(An undertaking of Bhaktapur Municipality)

Khwopa College of Engineering

Affiliated to Tribhuvan University

Libali-08, Bhaktapur, Nepal

A

PROPOSAL ON

"Embedding in Nepali Language"

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

Submitted by:

Manish Pyakurel (KCE077BCT020)

Rupak Neupane (KCE077BCT028)

Sarjyant Shrestha (KCE077BCT033)

Srijan Gyawali (KCE077BCT036)

Submitted to:

Department of Computer Engineering

Khwopa College of Engineering

Bhaktapur,Nepal

May, 2024

Abstract

In recent years, Natural Language Processing (NLP) has made significant strides, particularly in the development of word embeddings that capture both semantic and syntactic meanings of words. This proposal focuses on creating word embeddings for the Nepali language, which remains underrepresented in the realm of NLP due to its complex grammatical structure and rich character set. Despite the progress in NLP, low-resource languages like Nepali face challenges in data collection and model training. This study aims to address these challenges by leveraging pre-trained models and fine-tuning them with a substantial Nepali corpus. The proposed system will utilize transformer-based models, such as BERT, to generate contextualized word embeddings that can be applied to various NLP tasks, including sentiment analysis, machine translation, and question answering. By advancing NLP technologies for the Nepali language, we aim to enhance digital accessibility and empower communities through improved communication and educational tools.

Keywords: *Word Embeddings, Nepali Language, Natural Language Processing, BERT, Transformer Models, Contextualized Embeddings, Low-Resource Languages.*

Contents

List of Figures

List of Symbols and Abbreviation

AI	Aritificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
TF-IDF	Term Frequency-Inverse Document Frequency
XLM	Cross Lingual Language Model

CHAPTER 1

Introduction

1.1 Background Introduction

NLP is a branch of linguitics, computer science, and artificial intelligence concerned with computer human interaction, mainly how to design computers to process and evaluate huge volumes of natural language data [**?**]. Pre-training of an NLP model plays an essential role in transfer learning, where a language model will be trained on a vast corpus set and later fine-tune the model for a specific purpose [**?**]. Word embedding is a fundamental concept in NLP. It is a real-valued vector representation of words by embedding both semantic and syntactic meanings obtained from unlabeled large corpus [**?**]. It is of n-dimensional distributed representation of a text that attempts to capture the meanings of the words [**?**]. Word embeddings can be obtained using language modeling and feature learning techniques, where words or phrases from the vocabulary are mapped to vectors of real numbers [**?**]. Pre-trained word embeddings encode general word semantics and lexical regularities of natural language, and have proven useful across many NLP tasks, including word sense disambiguation, machine translation, and sentiment analysis, to name a few [**?**]. Word embeddings have been found to be very useful for many NLP tasks, including but not limited to Chunking [**?**], Question Answering [**?**], Parsing and Sentiment Analysis [**?**]. [**?**]

Types of Word Embedding Techniques [**?**]

Traditional Embeddings: Traditional word embeddings represent words as fixed vectors in an n-dimensional space, capturing semantic relationships between words. These embeddings are static and do not change based on context or training data.

Static Embeddings: Static word embeddings are pre-trained on a large corpus of text and do not change during model training. They are useful for tasks where word meanings remain constant across different contexts.

Contextualized Embeddings: Contextualized word embeddings, like BERT, are based on transformer models that can capture word meanings based on the context in

which they appear. These embeddings provide more accurate representations of words by considering the surrounding context during training.

Combined Word Embedding and Neural Network Models: Combining word embeddings with neural network models can enhance model accuracy in various natural language processing tasks such as sentiment classification, text categorization, and phrase prediction.

Nepali is one of the languages that uses Devanagari, a script used in many languages spoken in Asian countries. It is spoken by more than 20 million people, mainly in Nepal, and other places in the world including Bhutan, India and Myanmar [**?**]. It has been rarely used for Natural Language Processing services. Nepali can be quite complex due to its many sounds, grammar rules, and ways to change words. Due to its complex grammatical structure and rich characters, extracting fruitful information from the corpus has been challenging [**?**].

The advancement of NLP technologies adapted to individual languages, like Nepali, hold immense potential for empowering communities and enhancing accessibility to digital resources for Nepali language. By filling the gap between technological innovation and linguistic diversity, we can unlock new possibilities for communication and education.

1.2 Problem Statement

Even though Word Embeddings can be directly learned from raw texts in an unsupervised fashion, gathering a large amount of data for its training remains a huge challenge in itself for a low-resource language such as Nepali [**?**]. Despite having breakthroughs in the field of NLP, productive results with the Nepali language have not been achieved. One of the primary reasons for this is the need for more computational resources [**?**]. As mentioned in the most recent study in this topic (i.e NepaliBERT [**?**]), there is a lack of larger, more diverse, and context-rich dataset to enhance the accuracy and robustness of the word embeddings in Nepali language. This research study seeks to construct a more finely tuned model capable of generating embeddings for the Nepali corpus. It is seen that there is reduction of perplexity by using XLM.

1.3 Objective

The main aim of this project is:

- To develop context dependent word embedding for Nepali language.

CHAPTER 2

Literature Review

Most of the research that has been undertaken on the Nepali corpus was focusing on generating embeddings through traditional approaches like TFIDF, Wod2Vec and other embedding methods. [?] [?] [?] and [?] implemented TF-IDF on Nepali text for text classification and other purposes such as sentiment analysis. Similarly, Word2Vec approach in nepali corpus was implemented by [?] [?] and [?]. 300-Dimensional Word Embeddings for Nepali Language [?] has pre-trained Word2Vec model having 300-dimensional vectors for more than 0.5 million Nepali words and phrases. The embeddings generated using the methods described above are static, implying that each word retains only one vector representation regardless of its context of use. However, contemporary trends emphasize the adoption of contextual-dependent embeddings over their contextual-independent counterparts. As highlighted earlier, there have been limited studies on BERT within the Nepali context. [?] claimed to provide an efficient Nepali BERT embedding, but despite having a huge dataset they were short of computational resources due to which they had to compromise on the different BERT parameters. They modified the BERT model by averaging the hidden states from the last two hidden layers to get the embeddings, whereas, for getting the baseline results, instead of using any pre-trained word vectors, a trainable Keras embedding layer was used in front of the architecture mentioned above which automatically learns the word embeddings by only using the provided training examples. [?] and [?] also tried the capacity of BERT for cross-lingual in Natural Language Processing.

There are also studies done in XLM [?]. The paper compares a Nepali language model with a cross-lingual language model trained in Nepali but enriched with different combinations of Hindi and English data, showing how leveraging data from related languages can benefit low-resource languages like Nepali. 300-Dimensional Word Embeddings for Nepali Language [?] has pre-trained Word2Vec model having 300-dimensional vectors for more than 0.5 million Nepali words and phrases. The embeddings generated using the methods described above are static, implying that

each word retains only one vector representation regardless of its context of use. However, contemporary trends emphasize the adoption of contextual-dependent embeddings over their contextual-independent counterparts. As highlighted earlier, there have been limited studies on BERT within the Nepali context. [**?**] claimed to provide an efficient Nepali BERT embedding, but despite having a huge dataset they were short of computational resources due to which they had to compromise on the different BERT parameters. They modified the BERT model by averaging the hidden states from the last two hidden layers to get the embeddings, whereas, for getting the baseline results, instead of using any pre-trained word vectors, a trainable Keras embedding layer was used in front of the architecture mentioned above which automatically learns the word embeddings by only using the provided training examples. [**?**] and [**?**] also tried the capacity of BERT for cross-lingual in Natural Language Processing.

There are also studies done in XLM [**?**]. The paper compares a Nepali language model with a cross-lingual language model trained in Nepali but enriched with different combinations of Hindi and English data, showing how leveraging data from related languages can benefit low-resource languages like Nepali.

CHAPTER 3

Theoretical Background

3.1 Transformers

A transformer model is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence. Transformer models apply an evolving set of mathematical techniques, called attention or self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other. First described in a 2017 paper from Google [**?**], transformers are among the newest and one of the most powerful classes of models invented to date. They're driving a wave of advances in machine learning some have dubbed transformer AI. Stanford researchers called transformers "foundation models" in an August 2021 paper [**?**] because they see them driving a paradigm shift in AI.

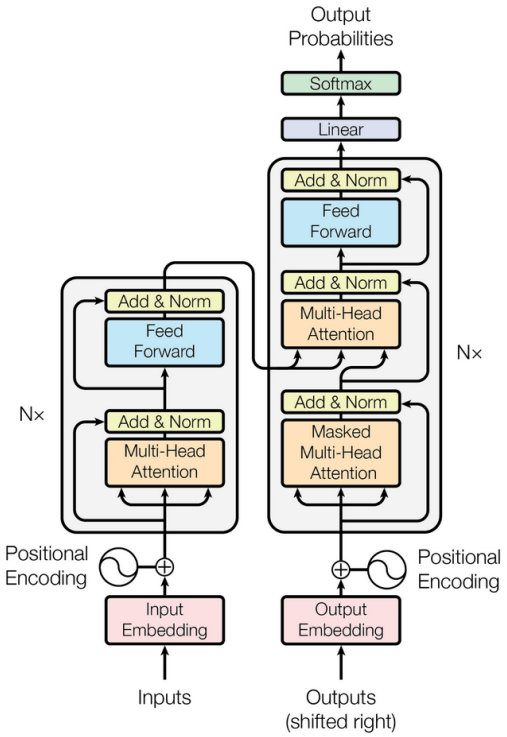

Figure 3.1: The Transformer - model architecture. [**?**]

3.2 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context [?]. BERT utilizes the Transformer architecture, which employs an attention mechanism to understand contextual relationships among words or sub-words within a text. The basic Transformer structure consists of two distinct components: an encoder, which processes the input text, and a decoder, which generates predictions for the given task. BERT architecture enables it to handle various NLP tasks effectively, such as question answering, sentiment analysis, and text classification, by fine-tuning on specific datasets.

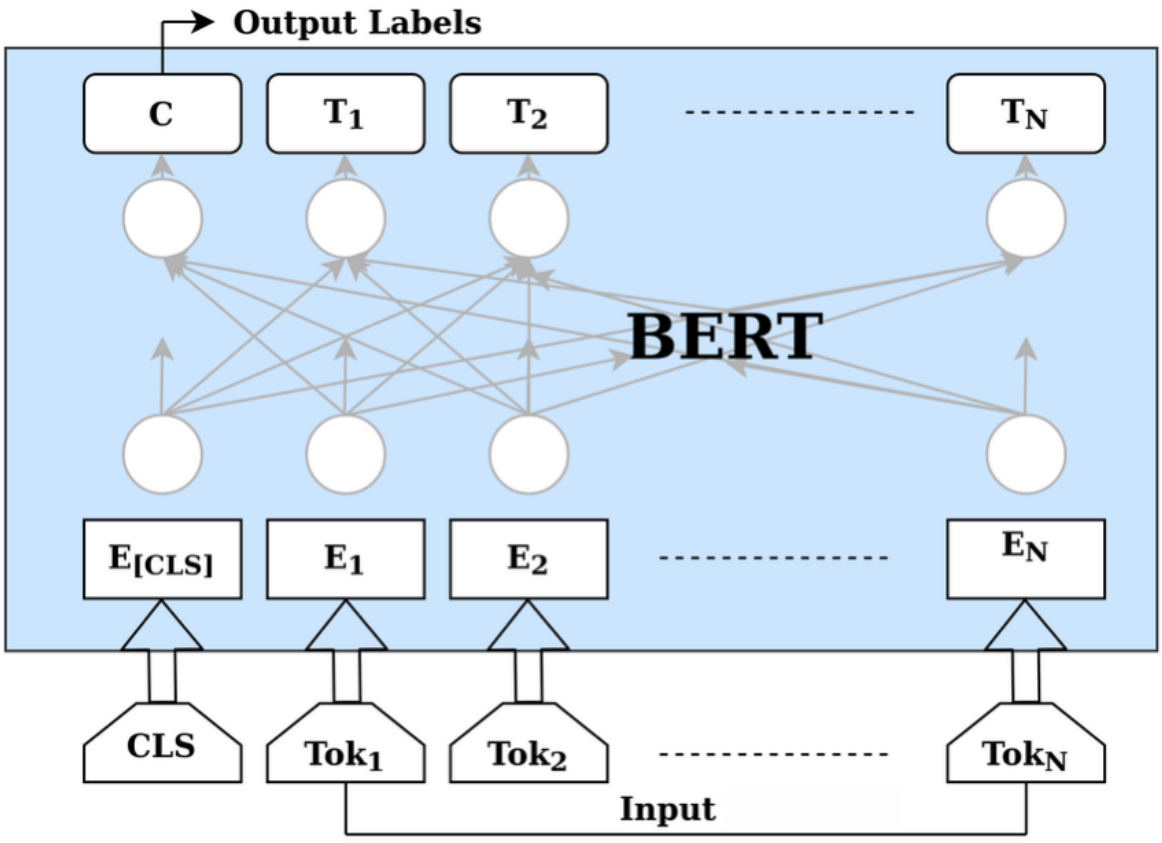

Figure 3.2: Illustrations of Fine-tuning BERT on Different Tasks. [?]

CHAPTER 4

Methodology

4.1 Software Development Approach

Agile development is a software development approach that emphasizes incremental progress and rapid cycles. It involves releasing small increments of functionality that build upon previous versions. Thorough testing is conducted for each release to ensure software quality. Agile is often employed for time-critical applications. Although this project is not time-critical this model seems to be the most optimal and practical in our case.

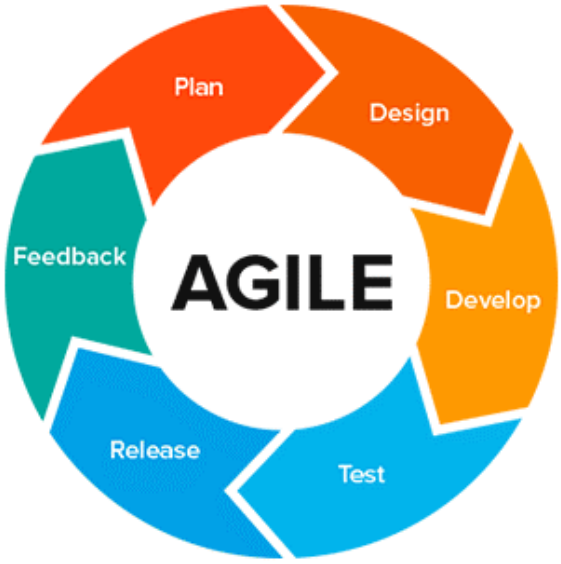

Figure 4.1: Agile Model for Software Development

source: https://mobilelive.medium.com/agile-development-a-comprehensive-guide-for-the-modern-era-d2fe9ae7b395

4.2 Proposed System Block Diagram

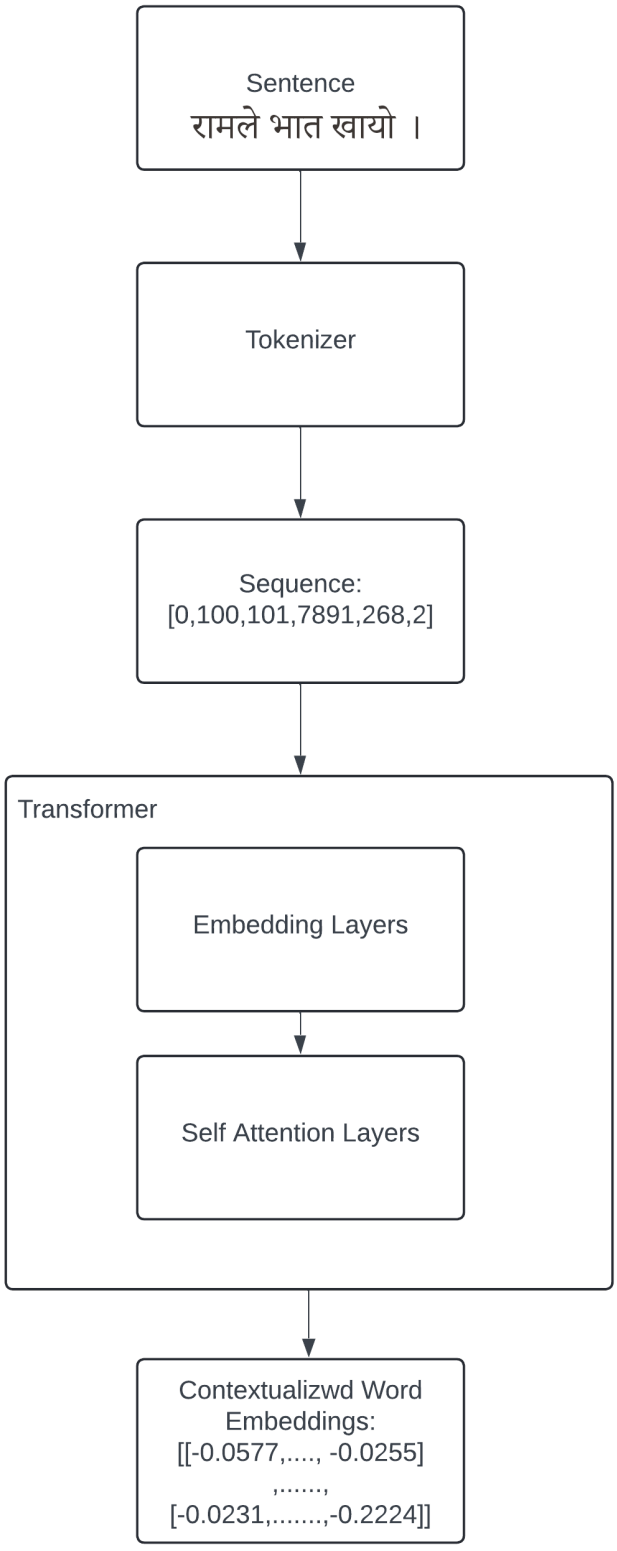

Figure 4.2: Block diagram of Proposed Sytem

4.3 Description of Working Flow of Proposed System

1. **Sentence Input**:

   - The process starts with a given sentence in the Nepali language: रामले भात खायो

2. **Tokenizer**:

   - The sentence is passed to a tokenizer, which breaks down the sentence into individual tokens. The tokens are then converted into a sequence of numerical IDs. For example, the sentence रामले भात खायो is converted into the sequence `[0, 100, 101, 7891, 268, 2]`.

3. **Transformer Model**:

   - The tokenized sequence is then fed into a Transformer model, which consists of several layers. This model processes the sequence to generate contextualized word embeddings.

     – **Embedding Layers**:

       * The first part of the Transformer model is the embedding layers. These layers convert the input token IDs into dense vectors of fixed size. These vectors capture semantic information about the words.

     – **Self-Attention Layers**:

       * After the embedding layers, the vectors pass through multiple self-attention layers. These layers allow the model to weigh the importance of different words in the sentence relative to each other, thereby capturing the context of each word in relation to the entire sentence.

4. **Contextualized Word Embeddings Output**:

   - The final output of the Transformer model is a set of contextualized word embeddings. Each word in the input sentence is now represented by a vector that encodes both its meaning and its context within the sentence.

For example, the embeddings might look like: [[-0.0577, ..., -0.0255], ..., [-0.0231, ..., -0.2224]].

CHAPTER 5

Work Completed

5.1 Dataset Preparation

- **Data Loading:** The project uses the *Kantipur* dataset, which consists of Nepali news articles. This data is loaded from a cleaned and combined TSV file. The data is organized into two columns: one for the *title* of the article and another for the *news content*.

- **Text Processing:** The raw Nepali text from the dataset is processed for tokenization. The custom tokenizer ensures that the text is cleaned and normalized, addressing issues like extra spaces and Unicode variations in Nepali characters.

5.2 Tokenizer Development

- **NepaliTokenizer:** A specialized tokenizer was implemented for Nepali text, called `NepaliTokenizer`. This class was designed to handle the unique features of the Nepali language, including:

  - *Character Ranges:* It checks whether a character falls within the Devanagari script (the script used for Nepali) and processes it accordingly.
  - *Punctuation Handling:* It accounts for Nepali punctuation marks (।, ॥, and others) and ensures they are treated as distinct tokens during tokenization.
  - *Suffix Handling:* Common Nepali suffixes and postpositions (like ले, को, बाट) are identified and separated as individual tokens to preserve their semantic roles in the language.

- **CustomBERTTokenizer:** A more advanced tokenizer, `CustomBERTTokenizer`, was built to handle:

  - *Vocabulary Building:* This tokenizer automatically builds a vocabulary of frequent words and tokens based on the input dataset. It dynamically

updates the vocabulary while ensuring it does not exceed the predefined maximum size (30,000 tokens).

- *Token Masking:* During pretraining, certain tokens are randomly masked (`[MASK]`) to create a Masked Language Modeling (MLM) task, which is a core part of training BERT models.
- *Encoding:* The tokenizer converts raw text into token IDs, adding special tokens such as `[CLS]` (for classification), `[SEP]` (to separate segments), and `[PAD]` (for padding sequences to a uniform length).

5.3 Dataset Processing

- **Data Preparation for Pretraining:** The dataset is prepared specifically for BERT pretraining, where the model learns to predict masked tokens (a form of unsupervised learning). For each article in the dataset:
  - *Input Sequences:* The raw text is encoded into a sequence of token IDs.
  - *Segment IDs:* Segment IDs are created, marking which tokens belong to which segment.
  - *Masked Tokens:* For the MLM task, some tokens in the sequence are randomly masked, and the model is trained to predict these missing words. The masked tokens are tracked for loss calculation during training.

5.4 Model Architecture

- **BERT Embedding Layer:** A custom embedding layer was built that includes:
  - *Token Embeddings:* Maps each token in the vocabulary to a dense vector.
  - *Segment Embeddings:* Differentiates between multiple segments (useful for tasks like question answering).
  - *Positional Embeddings:* Positional embeddings are added to provide information about the order of tokens in the sequence.
  - *Dropout:* A dropout layer is included to prevent overfitting during training.

- **Transformer Encoder:** The core of the BERT model is the Transformer Encoder, which processes sequences of token embeddings using attention mechanisms to capture relationships between tokens at various positions in the input sequence.
- **Pretraining Head (MLM):** The MLM head predicts the original tokens that were masked during pretraining. This is achieved by using a linear layer over the final hidden states produced by the encoder.

5.5 Training Loop

- **Training Workflow:** The model is trained using a cross-entropy loss function, which measures how well the predicted token IDs match the original token IDs. The optimizer (Adam) updates the model parameters to minimize this loss. The training loop runs over multiple epochs.
- **Testing Workflow:** The model's performance is evaluated on the test set after each epoch. The test loss is calculated using the same MLM loss function.
- **Loss Tracking:** Loss values for both the training and testing datasets are recorded after each epoch, and these are saved into text files for further analysis.

5.6 Data Splitting and Loader Implementation

- **Training and Test Split:** The dataset is divided into training and testing sets, with 80% used for training and 20% for testing.
- **DataLoader Implementation:** DataLoaders are used to manage batching, shuffling, and pinning memory for efficient data loading during training and testing.

5.7 Performance Metrics

Loss during training and testing is recorded and plotted against epochs to monitor performance improvements. A plot showing the loss values over epochs is generated, allowing us to monitor the model's training progress. Outputs:

5.8 Pending Tasks

- Evaluate the trained BERT model on downstream Nepali NLP tasks.
- Fine-tune the tokenizer and training hyperparameters for improved performance.
- Explore potential model optimizations, such as reducing memory overhead for larger datasets.