KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT707]

A MAJOR PROJECT MID-TERM REPORT ON

NEPALI HATE SENTIMENT DETECTION

Submitted by:

Aadarsha Regmi[kan077bct001]

Angel Tamang [kan077bct012]

Anil Bhatta [kan077bct013]

Gaurav Maharjan[kan077bct035]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

December, 2024

NEPALI HATE SENTIMENT DETECTION

Submitted by:

Aadarsha Regmi[kan077bct001]

Angel Tamang [kan077bct012]

Anil Bhatta [kan077bct013]

Gaurav Maharjan[kan077bct035]

Supervised by:

Dr. Ashim Khadka

Assistant Professor

Nepal College of Information Technology

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

December, 2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
ML	Machine Learning
SVM	Support Vector Machines
TF-IDF	Term Frequency – Inverse Document Frequency
CBOW	Continuous Bag Of Words
NLTK	Natural Language ToolKit
NepBERTa	Nepali Bidirectional Encoder Representations from Transformers Archit
CNN	Condensed Nearest Neighbour
NCR	Neighbourhood Cleaning Rule
SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
LASER	Language Agnostic SEntence Representations
GLOVE	GLObal Vectors for Word Representations

CHAPTER 1
INTRODUCTION

1.1 Background

Nepali is spoken by approximately 40 million people worldwide, and is a key medium of communication in Nepal and the Himalayan regions. Rooted in the Devanagari script, Nepali comprises 36 consonants, 13 vowels, and 10 numerals, reflecting its rich linguistic heritage and cultural significance [1]. As a unifying language among diverse communities, Nepali plays a pivotal role in preserving the socio-cultural fabric of South Asia. The rise of digital communication has transformed the way Nepali speakers interact, with approximately 13.50 million active social media users in Nepal as of January 2024, and Facebook alone accounting for over 91% of the social media market share. While these platforms have created opportunities for connection and discourse, they have also become breeding grounds for online hate speech and digital violence. This issue is particularly concerning in low-resource languages like Nepali, which lack dedicated tools for detecting and addressing offensive content, leaving their speakers vulnerable to online harassment. Gendered violence on social media is a prominent example of this growing issue which highlights the experiences of women in Nepali politics, including Ram Kumari Jhakri, who faced a barrage of misogynistic comments after expressing support for a grant [2]. Among these comments, one Facebook user wrote, “Hey whore, come over here, I’ll pay Rs 10,000 per minute.” The analysis further revealed that she received 199 insulting comments, 20 reputational attacks, and 22 physical threats. Such incidents discourage public participation, particularly for women, and undermine democratic processes in Nepal.

Hate speech, often defined as language intended to offend or discriminate, poses a significant threat to healthy communication and, in extreme cases, can lead to violence. While NLP techniques have shown promise in combating hate speech, their application in low-resource languages like Nepali remains underexplored. The lack of research and tools tailored for such languages also challenges global initiatives like the United Nations’ Leave No One Behind principle, which emphasizes inclusive development that uplifts all communities. This project aims to bridge this gap by developing a Nepali-specific hate speech detection system, fostering safer online spaces and promoting inclusivity in the digital realm.

1.2 Problem Statement

The rapid growth of digital platforms has amplified the presence of hate speech in Nepali online content, ranging from casual profanity to deeply offensive and harmful expressions. The linguistic nuances of Nepali and its lack of focus by major platforms have made detecting such content particularly challenging. This oversight perpetuates the spread of hate speech and marginalizes the Nepali-speaking community. While tools for detecting hate speech in major languages exist, they often fail to address the linguistic intricacies and resource constraints of underrepresented languages like Nepali.

1.3 Objectives

The specific objectives of the project are as follows:

i To develop a fine-tuned hate speech detection model tailored to the Nepali language by leveraging transfer learning techniques.

ii To create an easily installable and usable package, allowing users to test and implement hate speech detection in Nepali texts seamlessly.

iii To contribute to the broader field of hate speech detection by extending advancements to underrepresented languages with limited computational resources.

1.4 Application Scope

- Social Media Platforms: Enable automatic detection of Nepali hate speech to foster respectful online environment.
- Content Moderation: Assist moderators in identifying and removing offensive language from Nepali online forums and comment sections.
- Community Support: Empower Nepali-speaking communities by reducing exposure to online abuse and creating safer digital environments.
- Future Research: Serve as a benchmark for further studies in hate speech detec-

tion for low-resource languages, promoting inclusivity in natural language processing (NLP).

1.5 System Requirements

This project needs certain hardware and software requirements in order to be developed and run. These requirements are discussed below:

1.5.1 Development Requirements

Table 1.1: Development Requirements

Hardware Requirements	**Software Requirements**
• RAM: 8GB (minimum) • Processor: CPU with four or more threads • Storage: 512GB HDD or SSD (recommended) • GPU: 15GB (minimum)	• OS: Windows 7 and above • Python • Jupyter Notebook • PyCharm • Visual Studio Code • Google Colab • Frameworks: – NLTK (Natural Language Toolkit) – spaCy – textract – TensorFlow – PyTorch – Transformers

1.5.2 Deployment Requirements

Table 1.2: Deployment Requirements

Hardware Requirements	**Software Requirements**
• RAM: 8GB (minimum) • Processor: CPU with four or more threads • Storage: 512GB HDD or SSD (recommended) • GPU: 15GB (minimum)	• OS: Windows 7 and above • Visual Studio Code • Python, pip, wheel, twine, setuptools • pyPI account

1.6 Feasibility Study

1.6.1 Technical Feasibility

The project is technically feasible due to the use of well-established tools and technologies. The implementation relies on Python, transfer learning techniques, and libraries like TensorFlow, Hugging Face, and scikit-learn, which are widely supported and documented. It is compatible with existing operating systems (Windows, Linux, macOS) and can be executed on both CPU and GPU hardware, ensuring smooth development and deployment without requiring specialized infrastructure.

1.6.2 Economic Feasibility

The project is highly economically feasible as it can be developed using standard, cost-effective personal computers. Leveraging open-source technologies like Python programming language, pre-trained models, and libraries ensures there are no licensing costs. Datasets and resources required for development are freely available online, minimizing financial expenses. This makes the project a budget-friendly option for students and researchers.

1.6.3 Operational Feasibility

The system's operation is designed to be simple and user-friendly, requiring only a computer to install and run the package. With a clear and straightforward interface, users can easily test and evaluate Nepali text for hate speech detection. This ease of use ensures the system's operational viability, making it accessible for academic and practical applications alike.

1.6.4 Schedule Feasibility

Team with good learning capacity and understanding of the project ensures the completion of a ready product in 11 months.

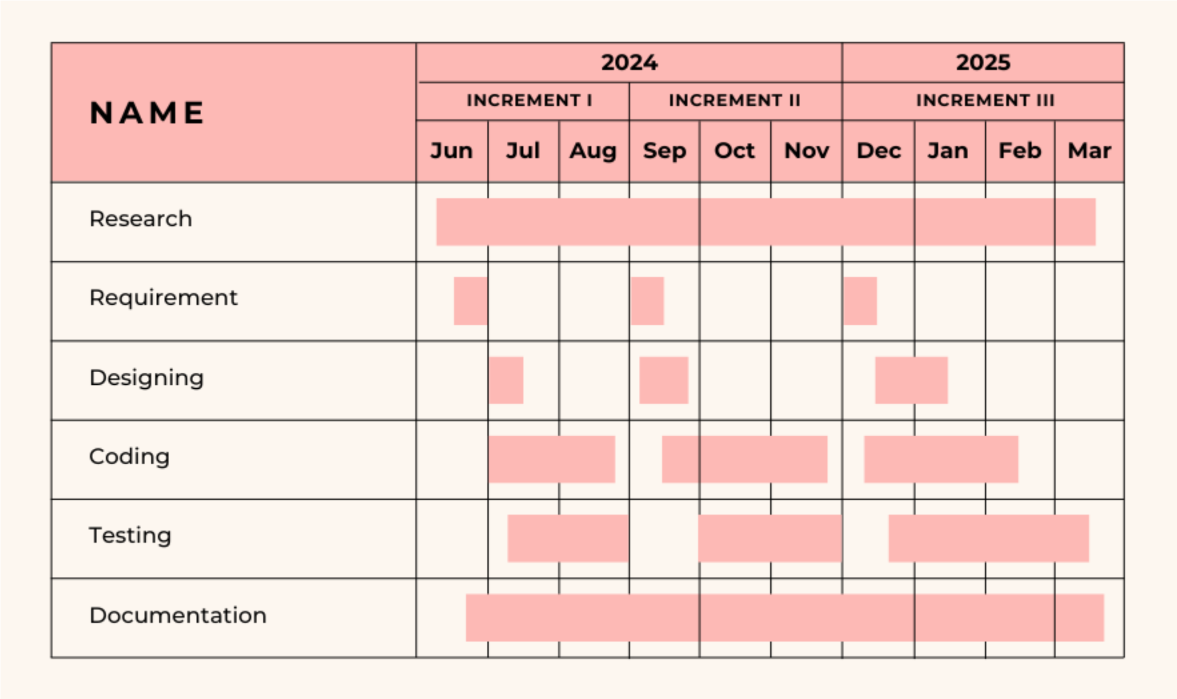


Figure 1.1: Gantt Chart

The team's project was initially on Topic modelling of Resumes and Job Descriptions. Job Descriptions can be scraped through a popular site 'Indeed'. However, the resumes on 'Hireitpeople' were deemed unsufficient. Our objective was to have a topic model that can provide topic distribution for resumes and job descritions. To cover all jobs in recent time is a job of large scale, so the focus was on technical jobs. Job Descriptions, for each job scraped around thousand documents, were ready. Resume for Software Engineers, Network administrators, Web Developer, Project Manager, Database Administrators and Security Analyst are widely available, but unfortunately other jobs such as technical support specialists, video game developers are scarce on the internet. The option of scraping Linkedin is problematic as it requires login session, and Linkedin have a rate limit of searching and viewing profile(80 per day) on new account and accounts with limited connectivity. Another option which, would be best if possible, was ProxyCurl. ProxyCurl manages huge database of linkedin profile, and provides its data through API(Application Programming Interface) for commercial use. This means that each profile scrap costs $0.020, and for our scale i.e 40000 resumes for 40 technical jobs could cost well over $800. This predicament led to change of our Project.

CHAPTER 2
LITREATURE REVIEW

2.1 Related Work

The paper “Nepali Offensive Language Detection Using Transfer Learning Techniques” [3] explores offensive language detection and sentiment classification in Nepali text using pre-trained models like XLM-RoBERTa, BERT, NepBERTa, and distilBERT-base-Nepali. For Named Entity Recognition (NER), distilBERT-base-Nepali achieved 36.77% F1-score with 85.67% accuracy, while deberta-base-Nepali reached an F1-score of 40.14% and 86.6% accuracy. In sentiment analysis, distilBERT-base-Nepali excelled with a 72.24% accuracy and 71.8% F1-score. However, since the NepSA dataset was created for aspect term extraction, and Sentiment classification, even with balanced dataset, has duplicate Text for a Class with Same or different class Polarity. This arises the question of data leakage while training the models which in turn leads to optimistic performance. This issue is addressed in our project by cleaning such instances.

Paper, ”Named-entity based sentiment analysis of Nepali news media texts”, scrapped text data from Nepali news medias: Kantipur Daily, NagarikNews, Online Khabar and Setopati, maintaining dataset with 2676 positive instance and 814 negative [4]. Classification was done by balancing each class to 814, which saw F1- score of 80.2 using SVM. The experimentation was done on SVM, Random Forrest and Decision Tree, while the embeddings for these classifiers were generated by testing Word2Vec and FastText.

Hate speech and abusive language have become a common phenomenon on Arabic social media. Automatic hate speech and abusive detection systems can facilitate the prohibition of toxic textual contents. The complexity, informality, and ambiguity of the Arabic dialects hindered the provision of the needed resources for Arabic abusive/hate speech detection research. The paper, ”L-hsab: A levantine twitter dataset for hate speech and abusive language” introduces the first publicly-available Levantine Hate Speech and Abusive (L-HSAB) Twitter dataset with Krippen-dorff’s /alpha value 76.5% [5]. The dataset consists of 5,846 tweets, labeled as Hate, Abusive, or Normal. To ensure a reliable annotation process, the authors provided clear guidelines, inter-annotator agreement metrics, and conducted evaluations to establish the dataset's consistency. Machine learning classifiers such as Naive Bayes and SVM were tested, with Naive Bayes achieving better results in both binary and multi-class classification tasks. Naive Bayes from NLTK achieved an average of 82% F1-score among Hate and Abusive class. Both classifiers were trained with several n-gram schemes with Term frequency weighting.

In their 2022 study, Maskey et al. conducted a comparative analysis of Transformer-based language models for Nepali text classification, focusing on DistilBERT, DeBERTa, and XLM-RoBERTa. The research aimed to identify effective strategies for pre-training and fine-tuning these models within the context of a low-resource language. The models were evaluated on a downstream classification task using the ”16 Nepali News” dataset, which comprises approximately 14,364 news documents categorized into 16 distinct groups. The findings revealed that all models surpassed the baseline accuracy of 80%. DeBERTa achieved the highest accuracy at 88.93%, highlighting the significance of training domain-adapted language models. DistilBERT attained a respectable accuracy of 88.31% with the least number of training steps, implying faster training for downstream tasks. The marginal performance difference compared to DeBERTa, combined with its smaller and lighter architecture, makes DistilBERT particularly well-suited for deployment in production environments with computational constraints. This study underscores the potential of leveraging efficient Transformer models like DistilBERT for low-resource languages, providing a foundation for future NLP research and applications in Nepali.

CHAPTER 3
METHODOLOGY

3.1 Working Mechanism

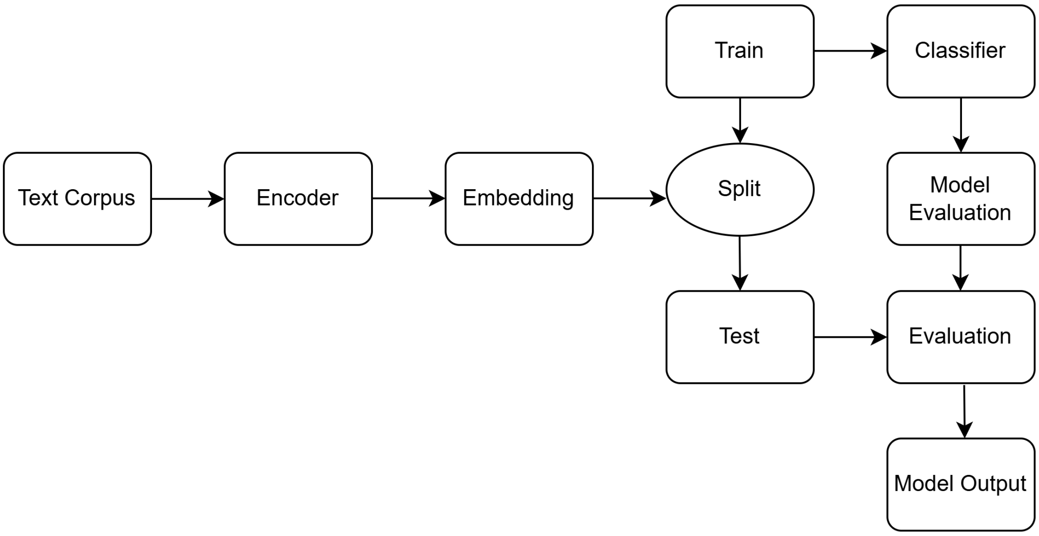
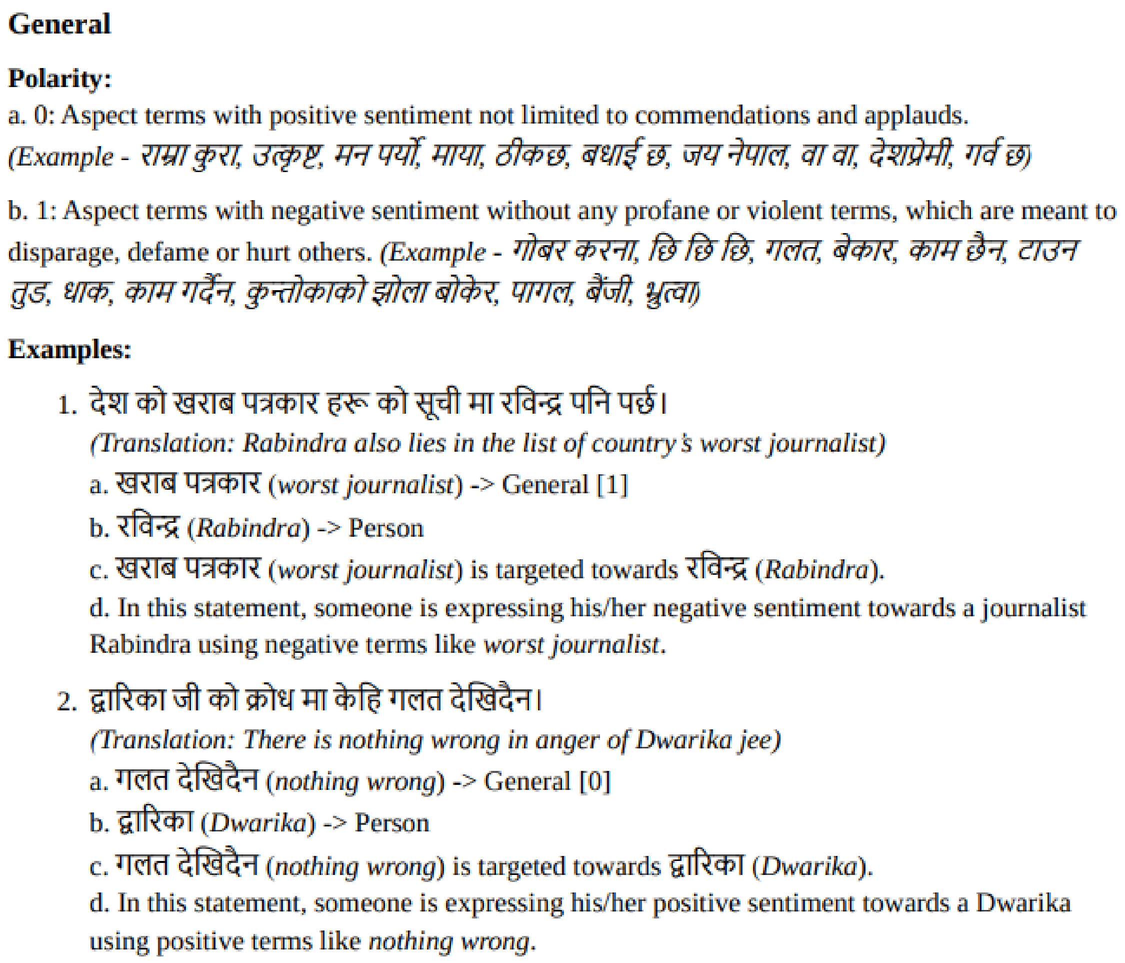


Figure 3.1: Block diagram of Working Mechanism

The proposed structure of the project includes different steps. The texts for sentiment classification after preprocessing are embedded into feature vectors through the usage of popular encoders such as Word2Vec. These encoded features are then split into Train and test set at the ratio of 7:3.

3.1.1 Data Collection

The Nepali Sentiment Analysis (NepSA) dataset serves as the foundation for our project, providing valuable insights into abusive content in the Nepali language. This dataset was specifically chosen for its unique features, which addresses the complex linguistic challenges associated with abusive language detection in Nepali [6]. The dataset was compiled from comments collected from popular Nepali YouTube channels, primarily within the News and Politics category, focusing on channels with the highest subscriber base. It consists of 3,068 comments extracted from 37 videos across 9 distinct channels. The dataset follows a binary sentiment polarity framework, categorizing comments into key aspects: General, Profanity, Violence. The total count of instances in the dataset account to 3529. Each comment is annotated based on the sentiment expressed towards a specific target entity, enabling a more detailed and context-specific understanding of the language rather than a generalized sentiment analysis.



General

Polarity:
a. 0: Aspect terms with positive sentiment not limited to commendations and applauds.
(Example - राम्रा कुरा, उत्कृष्ट, मन पर्यो, माया, ठीकछ, बधाई छ, जय नेपाल, वा वा, देशप्रेमी, गर्व छ)

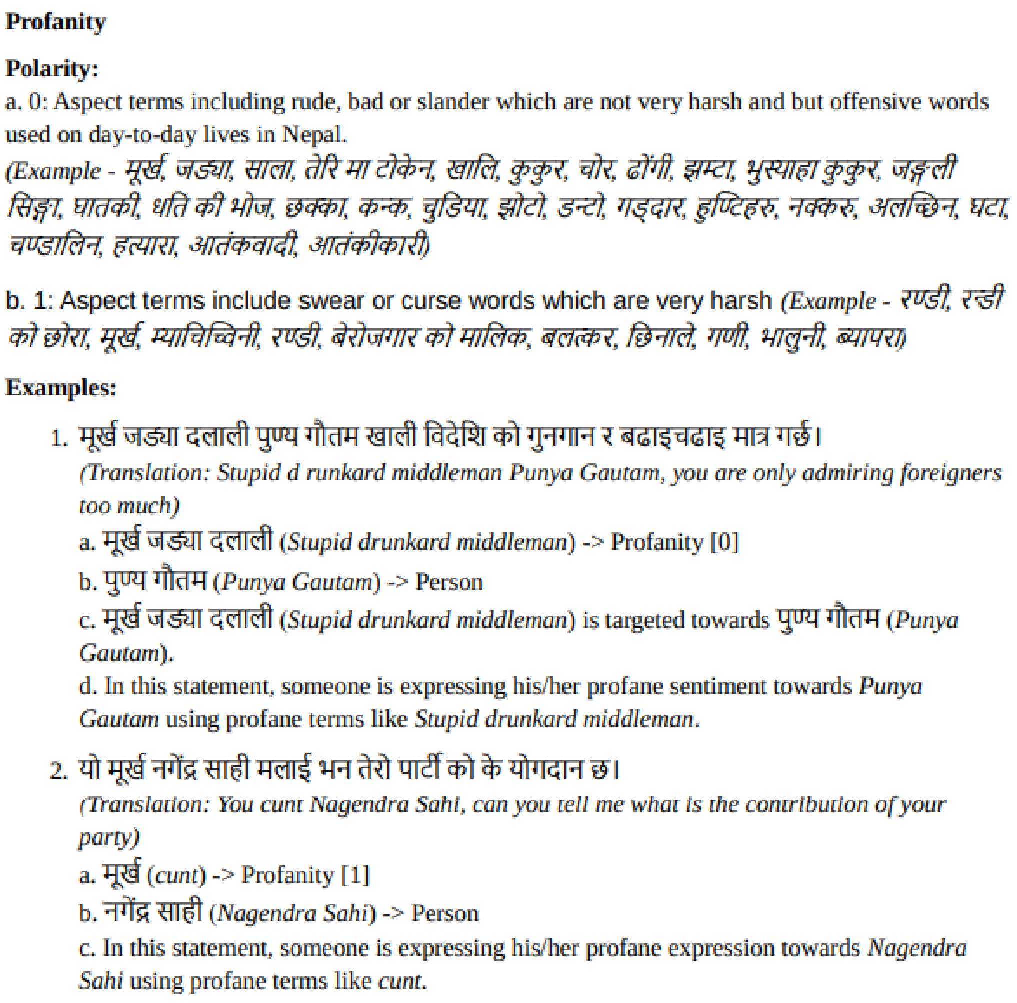
b. 1: Aspect terms with negative sentiment without any profane or violent terms, which are meant to disparage, defame or hurt others. *(Example - गोबर करना, छि छि छि, गलत, बेकार, काम छैन, टाउन तुड, धाक, काम गर्दैन, कुन्तोकाको झोला बोकेर, पागल, बैंजी, भुत्वा)*

Examples:

1. देश को खराब पत्रकार हरू को सूची मा रविन्द्र पनि पर्छ।
   (Translation: Rabindra also lies in the list of country's worst journalist)
   a. खराब पत्रकार (*worst journalist*) -> General [1]
   b. रविन्द्र (*Rabindra*) -> Person
   c. खराब पत्रकार (*worst journalist*) is targeted towards रविन्द्र (*Rabindra*).
   d. In this statement, someone is expressing his/her negative sentiment towards a journalist Rabindra using negative terms like *worst journalist*.

2. द्वारिका जी को क्रोध मा केहि गलत देखिदैन।
   (Translation: There is nothing wrong in anger of Dwarika jee)
   a. गलत देखिदैन (*nothing wrong*) -> General [0]
   b. द्वारिका (*Dwarika*) -> Person
   c. गलत देखिदैन (*nothing wrong*) is targeted towards द्वारिका (*Dwarika*).
   d. In this statement, someone is expressing his/her positive sentiment towards a Dwarika using positive terms like *nothing wrong*.

Figure 3.2: Class example 1



Profanity

Polarity:

a. 0: Aspect terms including rude, bad or slander which are not very harsh and but offensive words used on day-to-day lives in Nepal.

(Example - मूर्ख, जड्या, साला, तेरि मा टोकेन, खालि, कुकुर, चोर, ढोंगी, झम्टा, भुस्याहा कुकुर, जङ्गली सिङ्गा, घातकी, धति की भोज, छक्का, कन्क, चुडिया, झोटो, डन्टो, गड्दार, हुण्टिहरु, नक्करु, अलच्छिन, घटा, चण्डालिन, हत्यारा, आतंकवादी, आतंकीकारी)

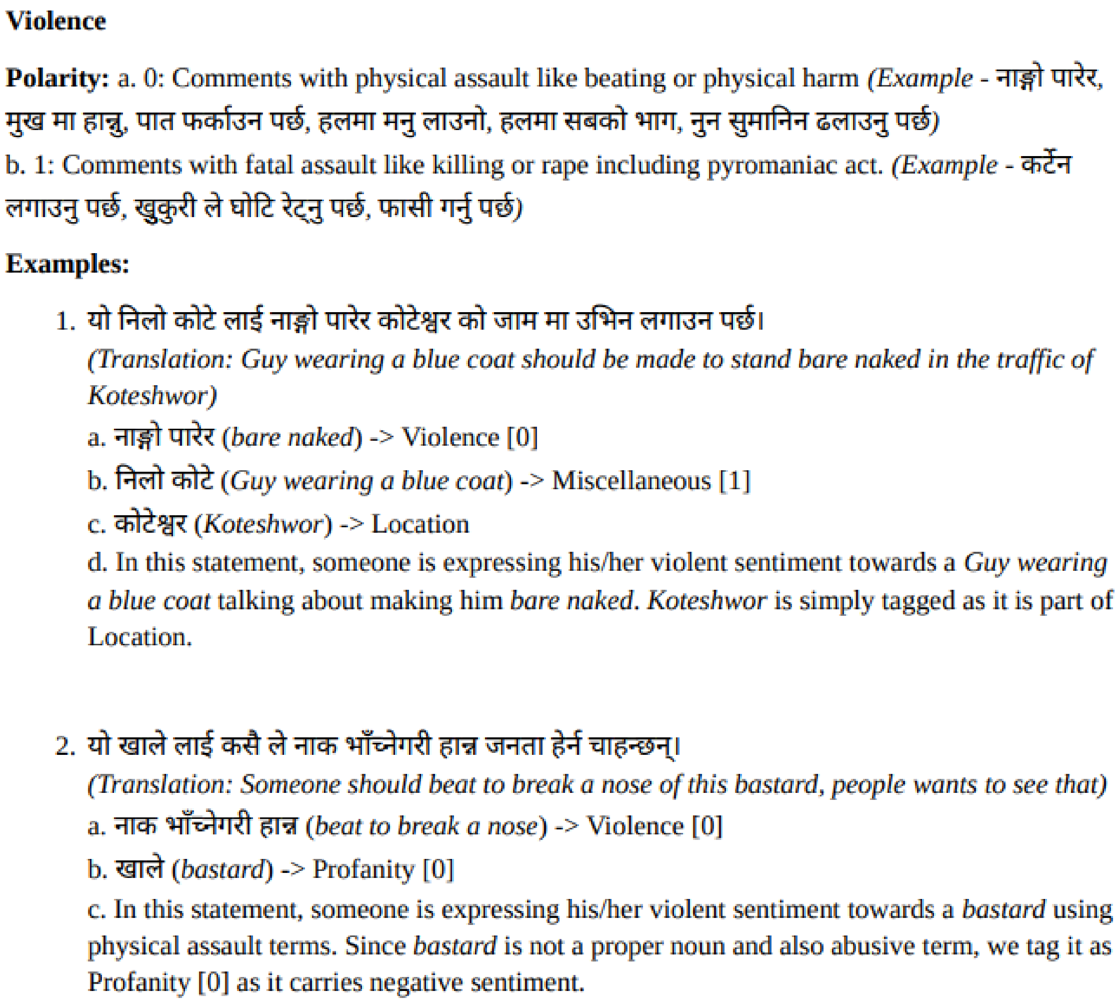
b. 1: Aspect terms include swear or curse words which are very harsh *(Example - रण्डी, रन्डी को छोरा, मूर्ख, म्याचिच्विनी, रण्डी, बेरोजगार को मालिक, बलत्कर, छिनाले, गणी, भालुनी, ब्यापरा)*

Examples:

1. मूर्ख जड्या दलाली पुण्य गौतम खाली विदेशि को गुनगान र बढाइचढाइ मात्र गर्छ।
   (Translation: Stupid d runkard middleman Punya Gautam, you are only admiring foreigners too much)
   a. मूर्ख जड्या दलाली (*Stupid drunkard middleman*) -> Profanity [0]
   b. पुण्य गौतम (*Punya Gautam*) -> Person
   c. मूर्ख जड्या दलाली (*Stupid drunkard middleman*) is targeted towards पुण्य गौतम (*Punya Gautam*).
   d. In this statement, someone is expressing his/her profane sentiment towards *Punya Gautam* using profane terms like *Stupid drunkard middleman*.

2. यो मूर्ख नगेंद्र साही मलाई भन तेरो पार्टी को के योगदान छ।
   (Translation: You cunt Nagendra Sahi, can you tell me what is the contribution of your party)
   a. मूर्ख (*cunt*) -> Profanity [1]
   b. नगेंद्र साही (*Nagendra Sahi*) -> Person
   c. In this statement, someone is expressing his/her profane expression towards *Nagendra Sahi* using profane terms like *cunt*.

Figure 3.3: Class example 2



Violence

Polarity: a. 0: Comments with physical assault like beating or physical harm *(Example* - नाङ्गो पारेर, मुख मा हान्नु, पात फर्काउन पर्छ, हलमा मनु लाउनो, हलमा सबको भाग, नुन सुमानिन ढलाउनु पर्छ)
b. 1: Comments with fatal assault like killing or rape including pyromaniac act. *(Example* - कर्टेन लगाउनु पर्छ, खुकुरी ले घोटि रेट्नु पर्छ, फासी गर्नु पर्छ)

Examples:

1. यो निलो कोटे लाई नाङ्गो पारेर कोटेश्वर को जाम मा उभिन लगाउन पर्छ।
   (Translation: Guy wearing a blue coat should be made to stand bare naked in the traffic of Koteshwor)

   a. नाङ्गो पारेर (*bare naked*) -> Violence [0]

   b. निलो कोटे (*Guy wearing a blue coat*) -> Miscellaneous [1]

   c. कोटेश्वर (*Koteshwor*) -> Location

   d. In this statement, someone is expressing his/her violent sentiment towards a *Guy wearing a blue coat* talking about making him *bare naked*. *Koteshwor* is simply tagged as it is part of Location.

2. यो खाले लाई कसै ले नाक भाँच्नेगरी हान्न जनता हेर्न चाहन्छन्।
   (Translation: Someone should beat to break a nose of this bastard, people wants to see that)

   a. नाक भाँच्नेगरी हान्न (*beat to break a nose*) -> Violence [0]

   b. खाले (*bastard*) -> Profanity [0]

   c. In this statement, someone is expressing his/her violent sentiment towards a *bastard* using physical assault terms. Since *bastard* is not a proper noun and also abusive term, we tag it as Profanity [0] as it carries negative sentiment.

Figure 3.4: Class example 3

3.1.2 Data Preprocessing

The process of pre-processing the data is very important as good data is very important for effective learning by Classifiers.

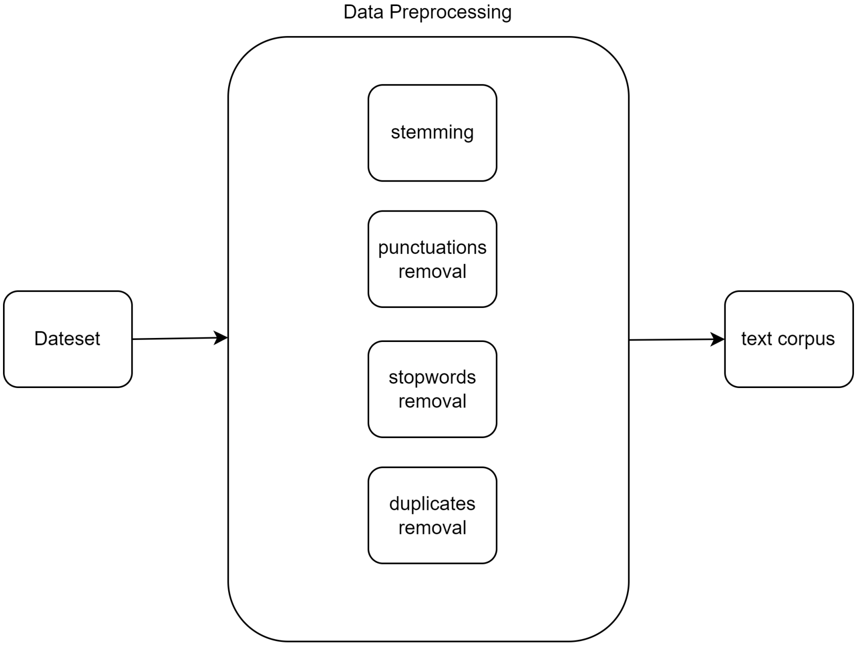


Figure 3.5: Block diagram of Data Preprocessing

The dataset was prepared using Nepali Stemmer.Furthermore, the punctuations in texts were removed followed by stopwords removal using NLTK stopwords package. Since, the dataset was created with priority on aspect term extraction there are instances of same text classified as duplicates with different Aspect(in same class, and different class). These duplicates, in class with higher count, was removed to have unique instances for training and avoid data leakage. By removing such duplicates the data instances amount to 2859.

Inorder to classify among textual multi-classes: ”GENERAL”, ”VIOLENCE” and ”PROFANITY”, label encoding is done.

3.1.3 Machine Learning Techniques

Support Vector Machine(SVM)

Support Vector Machine (SVM) constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. The Support Vector Machine (SVM) stands as a supervised machine learning algorithm with applications in both classification and regression tasks. While SVM is versatile enough to handle regression problems, its optimal utility lies in the domain of classification. The central objective of the SVM algorithm involves the identification of a hyperplane within an N-dimensional space that distinctly segregates the data points. The dimensionality of this hyperplane is contingent upon the quantity of input features. In the context of hate speech detection, the SVM algorithm aims to segregate hate speech from non-hate speech based on features extracted from the text. For two input features, the hyperplane is represented as a line, while for three input features, it assumes the form of a 2-dimensional plane. As the number of features increases, the hyperplanes become increasingly complex and multidimensional, enabling precise classification even in high-dimensional feature spaces [7].

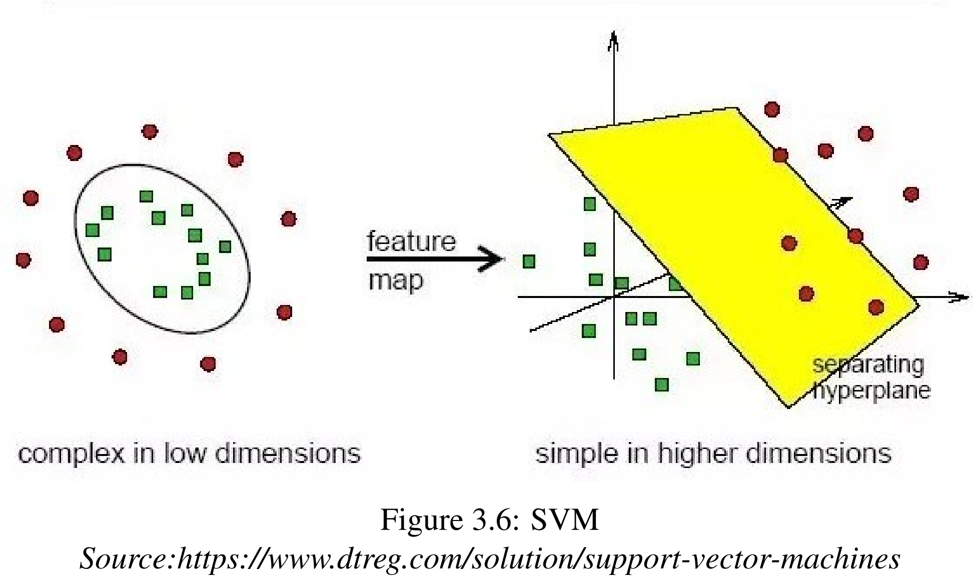


Figure 3.6: SVM
Source:https://www.dtreg.com/solution/support-vector-machines

AdaBoost

AdaBoost is a type of ensemble learning technique that operates iteratively, capitalizing on the errors of weak classifiers and amplifies their capabilities to mold them into potent classifiers. The essence of AdaBoost lies in its ability to combine several weak classifiers, enhancing the ones that showcase superior performance through higher weight assignments.

AdaBoost refers to a particular method of training a boosted classifier. A boosted classifier is a classifier of the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

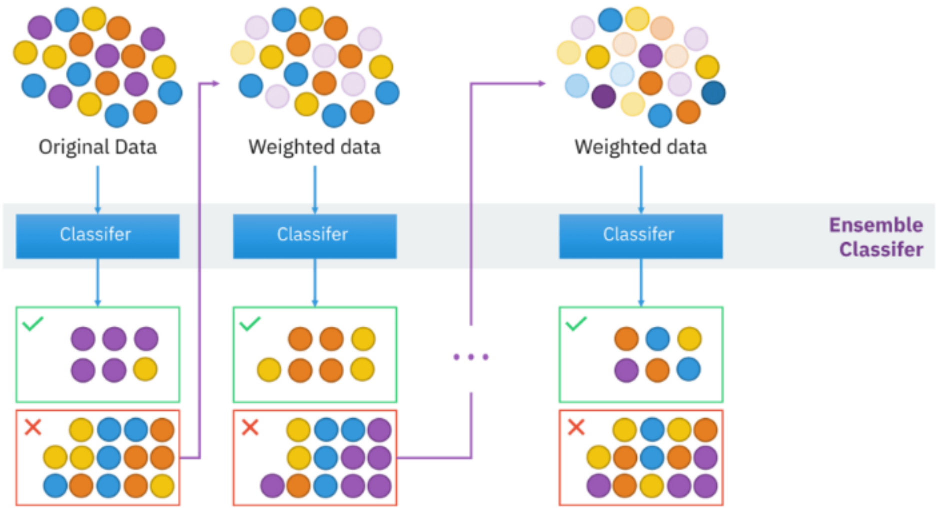
where each f_t is a weak learner that takes an object x as input and returns a value indicating the class of the object. For example, in the two-class problem, the sign of the weak learner's output identifies the predicted object class and the absolute value gives the confidence in that classification.

Each weak learner produces an output hypothesis h which fixes a prediction $h(x_i)$ for each sample in the training set. At each iteration t, a weak learner is selected and assigned a coefficient α_t such that the total training error E_t of the resulting t-stage boosted classifier is minimized.

$$E_t = \sum_{i} E\big[F_{t-1}(x_i) + \alpha_t h(x_i)\big]$$

Here $F_{t-1}(x)$ is the boosted classifier that has been built up to the previous stage of training and $f_t(x) = \alpha_t h(x)$ is the weak learner that is being considered for addition to the final classifier.

In the context of hate speech detection, the AdaBoost algorithm tries to improve clas-



sification by combining multiple simple models (weak learners) to create a stronger one. It works by focusing more on mistakes—if a weak model misclassifies some hate speech, the next model pays more attention to those errors. This way, each new model improves upon the previous one. AdaBoost creates a decision boundary that becomes more refined and accurate after several iterations, making it better at hate speech detection.

3.1.4 Word2Vec

Word2Vec is a neural network-based algorithm used to generate vector representations of words, capturing their semantic relationships within a text corpus. Word2Vec maps words into a continuous vector space where semantically similar words are positioned closer together. This results in dense word embeddings that effectively represent both syntactic and semantic properties of words.

Working Mechanism of Word2Vec

Word2Vec operates using two primary architectures: Continuous Bag of Words (CBOW) and Skip-Gram.

- CBOW predicts a target word based on its surrounding context words. For example, in the sentence “The cat sat on the ___,” the model predicts the missing word “mat” using the words around it. This method is efficient for larger datasets and captures general word associations.

- Skip-Gram works inversely, predicting the context words around a given target word. For example, given the word “cat,” the model predicts surrounding words like “the” or “sat.” Skip-Gram is particularly effective for learning embeddings for rare words, as it focuses on the word’s relationships in specific contexts.

Both architectures use a shallow neural networks to optimize the word embeddings. Through training, the model minimizes the error in predictions, adjusting the word vectors to reflect their contextual similarities and differences. This results in embeddings where words with similar meanings or usage patterns are positioned closer together in the vector space. In this project, Word2Vec is used to generate embeddings for Nepali text, enabling the detection of abusive and offensive language within specific contexts. By understanding the relationships between words such as ‘thief’ or ‘dog’ (in nepali text), the algorithm allows the system to identify clusters of abusive terms and interpret them based on their context. This enhances the accuracy and robustness of the hate speech detection model.

For embedding generation, we employed a pre-trained Word2Vec model which has 300-dimensional vectors for more than 0.5 million Nepali words and phrases [8] on CBOW architecture. This model was fine-tuned with our text corpus.

3.1.5 TF-IDF Formula

TF-IDF is a statistical measure that quantifies the importance of words in a document relative to a collection (corpus) of documents. It reflects how relevant a term is to a document by balancing two metrics: Term Frequency (TF) and Inverse Document Frequency (IDF).

1. **Term Frequency (TF):**

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

2. **Inverse Document Frequency (IDF):**

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t} + 1\right)$$

3. **TF-IDF Score:**

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

3.1.6 NepBERTa and Its Working Mechanism

NepBERTa is a specialized pre-trained language model for the Nepali language based on the robust BERT (Bidirectional Encoder Representations from Transformers) architecture. It uses deep learning to produce word embeddings tailored specifically to the linguistic nuances of Nepali. Unlike traditional models, NepBERTa understands the context of a word in all directions (left and right of the word), allowing it to generate more accurate and nuanced embeddings for Nepali text.

Steps for NepBERTa

Step 1: Tokenization

The input Nepali text is tokenized into a sequence of tokens,

$$x = [x_1, x_2, ..., x_n] \tag{3.1}$$

using a WordPiece tokenizer customized for the Nepali language. This ensures that the model can efficiently handle Nepali-specific words and subwords.

Step 2: Embeddings

Each token is mapped to a dense vector space using a learned embedding matrix **E**. The embedding for each token is the sum of three components:

$$e_i = E(x_i) + E_{seg}(x_i) + E_{pos}(i) \tag{3.2}$$

Where:

- $E_{seg}(x_i)$: Segment embedding, indicating whether the token belongs to the first or second segment (e.g., sentence A or sentence B).

- $E_{pos}(i)$: Position embedding, representing the token's position in the sequence.

Step 3: Encoder

The embedded tokens are passed through a multi-layer bidirectional Transformer encoder, consisting of L identical layers. Each layer applies self-attention and a feed-forward neural network (FFNN):

$$h^l = \text{TransformerLayer}(h^{l-1}) \tag{3.3}$$

Where:

- h^l: The output of the l-th layer.
- h^0: The embedding input.

Step 4: Pooling The final output of the encoder, h^L, is a sequence of vectors. The first vector, $h^L[0]$, is used as the overall representation of the input sequence:

$$p = h^L[0] \tag{3.4}$$

Step 5: Pre-training Tasks

NepBERTa is pre-trained using Masked Language Modeling (MLM), where random tokens in the Nepali text are masked, and the model learns to predict the original tokens based on the context. The MLM loss is defined as:

$$L_{\text{mlm}} = -\sum x_i \log \big(p(x_i \mid x_{-i})\big) \tag{3.5}$$

3.1.7 DistilBERT and its Key Methodologies

DistilBERT is a streamlined version of the BERT (Bidirectional Encoder Representations from Transformers) model, designed to enhance efficiency in natural language processing tasks. It reduces BERT's size by 40%, offers 60% faster inference, and is 71% faster on mobile applications. Despite being compact, it retains 97% of BERT's accuracy, making it an effective alternative to BERT. It achieves this through several key methodologies:

Knowledge Distillation

Knowledge distillation is a model compression technique where a smaller model (the student) is trained to replicate the behavior of a larger, pre-trained model (the teacher). In the case of DistilBERT, the BERT model serves as the teacher. The student model learns to predict the output probabilities of the teacher, capturing the nuanced information learned by the larger model. This process enables the student model to achieve high performance with reduced complexity.

Loss Functions in DistilBERT

1. **Distillation Loss (KL Divergence Loss)** Distillation hello loss is essential in training the student model during knowledge distillation. It's calculated by comparing the student's prediction probability to the soft probabilities provided by the teacher model. The soft probabilities for the student model are computed as:

   $$(y_S^T)_i = \frac{\exp((z_S)_i/T)}{\sum_j \exp((z_S)_j/T)}$$

   The distillation loss is then calculated as:

   $$\mathcal{L}_{KD} = \alpha T^2 * \mathcal{L}_{CE}(y_T^T, y_S^T) + (1-\alpha) * \mathcal{L}_{CE}(y_S, y_{true})$$

   where: **temperature T** smooths the probability distribution, making it easier for the student model to learn L(CE) KL divergence is used as the loss function to measure the difference between the teacher's and student's probability distributions.
2. **Masked Language Modeling (MLM) Loss**

   Like BERT, DistilBERT learns language patterns using masked language modeling (MLM). In this approach, some words in a sentence are randomly replaced with a special token, and the model predicts these masked words based on surrounding context. The loss is computed as:

$$\mathcal{L}_{MLM} = \sum -y_{true} \log(y_S)$$

3. **Cosine embedding loss** Cosine embedding loss helps the student model match the teacher model's embeddings by minimizing their cosine distance, preserving knowledge while reducing model size. Cosine similarity is defined as:

$$\cos(\mathbf{x_1}, \mathbf{x_2}) = \frac{\mathbf{x_1} \cdot \mathbf{x_2}}{||\mathbf{x_1}||||\mathbf{x_2}||}$$

   The Cosine embedding loss is computed as:

$$\mathcal{L}_{cos}(\mathbf{x_1}, \mathbf{x_2}) = 1 - \cos(\mathbf{x_1}, \mathbf{x_2})$$

   To effectively train the student model, DistilBERT employs a triple loss function that combines Distillation loss, MLM loss and Cosine embedding loss. The final training loss function is a weighted sum of the three loss components:

$$\mathcal{L} = \alpha_d T^2 * \mathcal{L}_{dist} + \alpha_{mlm} * \mathcal{L}_{MLM} + \alpha_{cos} * \mathcal{L}_{cos}$$

   The following hyperparameter values were used during the training of DistilBERT:
   Temperature (T) = 2.0
   Distillation weight () = 5.0
   MLM loss weight () = 2.0
   Cosine loss weight () = 1

3.1.8 DistilBERT Architecture

DistilBERT reduces the number of layers to 6 from 12 in the original BERT architecture, resulting in a model with 66M parameters. This reduction leads to a 60% increase in inference speed while maintaining approximately 97% of BERT's language understanding capabilities.

In the context of developing efficient natural language processing models for the Nepali language, DistilBERT emerges as a compelling choice due to its reduced model size and computational efficiency. A study titled ”Nepali Encoder Transformers: An Analysis of Auto Encoding Transformer Language Models for Nepali Text Classification” highlights the advantages of using smaller transformer models like DistilBERT. The research emphasizes that such models are particularly beneficial for low-resource languages like Nepali, where computational resources may be limited. By employing DistilBERT, which maintains a balance between performance and efficiency, it becomes feasible to develop robust language models without the need for extensive computational infrastructure.

Architecture Modification

DistilBERT reduces the number of layers in the original BERT architecture by half, resulting in a model with 40% fewer parameters. This reduction leads to a 60% increase in inference speed while maintaining approximately 97% of BERT’s language understanding capabilities. The streamlined architecture makes DistilBERT more suitable for deployment in resource-constrained environments. In the context of developing efficient natural language processing models for the Nepali language, DistilBERT emerges as a compelling choice due to its reduced model size and computational efficiency. A study titled ”Nepali Encoder Transformers: An Analysis of Auto Encoding Transformer Language Models for Nepali Text Classification” highlights the advantages of using smaller transformer models like DistilBERT. The research emphasizes that such models are particularly beneficial for low-resource languages like Nepali, where computational resources may be limited. By employing DistilBERT, which maintains a balance between performance and efficiency, it becomes feasible to develop robust language models without the need for extensive computational infrastructure.

Condensed Nearest Neighbour (CNN)

Condensed Nearest Neighbour (CNN) is a data reduction technique used to improve the efficiency of classification tasks, especially in small datasets. It works by selecting only the most informative samples that define decision boundaries, ensuring misclassified instances are retained while redundant data is removed. This makes CNN effective in simplifying datasets, reducing noise, and enhancing model performance without losing critical information.

Neighbourhood Cleaning Rule (NCR)

Neighbourhood Cleaning Rule (NCR) is an undersampling technique used to improve dataset quality by focusing on removing noisy or ambiguous samples. It works by identifying majority-class samples that are either misclassified by their nearest neighbours or surrounded by minority-class neighbours. These samples are removed, resulting in a cleaner and more balanced dataset. NCR is particularly useful for small datasets as it retains critical decision boundaries while reducing noise and ambiguity, improving the overall performance of classification models.

Random Under Sampler

Random Under Sampler is a simple undersampling technique used to handle class imbalance in datasets. It works by randomly removing samples from the majority class until its size matches the minority class. This helps balance the dataset and prevents the model from being biased toward the majority class. While it is fast and easy to implement, it can potentially remove important data points. It is most suitable when working with small datasets, as it reduces size without requiring extensive computational resources.

SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is an oversampling method used to address class imbalance in datasets by generating synthetic samples for the minority class. It works by selecting a sample from the minority class and creating new synthetic data points by interpolating between that sample and one of its nearest neighbours. This process helps balance the dataset without simply duplicating existing samples. SMOTE is effective in improving model performance, especially for small datasets, as it enhances representation of the minority class while preserving the overall data distribution.

ADASYN (Adaptive Synthetic Sampling)

ADASYN is an advanced oversampling technique that generates synthetic samples for the minority class, focusing more on harder-to-classify instances. It works by identifying samples in the minority class that are difficult to classify (i.e., surrounded by majority-class neighbours) and generating more synthetic data for these instances. This adaptive approach ensures better balance while improving decision boundaries for the minority class. ADASYN is particularly effective in small datasets with complex class distributions, as it reduces bias and enhances model performance in imbalanced datasets.

3.2 SONAR(Sentence-Level Multimodal and Language-Agnostic Representations)

SONAR, developed by Facebook Research, is a multilingual and multimodal embedding model that provides fixed-size representations for both text and speech. SONAR leverages deep learning architectures to generate embeddings that capture semantic relationships between words and sentences. The model supports over 200 languages, making it effective for low-resource languages like Nepali. Unlike traditional techniques like TF-IDF and Word2Vec, SONAR embeddings retain word meanings based on context. Text from different languages is embedded into a common space, allowing cross-lingual transfer.

3.2.1 Why over other models

SONAR outperforms LASER3 and LaBSE on xsim and xsim++ similarity search tasks. Unlike models like BERT and XLM-R, which provide variable-length representations, SONAR produces consistent 1024-dimensional embeddings.

3.2.2 How SONAR Works

SONAR transforms text and speech into embeddings using a Transformer-based encoder-decoder model. The process follows:

3.2.3 Sentence Tokenization

Given a sentence S consisting of words:

$$S = \{w_1, w_2, \dots, w_n\} \tag{3.6}$$

3.2.4 Encoding Function

SONAR applies a function f to transform the sequence into a fixed-size vector:

$$E(S) = f(w_1, w_2, \dots, w_n) \tag{3.7}$$

where $E(S)$ is a 1024-dimensional embedding vector.

3.2.5 Training Objectives

SONAR is trained using multiple loss functions:

- **Translation Loss** (L_{MT}): Ensures that sentence embeddings are effective for translation tasks.
- **Mean Squared Error Loss** (L_{MSE}): Aligns embeddings of different languages.

- **Denoising Auto-Encoding Loss** (L_{DAE}): Improves robustness by reconstructing noisy inputs.

The final training objective combines these:

$$L = L_{MT} + \alpha \cdot L_{MSE} + \beta \cdot L_{AE/DAE} \tag{3.8}$$

3.3 SONAR Architecture

SONAR follows a two-step training process:

3.3.1 Multilingual Text Sentence Embedding Space

- Uses an encoder-decoder Transformer model.
- Pretrained using the NLLB 1B model.
- Embeddings are generated through mean-pooling on token-level outputs.

3.3.2 Speech Sentence Embedding Space (Teacher-Student Approach)

- A speech encoder is trained to match text embeddings.
- Uses pre-trained w2v-BERT for initialization.
- Speech embeddings are mapped to the same 1024-dimensional space.

3.4 Differential Evolution

Differential evolution is a stochastic population-based method that is useful for global optimization problems. At each pass through the population, the algorithm mutates each candidate solution by mixing with other candidate solutions to create a trial candidate. There are several strategies for creating trial candidates, which suit some problems more than others.

The ’best1bin’ strategy is a good starting point binomial distribution. In this strategy, two members of the population are randomly chosen. Their difference is used to mutate the best member (the ‘best’ in ‘best1bin’), x_0, so far:

$$b' = x_0 + F \cdot (x_{r0} - x_{r1}) \tag{3.9}$$

where, F is the *mutation* parameter, x_0 is the base vector, x_{r0} and x_{r1} are two randomly selected vectors from the population. A trial vector is then constructed. Starting with a randomly chosen ith parameter, the trial is sequentially filled (in modulo) with parameters from b' or the original candidate. The choice of whether to use b' or the original candidate is made with a binomial distribution (the ‘bin’ in ‘best1bin’) - a random number in $[0, 1]$ is generated. If this number is less than the *recombination* constant, then the parameter is loaded from b', otherwise, it is loaded from the original candidate. The final parameter is always loaded from b'.

Once the trial candidate is built, its fitness is assessed. If the trial is better than the original candidate, then it takes its place. If it is also better than the best overall candidate, it also replaces that.

3.4.1 Parameters of Differential Evolution

1. **func : callable**

   The objective function to be minimized. Must be in the form f(x, *args), where x is the argument in the form of a 1-D array and args is a tuple of any additional fixed parameters needed to completely specify the function.

2. **bounds : sequence**

   Bounds for variables. (min, max) pairs for each element in x, defining the lower and upper bounds for the optimizing argument of func. It is required to have len(bounds) == len(x). len(bounds) is used to determine the number of parameters in x.

3. **maxiter : int, optional**

   The maximum number of generations over which the entire population is evolved. The maximum number of function evaluations (with no polishing) is:
   $(maxiter + 1) * popsize * len(x)$

4. **tol : float, optional**

   Relative tolerance for convergence, the solving stops when

$np.std(pop) <= atol + tol * np.abs(np.mean(population\_energies))$, where atol and tol are the absolute and relative tolerance respectively.

5. **mutation : float or tuple**

   The mutation constant. In the literature this is also known as differential weight, being denoted by F. If specified as a float it should be in the range [0, 2]. If specified as a tuple (min, max) dithering is employed. Dithering randomly changes the mutation constant on a generation by generation basis. The mutation constant for that generation is taken from U[min, max). Dithering can help speed convergence significantly. Increasing the mutation constant increases the search radius, but will slow down convergence.

6. **recombination : float, optional**

   The recombination constant, should be in the range [0, 1]. Increasing this value allows a larger number of mutants to progress into the next generation, but at the risk of population stability.

Optuna [9] is a software framework used for automating optimization process of hyperparameters. By trial and error process,it automatically searches for and finds the optimal hyperparameter values for excellent performance. It uses a history record of trials inorder to determine which hyperparameter values to try next. Using this data, it estimates a promising area and tries values in that area. Optuna repeats this process using the history data of trials that are completed so far. It employs a Bayesian optimization algorithm called Tree-structured Parzen Estimator [10]. It uses Pruning algorithm which in general works in two phases.

1. It periodically monitors the intermediate objective values
2. It terminates the trial that does not meet the predefined condition.

3.5 Development Model

The Incremental model, which is a mixture of waterfall and iterative development approach is opted, as the project demands few requirements and is fairly simple. As the name suggests, the major objective is focused as the first increment of the project. The activities in this model are completed iteratively and each outcome acts as an input for the next phase, thus increments are developed in such a way.

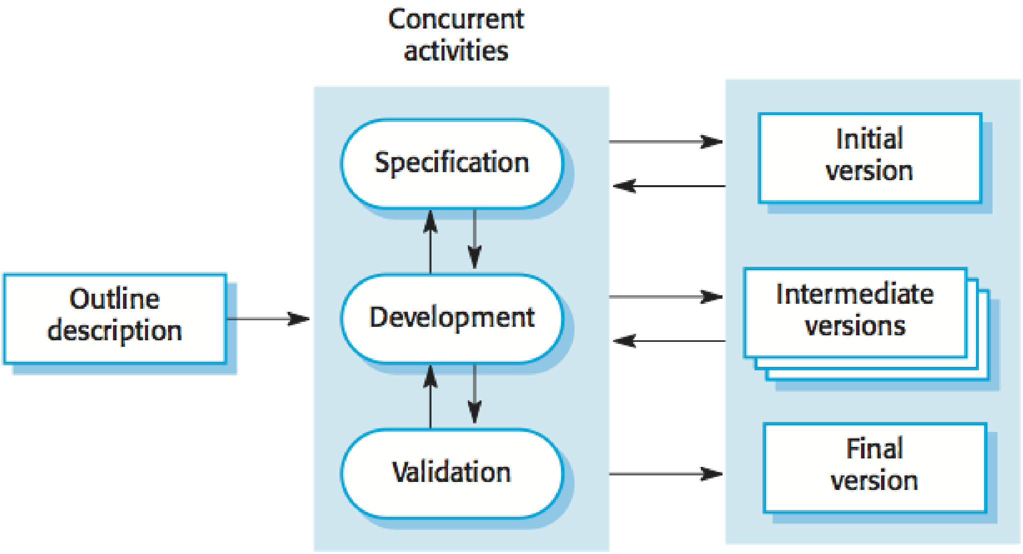


Figure 3.7: Development Model

3.5.1 Requirement Specification

The project requires an optimized model that can effectively classify among different class types while learning the context in texts. Finally, the model is interacted by users by installing package the enables this interaction. Through this package, the users of the project can implement business logic as per the model's classification

3.5.2 Development and Implementation

In this project, an incremental model is employed to continuously enhance and adapt the hate sentiment detection system. This approach allows the model to progressively improve as different encoding techniques and model experimentations are done over time, ensuring the system remains effective in identifying context patterns of hate speech in the Nepali language.

3.5.3 Verification and Validation

In order to approve that the product meets its defined objectives, verification and validation is exercised. This involves Meetings and Code reviews. The code itself is verified with inspections and walkthroughs among the group and even by external parties. Finally, the product is verified by testing its prediction on the testing split of dataset.

3.5.4 Increments

As discussed earlier the project went through several increments. The first increment solely focuses on choosing effective embeddings for Classifiers. The further increments are focused on optimizing the classifier exporting the model for deployment.
Once the model is deployed, scripts are written to interact with the model, which will be deployed as a python package

3.6 Evaluation Criteria

Upon the successful construction of a system model, it undergoes training using the training dataset sourced from the standardized dataset.The effectiveness of the system's performance is assessed by Accuracy, Precision, Recall, and F1 score .

Accuracy:

The ratio of true positives and true negatives to all positive and negative observations is the definition of model accuracy, a performance statistic for machine learning models.

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

Precision:

The percentage of labels that were correctly predicted positively is represented by the model precision score.

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

Recall:

The model's ability to properly forecast the positive out of actual positives is measured by the model recall score.

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

F1 Score:

The F1 score combines precision and recall using their harmonic mean, and maximizing the F1 score implies simultaneously maximizing both precision and recall.

$F1Score = \frac{2*Precision*Recall}{Precision+Recall}$

3.7 System Diagrams

Usecase Diagram

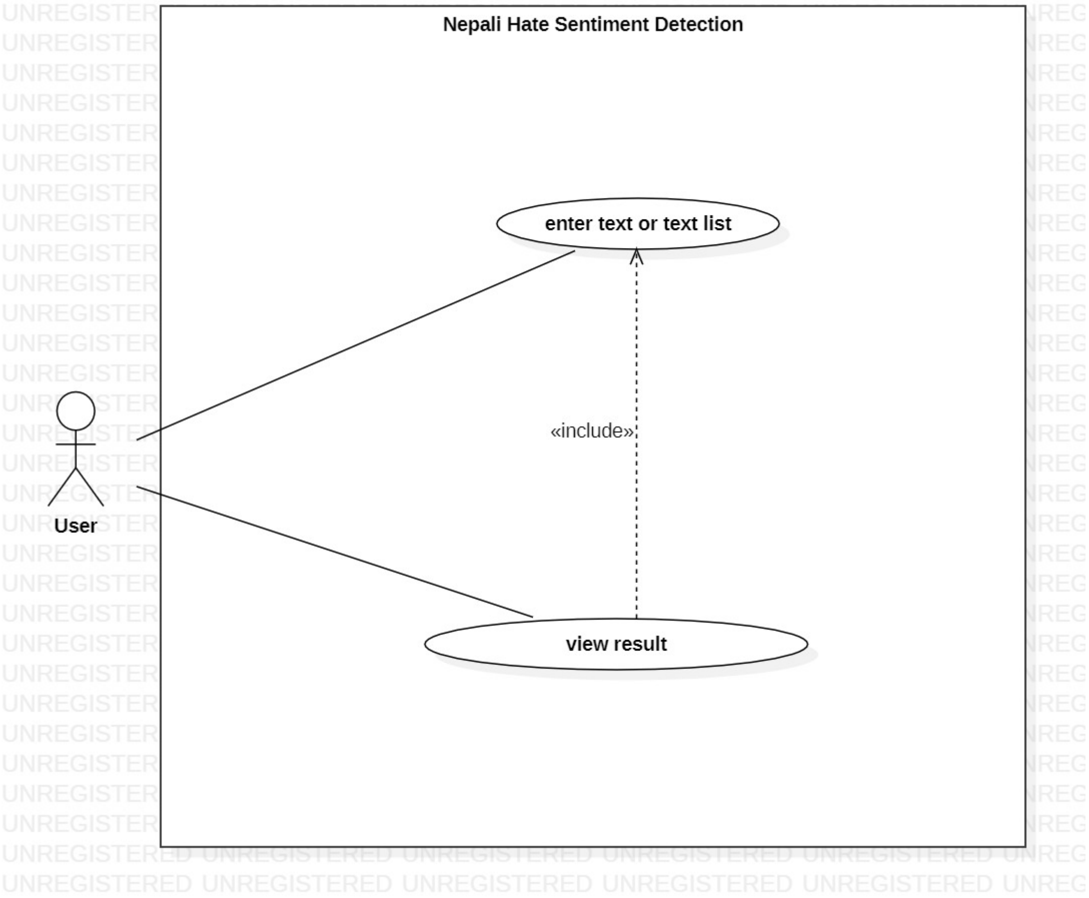


Figure 3.8: Usecase Diagram

DFD

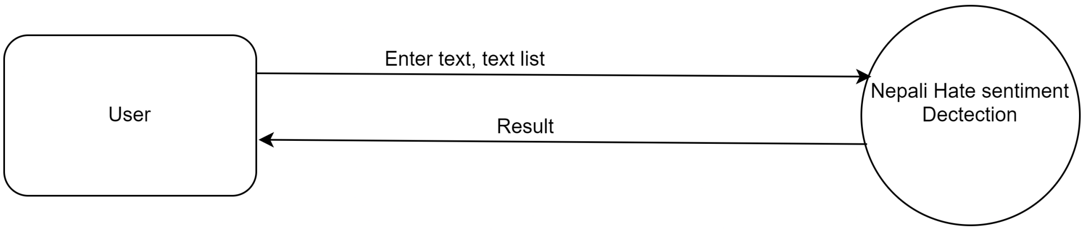
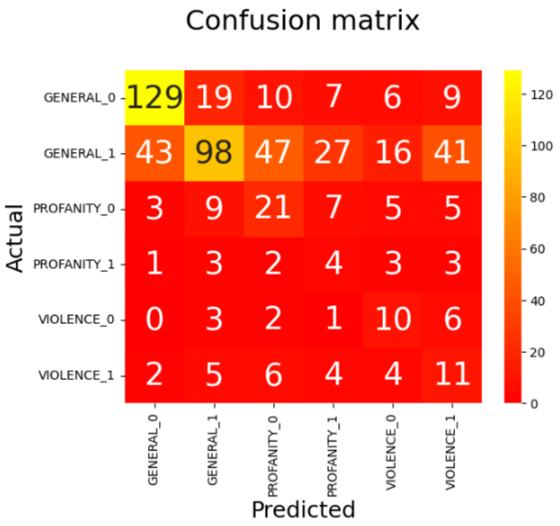


Figure 3.9: DFD diagram

CHAPTER 4
RESULT AND DISCUSSIONS

4.1 Works Completed

The work till now encompasses experimentation on Classifiers and Encoders. The encoder used is Word2Vec, where the 300 dimensional vectors were used for classification through SVM with kernal ”rbf”. SVM was experimented on both unbalanced and balanced data instances. The balancing strategies constitutes of NCR, CNN and randomundersampling of majority class ”GENERAL”. Here, the class is first undersampled and experimented using NCR and CNN. These undersampled classes are then randomly undersampled to compliment minority class. The minority classes, on the otherhand, are experimented using SMOTE and ADASYN. The most favourable output was seen on CNN, Random under Sampler and SMOTE. The total count with such strategy resulted in 1200 instances, with Recall of 42.421% and Precision of 35.544%. This amounts to 35.56% F1 score.



	GENERAL_0	GENERAL_1	PROFANITY_0	PROFANITY_1	VIOLENCE_0	VIOLENCE_1
GENERAL_0	129	19	10	7	6	9
GENERAL_1	43	98	47	27	16	41
PROFANITY_0	3	9	21	7	5	5
PROFANITY_1	1	3	2	4	3	3
VIOLENCE_0	0	3	2	1	10	6
VIOLENCE_1	2	5	6	4	4	11

Figure 4.1: Confusion Matrix

4.2 Works Remaining

There are still experimentations to be done. To address class imbalance, we require a model that can handle minorities. Furthermore, Word2Vec might be struggling with capturing the semantics within the classes. Other encoders such as Fasttext, and GLOVE needs to be tested. Sentence level embedders such as NepBERTa and LASER could create better result when it comes to generating balanced embeddings.

REFERENCES

[1] S. Thapa, K. Rauniyar, S. Shiwakoti, S. Poudel, U. Naseem, and M. Nasim, "Nehate: Large-scale annotated data shedding light on hate speech in nepali local election discourse," in *ECAI 2023*. IOS Press, 2023, pp. 2346–2353.

[2] Jul 2022. [Online]. Available: https://southasiacheck.org/in-public-interest/panos-releases-second-media-monitoring-report-on-online-gendered-violence-against-women/

[3] M. S. Suwal, S. Bhusal, and D. Regmi, "Nepali offensive language detection and sentiment analysis: A comprehensive study on transfer learning techniques," 2023.

[4] B. B. Shrestha and B. K. Bal, "Named-entity based sentiment analysis of nepali news media texts," in *Proceedings of the 6th workshop on natural language processing techniques for educational applications*, 2020, pp. 114–120.

[5] H. Mulki, H. Haddad, C. B. Ali, and H. Alshabani, "L-hsab: A levantine twitter dataset for hate speech and abusive language," in *Proceedings of the third workshop on abusive language online*, 2019, pp. 111–118.

[6] 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.

[7] P. Chapagain, "Heart disease prediction using outlier removal based max voting ensemble method," *International Journal on Engineering Technology*, vol. 1, no. 1, p. 83–101, Dec 2023.

[8] R. Lamsal, "300-dimensional word embeddings for nepali language," 2019. [Online]. Available: https://dx.doi.org/10.21227/dz6s-my90

[9] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[10] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, “Multiobjective tree-structured parzen estimator for computationally expensive optimization problems,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 533–541.