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

August 2023

Table of Contents

List of tables

List of Figures

List of Abbreviations

BERT	:	Bi-directional Encoder Representation Transformer
BiLSTM	:	Bidirectional LSTM
BPE	:	Byte Pair Encoding
CNN	:	Convolutional Neural Network
DT	:	Decision Tree
GPT	:	Generative Pre-Trained Transformer
GRU	:	Gated Recurrent Unit
LSA/LSI	:	Latent Semantic Analysis / Latent Semantic Indexing
LSTM	:	Long Short Term Memory
MCNN	:	Multi-channel CNN
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

ABSTRACT

Sentiment Analysis is the task

1. Introduction

1.1. Introduction

1.1.1. Sentiment Analysis

Sentiment Analysis (SA) 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].

1.1.2. Nepali Language and its trend on Social Media

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

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, which can be seen in Table 1. Along with these characters, Nepali also consists of different modifiers and half-forms.

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].

1.1.3. Transformer

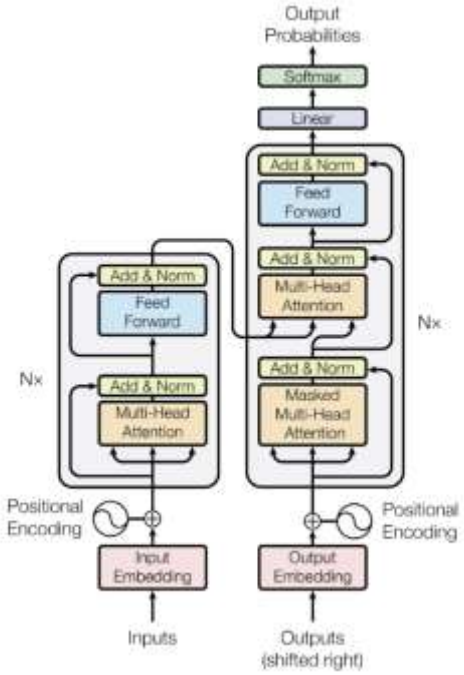

Figure 1: The transformer – model architecture [3]

In recent years, a state-of-the-art neural network architecture has revolutionized the field of Natural Language Processing (NLP), known as Transformer [3]. Transformers are neural network architectures that rely on attention mechanisms to encode and decode sequential data, as seen in Figure 1. The attention mechanism was first proposed by [4] 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 [3].

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 [3]. Based on their architectures, transformers can be divided into three categories as follows [5] :

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 [6] and GPT (GPT-1 [7], GPT-2 [8], GPT-3 [9]) 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.

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 [10]. 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 [8] [9].

1.2. 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 [10]. 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 handful of works are available [2].

The works that have been done in Nepali SA are mostly done using RNNs, CNNs, and traditional machine learning algorithms [11] [12] [13] [14] [15]. Some significant work has been done for building Nepali-only pre-trained Language Models, mostly BERT, such as NepaliBERT by [16], NepBERTa by [17] and distilBERT and DeBERTa by [18]. Some works have been done to classify Nepali news texts using transformers, such as in [18] and [19]. 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.

1.3. 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 [20]) 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.

1.4. 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 model's 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 tried to train and build optimal language models within the best capacity of the author as well as the resources and time available.

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.

1.5. Organization of the report

This report is organized in 7 chapters, excluding reference.

- Chapter **1. Introduction** includes brief introduction to sentiment analysis, trend of Nepali in digital platforms, introduction to transformer models.
- In Chapter **2. Background Study** , a brief detail of all the models used in this study are presented.

- Then in chapter **3. Literature Review**, works that have been done in the field of sentiment analysis of Nepali texts and use of transformers in Nepali texts are explored.
- In chapter **4. Research Methodology** and chapter **5. Implementation**, the overall conduct of this research study is discussed and how different models were trained and evaluated is also explained.
- Chapter **6. Result and Analysis** consists of comparative analysis of performance of different models on the task of sentiment analysis. Different findings of this research study are also presented in this chapter.
- In Chapter **7. Conclusion**, summary of all the research works and findings of this research study are given.

2. Background Study

2.1. Transformer

Transformers are deep learning architectures that completely rely on self-attention mechanisms [4] to encode and decode sequential data. The transformer is proposed by Vaswani et al. in [3]. 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.

Transformer is an encoder-decoder-based architecture. Both encoder and decoder are implemented as a stack of self-attention and point-wise, 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.

Two components make the transformer unique from other sequence-to-sequence architectures i.e. Self-Attention and Positional Encodings. In the following subsections, these two components are discussed in brief.

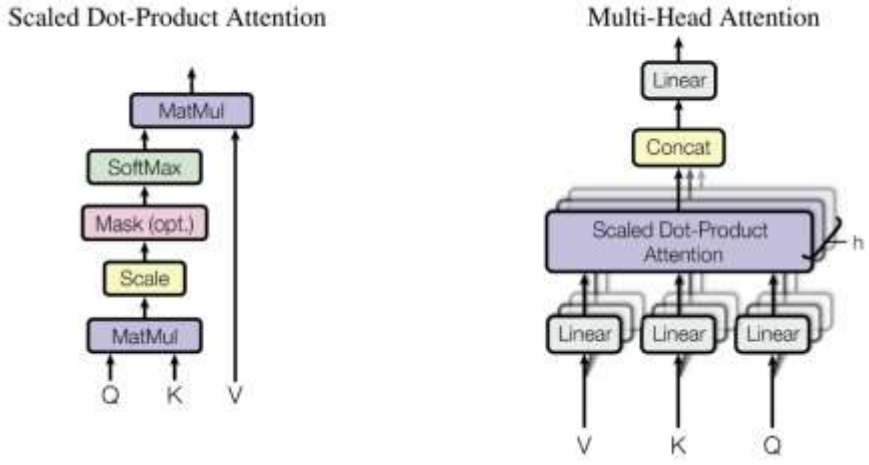

Figure 2: (Left) Scaled dot-product attention. (right) Multi-head attention [2]

2.1.1. 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.

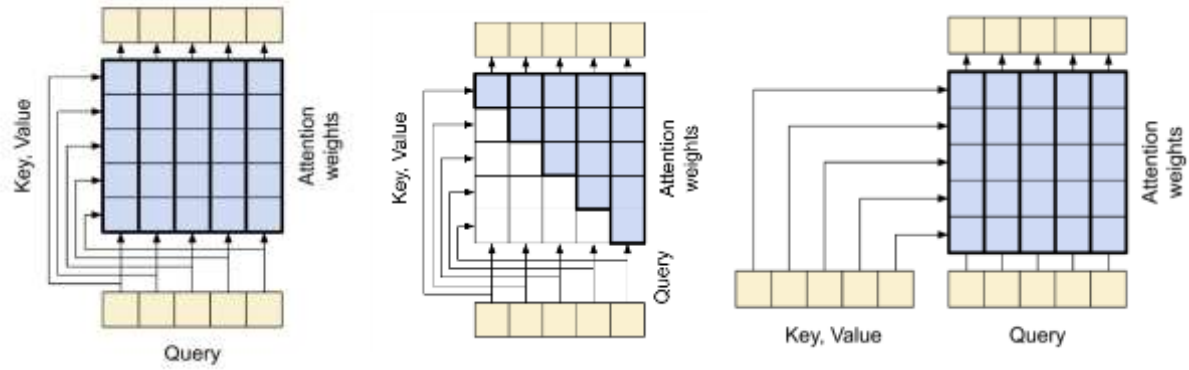

Figure 3: (left) Global Self-Attention Layer, (mid) Causal Attention Layer, and (right) Cross Attention Layer [21]

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

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

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).

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

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

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 returned by the model.

Though the authors in [3] 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 attention layer, which are named (1) global self-attention layer, (2) causal self-attention layer, and (3) cross attention layer, from the encoder - to - decoder respectively [21]. These differences can be seen in Figure 3.

2.1.2. 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:

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

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

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.

2.2. 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 [7]. The improved and better-performing revisions of GPT were later presented in [8] as GPT-2 and [9]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 [3], 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 we limit ourselves to GPT only. The disitlGPT2 model was also used, a distilled version of GPT2, which is discussed in the sub-section below.

2.2.1. 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 Huggingface by the authors of [20]. In [20], they introduced

distilBERT, and the same approach is applied to the creation of distilGPT2. The model is available in HuggingfaceHub.

2.3. 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 [6] as BERT. BERTs' 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 decoder part of the original transformer [3], 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 we limit ourselves to BERT only. However, these other models, mostly, are architecturally identical to BERT. To list some of those models, we have multilingual BERT, DistilBERT [20], RoBERTa [22], DeBERTa [23], XLM-RoBERTa [24], etc. NepaliBERT by [16], and NepBERTa by [17] are

two Nepali-only pre-trained BERT models. In the following subsections, we briefly discuss other BERT-based models which are used in this research.

2.3.1. Multilingual BERT (M-BERT)

Multilingual BERT (M-BERT) is a BERT model which is pre-trained in more than one language. M-BERT was built by authors of [6] along with the BERT model. M-BERT is pre-trained in 104 languages with MLM objective.

2.3.2. DistilBERT

The DistilBERT is a distilled version of BERT which was introduced in [20]. 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 [18].

2.3.3. DeBERTa

DeBERTa stands for Decoding-enhanced BERT with disentangled attention, which was proposed by [23]. 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 [18].

2.3.4. XLM-RoBERTa

XLM-RoBERTa is a multilingual language model. It was proposed in [24]. 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 [24] show that XLM-RoBERTa performs better than multilingual BERT.

2.4. 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 [25]. 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 4.

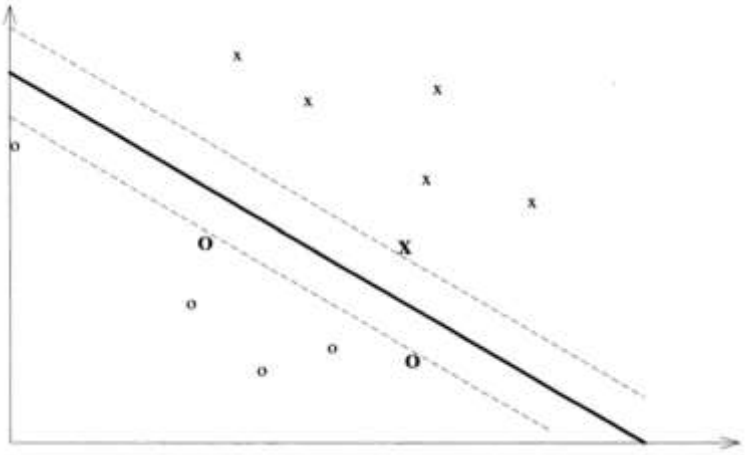

Figure 4: A maximal margin hyperplane with its support vectors highlighted [25]

The optimization problem for SVM can be written as

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

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.

3. 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 [11] 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 [12], 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 [12], 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 [26], 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 [13], 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 [14] and [15]. Also, the authors focus on the betterment of text representation and propose hybrid feature representations; TF-IDF weighted fastText-based method in [14], and hybrid fastText-based and domain-specific methods in [15]. The effectiveness of respective feature representation was tested

using 10 different traditional machine learning algorithms in [14], and a multi-channel CNN approach is proposed in [15].

The highest achieved performance, as reported in [14], 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 [15]. 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.

Table 2: Tokenization of Nepali texts by the BERT tokenizer used in [16].

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

In [27], 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 [28], 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 [29], 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, [29] 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-based based 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 [16] and [17] focuses on training the monolingual BERT language model on a fairly large Nepali corpus, 4GB and 14GB respectively. [18] and [19] builds upon the intuitions from [16] and [17] for Nepali news text classification. [16] 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 [18] pre-trained DistilBERT [20] and DeBERTa [23], 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 [27], 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 [18] 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 [19] go a step further on the use of transformer models for news classification. While [18] used DistilBERT, DeBERTa and two separate pre-trained BERT models, [19] uses BERT, RoBERTa, DistilBERT, DeBERTa, M-BERT, XLM-RoBERTa, and HindiRoBERTa. Along with those models, [19] also uses Bi-LSTM, MNB, RF, and SVM in their study. In their study, [19] 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 [18].

4. Research 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:

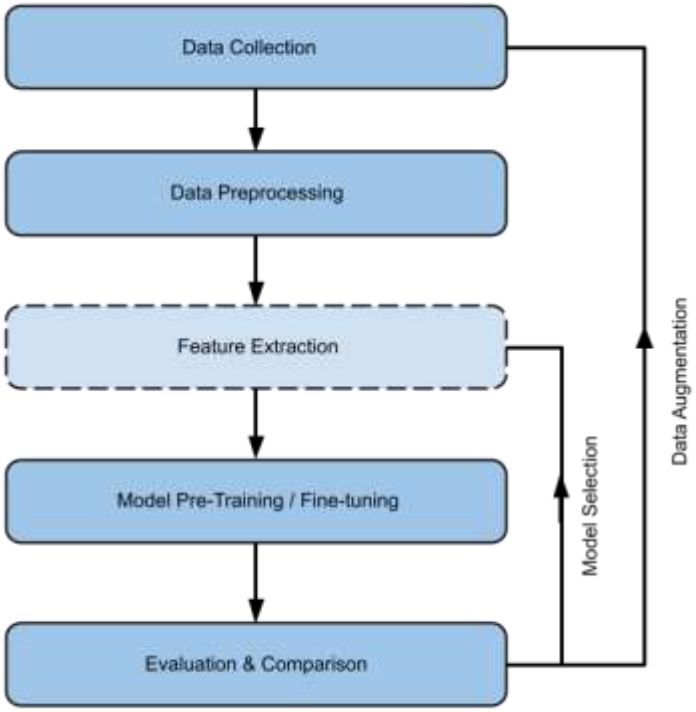

Figure 5: Research Workflow

1. First, we identified a viable dataset for sentiment analysis (NepCov19Tweets). 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 reasons 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 5. The feature selection step, shown in Figure 5, 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.

4.1. 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.

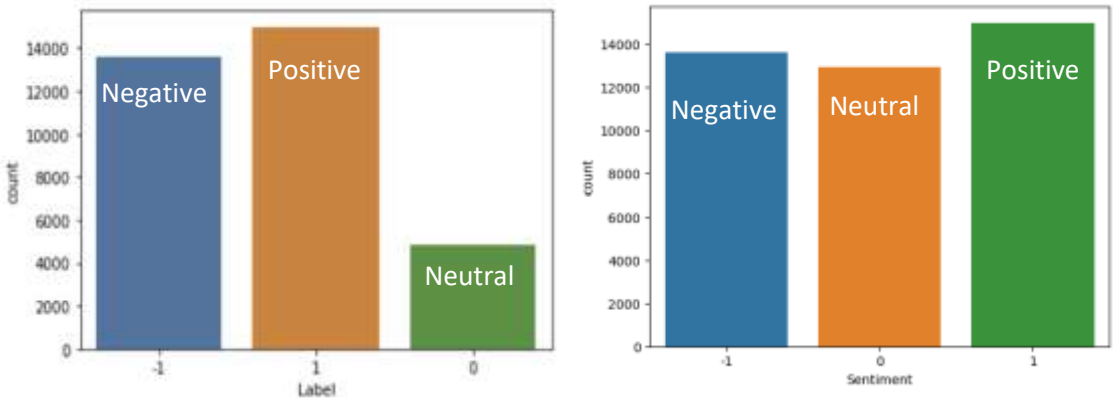

Figure 6: Data distribution per sentiment class. (a) NepCov19Tweets dataset (left) and (b) NepCov19TweetsPlus dataset (right)

i. *Sentiment Analysis*: NepCov19Tweets by [14] was taken as the main dataset for the sentiment analysis task. The dataset was accessed from Kaggle.
ii. *Data Augmentation*: The NepCov19Tweets dataset doesn't have balanced data over all the classes. This can be seen in Figure 6(a). The neutral class (label 0) contains less than half the data of what the other two classes each have.

Firstly, 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. After that, texts were confirmed manually. This augmented data was then added to the NepCov19Tweets dataset. The author of this study has termed the resulting dataset as NepCov19TweetsPlus. The data distribution can be seen in Figure 6.b.

Secondly, for overall augmentation, the Arabic COVID tweets dataset and English COVID tweets dataset were also identified on Kaggle. A portion of data from both dataset were translated into Nepali using Google Translate. This augmented data is tested separately.

iii. *News classification*: The news dataset collected by [28] is used for the news classification task. The intention was to fine-tune the transformer news classifier model to sentiment analysis.

iv. *Pre-Training Corpus:* A combined corpus of different Nepali language corpora was used. The corpus contains NepBERTa corpus [17], OSCAR corpus [30], CC-100 [24], News classification dataset [28], 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 HuggingfaceHub under dataset id raygx/Nepali-Extended-Text-Corpus.

4.2. 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 (less than 10 words) from 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.

4.3. Model Architecture

As mentioned in sections 2.2 and 2.3, The transformer models used in this thesis study are nepaliBERT [16], NepBERTa [17], distilBERT (say *NepDistilBERT*), and DeBERTa (say *NepDeBERTa*) [18], multilingual BERT [6], XLM-RoBERTa [24]. 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.

Table 3: Tokenization By Custom BPE Tokenizer

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

4.4. 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.

4.5. Model Pre-training and Fine-tuning

As mentioned in section 4.3, 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 4.2 were fine-tuned for sentiment analysis on the NepCov19TweetsPlus dataset. Only selective models were fine-tuned on NepCov19Tweets and other data-augmented dataset.

For model initialization and training, Huggingface, Tensorflow, and Scikit library were utilized.

4.6. 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.

To evaluate, calculate performance metrics, and visualize data, various libraries such as Scikit-Learn, Matplotlib, Seaborn, and Tensorflow were utilized.

5. Implementation

In this research, the comparative analysis of transformer based models for sentiment analysis of Nepali social media texts have been carried out. The objective was to research for method to maximize the models capacity to classify sentiments. For training and testing of these models different tools, platforms and libraries have been used.

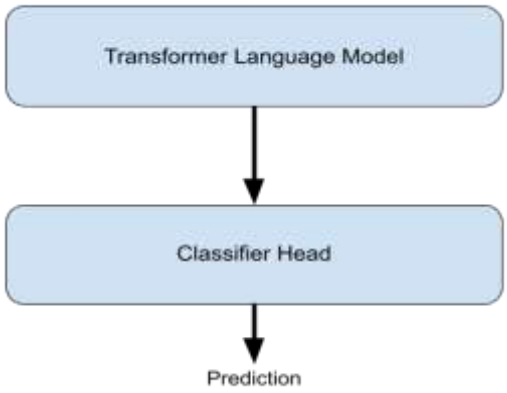

Figure 7: general organization of transformer based sentiment analysis model

5.1. Implementation tools, platforms and libraries

1. Python3 programming language was used.
2. Huggingface transformer library was used for the transformer model training and testing. All the models used from transformer library were based on Tensorflow backend. Tensorflow was used for computing confusion matrix and implementing feature extraction module.
3. Dataset library and tokenizer library form Huggingface is used for data preparation and tokenization.
4. Scikit-learn was used for the SVM model training and testing. 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. HuggingfaceHub and Kaggle dataset were used as data repositories.
7. The compute 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.

5.2. Implementation Details

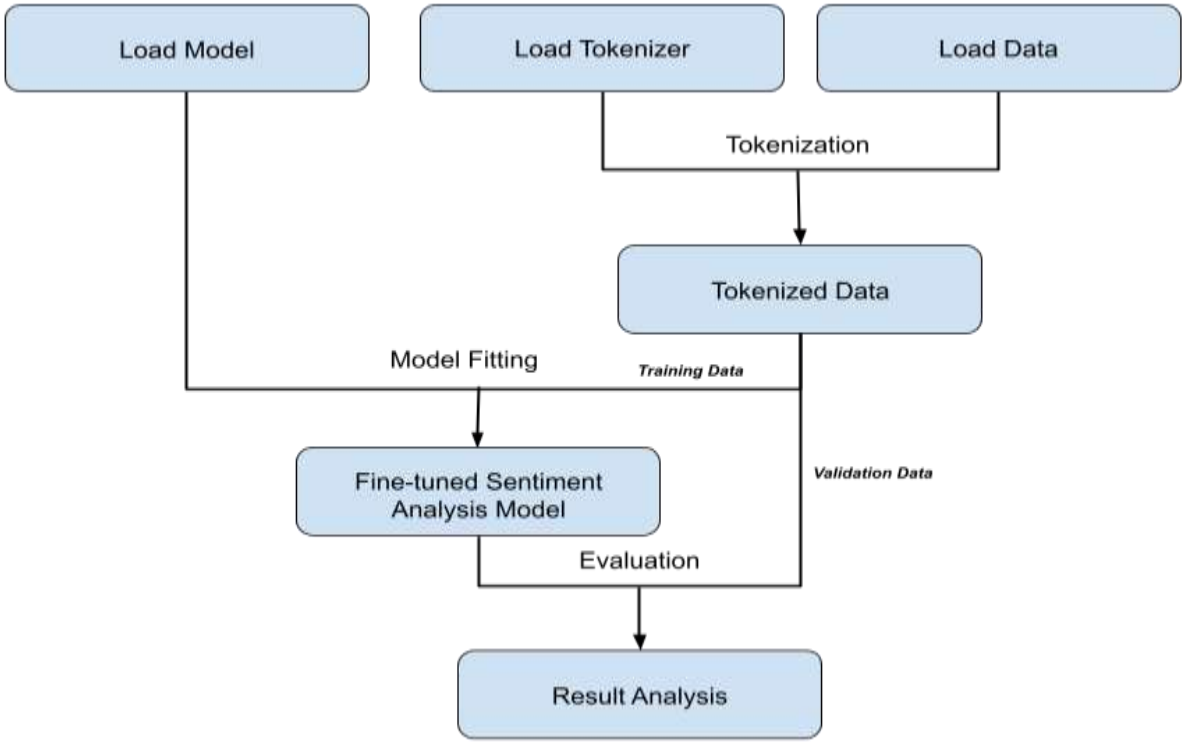

Figure 8: Model Fine-tuning for sentiment analysis workflow

5.2.1. Transformer based Sentiment Analysis

The training and testing of transformer based sentiment analysis models, be it BERT-based or GPT based, almost follows the same pattern. This is due to the generality of classification architecture as implemented in the Huggingface transformers library, which can be seen in Figure 7. 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 4.3.
ii. The classifier head. It is a single dense NN layer that contains neurons equal to the number of sentiment class.

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

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

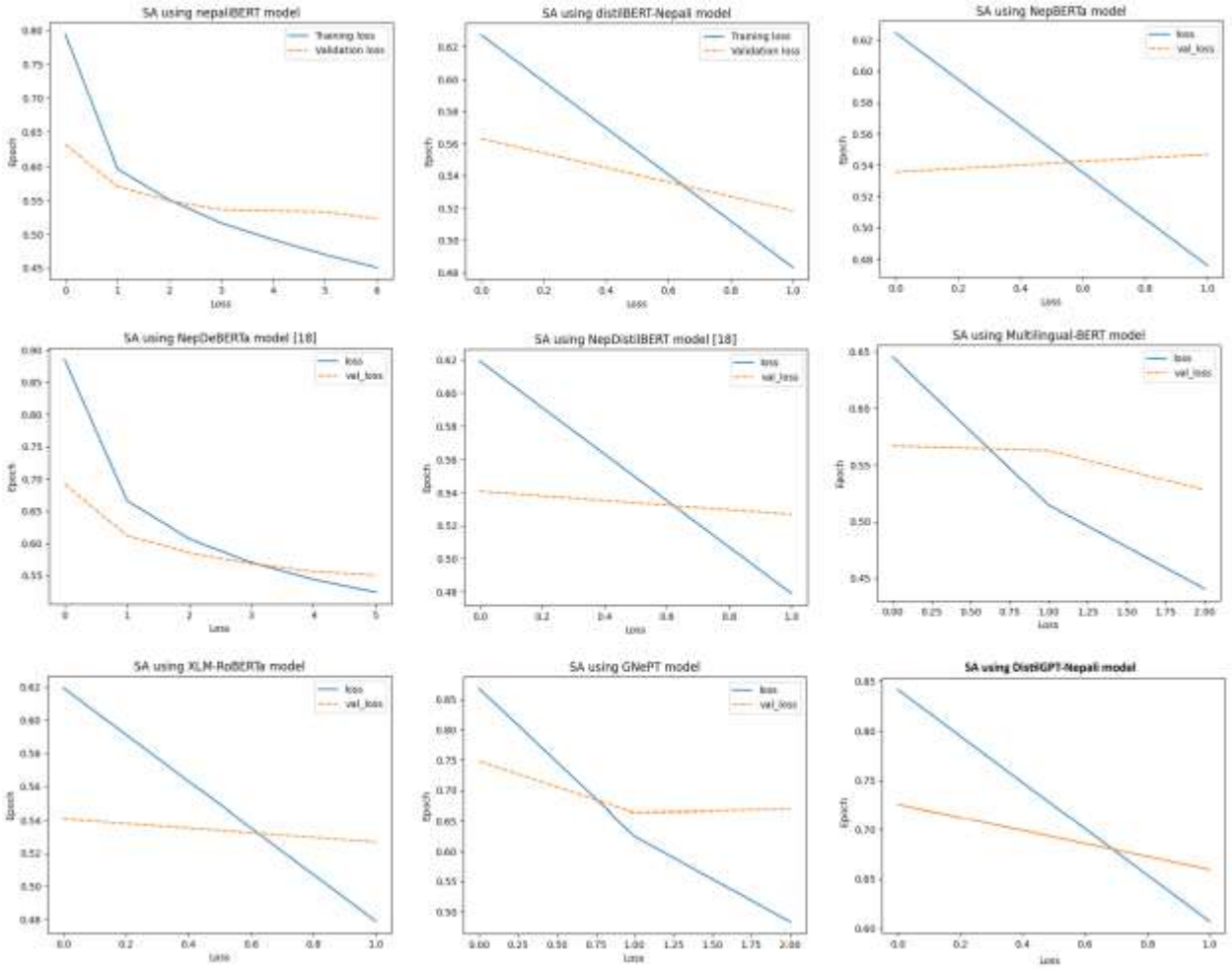

Figure 9: Training loss / Validation loss of Fine-tuning different transformer models for sentiment analysis on NepCov19TweetsPlus dataset

This workflow is depicted in Figure 8. The work flow is quite similar for all of 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 model performed their best with different hyper-parameters. In Figure 9, it can be seen that different models were trained for different number of epochs. The training instance shown in Figure 9 are the ones when the model gave its optimal performance.

In tokenization, padding strategy is different for BERT-based models and GPT-based models. BERT-based models use first token output as a feature for downstream task which is subjected to pooling layer [6], so the padding is appended to the text. In contrast, GPT-based models use last token output as a feature for downstream task, this is because of causal nature of decoder models, hence the padding is prepended to the text. This is important distinction to remember, especially in case of GPT-based models, because if padding is appended to the text then by the time model processes last token, it mostly observed padding token. Thus the model can't make distinction between different texts which have padding appended to it.

5.2.1.1.Hyper-parameters

This study doesn't make any changes to internal standard parameters of the transformer model like number of layers, dimension of model, hidden size, number of attention heads, dropout probabilities, etc. for most of the sentiment analysis task. However, a separate research in this study was done with reduced number of layers and increased number of attention heads in BERT model. This will be later discussed in analysis section 0. 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 training and validation set. The choice of batch-size was limited by the GPU memory available in the compute 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, learning rate from 1e-6 to 9e-6 were also used.

(iv) *Weight decay rate*: It is a regularization parameter of optimizer function. It penalizes model weights reducing the chances of model overfitting. In 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 rate ranging from 0.001 to 0.009 were also used.

5.2.2. Transformer Language Model Pre-training

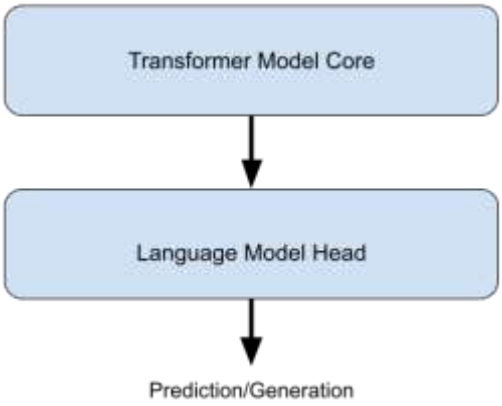

Figure 10: general organization of transformer language model

This research study had conducted the pre-training of three language models, two GPT2-based models and one BERT-based model. Language model consists of two components: one is transformer core and second is language model head as seen in Figure 10. The model core is either the decoder-block or encoder-block (depending on the model used) as seen in Figure 3. The language model head is the dense layer which makes prediction regarding 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.

5.2.2.1. 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 4.1. Data contained approximately 441 million tokens after tokenization and sequence of 512 tokens was used for training. The model is trained on Kaggle notebook environment using GPU P100 accelerator. The *AdamWeightDecay* optimizer was used and 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 by tensorflow documentation is recommended policy when using GPU. The masking probability was set to 25%, which is 10% higher than what was used by [6]. 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 11. This resulted in 17.31 model perplexity. The model has approximately 82 million parameters. 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

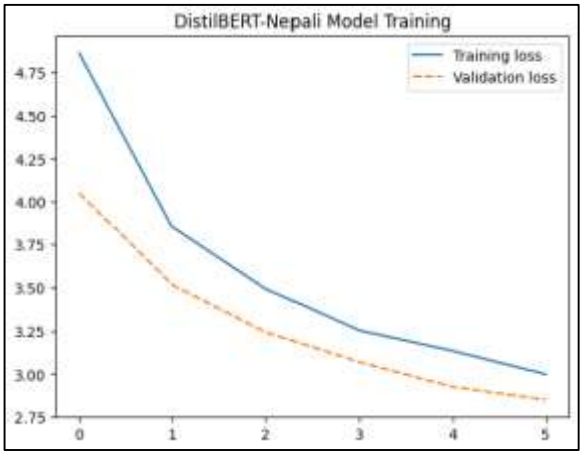

Figure 11: Training / Validation loss of Pre-trianing DistilBERT-Nepali.

5.2.2.2. 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, a sequence of 512 tokens were used. The compute platform used to train this model was 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 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 by tensorflow documentation is recommended policy when using TPU. 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 12. This resulted in 84.05 model perplexity. The model has approximately 82 million parameters. Model's internal parameters are

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

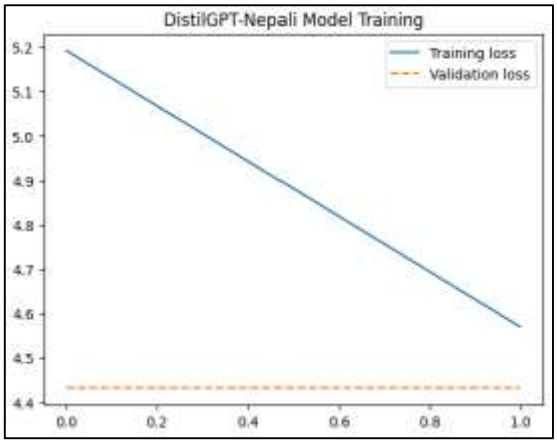

Figure 12: Training / Validation loss of Pre-trianing DistilGPT-Nepali.

5.2.2.3. 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, a sequence of 512 tokens were used. The compute platform used to train this model was 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. Also, tensorflow mixed precision was set to *mixed_bfloat16*, which by tensorflow documentation is recommended policy when using TPU. The optimizer used was *AdamWeightDecay*. The training of the model was done in two steps. In first step, the model was trained for two epochs with optimizer parameter learning rate and weight decay rate set to 2×10^{-5} and 0.035 respectively. The model attained training loss of 4.40 and validation loss of 4.27. In second step, the model was trained for single epoch with 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 13. This final

model perplexity was 66.59. The model has approximately 124.24 million parameters. Model's internal parameters are

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

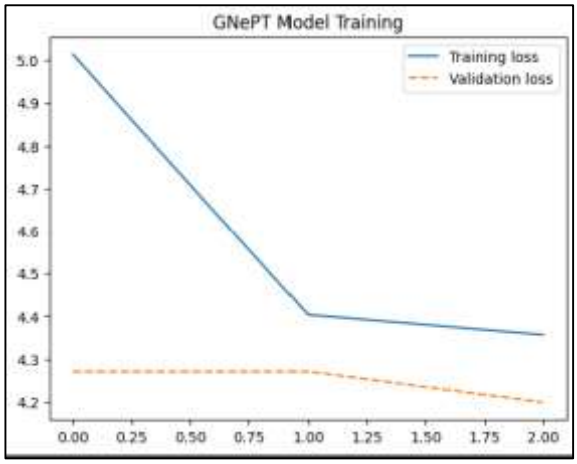

Figure 13: Training / Validation loss of Pre-trianing GNePT

5.2.3. Hybrid model based Sentiment Analysis

This study has taken two hybrid model based approach in consideration, (i) Using SVM model as classifier and (ii) Using MLP model as a classifier. In both approach, the transformer model's output embedding is used as a feature to train and evaluate the classifier model. In case of MLP model, it can be considered as pooling layer within the transformer sentiment analysis model. The embedding used is the first token embedding in case of BERT based models and last token embedding in case of GPT based models. All BERT based models, except for DistilBERT, has 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.

The overall working is similar to what is shown in Figure 8, except for the feature extraction which needs to be done separately in hybrid model approach. This is portrayed in Figure 14.

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

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 case of 2 model feature hybrid approach, the embedding size used to train MLP is 1536.

In 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.

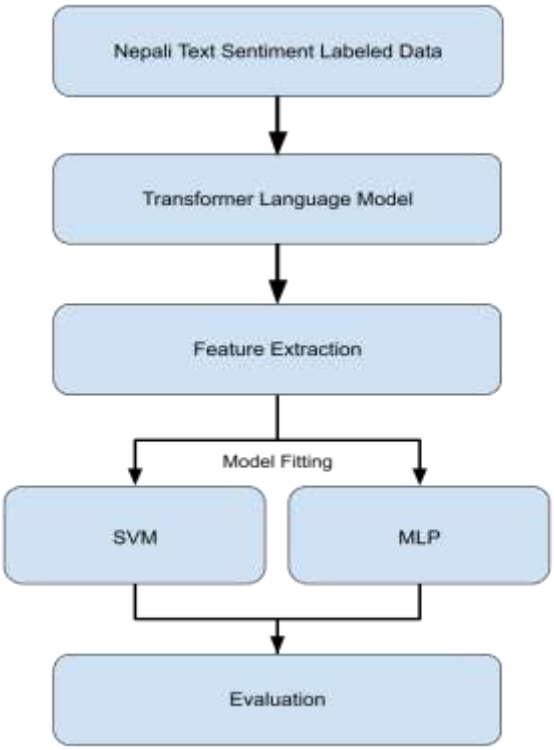

Figure 14: Workflow of Hybrid model based sentiment analysis approach

6. Result and Analysis

7. Conclusion

8. 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] 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.

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

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

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

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

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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].

[20] 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.

[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] 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].

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

[24] 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.

[25] N. Cristianini and J. S. Taylor, An Introduction to Support Vector Machines and Other Kernel- based Learning Methods, Cambridge University Press , 2002, p. pp 126.

[26] 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.

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

[28] 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.

[29] 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].