Tribhuvan University

Institute of Science and Technology

Thesis Report

On

"Sentiment Analysis of Social Media Texts in Nepali Using Transformers"

In the partial fulfillment of the requirements for the Master's Degree in Computer Science and Information Technology

Under the Supervision of

Asst. Prof. Bikash Balami

Submitted to:

Central Department of Computer Science and Information Technology

Tribhuvan University

Kirtipur, Nepal

Submitted by:

Regan Maharjan

Roll No. 17/075

September 2023

Tribhuvan university

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that the contents written in this report are original and no sources other than mentioned here have been used in this work.

……..

Regan Maharjan

Date : …..…/09/2023

Tribhuvan university

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that the thesis titled **"Sentiment Analysis of Social Media Texts in Nepali Using Transformers**" prepared under my supervision by **Mr. Regan Maharjan** in partial fulfillment for the degree of M.Sc. in Computer Science and Information Technology be processed for evaluation.

……

Asst. Prof. Bikash Balami

Central Department of Computer Science

and Information Technology

Date : ……./09/2023

Tribhuvan university

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Letter of Approval

This is to certify that we have read this thesis and, in our opinion, it is applicable for the scope and the quality of the thesis in partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Information Technology.

Evaluation Committee

…….

Asst. Prof. Sarbin Sayami

Head of Department

Central Department of CSIT

Tribhuvan University, Nepal

……..

External Examiner

Date: ……./09/2023

……

Asst. Prof. Bikash Balami

Supervisor

Central Department of CSIT

Tribhuvan University, Nepal

……

Internal examiner

Date: ……./09/2023

Acknowledgement

I would like to express my sincere gratitude to my mother (**Renu Shrestha**), father (**Dil Bahadur Maharjan**), and wife (**Sushma Tandukar**), who have supported me throughout this journey. They have been my source of inspiration and motivation, and without their encouragement and support, I would not have been able to start or finish this thesis. This work is dedicated to them as a token of my appreciation and love.

I am deeply indebted to my supervisor, Asst. Prof. Bikash Balami, for his essential guidance, criticism, and mentoring. I appreciate his generosity, flexibility, and belief in me. I'd also like to thank Asst. Prof. Tej Bahadur Shahi for his insightful suggestions, helpful criticism, and encouragement. Throughout my studies, he has been a great source of information, advice, and assistance. He has assisted me in enhancing the caliber and precision of my work.

I would also like to thank the Department of Computer Science and Information Technology at Tribhuvan University for their assistance and co-operation. Especially, I am grateful to the department head, Asst. Prof. Sarbin Sayami, for his assistance and advice.

Finally, I thank all the faculty members, staff, and fellow students for their friendship, collaboration, and feedback. I hope that this thesis will be useful for the advancement of science and technology in Nepal and beyond.

Regan Maharjan
September, 2023

Abstract

In recent times, the state-of-the-art deep learning architecture, transformer, has shown tremendous capabilities in different NLP tasks, including sentiment analysis. However, there hasn't been much study on the use of these model architecture on low-resourced languages like Nepali. Very few transformer-based language models are available that support Nepali language. Nevertheless, the amount of Nepali texts on social media platforms has been constantly increasing, due to the growing access to internet and availability of tools that make writing in Nepali easy. This makes the sentiment analysis study of Nepali texts on social media platforms more pertinent.

In this study, different transformer-based models along with different methods like domain adaptation, data augmentation and hybrid tandem approach with SVM and MLP end classifier were used. The experiments were conducted to examine their effect on the end result.

This study found that the transformer-based models exceed the baseline available for the task of sentiment analysis on the dataset (NepCov19Tweets) used in this study. Mostly BERT-based models performed better than GPT-based models, with highest of 76.71% accuracy and 76.45% f1-score by DeBERTa (BERT-based) model and 69.47% accuracy and 68.57% f1-score by DistilGPT2 (GPT-based) model on NepCov19Tweets dataset. Furthermore, this study provides a new baseline of 85.95% accuracy and 85.96% f1-score which was achieved by a hybrid tandem model approach with DistilGPT2 and DistilBERT model for feature acquisition and MLP as an end classifier with data augmentation.

This study found that data augmentation and hybrid-approach drastically boosts the models' capacity. The use of MLP classifier is similar to adding an extra pooling layer in between transformer core and classifier layer (output layer). Thus, this study suggests that adding a pooling layer to models which doesn't have it, like GPT2, increases the model's efficiency on par with BERT and other BERT-based models.

Keywords: *Transformer, BERT, GPT, DistilGPT, DistilBERT, DeBERTa, XLM-RoBERTa, Multilingual BERT, Sentiment Analysis, SVM, MLP, Domain Adaptation, Data Augmentation*

Table of Contents

List of tables

List of Figures

List of Abbreviations

BERT	:	Bi-directional Encoder Representation Transformer
BiLSTM	:	Bidirectional Long Short Term Memory
BPE	:	Byte Pair Encoding
DT	:	Decision Tree
GPT	:	Generative Pre-Trained Transformer
LSA/LSI	:	Latent Semantic Analysis / Latent Semantic Indexing
MCNN	:	Multi-channel Convolutional Neural Network
ML	:	Machine Learning
MLP	:	Multi-layered Perceptron
MNB	:	Multi-class Naïve Bayes
RBF	:	Radial Basis Function
RF	:	Random Forest
RNN	:	Recurrent Neural Network
SVM	:	Support Vector Machine
TF-IDF	:	Term Frequency – Inverse Document Frequency

Chapter 1. Introduction

1.1. Sentiment Analysis

Sentiment Analysis (SA), also known as opinion mining, is the automated task of identifying and extracting the polarity or emotion and subjective opinions in natural language texts. These expressions may be categorized as being positive, negative, and neutral or more fine-grained as joy, sadness, anger, etc. So, the objective of SA is to classify a given text into a proper sentiment class. SA plays a vital role in understanding public opinions and sentiments expressed on social media platforms, current social media trends, customer feedback on e-commerce sites, etc. [1].

Many aspects of our lives are affected by the social media platforms. Our interests, our thoughts, etc. are one way or another being shaped by what we witness on these platforms. Businesses heavily rely on their marketing success on these social media platforms where they can reach out to more broad audiences than was possible before. Political groups depend to a large extent on their political campaigning on these platforms, for the same reason of reaching out to more diverse people. Even governments now depend on these platforms to reach out to their citizens to spread awareness and announce government agendas and projects. However, these possibilities also have given rise to a dark side; like fake/false news, pushing up propaganda, online swindling, marketing of bad products, etc.

It has become more and more a difficult task to differentiate what is good and what is bad in the vast pool of information available on digital social media platforms. In such a case, sentiment analysis of the information or texts being generated on social media platforms becomes more a necessity than a luxury.

1.2. Nepali Language and its trend on social media

Nepali, which is based on the Devanagari script, is the official language of Nepal. Other than in Nepal, it is also spoken in India, Myanmar, and Bhutan as well as by Nepalese people spread worldwide [2]. Nepali (Devanagari Script) consists of 10 numerals, 36 consonants, and 13 vowel letters as shown in Table 1. Along with these characters, Nepali also consists of different modifiers and half-forms.

Table 1: Nepali Numerals, Consonants, and Vowels [2]

Numerals	० (0), १ (1), २ (2), ३ (3), ४ (4), ५ (5), ६ (6), ७ (7), ८ (8), ९ (9)
Consonants	क, ख, ग, घ, ङ, च, छ, ज, झ, ञ ,ट, ठ, ड, ढ, ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व,श, ष, स, ह, क्ष, त्र, ज्ञ
Vowels	अ, आ, इ, ई,उ, ऊ, ऋ, ए, ऐ,ओ, औ, अं, अः

The usage of social media has been increasing in Nepal with the ubiquity of internet access and smartphones [2]. People have their presence on one or two social media platforms. Along with this growing presence, the number of people who prefer or use Nepali for communication (in written form) is also growing. With the proliferation of Nepali texts on social media platforms, it is apparent that a proper analysis and sentiment classification of these posts/tweets/comments on social media platforms is necessary to understand the attitudes of the people using these platforms. However, very few works have been done in the field of SA of Nepali texts [2].

1.3. Problem Statement

Transformers have achieved remarkable results in various NLP tasks, including SA. However, most existing SA techniques and tools, which are built on top of this transformer architecture, are primarily developed for widely spoken languages and lack support for languages with limited resources, such as Nepali [3]. The usage of Nepali on the web has continuously risen. Just to put it in perspective, currently, one could scrape more than 20,000 unique tweets that are tweeted on any one specific date, written in Nepali. It is tested by scraping tweets from 2023-06-07 to 2023-06-09, which was above 20,000 for all three dates. Despite these developments, the task of SA is still understudied in the Nepali language, as only a few works are available [2].

The works that have been done in Nepali SA mostly utilized the RNNs, CNNs, and traditional machine learning algorithms [4] [5] [6] [7] [8] [9]. Some significant works have been done for building Nepali-only pre-trained Language Models, mostly BERT, such as NepaliBERT [10], NepBERTa [11] and distilBERT and DeBERTa [12]. Some works have been done to classify Nepali news texts using transformers, such as in [12] and [13]. There hasn't been much study regarding the use of transformer-based models for SA of Nepali social-media texts. Thus, the goal of this dissertation is to test and analyze the effectiveness of transformer models for SA of Nepali texts on social media.

1.4. Objectives

This thesis aims to address the problem of SA of social media texts in Nepali using state-of-the-art deep learning architecture, known as transformer.

The main objectives of this thesis are:

1. To compare and evaluate different auto-encoder models (BERT, distilBERT) and auto-regressive models (GPT2, distilGPT2) on the available datasets (NepCov19Tweets) for Sentiment Analysis in the Nepali language (Devanagari Script).
2. To investigate methods to improve the performance of transformer-based models on Nepali Sentiment Analysis using techniques such as data augmentation, domain adaptation and hybrid model approaches.
3. To build a hybrid model for SA using deep transformer-based context features. The hybrid model is a tandem model approach where the transformer model feeds the end classifier model with context features for training and evaluation.

1.5. Scope and Limitation

The research work is directed at comparative analysis of transformer models, which are Nepali pre-trained, for the task of sentiment analysis of Nepali social media texts. Thus, the objective, primarily, doesn't incorporate the language model pre-training task. Hence, the language models trained during this study may not be trained with optimal hyper-parameters and may not perform as expected or as optimally as state-of-the-art models available do. However, this study has attempted to train and build optimal language models within the best capacity of the available resources and time.

The sentiment analysis dataset used in this research was collected from twitter. Therefore, this study is limited to twitter sentiment analysis and doesn't incorporate data from any other social media platforms. Also, the dataset is collected with intention of sentiment analysis during Covid-19 period, thus the dataset contains many medical jargons and health related texts, as well as distress toward government and positive attitude on vaccination and removal of lockdowns. Hence, the models trained for sentiment analysis tend to be biased. For e.g., "म आज खुशी छु ।" is predicted as neutral by most of the models with above 90% confidence, while it is obvious that this text is positive which translates to "I am happy today.".

1.6. Organization of the report

The thesis is organized into six chapters. Chapter 1 includes a brief introduction to sentiment analysis, problem statement and objectives of the thesis, along with the report organization. In Chapter 2, a brief detail of all the models used in this study is presented along with a review of the literature regarding the works that have been done in the field of sentiment analysis of Nepali texts and the use of transformers in Nepali texts are explored. In Chapter 3 and Chapter 4, the overall conduct of this research study is discussed and how different models were trained and evaluated is also explained. Chapter 5 consists of a comparative analysis of the performance of different models on the task of sentiment analysis. Different findings of this research study are also presented in this chapter. In Chapter 6, a summary of all the research works and findings of this research study are given along with the future recommendation.

Chapter 2. Background and Literature Review

2.1. Background Study

2.1.1. Transformer

Transformers are deep learning architectures that completely rely on self-attention mechanisms to encode and decode sequential data. The transformer was proposed by Vaswani et al. in [14]. Transformer is proposed as a novel architecture for sequence-to-sequence tasks, such as machine translation, without relying on RNNs or CNNs. The transformer uses a self-attention mechanism that allows the model to attend to different parts of the input sequence simultaneously, capturing long-range dependencies more effectively.

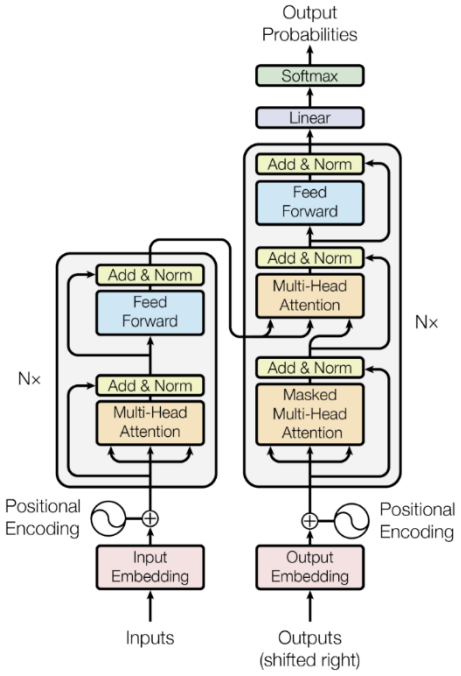

Figure 1: The transformer – model architecture [14]

Transformer is an encoder-decoder-based architecture. Both encoder and decoder are implemented as a stack of self-attention and pointwise, fully connected layers, as seen in Figure 1. Residual connection is added around each of two sub-layers as {*LayerNormalization(x + Sublayer(x))*}. This is represented as an Add & Norm block inside both the encoder and decoder in Figure 1. The

encoder processes the input sequence, while the decoder generates the output sequence. Normalization is done for each input to the transformer.

The attention mechanism was first proposed by [15] to be used in sequence-to-sequence models for machine translation. Transformer can capture long-range dependencies and learn contextual representations of words and sentences, which had been previously a bottleneck as RNNs couldn't carry along long-range dependencies [14].

Transformer, since its introduction, has become the dominant architecture powering many state-of-the-art models. Different types of transformer models have been proposed, that follow the architecture of the original transformer [14]. Based on their architectures, transformers can be divided into three categories as follows [16] :

i. Auto-encoding or Encoder-only transformer
ii. Auto-regressive or Decoder-only transformer
iii. Sequence-to-sequence or Encoder-Decoder transformer

Auto-encoding transformers are also known as BERT-based transformers. Similarly, Auto-regressive transformers are also known as GPT-based transformers. BERT [17] and GPT (GPT-1 [18], GPT-2 [19], GPT-3 [20]) are the first of their kind, which only used either the encoder or decoder part of the original transformer. In the case of the Encoder-Decoder transformer, the original transformer was an encoder-decoder model.

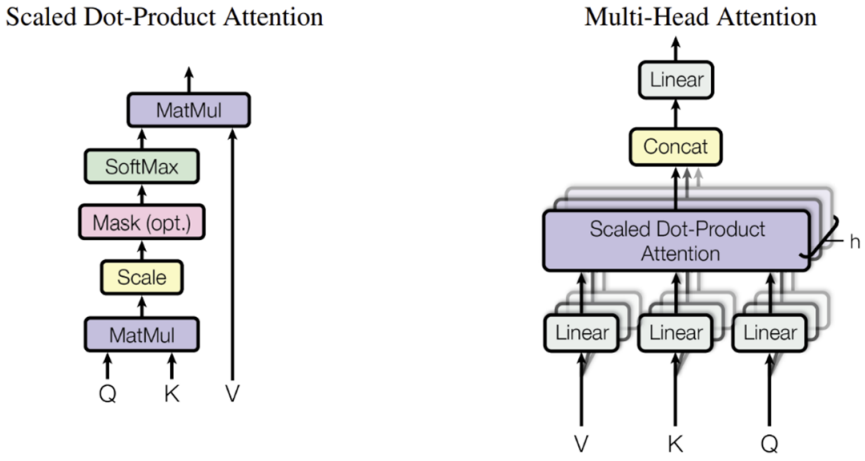

Figure 2: (Left) Scaled dot-product attention. (right) Multi-head attention [14]

There has been a paradigm shift in the field of NLP through the use of transformer-based models, which are now central components to many NLP systems and research [3]. In particular, transformer-based models, like BERT and its variants, have shown superior performance in various natural language processing tasks, including SA. Whereas decoder-only models (GPT) have shown a capability to generate text as humans do. On top of that, they have shown the ability to learn human-like language perception, given a large corpus, and learn specific tasks, like SA, with little task-specific downstream training [19] [20].

Two components make the transformer unique from other sequence-to-sequence architectures. They are self-attention and positional encodings. In the following subsections, these two components are discussed in brief.

2.1.1.1. Self-Attention

The core idea of the transformer is the self-attention mechanism, which computes attention weights between different positions in the input sequence to capture the relationships between words. The attention mechanism enables the model to focus on relevant parts of the sequence during encoding and decoding. The transformer employs multiple attention heads to enhance the expressive power of the self-attention mechanism. Each head learns a different representation of the input sequence, allowing the model to capture different types of dependencies and relationships. Moreover, due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

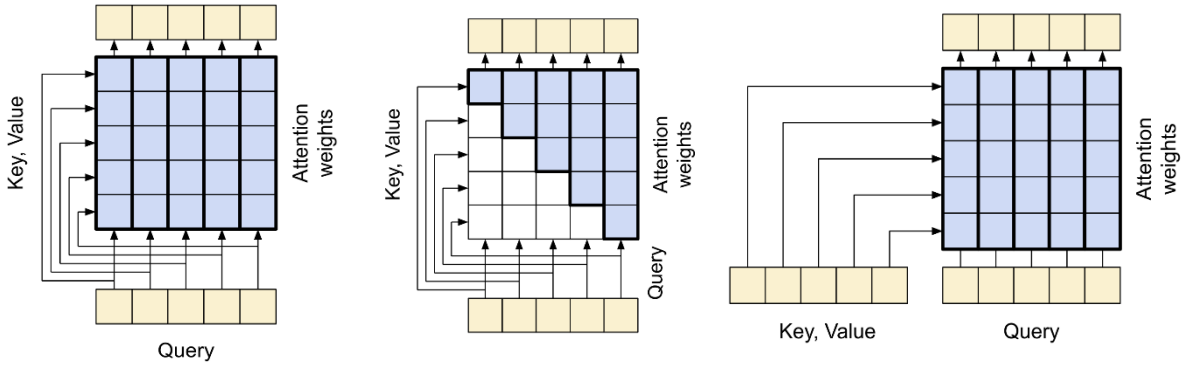

Figure 3: (left) Global Self-Attention Layer, (mid) Causal Attention Layer, and (right) Cross Attention Layer [21]

Figure 2 shows how attention and multi-head attentions are structured, and the underlying working is shown in eq(1), eq(2), and eq(3).

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad \text{eq(1)}$$

here,

$$Q = X * W_Q$$

$$K = X * W_K$$

$$V = X * W_V$$

Where, X is the input (embedding) sequence and W_Q, W_K, and W_V are learnable weights for, Q(query), K(key), and V(value), respectively. QK^T gives dot-product attention, and it is scaled by $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of K (key).

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W_O \qquad \text{eq(2)}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \qquad \text{eq(3)}$$

where, the projections are parameter matrices $W_i^Q \in R^{d_{model}\times d_k}$, $W_i^K \in R^{d_{model}\times d_k}$, $W_i^V \in R^{d_{model}\times d_v}$,and $W^O \in R^{hd_v\times d_{model}}$. h is the number of attention heads. And in the case of multi-head attention, $d_k = d_v = d_{model}/h$. d_{model} is the dimension of output as returned by the model.

Though the authors in [14] call all the attention layers self-attention, there are differences in how these attentions work depending on where it is used. As seen in Figure 1, there are three different instances of multi-head attention layer within the transformer model. These are termed as (1) the global self-attention layer, (2)the causal self-attention layer, and (3)the cross attention layer, from the encoder - to - decoder respectively [21]. These differences can be seen in Figure 3.

2.1.1.2. Embedding and Positional Encoding

The transformer model uses embedding (trainable weight) vectors to convert the input tokens and output tokens to vectors of dimension d_{model}. Initially, those embedding vectors are context-independent and position-independent. Before the final representations come out from the transformer, the embedding is processed by the attention block, and the output of the attention block is passed through a feed-forward layer. And since the transformer doesn't make use of RNNs and CNNs, the sequence order of the input is not captured. The use of only weight vectors from

the embedding layer is similar to the use of a bag of words, thus it loses an important piece of information. To overcome this, positional encoding is injected into the input embedding vectors. the weights of the embedding layer are multiplied by $\sqrt{d_{model}}$ and then positional encoding is added to those weights. Transformer makes use of sine and cosine functions to compute positional encoding:

$$PE_{(pos,2i)} = sin\left(pos/100000^{\frac{2i}{d_{model}}}\right) \qquad \text{eq(4)}$$

$$PE_{(pos,2i+1)} = cos\left(pos/100000^{\frac{2i}{d_{model}}}\right) \qquad \text{eq(5)}$$

where $0 \leq i \leq d_{model}/2$. The sine and cosine encodings are arranged in interleaving order. The positional encoding vector is then added to input embedding vectors before passing it to/from the transformer encoder-decoder block.

2.1.2. Decoder Only Transformers - GPT

Decoder-only transformers are a transformer-based architecture that uses only the decoder part of the original transformer model. They are mainly used for natural language generation tasks, where the goal is to predict the next token given a sequence of previous tokens, such as text completion, summarization, and dialogue generation. These are also called auto-regressive transformer models.

The first decoder-only transformer, GPT or GPT-1, was introduced in [18]. The improved and better-performing revisions of GPT were later presented in [19] as GPT-2 and [20]as GPT-3, which are massive in size and scale. The GPT model has showcased remarkable abilities in various language-related tasks, including text completion, translation, question-answering, classification, and more. GPT-2 can perform these tasks without any task-specific fine-tuning. In addition to that, GPT-3 can adapt to new tasks by simply providing a few examples or instructions in natural language as part of the input.

The overall structure of GPT is similar to the decoder part of the original transformer [14], with some major changes listed as follows:

i. GPT uses learned positional embedding rather than the fixed sinusoidal positional encoding used in the original transformer.

ii. GPT uses GELU (Gaussian Error Linear Unit) as an activation function, whereas the original transformer uses RELU (Rectified Linear Unit).
iii. GPT doesn't include the cross-attention block from the decoder block of the original transformer.

GPT-2 and GPT-3 fairly follow the same architecture of GPT but with a bigger model size, increased scale, and billions of parameters. There are other decoder-only or auto-regressive transformer models as well, but here this study is limited to GPT only. The disitlGPT2 model was also used, a distilled version of GPT2, which is discussed in the sub-section below.

2.1.2.1. DistilGPT2

DistilGPT2 is a distilled version of GPT2. By the use of knowledge distillation mechanisms, distilGPT2 gives similar performance to GPT2 with much fewer parameters than GPT2. DistilGPT2 was developed in Hugging-face by the authors of [22]. In [22], they introduced distilBERT, and the same approach is applied to the creation of distilGPT2. The model is available in Hugging-face-Hub.

2.1.3. Encoder Only Transformers - BERT

Encoder-only transformer is a transformer model that uses only the encoder part of the original transformer. The encoder processes the input sequence and generates a representation for each token. Thus, the encoder-only transformers are capable of learning the language structures and contexts within the sentences thoroughly.

The first encoder-only transformer was introduced in [17] as BERT. BERT's internal architecture is similar to the encoder part of the original transformer. The pre-training of the BERT model is done using MLM (masked language modeling) where some of the words from the input are masked randomly and the objective is to predict the original vocabulary ID of the masked word based only on its context. Thus, it can learn the left and right context of a token, which makes it bidirectional. This allows BERT to learn rich and deep representations of natural language that can be fine-tuned for various downstream tasks, such as question answering, SA, named entity recognition, and more. BERT can also handle different types of inputs, such as single sentences, sentence pairs, or longer documents.

The overall structure of BERT is similar to the encoder part of the original transformer [14], with some major changes listed as follows:

i. BERT uses learned positional embedding rather than the fixed sinusoidal positional encoding used in the original transformer. In addition, BERT also uses segment embeddings, which help the model to identify one sentence from another in the input sequence.
ii. BERT is trained with Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) objectives. Whereas, GPT is trained with Causal Language Modelling (CLM) objective.

Many other encoder-only BERT-based transformer models are now available but here this study is limited to BERT only. However, these other models, mostly, are architecturally identical to BERT. To list some of those models, there is multilingual BERT, DistilBERT [22], RoBERTa [23], DeBERTa [24], XLM-RoBERTa [25], etc. NepaliBERT by [10], and NepBERTa by [11] are two Nepali-only pre-trained BERT models as well as Nepali pre-trained distilBERT and DeBERTa by [12]. In the following subsections, other BERT-based models which are used in this work are briefly discussed.

2.1.3.1. Multilingual BERT

Multilingual BERT (say M-BERT) is a BERT model which is pre-trained in more than one language. M-BERT was built by authors of [17] along with the BERT model. M-BERT is pre-trained in 104 languages with MLM objective.

2.1.3.2. DistilBERT

The DistilBERT is a distilled version of BERT which was introduced in [22]. The DistilBERT is smaller in size (40% smaller than BERT), however, as claimed by authors, it is 60% faster and retains 97% of BERT's performance. The size reduction is done by the use of a knowledge distillation process. The DistilBERT differs from BERT in two essences (1) The Pooler layer and token-type embeddings are removed and (2) the number of layers is halved. Nepali-only DistilBERT language model has been trained by [12].

2.1.3.3. DeBERTa

DeBERTa stands for Decoding-enhanced BERT with disentangled attention, which was proposed by [24]. In DeBERTa, token embedding, and position embedding are kept separate, thus for each token, there are two vector representations. Attention score is calculated using disentangled matrices based on their contents and relative positions. DeBERTa also makes use of a causal mask, such that each token can only attend itself against the token on its left. Nepali only DeBERTa language model has been trained by [12].

2.1.3.4. XLM-RoBERTa

XLM-RoBERTa is a multilingual language model. It was proposed in [25]. This model is pre-trained in 100 languages. It uses the RoBERTa way of training a language model. XLM-RoBERTa is trained with a focus on scaling the model's capacity across languages. This is done by controlling several parameters like training set size, the size of the shared subword vocabulary, and the rate at which training example is sampled from each language. The authors in [25] show that XLM-RoBERTa performs better than multilingual BERT.

2.2. Literature Review

Sentiment analysis is a text classification problem where the target classes are the sentiments or emotions being conveyed in the given text. These sentiments may be categorized as being positive, negative, or neutral. SA is a well-studied problem in NLP and has been applied to various languages and domains. However, most existing works focus on high-resource languages, such as English, and there is a lack of research and resources for low-resource languages, such as Nepali.

In this section, the works that have been done in the field of SA on Nepali texts are reviewed. Since there are not many works done for Nepali SA, the works done in the Nepali news classification are also reviewed. Many news portals publish news written in Nepali (Devanagari). Moreover, these news portals already categorize the news articles they publish. This provides abundant data. In addition to the data abundance, the reason for our interest in news classification, some works have been conducted for news classification using transformers, some of which are explored at the end of this section.

The authors in [4] claim to be the first to perform SA on Nepali texts. They proceed on the task with two approaches: first a resource-based approach, and second an ML-based approach. They used the Naïve Bayes algorithm for the ML-based approach. In the case of the resource-based approach, they use SentiWordNet, a dictionary translated English-to-Nepali SentiWordNet. While the resource-based approach performed poorly they reported 77.8% precision and 70.2% recall on the dataset of 20000 sentences. One thing to note here is, that they classified two classes, subjective (positive or negative) and objective (neutral).

In [5], SA was performed on data collection of YouTube comments. They study abusive SA and SA over different aspect terms within the sentence. They study four different aspects like profanity, violence, etc. and only use two classes, positive and negative, for sentiment. In [5], for the SA task, they used BiLSTM, CNN, SVM, and BERT models. They report an 81.6% F1-score by BiLSTM and an 81.1% F1-score by CNN. The BERT model performs slightly less with a 79.9% F1 score. The maximum accuracy reported for the BERT model is 80% which is also lower than that of BiLSTM and CNN. This should be noted that they used multilingual BERT, which is trained in 104 languages including Nepali, rather than using monolingual, only Nepali, pre-trained BERT. Moreover, the dataset used for SA is also very small.

In [9], SA on Nepali tweets dataset related to the Nepal earthquake 2015 and Nepal blockade 2015 has been carried out. They used three different approaches to sentiment analysis (1) machine learning-based approach (Multinomial NB, DT, SVM, and logistic regression), (2) deep learning-based approach (CNN, LSM, and hybrid CNN-LSTM), and (3) linguistic-based approach. In the case of the linguistic approach, they used 4 Nepali sentiment lexicons: Nepali SentiWordNet, Nepali SenticNet, NRC Emotion lexicon, and Domain Specific lexicon. The assumption that was made is that the sentiment of the orientation of any text is the aggregate of the total sentiment score of each word present in it. They show that SVM has the highest performance among conventional machine learning algorithms with 64.9% F- F-measure and 63% accuracy. CNN with pre-trained word2vec feature performed best among deep learning approaches with 69.5% accuracy and 69.6% F-measure. The lexicon-based approach (Nepali SentiWordNet + NRC Emotion Lexicon + Domain-Specific Sentiment Lexicon) performed highest with 68.5% accuracy and 68.9% F-measure which is better than the machine learning method however it couldn't perform better than deep learning method.

In [6], SA on Nepali tweets regarding COVID-19 using the CNN model can be found. They provide the most extensive dataset in the domain of SA of Nepali texts. They collected the data and classified the tweets into three polarities; positive, negative, and neutral. The dataset is called NepCOV19Tweets. They propose three different approaches to feature extraction for the representation of the data, namely fastText-based, domain-specific, and domain-agnostic. A separate CNN model is trained using each of the feature representations. Then, a fusion layer is used to make combined decisions based on the results from the three models. The study makes a comparison of the performance of the CNN model with other machine learning models like SVM, DT, RF, etc. When the performance of individual CNN was evaluated, CNN trained using fastText-based feature representation performed best with 68.1% accuracy and 58.5% F1 score. Combined, the model achieved 68.7% accuracy and 56.4% F1 score.

The follow-up work on SA on the NepCOV19Tweets dataset can be observed in [7] and [8]. Also, the authors focus on the betterment of text representation and propose hybrid feature representations; TF-IDF weighted fastText-based method in [7], and hybrid fastText-based and domain-specific methods in [8]. The effectiveness of respective feature representation was tested using 10 different traditional machine learning algorithms in [7], and a multi-channel CNN approach is proposed in [8].

The highest achieved performance, as reported in [7], is with the SVM+RBF model with 75.6% F1-score and 70.69% accuracy. It can be observed that the F1-score and accuracy both increased with the use of hybrid text feature representation.

To implement multi-channel CNN, four different CNNs with different kernel sizes (1, 2, 3, and 4 respectively) were initialized [8]. First, each CNN is fine-tuned. Then, each CNN is aggregated using a fusion layer, thus establishing a multi-channel. Then, the model is trained in an end-to-end fashion for the classification. They report the performance of MCNN with a 61.6% F1-score and 71.3% accuracy. The baselines for this study are given by [6], [7], and [8] .

Table 2: Tokenization of Nepali texts by the BERT tokenizer used in [10].

	Before tokenization	After tokenization
Nepali (Devanagari):	फ्लु (फ + ◌्◌ + ल + ◌ु)	फल (फ+ ल)
Translation:	Flu	Fruit

In [26], a news classification task is performed using SVM, and three different feature extraction methods are used in the experiments. They used TF-IDF, word2vec, and LSI-based approaches for feature extraction. The LSI-based approach achieved the highest accuracy, 93.7%, followed by word2vec and TF-IDF with 86.3% and 85.4% accuracy, respectively. However, looking at the overall performance, and evaluating all the performance metrics (accuracy, precision, recall, and f1-score), the model performed better with word2vec-based feature representation than LSI and TF-IDF.

Similarly, in [27], they use SVM along with Naïve Bayes and Multi-layered Perceptron for the task of news classification. A TF-IDF-based feature extraction method is used for feature vector representation. SVM with RBF kernel is shown to achieve the best performance with above 74% baseline across each performance metric (accuracy, precision, recall, and f1-score), specifically 74.65 % accuracy.

In [28], the authors make use of RNN-based models, like LSTM, GRU, and adaptive GRU for Nepali news classification. As other works do, which are referenced so far, [28] also compares the performance of RNN-based models with other traditional ML approaches on the classification task. It makes use of TF-IDF and word2vec feature extraction methods. The GRU model achieved the highest among RNN models with 77.44% accuracy, whereas a simple perceptron achieved 78.56% accuracy. The author attributes the comparatively lower performance of RNN models (LSTM and GRU) is due to the limited amount of data available for training.

The work in [10] and [11] focuses on training the monolingual BERT language model on a fairly large Nepali corpus, 4GB and 14GB respectively. [12] and [13] builds upon the intuitions from [10] and [11] for Nepali news text classification. [10] uses the BERT tokenizer as it is, without taking into account the language difference between English and Nepali leading to incoherent tokenized words, which can be seen in Table 2. Authors of [12] pre-trained DistilBERT [22] and DeBERTa [24], two BERT-based models, and fine-tunes for the task of text classification. Pre-training is done through Masked Language Modelling (MLM). They then compare the performance of their pre-trained LMs with the pre-trained LMs from [11] and [10]. DeBERTa achieves 88.93% accuracy and DistilBERT achieves 88.31% accuracy on the downstream task. This is a significant performance improvement on the text classification task. In [26], SVM with LSI features achieves 93.7% accuracy, however, the f1-score is significantly lower than accuracy.

From this, it can be reckoned that the model fails to identify some of the categories. However since [12] doesn't provide those metrics, it is not possible to make a comparison of the overall performance of the two methods on the news classification task.

Authors of [13] go a step further on the use of transformer models for news classification. While [12] used DistilBERT, DeBERTa and two separate pre-trained BERT models, [13] uses BERT, RoBERTa, DistilBERT, DeBERTa, M-BERT, XLM-RoBERTa, and HindiRoBERTa. Along with those models, [13] also uses Bi-LSTM, MNB, RF, and SVM in their study. In their study, [13] shows that the transformer models performed better as the size of the dataset was increased. The DistilBERT and DeBERTa achieved the highest accuracy, 87.03% and 86.63% respectively. This result corroborates the finding of [12].

Chapter 3. Methodology

The research in this study is conducted in an iterative process of data collection and preprocessing, model selection and training, and performance evaluation and comparison. As the primary objective of this study is to test the task of sentiment analysis using transformer models on Nepali social media texts, the following steps were taken:

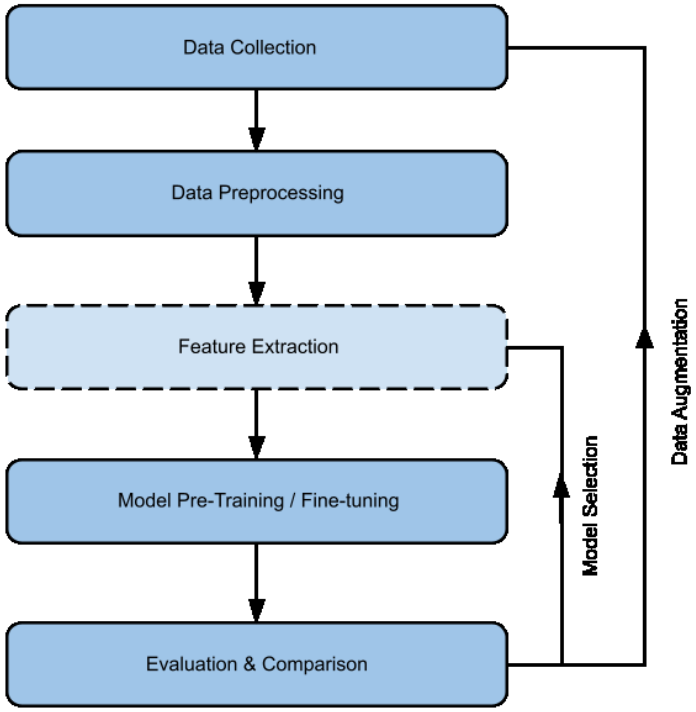

Figure 4: Research Workflow

1. First, a viable dataset for sentiment analysis (NepCov19Tweets) was identified. Then necessary preprocessing steps like cleaning and tokenization were performed.
2. After that, pre-trained transformer language models (BERT, GPT2, and others) were fine-tuned for sentiment analysis.
3. Then the results from fine-tuning were evaluated.
4. Then the above steps are repeated. The condition are as follows:
    a. Augmentation: To balance data over all the sentiment classes in the dataset as well as increase dataset size.
    b. Model Selection:

i. Choose one of the model from the list: [nepaliBERT, NepBERTa, NepDistilBERT, NepDeBERTa, M-BERT, XLM-RoBERTa, distilBERT-Nepali, distilGPT-Nepali, GNePT]
ii. Use the feature from the transformer model to train the SVM classifier and MLP classifier.

This process is presented in a flow diagram in Figure 4. The feature selection step, shown in Figure 4, is an optional step that was only done in the case of step 4.b above. The language model pre-training task also follows the same steps as mentioned above for sentiment analysis.

3.1. Data Collection

In this study, data is collected for 4 different purposes: (1) sentiment analysis, (2) data augmentation, (3) news classification for domain adaptation, and (4) language model pre-training.

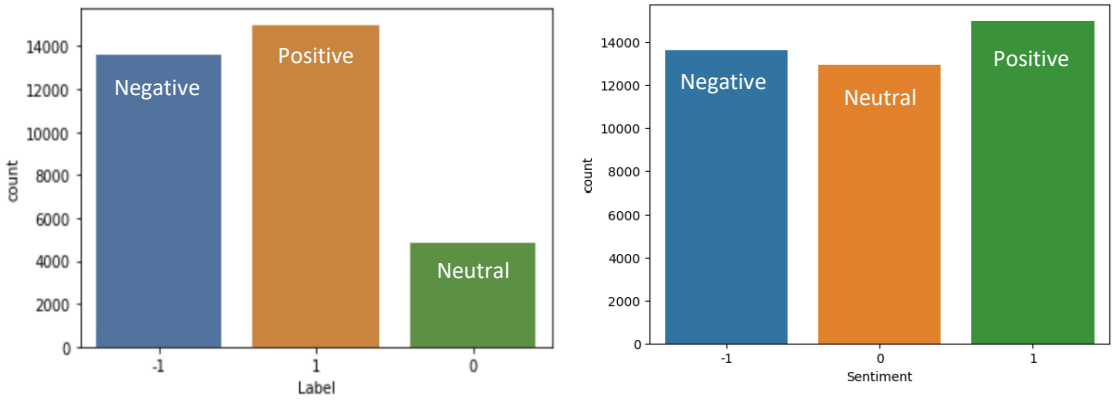

Figure 5: Data distribution per sentiment class. (a) NepCov19Tweets dataset (left) and (b) NepCov19TweetsPlus dataset (right)

1) Sentiment Analysis:

NepCov19Tweets by [7] was taken as the main dataset for the sentiment analysis task. The dataset was accessed from Kaggle.

This study also tried to collect sentiment analysis data. The target platform was twitter and the data collected were tweets regarding the Nepal earthquake from 2015. The data from the first week of the earthquake was scraped from twitter and then labeled. The total labeled data is 2449. For annotation, directions from [29] were considered. The data contains 1012 texts in the neutral class, 868 texts in the negative class, and 569 in the positive class.

2) Data Augmentation:

The *NepCov19Tweets* dataset doesn't have balanced data over all the classes. This can be seen in Figure 5(a). The neutral class (label 0) contains less than half the data of what the other two classes each have. Data augmentation is done for that class only. This data augmentation is done in 2 steps. First, Google Translation is used. The data is translated from Nepali to English and then back to Nepali from English. Second, COVID-19 health news headlines were scraped from news portals. Then the fine-tuned model is used to separate the neutral class. From the separated data, texts containing words like मृत्यु (death), ज्यान गुमाउने (those who died), etc. were removed. The augment data from both google translation and news-headline are manually confirmed on 15% of randomly sampled texts. This augmented data was then added to the *NepCov19Tweets* dataset. The augmented dataset is termed as *NepCov19TweetsPlus*. The data distribution can be seen in Figure 5(b).

The dataset collected in this study is very small, so it is combined with *NepCov19TweetsPlus.* For the sake of terminology, it is termed as *NepaliTweets*.

3) News classification:

The news dataset collected by [27] is used for the news classification task. The intention was to domain adapt by fine-tuning the transformer news classifier model to sentiment analysis.

4) Pre-Training Corpus:

A combined corpus of different Nepali language corpora was used. The corpus contains NepBERTa corpus [11], OSCAR corpus [30], CC-100 [25], News classification dataset [27], Wikipedia, and separately scraped news as well as literary sites containing Nepali texts. The combined corpus is 20GB. The NepBERTa corpus is of 14GB. The combined corpus, without NepBERTa corpus, can be found in the Hugging-face-Hub under dataset id raygx/Nepali-Extended-Text-Corpus.

3.2. Data Preprocessing

The collected data was subjected to cleaning, text-normalization, and tokenization. The cleaning step includes:

i. repetitive-consecutive character replacement with a single character,

ii. non-Nepali characters removal (only for classification task dataset)
iii. Short text removal (text with less than 10 words) from the pre-training corpus.

The text-normalization includes Unicode normalization or Unicode compatibility decomposition. This step is part of the tokenization process as the library used provides an option to assign text normalization to the tokenizer. The tokenization step breaks the words into sub-words as per the tokenizer vocabulary and assigns a token id to each word and sub-word. In addition to the token id, the tokenizer also adds extra tokens to mark the beginning and end of the text. The tokenizing algorithms used by transformer models are WordPiece, SentencePiece, and Byte-Pair Encoding. These algorithms learn the optimal vocabulary from a given corpus such that the tokenization will lead to as few unknown tokens as possible.

3.3. Model Architecture

As mentioned in sections 2.1.3, The transformer models used in this thesis study are nepaliBERT [10], NepBERTa [11], distilBERT (say *NepDistilBERT*) and DeBERTa (say *NepDeBERTa*) [12], multilingual BERT [17], XLM-RoBERTa [25]. Other than that, this study pre-trained a distilBERT model which is termed as *distilBERT-Nepali*.

In the case of the GPT model, during the time of this study, there were no GPT-based Nepali pre-trained models available. Hence, this study pre-trained a GPT2 and distilGPT2 model, which are termed as *GNePT* (Generative Nepali Pre-trained Transformer) and *distilGPT-Nepali* respectively.

Other than the transformer model, this study also uses SVM. SVM is used for a hybrid model where the features from the above-mentioned transformer model are used to train the SVM classifier. This study also uses a separate 3-layer MLP (input, hidden, and output) where the input layer is the transformer model. The transformer model itself uses a dense NN layer for classification, however, the standard method (and also in the implementation of the library used) is that the classification layer is a single output NN layer. This thesis has attempted to study the effect of adding a hidden layer, between the transformer model and the output classifier layer, to the performance of the model.

3.3.1. Support Vector Machines

Support Vector Machines (SVM) is a system for efficiently training the linear learning machines in the kerned-induced feature spaces while respecting the insights provided by the generalization theory, and exploiting the optimization theory [31]. SVM aims to devise a computationally efficient path to learning good separating hyperplanes in a high-dimensional feature space. The hyperplane then can be used to classify unseen data. This is illustrated in Figure 6.

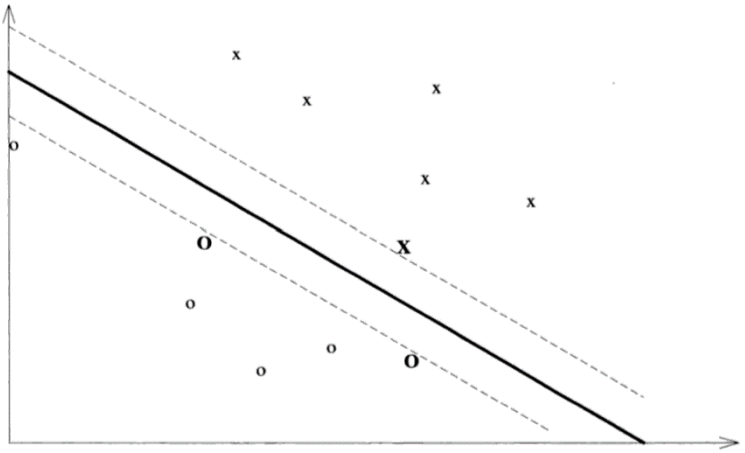

Figure 6: A maximal margin hyperplane with its support vectors highlighted [31]

The optimization problem for SVM can be written as

$$\text{Minimize(f)} = \frac{|w|^2}{2} \qquad \text{eq(6)}$$

subject to constraints $y_i(w.x_i - b) \geq 1$

Where, $x_i \in x$ (a point data vector), $y_i \in (+1, -1)$, w is the weight vector and b is bias.

The SVM applies a kernel trick to work on data that are not linearly separable, which internally maps the input feature into another feature space of higher dimension that is linearly separable [7].

Table 3: Tokenization By Custom BPE Tokenizer

	Before tokenization	After tokenization
Nepali (Devanagari):	फ्लु	[CLS] फ्लु [SEP]

3.4. Feature Extraction and Hybrid Approaches

Features extracted from the transformer model were used to train SVM and MLP classifiers. The transformer model was first fine-tuned for sentiment analysis before feature extraction. The hybrid approach also includes using two transformer model features to train the SVM and MLP classifier. For this mixed transformer feature approach, a BERT + GPT2 and DistilBERT + DistilGPT combination were used.

3.5. Model Pre-training and Fine-tuning

As mentioned in section 3.3, this study pre-trained disitlBERT-Nepali, distilGPT-Nepali, and GNePT models. The distilGPT-Nepali and the GNePT were pre-trained on 20GB of combined corpus whereas distilbERT-Nepali was trained on only 6GB of data (excluding NepBERTa corpus). The tokenizer used for these pre-trained models was also custom-trained. All three models make use of the same tokenizer. The tokenizer used is word-level BPE and it almost works similar to the tokenizer used by BERT. However, this tokenizer does not have the tokenization anomaly as shown in Table 2. The result of tokenization is shown in Table 3. The tokenizer tokenized the word as it is.

All the models specified in section 3.3 were fine-tuned for sentiment analysis on the NepCov19Tweets and NepCov19TweetsPlus dataset. Only handful of models were fine-tuned on NepaliTweets dataset as well as handful of models were used on domain adaptation and hybrid model approaches.

3.6. Evaluation and Comparison

Fine-tuned SA models were evaluated on the validation set separated from the collected Nepali social media text dataset (NepCov19Tweets, NepCov19TweetsPlus). The performance evaluation of SA models was done using standard evaluation metrics like accuracy, precision, recall, and F1-score.

3.6.1. Confusion Matrix

Confusion matrix is a tabular representation of the performance of classification model on evaluation. It helps visualize the efficiency of model to classify the data points in evaluation set

by portraying the amount of data points in predicted labels in contrast to the actual labels the data point belongs to. The confusion matrix is an NxN matrix, where N is the number of target classes.

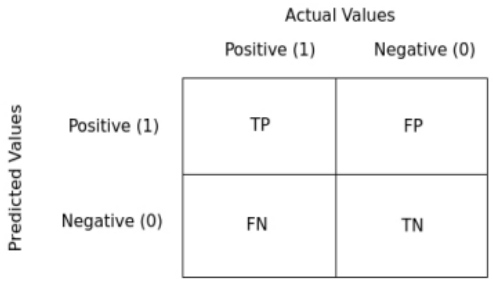

Figure 7 Confusion Matrix

A confusion matrix is shown in Figure 7 with two class (positive and negative), where,

TP → True Positive; represents positive data which is correctly labeled as positive

FP → False Positive; represents a negative data which is incorrectly labeled as positive

FN → False Negative; represents a positive data which is incorrectly labeled as negative

TN → True Negative; represents a negative data which is correctly labeled as negative

3.6.2. Accuracy

Accuracy is a ratio of the total number of correct prediction (TP + TN) to the total number of predictions made (TP + FP + TN + FN). This shows the model's overall capability to correctly identify any specific data point (text to a correct sentiment class).

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

3.6.3. Precision

Precision is a measure of ratio of true positive prediction to the total number of positive predictions (TP + FP) made by the model. It is a class-wise evaluation metric which shows model's capability to be correct out of all the predictions it made in favor of a particular class.

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

3.6.4. Recall

Recall is a measure of ratio of correct prediction made in particular class (TP) to the total number data points in that particular class (TP + FN). It is also a class-wise evaluation metric which shows model's capability to remember the characteristics of each class, thus being able to correctly identify the data points that belongs to that class.

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

3.6.5. F1 Score

F1 score is calculated as a harmonic mean of precision and recall. Thus, it symmetrically portrays both precision and recall in one metric.

$$\text{F1 score} = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

Chapter 4. Implementation

Various transformer-based models were fine-tuned to carry out the task of SA. After fine-tuning, the comparative analysis was carried out. For training and testing of these models different tools, platforms and libraries have been used.

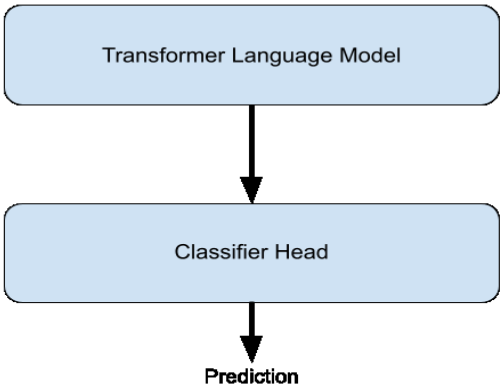

Figure 8: general organization of transformer-based sentiment analysis model

4.1. Implementation tools, platforms and libraries

1. Python3 programming language was used.
2. Hugging-face transformer library was used for the transformer model training and testing. All the models used from the transformer library were based on the Tensorflow backend. Tensorflow was also used for implementing MLP model as well as computing confusion matrix and implementing feature extraction module.
3. The dataset library and tokenizer library form Hugging-face are used for data preparation and tokenization, as well as Pandas and Numpy were also used.
4. Scikit-learn was used for the SVM model instantiation and training. Scikit-learn was also used for computing performance metrics (accuracy, precision, recall and f1-score).
5. Matplotlib and Seaborn were used for confusion matrix visualization as well as training loss/accuracy visualization.
6. Hugging-face-Hub and Kaggle datasets were used as data repositories.
7. The computing platforms used in this research are as follows:
    a. Personal Laptop:

i. CPU: 11th Gen Intel® Core™ i9-11900H @ 2.50GHz
ii. RAM: 32 GB
iii. GPU: Nvidia GeForce RTX 3050 Ti Laptop GPU with 4 GB memory
iv. Text Editor: VS code and Jupyter Notebook

b. Google Colab notebook environment with T4, V100, and A100 GPU accelerators.
c. Kaggle notebook environment with T4 and P100 GPU, and TPU V3-8 accelerators.

4.2. Implementation Details

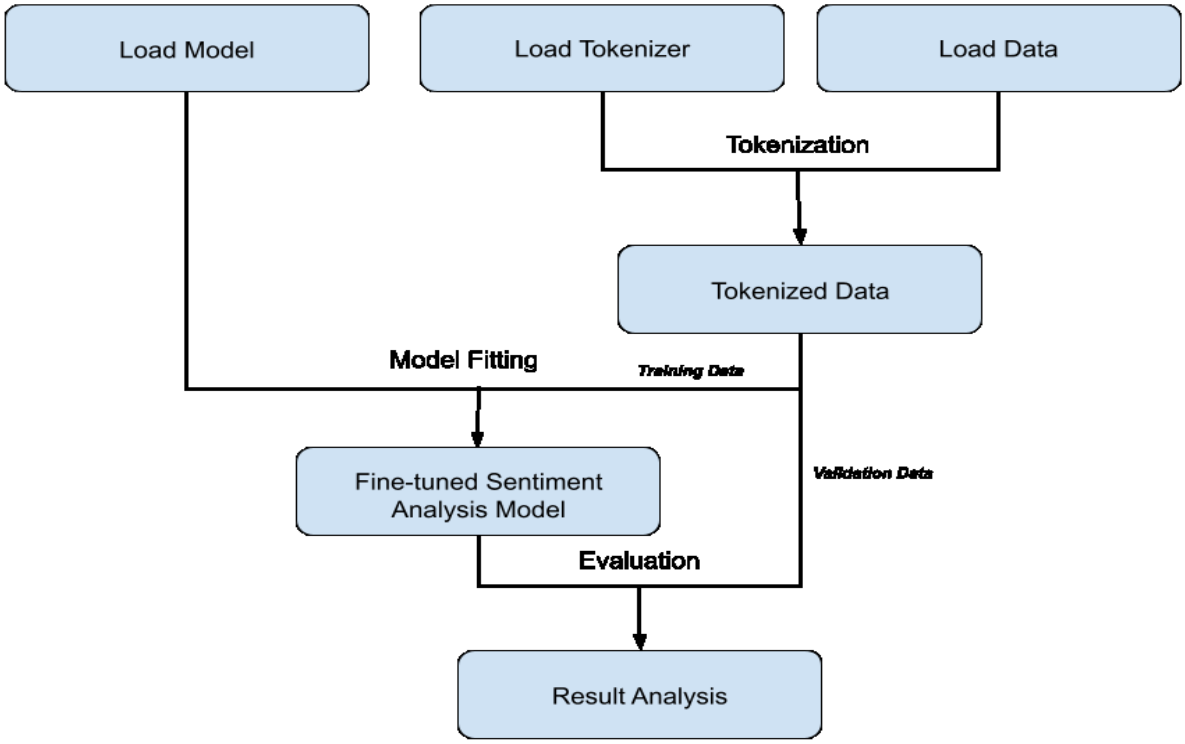

Figure 9: Model Fine-tuning for sentiment analysis workflow

4.2.1. Transformer-based Sentiment Analysis

The training and testing of transformer-based sentiment analysis models, be it BERT-based or GPT-based, almost follow the same pattern. This is due to the generality of classification architecture as implemented in the Hugging-face transformers library, which can be seen in Figure 8. The classification architecture for sentiment analysis consists of two components:

i. The transformer language model. These are the language models that are mentioned in section 3.3.

ii. The classifier head. It is a single dense NN layer that contains neurons equal to the number of sentiment classes.

The workflow of model fine-tuning of sentiment analysis is done with the following steps:

i. Tokenizer and Dataset loading
ii. Tokenization
iii. Model loading and hyper-parameter tuning
iv. Model fitting
v. Evaluation

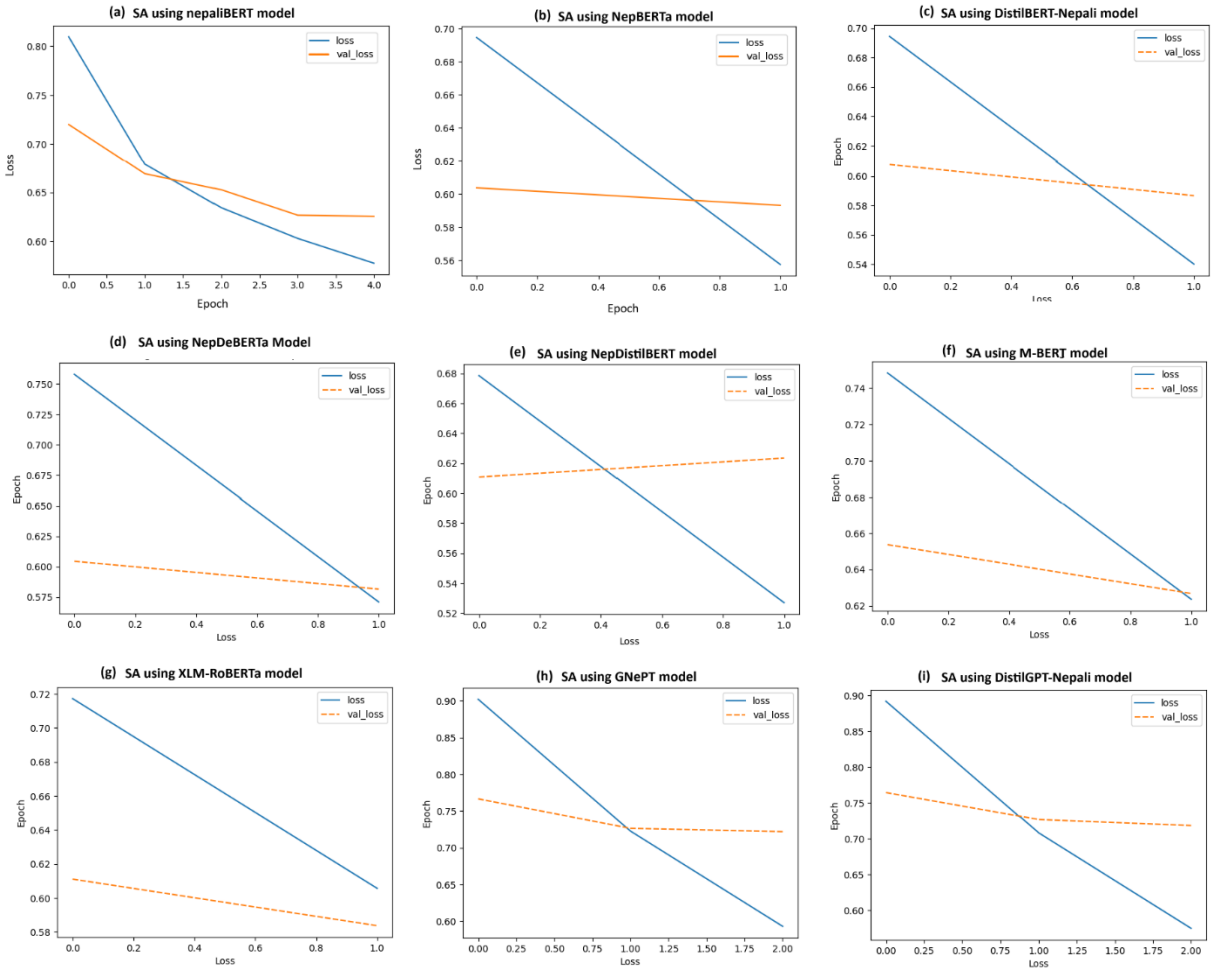

Figure 10: Training loss / Validation loss of Fine-tuning (a) nepaliBERT, (b) NepBERTa, (c) DistilBERT-nepali, (d) NepDeBERTa, (e) NepDistilBERT, (f) M-BERT, (g) XLM-RoBERTa, (h) GNePT, and (i) DistilGPT-Nepali transformer models for sentiment analysis on NepCov19Tweets dataset

This workflow is depicted in Figure 9. The work flow is quite similar for all of the transformer model fine-tuning. The difference lies in tokenization and hyper-parameter tuning. Different hyper-parameters have been used to fine-tune different models. This is because the objective was to get the optimally fine-tuned model and most of the models performed their best with different hyper-parameters. In Figure 10, it can be seen that different models were trained for different number of epochs. The training instances shown in Figure 10 are the ones when the respective model gave its optimal performance.

In tokenization, the padding strategy is different for BERT-based models and GPT-based models. BERT-based models use the first token output as a feature for downstream tasks which is subjected to a pooling layer [17], so the padding is appended to the text. In contrast, GPT-based models use the last token output as a feature for downstream tasks, this is because of the causal nature of decoder models, hence the padding is prepended to the text. This is an important distinction to remember, especially in the case of GPT-based models, because if padding is appended to the text then by the time model processes last token, it mostly observes the padding token. Thus the model can't make distinction between different texts which have padding appended to them.

4.2.1.1.Hyper-parameters

This study doesn't make any changes to internal standard parameters of the transformer model like the number of layers, dimension of the model, hidden size, the number of attention heads, dropout probabilities, etc. for most of the sentiment analysis task. The hyper-parameters that were subject to tuning in this research are

(i) *Number of epochs*: This study observes one-shot learning and few-shot learning. Thus, the number of epochs used for different models ranges from 1 to 12.

(ii) *Batch size*: The batch-size used was 16 for both the training and validation set. The choice of batch-size was limited by the GPU memory available in the computing platforms used. Increasing the batch size from 16 results in out-of-memory.

(iii) *Learning rate*: In case of one-shot learning and two-shot learning, learning rate of 0.00002 (2e-5) was used. Other than that, a learning rate from 1e-6 to 9e-6 were also used.

(iv) *Weight decay rate*: It is a regularization parameter of the optimizer function. It penalizes model weights reducing the chances of model overfitting. In the case of one-

shot learning and two-shot learning, weight decay rate of 0.01 and 0.03 are used. Other than that, weight decay rates ranging from 0.001 to 0.009 were also used.

4.2.1.2. Training set and validation set

The dataset is split into training set and validation set. The ratio used for training set and validation set is 80:20. A module available in hugging-face datasets library is used to split the dataset into training set and validation set. The module selects approximately 20% of data from each class and separates it for validation set and remaining is set to training set.

4.2.2. Transformer Language Model Pre-training

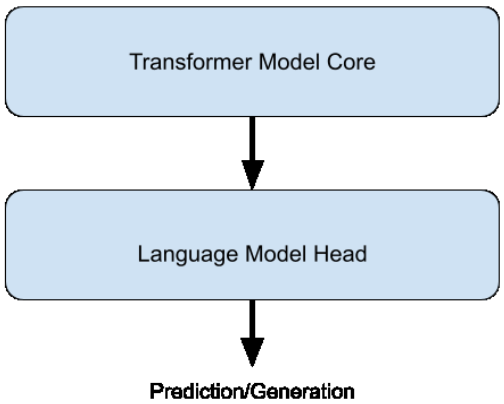

Figure 11: General organization of transformer language model

This research study conducted the pre-training of three language models, two GPT2-based models and one BERT-based model. The language model consists of two components: one is the transformer core and second is the language model head as seen in Figure 11. The model core is either the decoder-block (decoder-only) or encoder-block (encoder-only), depending on the model used, as seen in Figure 1. The language model head is the dense layer which makes predicts the next token (if GPT) or mask token (if BERT). The number of neurons in language model head is equal to the vocabulary size of the model i.e. total number of tokens used to represent the language.

4.2.2.1. Pre-Training DistilBERT-Nepali

DistilBERT-Nepali is the DistilBERT (BERT-based) model. It is trained on 6 GB Nepali corpora, *Nepali-Extended-Text-Corpus* as mentioned in section 3.1. The data contained approximately 441 million tokens after tokenization and 512 tokens per sequence was used for training. The model is trained on the Kaggle notebook environment using GPU P100 accelerator. The *AdamWeightDecay*

optimizer was used and the optimizer parameter learning rate and weight decay rate were set to 5×10^{-5} and 0.01 respectively. Also, Tensorflow mixed precision was set to *mixed_float16*, which, as per Tensorflow documentation, is the recommended policy when using GPU. The masking probability was set to 25%, which is 10% higher than what was used by [17]. The training was done in 3 batches of data. The GPU memory provided by the platform wasn't sufficient to load all the data at once. So, the total training had 6 runs with 1 epoch per run. Therefore, the total number of epochs is 2. The batch-size used for training is 16. The model's internal parameter wasn't changed. The model attained 2.9975 training loss and 2.8496 validation loss by the end of the training, this can be seen in Figure 12. This resulted in 17.31 model perplexity. The model has approximately 82 million parameters. The model's internal parameters are

- Context length: 512
- Dimension of model (embedding size): 768
- Number of layers: 6
- Number of attention heads: 12
- Vocab size: 50000
- Feed forward dimension: 3072

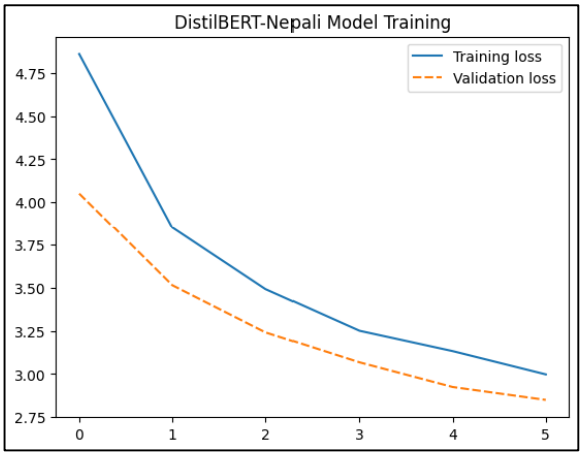

Figure 12: Training / Validation loss of Pre-trianing DistilBERT-Nepali.

4.2.2.2. Pre-Training DistilGPT-Nepali

DistilGPT-Nepali is GPT2-based DistitlGPT2 model with knowledge distillation applied similar to DistilBERT. It is trained on 20 GB of Nepali corpora, which on tokenization resulted in 1.5 billion tokens. And for training, 512 tokens per sequence was used. The computing platform used to train this model was the Kaggle notebook environment with TPU V3-8 accelerator. The model's

internal parameter wasn't changed when initialization. The batch size used for training is 8 per TPU core, so the total batch size was 64. The *AdamWeightDecay* optimizer was used and the optimizer parameter learning rate and weight decay rate were set to 5×10^{-5} and 0.03 respectively. Also, Tensorflow mixed precision was set to *mixed_bfloat16*, which is the recommended policy when using TPU, as per the Tensorflow documentation. The model was trained for 2 epochs. The model attained 4.57 training loss and 4.43 validation loss, this can be seen in Figure 13. This resulted in 84.05 model perplexity. The model has approximately 82 million parameters. The model's internal parameters are

- Context length: 1024
- Embedding size: 768
- Number of layers: 6
- Number of attention heads: 12
- Vocabulary size: 50000

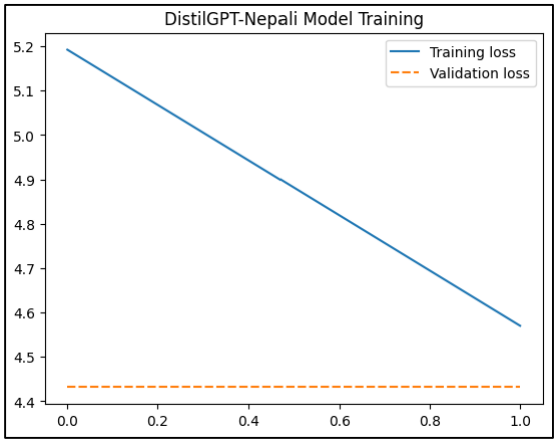

Figure 13: Training / Validation loss of Pre-trianing DistilGPT-Nepali.

4.2.2.3. Pre-training GNePT

GNePT is GPT2-based model. It is trained on 20 GB of Nepali corpora, which on tokenization resulted in 1.5 billion tokens. And for training, 512 tokens per sequence was used. The computing platform used to train this model was the Kaggle notebook environment with TPU V3-8 accelerator. The model's internal parameter was not changed when initialization. The batch size used for training is 8 per TPU core, so the total batch size was 64. Also, Tensorflow mixed precision was set to *mixed_bfloat16*, which is the recommended policy when using TPU, as per the Tensorflow documentation. The optimizer used was *AdamWeightDecay*. The training of the

model was done in two steps. In the first step, the model was trained for two epochs with an optimizer parameter learning rate and weight decay rate set to 2×10^{-5} and 0.035 respectively. The model attained a training loss of 4.40 and a validation loss of 4.27. In the second step, the model was trained for a single epoch with an optimizer parameter learning rate and weight decay rate set to 2×10^{-6} and 0.0035 respectively. The model attained 4.35 training loss and 4.19 validation loss, this is show in Figure 14. This final model perplexity was 66.59. The model has approximately 124.24 million parameters. The model's internal parameters are

- Context length: 1024
- Embedding size: 768
- Number of layers: 12
- Number of attention heads: 12
- Vocabulary size: 50000

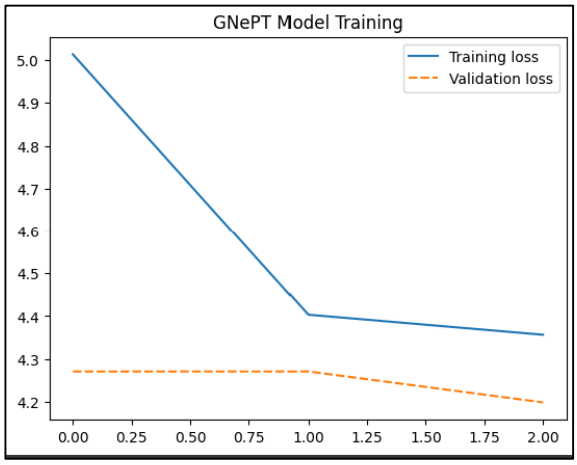

Figure 14: Training / Validation loss of Pre-trianing GNePT

4.2.3. Hybrid model-based Sentiment Analysis

The hybrid model is a tandem model architecture where two different models work for the end goal of sentiment analysis. One is the feature extraction model and another is the end classifier model. The feature extraction model in this study is a transformer model. This study has taken two hybrid model-based approach in consideration, (i) using the SVM model as an end classifier and (ii) using the MLP model as an end classifier. In both approaches, the transformer model's output embedding is used as a feature to train and evaluate the classifier model. In the case of the MLP model, it can be considered as pooling layer within the transformer sentiment analysis model. The

embedding used is the first token embedding in the case of BERT-based models. In contrast, the last two token embedding is used in the case of GPT-based models. The two feature vector are then subjected to *LogSumExponentional* function to obtain a single vector. All BERT-based models, except for DistilBERT, have an extra pooling layer that takes the first token embedding to give the final embedding. In case of GPT-based models there is no pooling layer.

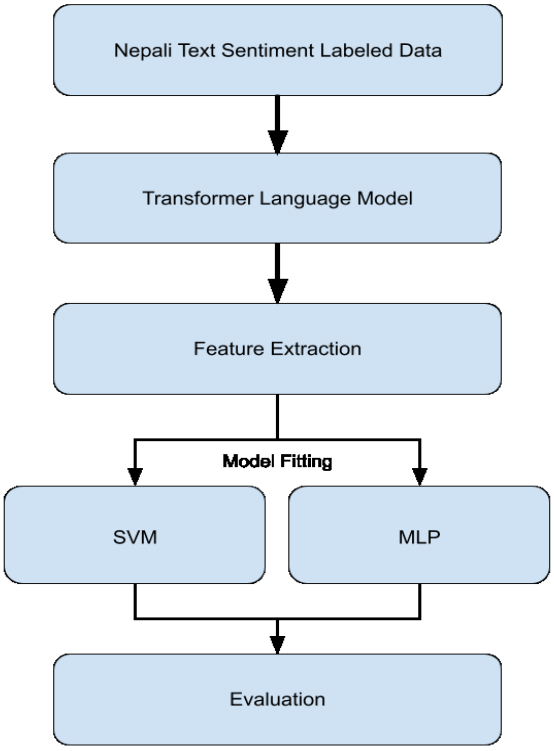

Figure 15: Workflow of Hybrid model-based sentiment analysis approach

This study also conducts a combined model feature extraction approach, where features from two transformer models combined is used to train the underlying end classifier (SVM and MLP). The overall working is similar to what is shown in Figure 9 (where the classifier head is our end classifier), except for the feature extraction that needs to be done separately in the hybrid model approach. This is portrayed in Figure 15.

The MLP model is a 3-layer neural network, input layer, hidden layer, and output layer built using Tensorflow's Keras API. The hidden layer of MLP has the same size as the input layer. The size of the input layer is 768 as all of the models used, be it BERT-based or GPT-based, have the model dimension (embedding size) of 768. In the case of 2 model feature hybrid approach, the context

features from two models are concatenated to make a single context feature vector. Thus the size of context feature vector is 1536.

In the case of SVM, the Scikit-learn implementation is used. The kernel used in SVM is RBF kernel. And OVO (one-versus-one) approach for multiple class classifier is applied. All the other parameters of the SVM model are default parameters as provided in Scikit-learn. Feature dimension used for the training of SVM model is same what is used for MLP model, 768 in case of single model context features and 1536 in case of 2 model context features.

Chapter 5. Result and Analysis

5.1. Sentiment Analysis on NepCov19Tweets Dataset

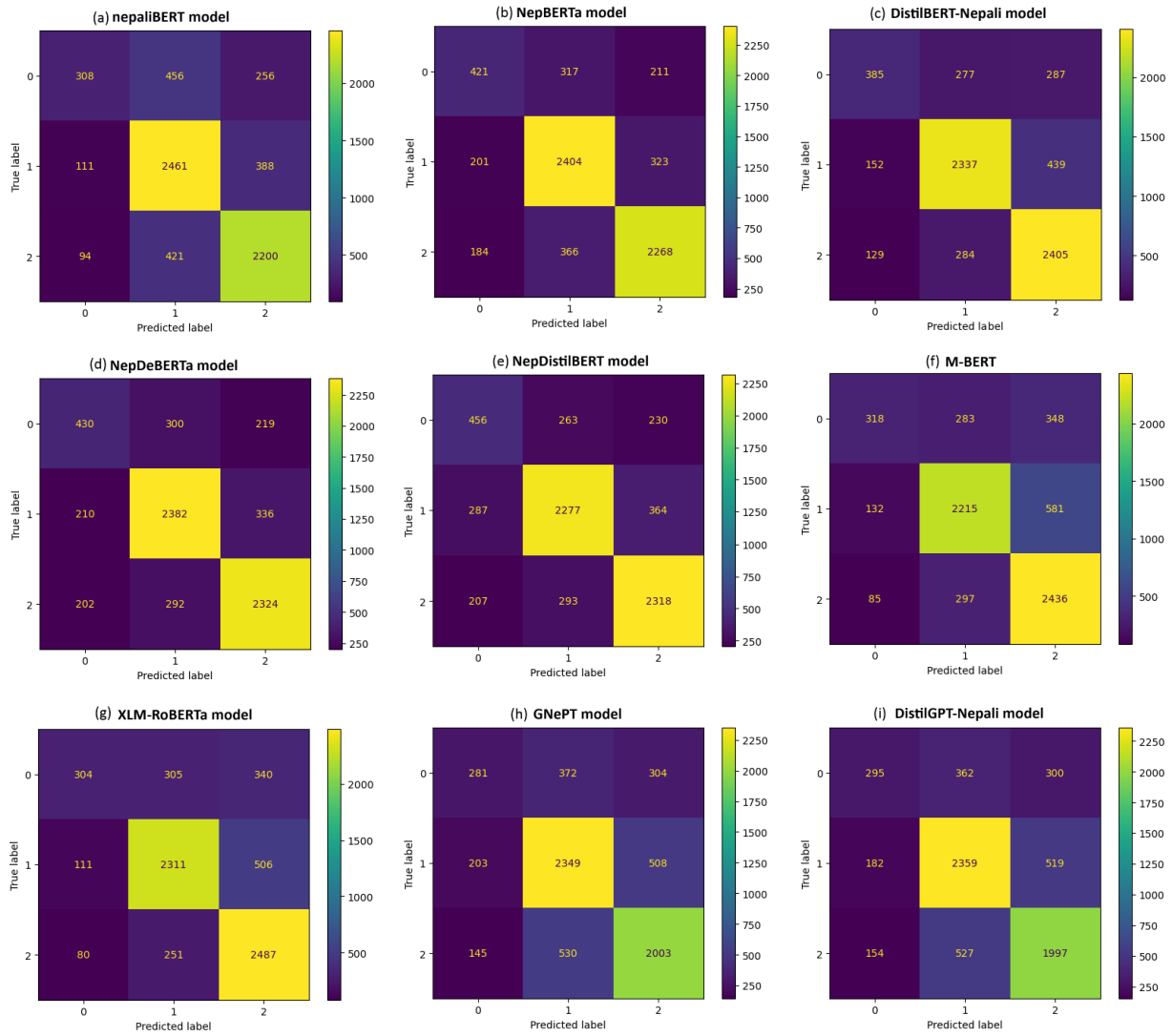

Figure 16: Confusion matrix - NepCov19Tweets (0 – Neutral, 1 – Positive, 2 – Negative)

The model's performance after training each model on validation split (approx. 20% of data from each class) of *NepCov19Tweets* dataset is shown in Table 4. The language models that were trained as part of this study has been represented with an (*) in Table 4 and other tables later. The precision, recall and F1-score are computed as weighted average of each class. It was observed that all the BERT-based models performed better than the GPT-based models. It was also observed that all the models performed comparatively better than what was achieved in [6], [7] ,and [8]. However,

the overall accuracy of MCNN in [8] is higher than what is achieved by GPT-based models in this study. Similarly, the SVM+RBF model with hybrid feature (FastText and TF-IDF feature) in [7] performed better than GPT-based models in this study.

Table 4: Performance metrics on evaluation of SA model on NepCov19Tweets dataset.

	Precision	Recall	Accuracy	F1-score
nepaliBERT	0.7311	0.7421	0.7421	0.7277
NepBERTa	0.7553	0.7607	0.7607	0.7573
DistilBERT-Nepali *	0.7579	0.7657	0.7657	0.7587
NepDeBERTa	0.7624	0.7671	0.7671	0.7645
NepDistilBERT	0.7546	0.7544	0.7544	0.7543
M-BERT	0.73.55	0.7422	0.7422	0.7309
XLM-RoBERTa	0.7536	0.7620	0.7620	0.7488
GNePT *	0.6787	0.6920	0.6920	0.6826
DistilGPT-Nepali *	0.6824	0.6947	0.6947	0.6857

The model fitting procedure provided by the transformers library doesn't provide option to pass the metrics like accuracy to be used, so the default metrics during the training that can be observed are the training loss and the validation loss. The training loss and validation loss of the models on *NepCov19Tweets* is shown in Figure 10. A strong correlation between the model's accuracy and the loss was observed (training/validation). Here, NepBERTa, DistilBERT-Nepali, NepDeBERTa, and XLM-RoBERTa are the model's that achieve accuracy above 76%, and in loss graph (Figure 10) these are the model's that have validation loss below 0.6. The highest performance is by NepDeBERTa model with 76.71% accuracy and 76.45% f1-score.

In Figure 16, the confusion matrix of each model is displayed. The values in those confusion matrices are the ones used to compute the evaluation metric values in Table 4. In those confusion matrices, it can be witnessed that the model's performance for label 0 (Neutral class) is very poor. None of the model was able to predict even the 50% of the total data in that class correctly.

This study assumed the reason for this poor performance of models for the neutral class is the highly disproportionate amount of data available in the neutral class of *NepCov19Tweets* dataset. This is the motivation that lead this study for the data augmentation of neutral class leading to creation of *NepCov19TweetsPlus* dataset. The result analysis of sentiment analysis on *NepCov19TweetsPlus* dataset using transformer models are discussed in the next section.

5.2. Sentiment Analysis on NepCov19TweetsPlus Dataset

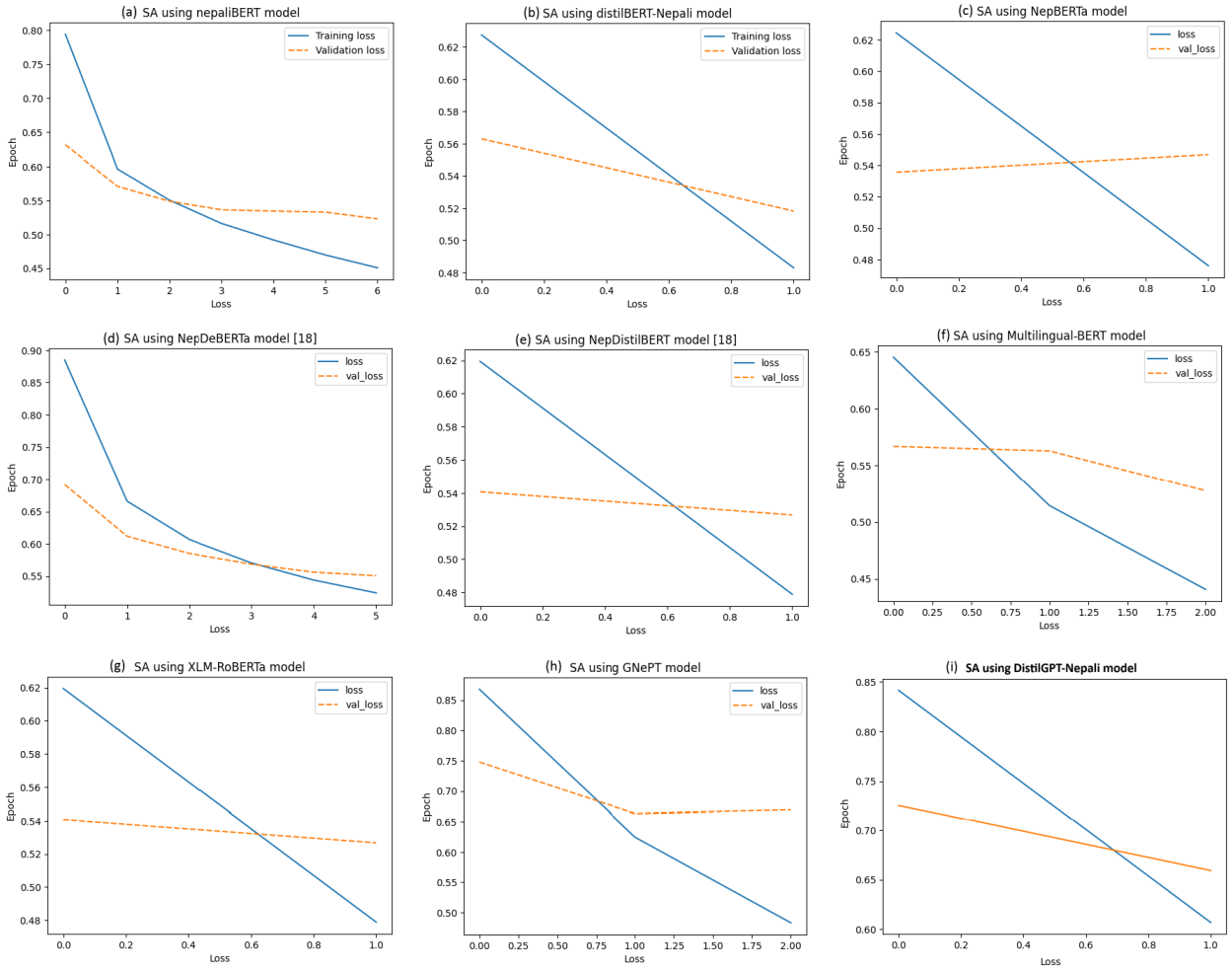

Figure 17: Training loss / Validation loss of Fine-tuning (a) nepaliBERT, (b) NepBERTa, (c) DistilBERT-nepali, (d) NepDeBERTa, (e) NepDistilBERT, (f) M-BERT, (g) XLM-RoBERTa, (h) GNePT, and (i) DistilGPT-Nepali transformer models for sentiment analysis on NepCov19TweetsPlus dataset

After adding the data in neutral class through data augmentation, all the models trained on this new dataset performed better than they previously did with *NepCov19Tweets* dataset. This can be observed in Table 5, there is 2% - to - 7% increase in models' performance. The models' capacity to recognize neutral class increased significantly for each class, however, the models' capacity to classify positive and negative class changed only slightly and fairly the same as it was before. The confusion matrices in Figure 18 gives a glance at the class-wise classification capacity of different

models. The performance of all the BERT-based models are fairly similar, with highest of 79.73% accuracy by nepaliBERT. The performance of nepaliBERT exceeded that of NepDeBERTa which performed the best on NepCov19Tweets dataset, as can be seen in section 5.1.

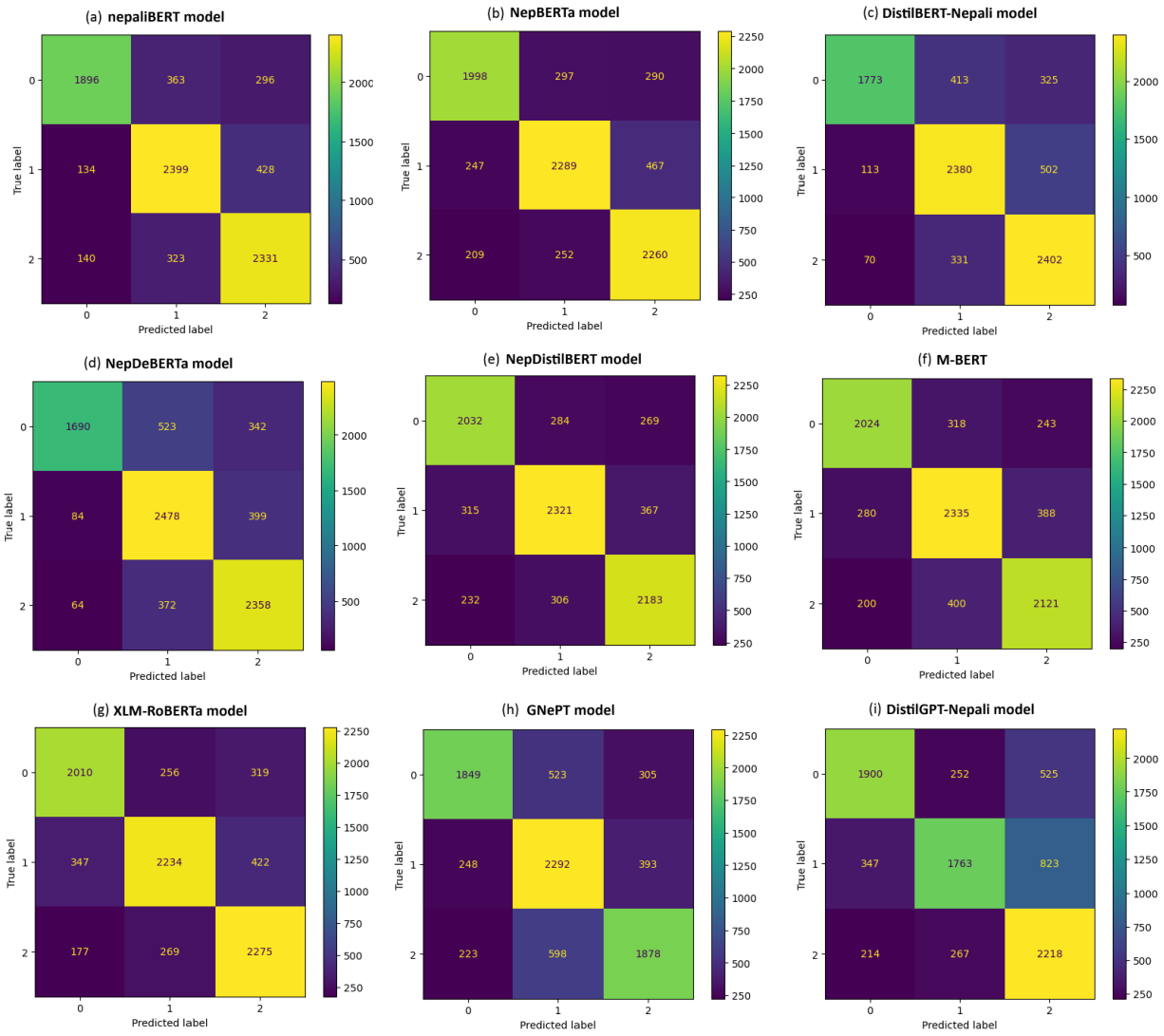

Figure 18: Confusion matrix - NepCov19TweetsPlus (0 – Neutral, 1 – Positive, 2 – Negative)

Despite the boost in all of the models' performance, the GPT-based models are still performing very poorly, in comparison to the performance of every other BERT-based models. The validation loss of the model dropped below 0.55 for BERT-based models. However, the loss didn't drop as much for GPT-based models. This can be seen in the loss graph in Figure 17. The BERT-based models achieved above 78% in each performance metrics, except for M-BERT with slightly below

78%. After witnessing the models' performance on using *NepCov19TweetsPlus* dataset, all the later sections make use of *NepCov19TweetsPlus* for sentiment analysis.

Table 5: Performance metrics on evaluation of SA model on NepCov19TWeetsPlus dataset.

	Precision	Recall	Accuracy	F1-score
nepaliBERT	0.8022	0.7973	0.7973	0.7975
NepBERTa	0.7901	0.7879	0.7879	0.7879
DistilBERT-Nepali *	0.7995	0.7889	0.7889	0.7889
NepDeBERTa	0.8003	0.7853	0.7853	0.7844
NepDistilBERT	0.7869	0.7866	0.7866	0.7866
M-BERT	0.7802	0.77987	0.77987	0.77998
XLM-RoBERTa	0.7864	0.7846	0.7846	0.7842
GNePT *	0.7306	0.7244	0.7244	0.7247
DistilGPT-Nepali *	0.7235	0.7078	0.7078	0.7069

5.3. Domain Adaptation

For domain adaptation, this study fine-tunes a transformer-based Nepali news classification model for Nepali sentiment analysis. Before proceeding with domain adaptation on every transformer-based model used in study, a feasibility study was done using the model with best performance on previous section. Thus, *nepaliBERT* model was chosen.

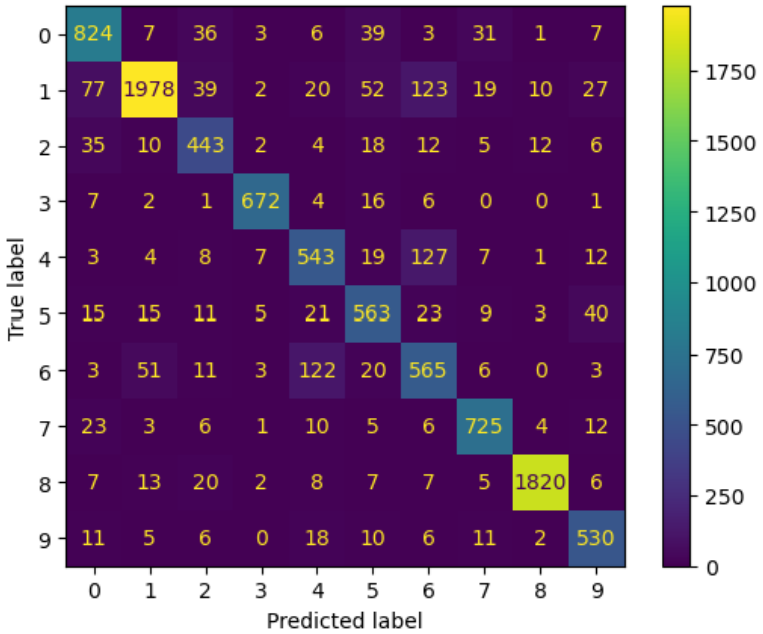

Figure 19: Confusion matrix - News Classification

Firstly, *nepaliBERT* was fine-tuned on news dataset with 10 categories. The news classification model achieved 86% accuracy along with 86.24% F1-score and 86.67% precision. The model's performance on validation set can be observed on confusion matrix in Figure 19. The labels in Figure 19 represent ArthaBanijya, Bichar, Desh, Khelkud, Manoranjan, Prabas, Sahitya, SuchanaPrabidhi, Swasthya, Viswa in this same order respectively.

After obtaining a news classification model, that model was then fine-tuned for sentiment analysis. The fine-tuned model, however performed good, didn't perform better. The performance was similar to its performance on NepCov19TweetsPlus without domain adaptation. The model attained 79.62% F1-score, 79.59% accuracy, and 80.05% precision; this is similar to what was observed in previous section. Henceforth, this study didn't proceed further with domain adaptation.

5.4.Sentiment Analysis Using Hybrid Approach

Table 6: Performance metrics on evaluation of SVM and MLP classifier using transformer model output as feature on NepCov19TweetsPlus Dataset

	Precision	Recall	Accuracy	F1-score
nepaliBERT+SVM	0.8429	0.8389	0.8389	0.8393
nepaliBERT+MLP	0.7965	0.7942	0.7942	0.7944
DistilBERT-Nepali + SVM	0.8549	0.8523	0.8523	0.8522
DistilBERT-Nepali + MLP	0.8497	0.8487	0.8487	0.8488
GNePT + SVM	0.8281	0.8230	0.8230	0.8232
GNePT + MLP	0.8248	0.8223	0.8223	0.8226
DistilGPT-Nepali + SVM	0.8195	0.8144	0.8144	0.8146
DistilGPT-Nepali + MLP	0.8246	0.8206	0.8206	0.8208
Hybrid-Feature + SVM	0.8563	0.8539	0.8539	0.8540
Hybrid-Feature + MLP	0.8570	0.8565	0.8565	0.8566
distilHybrid-Feature + SVM	0.8599	0.8581	0.8581	0.8582
distilHybrid-Feature + MLP	0.8604	0.8595	0.8595	0.8596

For hybrid approach, this study obtains the feature vector from the transformer model and then uses it to train SVM and MLP classifier. The fine-tuned models which were trained in section 5.2 are used in hybrid approach. For model selection the same approach from section 5.3 is used, which is to select the model with highest performance. In total four models were selected, two BERT-based and two GPT-based. Hence, the *nepaliBERT* and the *distilBERT-Nepali* were chosen from BERT-based models, whereas there are only two GPT-based models used in this study,

GNePT and *distilGPT-Nepali,* so they were used. In case of two model combined feature, a combindation of {*nepaliBERT* and *GNePT*} (Hybrid-Feature)*, and* {*distilBERT-Nepali* and *distilGPT-Nepali*} (distilHybrid-Feature) is used. Thus, there are total of 6 different SVM models and 6 different MLP models.

The performance of 12 models (6 SVM, 6 MLP) trained is shown in Table 6. There it can be noticed that the overall performance boost due to use of SVM and MLP classifier is fairly same for each of the feature extraction, except for *nepaliBERT* where the MLP classifier has no any effect in model's performance. The performance of *nepaliBERT* with MLP classifier is similar to its performance as seen in section 5.2. This observation in this section gave off two insights,

i. using external classifier with transformer output feature vectors is better approach to sentiment analysis rather than only using transformer model, and
ii. adding a pooling layer to the *DistilBERT-Nepali*, *DistilGPT-Nepali*, and *GNePT* (i.e. any model which don't have a pooling layer between language model core and classification head like BERT does) increases the models' efficiency in classification task like sentiment analysis.

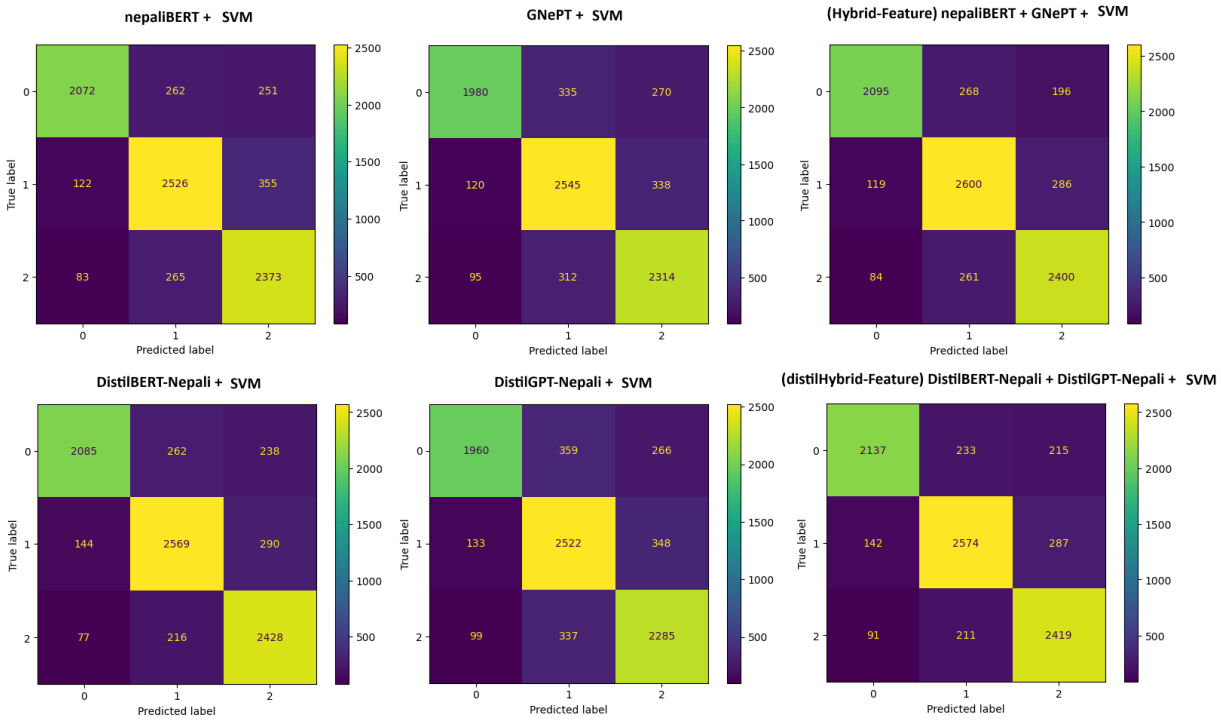

Figure 20: Confusion Matrix - SVM classifier NepCov19TweetsPlus Dataset

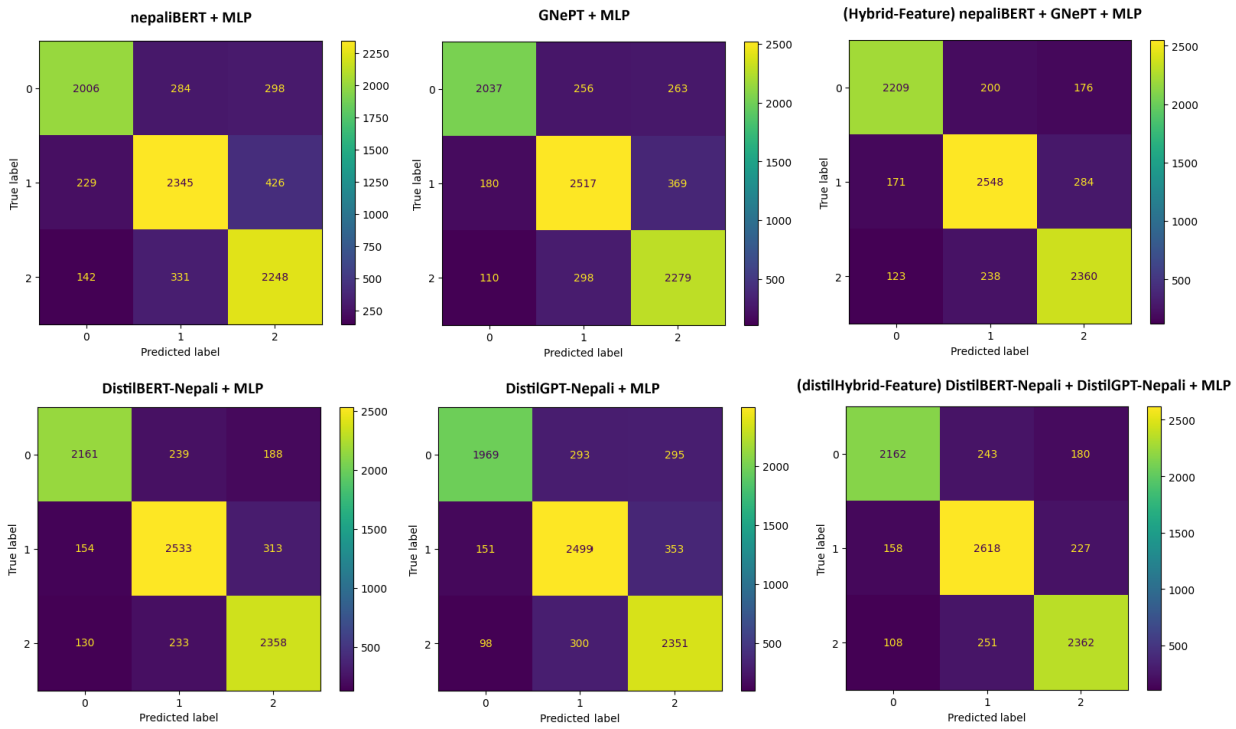

Figure 21: Confusion Matrix - MLP classifier NepCov19TweetsPlus Dataset

It is observed that the *DistilBERT-Nepali + SVM* has the highest performance among others, in case of single model feature. However, the MLP-based hybrid model performed slightly better in case of two model feature. The overall highest score is attained by *distilHybrid-Feature + MLP*, which scored 85.95% accuracy and 85.96% F1-score. The models' performance across different sentiment class can be seen in confusion matrices in Figure 20 and Figure 21.

5.5. Sentiment Analysis on NepaliTweets Dataset

The data collected in this study is very small, only 2499 labeled texts. But, the transformer models require very large number of data. Thus, it was combined with *NepCov19TweetsPlus* and then models were trained and evaluated. For this task, four models were selected same as in previous section 5.4. The performance of these models are shown in Table 7 and as confusion matrix in Figure 22. It was observed that the capacity of *nepaliBERT, DistilBERT-Nepali,* and *GNePT* model slightly decreased on NepaliTweets in comparison to what it achieved with only *NepCov19TweetsPlus* dataset. In contrast, the performance of *DistilGPT-Nepali* slightly increased to what it achieved on *NepCov19TweetsPlus* dataset.

Even here, it can be noticed that BERT-based models perform better than GPT-based models, significantly. There is difference of minimum 6% across all the performance metrics between BERT-based models and GPT-based models.

Table 7 Performance metrics on evaluation of transformer models on NepaliTweets dataset

	Precision	Recall	Accuracy	F1-score
nepaliBERT	0.7943	0.7843	0.7843	0.7834
DistilBERT-Nepali *	0.7955	0.7747	0.7747	0.7740
GNePT *	0.7154	0.7124	0.7124	0.7125
DistilGPT-Nepali *	0.7168	0.7149	0.7149	0.7150

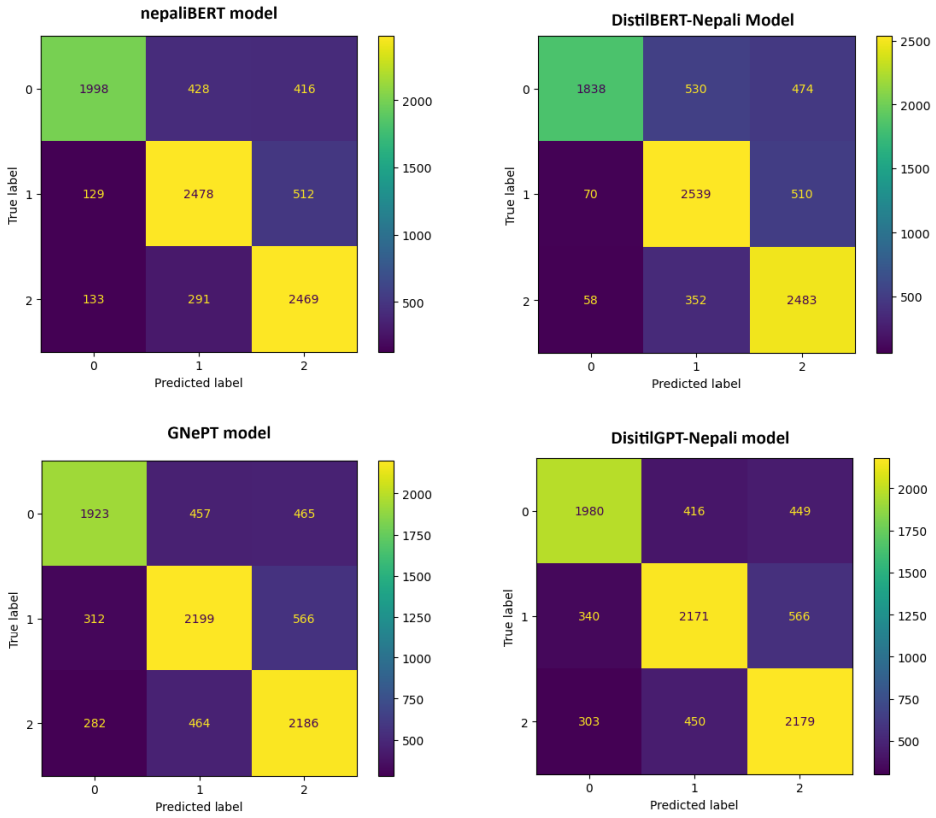

Figure 22: Confusion Matrix – NepaliTweets dataset

Chapter 6. Conclusion and Future Recommendation

6.1. Conclusion

The main objective of this study was to perform sentiment analysis on Nepali social media texts using transformer models and compare their performance. The maximum result attained in this study is on NepCov19TweetsPlus dataset by the use of hybrid tandem model approach where combined feature from DistilBERT-Nepali and DistilGPT-Nepali were used to train a MLP end classifier, which is 86.04% precision, 85.95% accuracy and 85.96% f1-score.

In this experiment, it was found that the BERT-based models generally performed better than GPT-based models for the task of sentiment analysis of Nepali texts. The maximum of 76.71% accuracy and 76.45% f1-score was attained in NepCov19Tweets dataset by NepDeBERTa model, 79.73% accuracy and 79.75% f1-score was attained in NepCov19TweetsPlus dataset by nepaliBERT model, and 85.23% accuracy and 85.22% f1-score was attained in NepCov19TweetsPlus dataset by distilBERT-Nepali + SVM hybrid model approach. The above results show that the use of data augmentation and hybrid model approach significantly increase the models' performance.

In case of hybrid model approach, it was generally observed that using an external classifier, either MLP or SVM, boost the performance. However, there is an outlier in nepaliBERT model, which didn't perform any better with MLP classifier than it did without it. This may be attributed to the fact that there is already a pooling layer in BERT architecture which is absent in other three transformer models used in hybrid approach.

The main takeaway from this experimental research study is that adding a pooling layer in the transformer model, like GPT2 which do not have a pooling layer, makes it robust to the downstream task of sentiment analysis, or classification in general. The results obtained show that adding pooling layer increases efficiency of GPT2 models on par with BERT models. GNePT + MLP model achieved 82.23% accuracy and distilGPT-Nepali + MLP achieved 82.06% accuracy which is higher than that of nepaliBERT + MLP and nepaliBERT in NepCov19TweetsPlus dataset. This should also be noted that DistilBERT model also doesn't have a pooling layer in it.

6.2. Future Recommendation

This study used a separate MLP classifier and suggests that adding a pooling layer boosts the models' performance on the downstream task of classification. To be more concrete, this

hypothesis needs to be tested by actually adding a pooling layer to the transformer models, that doesn't have it, and then evaluating it after fine-tuning.

One of the main limitations within this study is the limited availability of sentiment annotated Nepali social media texts. There needs to be focus on collecting more labeled data from different domains. Also, the performance outlier of nepaliBERT+MLP needs to be validated by testing it with other BERT models. This needs to be done to be more concrete regarding that MLP end classifier with BERT model didn't perform better because there is already a pooling layer in BERT.

References

[1] B. Liu, Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press, 2015.

[2] T. B. Shahi and C. Sitaula, "Natural language processing for Nepali text: a review," *Artificial Intelligence Review, Springer,* vol. 55, pp. 3401-3429, 2021.

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. v. Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji and others.., "On the Opportunities and Risks of Foundation Models," 2022.

[4] C. P. Gupta and B. K. Bal, "Detecting Sentiment in Nepali Texts: A Bootstrap Approach for Sentiment Analysis of texts in the Nepali Language," 2015.

[5] O. M. Singh, S. Timalsina, B. K. Bal and A. Joshi, "Aspect Based Abusive Sentiment Detection in Nepali Social Media Texts," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020.

[6] C. Sitaula, A. Basnet, A. Mainali and T. B. Shahi, "Deep Learning-Based Methods for Sentiment Analysis on Nepali COVID-19-Related Tweets," *Computational Intelligence and Neuroscience,* 2021.

[7] T. B. Shahi, C. Sitaula and N. Paudel, "A Hybrid Feature Extraction Method for Nepali COVID-19-Related Tweets Classification," 2022.

[8] C. Sitaula and T. B. Shahi, "Multi-channel CNN to classify Nepali Covid-19 related tweets," 2022.

[9] 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,* p. 1–12, 2020.

[10] S. Pudasaini, A. Tamang, S. Lamichhane, S. Adhikari, S. Adhikari, S. Thapa and J. Karki, "Pre-training of Masked Language Model in Nepali," in *36th Conference on Neural Information Processing Systems*, 2022.

[11] M. Gautam, S. Timalsina and B. Bhattarai, "NepBERTa: Nepali Language Model Trained in a Large Corpus," in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the*, 2022.

[12] U. Maskey, M. Bhatta, S. R. Bhatta, S. Dhungel and B. K. Bal, "Nepali Encoder Transformers: An Analysis of Auto Encoding Transformer Language Models for Nepali Text Classification," in *Proceedings of SIGUL2022 @LREC2022*, 2022.

[13] S. Suwanchai, S. R. Tamrakar and Chaklam, "Comparative Evaluation of Transformer-Based Nepali Language Models," [Online]. Available: https://assets.researchsquare.com/files/rs-2289743/v1/aa3f3ba4a38a880db3d6c5dc.pdf?c=1670229384. [Accessed 19 06 2023].

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. A. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[15] D. Bahdanau, K. Cho and Y. Bengio, "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE," in *ICLR* , 2015.

[16] Huggingface, "How do Transformers work?," Huggingface, [Online]. Available: https://huggingface.co/learn/nlp-course/chapter1/4. [Accessed 27 8 2023].

[17] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Google AI Language, 2019.

[18] A. Radford and K. Narasimhan, "Improving Language Understanding by Generative Pre-Training," 2018.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners".

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh and Dani, "Language Models are Few-Shot Learners," Open AI, 2020.

[21] "Neural Machine Translation with a Transformer and Keras," Tensorflow, 03 06 2023. [Online]. Available: https://www.tensorflow.org/text/tutorials/transformer. [Accessed 24 06 2023].

[22] V. Sanh, L. Debut, J. Chaumond and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 26 7 2019. [Online]. Available: https://arxiv.org/pdf/1907.11692.pdf. [Accessed 23 06 2023].

[24] P. He, X. Liu, J. Gao and W. Chen, "DeBERTa: Decoding-enhanced BERT with Disentangled Attention," in *ICLR* , 2021.

[25] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale," arXiv, 2019.

[26] K. Kafle, D. Sharma, A. Subedi and A. K. Timalsina, "Improving Nepali Document Classification by Neural Network," in *Proceedings of IOE Graduate Conference*, 2016.

[27] T. B. Shahi and A. K. Pant, "Nepali News Classification using Naive Bayes, Support Vector Machines and Neural Networks," in *International Conference on Communication, Information & Computing Technology (ICCICT)*, Mumbai, 2018.

[28] O. M. Singh, "Nepali Multi-Class Text Classification," 2018. [Online]. Available: https://oya163.github.io/assets/resume/Nepali_Text_Classification.pdf. [Accessed 19 6 2023].

[29] S. M. Mohammad, "A Practical Guide to Sentiment Annotation: Challenges and Solutions," in *Proceedings of NAACL-HLT 2016*, San Diego, California, 2016.

[30] P. J. O. Su'arez, B. S. Romary and L. Romary, "A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[31] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel- based Learning Methods, Cambridge University Press, 2002.

Appendix

All the experiments conducted in this study, model training and fine-tuning for sentiment analysis and language model training, are available as Jupyter Notebook snapshots on the GitHub repository ***https://github.com/RayGone/Thesis***.

The datasets used and the fine-tuned models can be found on Hugging-face-Hub under user profile ***https://huggingface.co/raygx***.