Tribhuwan University

Institute of Engineering

Pulchowk Campus

A Project Proposal on:

Transformer based model for Nepali Language Generation, Spelling Correction and Automatic Speech Recognition

---

Submitted By:

Nabin Da Shrestha(076bct037)

Nirajan Bekoju(076bct039)

Nishant Luitel (076bct041)

Submitted To:

Department of Electronics and Computer Engineering

Submission Date:

20^{th} January, 2023

Contents

List of Figures

List of Abbreviations

NLP Natural Language Processing

NPLM Neural Probabilistic Language Model

ASR Automatic Speech Recognition

RNN Recurrent Neural Network

GRU Gated Recurrent Unit

LSTM Long Short-Term Memory

MFCC Mel Frequency Cepstral Coefficients

Chapter 1

Introduction to Nepali Language Model

1.1 Abstract

This proposal is prepared for the development of the Text Generation Model(Language Model), Spelling Correction system, and Automatic Speech Recognition system for the Nepali language. The broad objective of our project is to use emerging NLP technologies in the Nepali Language so as to enhance communication and accessibility for Nepali speakers. The project will involve the implementation of machine learning models and the utilization of a large corpus of Nepali text and speech data to train and fine-tune the models. A Language Model(LM) is a powerful computational model that predicts the likelihood of a sequence of words or characters based on the context provided. The successful training of an LM will result in a robust and accurate text generation model capable of being used as a basis for the development of various other NLP systems like Automatic Speech Recognition, Spelling Correction, Nepali Text Summarization, Question Answering, Content Generation, Paraphrasing, etc. This proposal aims to develop one such general LM for the Nepali language and show how the generative model can be used to empower two important downstream applications: Spelling Correction and Speech Recognition.

Keywords: Nepali Text Generation, Nepali Speech-to-Text, Transformer, LSTM, MFCC

1.2 Introduction

This proposal outlines a project focused on text generation, spelling correction, and automatic speech recognition for the Nepali Language. The objective of this endeavor is to develop advanced language processing technologies specifically tailored to the Nepali language, facilitating improved communication, information access, and interaction for Nepali-speaking individuals.

Nepali, as one of the prominent languages in Nepal and various regions of South Asia, holds significant cultural and linguistic importance. However, the availability of robust language processing tools and resources for Nepali remains limited compared to widely studied languages. This project aims to bridge this gap by developing innovative solutions that address the specific linguistic characteristics and challenges of the Nepali language.

The project will primarily focus on three key areas: Nepali text generation, Spelling Correction, and Nepali automatic speech recognition.

1.2.1 Nepali Text Generation

Nepali text generation involves the creation of algorithms and models capable of generating coherent and contextually appropriate Nepali text. This technology holds immense potential in applications such as natural language interfaces, chatbots, content generation, and machine translation systems. By developing advanced techniques for Nepali text generation, we can significantly enhance the quality and efficiency of communication and content creation in Nepali.

1.2.2 Nepali Spelling Corrector

Spelling correction involves identifying and rectifying misspelled words in text documents, thereby enhancing readability, clarity, and overall quality of written content. In the realm of social media and online communication platforms, spelling correction algorithms are instrumental in automatically suggesting corrections and avoiding misunderstandings caused by misspellings. By automatically detecting and rectifying spelling errors, these algorithms ensure clearer communication, enhance information retrieval, and facilitate more accurate translations.

1.2.3 Nepali Automatic Speech Recognition

Nepali automatic speech recognition aims to design and develop algorithms and systems that can accurately transcribe spoken Nepali into written text. This technology has the potential to revolutionize various domains, including transcription services, voice-controlled applications, voice assistants, and accessibility tools for individuals with hearing impairments. By leveraging state-of-the-art techniques in speech recognition, we can unlock new opportunities for Nepali speakers to interact with technology and access information more seamlessly.

The successful implementation of this project will require extensive research, data collection, algorithm development, and rigorous evaluation. We will adopt a multidisciplinary approach, combining expertise in natural language processing, machine learning, linguistics, and Nepali language resources.

Furthermore, this project aims to foster collaborations with language experts, academic institutions, industry partners, and native speakers to ensure the authenticity and effectiveness of the developed technologies. Additionally, we will explore avenues for open-source contributions and knowledge sharing, aiming to create a lasting impact in the Nepali language processing community.

By undertaking this project, we aspire to make significant strides in advancing the capabilities of Nepali language processing technologies, empowering Nepali speakers.

1.3 Problem Statement

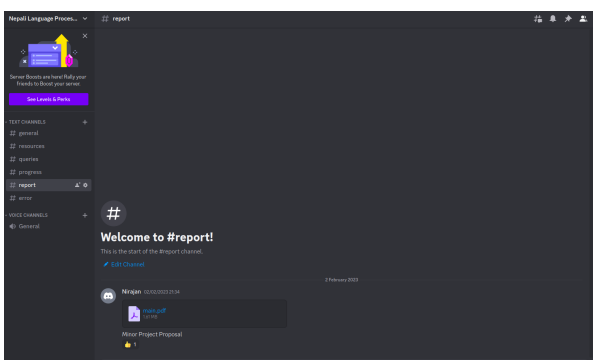
1. There is no proper development of various NLP tasks like text generation, text summarization, image captioning, text to speech, etc. due to lack of reliable Nepali language model

2. Nepali Language is rich in vocabulary, and it is difficult to choose the best possible vocab.
3. Substantial portion of Nepalese population is unable to write properly, despite having clear speech. This issue can be addressed using Nepali ASR.
4. Transcribing and Documenting the spoken Nepali content is time-consuming and requires greater effort which can be made easier with ASR.

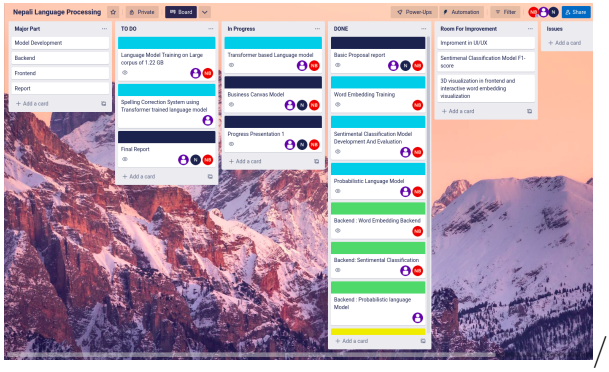
1.4 Objectives

1. To develop Nepali language model.
2. To use the language model for developing Spelling Correction System and Automatic Speech Recognition System for the Nepali Language.

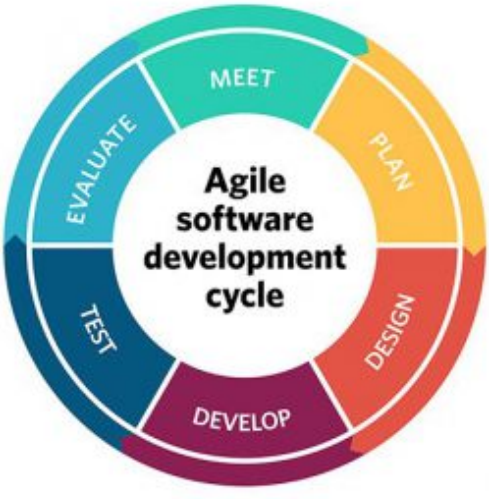
1.5 Project Management



(a) Discord



(b) Trello



(c) Agile Methodology

Figure 1.1: Project Management

Discord will be used as it offers a versatile and user-friendly environment for team communication. Trello will be use as project management tool as it offers a visual and intuitive interface for organizing tasks and tracking project progress. The default agile board will be used for managing this project.

1.6 Expected Outcome

For language generation model, after user input some text, language generation model should suggest next words as shown below.

Figure 1.2: Nepali Text Generation

For the Spelling Correction system, the input is possibly misspelled Nepali sentences, and the output is contextually determined possible candidate correction words for every incorrect word in the input. Finally, for the Nepali ASR, the model should be able to take Nepali audio as an input and convert it to Nepali text.

Chapter 2

Literature Review

2.1 A Neural Probabilistic Language Model

2.1.1 Introduction

Goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the curse of dimensionality. NPLM proposes to fight the curse of dimensionality by learning a distributed representation of words which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously a distributed representation for each word along with the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar to the words forming an already seen sentence. The idea of the proposed model can be summarized as follows

- associate with each word in the vocabulary a distributed word feature vector (a real-valued vector in $\mathbb{R}^m$, where m is the size of embedding vector.)
- express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence
- learn simultaneously the word feature vectors and the parameters of that probability function

2.1.2 NPLM Architecture

The basic form of the model is shown in the figure 2.1. The objective is to learn the function f(w_t, w_{t-1} ...., w_{t-n}) = P(w_t | w_1^{t-1}), in the sense that it gives high out-of-sample likelihood. A mapping C from any element of V to a real vector C(i) ϵ $\mathbb{R}^m$. It represents the distributed feature vector associated with each word in the vocabulary. In practice, C is represented by a |V| x m matrix (of free parameters)

From the direct architecture figure 2.1, f(i, w_{t-1}, w_{t-2}...., w_{t-n}) = g(i, $C(w_{t-1})$, $C(w_{t-2})$ ..., C(w_{t-n})). Softmax function is used in the output layer of the neural network to get the probability of the target word.

The function f is the composition of two mappings (C and g), with C being shared across all words in the context. The function g may be implemented by a feed-forward or recurrent neural network or another parameterized function, with parameters θ.

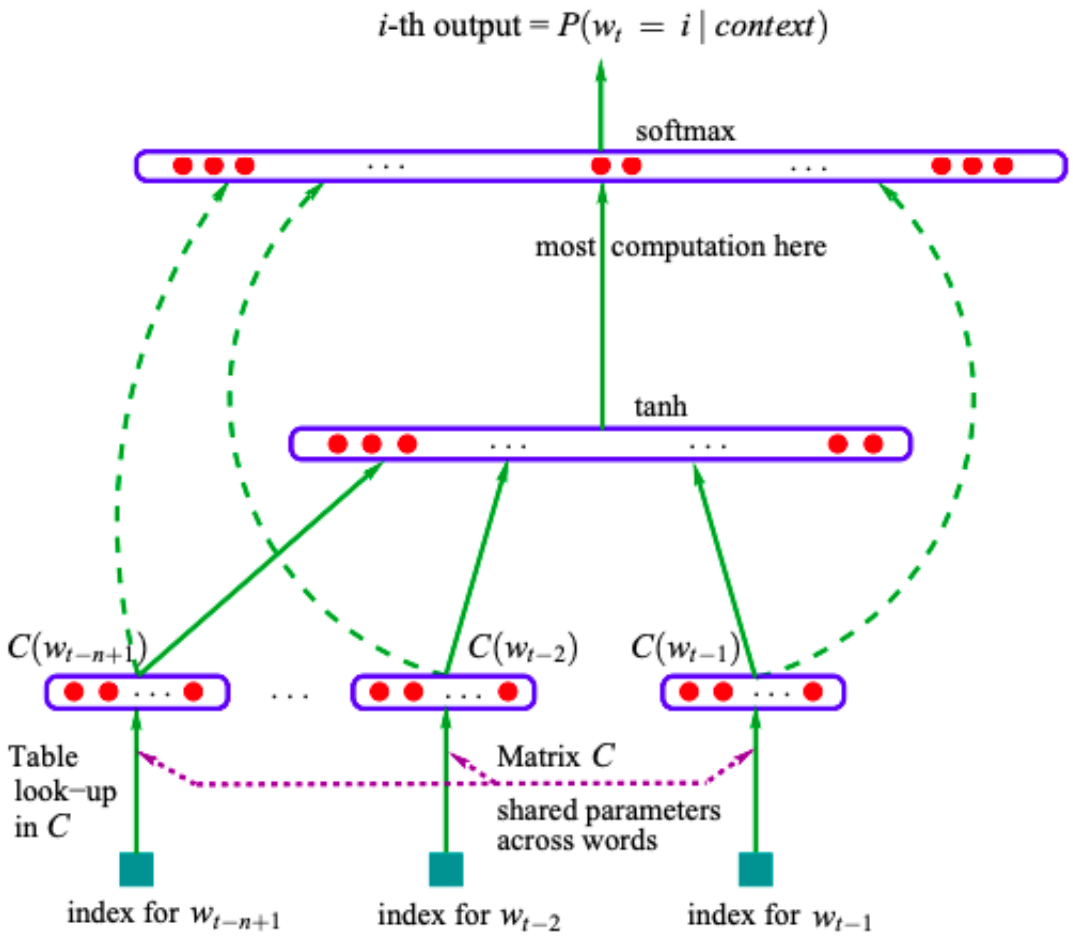


Figure 2.1: Direct Architecture of Probabilistic Language Model

2.1.3 Conclusion and Results

The main result of this experiment is that the neural network performs much better than the smoothed trigram. Greater the length of the context words and higher the number of hidden units, the model becomes more efficient. Moreover, direct architecture was found to be better by about 2% than the cycling architecture.

It can be deduced that the neural probabilistic model performs better due to the advantage of the learned distributed representation to fight the curse of dimensionality.

2.2 Attention is all you need

2.2.1 Introduction

Before the introduction of the transformer, the dominant sequence transduction models were based on the complex recurrent or convolutional neural networks that include an encoder and a decoder. But this paper introduced a new network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing the modeling of dependencies without regard to their distance in the input or output sequences. In all but a few cases, however, such attention mechanisms are used in conjunction with a recurrent network.

2.2.2 Transformer Model Architecture

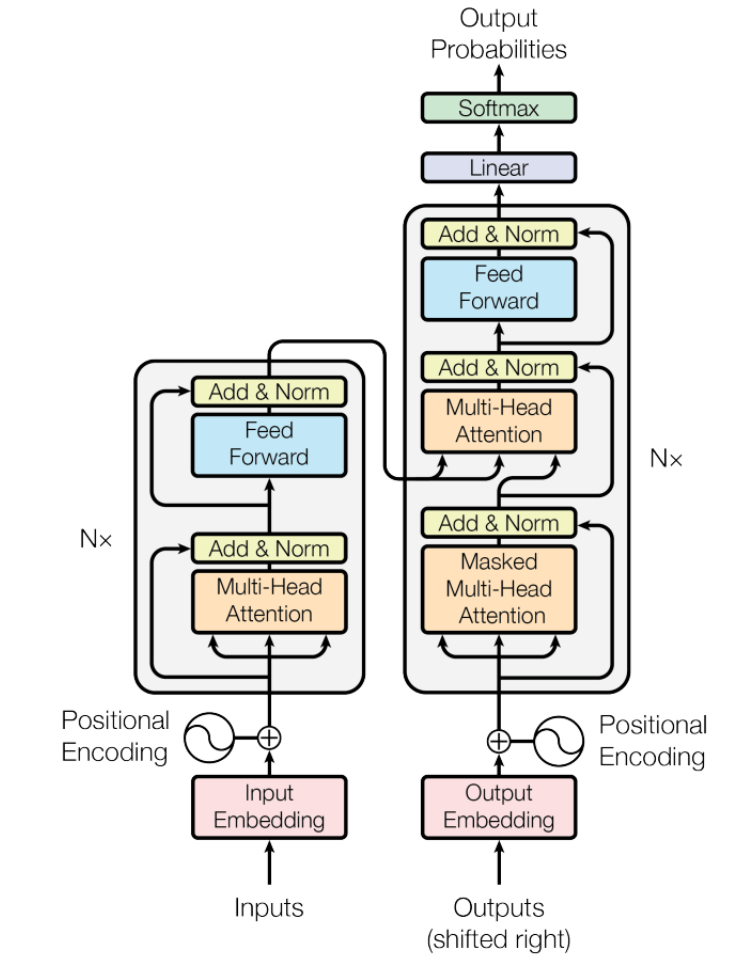


Figure 2.2: The Transformer - model architecture

Encoder and Decoder Stacks

The encoder is composed of a stack of N = 6 identical layers. Each layer has two sublayers: multi-head self-attention mechanism, and the simple position wise fully connected feed-forward network. Residual connection around each of two sub layers, followed by a layer normalization is employed in the encoder.

The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, residual connections around each of the sub layers, followed by layer normalization is employed. Self attention sub-layer in the decoder stack is modified to prevent the positions from attending to subsequent positions.

Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

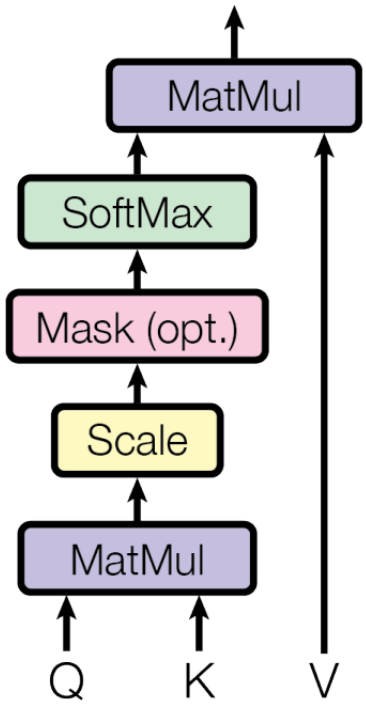


Figure 2.3: Scaled Dot-Product Attention

In scaled dot-product attention, the input consists of queries and keys of dimension d_k and values of dimension d_v. We compute the dot products of the query with all keys, divide each by d_k and apply a softmax function to obtain the weights on the values.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

In multi-head attention, instead of performing a single attention function with d_{model} -dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k , d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in figure 2.4.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_2)W^o \tag{2.2}$$
$$\text{where } head_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V)$$

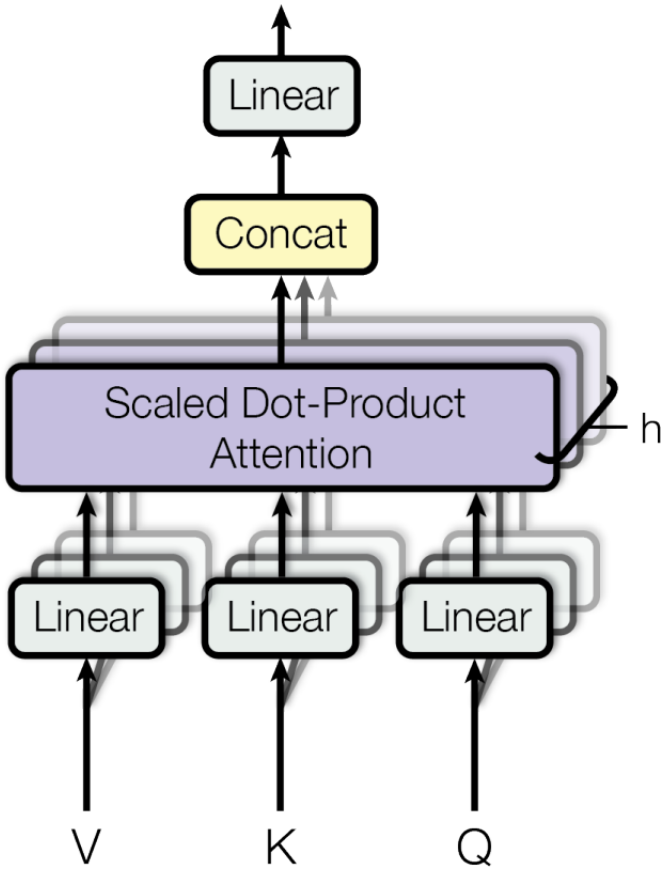


Figure 2.4: Multi-Head Attention consists of attention layer running in parallel

Position wise Feed Forward Neural Network

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically.

This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.3}$$

Embedding and Softmax

Similarly to other sequence transduction models, Learned embeddings are used to convert the input tokens and output tokens of dimension d_{model}. Usual learned linear transformation and softmax function are used to convert the decoder output to predict next-token probabilities.

Positional Encoding

Since the model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, some information about the relative or absolute position of the tokens in the sequence must be injected. To this end, "positional encoding" are added to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encoding have the same dimension model as the embeddings, so that the two can be summed. Sine and cosine functions of different frequencies are used for positional encoding as follows

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.4}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.5}$$

Where *pos* is the position and i is the dimension. That is, each dimension of the positinal encoding corresponds to a sinusoid.

2.2.3 Conclusion and Results

In the machine translation task, the transformer model outperformed all previous state of the art models which have used the RNNs, GRUs and LSTMs. To evaluate if the Transformer can generalize to other tasks, experiments on English constituency parsing were performed and despite the lack of task-specific tuning, the transformer model performed surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar.

In this work, the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention is presented.

Till today, various transformer models have been developed which have performed better on all kinds of natural language processing tasks like language modeling, text classifications, questions answering, machine translation, sentence similarity, summarization, etc.

2.3 Nepali Spelling Correction

In the pre-computer era, spelling correction was primarily a manual process. Dictionaries and grammar books were used to verify correct spellings, and proofreaders manually identified and

corrected spelling errors in written texts. With the advent of computers, rule-based spelling correction systems emerged. These systems relied on predefined rules to identify and correct spelling mistakes. Rules were typically based on common spelling patterns, phonetics, and grammatical rules. One phonetical rule-based algorithm is Soundex Algorithm. However, these early systems had limited effectiveness due to the vast number of language-specific exceptions and irregularities. Another set of algorithms is based on a combination of dictionary-based and rule-based. For example, Hanspell is a dictionary-based spelling correction engine that applies certain transformations to find the rank of candidate words in a dictionary to make predictions for the correct spelling.

Apart from using rules, statistical methods are also popular. These approaches used large corpora of correctly spelled words to estimate the likelihood of different word sequences. By comparing input words to these statistical models, spelling errors could be identified, and suggestions for corrections generated. Early statistical models often used n-gram language models and algorithms like the noisy channel model. The noisy channel model, though developed by Shannon in 1948, was first proposed for spelling correction in IBM Watson Research(Mays et al., 1991) and at Bell Laboratories (Kernighan et al. 1990) with the idea of combining prior and likelihood.

One can use statistical models to utilize corpora of unlabeled text to build both the likelihood model(error model) and the prior. Whitelaw et al.(2009), used huge texts of web pages to train the n-gram, prior model. They used a partitioning-based likelihood model developed by Brill and Moore(2000). In this approach, words were partitioned into substrings and the substrings were aligned with the possible correct word. The simple counting-based probability can be calculated for all such substrings using triples of the correct word, misspelled word, and frequency extracted from the corpus in an unsupervised way. The approximate likelihood probability can then be calculated by multiplying all probability values for the substring.

2.4 Nepali Speech Recognition Using CNN, GRU, and CTC

This paper presents an idea to build the Nepali ASR system to convert the spoken Nepali language to its textual representation using a CNN, GRU, and CTC model. The features in the raw audio are extracted by using the MFCC algorithm. MFCC features are a sequence of Acoustic feature vectors where each vector represents information in a small-time window of the signal. CNN is used to capture high-level spatial features from the image. The plot of MFCC can be viewed as a transformed intensity of frequencies over time which resembles images, hence CNN can be used to capture high-level features in the spatial domain. GRU is responsible for constructing the acoustic model. The decoding is carried out using a CTC network. The CTC is based on Bayes' decision theory. It receives output from the softmax function.

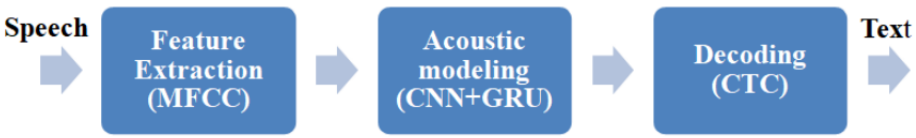


Figure 2.5: Architecture of proposed ASR system

The experimental setup is carried out on the GPU MX150. For the pre-processing, feature extraction, training, and testing, python and its library have been used. The obtained results from various experiments were as follows

Experiment	learning rate	batch size	total epochs	WER
1	0.03	100	44	90
2	0.03	300	100	80
3	0.015	50	100	11

Figure 2.6: Summary of experiments with the results

Chapter 3

Requirement Analysis and Feasibility Study

3.1 Functional and Non-Functional Requirements

3.1.1 Functional Requirements

1. **R1: Text Generation Model**

   Description: Nepali Language modeling based on both probabilistic and neural approach. Based on the trained language model, it should be able to auto generate the next word given context words.

   (a) **Predict next word**

   Input: An incomplete sentence

   Output: List of words along the probabilities of being the next word.

2. **R2: Nepali Spelling Corrector**

   Description: Nepali Spelling Corrector is to be developed and should be able to correctly find misspelled words and provide corrected suggestion based on the context.

   (a) **Convert Speech To Text**

   Input: Nepali Sentences

   Output: Possible candidate correction for words with high error chance.

3. **R3: Nepali Speech-to-Text**

   Description: Nepali Speech-to-Text model is to be developed and should be able to convert nepali speech to text accurately.

   (a) **Convert Speech To Text**

   Input: Nepali Audio

   Output: Text corresponding to the input nepali audio

4. **R4: Interactive Web Application**

   Description: All the requirements should be easily accessible through a website and APIs.

5. **R5: Browser Extension**

   Description: The text generation and speech to text should be accessible through browser extension.

3.1.2 Non-Functional Requirements

1. **R1: Performance and scalability**
   (a) The load time for the user interface screens shall take no longer than 3 seconds.
   (b) Queries shall return results within 3 seconds.
2. **R2: Design Constraints**
   (a) The system shall be developed using python and postgresql databases.
3. **R3: Standard Compliance**
   (a) The graphical user interface shall have a consistent look and feel.
4. **R4: Availability**
   (a) The system shall be available all time.
5. **R5: Portability and Compatibility**
   (a) The system shall be able to run in all browsers.
6. **R6: Reliability and Maintainability**
   (a) The mean time to recover from a system failure must not be greater than 2 hours.
   (b) Rate of system failure should not be greater than twice a month.

3.2 Hardware Requirements

1. GeForce RTX 3060 for prototype development
2. Minimum 32 GB RAM
3. Minimum 5 GB Storage for Hosting

3.3 Software Requirements

1. Programming Language : Python, Javascript
2. Libraries : Tensorflow, keras, scikit-learn, spacy, nltk, and other python libraries
3. Framewoks : Django, Django Rest Framework, React
4. Application software : Postman

5. Target Platform : Web

3.4 Feasibility Study

The feasibility study encompassed different domains, including technical, economic, legal, and time-related aspects. The findings in each domain are outlined below.

3.4.1 Technical Feasibility

For our system, the required software and hardware were readily available to us. So we can state with confidence that the project is technically feasible.

3.4.2 Economic Feasibility

Since most of the hardware was already in our possession and the software used was mostly free, we can declare that the system is economically feasible.

3.4.3 Legal Feasibility

Since we are using open source data for building language model, we can state that the project is legally feasible.

3.4.4 Scheduling Feasibility

We have carefully strategized our project completion approach, employing the Agile model and breaking it down into components. It can be concluded that our timeline for project completion is viable.

Chapter 4

Proposed Methodology

Our project will be divided into two distinct phases. Phase 1 will involve the development of a Nepali Language model dedicated to text generation and spelling correction. Once Phase 1 is successfully completed, we will proceed to Phase 2, which will focus on the development of Nepali Automatic Speech Recognition.

4.1 Nepali Language Model for Text Generation

During this phase, our focus will be on creating a Nepali language model for text generation. To achieve this, we will work on building both a transformer model and a GPT-based model. The architectures of these models are outlined below:

4.1.1 Transformer based model for text generation

In this phase, our transformer-based model will utilize a standard transformer module based on the paper "Attention is All You Need" [1] as discussed in the literature review section. However, for the language generation task, we will specifically train only the transformer encoder. The objective of language modeling is to estimate the probability of a given word following a sequence of words.

The model architecture involves passing a sequence of tokens through an embedding layer, which is then followed by a positional encoding layer to capture the word order. The next steps include employing multi-head attention and a feed-forward network, as depicted in the figure below. To ensure effective modeling, we will utilize six layers of the transformer encoder.

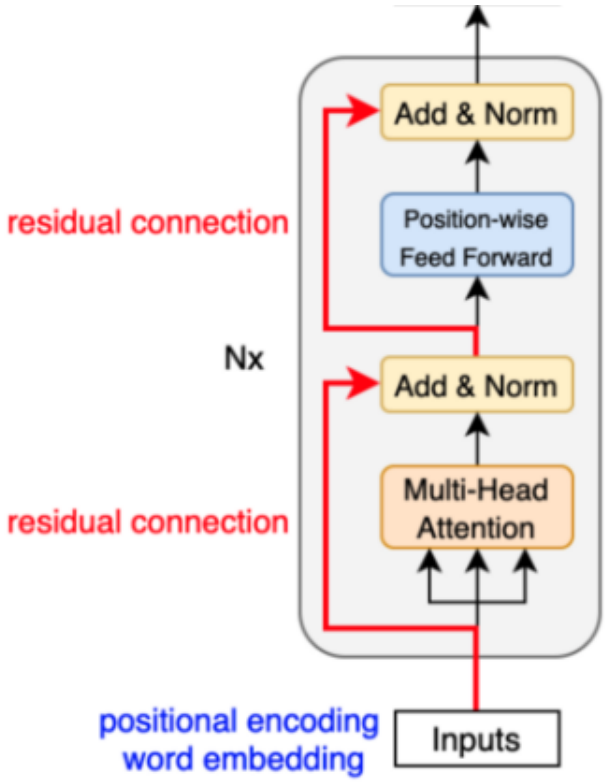


Figure 4.1: Transformer Encoder (src: https://kikaben.com/transformers-encoder-decoder/)

4.2 Spelling Correction

4.3 Nepali Automatic Speech Recognition

The audio input feature in the Transformer architecture for speech recognition is a sequence of Mel-frequency cepstral coefficients (MFCCs) that are extracted from the audio signal. MFCCs are commonly used in speech recognition systems as they capture the spectral envelope of the speech signal and are robust to noise and channel distortions. Other description of the architecture is same as in the "Attention is All You Need" [1].

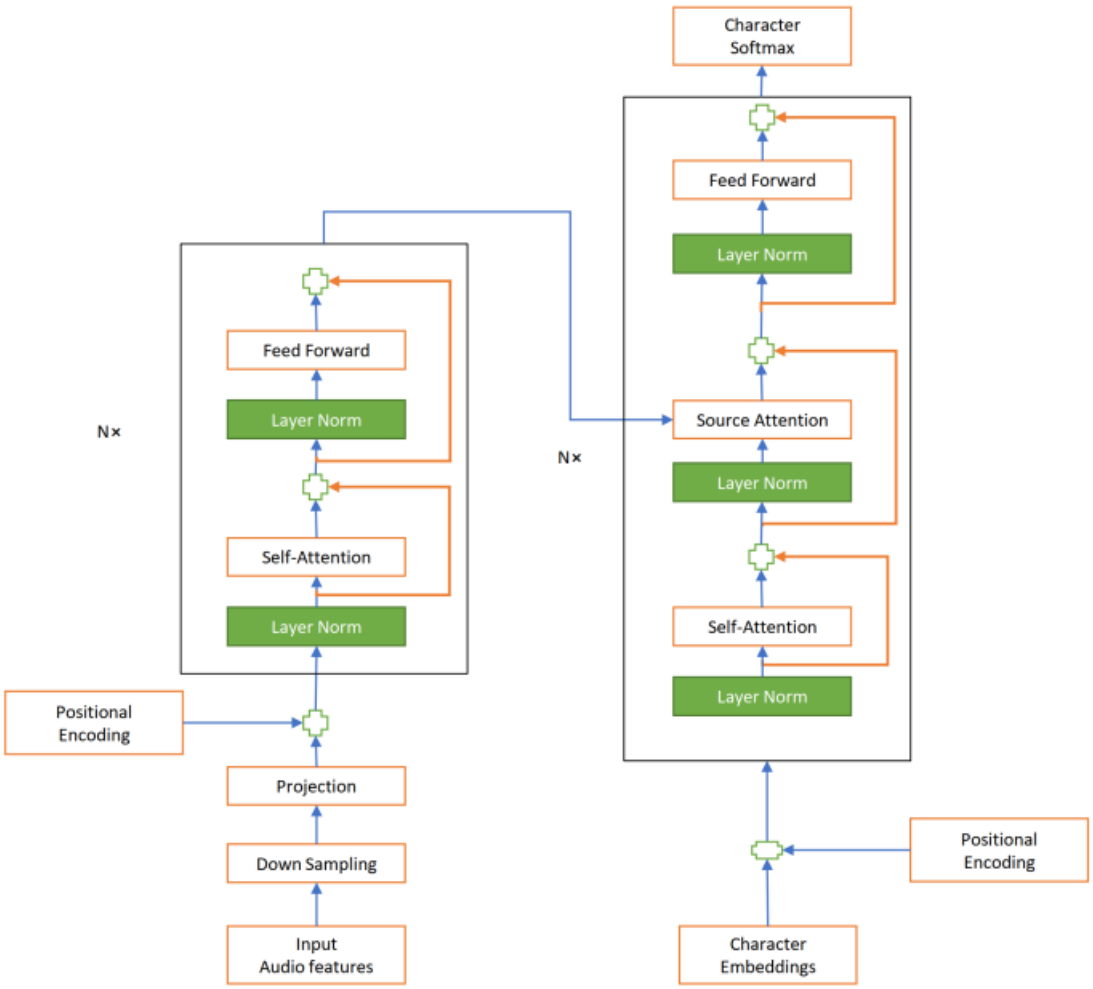


Figure 4.2: Transformer Network for ASR [9]

Chapter 5

Conclusion

In conclusion, the development of Nepali text generation, Spelling correction system, and Nepali Automatic Speech Recognition (ASR) systems holds immense potential for advancing linguistic technology and fostering broader accessibility and inclusivity within the Nepali language community. By harnessing the power of natural language processing and speech recognition technologies, we can unlock new avenues for automated Nepali content creation, translation, transcription, and accessibility services. These advancements will not only enhance communication and engagement across various sectors but also contribute to preserving and promoting the rich linguistic heritage of Nepal. Investing in the development of these systems will facilitate seamless interaction, empowerment, and innovation for Nepali speakers, ultimately fostering greater linguistic diversity and cultural exchange on a global scale.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[5] S. Timilsina, M. Gautam, 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 12th International Joint Conference on Natural Language Processing*, pp. 273–284, 2022.

[6] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, *et al.*, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 449–456, IEEE, 2019.

[7] B. Joshi, B. Bhatta, S. P. Panday, and R. K. Maharjan, "A novel deep learning based nepali speech recognition," in *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2022, Volume 2*, pp. 433–443, Springer, 2022.

[8] B. Bhatta, B. Joshi, and R. K. Maharjhan, "Nepali speech recognition using cnn, gru and ctc," in *Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020)*, pp. 238–246, 2020.

[9] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *arXiv preprint arXiv:1904.13377*, 2019.