etc. all can be present in TikTok. In regard to these considerations, our research will focus on the TikTok contents of Nepal with an aim to recognize its influence and impact on the society. It is also important to build tools which can be used by the regulators and platform administrators to filter or block contents which violates the user policy. So, our research aims to contribute to how a dynamic platform such as TikTok plays a significant role in the digital era.

1.2 Research Contribution

The main objective of our thesis is to assess the TikTok's comments of Nepali users based on several parameters using Natural Language Processing(NLP) techniques. Our detailed objectives are as follows:

- Investigate the emotional tone and sentiment trends displayed by Nepali TikTok users, aiming to identify the prominent emotions and their variations over time, and also study the potential links between sentiment patterns and external factors.
- Develop a robust model for automated Hate Speech and Offense Detection in Nepali TikTok comments, with a focus on identifying cultural insensitivity,

gender biases, racism, etc. and examining its impact on overall society.

- Analyze the content of Nepali TikTok users to identify their political leanings and affiliations, with an aim to uncover any patterns in political expression on the platform.
- Evaluate the relationship between political content and user interactions, understanding the influence of TikTok in shaping political scenarios in Nepal.
- Develop a comprehensive multi-label classification system to categorize Nepali TikTok content into various topics, including but not limited to sports, politics, social issues, pop culture, lifestyle, beauty, etc.
- Examine the distribution of contents across different topics and explore any interesting relationships between user demographics and content preferences, providing insights into the diversity of content on the platform.

1.3 Thesis Outline

Chapter 1
A summary of how in Nepal, TikTok gained attention as a platform for self-expression and the contemporary issues among diverse age groups and backgrounds.It highlights that the government has insisted to made the decision to outlaw the TikTok app due to widespread distribution of offensive content which causes endangering public order, religious intolerance, violence and promoting misogyny and caste-ism. The chapter presents the central idea of thesis, which is the content analysis of Nepali TikTok users using Natural Language Processing. It provides the context of thesis by emphasizing the problem statement and Research contribution.

Chapter 2
In this chapter a throughout assessment of literature review is conducted with the emphasize study on baseline models such as Naïve Bayes, SVM and Random Forest. It highlights that we have studied a lot of papers and articles related to the Deep learning models and Advanced Transformer Models. It provides the insight on the NLP and Transformer Models on content analysis of Nepali TikTok comments.

Chapter 3
This chapter represents the throughout description of the methodology. It describes that how we collect data from Nepal's TikTok videos, by building a web scraper using JavaScript and Python. It also explores the text pre-processing such as removing URLs, and @USER mentions, etc. and then annotating the data for different tasks using Label studio or Ms Excel. Furthermore, the chapter explains the details of the data splitting strategies used, giving a solid basis for model training and further study.

Chapter 4
In this chapter, a throughout statistical and illustrative details of study's implementation and performances are provided. It explores the performance measures with confusion matrix showing accuracy and F1-score. A thorough analysis of Baseline

models enriches the discussion by showcasing their specific strengths and weaknesses. In order to provide more understanding and accuracy, the deep learning models and transformer models are implemented.

Chapter 5
Our conclusions are outlined in this chapter along with our plans for the future. It offers recommendations for further research as well as a summary of the key findings from the study. Our main goal is to come up with new best and optimized models which will be able to do the content analysis of the text. This uplifting chapter emphasizes how the recommended system is still in the process of being developed and offers the foundation for further advancements and practical applications.

Chapter 2

Related Work

Unlike resource rich languages like English, Nepali is a low resource language and there are limited resources available for text processing and analyzing. Natural Language Processing has not matured fully in context of Nepali language.

A paper presented by Piryani et al. [23] explored machine learning and deep learning techniques, which included Support Vector Machine, Decision Tree, Logistic Regression, Multinomial Naïve Bayes, Convolutional Neural Network(CNN), Long Short-Term Memory(LSTM), and CNN-LSTM, and proposed lexicon-based approaches for sentiment analysis on Nepali tweets. The authors highlighted the necessity of employing a dictionary-based approach and the creation of Nepali sentiment lexicons due to the lack of standard Part-Of-Speech(POS) tagger and lexical resources for Nepali, a Subject-Object-Verb language. These lexicons were built using the Nepali-Nepali dictionary, Nepali-English dictionary, and available English sentiment lexicons. The lexicon-based approach preprocesses the tweet, identifies opinion-oriented features, and computes the sentiment of the tweet. The findings illuminated the superiority of deep learning models against both lexicon-based and machine learning approaches, whereas lexicon-based approaches performed better than machine learning models. Finally, the authors created and released Nepali SenticNet and Nepali SentiWordNet sentiment lexicons.

Similarly, in another study, Gupta and Bal [6] discussed two methods for sentiment analysis in Nepali text: lexical resource based and machine learning based. In the first method, the authors developed Bhavanakos, a lexical resource. The authors took all the words from SentiWordNet(English), after that translated the words to Nepali, and then eliminated stop words. In the second method, the authors used the Naïve Bayes classifier to classify Nepali text. The resource based method achieved higher precision value whereas machine learning classifier achieved higher recall value.

Another study by Tamrakar et al. [25] explored two significant classification methods in Machine Learning: Support Vector Machine and Naïve Bayes. The authors used Support Vector Machine for sentiment classification through the linear separation of data. The higher dimensionality of instances and features supports the efficiency of this method. In parallel, the study also thoroughly examined Naïve Bayes method. The Naïve Bayes method creates a classification technique that as-

sumes independence between features within a class. By probabilistically analyzing document-term matrix or numerical feature vectors, the Naïve Bayes method successfully extracts features essential for training the Naïve Bayes classifier.

In another work the Thapa and Bal [10] used Logistic Regression, Naïve Bayes, and Support Vector Machine to classify sentiments in Nepali subjective texts. First, the study examined Term Frequency-Inverse Document Frequency(TF-IDF) and Bag-of-words(BoW). Simialry, the authors again evaluated TF-IDF and BoW, but with stop words removed. The authors observed that removing stop words declined the classifier's performance, and concluded TF-IDF and BoW to be an important features in the classification models. From this study Multinomial Naïve Bayes proved to be the best classifier to classify subjective Nepali texts. Likewise, in another study Pant and Yadav [4] used Naïve Bayes classifier to classify movie reviews and achieved similar results as Thapa and Bal [10].

Another research [24] proposed a dataset for sentiment analysis in Nepali social media, by using code-mixed and code-switched comments from Nepali YouTube posts, with an emphasis on development of a new Targeted Aspect Based Abusive Sentiment Analysis (TABSA) dataset in Nepali social media data. The authors discussed data collection pipeline, annotation schema, and machine learning algorithms used to detect aspect terms, targeted terms, and their polarity. The multilingual BERT model was utilized for Aspect Term Extraction and BiLSTM model for sentiment classification tasks, attaining 57.978 % and 81.60% F1 scores, respectively. Difficulties such as subjective aspect term identification and the presence of grammatical and syntactical errors in data from social media underlines the necessity for strict guidelines and expertise in constructing reliable sentiment analysis models for Nepali social media texts.

A handful of research and studies have been conducted to detect hate and offense in Nepali text. One of the prominent study was done by Niraula, Dulal, and Koirala [31]. The authors detected offensive language present in Nepali social media by using four machine learning classifiers: Logistic Regression, Support Vector Machine, Random Forest, and Multilingual BERT. By collecting Nepali texts from diverse social media platforms, the authors outlined the offensive language frequently used in Nepali social media and discussed the difficulties in processing Nepali text. Furthermore, this study contributes to the offensive language detection in Nepali text by releasing human annotated data sets.

Furthermore, Niraula, Dulal, and Koirala [22] published a corpus-based study of offensive language in Nepali language. Their primary focus were linguistic taboos and euphemisms. The authors collected and evaluated more than 7000 social media posts present with linguistic taboos and offensive terms and classified more than 1000 tabooed and contextually offensive phrases into 18 categories.

Another study [27] presented a language-based survey of hate speech detection in Asian languages and analyzes the approaches used in language-specific studies for hate speech detection in Asian languages. Studies in different languages like Nepali, Indonesian, Hindi, Bengali, etc. used machine learning approaches such as Naïve

Bayes, Random Forest, Decision Tree, Logistic Regression, and Support Vector Machine for automated hateful text detection.

To our best knowledge, there are no previous studies done to detect political stance in Nepali text. A research by Iyyer et al. [3] proposed a Recursive Neural Network (RNN) architecture to identify the political viewpoint expressed in a sentence, taking into account the compositional aspect of language. The model considers the syntax of the text, unlike prior methods that rely on Bags-of-words or wordlists. The authors crowd sourced political annotations at phrase and sentence level to highlight the significance of modeling subsentential features, and their model outperforms previous models on both newly annotated and existing datasets. The paper underlines the disparity among lower-level phrases, that are usually neutral, and entire sentences, which are inclined to be biased.

Another study [34] offered a novel semi-supervised stance detector named SANDS, which uses homophily features across the social network to identify and categorize stance in social media text. It starts with several labeled tweets and develops multiple deep feature views of the tweets. It also utilizes a distant supervision signal from the social network to offer a surrogate loss signal to the component learners. The authors used 270,000 tweets from US and India, follower-followee graphs of the twitter users, and 8,000 tweets annotated by the linguists. The authors achieved macro-F1 scores of 0.55 and 0.49 on US and India based tweets.

Similary, another paper [33] provided an overview of the different approaches and methods used in political ideology detection, highlighting the importance of understanding the political system and the challenges associated with labeling political ideologies in textual news articles. Political ideology is closely associated with political parties and voters having different views on policies. The authors utilize deep neural networks to distinguish political ideologies, achieving F1-Score of 0.9.

In terms of multi-label topic classifcation there also exists a dearth of resources and studies done in Nepali Language. A study by Liu et al. [30] highlights the use of machine learning approaches like problem transformation and algorithm adaptation for multi-label learning, emphasizing the importance of deep learning models like CNN-based, RNN-based, and transformer-based techniques. It emphasized the need to gather semantic information from multiple levels and granularities of text. The recommended tALBERT-CNN technique outperforms the LDA topic model and the ALBERT model in multi-label text classification performance. The tALBERT-CNN model combines word-level and document-level semantic vectors, resulting in better results than fusing document-level topic vectors and semantic vectors. It outperforms competitor techniques on three independent datasets and has higher applicability.

Similarly, a paper by Fujino et al. [2] focused on addressing the multi-label text categorization problem by combining models and maximizing the F1-score, providing good statistical classifiers with generalization ability. Text categorization is a fundamental task in natural language processing, where each text document is assigned to one or more categories. The paper proposed a classifier design method based

on model combination and F1-score maximization. The method involved designing multiple models for binary classification per category and combining them to maximize the F1-score of the training dataset. The experimental results confirmed the usefulness of the proposed method, especially for datasets with many combinations of category labels. The direct mapping method, which models a map directly from data samples to the combination of assigned category labels, was experimentally shown to be inferior than the proposed method. The paper introduced the LRM-L classifier design approach, which estimated the discriminative function to maximize the F1-score of the training dataset using a logistic function. The paper also presented the model combination by micro-averaged F1-score maximization (MC-Fm) formulation, which used a Gaussian prior and a quasi-Newton method to estimate the discriminative function. Overall, the results suggested that the model combination based on the F1-score maximization approach is a valuable technique for improving the performance of multi-label text categorization models.

Finally, in another study Gonçalves and Quaresma [1] applied Support Vector Machines for text categorization. Two basic linguistic processes, stop-word removal and stemming/lemmatization, were implemented and assessed in multilabel text classification issues. The conventional vector representation, which is often used, loses order information and does not leverage syntactic or semantic information. Experiments were done on the Reuters and Portuguese legal datasets, utilizing the Porter method for stemming and a Portuguese lexical database, POLARIS, for lemmatization. The results indicated that the Reuters dataset performed well, but the PAGOD dataset had mixed results. Classifying abstract notions is more complicated and needs a more complex technique, such as employing word order and syntactic or semantic information. The study suggested exploring advanced document representations and feature selection strategies to enhance classifier performance and address class imbalance in binary classifiers.

We went through a large number of research papers and publications for this literature study. Some of those articles only used a specific model for sentiment analysis; another paper just considered political stance detection; and other research papers only included multi-label classification or hate offense detection. Not a single study contains all of the analysis and detection's from any of those studies. Most of their models does not work properly for Nepali language, also they didn't get proper performance. In previous work they didn't preprocess the Nepali data properly but we collected by ourselves and cleaned data very properly. However, in this study, we employed the majority of the best models for each of the four subtasks, including multi-label classification, political stance recognition, sentiment analysis, and hate Offense detection, all in a single report. In our paper, all our models (SVM, NB, RF, Bi-LSTM, Bi-GRU and BERT) are working fine for Nepali language. We got result as accuracy, precision, Recall, and F1-Score. They didn't use graphs or ROC in their, but in our paper we have included all those necessary graph and ROC curve.

Chapter 3

Methodology

The first step of our study is to collect data from popular and viral TikTok videos relating and originating from Nepal. So, we will built a web scraper using JavaScript and Python to scrape comments from the TikTok videos. After data scraping, we pre-processed the comments by removing URLs, @USER mentions, etc. Then, we annoatated the data by using Google Sheets.

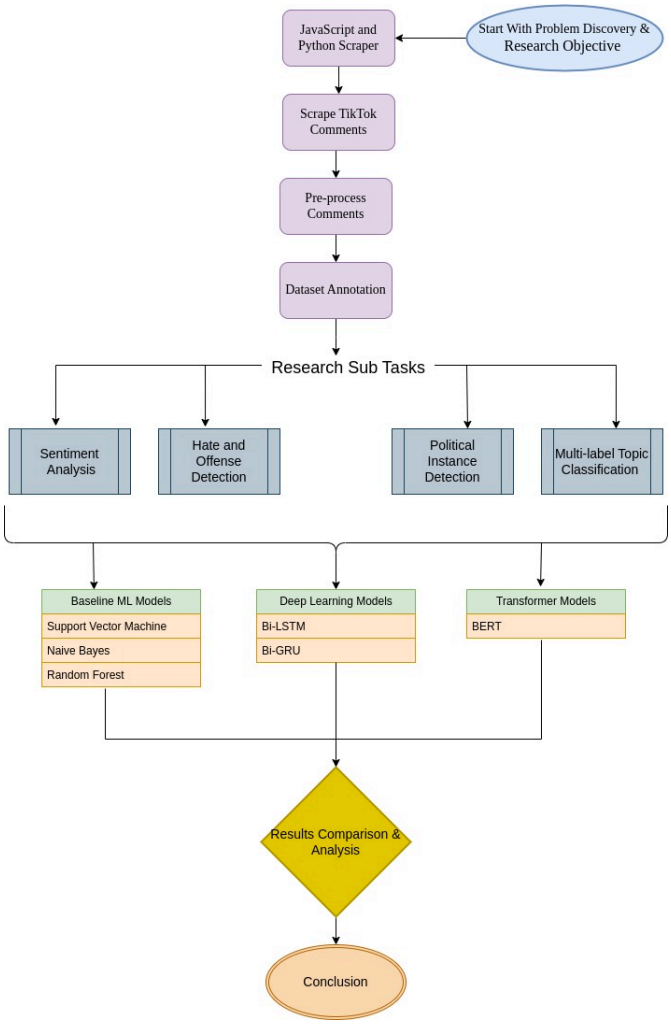


Figure 3.1: Workflow

After the data preparation, we experimented the data on conventional Machine

Learning algorithms like Support Vector Machine, Random Forest, Multinomial Naïve Bayes; Deep Learning models: Bi-LSTM Bi-GRU and Transformer models like BERT on our dataset. Finally, based on the result we selected the best performing models and analysis the performance for each task and came up with appropriate conclusion.

3.1 Data and Dataset Description

3.1.1 Dataset Description

First, we listed popular and diversified TikTok accounts from Nepal with high audience engagement. Selected account includes politicians, celebrities, athletes, artists, comedians, health workers, travel guides, chefs, etc along with the accounts posting news and updates on day to day activity of Nepal. In our data we tried to includes all the aspects of Nepali society. We used a Python and JavaScript based web scraping tool to collect captions and comments from the selected TikTok posts. Each team member scraped about 100,000 comments. After doing preliminary data analysis we found that the comments were in a variety of ethnic Nepali languages along with transliterated and code switched-mixed languages. For our study we filtered 65,000 comments in Nepali script written in Devanagari. From the pool of selected 65,000 comments we annotated 5,000 comments for this research.

Figure 3.2: Raw Data Collection showing all type of Combinations

3.1.2 Data Description

comment_text	SENTI	OFF	POL	PGS	FT	EYE	HBF	FR	EMP	BF	SW	DO
पनौती प्रचण्ड	1	1	1	1	0	0	0	0	0	0	0	0
भ्रष्टाचार गर्न भन्दा बरु खेल हरु मा लगाए राम्रो हुन्थ्यो ।अरु बेला सहयोग गर्ने होइन।चासो गर्ने होइन । जब गेम मा जन्छ तब नाटक देखाउन जानेका डाका।	1	1	1	1	0	0	1	0	0	0	0	0
राम्रो गर्नुस् देश तपाईहरुको साथमा छ।शुभकामना नेपाली टिम	2	0	1	1	0	0	1	0	0	0	0	0
खेलमा हार जित भै हाल्छ..!! फर्का नेपाली टिम अब जित्नु पर्छ ।	2	0	0	0	0	0	1	0	0	0	0	0
पनौती आए पछि	1	1	0	0	0	0	0	0	0	0	1	0

Figure 3.3: Sample Annotated Comments for four Tasks

To annotate the comments, the dataset was divided into two equal halves. Each half was annotated by a pair of annotators. Each comment was annotated by bilateral agreement between the pair, in case of disagreement a third annotator was asked to annotate the comment in question. Microsoft Excel was used to annotate the data.

- Text ID: This is a unique 16 digit identifier for the comment text.
- Comment Text: Contains the textual data to be annotated.
- Sentiment Column: Neutral (0), Negative (1), and Positive (2)

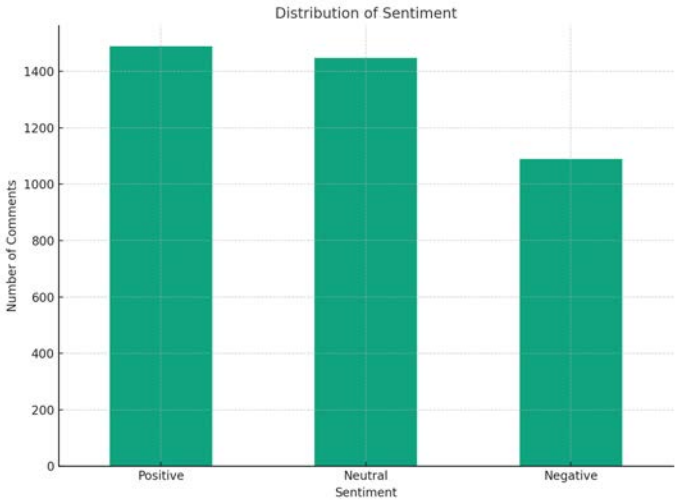


Figure 3.4: Distribution of Sentiment Comments

- Hate-Offensive Column: Not Hateful/Offensive (0), and Hateful/Offensive (1)

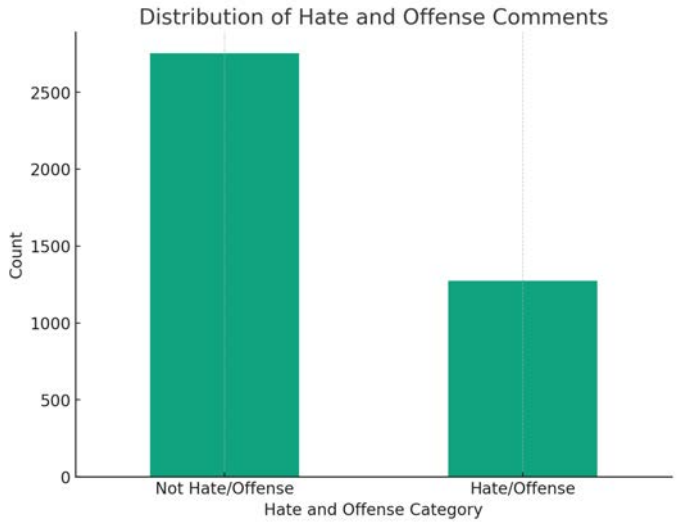


Figure 3.5: Distribution of Hate and Offence Comments

- Political Stance Column: Not Political (0) and Political (1)

  For Multi-label Topic Classification we divided the comments into 9 different categories.

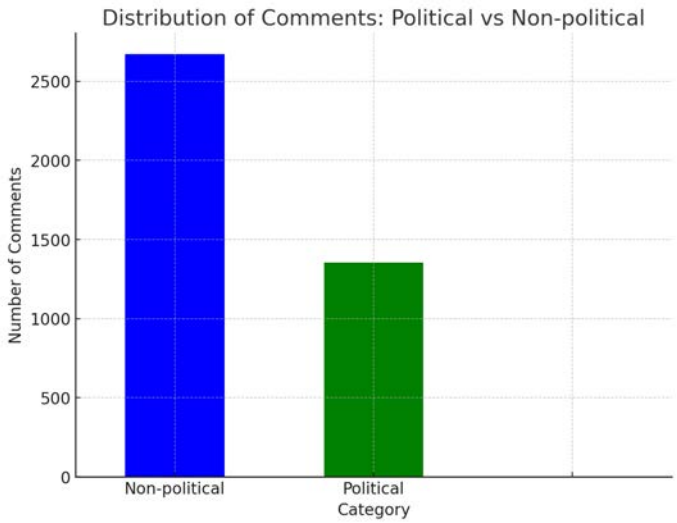


Figure 3.6: Distribution of Political and Non-Political Comments

- Politics, Governance, and Social Concern (PGS): These comments relate to national politics, day to day government affairs, social concerns, etc.
- Food and Travel (FT): Relating to cuisine, restaurants, grocery, tourism, etc.
- Education, Youth, and Employment (EYE): Relating to education, learning, youth, innovation, employment, etc.
- Health Beauty and Fitness (HBF): Includes makeup, fashion, gym culture, athleticism, medicine, etc.
- Family and Relationship (FR): Includes personal growth, family issues, marital issues, etc.
- Entertainment, Music, and Pop-Culture (EMP): Includes content from movies, music, celebrities, arts, drama, etc.
- Business and Finance (BF): About stock exchange, market production, national finance, banking, economic institutions, startups, business, etc.
- Spirituality and Well Being (SW): About religion, meditation, philanthropy, personal beliefs, etc.
- Daily Dairies and other Hobbies (DO): Daily mini vlogs, lifestyle, personal interests, etc.

In the all the tasks, our dataset was imbalanced. To solve this problem we used Synthetic Minority Over-sampling Technique (SMOTE). It generates the sample for the minor class based on interpolating the minority class by selecting two or more stances for generating the new one. By doing SMOTE the accuracy will improve because and help proper model training and testing. This technique avoids the duplication of data, by providing a new sample of data, and reduces the risk of over-fitting to some extent.

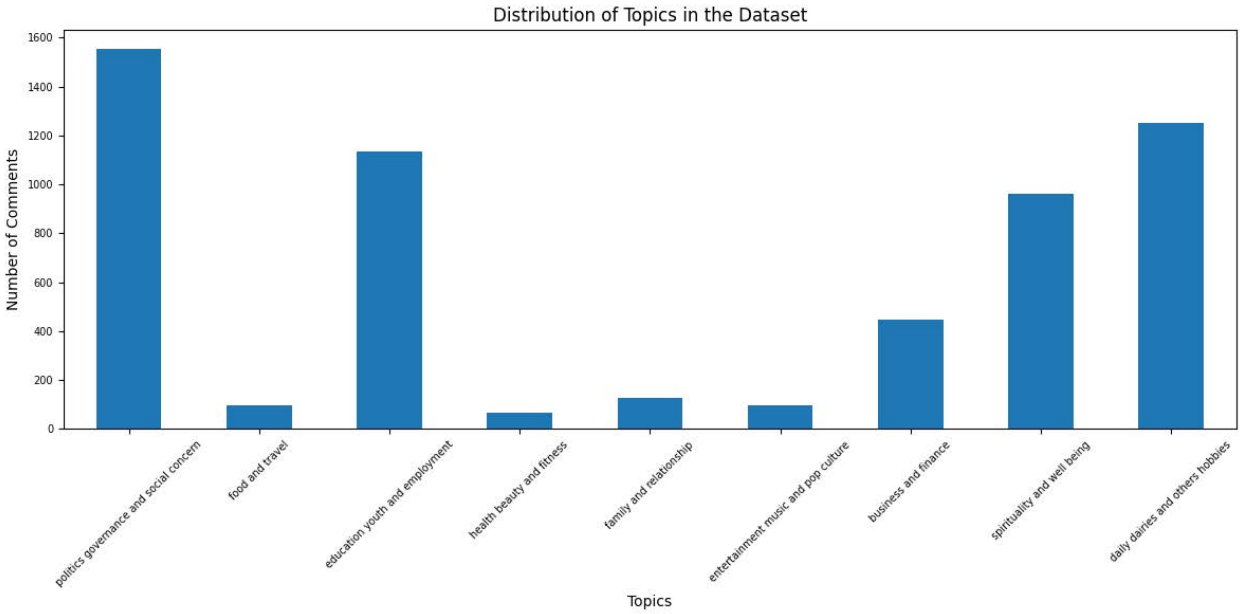


Figure 3.7: Distribution of Multi-label Comments

Figure 3.8: Wordcloud of frequently used words

3.1.3 Data Prepossessing

Natural Language Processing for Nepali-language requires a number of important stages, data collection being the first step. After that, text cleaning was done to get rid of any non-textual components, emoticons, and unnecessary characters.

To concentrate on more significant content, stop words were also removed by using NLTK. The Nepali text was tokenized, breaking it down into individual words. Similarly to normalize words to their base forms we did stemming and lemmatization. On average each comment was seven to ten words.

The dataset was split into training (80%) and testing (20%) sets. After that the text was transformed into numerical vectors using TF-IDF, for machine learning models.

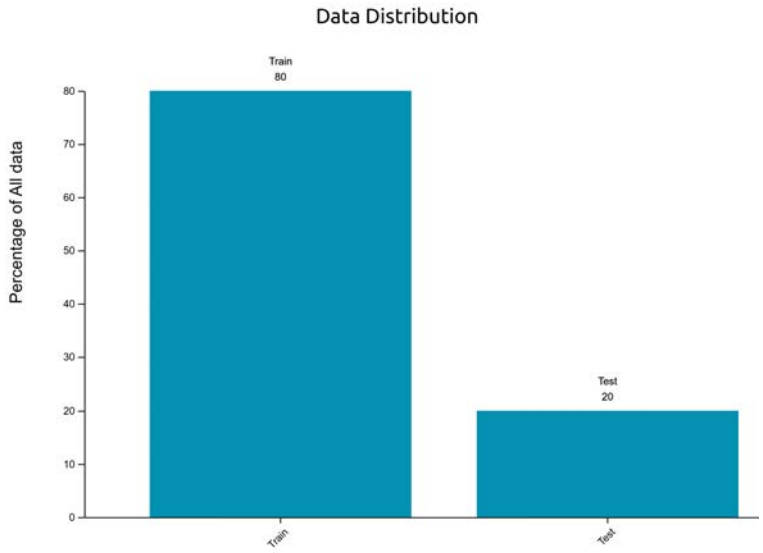


Figure 3.9: Train Test split of Dataset

3.2 Baseline Machine Learning Models

For this research, we used three Machine Learning models: Support Vector Machine (SVM), Naïve Bayes (NB) and Random Forest (RF).

3.2.1 Support Vector Machine (SVM)

Support Vector Machine(SVM) is utilized for tasks involving classification and regression. It operates by identifying the hyperplane that effectively separates the classes within the feature space. Our initial implementation involved creating a classifier known as Linear SVM, which utilizes a Bag of Words (BoW) features weighted by Term Frequency Inverse Document Frequency (TF-IDF) [11]. TF-IDF proves to be advantageous as it overcomes the limitations of word count feature, which only considers the word frequency without taking into account the context or sentiment of the sentence. It achieves this by reducing the significance of terms that appear in all documents while amplifying those that are rare. We used cross validation and grid search to find optimal hyper-parameters for our SVM model. We need to build a pipeline so that we do not get features from the validation folds when building each training model.

To accomplish this we employed the TF-IDF Vectorizer and Linear Support Vector Machine classifier (SVC) from the Scikit-learn library. We retrained this model using validation techniques specifically utilizing K-fold cross validation. This method involves dividing the dataset into k sections where each part serves for testing while others are utilized for training purposes [21]. This process is repeated k times with each section being tested. The results, from each iteration can then be averaged to

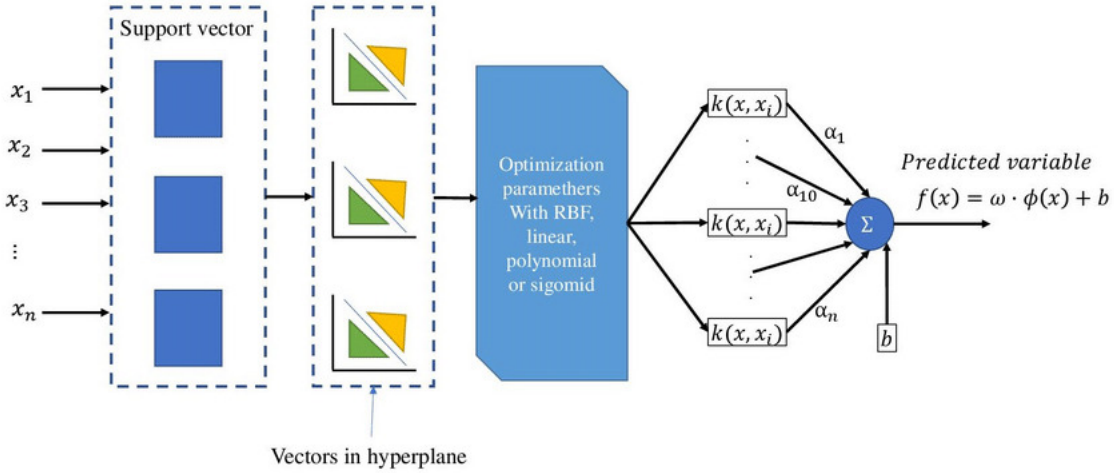


Figure 3.10: Architecture of SVM [9]

provide an estimation.

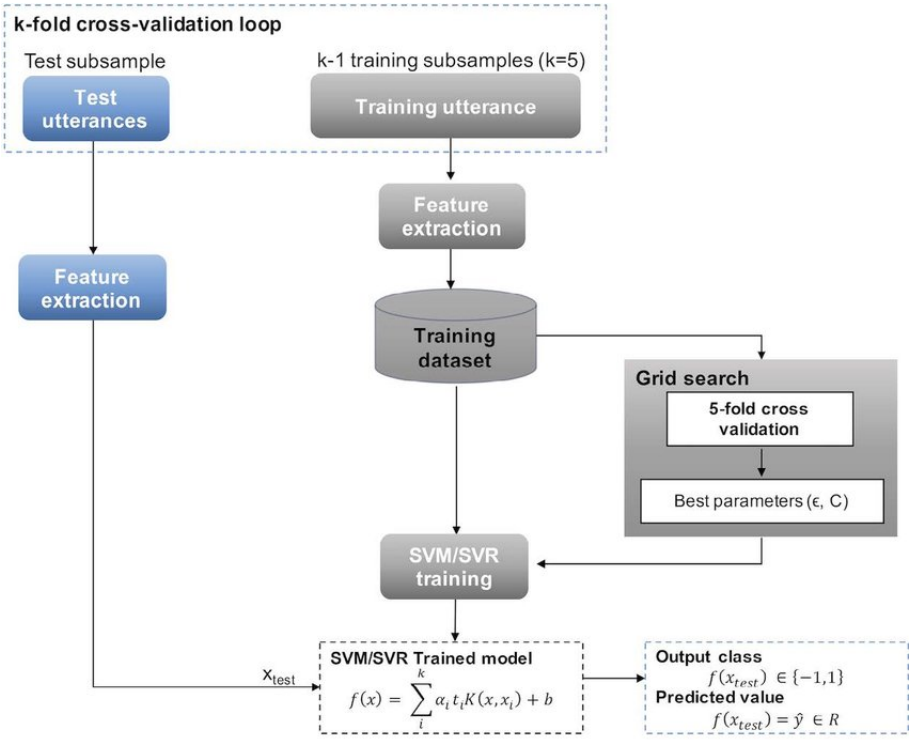


Figure 3.11: K fold cross validation [21]

Hyperplane and Decision Boundary

In SVM, the goal is to find the optimal hyperplane that maximizes the margin between the classes. For binary classification problem, the decision boundary is defined by the hyperplane equation:

$$\mathbf{w}^\top \mathbf{x} + b = 0 \tag{3.1}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and b is the bias term.

Margin

The margin is the distance between the hyperplane and the nearest data point from each class. The objective of SVM is to maximize this margin. The margin γ can be expressed as:

$$\gamma = \frac{2}{\|\mathbf{w}\|} \tag{3.2}$$

Optimization Problem

To find the optimal hyperplane, we need to solve the following optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.3}$$

subject to the constraints:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \forall i \tag{3.4}$$

where y_i are the class labels ($y_i \in \{1, -1\}$) and $\mathbf{x}_i$ are the training examples.

Lagrangian and Dual Form

By introducing Lagrange multipliers α_i, the optimization problem can be transformed into its dual form:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{3.5}$$

subject to the constraints:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0 \quad \forall i \tag{3.6}$$

Support Vectors

The data points for which $\alpha_i > 0$ are called support vectors. These points lie on the margin and are crucial in defining the hyperplane.

Kernel Trick

For non-linearly separable data, the kernel trick can be implemented to map the input features into a higher-dimensional space where a linear separation is possible. Common kernels include:

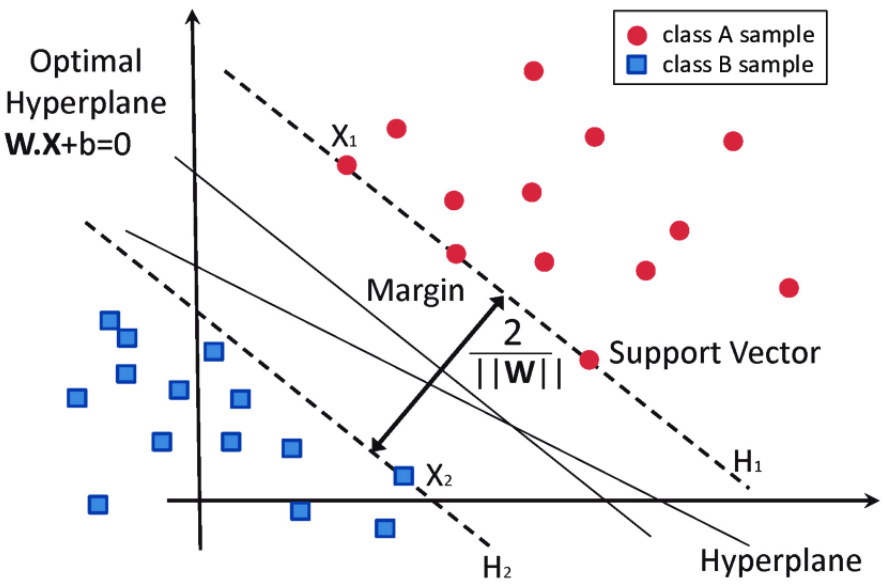


Figure 3.12: SVM Classification with Optimal Hyperplane and Margins [26]

- **Linear Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j \tag{3.7}$$

- **Polynomial Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d \tag{3.8}$$

- **Radial Basis Function (RBF) Kernel**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \tag{3.9}$$

Decision Function

The decision function for classifying a new data point $\mathbf{z}$ is:

$$f(\mathbf{z}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right) \tag{3.10}$$

3.2.2 Naïve Bayes (NB)

The Naïve Bayes model is a probabilistic classifier based on Bayes' Theorem, which provides a principled way of calculating the probability of a hypothesis given prior knowledge [8].

Bayes' Theorem

Bayes' Theorem states:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)} \tag{3.11}$$

where:

- $P(c|x)$ is the posterior probability of class c given feature vector x.
- $P(x|c)$ is the likelihood of feature vector x given class c.
- $P(c)$ is the prior probability of class c.
- $P(x)$ is the marginal likelihood of feature vector x.

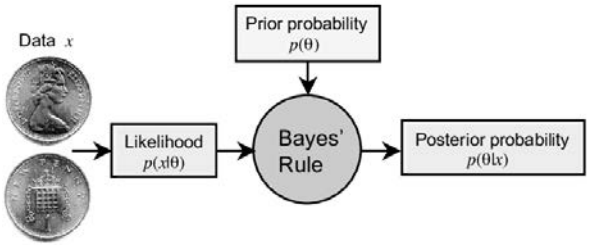


Figure 3.13: Bayes Theorem[17]

Prior Probability

The prior probability of class C_k is given by:

$$P(C_k) = \frac{\text{Number of samples in class } C_k}{\text{Total number of samples}} \tag{3.12}$$

Assumption of Conditional Independence

The model assumes that the features are conditionally independent given the class label, simplifying the computation to:

$$P(c|x) \propto P(c) \prod_{i=1}^{n} P(x_i|c) \tag{3.13}$$

where x_i is the i-th feature.

Types of Naïve Bayes Models

Gaussian Naïve Bayes (GNB)

Gaussian Naïve Bayes is used for continuous data and assumes that the features follow a Gaussian (normal) distribution. It assumes that the continuous values associated with each class are distributed according to a Gaussian distribution and the features are conditionally independent given the class label and that each feature follows a normal distribution [15]. It trains by computing the prior probability for each class based on the training data and for each feature in the dataset computes the mean and variance for each class.

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left(-\frac{(x_i - \mu_c)^2}{2\sigma_c^2}\right) \tag{3.14}$$

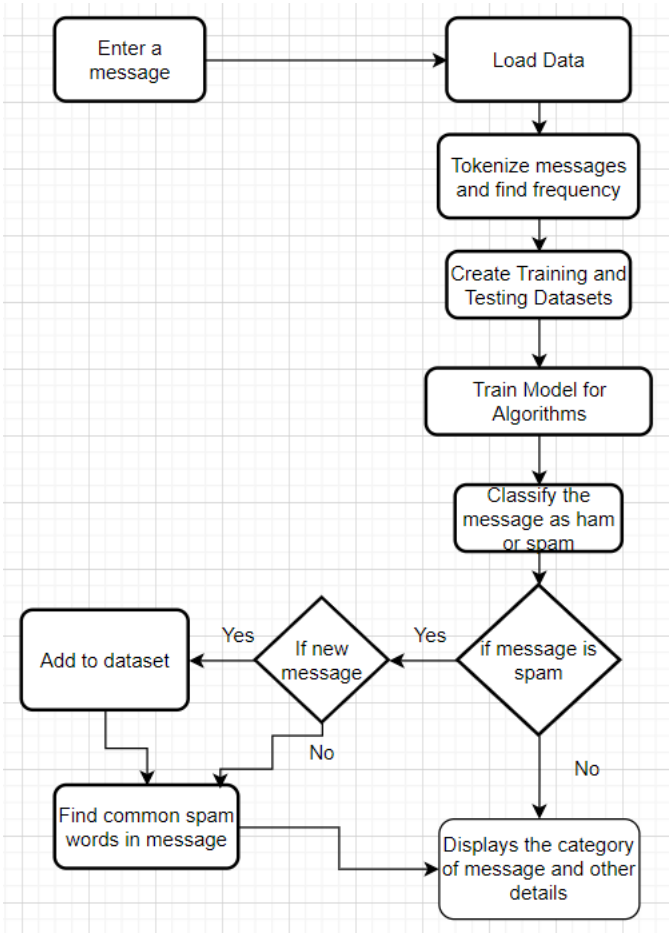


Figure 3.14: Architecture of Naïve Bayes [37]

The overall class probability is computed as:

$$P(c|x) \propto P(c) \prod_{i=1}^{n} P(x_i|c) \tag{3.15}$$

Multinomial Naïve Bayes (MNB)

We introduce the Multinomial Naïve Bayes classifier as it simplifies assumption about how the features interact. MNB performs well with discrete data, typically used for text classification purposes. It is based on the multinomial distribution, which describes the probability of observing a certain set of counts across a fixed number of categories [16]. It calculates the probability of a particular feature vector given a class by counting the occurrences of each feature in the training dataset and assumes that the features are conditionally independent given the class label. Compute the prior probability for each class based on the training data and conditional probability of each feature given each class. Assign the class with the highest posterior probability to the document by calculating the posterior probability for each class using the prior and the conditional probabilities.

$$P(c|d) \propto P(c) \prod_{i=1}^{n} P(w_i|c) \tag{3.16}$$

where w_i is the i-th word in the document d.

TF-IDF Vectorizer

The TF-IDF Vectorizer is used to convert a collection of raw documents into a matrix of TF-IDF features.

- **Term Frequency (TF)**: Measures how frequently a term t appears in a document d.

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{3.17}$$

- **Inverse Document Frequency (IDF)**: Measures how important a term is by weighing down the frequent terms while scaling up the rare ones.

$$\text{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right) \tag{3.18}$$

- **TF-IDF Score**: The product of TF and IDF.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{3.19}$$

3.2.3 Random Forest (RF)

The Random Forest technique is a type of collaborative learning approach that creates decision trees while training and then determines the common class (for classification) or the average prediction (for regression) from the individual trees [12]. Each tree, within the forest is trained on a subset of the data using bootstrapping. Makes its predictions autonomously.

Decision Tree Prediction

A decision tree T makes predictions $\hat{y}$, for an input vector $\mathbf{x}$ by dividing the feature space step by step based on rules leading to a prediction at a leaf node [13]. The prediction process, in a decision tree can be illustrated as follows;

$$\hat{y} = T(\mathbf{x}) \tag{3.20}$$

In the context of a Random Forest, these individual tree predictions are aggregated to produce the final output.

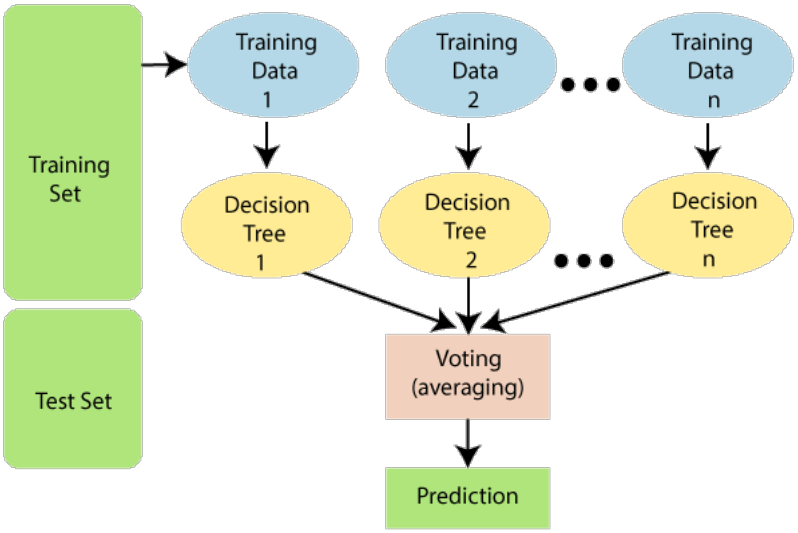


Figure 3.15: Architecture of RF [45]

Random Forest Prediction

For classification, the final prediction $\hat{y}$ is determined by majority voting:

$$\hat{y} = \text{mode}\{T_1(\mathbf{x}), T_2(\mathbf{x}), \ldots, T_n(\mathbf{x})\} \tag{3.21}$$

For regression, the final prediction $\hat{y}$ is the average of the predictions from all trees:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} T_i(\mathbf{x}) \tag{3.22}$$

where $T_i(\mathbf{x})$ is the prediction of the i-th tree.

Ensemble Learning and Prediction

The concept of enhancing performance by merging models is known as ensemble learning. Random Forest falls under this category. Ensemble learning works by bringing the predictions of individual models (known as base learners) to enhance overall performance. Lets say we have a group of base learners represented by $H_1, H_2, \ldots, H_n$ and an input vector denoted as $\mathbf{x}$. This method of learning merges the predictions made by these base learners to generate the prediction $\hat{y}$.

$$\hat{y} = \text{Ensemble}(H_1(\mathbf{x}), H_2(\mathbf{x}), \ldots, H_n(\mathbf{x})) \tag{3.23}$$

The function Ensemble$(\cdot)$ aggregates the predictions of the base learners, which could involve averaging, voting, or other aggregation methods depending on the task (classification, regression) and the ensemble algorithm (e.g., Random Forest, Gradient Boosting, etc.).

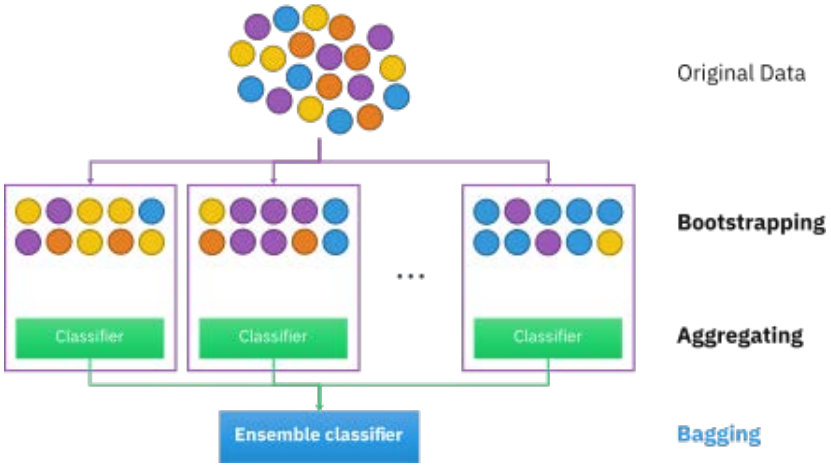


Figure 3.16: Bagging Ensemble [47]

Bootstrap Aggregating (Bagging)

Bootstrap aggregating, also referred to as bagging is a technique, in learning that seeks to enhance the reliability and precision of machine learning models by averaging or merging base models trained on various portions of the training data.

Bootstrap Aggregating Process

Create B bootstrap samples $D_1^*, D_2^*, \ldots, D_B^*$ by selecting data from the training dataset D. Develop a base model M_i for each bootstrap sample D_i^*. Merge the forecasts of all base models to produce the prediction.

Bootstrap Aggregating Equation

Let $M_i(\mathbf{x})$ denote the prediction of the i-th base model for the input vector $\mathbf{x}$. The final prediction $\hat{y}$ using bootstrap aggregating can be expressed as:

$$\hat{y} = \frac{1}{B} \sum_{i=1}^{B} M_i(\mathbf{x}) \tag{3.24}$$

Random Feature Selection

At each split in the decision tree, a random subset of features is selected to determine the best split [14]. This randomness helps to ensure that the trees are not too correlated.

Let p be the total number of features in the dataset, and m be the number of features randomly selected at each split [47]. The random feature selection process can be represented as follows:

$$\text{Random Subset}(p, m) = \{f_{j_1}, f_{j_2}, \ldots, f_{j_m}\} \tag{3.25}$$

where $f_{j_1}, f_{j_2}, \ldots, f_{j_m}$ are m randomly selected features from the total p features.

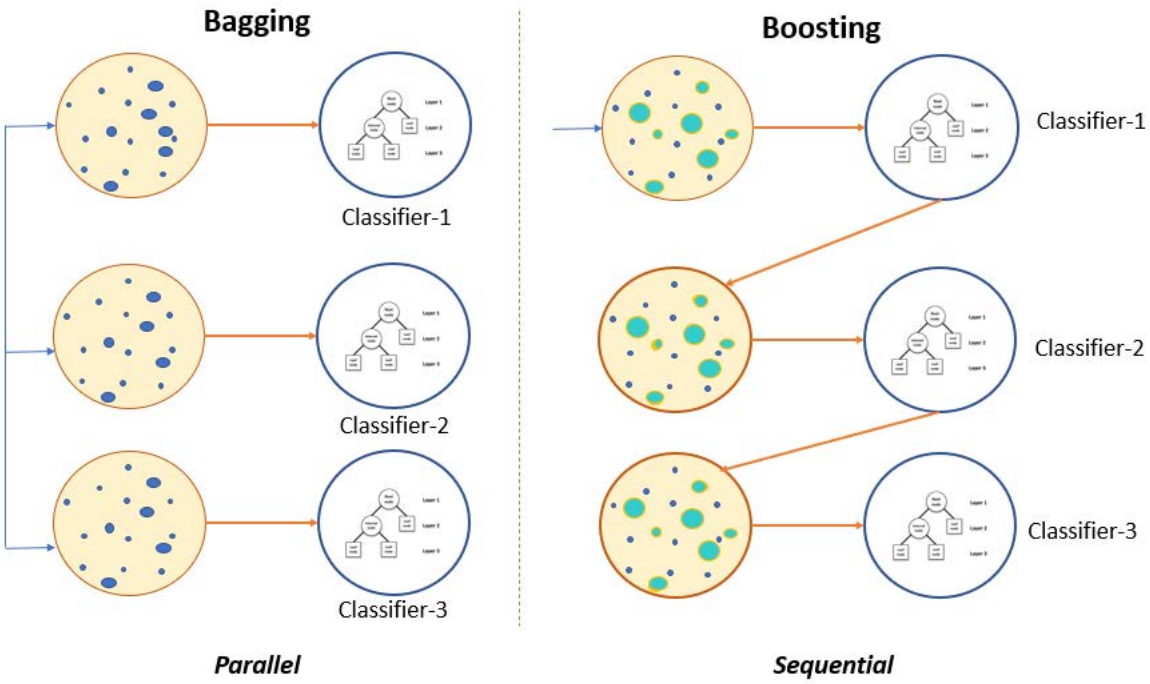


Figure 3.17: Bagging-Boosting [47]

3.3 Deep Learning Models

The branch of Machine Learning which Deals with neurons with many layers is known as the Deep Neural Network. These networks similarly to human brain neurons and are designed to simulate the information, enabling computers to recognize patterns and make suitable decisions with minimum amount of human input. Deep Neural networks can extract information from high range of data, hierarchical complex representation of data, and compute certain tasks such as Speech Recognition, Natural Language Processing, and automate systems. The Deep Learning models have been the key of revolution in the NLP field which offers powerful tools like Sentiment Analysis. The branches of Deep learning models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Transformers are the tool which we can use it for textual data representation and evaluation.

Nowadays, Machine Learning as well Deep Learning is growing exponentially in applications such as Machine Translation, Sentiment Analysis, Chat bots, and Text generation. In deep learning branches, RNNs, CNNs, LSTMs can learn human language properly and do languages tasks based on their learning. These models first convert the language sentences into tokens and then the tokens transliterated into vector using certain word embedding technique like Word2Vec, GloVe to understand the semantic information and nuances of words. The models can take input from these word embedding, learn and make predictions based on sequential semantic information. This whole process allows Deep Learning models to perform complex language tasks with high performance metrics, which makes them one of the essential tool in Natural Language Processing.

Simple Recurrent Neural Networks (RNNs) exhibit substantial constraints, espe-

cially in addressing long-term dependencies due to difficulties like vanishing and ballooning gradients, which impair their ability to retain information over longer sequences. This makes them less effective in activities needing context from earlier in the series. Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU) networks overcome these problems by integrating techniques to better manage long-term dependence. They achieve this through gating mechanisms that control the flow of information and maintain context over prolonged duration. Additionally, their bidirectional architecture analyzes input in both forward and backward directions, gathering context from both past and future states, hence boosting the model's capacity to learn and anticipate complex sequential data more correctly.

We used two Deep Learning Models: Bidirectional Long Short Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU).

3.3.1 Bi-LSTM & Bi-GRU

LSTM

Long Short-Term Memory (LSTM) networks excel in handling language tasks due to their ability to manage long-term dependencies in sequential data. The architecture includes three types of gates—forget, input, and output—that control the flow of information through the network. These gates use sigmoid and tanh activation functions to selectively retain or discard information, allowing LSTMs to maintain context over extended sequences, crucial for tasks like Language modeling, Translation, and Sentiment Analysis [7]. The forget gate determines which information from the previous cell state should be discarded, the input gate decides which new information should be stored in the current cell state, and the output gate controls what information from the current cell state should be outputted [32]. This gating mechanism helps LSTMs handle the vanishing gradient problem by ensuring that gradients remain within a useful range during back-propagation [20].

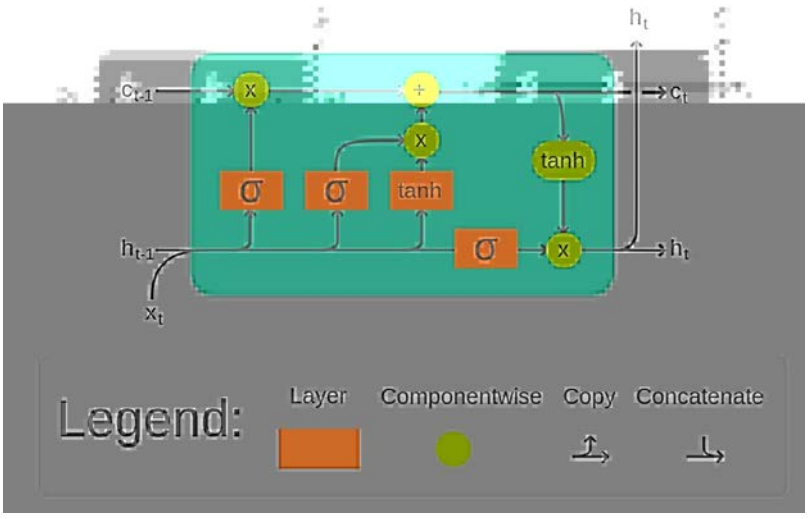


Figure 3.18: Architecture of LSTM [44]

GRU

A particular type of recurrent neural network is the Gated Recurrent Unit (GRU). The GRU is a more recent development in recurrent neural networks and is somewhat similar to the LSTM. It simplifies the model by combining the forget gate and the input gate into a single update gate, thus simultaneously mixing the hidden state and the cell state. The basic architecture of the GRU is shown in the figure below. Due to its fewer parameters and simpler structure, the GRU is less prone to over fitting during training and reduces the training time [40]. The GRU has two gates: the reset gate (r) and the update gate (z). The update gate controls how much of the previous memory to retain, while the reset gate determines how to combine the current input with the previous memory. By setting the reset gate to all 1s and the update gate to all 0s, the GRU can revert to the basic RNN.

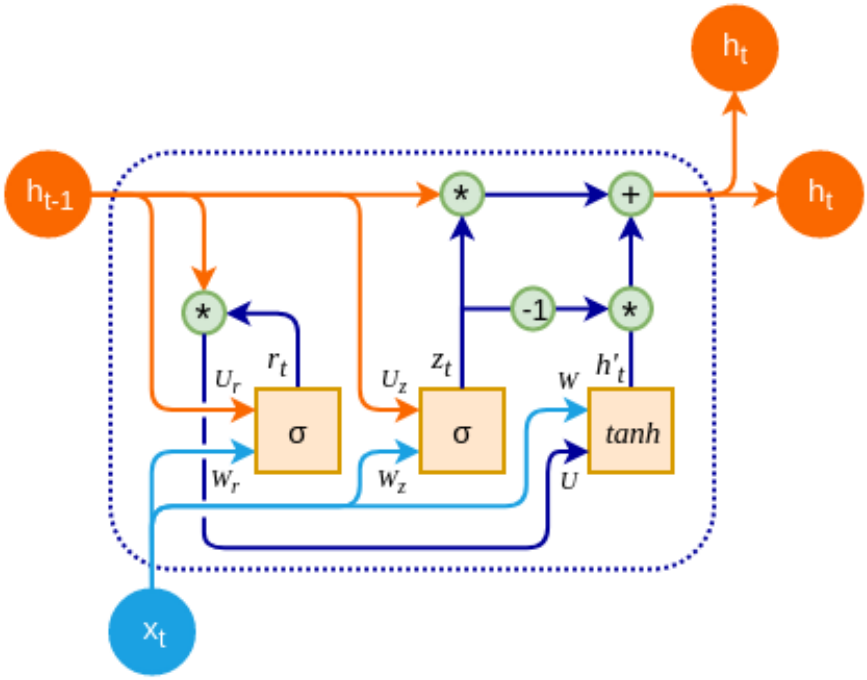


Figure 3.19: Architecture of GRU [48]

Bi-LSTM and Bi-GRU

Recurrent neural networks, such as Gated Recurrent Units (GRUs), are becoming more and more popular because of their efficiency and more straightforward design than Long Short-Term Memory (LSTM) networks. Even though LSTMs are good at managing long-term dependencies in sequential data, in some situations their predictive power is limited since they can only analyze data from previous states. In order to overcome this restriction, bidirectional LSTMs, or Bi-LSTMs, use two LSTMs: one processes the input sequence from start to finish, while the other processes it from end to start. Bi-LSTMs can now collect context from both past and future states, which enhances their performance in language challenges where it's important to grasp both words that come before and after a word [19] and minaee2019deep. Likewise, GRUs can be set up in a bidirectional fashion using

their reset and update gates, offering the same benefit of processing sequences in both ways. This dual strategy improves the model's capacity to produce a more thorough knowledge of the input data, resulting in improved predictions and richer representations, as shown in figure 3.6 [19]. From both forward and backward passes, Bi-LSTMs and Bi-GRUs produce probability vectors; the combined output of these vectors makes use of the entire input sequence's context.

Architecture of Bi-LSTM

In our endeavor to perform sentiment analysis on Nepali sentences, we harnessed the power of GloVe embeddings, augmented by NLTK tokenization, to imbue our model with a nuanced understanding of contextual semantics [5], [29]. GloVe's methodology, rooted in co-occurrence statistics, facilitates the construction of word-context matrices, enabling the model to discern semantic relations and capture the intricate nuances of language usage. This embedding technique empowers our model to comprehend the contextual meanings of words and their interrelationships within sentences, laying a robust foundation for accurate sentiment analysis and multi-label classification tasks.

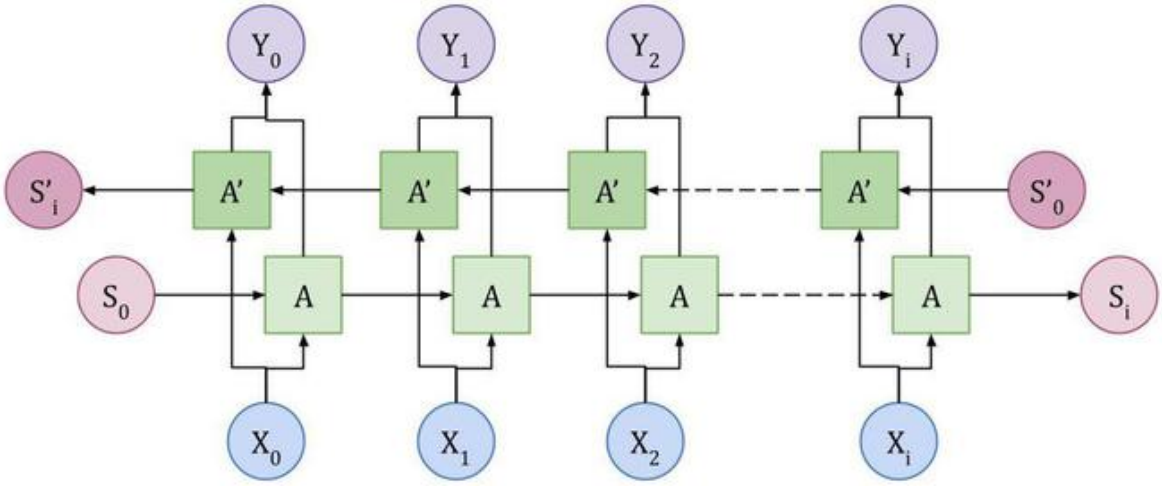


Figure 3.20: Architecture of Bidirectional LSTM [42]

Our approach further integrates Bidirectional LSTMs (Bi-LSTMs), a sophisticated architecture that processes input sequences from both forward and backward directions [38]. By leveraging the bidirectional nature of these LSTMs, our model comprehensively captures the contextual nuances of the input text, enhancing its ability to discern sentiment and classify topics effectively. With a hidden layer size of 2h in the bidirectional setup, the LSTMs adeptly preserve information from both preceding and subsequent words, thereby enriching the model's understanding of the input sequence [36]. Moreover, employing sigmoid activation for binary tasks and softmax for multi-class problems, the classification layer generates probability distributions for each class, facilitating precise classification outcomes. This holistic architecture, integrating GloVe embeddings with Bi-LSTMs, equips our model to deliver robust sentiment analysis and multi-label topic classification, thereby advancing the capabilities of natural language processing in Nepali text analysis.

Sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \qquad \text{where } z \text{ is the input} \tag{3.26}$$

Softmax:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, \qquad \text{where } z_i \text{ is the input for class } i \tag{3.27}$$

The detailed workings of the Bi-LSTM can be understood through its core equations. The forget gate uses a sigmoid function to produce a value between 0 and 1, determining the extent to which the previous cell state should be retained or forgotten. The input gate decides which new information is added to the cell state, using a sigmoid function to act as a gate and a tanh function to provide the candidate values. The cell state is updated by blending the retained old information and the new candidate values. Finally, the output gate uses a sigmoid function to decide what part of the cell state should be output, and a tanh function to scale the cell state values to ensure they fall within a suitable range. These operations collectively enable LSTMs to effectively process and understand complex language data by maintaining and updating context information over long sequences.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.28}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.29}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.30}$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.31}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.32}$$
$$h_t = o_t \odot \tanh(c_t) \tag{3.33}$$

Where:

f_t, i_t, o_t are the forget, input, and output gates, respectively.
$\tilde{c}_t$ is the candidate cell state.
c_t is the cell state.
h_t is the hidden state/output.
W_f, W_i, W_o, W_c are weight matrices.
b_f, b_i, b_o, b_c are bias vectors.
$[h_{t-1}, x_t]$ denotes concatenation of h_{t-1} and x_t.
σ denotes the sigmoid activation function.
$\odot$ denotes element-wise multiplication.

Architecture of Bi-GRU

A Bidirectional Gated Recurrent Unit (Bi-GRU) is designed using two GRU networks that process input sequences simultaneously in opposing directions. The bidirectional feature of the model enhances its comprehension and prediction of sequences by enabling it to capture context from both past and future states. To enable the selective updating and forgetting of information, the model uses two sets

of gates—update and reset gates—in each direction. This architecture enables Bi-GRU networks to perform tasks like sequential data analysis and natural language processing that need deep context knowledge.

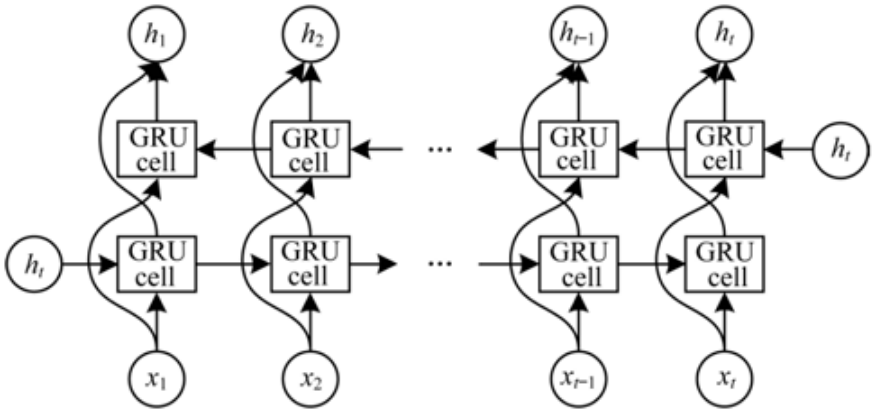


Figure 3.21: Architecture of Bi-GRU [40]

Update Gate
The update gate in a Bidirectional Gated Recurrent Unit (Bi-GRU) operates similar to the input and forget gates in an LSTM, facilitating the selective updating of information within the model. Utilizing the sigmoid activation function, this gate regulates the flow of data by determining which values to retain or discard based on their relevance. The update gate z_t is calculated using the current input x_t and the previous hidden state h_{t-1}:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.34}$$

Where:

- W_z is the weight matrix for the update gate corresponding to the input x_t.
- U_z is the weight matrix for the update gate corresponding to the hidden state h_{t-1}.
- b_z is the bias vector for the update gate.
- σ is the sigmoid activation function.

Reset Gate
The reset gate, another crucial component of the Bi-GRU, plays a pivotal role in regulating the retention or forgetting of prior information. By modulating the influence of the previous hidden state on the current state, the reset gate enables the model to manage sequences with long-term dependencies effectively [46]. The update gate r_t is calculated using the current input x_t and the previous hidden state h_{t-1}:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.35}$$

Where:

- W_r is the weight matrix for the reset gate corresponding to the input x_t.
- U_r is the weight matrix for the reset gate corresponding to the hidden state h_{t-1}.
- b_r is the bias vector for the reset gate.
- σ is the sigmoid activation function.

To compute the candidate hidden state $\tilde{h}_t$ in a Gated Recurrent Unit (GRU), we first derive parameterized input and hidden state vectors for each gate by performing element-wise multiplication between the relevant vectors and their corresponding gate weights. A notable distinction lies in the calculation of the Current Memory Gate [43], where the Reset Gate and the previous hidden state vector undergo element-wise multiplication. Subsequently, this resulting vector is parameterized and added to the existing input vector, which has also been parameterized. The formula for the candidate hidden state $\tilde{h}_t$ in a GRU is expressed as:

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{3.36}$$

Where:
- $\tilde{h}_t$ is the candidate hidden state at time step t.
- W_h is the weight matrix associated with the candidate hidden state.
- r_t is the reset gate at time step t.
- h_{t-1} is the previous hidden state.
- x_t is the current input.
- b_h is the bias term for the candidate hidden state.
- $\odot$ denotes element-wise multiplication.
- $[\cdot, \cdot]$ denotes the concatenation of vectors.
- tanh is the hyperbolic tangent activation function.

To compute the current hidden state h_t in a Gated Recurrent Unit (GRU), a vector of ones, denoted as 1 in mathematics and having the same dimensions as the input, must first be defined. The update gate is then multiplied element-wise by the previous hidden state vector. Next, this result is subtracted from the vector of ones to generate a new vector, which is subsequently multiplied element-wise by the current value of the memory gate. Finally, the two resulting vectors are summed to obtain the current hidden state vector. The formula for the hidden state h_t in a GRU is articulated as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{3.37}$$

Where:
- h_t is the hidden state at time step t.

- z_t is the update gate at time step t.
- h_{t-1} is the previous hidden state.
- $\tilde{h}_t$ is the candidate hidden state at time step t.
- $\odot$ denotes element-wise multiplication.

Bi-LSTM and Bi-GRU in Nepali Data

Bidirectional Gated Recurrent Unit (Bi-GRU) and Bidirectional Long Short-Term Memory (Bi-LSTM) architectures provide notable benefits in multi-label topic categorization, hate crime detection, and Nepali sentiment analysis applications. By processing text sequences bidirectionally, these models can better comprehend the contextual meaning of words and phrases in Nepali sentiment analysis, capturing the subtleties of sentiment expressions. Because of this bidirectional approach, the models are better able to reliably determine sentiment by taking into account both words that come before and after them. Furthermore, by utilizing their bidirectional processing abilities, the Bi-LSTM and Bi-GRU models perform exceptionally well in the identification of hate crimes by recognizing derogatory language and hate speech. This enables them to identify and classify offensive content more precisely by capturing minute linguistic clues and contextual data.

To sum up, Bi-LSTM and Bi-GRU architectures excel in multi-label topic classification problems by effectively managing sequences with numerous labels or categories. Because of their bidirectional nature, they may capture topic relationships in both forward and backward directions, which makes it easier to comprehend the input text in its entirety. This comprehensive method improves the models' capacity to reliably classify text into many subjects. Furthermore, bidirectional processing helps both Bi-LSTM and Bi-GRU overcome the vanishing gradient issue, which keeps long-range relationships intact and enables them to efficiently gather contextual data for the duration of the sequence. In general, the bidirectional architectures of Bi-LSTM and Bi-GRU provide strong solutions for multi-label topic classification, hate crime detection, and sentiment analysis in Nepali, offering more precise and nuanced insights into the nepali data.

3.4 Transformer Model

Transformers are a type of deep learning architecture that has revolutionized the field of Natural Language processing. Published in the paper, "Attention is All You Need" by Vaswami et al. in 2017, [41] transformers have become the foundational architecture for many state of the art NLP models.

Self-attention Mechanism
At the core of transformers lies the self-attention mechanism. Self-attention allows the model to weigh the importance of different words in a sentence relative to each other by computing a score for each word in relation to every other word in the input sequence, which allows the model to capture long-range dependencies and contextual relationships more efficiently that traditional RNNs or CNNs.

Encoder-Decoder Architecture

The encoder processes the input sequence and produces a set of context-aware representations. It contains multiple identical layers, each layer consists of two sub-layers: a multi-head self-attention mechanism and position-wise fully connected feed-forward network. The decoder generates the output sequence from the encoder's representations. It also consists of multiple identical layers, but each layer has an additional sub-layer that performs attention over the encoder's output.

Positional Encoding
Since transformers lack understanding of the sequential order of words, positional encodings are added to the input embeddings to provide information about the position of each word in the sequence.

Multi-head Attention
Multi-head attention allows the model to focus on different parts of the the input sequence simultaneously from multiple perspectives, this improves the model's ability to focus on various aspects of the data.

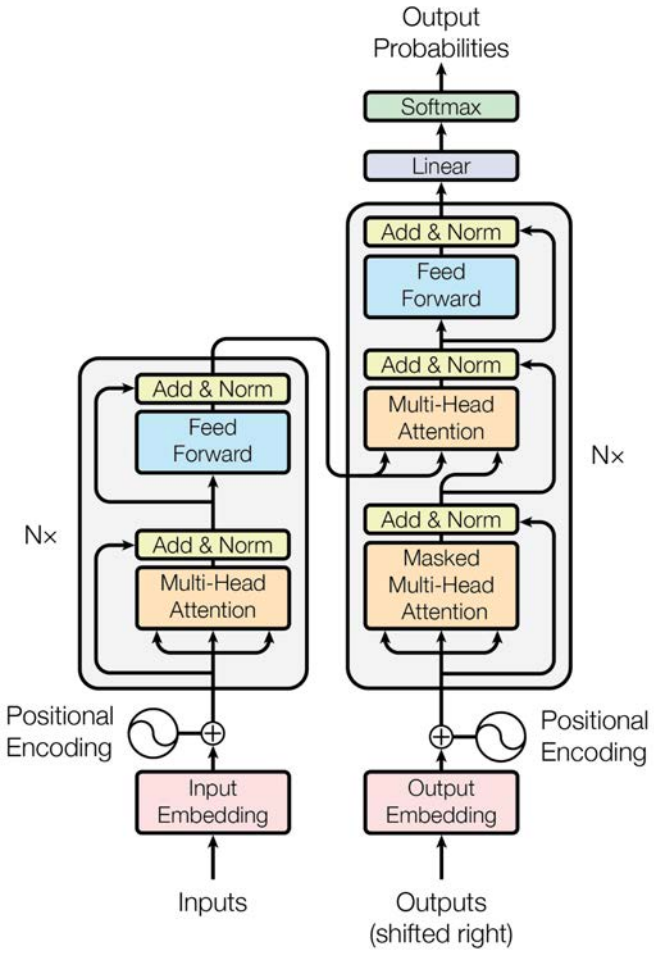


Figure 3.22: Architecture of Transformer [41]

3.4.1 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a transformer based model introduced in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. [18] BERT has achived significant results accross various NLP tasks by pre-training a large corpora and fine tuning on specific tasks.

BERT is pre-trained on a large corpus of text using two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM BERT randomly masks some words in the input sequence and tasks the model with predicting the masked words based on the surrounding context. This encourages the model to learn bidirectional representations of the text. NSP BERT is trained to predict whether two sentences in a pair are consecutive in the original text or not. This helps BERT capture relationships between sentences and understand discourse-level information.

BERT consists of a stack of transformer encoder layers. The transformer architecture allows BERT to capture contextual information effectively through self-attention mechanisms. BERT utilizes a multi-layer bidirectional Transformer encoder, where each layer has a fixed number of self-attention heads and feed-forward neural network layers. BERT's transformer architecture enables it to capture complex linguistic patterns and dependencies in the input text.

BERT tokenizes input text into subword units using a WordPiece tokenizer. This allows BERT to handle out-of-vocabulary words and improve generalization.

After pre-training, BERT can be fine-tuned on downstream NLP tasks such as text classification, named entity recognition, question answering, and more. During fine-tuning, BERT is typically used as a feature extractor, where the pre-trained model is frozen, and task-specific layers (e.g., classification layer) are added on top of BERT's output representation. Fine-tuning BERT on task-specific datasets allows the model to adapt its representations to the specific task, resulting in improved performance.

BERT captures contextual information from both left and right contexts of each word in the input sequence, enabling it to understand the meaning of words in context. BERT's pre-trained representations can be fine-tuned on downstream tasks with limited labeled data, leading to improved performance compared to training models from scratch.

Chapter 4

Result Analysis

For calculating the performance of the four tasks that include Sentiment Analysis, Hate-Offense Detection, Political Stance Detection, and Multi-label Topic Classification. We used Machine Learning, Deep Learning, and Transformers models. First, we trained and tested Machine Learning models: Support Vector Machine, Multinomial Naïve Bayes, and Random Forest by using ML libraries like pandas, numpy, matplotlib, and scikit-learn. After that for improving performance, we used complex deep learning models: Bi-LSTM and Bi-GRU. For using those deep learning models we used NLTK, Keras, Tensorflow, pickle, etc. At last, we tried one Transformer model: BERT in which we got a more accurate performance for Nepali text's sentiment analysis. We used 5000 triple verified annotated Nepali TikTok comments for those models. All model's performances are listed below.

4.1 Baseline Model's Performance

For baseline models we got model-wise different performances for: SVM, MNB, and RF.

4.1.1 Training & Testing on SVM

SVM model demonstrates a thorough approach to text classification. We cleaned the text data by pre-processing it and used nepali stopwords to remove them for better accuracy. Initially, it vectorizes Nepali text data using 'CountVectorizer', converting text into numerical feature vectors. These vectors are then optionally transformed into dense arrays. A custom ROC AUC scorer is defined for multi-class classification, ensuring robust evaluation. The 'GridSearchCV' method is utilized to perform an exhaustive search over specified SVM hyperparameters with cross-validation, optimizing model performance. The best model, identified through this grid search, is trained on the training data and evaluated on the test data, achieving a well-tuned and effective classification model. We split the 5,000 data in the ratio of 80% training and 20% testing and random state of 42. We have worked on SVM model for four tasks: Sentiment Analysis, Hate-Offense Detection, Political Stance Detection and Multi-label Topic Classification. The obtained results are:

For Sentiment Analysis,

Table 4.1: Classification Report-Sentiment Analysis

Class	Precision	Recall	F1-Score	Support
0	0.79	0.94	0.86	2429
1	0.90	0.83	0.86	1789
2	0.94	0.52	0.67	724
Accuracy	0.84			
Macro Avg	0.88	0.77	0.80	4942
Weighted Avg	0.85	0.84	0.83	4942

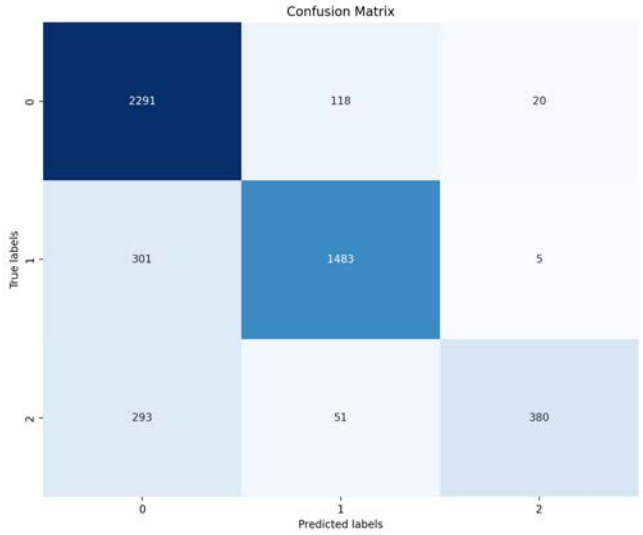


Figure 4.1: Confusion Matrix-Sentiment Analysis of SVM

For Hate-Offense Detection and Political Stance Detection,

Table 4.2: Classification Report- Hate Offence

Roc-accuracy	accuracy	F1	Precision	Recall
0.674	0.668	0.539	0.496	0.59

Table 4.3: Classification Report - Political stance

Roc-accuracy	accuracy	F1	Precision	Recall
0.772	0.777	0.544	0.522	0.569

For Multi-label Topic Classification,

Table 4.4: Classification Report - Multi-label Topic Classification

Roc-accuracy	accuracy	F1	Precision	Recall
0.772	0.777	0.544	0.522	0.569

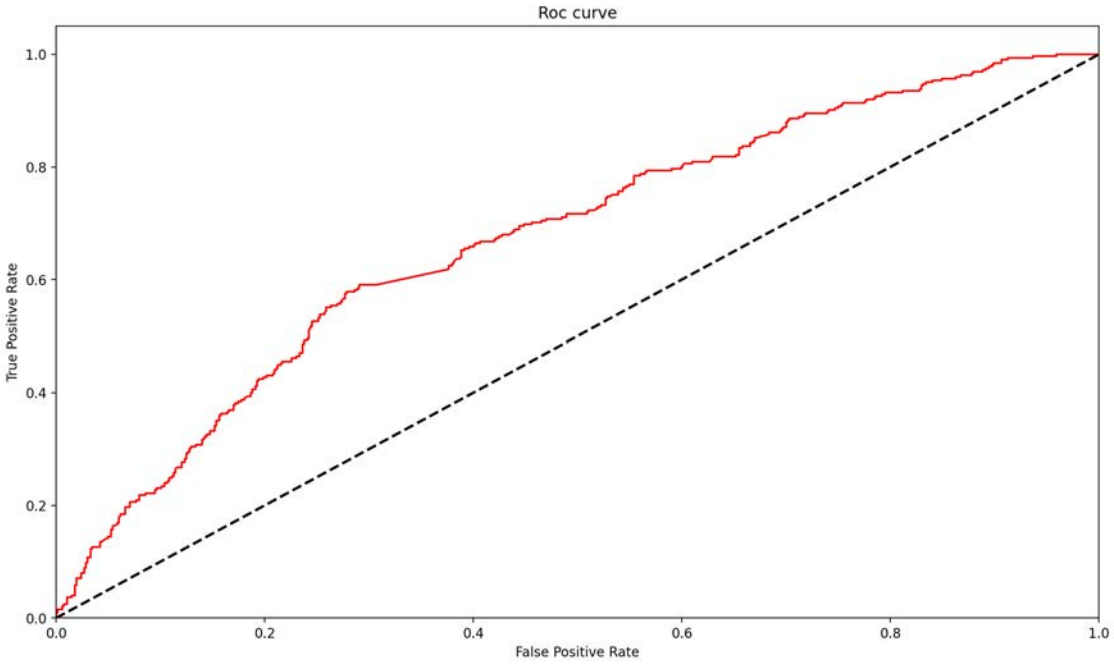


Figure 4.2: Hate Offence-ROC Curve

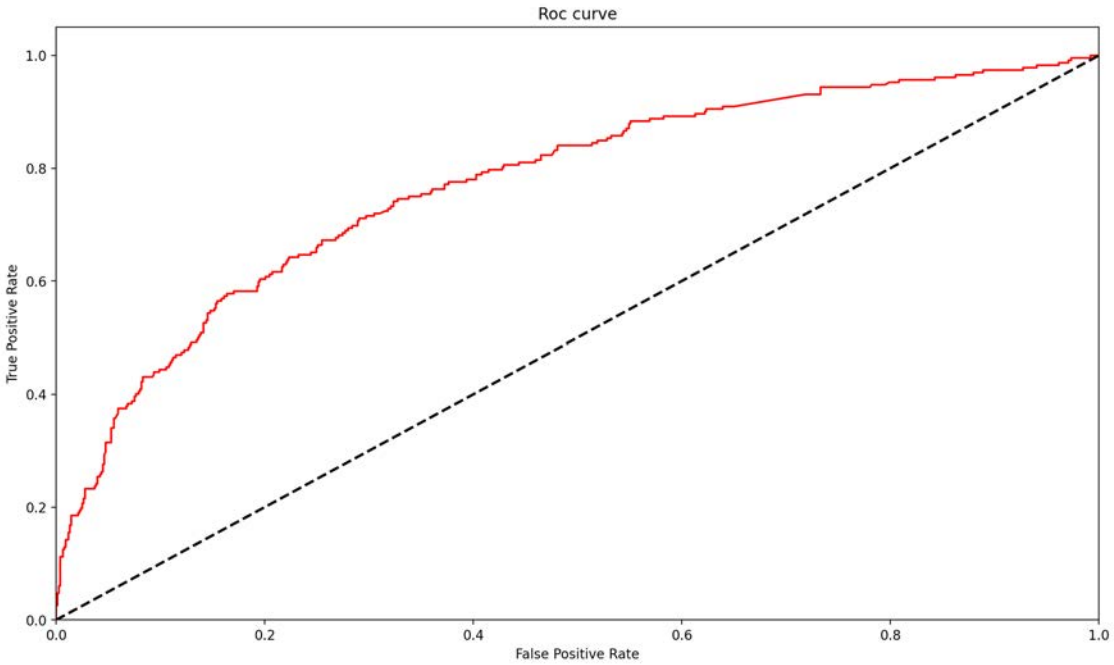


Figure 4.3: Political stance-ROC Curve

4.1.2 Training and Testing on NB

Naïve Bayes models, encompassing both Multinomial and Gaussian variants, provide a straightforward yet effective approach to text classification tasks. Beginning with data pre-processing, removing unwanted elements and employing techniques like stop word removal, for Nepali language.

During the training phase, the model learns from labeled data, where it is fed with

	Precision	Recall	F1-Score	Support
0	0.53	0.74	0.62	466
1	0.53	0.42	0.47	372
2	0.66	0.19	0.30	151
Accuracy	0.54			
Macro Avg	0.57	0.45	0.46	989
Weighted Avg	0.55	0.54	0.51	989

Table 4.5: Classification Report-Sentiment Analysis of NB

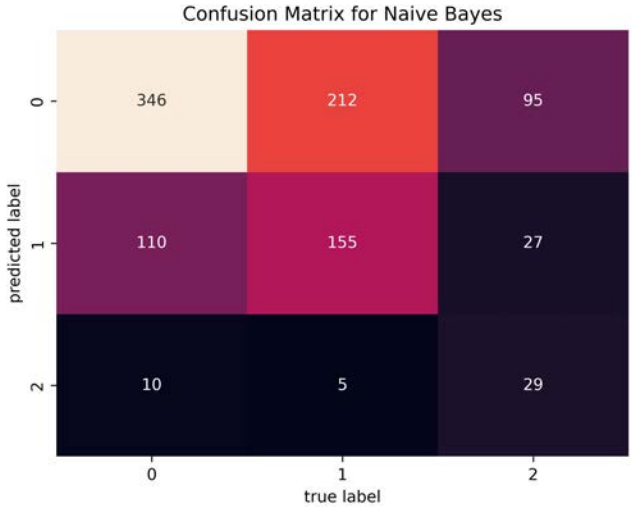


Figure 4.4: Confusion matrix-Sentiment Analysis of NB

text documents and their corresponding labels, enabling it to understand the parameters necessary for estimating class probabilities based on the features present in the text. Testing follows, where the model's performance is evaluated on a separate test set containing unseen documents, and its predictions are compared with true labels to assess accuracy. Vectorizing is crucial as it transforms the raw text into numerical format using techniques like CountVectorizer, which converts documents into feature vectors based on word frequencies. For classification, the vectorized text is processed to calculate the probability of each class for a given document. In Multinomial Naïve Bayes, this involves using the frequency of words, while Gaussian Naïve Bayes assumes a Gaussian distribution of features. Finally, probability estimation employs Bayes' theorem to determine the posterior probability of each class given the features, with the naive assumption of feature independence simplifying the calculations. The class with the highest probability is then assigned as the predicted class, completing the classification process. This integrated approach ensures that Naïve Bayes models effectively handle text classification tasks by leveraging learned probabilities and straightforward probabilistic calculations for all four tasks: Sentiment Analysis, Hate-Offense Detection, Political Stance Detection and Multi-label Topic Classification. We split the 5000 data in the ratio of 80% training and 20% testing and random state of 42. The obtained results:

	Precision	Recall	F1-Score	Support
0	0.72	0.92	0.81	1188
1	0.57	0.24	0.33	542
Accuracy	0.70			
Macro Avg	0.65	0.58	0.57	1730
Weighted Avg	0.68	0.70	0.66	1730

Table 4.6: classification Report- Hate Offense (NB)

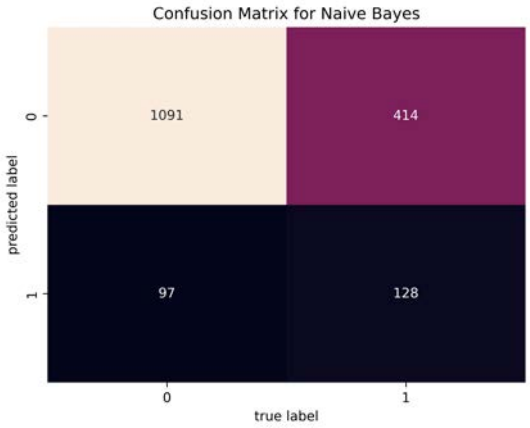


Figure 4.5: Confusion matrix-Hate Offence of NB

	Precision	Recall	F1-Score	Support
0	0.83	0.95	0.88	757
1	0.67	0.34	0.46	232
Accuracy	0.81			
Macro Avg	0.75	0.65	0.67	989
Weighted Avg	0.79	0.81	0.78	989

Table 4.7: Classification Report-Political stance(NB)

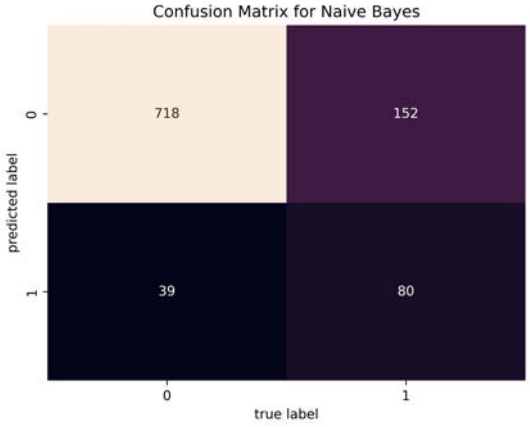


Figure 4.6: Confusion matrix-Political stance of NB

Roc-accuracy	accuracy	F1	Precision	Recall
0.772	0.777	0.544	0.522	0.569

Table 4.8: Classification Report - Multi-label Topic Classification

4.1.3 Training and Testing on RF

The model begins its operation by instantiating a Decision Tree object with a maximum depth of 5. This initialization lays the foundation for constructing a decision tree, a fundamental component of the subsequent Random Forest ensemble. The training process commences with the fit method, which recursively builds the decision tree by evaluating optimal feature splits based on Gini impurity. This recursive approach ensures that at each node, the algorithm identifies the feature that best separates the data into homogeneous classes. Upon reaching either a maximum depth or a node where all instances belong to the same class, the algorithm terminates, resulting in the creation of leaf nodes containing class labels.

Metric	**Value**
Accuracy	0.4712
F1-score	0.46
Precision	0.2220
Recall	0.4712

Table 4.9: Classification report of Sentiment Analysis(RF)

As the Decision Tree model completes its training phase, attention turns to the Random Forest ensemble. An instance of the Random Forest class is created, specifying 25 decision trees, each with a maximum depth of 5. The fit method of the Random Forest class orchestrates the training process, iterative creating bootstrap samples from the training data and fitting decision trees to each sample. By employing bootstrap sampling, the model fosters diversity among the constituent decision trees, mitigating the risk of over-fitting and enhancing overall predictive performance.

Metric	**Value**
Accuracy	0.6724
F1-score	0.6017
Precision	0.5698
Recall	0.6424

Table 4.10: Classification report of Hate Offence(RF)

Utilizing the predict method of the Decision Tree class, predictions are generated for each instance in the test set, leveraging the constructed decision tree. Similarly, predictions are made with the Random Forest model, aggregating the outputs of all decision trees through majority voting. These predictions are then compared against the true labels to evaluate the models' accuracy in correctly classifying the sentiment of the test instances.

Metric	Value
Accuracy	0.7654
F1-score	0.7417
Precision	0.7259
Recall	0.7654

Table 4.11: Classification report of Political stance(RF)

The accuracy of the models is subsequently calculated by dividing the number of correctly predicted instances by the total number of instances in the test set. This accuracy metric serves as a quantitative measure of the models' performance on unseen data, providing valuable insights into their effectiveness in sentiment classification. By meticulously traversing the training and testing phases, the models showcase their ability to discern sentiment patterns and generalize to new data, underscoring the utility of decision trees and random forests in sentiment analysis tasks. We split the 5,000 data in the ratio of 80% training and 20% testing and random state of 42. We have worked on SVM model for four tasks. The obtained results are:

Metric	Value
Accuracy	0.6011
F1-score	0.5992
Precision	0.6023
Recall	0.5884

Table 4.12: Classification report of Multi-label Topic Classification

4.2 Deep Learning Model's Performance

For Deep Learning models we got model-wise different performances for: Bi-LSTM and Bi-GRU.

4.2.1 Training and Testing on Bi-LSTM

Bidirectional actually takes the input number vector of data in a sequential manner and processes through its memory, hidden layer, and generate output depending upon the activation functions provided. First, we tokenize the Nepali sentences using the NLTK library, and after that embedded every token using GloVe embedding, and those word vectors are passed into Bi-LSTM. As we have four tasks to do some parameters might be changed for some tasks but altogether they are almost the same. We had 5,000 proper annotated data for model training and tests so split in the ratio of 80% and 20% for train & test. We used two hidden dense LSTM layers for generating output and two activation functions: Sigmoid & Softmax. Table 4.1 gives the details explanations about what hyper-parameter we used for all four tasks [28].

Table 4.13: Hyperparameter Settings for Bi-LSTM

Hyperparameters	Initial Value	Hyperparameter Space	Optimal Value
Embedding Dimension	64	64, 100, 128, 200, 256, 512, 1024	128
Batch Size	32	16, 32, 64, 128, 256, 512	64
Dropout	0.1	0.1, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5	0.3 - 0.4
Optimizer	Adam	SGD, RM Sprop, Adam	Adam
Learning Rate	0.1	0.1, 0.01, 0.0001, 0.00001, 0.000001	0.00001
Number of Epochs	5	5, 10, 15, 20, 25, 30, 40	20

Sentiment Analysis - Bi-LSTM

For solving Sentiment Analysis of Nepali sentence we used Bi-LSTM with different variations of Hyper-parameter values and came to the optimal. This task has three classes: Positive, Negative & Neutral. As we used 80% data for training and 20% data for training, the LSTM gave a good performance on that. As sentiment analysis has three classes we used softmax as the activation function, softmax calculates the probability of each class and chooses the maximum one. For other hyper-parameters, we used 64 batch size initialize very high epoch about 1000 and used early stopping with patience 20 and dropout is about 0.4. The optimizer is Adam and we used the Categorical Cross entropy Loss function for calculating the model train loss & learning rate is 0.00001. The model units used are 128 and 64 respectively. For Sentiment analysis, we got 56.11% accuracy and 50.99% F1-score. The confusion matrix is shown in Table 4.2.

Table 4.14: Confusion Matrix - Sentment Analysis

	Neutral	Negative	Positive
Neutral	330	102	32
Negative	180	190	9
Positive	90	19	42

Hate-Offense Detection - Bi-LSTM

To solve Hate Offense Detection we used Bi-LSTM with different variations of Hyper-parameter values and came to the optimal. This task has two classes: Offensive & Not Offensive. As we used 80% data for training and 20% data for training, the LSTM gave a good performance on that. As Hate-offense detection has two classes we used sigmoid as the activation function, sigmoid return probability from 0 and 1, and used for binary separation. For other hyper-parameters, we used 64 batch size initialize very high epoch about 1000 and used early stopping with patience 20 and dropout is about 0.4. The optimizer is Adam and we used the Binary cross entropy Loss function for calculating the model train loss & learning rate is 0.00001. The model units used are 128 and 64 respectively. For Hate-Offense detection we got 73.50% accuracy and 73.71% F1-score. The confusion matrix is shown in Table 4.3.

Table 4.15: Confusion Matrix - Hate-Offense Detection

	Not offensive	Offensive
Not Offensive	560	99
Offensive	160	160

Political Stance Detection - Bi-LSTM

Political stance detection finds whether the text is politically linked or not for that we used Bi-LSTM with different variations of Hyper-parameter values and came to the optimal. This task has two classes: Political & Not Political. As we used 80% data for training and 20% data for training, the LSTM gave a good performance on that. As political stance detection has two classes we used sigmoid as the activation function, sigmoid return probability from 0 and 1, and used for binary separation. For other hyper-parameters, we used 64 batch size initialize very high epoch about 1000 and used early stopping with patience 20 and dropout is about 0.4. The optimizer is Adam and we used the Binary cross entropy Loss function for calculating the model train loss & learning rate is 0.00001. The model units used are 128 and 64 respectively. For political stance detection, we got 84.93% accuracy and 84.90% F1-score. The confusion matrix is shown in Table 4.4.

Table 4.16: Confusion Matrix - Political stance Detection

	Non Political	Political
Non Political	680	72
Political	77	160

Multi-label Topic Classification - Bi-LSTM

Multi-label is defined as the daily regular types of sentence that tend to belong as one of the types, for this, we used bi-LSTM with optimal values of hyper-parameters. This task has nine classes: Political Governance and Social Concern (PGS), Food and Travel (FT), Education Youth and Employment (EYE), Health Beauty and Fitness (HBF), Family and Relationship (FR), Entertainment Music and Pop-Culture (EMP), Business and Finance (BF), Spirituality and Well-being (SW) & Daily Dairies and Other hobbies (DO). As we used 80% data for training and 20% data for training, the LSTM gave a good performance on that. As this has nine classes we used softmax as the activation function, softmax calculates the probability of each class and chooses the maximum one. For other hyper-parameters, we used 64 batch size initialize very high epoch about 1000 and used early stopping with patience 20 and dropout is about 0.4. The optimizer is Adam and we used the categorical cross entropy Loss function for calculating the model train loss & learning rate is 0.00001. The model units used are 128 and 64 respectively. For multi-label topic classification, we got 65% accuracy and 64.66% F1-score.

Bi-LSTM on Four Tasks: Now we can observe all the performance metrics of four tasks in table 4.5. Although all are shown in one single table they serve their

property and characteristics. We tried our best to keep the results high as much as possible although there can be a certain increase in given results by changing some hyper-parameters.

Table 4.17: All Tasks Results using Bi-LSTM

Task	Accuracy	Precision	Recall	F1-Score
Sentiment Analysis	56.11%	56.93%	45.91%	50.92%
Hate-Offense Detection	73.5%	73.55%	73.85%	73.71%
Political stance Detection	84.93%	84.99%	84.79%	84.9%
Multi-label Topic Classification	65%	67%	62.37%	64.66%

4.2.2 Training and Testing on Bi-GRU

GRU's basic concept is to update the network's hidden state only on a chosen subset of time steps by means of gating methods. Information entering and leaving the network is managed by the gating mechanisms. While Bi-GRU takes an input vector from both directions and try to capture the context of a word in a sequence. In this research, we uses Nepali stopwords which enables the model to focus on terms that are more important in differentiating across various classes or categories. Moreover, it also reduces the efficiency by reducing the number of tokens by filtering out stopwords that speed up model training times and increase computational efficiency. And then we tokenize the comments which we scraped from the influential Tik-Tok accounts, using nltk.tokenize library. Then those tokenized comments are embedded in Nepali GloVe using GloVe embedding and then those input vectors are passed though Bi-GRU Model. We had 5000 proper annotated data for model training and tests so split in the ratio of 80% and 20% for train & test. In this model, we have used two hidden dense layer each unit has 3 neurons and two activation functions in the output layer named as Sigmoid and Softmax. Table 4.2.1 gives the details explanations about the hyper-parameter that we have used for all four tasks.

Table 4.18: Hyperparameter Settings for Bi-GRU

Hyperparameters	Initial Value	Hyperparameter Space	Optimal Value
Embedding Dimension	64	64, 100, 128, 200, 256, 512, 1024	128
Batch Size	32	16, 32, 64, 128, 256, 512	64
Dropout	0.1	0.1, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5	0.3 - 0.4
Optimizer	Adam	SGD, RM Sprop, Adam	Adam
Learning Rate	0.1	0.1, 0.01, 0.0001, 0.00001, 0.000001	0.00001
Number of Epochs	5	5, 10, 15, 20, 25, 30, 40	20

Sentiment Analysis - Bi-GRU

We discovered the most optimal approach when utilizing Hyper parameter adjustment of Bi-GRU to solve the Sentiment analysis of Nepali Tik-Tok comments. In this analysis, we have to deal with three classes they are Positive(2), Negative(1) and Neutral(1).The Bi-GRU performed well since we used 80% of the data for training and 20% of the data for that purpose. We chooses Softmax as the activation

function because sentiment analysis has multi-label classes; softmax determines the maximum likelihood of each class and also squeezes the value of x of Sigmoid. In the hyper parameter tuning, we took 1000 epochs and batch size 64, and to stop the model from over fitting we have introduced early stopping while training the model. In this model we import a linear stack of layer which add one layer at a time in sequence. We have set trainable= False, which indicates that the weight of embedding layer should not be updated during the training. Two GRU layers, each with 256 and 128 units, were added and apart from that a robust optimizer Adam is used. We have added a dense layer of unit 3, which typically matches the number of classes in classification task. Dropout of 0.3 is introduced which randomly takes 30% of the input units to 0 and at each update during training. Moreover, the learning rate of 0.00001 and categorical cross entropy loss entropy is used to compile the model. For sentiment analysis, we have got 54.9% accuracy and 50.37% F1-score. Here is the confusion matrix shown in table 4.7

Table 4.19: Confusion Matrix - Sentiment Analysis

	Neutral	Negative	Positive
Neutral	350	100	10
Negative	200	160	6
Positive	100	20	30

Hate-Offense Detection - Bi-GRU

For Hate Offense Detection, we utilizes different variation of hyper parameter of Bi-GRU. In this study, the sets of are categorized in two classes- offensive and Not offensive. Considering that we trained the GRU with 80% and 20% of the data, respectively, it performed well. Using sigmoid as the activation function, which returns a probability between 0 and 1, we were able to separate them into two groups of hate-offense detection. In the hyper parameter tuning, we took 1000 epochs and batch size 64, and to stop the model from over fitting we have introduced early stopping while training the model. In this model we import a linear stack of layer which add one layer at a time in sequence.Two GRU layers, each with 128 and 64 units, were added and apart from that a robust optimizer Adam is used. We have added a dense layer of unit 2, which typically matches the number of classes in classification task. Dropout of 0.4 is introduced along with the learning rate of 0.00001 and binary cross entropy loss entropy is used to compile the model. In Hate Offense detection, we have got 83.01% accuracy and 83.05% F1-score. Here is the confusion matrix of Hate-offense detection in table 4.8

Table 4.20: Confusion Matrix - Hate-Offense Detection

	Not offensive	Offensive
Not Offensive	680	72
Offensive	96	140

Political Stance Detection - Bi-GRU

By utilizing Bi-GRU with various hyper-parameter value adjustments, we were able to determine whether or not the text is politically connected. This process of political stance detection allows us to arrive at the best result. Two classes are available for this task: Political and Not Political. Considering that we trained the GRU model with 80% and 20% of the data, respectively, it performed well. We were able to divide the dataset into two groups for hate-offense detection using the sigmoid activation function, which provides a probability between 0 and 1. We used a batch size of 64 and 1000 epochs for the hyperparameter tuning. We also added early stopping during the model's training to prevent the model from overfitting. Two GRU layers, each with 128 and 64 units, were added and apart from that a robust optimizer Adam is used. A dense layer of unit 2 has been added; this usually corresponds to the number of classes in the classification task. The model is constructed using binary cross entropy loss entropy, with a dropout of 0.4 and a learning rate of 0.00001. For Political stance detection, we have got 84.12% accuracy 84.04% F1-score. Here is the confusion matrix of political stance Detection shown in Table 4.9

Table 4.21: Confusion Matrix - Political stance Detection

	Non Political	Political
Non Political	690	65
Political	92	140

Multi-label Topic Classification - Bi-GRU

We utilized Bi-GRU with the ideal values of the hyper-parameters to identify multilabel, which is defined as the typical types of sentences that occur on a daily basis and tend to fall into one of the categories. Nine classes make up this task: Social Concern and Political Governance (PGS), Food and Travel (FT), Education, Youth, and Employment (EYE), Health, Beauty, and Fitness (HBF), Family and Relationships (FR), Entertainment, Music, and Pop Culture (EMP), Business and Finance (BF), Spirituality and Well-Being (SW), & Daily Dairies and Other Hobbies [DO]. Considering that we trained the GRU with 80% and 20% of the data, respectively, it performed well. Softmax was chosen as the activation function because it computes the probability of each of the nine classes and selects the maximum one. We used 64 batch sizes, a very high epoch of about 1000, early stopping, and a dropout of 0.4 for the other hyper-parameters. In this study, we got 68.02% accuracy and 73.77% F1-score.

Bi-GRU on Four Tasks:
Table 4.10 displays all of the performance matrices for the four tasks that were examined in the study. Despite being displayed in a single table, each serves its own purpose and has its own attributes. Although there may be some rise in the results by adjusting some hyper-parameters, we made every effort to maintain the high results as much as possible.

Table 4.22: All Tasks Results using Bi-GRU

Task	Accuracy	Precision	Recall	F1-Score
Sentiment Analysis	54.90%	55.01%	47.01%	50.37%
Hate-Offense Detection	83.01%	83.40%	83.30%	83.05%
Political stance Detection	84.12%	84.19%	84.23%	84.04%
Multi-label Topic Classification	68.02%	70.01%	58.45%	66%

4.3 Transformer Model

Here we train and test our dataset on BERT.

4.3.1 BERT: Bidirectional Encoder Representations from Transformers

A BERT-based model was fine-tuned for all the tasks using the 'bert-base-multilingual-cased' variant. The dataset was split into training and testing sets, preprocessed, and tokenized using BERT's tokenizer. The model was configured for sequence classification with labels and trained on a T4 GPU for 15 epochs. The training process utilized the AdamW optimizer with a learning rate of 2e-5 and epsilon of 1e-8, and a batch size of 16. During training, gradient clipping with a maximum norm of 1.0 was applied to stabilize the training process. After training, the model's performance was evaluated on the test set, achieving an accuracy along with precision, recall, and F1 scores as shown in the table below.

Table 4.23: All Tasks Results using BERT

Task	Accuracy	Precision	Recall	F1-Score
Sentiment Analysis	75.63%	75.01%	76%	75%
Hate-Offense Detection	62.04%	63%	62%	62.56%
Political stance Detection	77.05%	76%	77%	76.06%
Multi-label Topic Classification	71.06%	70.82%	71.04%	70%

4.4 All Models Comparison and Analysis

Here we compare the results between SVM, NB, RF, Bi-LSTM, Bi-GRU, and BERT on all of our tasks.

For Sentiment Analysis task Support Vector Machine (SVM) performed with the best with accuracy of 84% and F1-score of 84%. In our case SVM performed well because it is effective with smaller dataset and less prone to overfitting. Since the Nepali text is limited, SVM can leverage its ability to find optimal hyperplanes that maximizes the margin between classes, often leading to better performance.

For Hate and Offense task Bi-GRU performed the best with accuracy of 83% and F1-score of 83.2%. Bi-GRU's strong performance on this task is likely due to its

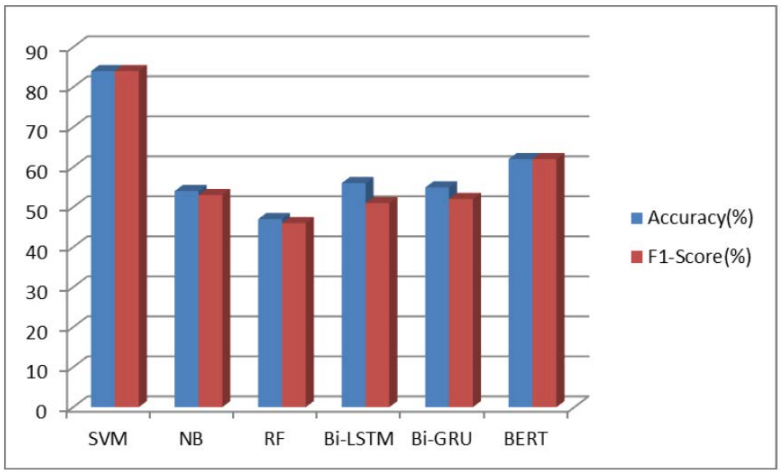


Figure 4.7: Bar Graph of Sentiment Analysis Performance

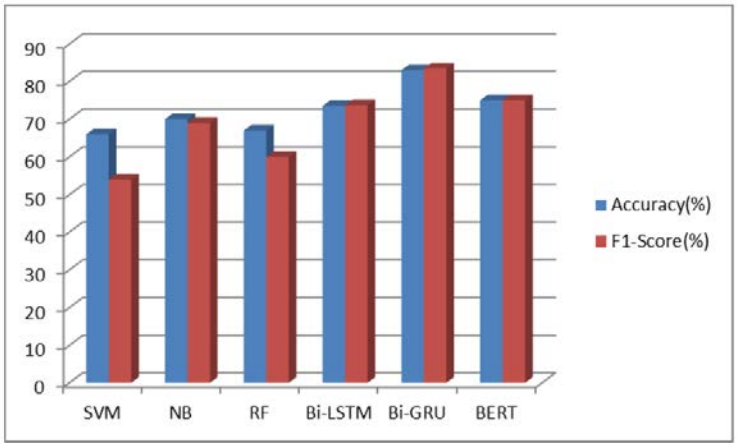


Figure 4.8: Bar Graph of Hate Offense Detection

ability to effectively capture bidirectional dependencies and manage long-term context, combined with its relatively efficient training process and capacity to work well with limited data.

For Political stance Detection task Bi-LSTM performed the best with accuracy of 84% and F1-score of 84%. Bi-LSTM's process the input text in both forward and backward directions. This means that they can capture information from both left and right contexts for each word. This bidirectional nature is particularly useful in understanding the full context of a sentence, which is essential in detecting political stances where the meaning can depend on words that appear later in the sentence.

For Multi-Label topic classification task BERT performed the best with accuracy of 71% and F1-score of 70%. BERT's superior performance on multi-label topic classification for Nepali text can be attributed to its advanced contextual understanding through bidirectional self-attention, extensive pre-training on large datasets, effective handling of out-of-vocabulary words, fine-tuning capabilities, and natural support for multi-label classification.

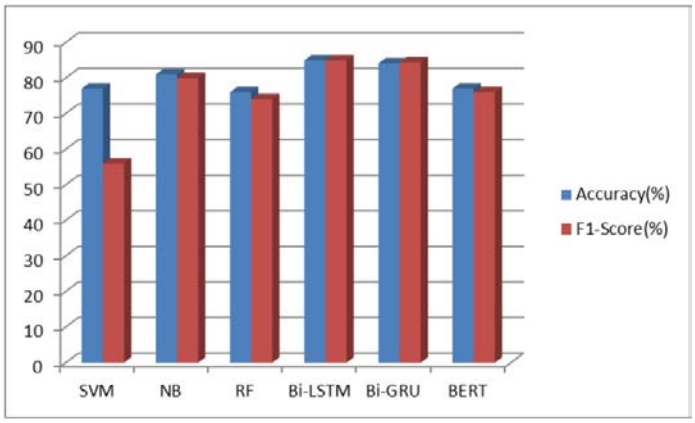


Figure 4.9: Bar Graph of Political stance Detection Performance

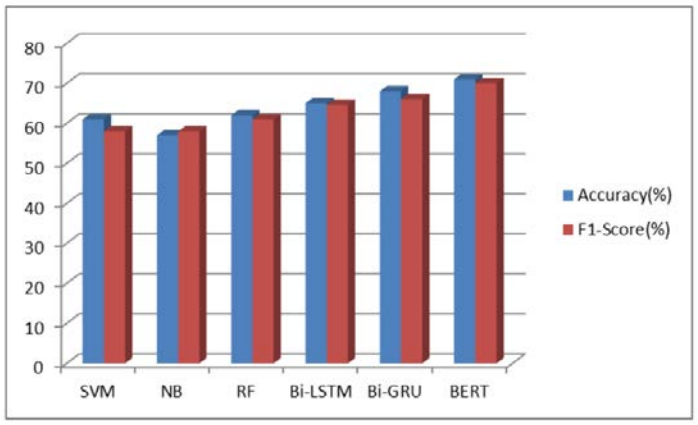


Figure 4.10: Bar Graph of Multi-label Topic Classification Performance

Chapter 5

Conclusion

5.1 Conclusion

In this study, we proposed to analyze TikTok's content in Nepal using Natural Language Processing and Machine Learning tools to conclude how TikTok is influencing conversations in society. We identified techniques and methods to process and analyze content available on dynamic platforms like TikTok. We conducted Sentiment Analysis, Hate-Offense Detection, Political Stance Detection, and Multi-label Topic Classification. We used different types of models for those tasks, models include Machine Learning models (SVM, MNB, and RF), Deep Learning Models (Bi-LSTM and Bi-GRU) and Transformer model (BERT). For Nepali NLP enthusiasts/researchers this research will be a great source of help.

To sum up, in today's digital world, the scope and use of Artificial Intelligence and Machine Learning is increasing which directly or indirectly affects us in both positive and negative manner. We can use Sentiment Analysis and other subsequent tasks to understand society, gauze trends, record public reactions, plan product marketing, etc. Furthermore, in platforms like TikTok we can address the cases of cyber bullying, digital harassment, scams, crimal activity, etc. Regulators and platform administrators can come up with auto moderation of content or develop new usage policies based on the research like ours.

5.2 Future Work

Lastly, we successfully implemented four tasks using different types of models and we hope that our research can be integrated into other languages like Bengali, Hindi, Urdu, etc. We have great hopes that this research will assist and make effective progress in the NLP field for Nepali researchers. Furthermore, we plan to build a multidimensional model that can handle audio, video, and text to better understand the contents available in TikTok not just in the context of Nepal but globally.

Bibliography

[1] T. Gonçalves and P. Quaresma, "The impact of nlp techniques in the multilabel text classification problem," in *Intelligent Information Processing and Web Mining: Proceedings of the International IIS: IIPWM '04 Conference held in Zakopane, Poland, May 17–20, 2004*, Springer, 2004, pp. 424–428.

[2] A. Fujino, H. Isozaki, and J. Suzuki, "Multi-label text categorization with model combination based on f1-score maximization," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[3] M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik, "Political ideology detection using recursive neural networks," in *Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, 2014, pp. 1113–1122.

[4] A. K. Pant and A. Yadav, "Sentiment analysis on nepali movie reviews using machine learning," 2014.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[6] C. P. Gupta and B. K. Bal, "Detecting sentiment in nepali texts: A bootstrap approach for sentiment analysis of texts in the nepali language," in *2015 international conference on cognitive computing and information processing (ccip)*, IEEE, 2015, pp. 1–4.

[7] V. Sahayak, V. Shete, and A. Pathan, "Sentiment analysis on twitter data," *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, vol. 2, no. 1, pp. 178–183, 2015.

[8] L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari, "Sentiment analysis of review datasets using naïve bayes and k-nn classifier," *arXiv preprint arXiv:1610.09982*, 2016.

[9] E. García-Gonzalo, Z. Fernández-Muñiz, P. J. Garcia Nieto, A. Sánchez, and M. Menéndez, "Hard-rock stability analysis for span design in entry-type excavations with learning classifiers," *Materials*, vol. 9, p. 531, Jun. 2016. DOI: 10.3390/ma9070531.

[10] L. B. R. Thapa and B. K. Bal, "Classifying sentiments in nepali subjective texts," in *2016 7th International conference on information, intelligence, systems & applications (IISA)*, IEEE, 2016, pp. 1–6.

[11] M. S. Islam, F. Elahi, and S. I. Ahmed, "A support vector machinemixed with tf-idf algorithm to categorize bengali document," Apr. 2017. DOI: 10.1109/ECACE.2017.7912904.

[12] Y. Al Amrani, M. Lazaar, and K. E. El Kadiri, "Random forest and support vector machine based hybrid approach to sentiment analysis," *Procedia Computer Science*, vol. 127, pp. 511–520, 2018.

[13] A. Bayhaqy, S. Sfenrianto, K. Nainggolan, and E. R. Kaburuan, "Sentiment analysis about e-commerce from tweets using decision tree, k-nearest neighbor, and naıve bayes," in *2018 international conference on orange technologies (ICOT)*, IEEE, 2018, pp. 1–6.

[14] M. A. Fauzi, "Random forest approach fo sentiment analysis in indonesian," *Indones. J. Electr. Eng. Comput. Sci*, vol. 12, pp. 46–50, 2018.

[15] U. W. Wijayanto and R. Sarno, "An experimental study of supervised sentiment analysis using gaussian naıve bayes," in *2018 International Seminar on Application for Technology of Information and Communication*, IEEE, 2018, pp. 476–481.

[16] M. Abbas, K. A. Memon, A. A. Jamali, S. Memon, and A. Ahmed, "Multinomial naïve bayes classification model for sentiment analysis," *IJCSNS Int. J. Comput. Sci. Netw. Secur*, vol. 19, no. 3, p. 62, 2019.

[17] D. Berrar, *Bayes' theorem and naïve bayes classifier.* 2019.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[19] A. Purwarianti and I. A. P. A. Crisdayanti, "Improving bi-lstm performance for indonesian sentiment analysis using paragraph vector," in *2019 International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, IEEE, 2019, pp. 1–5.

[20] S. Wen, H. Wei, Y. Yang, *et al.*, "Memristive lstm network for sentiment analysis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 3, pp. 1794–1804, 2019.

[21] F. M. Espinoza-Cuadros, "Study of speech and craniofacial features in obstructive sleep apnea patients," Ph.D. dissertation, Jan. 2020. DOI: 10.20868/UPM.thesis.51768.

[22] N. B. Niraula, S. Dulal, and D. Koirala, *Linguistic taboos and euphemisms in nepali*, 2020. arXiv: 2007.13798 [cs.CL].

[23] R. Piryani, B. Piryani, V. K. Singh, and D. Pinto, "Sentiment analysis in nepali: Exploring machine learning and lexicon-based approaches," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 2, pp. 2201–2212, 2020.

[24] O. M. Singh, S. Timilsina, B. K. Bal, and A. Joshi, "Aspect based abusive sentiment detection in nepali social media texts," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2020, pp. 301–308.

[25] S. Tamrakar, B. K. Bal, and R. B. Thapa, "Aspect based sentiment analysis of nepali text using support vector machine and naïve bayes," *Technical Journal*, vol. 2, no. 1, pp. 22–29, 2020.

[26] J. Álvarez-Alvarado, G. Moreno, S. Obregón-Biosca, G. Ronquillo, E. Ventura-Ramos, and J. Trejo-Perea, "Hybrid techniques to predict solar radiation using support vector machine and search optimization algorithms: A review," *Applied Sciences*, vol. 11, p. 1044, Jan. 2021. DOI: 10.3390/app11031044.

[27] L. Dhanya and K. Balakrishnan, "Hate speech detection in asian languages: A survey," in *2021 international conference on communication, control and information sciences (ICCISc)*, IEEE, vol. 1, 2021, pp. 1–5.

[28] E. Hossain, O. Sharif, M. Hoque, and I. Sarker, "Sentilstm: A deep learning approach for sentiment analysis of restaurant reviews," in Apr. 2021, pp. 193–203, ISBN: 978-3-030-73049-9. DOI: 10.1007/978-3-030-73050-5_19.

[29] P. Koirala and N. B. Niraula, "Npvec1: Word embeddings for nepali-construction and evaluation," in *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 2021, pp. 174–184.

[30] W. Liu, J. Pang, N. Li, X. Zhou, and F. Yue, "Research on multi-label text classification method based on talbert-cnn," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, p. 201, 2021.

[31] N. B. Niraula, S. Dulal, and D. Koirala, "Offensive language detection in nepali social media," in *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, 2021, pp. 67–75.

[32] S.-H. Noh, "Analysis of gradient vanishing of rnns and performance comparison," *Information*, vol. 12, no. 11, p. 442, 2021.

[33] K. M. Alzhrani, "Political ideology detection of news articles using deep neural networks.," *Intelligent Automation & Soft Computing*, vol. 33, no. 1, 2022.

[34] S. Dutta, S. Caur, S. Chakrabarti, and T. Chakraborty, "Semi-supervised stance detection of tweets via distant network supervision," in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 241–251.

[35] khoros, *The 2022 social media demographics guide*, Intelligent Automation & Soft Computing, 2022.

[36] V. R. Kota and S. D. Munisamy, "High accuracy offering attention mechanisms based deep learning approach using cnn/bi-lstm for sentiment analysis," *International Journal of Intelligent Computing and Cybernetics*, vol. 15, no. 1, pp. 61–74, 2022.

[37] S. Biswas, S. Ghosh, S. Roy, R. Bose, and S. Soni, "A study of stock market prediction through sentiment analysis," *Mapana Journal of Sciences*, vol. 22, Feb. 2023. DOI: 10.12723/mjs.64.6.

[38] U. Mahadevaswamy and P. Swathi, "Sentiment analysis using bidirectional lstm network," *Procedia Computer Science*, vol. 218, pp. 45–56, 2023.

[39] T. Online, *Govt bans tiktok in nepal*, The Himalayan Times, 2023.

[40] Y. Tian, "Multi-label text classification combining bert and bi-gru based on the attention mechanism," 2023.

[41] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[42] *Bidirectional LSTM in NLP - GeeksforGeeks — geeksforgeeks.org*, https://www.geeksforgeeks.org/bidirectional-lstm-in-nlp/, [Accessed 20-05-2024].

[43] *Gated Recurrent Unit Networks - GeeksforGeeks — geeksforgeeks.org*, https://www.geeksforgeeks.org/gated-recurrent-unit-networks/, [Accessed 19-05-2024].

[44] *Long short-term memory - Wikipedia — en.wikipedia.org*, https://en.wikipedia.org/wiki/Long_short-term_memory, [Accessed 20-05-2024].

[45] *Machine Learning Random Forest Algorithm - Javatpoint — javatpoint.com*, https://www.javatpoint.com/machine-learning-random-forest-algorithm, [Accessed 20-05-2024].

[46] M. Phi, *Illustrated Guide to LSTM's and GRU's: A step by step explanation — towardsdatascience.com*, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, [Accessed 19-05-2024].

[47] S. E. R, *Understand Random Forest Algorithms With Examples (Updated 2024) — analyticsvidhya.com*, https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/, [Accessed 20-05-2024].

[48] I. Vasilev, *Advanced Deep Learning with Python — oreilly.com*, https://www.oreilly.com/library/view/advanced-deep-learning/9781789956177/8ad9dc41-3237-483e-8f6b-7e5f653dc693.xhtml, [Accessed 20-05-2024].