(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 [3]. 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 [4]. 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 [5]. It is of n-dimensional distributed representation of a text that attempts to capture the meanings of the words [3]. 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 [6]. 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 [7]. Word embeddings have been found to be very useful for many NLP tasks, including but not limited to Chunking [8], Question Answering [9], Parsing and Sentiment Analysis [10]. [11]

Types of Word Embedding Techniques [12]

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 [13]. 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 [4].

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 [14]. 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 [4]. As mentioned in the most recent study in this topic (i.e NepaliBERT [4]), 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. [15] [16] [17] and [18] 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 [15] [19] and [20]. 300-Dimensional Word Embeddings for Nepali Language [21] 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. [14] 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. [22] and [23] also tried the capacity of BERT for cross-lingual in Natural Language Processing.

There are also studies done in XLM [24]. 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 [21] has pre-trained Word2Vec model having 300-dimensional vectors for more than 0.5 million Nepali words and phrases. The embed-

dings 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. [14] 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. [22] and [23] also tried the capacity of BERT for cross-lingual in Natural Language Processing.

There are also studies done in XLM [24]. 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 [1], 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 [25] because they see them driving a paradigm shift in AI.

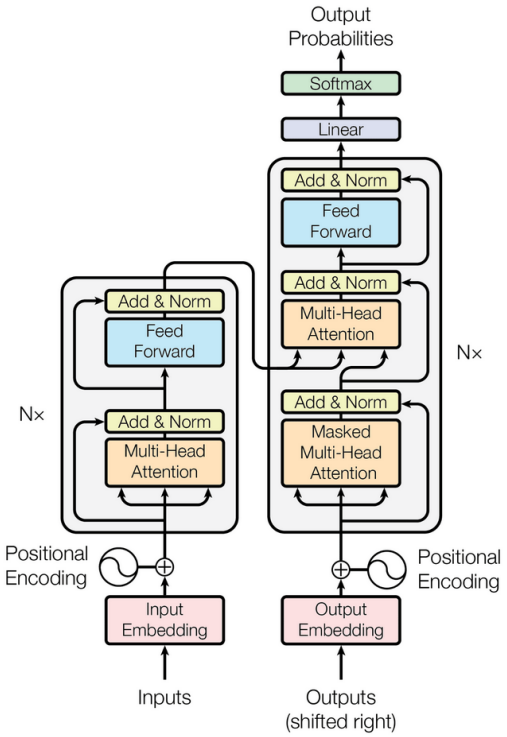

Figure 3.1: The Transformer - model architecture. [1]

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 [26]. 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.

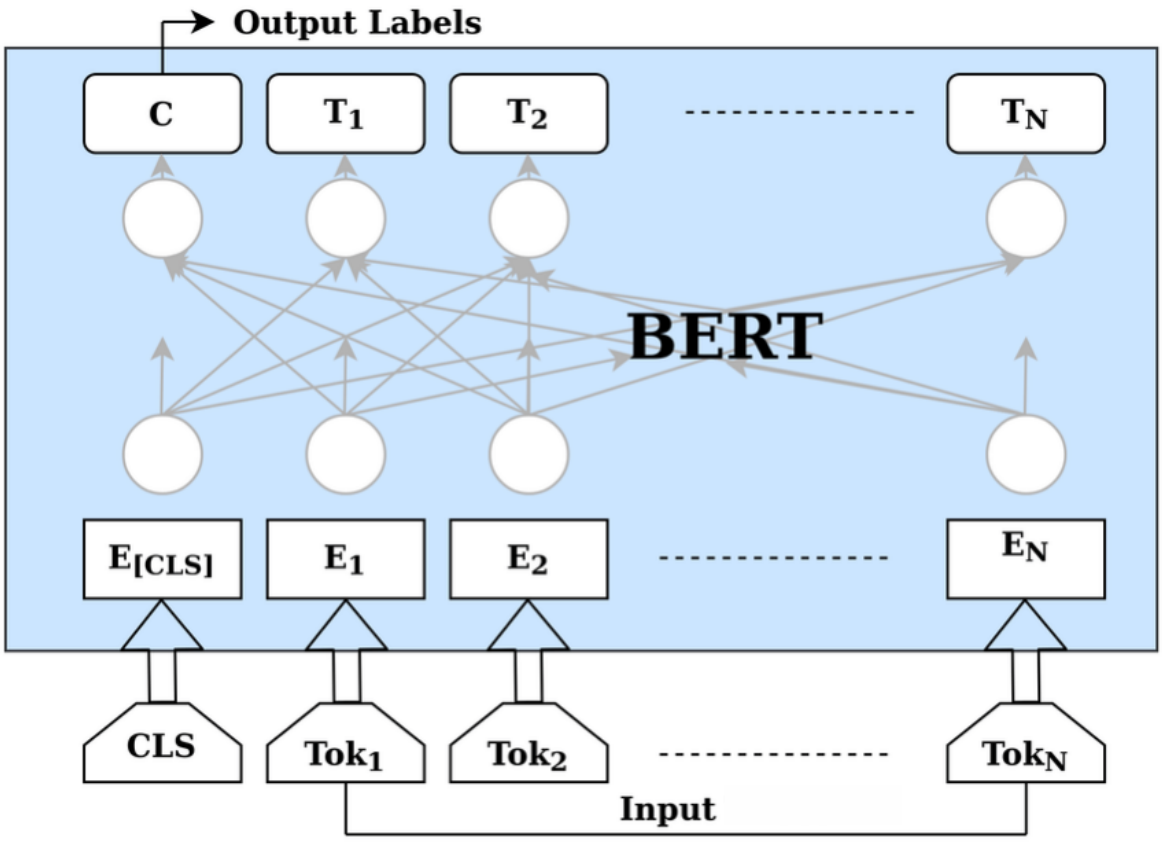

Figure 3.2: Illustrations of Fine-tuning BERT on Different Tasks. [2]

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.

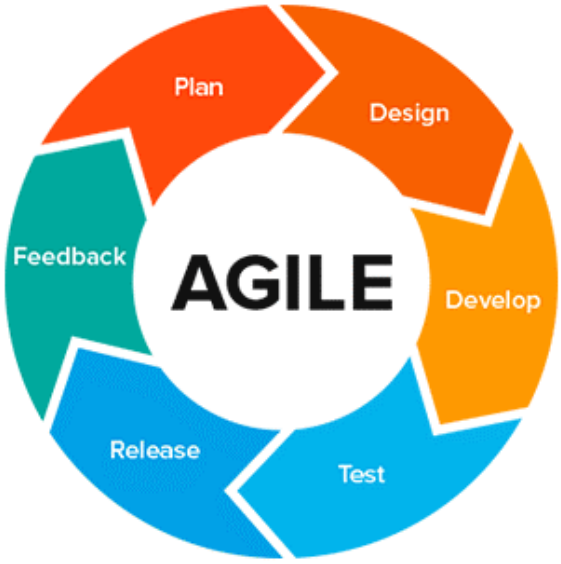

Figure 4.1: Agile Model for Software Development

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

4.2 Proposed System Block Diagram

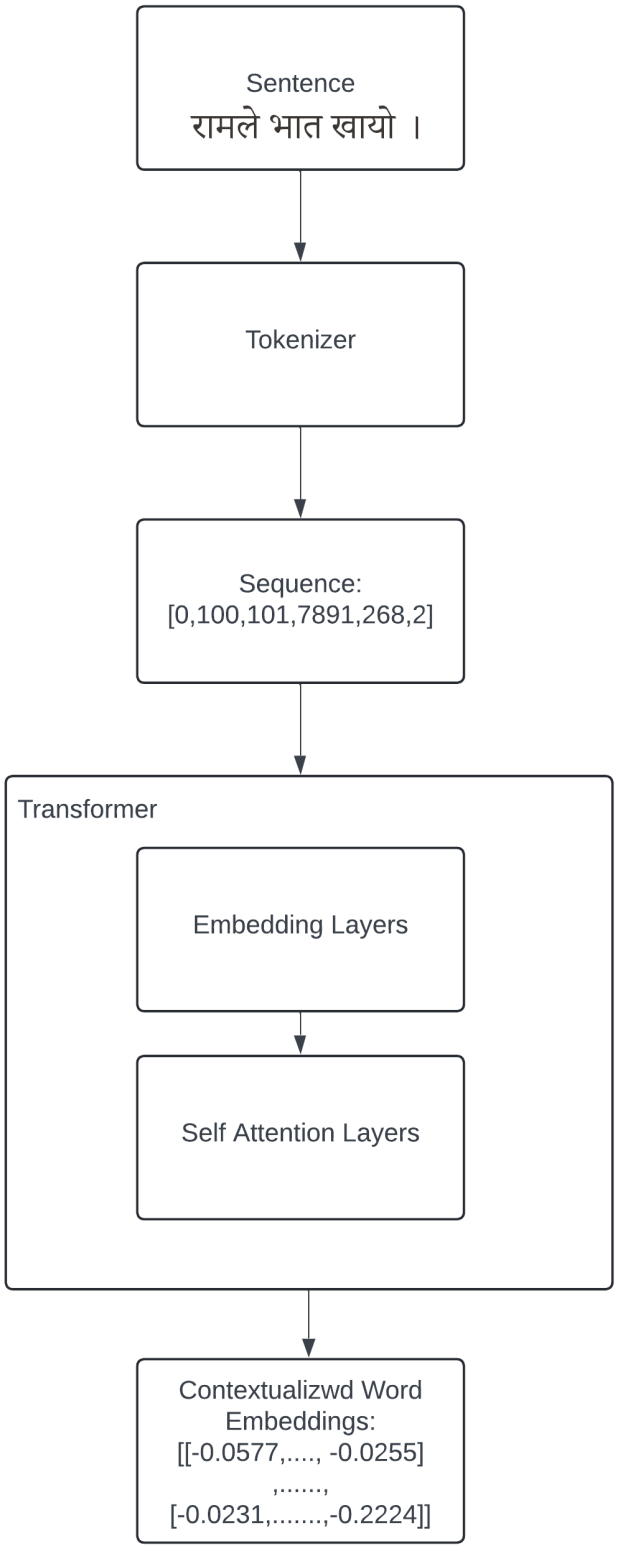

Figure 4.2: Block diagram of Proposed Sytem

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

System Design

5.1 Requirement Analysis

5.1.1 Functional Requirements

These are specifications that describe the fundamental capabilities and behaviors a system or product must exhibit to meet the users' needs and achieve its intended purpose.

5.1.1.1 Tokenization

The system should be able to tokenize Nepali text into individual words or subword units, considering the complexities of the Nepali script.

5.1.1.2 Embedding Generation

Generate dense vector representations (embeddings) for each word or subword unit in the Nepali vocabulary. These embeddings should capture semantic relationships between words.

5.1.1.3 Embedding Lookup

Allow users to retrieve the embedding vector for any given word or subword unit in the Nepali vocabulary.

5.1.1.4 Similarity Calculation

Calculate semantic similarity between words or subword units based on their embedding vectors. Users should be able to compare words and get similarity scores.

5.1.2 Non-Functional Requirements

These are the characteristics and qualities that describe how a system should behave and perform.

5.1.2.1 Usability

The system should have a user-friendly interface or API that allows users to easily interact with word embeddings without requiring deep technical knowledge.

5.1.2.2 Reliability

The system aims to be highly available, with minimal downtime, to ensure users can access word embedding functionalities when needed.

5.1.2.3 Interoperability

The system ensures compatibility with various operating systems, programming languages, and NLP frameworks to facilitate integration with existing systems and workflows.

5.1.2.4 Maintainability

The system with a modular architecture will enable easy maintenance, updates, and future enhancements.

5.1.3 Feasibility Study

The following points describes the feasibility of the project.

5.1.3.1 Economic Feasibility

The total expenditure of the project is just computational power. The computational resources can be fulfilled with the help of college. Therefore, the project is economically feasible.

5.1.3.2 Technical Feasibility

While existing datasets on this topic are available, they are insufficient. But, by using the abundance of Nepali news articles, books, and literature accessible online can augment our corpus significantly through web scraping.

5.1.3.3 Operational Feasibility

The operational processes, including web-scraping and model training, are well-defined and can be efficiently carried out by the project team. Additionally, the project aligns with the existing technical infrastructure and capabilities, making it operationally feasible.

CHAPTER 6

Implementation Plan

6.1 Schedule(Gantt Chart)

Figure 6.1: Gantt Chart

6.2 Hardware and Software requirements

6.2.1 Software Requirement

This project requires following softwares:

Python

Python is our primary language for this project, chosen for its versatility and simplicity. All machine learning frameworks are imported using Python, leveraging its dominant position in data science and machine learning. This allows us to seamlessly integrate powerful libraries like TensorFlow and PyTorch for efficient model development. It also provides essential libraries like Scikit-learn, Numpy, Pandas, etc.

Pytorch

Pytorch is a an open-source machine learning framework. It provides a flexible and dynamic computational graph, which allows for easy experimentation and rapid development of deep learning models.

Natural Language Toolkit

Natural Language Toolkit (NLTK) is a free and open source Python library for natural language processing. NLTK provides stemming, lowercase, categorization, tokenization, spell check, lemmatization, and semantic reasoning text processing packages. It gives access to lexical resources like WordNet.

6.2.2 Hardware Requirement

Memory

Word embedding models like BERT need a lot of memory to work. Training BERT needs several gigabytes of RAM, maybe even more if the model is really big.

Processing Power

Training these models takes a lot of computing power, especially when working with big datasets or complex models. Having a strong computer with a good CPU or GPU

can help speed up the training process. GPUs are especially useful for this because it can handle lots of tasks at once.

Storage Space

Storing all the data and models can take up a lot of space. Adequate disk space is required to store necessary files, including the dataset, the models, and any other files related to the project.

CHAPTER 7

Work Completed

7.1 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.

7.2 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).

7.3 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.

7.4 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.

7.5 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.
- **Validation Workflow:** The model's performance is evaluated on the validation set after each epoch. The validation 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.

7.6 Data Splitting and Loader Implementation

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

7.7 Performance Metrics

Loss during training and validation 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.

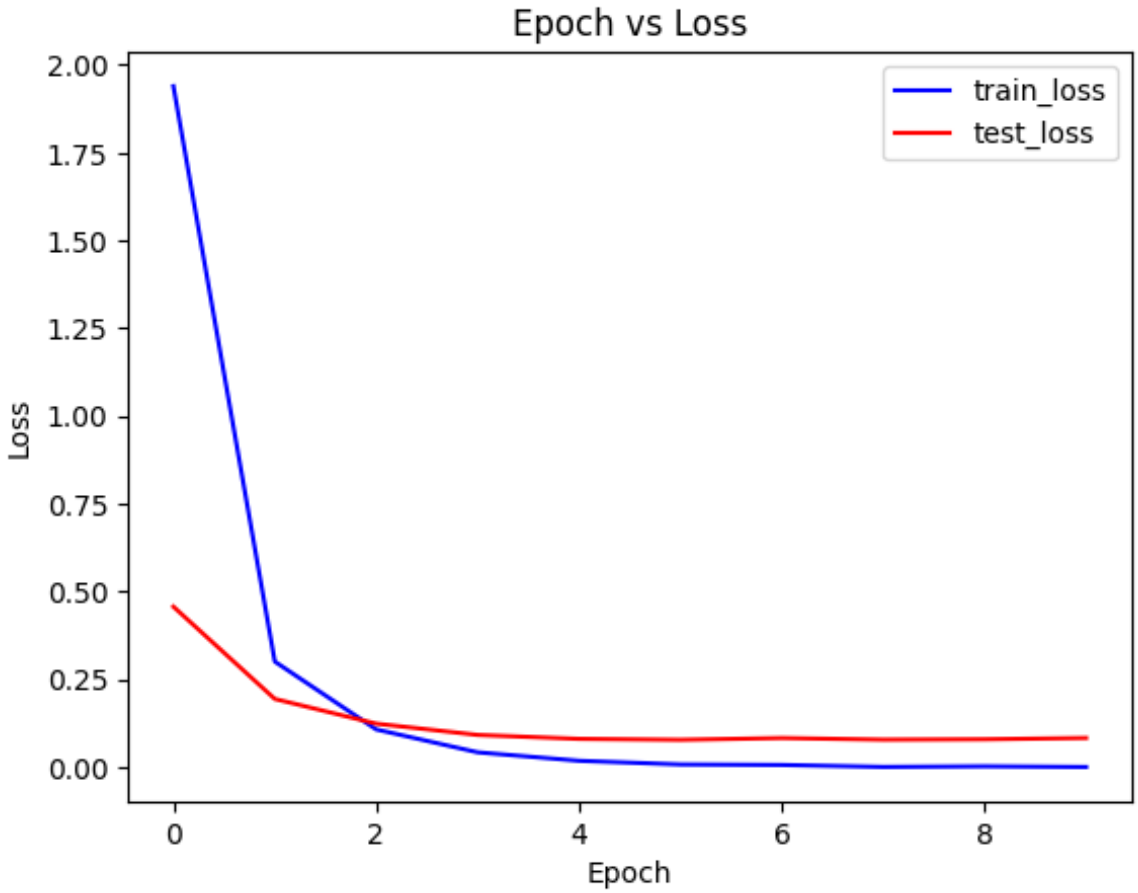

Figure 7.1: Training graph

7.8 Future Work

Our ongoing efforts focus on generating and evaluating Nepali text embeddings for enhanced information retrieval and related applications. The following steps outline the key areas of work in progress:

1. **Embedding Generation:**
    - Generate embeddings using the *original model* and the *fine-tuned model* on test articles to avoid data leakage from training data.
2. **Text Splitting:**
    - Apply techniques such as *LangChain recursive splitting* to divide articles into smaller text chunks, facilitating effective embedding and retrieval.

3. **Embedding Comparison:**

    - Compare embeddings from both models for each text chunk using similarity metrics such as *cosine similarity* or other distance measures.

4. **Question Generation:**

    - Use high-performing language models, such as *LLaMA 3.1 70B*, to generate contextually relevant questions from the text chunks.

5. **Information Retrieval Evaluation:**

    - Retrieve *Top-K text chunks* based on embedding similarity scores for each question and evaluate performance using metrics like *hit rate*.

6. **LLM Response Testing:**

    - Validate the retrieved information by prompting the LLM with retrieved text and assessing its accuracy in answering the generated questions.

This approach provides a structured framework for evaluating the fine-tuned model while enabling future advancements in information retrieval and related applications.

CHAPTER 8

Work to be done

8.1 Expand Dataset Further

Increase the size and diversity of the dataset by collecting more Nepali legal texts. This will improve the model's ability to generalize and perform well on a wider range of inputs.

8.2 Prepare Transformer-Based Embedding Layer and Self-Attention Layers

Implement the core components of a transformer model, including embedding layers to convert words into numerical representations and self-attention layers to capture dependencies between words in the text. These components are crucial for understanding contextual relationships within the data.

8.3 Train, Test, and Validate the Model

Train the model using the expanded dataset to learn from the data and optimize its performance on legal text tasks. Test the model on separate datasets to evaluate its accuracy and generalization capabilities. Validate the model's performance against predefined metrics to ensure it meets quality standards.

8.4 Hyperparameter Tuning

Fine-tune the model by adjusting hyperparameters such as learning rate, batch size, and dropout rates. This process aims to optimize model performance and efficiency, leading to better results during training and inference.

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[2] Sunil Gundapu and Radhika Mamidi. Transformer based automatic covid-19 fake news detection system, 01 2021.

[3] Deepak S Asudani, Narendra K Nagwani, and Pradeep Singh. Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial Intelligence Review*, 22(1):1–81, Feb 2023. Epub ahead of print.

[4] Shushanta Pudasaini, Subarna Shakya, Aakash Tamang, Sajjan Adhikari, Sunil Thapa, and Sagar Lamichhane. Nepalibert: Pre-training of masked language model in nepali corpus. pages 325–330, 10 2023.

[5] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C.-C. Jay Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8:e19, 2019.

[6] Wikipedia contributors. Word embedding — Wikipedia, the free encyclopedia, 2024. [Online; accessed 4-May-2024].

[7] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. Word-class embeddings for multiclass text classification, 2019.

[8] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In Jan Hajič, Sandra Carberry, Stephen Clark, and Joakim Nivre, editors, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[9] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. pages 41–47, 07 2003.

[10] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[11] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey, 2023.

[12] A comprehensive review on word embedding techniques. pages 538–543, 2023.

[13] Nobal B. Niraula, Saurab Dulal, and Diwa Koirala. Linguistic taboos and euphemisms in nepali, 2020.

[14] Pravesh Koirala and Nobal B. Niraula. NPVec1: Word embeddings for Nepali - construction and evaluation. In Anna Rogers, Iacer Calixto, Ivan Vulić, Naomi Saphra, Nora Kassner, Oana-Maria Camburu, Trapit Bansal, and Vered Shwartz, editors, *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 174–184, Online, August 2021. Association for Computational Linguistics.

[15] Janardan Bhatta, Dipesh Shrestha, Santosh Nepal, Saurav Pandey, and Shekhar Koirala. Efficient estimation of nepali word representations in vector space. *Journal of Innovations in Engineering Education*, 3(1):71–77, Mar. 2020.

[16] Oyesh Mann Singh. Nepali multi-class text classification. 2019.

[17] Tej Bahadur Shahi and Ashok Kumar Pant. Nepali news classification using naive bayes, support vector machines and neural networks. In *2018 international conference on communication information and computing technology (iccict)*, pages 1–5. IEEE, 2018.

[18] Samujjwal Ghosh and Maunendra Sankar Desarkar. Class specific tf-idf boosting for short-text classification: Application to short-texts generated during disasters. In *Companion Proceedings of the The Web Conference 2018*, pages 1629–1637, 2018.

[19] Kaushal Kafle, Diwas Sharma, Aayush Subedi, and Arun Timalsina. Improving nepali document classification by neural network. 03 2018.

[20] Ashok Basnet and Arun Timalsina. Improving nepali news recommendation using classification based on lstm recurrent neural networks. pages 138–142, 10 2018.

[21] Rabindra Lamsal. 300-dimensional word embeddings for nepali language, 2019.

[22] Rajan. Nepalibert. `https://huggingface.co/Rajan/NepaliBERT`, 2021.

[23] Milanmg. Bert-nepali. `https://huggingface.co/Milanmg/Bert-Nepali`, 2022.

[24] Alexis CONNEAU and Guillaume Lample. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[25] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.