Leveraging Transliteration, Spelling Detection and Correction, Parts of Speech Tagging and Next Word Prediction for Effective Nepali Typing

Rajan Ghimire
Department of Computer and Electronics Engineering
Kantipur Engineering College
Dhapakhel, Lalitpur
rjnghimire@outlook.com

Rikesh Basnet
Department of Computer and Electronics Engineering
Kantipur Engineering College
Dhapakhel, Lalitpur
rikesbasnet@outlook.com

Raul Shahi
Department of Computer and Electronics Engineering
Kantipur Engineering College
Dhapakhel, Lalitpur
rahulfromnepal2013@gmail.com

Sarina Joshi
Department of Computer and Electronics Engineering
Kantipur Engineering College
Dhapakhel, Lalitpur
josheesarina@gmail.com

***Abstract*—Leveraging Transliteration, Spelling Detection and Correction, Parts of Speech Tagging and Next Word Prediction for Effective Nepali Typing is a research based on Natural Language Processing built with an idea to assist people in faster, accurate and efficient Nepali language typing. There has been only a handful of researches based on Nepali Language Processing. Only a countable number of researches have been done in this arena so, we brought up this idea with an intent of improving the efficiency of typing in our native language in today's digital platform. In today's world, the newspapers have turned into an online news portal, the official documents, administrative paper works, application letters everything has been transformed into digitized form. This brings essence to such an application where people can type in their native language in a hassle-free manner. They are important for the flow of correct information in form of text. Nepali is the only understandable means of communication for the majority of the population in Nepal. A lot of text in the form of newspapers, books, novels, magazines, web pages, and other documents is typed in Nepali. We, in this study, have attempted to build such a versatile application that integrates Devanagari transliteration, Next Word Prediction, Spell Correction and Parts of speech Recognition all in a single application. Next Word Prediction and Parts of speech Recognition is performed using custom Bidirectional Encoder Representations from Transformers (BERT) model trained using Nepali dataset.**

***Index Terms*—effective Nepali typing, Devanagari Transliteration, Spell Correction, Parts of Speech Recognition, Next Word Prediction, BERT**

I. Introduction

With the modernization of browsers, the font compatibility issues have no longer been much of a problem. However, Nepali typing is not as easy as it seems. We can either opt for a Nepali keyboard to type in Nepali or extract Nepali words from various transliteration tools available online. Since we are much used to typing in English language, most of the people find it difficult to type in Nepali using a Nepali keyboard and the prevalent online conversion tools are not error free. It becomes hectic to find the appropriate words with correct grammatical syntax in online platforms. Accurate Nepali script writing has become a common challenge of Nepalese people. From the language we use for daily communication to professional and administrative field, Nepali language is used extensively. It is not promising to see that for such an extensively used language, even the official documents, administrative paper works contain script errors. Technological surge has brought the people closer than ever. In such case, thus made script errors are spread across the globe within split seconds and in the worst case those minor overlooked errors change the entire context of the intended subject matter. The thing that motivated us to do this study was that there was no such platform which rendered error-free and convenient Devanagari transliteration. Consequently, aspiring to solve these problems, we intend to perform this study which integrates different features in order to assist people in faster, accurate and efficient Nepali language typing. While using Devanagari Transliteration, we cannot assure complete error free translation. Hence, we also intend to incorporate features such as error detection and correction. Since newspapers have turned into online news portals, official documents, administrative paper works, application letters everything has been transformed into digitized form, it is absolutely necessary that the errors are reduced as much as possible. In addition to that, we will also incorporate the feature such as next word prediction which will help to type sentences faster. Next word prediction will help to predict the words that are contextual and relevant to the precedent words. The Parts of speech recognition feature will recognize part of speech entities which will be useful to get acquainted to grammatical syntax and semantics of Nepali language in an organized way. Our approach is to bring up such a versatile system which will assist in faster, accurate and efficient Nepali language typing.

II. Related Works

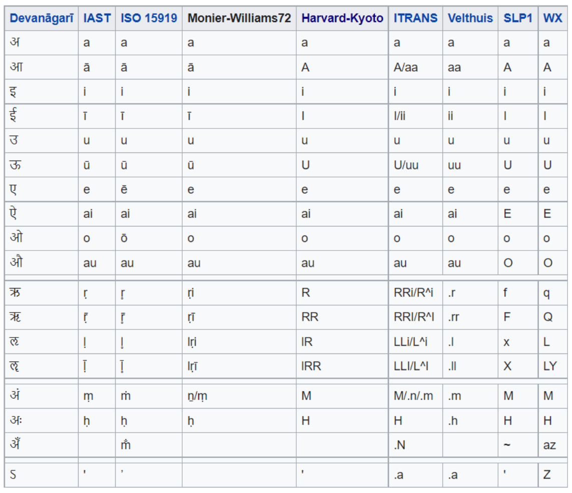
There are several pre-existing applications which render services to type in Nepali language using Devanagari Transliteration. We are passionate about solving this issue by bringing out such a system which not only allows users to type in our native language but that too with an optimum accuracy and faster performance. This research aims to combine multiple functionalities like Devanagari Transliteration, Auto-correction, Next Word Prediction, Part of Speech Recognition in a single application.

J. Nair and A. Sadasivan [1] describe different types of transliteration strategies and give the algorithm to transliterate into hindi using the phonetic strategies and Harvard-Kyoto convention of Devanagari transliteration. R. Kumar et al.[2] presents an algorithm for identifying and correcting any spelling mistakes in a text in a particular language. By describing about the types of spell errors and using dictionary lookup technique the algorithm detects the spelling errors in the input sentence or paragraph and also corrects these errors by replacing them with words that are present in the look up dictionary that have longest common subsequence. A. Pal and A. Mustafi [3] describes about the accuracy of Indic languages such as Hindi generated from Optical Character Recognition and uses automatic spelling error detection and context-sensitive error correction. For context-sensitive error correction, this research has used a transformer BERT in conjunction with Levenshtein distance commonly known as Edit distance. This research has also used lookup dictionary and context-based named entity recognition for detection of possible spelling errors in the text. A. ’Sharma’ and P. ’Jain’[4] uses dictionary lookup technique along with edit distance for possible correction of erroneous words. Along with that, this research uses N gram technique to deal with real word errors. T. Wolf and V. Sanh [5] describes how transformer architectures have facilitated building higher capacity models and pre training has made it possible to effectively utilize this capacity for a wide variety of tasks. A. Vaswani et al. [6] proposes a Transformer which is based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. J. Devlin et al. [7] introduces a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. It explains that BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. POS tagging of Nepali Text was carried out using simple RNN, LSTM, GRU and Bi-directional LSTM in a Nepali tagged corpus of tag size 40 in Sayami et al.[8]

III. Methodology

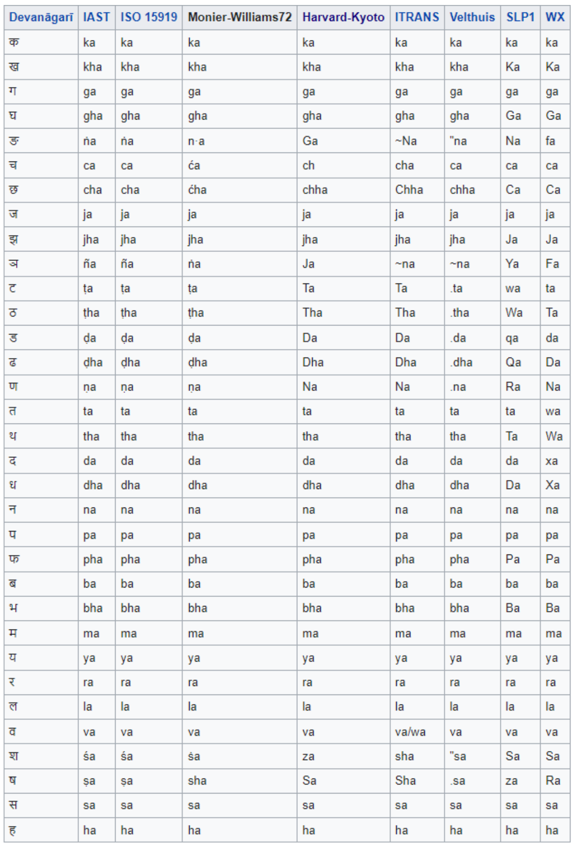
A. Devanagari Transliteration

Transliteration refers to a process by which written text (alphabetic letters and symbols) of one language is used to express the closest corresponding sound of another language. Devanagari transliteration is to enable user to type in English alphabet which is converted to the corresponding Devanagari. There exist four methods of transliteration like: phoneme based transliteration, grapheme based transliteration and correspondence based transliteration and hybrid transliteration [1]. In our research we use phoneme based transliteration method because in this approach transliteration key is pronunciation or the source phoneme rather than spelling so it make easier while typing. There are different schemes available to represent the Devanagari consonant and vowels in equivalent phonetic representation. They are: IAST, ISO 15919, Monier-Williams72, Harvard-Kyoto, ITRANS, Velthuis, SLP1 and WX which is represented in figure 1, figure 2 and figure 3.



Devanāgarī	IAST	ISO 15919	Monier-Williams72	Harvard-Kyoto	ITRANS	Velthuis	SLP1	WX
अ	a	a	a	a	a	a	a	a
आ	ā	ā	ā	A	A/aa	aa	A	A
इ	i	i	i	i	i	i	i	i
ई	ī	ī	ī	I	I/ii	ii	I	I
उ	u	u	u	u	u	u	u	u
ऊ	ū	ū	ū	U	U/uu	uu	U	U
ए	e	ē	e	e	e	e	e	e
ऐ	ai	ai	ai	ai	ai	ai	E	E
ओ	o	ō	o	o	o	o	o	o
औ	au	au	au	au	au	au	O	O
ऋ	ṛ	r̥	ṛi	R	RRi/R^i	.r	f	q
ॠ	ṝ	r̥̄	ṛī	RR	RRI/R^I	.rr	F	Q
ऌ	ḷ	l̥	lṛi	lR	LLi/L^i	.l	x	L
ॡ	ḹ	l̥̄	lṛī	lRR	LLI/L^I	.ll	X	LY
अं	ṃ	ṁ	ṉ/ṃ	M	M/.n/.m	.m	M	M
अः	ḥ	ḥ	ḥ	H	H	.h	H	H
अँ		m̐			.N		~	az
ऽ	'	'		'	.a	.a	'	Z

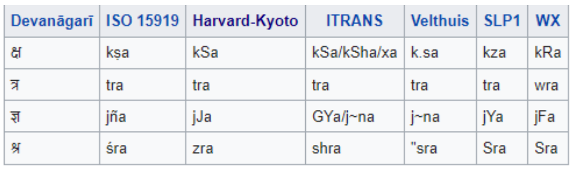
Figure 1. Different Phonetic schemes for Devanagari vowels [10]



Devanāgarī	IAST	ISO 15919	Monier-Williams72	Harvard-Kyoto	ITRANS	Velthuis	SLP1	WX
क	ka	ka	ka	ka	ka	ka	ka	ka
ख	kha	kha	kha	kha	kha	kha	Ka	Ka
ग	ga	ga	ga	ga	ga	ga	ga	ga
घ	gha	gha	gha	gha	gha	gha	Ga	Ga
ङ	ṅa	ṅa	n·a	Ga	~Na	"na	Na	fa
च	ca	ca	ća	ch	cha	ca	ca	ca
छ	cha	cha	ćha	chha	Chha	chha	Ca	Ca
ज	ja	ja	ja	ja	ja	ja	ja	ja
झ	jha	jha	jha	jha	jha	jha	Ja	Ja
ञ	ña	ña	ṅa	Ja	~na	~na	Ya	Fa
ट	ṭa	ṭa	ṭa	Ta	Ta	.ta	wa	ta
ठ	ṭha	ṭha	ṭha	Tha	Tha	.tha	Wa	Ta
ड	ḍa	ḍa	ḍa	Da	Da	.da	qa	da
ढ	ḍha	ḍha	ḍha	Dha	Dha	.dha	Qa	Da
ण	ṇa	ṇa	ṇa	Na	Na	.na	Ra	Na
त	ta	ta	ta	ta	ta	ta	ta	wa
थ	tha	tha	tha	tha	tha	tha	Ta	Wa
द	da	da	da	da	da	da	da	xa
ध	dha	dha	dha	dha	dha	dha	Da	Xa
न	na	na	na	na	na	na	na	na
प	pa	pa	pa	pa	pa	pa	pa	pa
फ	pha	pha	pha	pha	pha	pha	Pa	Pa
ब	ba	ba	ba	ba	ba	ba	ba	ba
भ	bha	bha	bha	bha	bha	bha	Ba	Ba
म	ma	ma	ma	ma	ma	ma	ma	ma
य	ya	ya	ya	ya	ya	ya	ya	ya
र	ra	ra	ra	ra	ra	ra	ra	ra
ल	la	la	la	la	la	la	la	la
व	va	va	va	va	va/wa	va	va	va
श	śa	śa	śa	za	sha	"sa	Sa	Sa
ष	ṣa	ṣa	sha	Sa	Sha	.sa	za	Ra
स	sa	sa	sa	sa	sa	sa	sa	sa
ह	ha	ha	ha	ha	ha	ha	ha	ha

Figure 2. Different Phonetic schemes for Devanagari consonants[10]

We used following phonetic schemes:
ka: "क", ca: "क", kha: "ख", ga: "ग", gha: "घ", Ga: "ङ", cha: "च", chha: "छ", xa: "छ", ja: "ज", jha: "झ", Ya: "ञ", Ta: "ट", Tha: "ठ", Da: "ड", Na: "ण", ta: "त", tha: "थ", da: "द", dha: "ध", na: "न", pa: "प", pha: "फ", fa: "फ", ba: "ब", bha: "भ", ma: "म", ya: "य", ra: "र", la: "ल", wa: "व", va: "व", sha: "श", sa: "स", Sa: "ष", ha: "ह", ksha: "क्ष", tra: "त्र", gya: "ज्ञ", a: "अ", aa: "आ", i: "इ", ii: "ई", u: "उ", uu: "ऊ", e: "ए", ai: "ऐ", ri: "ऋ", o: "ओ", au: "औ", am: "अं"



Devanāgarī	ISO 15919	Harvard-Kyoto	ITRANS	Velthuis	SLP1	WX
क्ष	kṣa	kSa	kSa/kSha/xa	k.sa	kza	kRa
त्र	tra	tra	tra	tra	tra	wra
ज्ञ	jña	jJa	GYa/j~na	j~na	jYa	jFa
श्र	śra	zra	shra	"sra	Sra	Sra

Figure 3. Different Phonetic schemes for irregular consonant clusters[10]

Steps for Transliteration:
Step 1: START
Step 2: Get word as input
Step 3: Initialization,
mapping_table, i=0,matras,direct_mapping_table,output=""
Step 4: IF word in the direct_mapping_table THEN
output=direct_mapping_table[word]
GOTO Step 8
Step 5: IF len_of_word-i>=3 THEN
token=3 characters of word from ith index
ELSE
token=all characters of word from ith index
Step 6: IF token in the mapping_table THEN
output=output+mapping_table[token]
i=i+length_of_token
IF matras exist THEN
append on output
IF token not in the mapping_table THEN
token=all characters of token except ith index
Step 7: Repeat Step 4 and Step 6 Untill we get token and i>length_of_word
Step 8: END

B. Error Detection and Correction

For error detection, when each word is typed, it is compared with all the tokens in the look up dictionary using counting bloom filter. If the input word is present in the lookup dictionary, the same word (i.e. the correct word) is returned. If the word is not present,the error is detected.

1) Dictionary Lookup: A dictionary is a source that contains list of correct words of a particular language. The non-word errors can be easily detected by checking each word against a dictionary. These resources are used for preparing, processing and managing linguistic information and knowledge needed for the computational processing of natural language. In this system, either we gather collection of Nepali word from Nepali Dictionary or use Nepali WordNet.

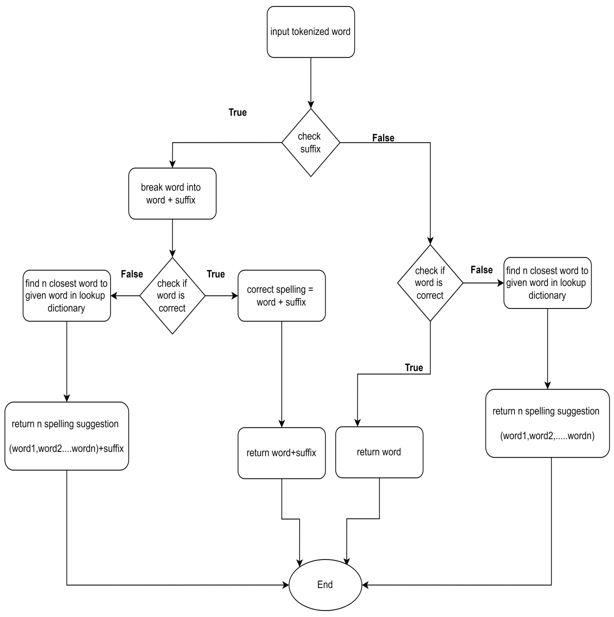

input tokenized word
True
check suffix
False
break word into word + suffix
False
check if word is correct
True
find n closest word to given word in lookup dictionary
correct spelling = word + suffix
check if word is correct
False
find n closest word to given word in lookup dictionary
True
return n spelling suggestion (word1,word2....wordn)+suffix
return word+suffix
return word
return n spelling suggestion (word1,word2,.....wordn)
End

Figure 4. Error Detection and Correction

2) Counting Bloom Filter: Counting Bloom Filters are a special type of a data structure that can tell us whether an item is present in the given set or not. This data structure has an excellent constant time complexity as O(k) where ’k’ is the number of hash functions. They are also very memory efficient since only minimal data is stored. The reason we use counting bloom filters is to see if each of the words that is inputted is in the initialized dictionary. In other words, we are checking if the word has been correctly spelled. If the word is correctly spelled, no more steps needs to be taken and we can simply return the word without any modifications. In our implementation, it will return True. But, if the word is not found, this guarantees that the word is misspelled because false negatives are not seen in counting bloom filters. So, we return False and further processing occurs. In this study, the hash functions that we will be using for Counting Bloom Filter are mentioned below:

- MD5 Hash
  This hash function accepts sequence of bytes and returns 128 bit hash value, usually to check data integrity but has security issues.
- SHA 1 Hash
  In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard.
- SHA 224 Hash
  SHA-224 is a one-way hash function that provides 112 bits of security, which is the generally accepted strength of Triple-DES [3DES]. This document makes the SHA-224 one-way hash function specification available to the Internet community, and it publishes the object identifiers for use in ASN.
- SHA256 Hash
  SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. The 256-bit key makes it a good partner-function for AES. It is defined in the NIST (National Institute of Standards and Technology) standard ‘FIPS 180-4’. NIST also provide a number of test vectors to verify correctness of implementation
  If the word is not present in the lookup dictionary, then Levenshtein distance with each word in the lookup dictionary is calculated. To find the possible correct word, a list of words with Levenshtein distance greater than 0.75 is created. From that list, longest common subsequence between the input token and the created list is found.After this, the word with longest common subsequence is returned as the correct word.

3) Levenshtein Distance: Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. Suppose if str1=”RAM” and str2=”ROM” then levenshtein distance between str1 and str2 is 1.

C. Bidirectional Encoder Representations from Transformers(BERT)

BERT stands for Bidirectional Encoder Representation from Transformer.

- Encoder Representations: Language modelling system, pretrained with unlabeled data. Then Fine Tuning.
- From Transformers: based on powerful NLP Algorithm. Defines the architecture of BERT.
- Bidirectional: uses with left and right context when dealing with a word. Defines the training process.

It is the state-of-the-art embedding model published by Google. It has created a major breakthrough in the field of NLP by providing greater results in many NLP tasks, such as question answering, text generation, sentence classification, and many more. BERT is a language model which is first trained to produce word features/word embeddings. These features are rich, context aware and bidirectional. After pre-training, we may use these features and fine-tune the model on our actual task (downstream task). BERT is completely

based on Transformer Architecture. It removes the transformer decoder and formed by multi layers of the transformer encoder.Though there is huge pile of text data available online, if we want to split the data to perform a specific task, then we are only left with fraction of human labelled dataset. But for the deep learning NLP models to perform well, we need hundreds and thousands of data. To overcome this problem, the concept of training general purpose language representation models on huge amount of unlabeled text data from internet was introduced. Then we can fine-tune this general purpose pretrained Bert model on specific task. This approach results in better accuracy compared to training on small task specific dataset from scratch. Bert is one of the latest advancements in the field of NLP pre-training.The core idea behind BERT is non-directional approach. The language models before BERT process a text sequence during training either from left-to-right or combined left-to-right and right-to-left. However, BERT looks in both direction and uses the full context of the sentence, both left and right at the "same time". This "same-time part" makes the BERT non-directional. But why this non-directional approach is powerful? Pretrained language models are of 2 types: Context Free and Context-based. For context free models like Word2vec, they always generate single word embedding for each word in a vocabulary. For example: the word साँचो in two sentences "सधै साँचो बोल्नु पर्छ" and "घरको साँचो कहाँ छ? " have different meaning however they will have same embedding vector. On the other hand, context-based models generate embedding representation of each word based on other word in a sentence. The word साँचो in sentence "सधै साँचो बोल्नु पर्छ" refers to Truth and in the sentence "घरको साँचो कहाँ छ? " it refers to key. As BERT represents the word साँचो using both previous and next context "सधै --- बोल्नु पर्छ", it will have different representation in different sentence. This makes BERT deeply bidirectional.
BERT is trained using two tasks:

1) Masked Language Modelling (MLM): MLM objectives: Bi-directionality and context awareness
Before feeding input sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the masked words, based on the context provided by the other non-masked words in the sequence. Then loss function calculates loss by considering only the prediction of the masked values and ignores the prediction for the non-masked words.

D. Next Sentence Prediction (NSP)

NSP objectives: To model relationships between sentences

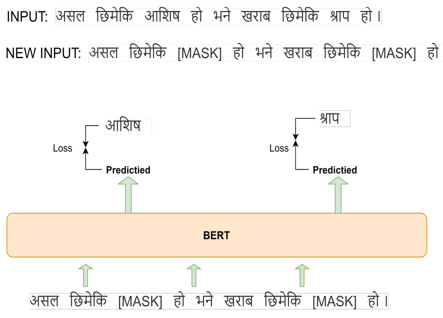

INPUT: असल छिमेकि आशिष हो भने खराब छिमेकि श्राप हो।
NEW INPUT: असल छिमेकि [MASK] हो भने खराब छिमेकि [MASK] हो।
आशिष
Loss
Predictied
श्राप
Loss
Predictied
BERT
असल छिमेकि [MASK] हो भने खराब छिमेकि [MASK] हो।

Figure 5. Masked Language Modelling.

To model relation between two sentences, BERT uses NSP at the training process. At training process BERT gets a pair of sentences and it learns whether the second sentence is next sentence or not. At training process model is fed with two input sequences where:

- 50 % of the time the next sentence is second sentence
- 50 % of the time next sentence is a random sentence form the corpus

Now BERT is required to predict whether the second sentence is random or not. To help the model distinguish between two sentences in training, the input is processed following way before entering the model.

1) Token Embeddings: A [CLS] token is added to beginning of each sentence and [SEP] token is added to end of each sentence.
2) Segment Embeddings: To distinguish between sentences, a marker indicating Sentence A or Sentence B is added to each token.
3) Positional Embeddings: A positional embedding is added to each sentence indicating its position in sentence.

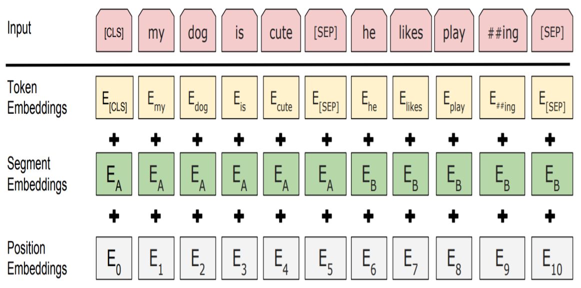

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 6. Processing of Input Sentence [7]

We take output of the [CLS] token which encodes the whole sequence, and add a feed-forward layer with a single output neuron (since we only have two classes: 1 and 0). This objective is usually trained by Binary Cross Entropy Loss.

E. Training of language model

The first and foremost step to train any language model is to collect a huge dataset. For the collection of a large volume of datasets, we used Nepali news sites. News scraping was performed on various Nepali news sites like Online Khabar, Setopati, Ratopati, etc. for the purpose of data collection. After the collection of the dataset, preprocessing was performed. Preprocessing of the dataset includes omission of special characters, symbols, etc. After the preprocessing was done, each line contains a single Nepali sentence. The second step for Bert Model Training is to select a suitable tokenizer. The available tokenizers are BertWordPieceTokenizer, ByteLevelBPETokenizer, CharBPETokenizer. Among the various available tokenizers, we selected BertWordPieceTokenizer. BertWordPieceTokenizer breaks words into sub-words to optimize vocabulary and cover most, if not all, possible textual occurrences of a word from outside the training text. And in BertWordPieceTokenizer, we set the parameter strip_accents=False if not, the tokenizer will remove the diacritics. For training, we used the training API of Hugging Face in which we set masked language modeling to 15% meaning, 15% of the input text corpus was randomly masked and the model tries to predict the masked words. In this way, around 70 lakhs of lines were used for the training of one epoch. The Bert model consists of 6 hidden layers and 12 attention heads.

IV. Next Word Prediction Using BERT

As we know that BERT is trained using masked language modelling (MLM), we can use this feature of BERT for task of next word prediction (NWP). After completing training of the BERT, it will be able to predict masked token effectively. So, for next word prediction, after each word prediction we will transfer the [MASK] token to end of the sentence. For example:
घरको [MASK]
घरको साँचो [MASK]
घरको साँचो कहाँ [MASK]
घरको साँचो कहाँ छ [MASK] …………

A. Part of Speech Tagging

Part of speech tagging is defined as process of categorizing words in a sentence corresponding with a particular part of speech based on word's definition and context. Parts of speech tags are properties of words that define its main context, function and usage in sentence. Example of part of speech tags are: Noun, Pronoun, Verb, Adverb, Adjectives, Conjunction and Preposition etc. Each word in a sentence will have a part of speech associated to it and it describes its function. It also describes relation of one word with other words in the sentence. Part of speech can also be used for making assumptions about semantics as the part of speech describes characteristics structure of lexical terms within a sentence. For part of speech tagging of Nepali language, we are using total 25 tags. Some of the major tags with their category definition is shown below: For the task of tags classification, we are fine tuning the BERT model which is trained on huge unlabeled Nepali text data. We are using the

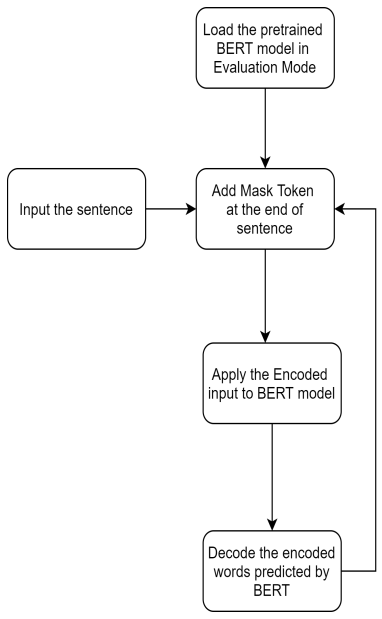
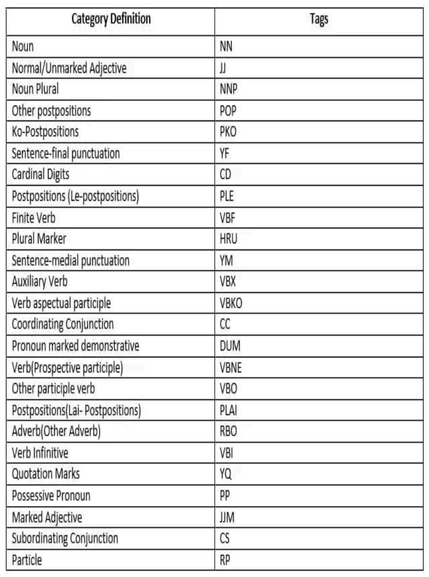
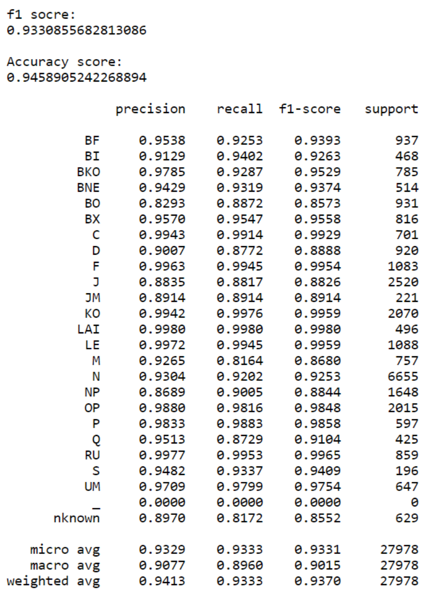

Load the pretrained BERT model in Evaluation Mode
Input the sentence
Add Mask Token at the end of sentence
Apply the Encoded input to BERT model
Decode the encoded words predicted by BERT

Figure 7. Next Word Prediction



Category Definition	Tags
Noun	NN
Normal/Unmarked Adjective	JJ
Noun Plural	NNP
Other postpositions	POP
Ko-Postpositions	PKO
Sentence-final punctuation	YF
Cardinal Digits	CD
Postpositions (Le-postpositions)	PLE
Finite Verb	VBF
Plural Marker	HRU
Sentence-medial punctuation	YM
Auxiliary Verb	VBX
Verb aspectual participle	VBKO
Coordinating Conjunction	CC
Pronoun marked demonstrative	DUM
Verb(Prospective participle)	VBNE
Other participle verb	VBO
Postpositions(Lai- Postpositions)	PLAI
Adverb(Other Adverb)	RBO
Verb Infinitive	VBI
Quotation Marks	YQ
Possessive Pronoun	PP
Marked Adjective	JJM
Subordinating Conjunction	CS
Particle	RP

Figure 8. POS Tagging



```
f1 socre:
0.9330855682813086

Accuracy score:
0.9458905242268894

              precision    recall  f1-score   support

          BF     0.9538    0.9253    0.9393       937
          BI     0.9129    0.9402    0.9263       468
         BKO     0.9785    0.9287    0.9529       785
         BNE     0.9429    0.9319    0.9374       514
          BO     0.8293    0.8872    0.8573       931
          BX     0.9570    0.9547    0.9558       816
           C     0.9943    0.9914    0.9929       701
           D     0.9007    0.8772    0.8888       920
           F     0.9963    0.9945    0.9954      1083
           J     0.8835    0.8817    0.8826      2520
          JM     0.8914    0.8914    0.8914       221
          KO     0.9942    0.9976    0.9959      2070
         LAI     0.9980    0.9980    0.9980       496
          LE     0.9972    0.9945    0.9959      1088
           M     0.9265    0.8164    0.8680       757
           N     0.9304    0.9202    0.9253      6655
          NP     0.8689    0.9005    0.8844      1648
          OP     0.9880    0.9816    0.9848      2015
           P     0.9833    0.9883    0.9858       597
           Q     0.9513    0.8729    0.9104       425
          RU     0.9977    0.9953    0.9965       859
           S     0.9482    0.9337    0.9409       196
          UM     0.9709    0.9799    0.9754       647
           _     0.0000    0.0000    0.0000         0
      nknown     0.8970    0.8172    0.8552       629

   micro avg     0.9329    0.9333    0.9331     27978
   macro avg     0.9077    0.8960    0.9015     27978
weighted avg     0.9413    0.9333    0.9370     27978
```

Figure 9. POS Classification Report

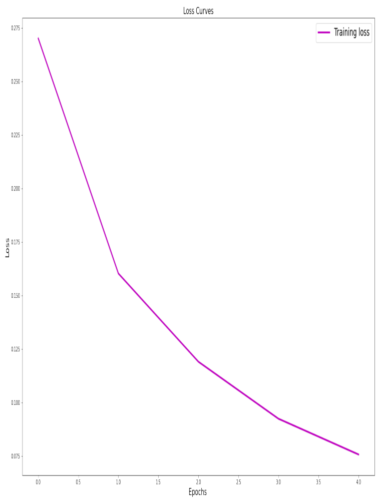


Figure 10. POS Loss Function

BERT model for this task because of its bidirectional context awareness characteristics which is useful while understanding the part of speech tag of each token and our labelled data for tags identification wont be that huge. BertForTokenClassification provides class for token-level predictions. BertForTokenClassification is a fine tuning model that wraps our pretrained BertModel and adds token-level classifier on top of our pretrained BertModel. The output of last hidden layer of sequence is passed to token-level classifier which is a linear layer.

V. RESULT

For transliteration, input is taken from user in Roman script and then Devanagari transliteration is performed i.e. when user types ”mero” we get मेरो as transliteration output.The transliterated word is checked for any error. If error is not detected, the word remains as it is. If error is detected, it corrects the errors.After transliteration when we get एकपलट as output it is detected as misspelled word. So for error correction words: "एकपल्ट", "एकपल", "एकपट", "एकैप-ल्ट", "एकपटक", "एकैपल", "एकपेट", "एपल", "पलट", "कपट" is suggested. After error detection and correction, next word is predicted according to the context.When काठमाडौं नेपालको is passed for next word prediction, we get words: "राजधानी", "सरकार", "छ", "राजनीति", "नाम" as output. Part of Speech recognition was performed after the completion of each sentence. When मेरो घर दोलखा हो । is passed for POS tagging we get "घर": ”Noun”, "दोलखा": ”Noun”,"मेरो": ”Pronoun”, "हो": ”Auxiliary Verb” as output. For the dataset for Part of Speech Tagging, we retrieved the dataset from [9] and the given dataset format was converted into CSV format. We performed fine tuning on pre-trained BERT model for the classification of POS. POS tagging of Nepali Text was carried out in a Nepali tagged corpus of tag size 25. Thus obtained validation accuracy was 94.45% and f1 score was 0.933. The POS tagging training loss function plot and classification report is shown in figure 9 and figure 10.

VI. CONCLUSION

For transliteration, we use direct mapping approach by using the direct mapping table which has around 40,000 mapping in addition to the existing Harvard-Kyoto Convention by J. Nair and A. Sadasivan [1] If the input word is not present in direct mapping table we proceed for algorithmic approach where we combine different Devanagari vowels and consonants mapping schemes which makes it easier to type for a user having zero knowledge of the mapping table.For example, if the user wants to type श, the user will type ”sha” if he/she has no knowledge of mapping instead of ”za”.

For error detection, our approach is better than approaches by A. Sharma and P. Jain [4] because A. Sharma and P. Jain uses sequential search, Big-O notation of a sequential search is $O(n)$.where, n refers to the number of the items. we use the Counting Bloom filter(CBF) to check whether the word is present in the lookup dictionary or not. The benefit of using CBF is that it has the constant time complexity $O(k)$ where k is the no of the hash function used and it is memory efficient.

For a particular word there can a number of suffixes appended to it, for instance घरतिर, घरमाथि, घरको, घर–

भित्र, etc. Therefore, it will be impractical to add each word along with its suffixes in the lookup dictionary. Impractical in the sense that the size of the lookup dictionary becomes very large which in turn increases the searching time in the dictionary. Keeping this into mind, we separated suffix from the main word and the spelling checking was performed only for the main word. This approach reduces the number of words on the lookup dictionary.

For next word prediction and POS tagging we use BERT. We used a pre-trained BERT model which is trained using a limited dataset for a single epoch which implies that if we have a better model trained using a large volume of dataset, the accuracy of POS tagging can be further increased.

ACKNOWLEDGEMENT

It is a great honor to convey special thanks to all those helping hands without whom it would be hard for us to do this research. This research has been possible with patience, support and, encouragement that we have received from many personalities of our institution. First, we would like to acknowledge our college Kantipur Engineering College for providing the opportunity to carry out this study.

REFERENCES

[1] J. Nair and A. Sadasivan, “A Roman to Devanagari Back-Transliteration Algorithm based on Harvard-Kyoto Convention,” in 2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019, 2019.
[2] R. Kumar, M. Bala, and K. Sourabh, “A study of spell checking techniques for Indian Languages,” Tech. Rep. 1, 2018.
[3] A. Pal and A. Mustafi, “Vartani Spellcheck – Automatic Context-Sensitive Spelling Correction of OCR-generated Hindi Text Using BERT and Levenshtein Distance,” arXiv, 2020.
[4] A. ’Sharma’ and P. ’Jain’, “Hindi Spell Checker,” 2013.
[5] T. Wolf and V. Sanh, “HuggingFace’s Transformers: State-of-the-art Natural Language Processing,” 2019
[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, vol. 2017-December, 2017.
[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, vol. 1, 2019.
[8] Sayami, Sarbin, Tej Bahadur Shahi, and Subarna Shakya. Nepali POS Tagging Using Deep Learning Approaches. No. 2073. EasyChair, 2019.
[9] Center for Language Engineering, ’POS Tagged Nepali Corpus’, 2010 [Online]. Available: https://cle.org.pk/software/ling_resources/UrduNepaliEnglishParallelCorpus.htm. [Accessed: 18-Feb- 2022].
[10] [’Devanagari transliteration - Wikipedia’, En.wikipedia.org, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Devanagaritransliteration. [Accessed: 03- Mar- 2022].