Major Project Report On

Parts of Speech Tagger for Nepali Language

By

Mohnish Keeni (Reg. No.-201900167)

and

Rishabh Prasad (Reg. No.-201900097)

In partial fulfilment of requirements for the award of degree in
Bachelor of Technology in Computer Science and Engineering
(2023)

Under the Project Guidance of

Dr. Yumnam Nirmal
Assistant Professor

Ms. Nitisha Pradhan
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

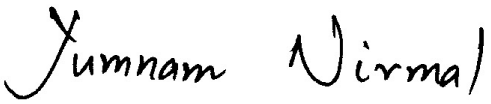
SMIT SIKKIM MANIPAL UNIVERSITY
SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY

(A constituent college of Sikkim Manipal University)

MAJITAR, RANGPO, EAST SIKKIM – 737136

PROJECT COMPLETION CERTIFICATE

This is to certify that **Mohnish Keeni** and **Rishabh Prasad** of Computer Science and Engineering Department of Sikkim Manipal Institute of Technology (SMIT) bearing registration numbers **201900167** and **201900097** respectively have worked under my guidance from January 2023 and has successfully completed the project entitled "**Parts of Speech Tagger for Nepali Language**" in partial fulfilment of the requirements for the award of Bachelor of Technology.

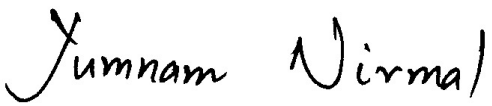


Dr. Yumnam Nirmal
Assistant Professor
CSE Department
SMIT
Majitar, East Sikkim – 737136

Ms. Nitisha Pradhan
Assistant Professor
CSE Department
SMIT
Majitar, East Sikkim – 737136.

Project Review Certificate

This is to certify that the work recorded in this project report entitled "**Parts of Speech Tagger for Nepali Language**" has been jointly carried out by **Mohnish Keeni** and **Rishabh Prasad** bearing registration numbers **201900167** and **201900097** respectively of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering. This report has been duly reviewed by the undersigned and recommended for final submission for Major Project Viva Examination.



Dr. Yumnam Nirmal
Assistant Professor
CSE Department
SMIT
Majitar, East Sikkim – 737136

Ms. Nitisha Pradhan
Assistant Professor
CSE Department
SMIT
Majitar, East Sikkim – 737136.

Certificate of Acceptance

This is to certify that the below mentioned student(s) of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology (SMIT) have worked under the supervision of Dr. Yumnam Nirmal and Ms. Nitisha Pradhan from 10 Jan 2023 to 10 May 2023 on the project entitled "Parts of Speech Tagger for Nepali Language"

The project is hereby accepted by the Department of Computer Science & Engineering, SMIT in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student(s)	Project Venue
201900167	Mohnish Keeni	SMIT
201900097	Rishabh Prasad	SMIT

……

Dr. Udit Kumar Chakraborty
Professor & HOD
Computer Science & Engineering Department
Sikkim Manipal Institute of Technology
Majhitar, Sikkim – 737136

Declaration

I, the undersigned, hereby declare that the work recorded in this project report entitled "**Parts of Speech Tagger for Nepali Language**" in partial fulfilment for the requirements of award of B.Tech (CSE) from Sikkim Manipal Institute of Technology (A constituent college of Sikkim Manipal University) is a faithful and bonafide project work carried out at "**Sikkim Manipal Institute of Technology**" under the supervision and guidance of **Dr. Yumnam Nirmal and Ms. Nitisha Pradhan** of **Computer Science and Engineering Department, Sikkim Manipal Institute of Technology .**

The results of this investigation reported in this project have so far not been reported for any other Degree / Diploma or any other technical forum.

The assistance and help received during the course of the investigation have been duly acknowledged.

Mohnish Keeni (Reg: 201900167) Rishabh Prasad (Reg: 201900097)

Acknowledgement

I wish to express my sincere thanks to **Dr. Yumnam Nirmal, Assistant Professor, CSE Department, SMIT** and **Ms. Nitisha Pradhan, Assistant Professor, CSE Department, SMIT** for providing me an opportunity to carry out project work on "Title" and their unlisted encouragement and guidance carrying out this project work.

I would like to express my sincere thanks to **(Prof.) Dr. Udit Kumar Chakraborty**, HOD, Computer Science and Engineering Department for allowing me to carry out my project from **10th January 2023 to 10th May 2023** and valuable support and guidance during the project period.

I would like to express my humble gratitude to **Mr. Biswaraj Sen, Professo**r, **Mr. Santanu Kumar Misra, Associate Professor, Mr. Saurav Paul, Assistant Professor-I, and Mrs Chitrapriya N., Assistant Professor-I,** Project Coordinators, Computer Science and Engineering Department, Sikkim Manipal Institute of Technology for their unlisted encouragement and their timely support and guidance till the completion of the project work.

Lastly, I wish to avail myself of this opportunity, express a sense of gratitude and love to all our teachers, staff of the department of Computer Science and Engineering Department, friends and fellow for their support and help.

Mohnish Keeni (Reg: 201900167) Rishabh Prasad (Reg: 201900097)

DOCUMENT CONTROL SHEET

1	Report No	CSE/Major Project/ In House/B.Tech/IH5/2023
2	Title of the Report	Parts of Speech Tagger for Nepali Language
3	Type of Report	Technical
4	Author	Mohnish Keeni and Rishabh Prasad
5	Organizing Unit	Sikkim Manipal Institute of Technology
6	Language of the Document	English
7	Abstract	The webpage allows users enter to raw Nepali sentences and get Parts-of-Speech annotated sentences as output.
8	Security Classification	Generals
9	Distribution Statement	General

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Parts of Speech (POS) tagging is a critical task in Natural Language Processing (NLP), as it involves identifying and labeling words in a sentence as nouns, verbs, adjectives, adverbs, etc.

This project was focused on the development of a POS (Part of Speech) tagging system for the 'Nepali' language, which is the official language of the country of Nepal [1].

The project was built using the Python programming language and many of its libraries, as well as ReactJS [2] and Flask [3] for the user interface of the application. It starts with the collection and pre-processing of a corpus of Nepali text. This corpus is then used to train a Bi-LSTM (Bidirectional - Long Short Term Memory) model for POS tagging. The model was evaluated using various performance metrics, such as accuracy and F1 score.

The corpus used to train and test the model was the Nepali National Corpus [4] which consists of 14.5 million words from books, newspaper articles and web-text. This corpus is parts-of-speech annotated with the Nelralec tag-set [5] which consists of 112 unique tags.

The trained model was found to have an accuracy of 98.6% and performs well for compound words (words made up of two or more words) which was an area where prior taggers were found lacking.

POS tagging system finds a wide range of application, that serves as a crucial tool for Nepali Language Processing and for the development of various applications such as translation, classification and many others.

Furthermore, this project demonstrates the feasibility of using Python and its existing libraries to develop NLP solutions for computationally under-resourced languages like Nepali. It is also an important step towards building more comprehensive NLP systems for Nepali, which could have a significant impact on the Nepali language community.

Developing a robust POS tagging system with high accuracy, shall open new avenues for NLP research and applications in the Nepali language.

1. INTRODUCTION

1.1 General Overview of the Problem

Parts of Speech (POS) tagging is a task in the field of language processing involving labelling words in a sentence as different parts of speech, such as nouns, verbs, adjectives, and adverbs.

For example, labelled sentences have been mentioned below:

• "**बाजेलाई नाम भन्नुहोस्**", the word "**बाजे**" would be labelled as "noun", the word "**लाई**"as a "postposition" the word "**नाम**" as a "noun", and " **भन्नुहोस्**" as a " verb" .

• "**The cat is sleeping on the couch**", the words "**cat**" and "**couch**" would be labelled as nouns, "**is**" as a verb, and "**sleeping**" as an adjective.

In this project, the focus was to create a system that can automatically label the parts of speech for sentences written in the Nepali language. To do this, our system has used a large collection of annotated Nepali sentences and the Bidirectional-Long Short Term Memory (Bi-LSTM) machine learning model to learn how to identify the parts of speech.

The goal of the project was to make it easier for computers to understand the Nepali language and improve the accuracy of language-related tasks, such as machine translation and text classification. By successfully creating a Parts of Speech tagging system for Nepali, our project also helps to pave the way for future language processing tools for this under-resourced language.

1.2 Literature Survey:

The paper by **Tej Bahadur Shahi [6]** describes the development of a Support Vector Machine (SVM) based Part-of-Speech (POS) tagger for the Nepali language. The SVM approach to POS tagging is efficient, portable, scalable and trainable, and has been successful in text classification. The authors develop the SVM tagger using a rich feature set to model the characteristics of the Nepali language, which is morphologically rich. The SVM-based tagger is compared to rule-based and hidden Markov Model (HMM) approaches, which cannot handle many features of the Nepali language. The study concludes that the SVM tagger provides attainable accuracy for Nepali language processing tasks. One drawback of SVM taggers is that they are slower in training than other taggers.

In another paper by **Archit Yajnik [7]** presents a study on Part of Speech (POS) tagging for Nepali text using the Hidden Markov Model (HMM) and the Viterbi algorithm. The study found that the Viterbi algorithm is computationally faster and more accurate than HMM, achieving 95.43% accuracy. The article provides a brief overview of HMM and Viterbi algorithm and elaborates on the methodology and experimental details. The article concludes with a discussion on the mismatches that took place during the error analysis. The HMM based tagger was found to have a very low accuracy (76.97%). The Viterbi algorithm, while more accurate than the HMM based tagger, wrongly tagged tokens belonging to NNP class as NN in 35 out of 48 occurrences.

Another paper by **Archit Yajnik [8]** presents a novel algorithm for Part-of-Speech (POS) tagging in Nepali text using Artificial Neural Networks (ANNs). The algorithm uses three ANN techniques (Radial Basis Function, General Regression Neural Networks, and Feedforward Neural Network) and extracts features from the marginal probability of a Hidden Markov Model. The article compares the results of all three techniques using two annotated tagged sets for training and testing and finds that the GRNN based POS tagging technique produces the best results with 100% and 98.32% accuracy for both training and testing sets respectively. The article is divided into five sections, including a brief of traditional ANN architectures and an explanation of the GRNN technique. A drawback of this approach is that it is not context accurate as both training and test database does not contain the words with multiple tags i.e., same Nepali word with more than one different tag.

In the paper by **Greeshma Prabha [9]**, a deep learning-based POS tagger for the Nepali language is proposed. The authors aim to build multiple taggers using RNN, LSTM, GRU, and their bidirectional variants, evaluated them based on accuracy, precision, recall, and F1-score, and compare the performance of the taggers based on those metrics. The results showed that the Bi-LSTM based model outperforms the others with over 99% accuracy. A drawback to this work is that the Corpus used as a dataset was tagged using a tagset comprising of only 43 tags which is inferior in comparison to the Nelralec tagset [5] with 112 tags which is capable producing finer results as a greater number of contexts are considered.

1.3 Problem Definition

- Nepali Language being computationally under-resourced language lacks robust Parts of Speech (POS) tagging system with good accuracy specifically for compound words and ambiguous words.

- POS tagging is a fundamental task in Natural Language Processing, and it involves identification of various grammatical category of each word in a sentence like noun, verbs, pronouns etc.

- This information is crucial for a variety of applications such as text-to-speech synthesis, machine translation and information retrieval. However, there is currently a lack of high-quality POS tagging systems for Nepali, which limits the potential of Natural Language Processing in this language.

- The Nepali language also presents difficulties with compound words. Compound words are made up of two or more individual words, and their meaning is not always straightforward from the individual words. This makes POS tagging for compound words challenging.

- Current systems lack a user interface and have a limitation in the size of their vocabulary.

- Moreover, currently developed POS taggers have not been deployed and made available to use for further development of NLP applications for the Nepali language.

1.4 Analysis of the problem and the SRS

The Nepali language lacks a robust and accurate Parts of Speech (POS) tagging system, particularly for compound words and ambiguous words. POS tagging is a fundamental task in Natural Language Processing that involves identifying various grammatical categories of each word in a sentence, such as nouns, verbs, and pronouns. This information is critical for a variety of applications, including text-to-speech synthesis, machine translation, and information retrieval.

However, the current lack of high-quality POS tagging systems for Nepali hinders the potential of Natural Language Processing in this language. The Nepali language presents further difficulties with compound words and words that can have multiple classes in different contexts. Compound words are made up of two or more individual words, and their meaning is not always straightforward from the individual words, making POS tagging for compound words challenging. Some words in Nepali can also have multiple classes depending on their context, further complicating the POS tagging process.

Complex NLP tasks such as sentiment analysis require a very fine understanding of context which is not provided by most of the currently available taggers which have fewer number of tags in the tag-set.

Furthermore, the lack of a user interface and limited vocabulary size in current Nepali POS taggers also hinders their practical application. A user interface can improve the accessibility of the tool and enable users to easily input text and view the output. Moreover, a limited vocabulary size restricts the range of words that the POS tagger can accurately tag, which is a significant limitation for practical use. Developing a POS tagger with an intuitive user interface and an expanded vocabulary can greatly enhance its usability and effectiveness for a variety of NLP applications.

Addressing these issues requires developing an accurate and reliable POS tagging system for Nepali that can handle compound words and ambiguous words. Additionally, developing a user-friendly interface and expanding the vocabulary of the system could improve its usability. Finally, deploying the system and making it available for further development of NLP applications for Nepali could be crucial for future advancements in this field.

Functional Requirements:

- Function name: POS Tagging

  Description: The system should will able to precisely recognize and label the different grammatical categories of each word in a sentence. For instance, it should correctly identify whether a word is a noun, verb, adjective, adverb, pronoun, preposition, conjunction, or interjection in a sentence.

  Input: Nepali sentence

  Output: Labelled grammatical categories of each word.

- Function Name: User Interface
  Description: With the help of an interface, users can input Nepali sentences into the system.

  Input: Nepali sentence

  Output: POS tagged sentence.

- Function Name: POS tagging of Compound Words
  Description: The system should be able to deal with compound words effectively, ensuring that their meaning is preserved without errors or confusion.

  Input: Nepali sentence having compound words

  Output: POS tagged sentence.

- Function Name: POS Tagging of Unknown Words
  Description: The system must be equipped with the capability to manage unknown words, ensuring that they are not overlooked or incorrectly processed due to being out of the system's vocabulary.

  Input: Nepali sentence having words that are not in dataset.

  Output: POS tagged sentence.

- Function name: Downloading output as a text file
  Description: A button present when on click User will to be able to retrieve the POS tagged output in the form of a text file via download, which can be easily shared and further processed as necessary.

  Input: Click on Download button.

Output: POS tagged output in the form of a text file.

- Function name: Deployability of Solution
  Description: The system should be employable for practical use in NLP applications for Nepali Language Processing.

  Input: Nepali language text data for NLP applications.

  Output: Accurate and efficient processing of the input data, suitable for various NLP applications.

Non Functional Requirements:

- Performance: The web server must be able to handle and support multiple instances of the webpage. The time between request and reply should be less. Minimum time should be taken by the webpage to display information.
- Usability: To provide users with only requested and specific information.
- Availability: It is a webpage, so it will be available as long as the server is up and running.
- Interoperability: The webpage is interoperable on various web browsers thus increasing usability and flexibility.

1.5 Solution Strategy

- **Data Collection:** Collect a large annotated corpus of Nepali text such as news articles, books, and social media posts. Ensure that the corpus is diverse and represents various genres and styles of Nepali writing.
- **Data Pre-processing**: Clean and pre-process the corpus to remove any erroneous data. Convert the annotated corpus into a format that can be used for training and evaluation, such as a CSV file or a Pandas data frame. Split the corpus into train, validation, and test sets after converting it into a format which is suitable for the model to understand.
- **Model Development:** Define the architecture of the Bi-LSTM model for POS tagging in Nepali. Convert the words and tags in the corpus into numeric values using word embeddings. Train the Bi-LSTM model on the training set using the Keras library in Python. Evaluate the performance of the model on the validation set and tune the hyperparameters to improve its accuracy.
- **Model Evaluation:** Evaluate the performance of the trained model on the test set using metrics such as accuracy, precision, recall, and F1 score. Identify the errors made by the model and analyse them to improve its accuracy.
- **User Interface Development:** Develop a user interface using ReactJS for the front-end, Flask for the back-end, and Node.js for server-side scripting. The user interface should allow users to input Nepali text and receive the corresponding POS tags predicted by the Bi-LSTM model. The interface should be user-friendly, intuitive, and responsive to ensure a seamless experience for the users.
- **Deployment:** The final step will be to deploy the developed model for practical use in NLP applications for Nepali language processing. The system will have a simple and user-friendly interface to make it easy for users to deploy and use.

1.6 Preliminary User's Manual

The webpage contains an input textbox, a submit button and a download text file button.

Fig 1: Webpage when first loaded by user

When a user enters a Nepali sentence in the textbox and presses the submit button, the sentence is tagged and returned to the user as shown below:

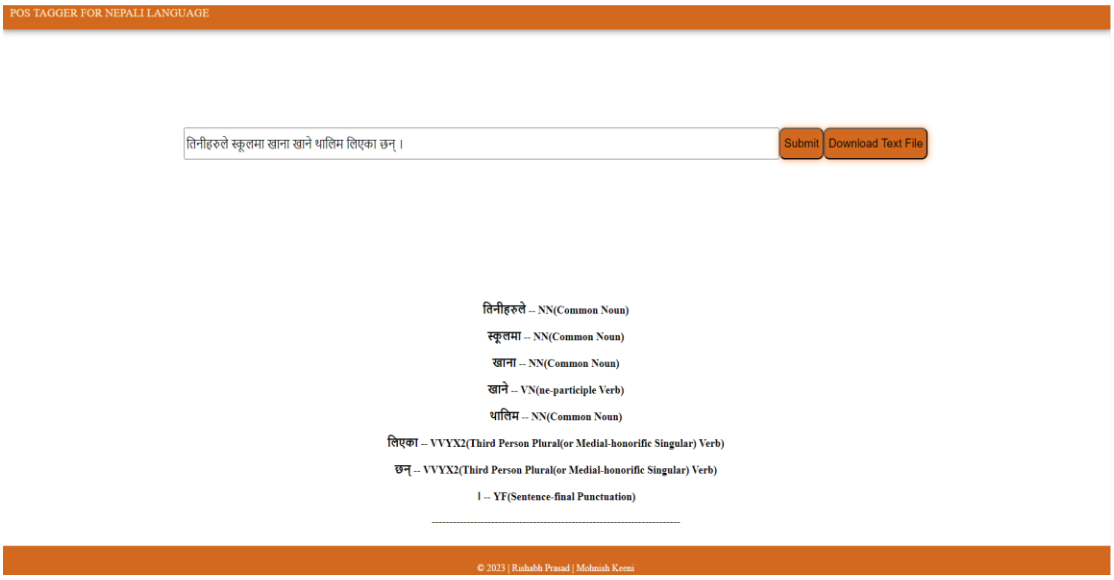


Fig 2: Webpage after user enters text and presses the submit button

After receiving the tagged output when a user clicks the download text file button a text file containing the output is generated and downloaded on the users machine as shown below:

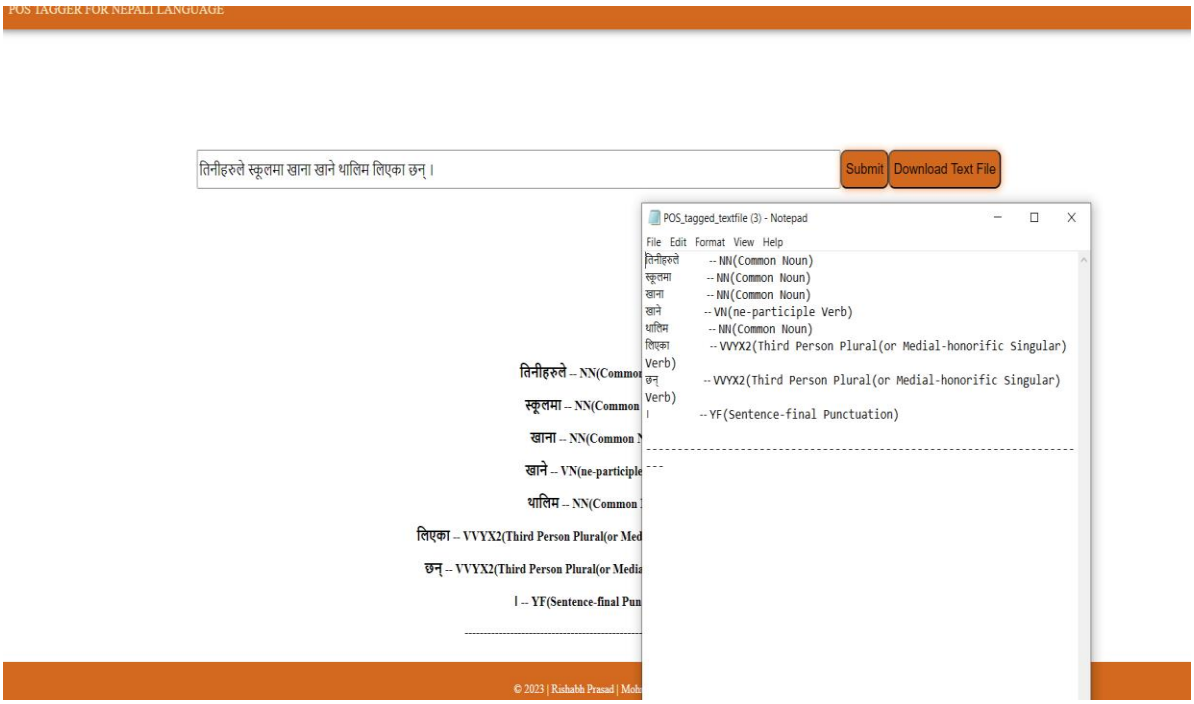


Fig 3: Webpage after user enters text and clicks the download file button.

1.7 Organization of the report

The problem is discussed in detail in section 1 along with the planned approach to arrive at a viable solution to it.

The next section that is section 2 shows various design diagrams for the system.

The 3rd section describes the dataset used to train and validate the model as well as the tagset used to perform Parts-of-Speech tagging.

The 4th section shows various test plans and response obtained during the testing phase. It mainly highlights those test cases that generate an error condition for various wrong input provided by user.

Section 5 discusses the various implementation processes applied for the development of the model and the webpage. It also highlights the technologies that have been used and for what purpose in the development process.

The 6th section highlights the various results that are obtained after analysis is done.

The last section that is section 7 discuss about the achievements obtained, the problems encountered and how it was tackled, future scope and the limitations.

2. DESIGN STRATEGY FOR SOLUTION

2.1 Block Diagram:

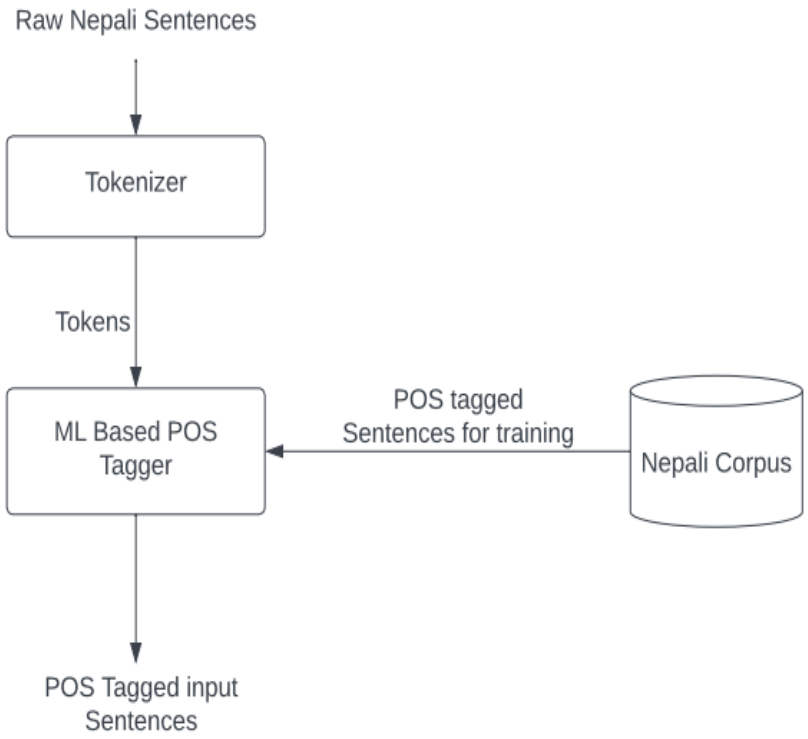


Fig 4: Block Diagram for Parts-of-Speech tagger

The ML based POS tagger was trained on POS annotated sentences from the Nepali National Corpus dataset and the trained tagger was then used to predict POS tags of tokens in raw Nepali sentences. The raw Nepali sentences were tokenized before passing them to the model for tagging.

2.2 Architecture of Neural Network:

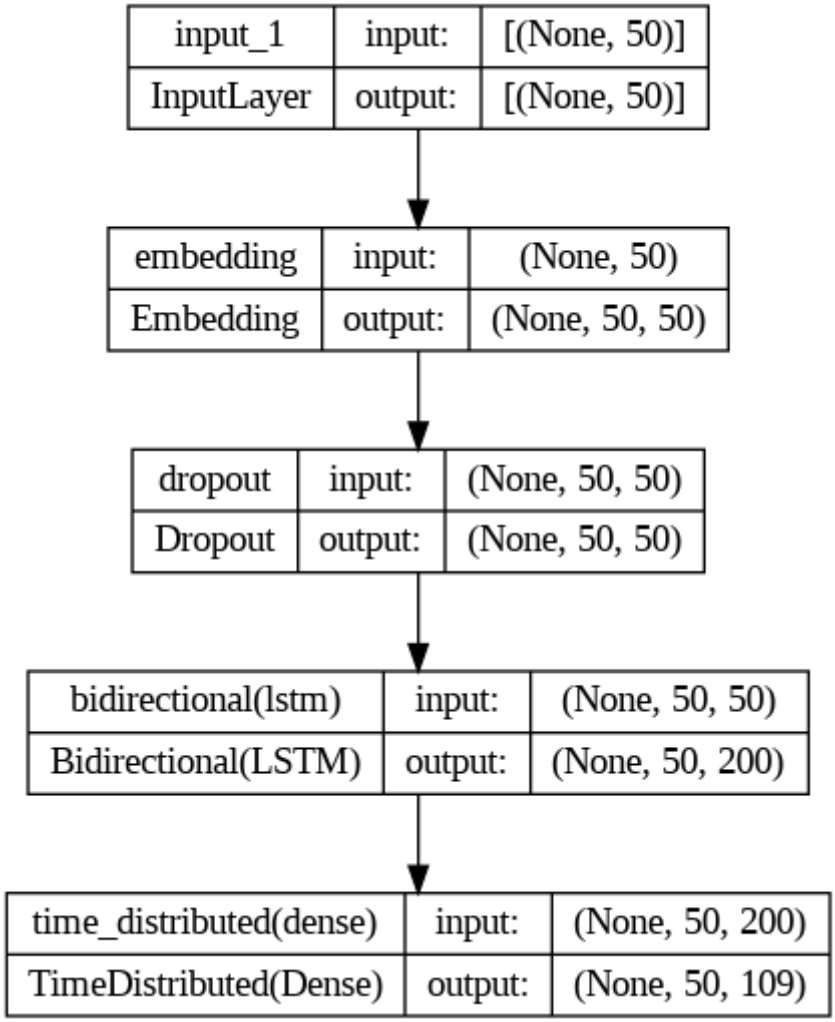


Fig 5: Architecture of Neural Network

The above figure illustrates the input shape and output shape of each layer in the neural network.

2.3 Flowchart of Webpage:

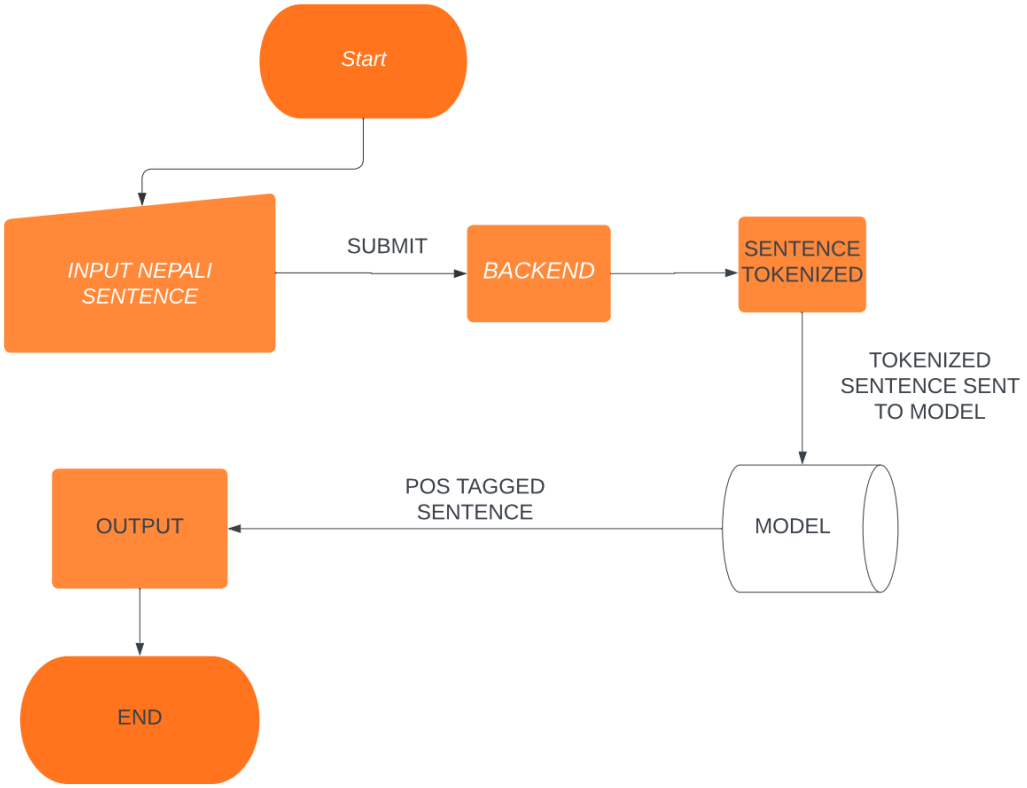


Fig 6: Flow Chart of Webpage

When Nepali text is entered in the text box and submit button is pressed, the Flask based backend creates an array containing sentences in the text and then tokenizes each sentence. The tokenized sentences are then passed to the model for tagging and the POS tagged output is sent to the ReactJS based frontend to be displayed.

3. CORPUS AND TAGSET

The Nepali National corpus is a large collection of Nepali texts comprising of approximately 14 million words extracted from books, webtext and news articles in XML format. The Nelralec project, which was sponsored by the Asia IT & C Program of the European Commission, established the corpus. The texts in the corpus were initially PoS tagged by Bal Krishna Bal of Language Technology Kendra and Andrew Hardie of Lancaster University. This corpus is tagged using the Nelralec tagset [5] which comprises of 122 unique tags. The tagset has been described in the table below:

CATEGORY	**CATEGORY DEFINITION**	**POS TAG**
NOUN	Common Noun	NN
	Proper Noun	NP
PRONOUNS	First Person Pronoun	PMX
	First Person Possessive Pronoun with Masculine Agreement	PMXKM
	First Person Possessive Pronoun with Feminine Agreement	PMXKF
	First Person Possessive Pronoun with Other Agreement	PMXKO
	Non-honorific Second Person Pronoun	PTN
	Non-honorific Second Person Possessive Pronoun with Masculine Agreement	PTNKM
	Non-honorific Second Person Possessive Pronoun with Feminine Agreement	PTNKF
	Non-honorific Second Person Possessive Pronoun with Other Agreement	PTNKO
	Medial-honorific Second Person Pronoun	PTM

	Medial-honorific Second Person Possessive Pronoun with Masculine Agreement	PTMKM
	Medial-honorific Second Person Possessive Pronoun with Feminine Agreement	PTMKF
	Medial-honorific Second Person Possessive Pronoun with Other Agreement	PTMKO
	High-honorific Second Person Pronoun	PTH
	High-honorific Unspecified-person Pronoun	PXH
	Royal-honorific Unspecified-person Pronoun	PXR
	Reflexive Pronoun	PRF
	Possessive Reflexive Pronoun with Masculine Agreement	PRFKM
	Possessive Reflexive Pronoun with Feminine Agreement	PRFKF
	Possessive Reflexive Pronoun with Other Agreement	PRFKO
PRONOUNS DERIVED USING -AI	First Person Possessive Pronoun without Agreement	PMXKX
	Non-honorific Second Person Possessive Pronoun without Agreement	PTNKX
	Medial-honorific Second Person Possessive Pronoun without Agreement	PTMKX

	Possessive Reflexive Pronoun without Agreement	PRFKX
PRONOUN DETERMINERS	Masculine Demonstrative Determiner	DDM
	Feminine Demonstrative Determiner	DDF
	Other-agreement Demonstrative Determiner	DDO
	Unmarked Demonstrative Determiner	DDX
	Masculine Interrogative Determiner	DKM
	Feminine Interrogative Determiner	DKF
	Other-agreement Interrogative Determiner	DKO
	Unmarked Interrogative Determiner	DKX
	Masculine Relative Determiner	DJM
	Feminine Relative Determiner	DJF
	Other-agreement Relative Determiner	DJO
	Unmarked Relative Determiner	DJX
	Masculine General Determiner-pronoun	DGM
	Feminine General Determiner-pronoun	DGF
	Other-agreement General Determiner-pronoun	DGO
	Unmarked General Determiner-pronoun	DGX
QUESTION MARKER	Question Marker	QQ
ADJECTIVE	Masculine Adjective	JM

	Feminine Adjective	JF
	Other-agreement Adjective	JO
	Unmarked Adjective	JX
	Sanskrit-derived Comparative or Superlative Adjective	JT
VERB	Infinitive Verb	VI
	Masculine d-participle Verb	VDM
	Feminine d-participle Verb	VDF
	Other-agreement d-participle Verb	VDO
	Unmarked d-participle Verb	VDX
	e(ko)-participle Verb	VE
	ne-participle Verb	VN
	Sequential Participle-converb	VQ
	Command-form Verb, Non-honorific	VCN
	Command-form Verb, Mid-honorific	VCM
	Command-form Verb, High-honorific	VCH
	Subjunctive/Conditional e-form Verb	VS
	i-form Verb	VR
	First Person Singular Verb	VVMX1
	First Person Plural Verb	VVMX2
	Second Person Non-honorific Singular Verb	VVTN1
	Second Person Plural(or Medial-honorific Singular) Verb	VVTX2
	Third Person Non-honorific Singular Verb	VVYN1

	Third Person Plural(or Medial-honorific Singular) Verb	VVYX2
	Feminine Second Person Non-honorific Singular Verb	VVTN1F
	Feminine Second Person Medial-honorific Singular Verb	VVTM1F
	Feminine Third Person Non-honorific Singular Verb	VVYN1F
	Feminine Third Person Medial-honorific Singular Verb	VVYM1F
	First Person Singular Optative Verb	VOMX1
	First Person Plural Optative Verb	VOMX2
	Second Person Non-honorific Singular Optative Verb	VOTN1
	Second Person Plural(or Medial-honorific Singular) Optative Verb	VOTX2
	Third Person Non-honorific Singular Optative Verb	VOYN1
	Third Person Plural(or Medial-honorific Singular) Optative Verb	VOYX2
ADVERB	Adverb	RR
	Demonstrative Adverb	RD
	Interrogative Adverb	RK
	Relative Adverb	RJ
POSTPOSITION	Postposition	IH
	Plural-collective Postposition	IH

	Ergative-instrumental Postposition	IE
	Accusative-dative Postposition	IA
	Masculine Genitive Postposition	IKM
	Feminine Genitive Postposition	IKF
	Other-agreement Genitive	IKO
NUMERALS	Cardinal Number	CD
	Masculine Ordinal Number	MOM
	Feminine Ordinal Number	MOF
	Other-agreement Ordinal Number	MOO
	Unmarked Ordinal Number	MOX
NUMERAL CLASSIFIERS	Masculine Numeral Classifier	MLM
	Feminine Numeral Classifier	MLF
	Other-agreement Numeral Classifier	MLO
	Unmarked Numeral Classifier	MLX
CONJUNCTIONS	Coordinating Conjunction	CC
	Subordinating Conjunction appearing after the clause it subordinates	CSA
	Subordinating Conjunction appearing before the clause it subordinates	CSB
PUNCTUATION	Sentence-final Punctuation	YF
	Sentence-medial Punctuation	YM
	Quotation Marks	YQ
	Brackets	YB

PARTICLE	Particle	TT
INTERJECTION	Interjection	UU
OTHERS	Foreign Word in Devnagari	FF
	Foreign Word not in Devnagari	FS
	Abbreviation	FB
	Mathematical Formula	FO
	Letter of the Alphabet	FZ
	Unclassifiable	UU
NULL TAG	Null Tag	NULL

Table 1: Nelralec Tagset[]

Our model was trained, validated and tested on the first 1800 smallest (in terms of file size) files present in the Nepali National Corpus dataset due to computational limitations.

The following tags are not represented in the Nepali National Corpus dataset: 'DJF', 'CD', 'MLM' and 'NULL'.

4. DETAILED TEST PLAN

SI.NO	Test Case	Response
1.	User enters a Nepali sentence in the textbox and presses submit button.	The sentence is sent to the server where the model predicts the tags for each word and returns the result.
2.	User enters multiple sentences in the textbox and presses the submit button.	The input is sent to the server where it is split into sentences and the model predicts the tags for each word and returns the result.
3.	User clicks the download text file button without clicking submit button after entering text in the text box.	The generated text file will have the following error: "NA Enter devanagari sentence and click on submit "
4.	User clicks the download text file button after clicking submit button.	The generated text file will have the tagged sentences.
5.	User clicks submit button without entering a sentence.	Alert box displays that there is no input.
6.	User enters non-Devanagari text in the textbox and presses submit button	Alert box displays that the input contains non-Devanagari characters.
7.	Server not started	Web browser displays error.

Table 2: Test Cases and Responses

5. IMPLEMENTATION DETAILS

5.1 Extracted the data from the XML files and stored it as a CSV file.

The Nepali National Corpus dataset consists of a large number of files containing parts-of-speech tagged Nepali text. The format of the data however is .xml which is not easy to work with.

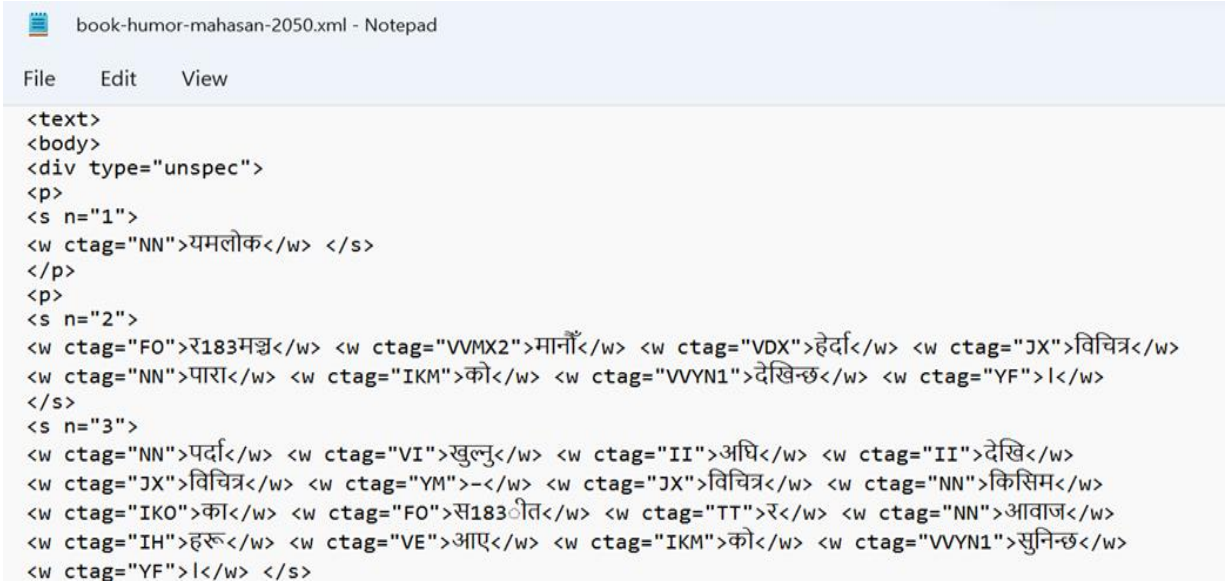


Fig 7: Sample .xml File from Nepali National Corpus

Hence, we have first extracted the data from the xml files and stored it in a more easy to work with format in .csv. The below table shows how the data looks once it has been extracted from .xml format and stored as a .csv.

ctag	**word**
NP	यादव
JX	काला
VVYX2	बनिरहेछन्
NN	राजी
DDM	यस्तो
RR	निकै
NN	दुनियाँ
RR	पहिल्यै

Table3: Sample of data after converting to CSV

The algorithm for the function process_xml_files(xml_folder) below shows how the data is extracted from the .xml files and stored in a .csv file:

Step 1. Start

Step 2. Open a new CSV file in write mode.

Step 3. Create a CSV writer object to write data to the CSV file.

Step 4. Write the column headers ‘ctag’ and ‘word’ to the CSV file.

Step 5. Loop through each file in the xml_folder and its subdirectories using glob.

5.1 Parse each XML file using ElementTree and get the root element of the XML tree.

5.2 Print a message indicating the current file being processed.

5.3 Loop through each 'w' element in the XML tree using iter() function.

5.3.1 Get the 'ctag' attribute value of the 'w' element.

5.3.2 Get the text content of the 'w' element and assign it to the 'word' variable. If the 'w' element is None, assign None to the 'word' variable.

5.3.3 Write the 'ctag' and 'word' values as a new row in the CSV file using the CSV writer object.

Step 6. Print a message indicating that the processing has completed after all XML files have been processed.

Step 7. Stop

5.2 Processing the data in the CSV file ,defining the model and training it:

The data in the CSV file is read into a pandas dataframe and the rows containing erroneous tags were removed. Sentences are then extracted from the dataframe by splitting at the end of sentence markers(| , !, and ?) and stored in a list. Each sentence was stored as a sub-list where each word was stored in a tuple along with its tag as shown in the below example:

[('यसै', 'DDX'), ('मा', 'II'), ('नै', 'TT'), ('पहिलो', 'MOM'), ('चरण', 'NN'), ('को', 'IKM'), ('वार्ता', 'NN'), ('समाप्त', 'JX'), ('भयो', 'VVYN1'), ('।', 'YF')]

The data in the 'word' column is then tokenized (converted to integers) and a word2id dictionary is created to assign each unique item in the column to a unique integer. Similarly, the 'ctags' column is tokenized and a tag2id dictionary is created which maps each unique item in the 'ctag' column to a unique integer value.

Next the data is split into X and y, where X consists lists where each list contains the tokenized form of a sentence and y consists of lists where each lists contains the tokenized form of the corresponding tags for the sentence.

Each list in X and y is then padded with Dummy values upto 50 words to ensure all inputs to the model are of fixed length. Sequences of having more than 50 items are truncated to 50 items.

The y variable is then one-hot-encoded.

The processed data is then split into training(120981 sentences) , testing(33306 sentences) and validation sets(13442 sentences).

The X, y, tags2id and words2id variables are then saved as files using the pickle python library.

A Keras sequential model was then defined with an input layer, an embedding layer followed by a dropout layer, bidirectional LSTM layer and a dense layer as the output layer.

The model was then trained on 120981 sentences for 3 epochs and achieved training accuracy of 98.61% and validation accuracy of 98.59%.

The algorithm below shows the data in the CSV is processed and then how the model is defined and trained:

Step 1. Start

Step 2. Import necessary libraries.

Step 3. Define the list of ctags and their respective definitions.

Step 4. Read the data from a CSV file and store it in a variable called 'data'.

Step 5. Filter the data based on the ctags defined earlier.

Step 6. Reset the index of the filtered data.

Step 7. Iterate over each row of the filtered data:

7.1 If the word is not a punctuation mark, add it to the 't' list.

7.2 If the word is a punctuation mark, add it to the 't' list and append the 't' list to the 'sents' list. Then, reset the 't' list to an empty list.

Step 8. Get the list of unique words and add a "DUMMY" word at the end of it.

Step 9. Get the list of unique ctags and add a "." at the end of it.

Step 10. Create a dictionary called 'word2id' that maps each word to a unique ID.

Step 11. Create a dictionary called 'tag2id' that maps each tag to a unique ID.

Step 12. Pad the sequences of words and tags to a maximum length of 50 and convert them to numerical representations from the word2id and tag2id dictionaries.

Step 13. Convert the tags to one-hot encoded vectors.

Step 14. Split the data into training, testing and validation sets.

Step 15. Save the dictionaries word2id and tag2id, X and y data, and the tags and words lists using the 'pickle' library.

Step 16. Define a Keras Sequential model.

Step 17. Add the input layer of the model to take input of shape (50,)

Step 18. Add a word embedding layer to convert each data point to a matrix of shape(50,50).

Step 19. Add a dropout layer to convert 10% of the datapoints to 0.

Step 20. Add a Bi-LSTM layer with 100 LSTM units in each direction.

Step 21. Add a time-distributed dense layer with softmax activation as the output layer.

Step 22. Compile the model with the ‘rmsprop’ optimizer.

Step 23. Fit the model on the training data for 3 epochs.

Step 24. Save the model as an h5 file.

Step 25. Stop

5.3 Testing the model

The trained model was loaded and was tested on the test data to get its performance in terms of accuracy, precision and f1 score. Accuracy is the ratio of correctly predicted observations to the total number of observations in the dataset. It measures how often the model is correct in its predictions. Precision is the ratio of true positives (correctly predicted positive observations) to the total number of positive predictions. It measures the accuracy of positive predictions. Recall is the ratio of true positives to the total number of actual positive observations in the dataset. It measures the ability of the model to identify positive observations. F1 score is the harmonic mean of precision and recall, which provides a single measure of the model's performance that takes both precision and recall into account. The actual and predicted labels for a few sentences in the test set were also checked for performance.

The algorithm below shows how the testing was done:

Step 1. Start

Step 2. Import the necessary libraries and modules.

Step 3. Load the word2id, tag2id, X, y, tags, and words using the pickle module.

Step 4. Split the data into training and testing sets using train_test_split from sklearn.model_selection .

Step 5. Load the saved Keras model using keras.models.load_model .

Step 6. Evaluate the loaded model on the test data using evaluate method from Keras

Step 7. Display the test accuracy and test loss.

Step 8. Predict the labels for the test data using the loaded model.

Step 9. Convert the predicted probabilities into class labels using argmax method.

Step 10. Convert the true labels into class labels using argmax method.

Step 11. Reshape y_test_classes and y_pred_classes to compatible format using ravel method.

Step 12. Calculate precision using precision_score from sklearn.metrics .

Step 13. Calculate recall using recall_score from sklearn.metrics .

Step 14. Calculate f1-score using f1_score from sklearn.metrics .

Step 15. Display the precision, recall, and f1-score.

Step 16. Print actual and predicted results for the first 100 sentences in the test set using a for loop.

Step 17. Stop

5.4 Frontend of the application.

The frontend of the POS tagger, built using React, provides a user interface for users to input text and view the output of the POS tagging model. It gives user-friendly experience for the user when interacting with the POS tagger.

The algorithm below shows how the frontend was designed:

Step 1. Start

Step 2. Import React, useState, useEffect from "react" and Copywritecom from "./Copywritecom" and "../css flies/Form.css".

Step 3. Create a functional component Form() using useState hook.

Step 4. Define initial states for text and outputText using useState hook.

Step 5. Define the URL name for API call as url_name = "http://127.0.0.1:5000/".

Step 6. Define manage_text as outputText.

Step 7. Define WordLabelPairs component using map function.

Step 8. Split the manage_text using newline character "\n" and store the result in pairs variable.

Step 9. Render each word and its corresponding label using pairs.map() function.

Step 10. Define a function handleSubmit for form submission.

Step 11. Prevent default form submission behaviour using e.preventDefault().

Step 12. Check if the input text is empty or not, if empty alert the user and return.

Step 13. Check if the input text is in the Devanagari script, if not alert the user and return.

Step 14. Split the sentence using space separator and store the result in words variable.

Step 15. Define an empty array myArray and a string w.

Step 16. Iterate through each word in wordss and concatenate it with w variable.

Step 17. If the word is either "?" or "!" or "।", push w into myArray and reset w to an empty string.

Step 18. Iterate through each element in myArray.

Step 19. Define a text_line variable with the value "postagger?text=" +

element.

Step 20. Make an asynchronous call to the API using fetch() function and store the response in response1 variable.

Step 21. Parse the response using json() function and store the output in val variable.

Step 22. Append val to final_out variable.

Step 23. Set the outputText to the final_out using setOutputText() function.

Step 24. Define a useEffect() hook that sets the focus to the input field when the component mounts.

Step 25. Render a form with input field, button and WordLabelPairs component.

Step 26. Export the Form component.

Step 27. Stop.

5.5 Backend and connection with the model.

The backend of the POS tagger application uses Flask to receive input from the frontend. The input text is tokenized into sentences and passed to the model for processing. Once the model generates the output, it is sent back to the frontend for display. Flask acts as the intermediary between the frontend and the model, allowing for seamless communication and processing of data.

The algorithm below shows how the backend was designed:

Step 1. Start.

Step 2. Import required libraries

Step 3. Load the Nepali model using the fasttext.load_model() function and pass the path of the pre-trained model as an argument

Step 4. Load the pickle files using the open() function and store them in respective variables. The pickle files are:

a. word2id_1800.pickle
b. tag2id_1800.pickle
c. X_1800.pickle
d. tags_1800.pickle
e. words_1800.pickle
f. word_vectors_1800.pickle
g. tree.pickle

Step 5. Define a function named 'postagger' and decorate it with the '@app.route' decorator. The function will be called when the client sends an HTTP GET request to the server at the URL endpoint '/postagger'

Step 6. Within the function, extract the 'text' parameter from the request using the request.args.get() function and store it in a variable called 'text'.

Step 7. Call the function named 'fun' with 'text' as the argument and store the returned result in a variable called 'pos_tagged'

Step 8. Create a dictionary called 'my_data' and assign 'pos_tagged' to it with the key 'data'

Step 9. Return 'my_data' as a JSON response using the jsonify() function

Step 10. Define a function named 'fun' which takes 'text' as the parameter

Step 11. Preprocess the input text using the following steps:

a. Tokenize the text using the fasttext.tokenize() function

b. Convert the tokenized text into a list of word indices using the word2id dictionary

c. Pad the list of word indices to a fixed length using the pad_sequences() function

Step 12. Make a prediction using the pre-trained model by calling the predict() function with the padded sequence as the argument. The returned result is a list of predicted tags.

Step 13. Convert the predicted tags from their integer representation to their string representation using the tag2id dictionary

Step 14. Create a list called 'pos_tagged' to store the tagged words

Step 15. Loop through the predicted tags and their corresponding words, and append a dictionary containing the word and its predicted tag to the 'pos_tagged' list

Step 16. Return the 'pos_tagged' list.

Step 17. Stop.

6. RESULTS AND DISCUSSIONS

The Nepali parts-of-speech tagger was built using a Bi-LSTM based Keras model. The summary of the model is given in the below figure.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 50)]              0

 embedding (Embedding)       (None, 50, 50)            7185800

 dropout (Dropout)           (None, 50, 50)            0

 bidirectional (Bidirectiona  (None, 50, 200)          120800
 l)

 time_distributed (TimeDistr  (None, 50, 109)          21909
 ibuted)

=================================================================
Total params: 7,328,509
Trainable params: 7,328,509
Non-trainable params: 0
_________________________________________________________________
```

Fig 8: Summary of the model.

The model was then trained on 120981 sentences and was found to converge after 3 epochs.

```
Epoch 1/3
1891/1891 [==============================] - 602s 316ms/step - loss: 0.4191 - categorical_accuracy: 0.9027 - val_loss: 0.1186 - val_categorical_accuracy: 0.9724
Epoch 2/3
1891/1891 [==============================] - 599s 317ms/step - loss: 0.0879 - categorical_accuracy: 0.9795 - val_loss: 0.0716 - val_categorical_accuracy: 0.9829
Epoch 3/3
1891/1891 [==============================] - 601s 318ms/step - loss: 0.0583 - categorical_accuracy: 0.9861 - val_loss: 0.0590 - val_categorical_accuracy: 0.9859
```

Fig 9: Model training and accuracy and loss scores on training and validation set.

The accuracy and loss graphs illustrating the convergence of the model are given below. Here, convergence means the value of validation loss and validation accuracy are almost equal to the training loss and training accuracy. When the model converges, it is said to be the best possible fit for the problem.

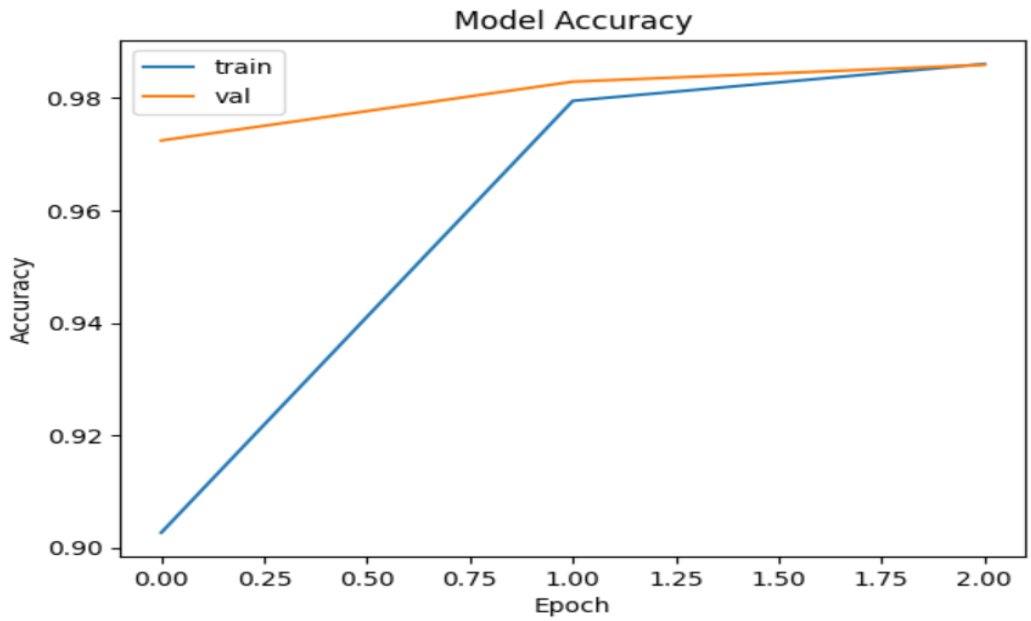


Fig 10: Accuracy Graph

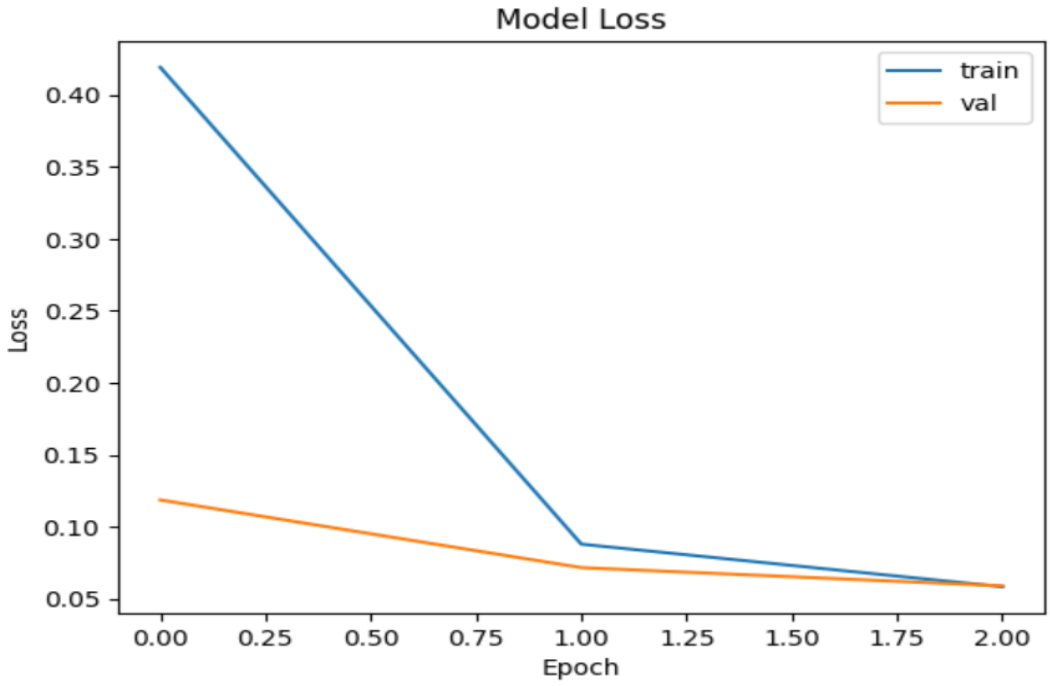


Fig 11: Loss Graph

The model was then tested on the test set (33306 sentences) and its performance was measured on various metrics viz. accuracy, loss, precision, recall and f1-score.

```
1051/1051 [==============================] - 38s 36ms/step - loss: 0.0588 - categorical_accuracy: 0.9861
Test accuracy: 0.9860798716545105
Test Loss: 0.058771152049303055
1051/1051 [==============================] - 33s 32ms/step
Precision:  0.9859426444990462
Recall:  0.986079866690472
f1-score:  0.9853399010210044
```

Fig 12: Accuracy, loss, precision, recall and f1-score of the trained model.

The actual and predicted labels for a few sentences in the test set were also checked for performance. The result for one such sentence is given below.

```
1/1 [==============================] - 0s 52ms/step
बेला          --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
न              --Predicted:CC(Coordinating Conjunction)----Actual:CC(Coordinating Conjunction)
कुबेला        --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
के             --Predicted:TT(Particle)----Actual:DKX(Unmarked Interrogative Determiner)
को             --Predicted:IKM(Masculine Genitive Postposition)----Actual:IKM(Masculine Genitive Postposition)
आँसु           --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
भनेर           --Predicted:VQ(Sequential Participle-converb)----Actual:VQ(Sequential Participle-converb)
छेउ            --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
बाट            --Predicted:II(Postposition)----Actual:II(Postposition)
उस             --Predicted:DDX(Unmarked Demonstrative Determiner)----Actual:DDX(Unmarked Demonstrative Determiner)
की             --Predicted:IKF(Feminine Genitive Postposition)----Actual:IKF(Feminine Genitive Postposition)
स्वास्नी      --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
,              --Predicted:YM(Sentence-medial Punctuation)----Actual:YM(Sentence-medial Punctuation)
छोरा           --Predicted:NN(Common Noun)----Actual:NN(Common Noun)
कराउँछन्      --Predicted:VVYX2(Third Person Plural(or Medial-honorific Singular) Verb)----Actual:VVYX2(Third Person
।              --Predicted:YF(Sentence-final Punctuation)----Actual:YF(Sentence-final Punctuation)
```

Fig 13: Actual and predicted labels for a sample sentence in the test set.

Once the model was trained and ready to be deployed, a user-interface was developed for it which is illustrated in the below figure.

Fig 14: Completed front end for the application.

The backend logic for integrating the model with the front end was then written and Nepali text submitted to the webpage was successfully being tagged as shown below.

Fig 15: Sample output on the front-end.

Clicking the download text file button generates a text file which is then downloaded on the users machine as shown below:

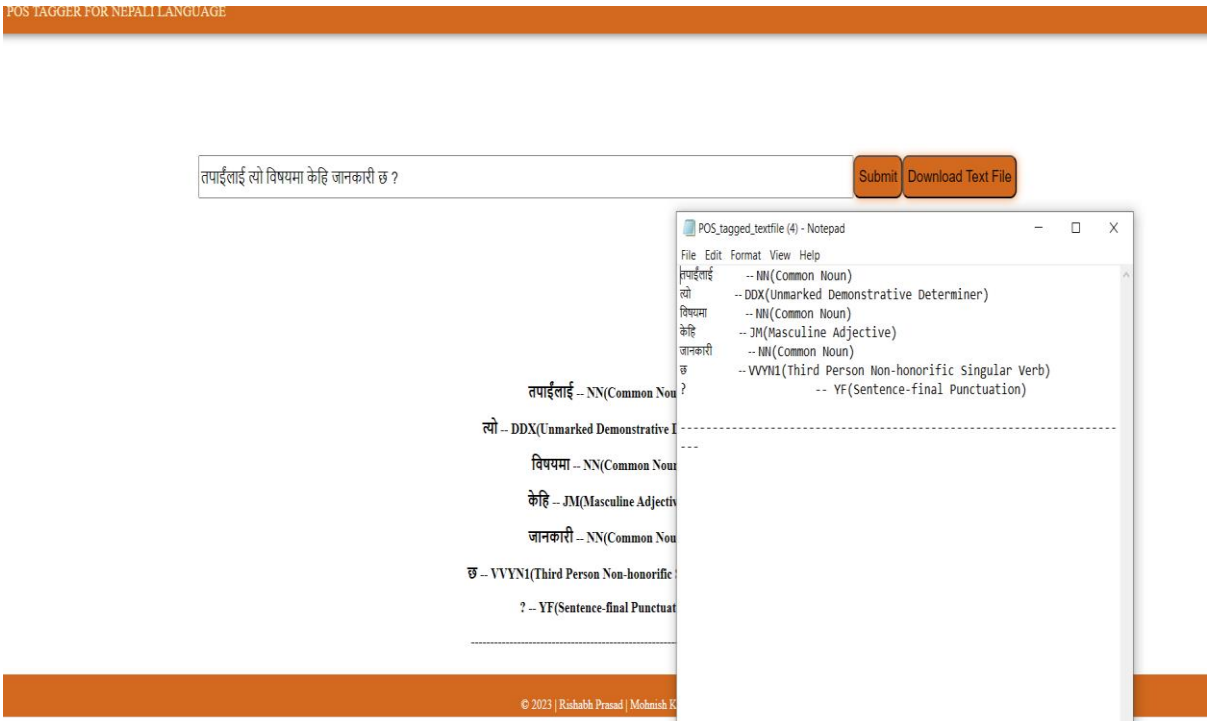


Fig 16: Sample output text file.

7. SUMMARY AND CONCLUSION

7.1 Summary of Achievements

The trained Bi-LSTM based model has achieved a good accuracy of 98.6% at tagging raw Nepali text. The use of the Bi-LSTM deep learning technique, which captures context in both directions (left to right and vice versa), has ensured that the model is able to be able to perform well in tagging the sequences. This is essential as the parts-of-speech tag for a word in a sentence is not only dependent on the word but also the words surrounding it. The performance of the model on compound words (words made up of two or more words or words containing a postposition) was also found to be excellent. Additionally, unlike other models, our model is not limited by vocabulary size which is common problem in most taggers.

The web application provides the user a user-friendly interface to enter raw Nepali text to tag and receive tagged sentences as output. The web application was found to be responsive and tags the raw Nepali text almost instantaneously. Additionally, the download feature allows the user to download the tagged text as a text file which can then be used for developing further NLP applications for the Nepali language.

7.2 Main difficulties encountered and how they were tackled

One of the main challenges encountered during the course of the project was out of vocabulary words i.e., words that were not present in the dataset. Since such words were not present in our word2id dictionary there was a key error when tokenizing (converting to integers) them before passing it to the model for tagging. This issue was remedied by using pre-trained fasttext embeddings[10] to convert each word in the word2id dictionary to its matrix representation(embedding) and then storing these in a tree for fast querying. When an unknown word is encountered during the tokenization process, the tree is then queried for the embedding for that word, the result of which returns the word that is morphologically or context-wise, similar to the unknown word. The id for the closest match is then placed in the sequence in place of the unknown word and passed to the model for tagging.

7.3 Limitations of the Project

The model is unable to make prediction on sentences containing non-Devanagari characters other than punctuation marks and numbers. Hence, for such sentences the non-Devanagari characters will have to be manually removed by the user before submitting it for tagging.

Moreover, the input text is required to be separated by whitespace after each token (word/punctuation marks) before submitting for tagging. If the input text is lacking a whitespace between tokens, then they will bet considered to be part of the same token.

The model does not perform well for ambiguous words i.e., words that can be tagged differently in different contexts.

The model can only handle sentences having 50 or less words in it.

7.4 Future Scope of Work

Usability can be further enhanced in the future by allowing the user to upload raw Nepali text files to tag instead of having to manually enter text in the text box.

In the future we can make the downloadable file to have a better structure like an XML document so that it can become easier for the user to extract sentences and words.

Moreover, there is a large scope for improvement when it comes to the tagging of ambiguous words.

The application can be hosted on a server and made available for further development of NLP applications for the Nepali language.

8. REFERENCES

[1] Culture and Society of Nepal, https://mofa.gov.np/about-nepal/culture-society/

Accessed on 9th May, 2023

[2] ReactJS, https://react.dev/ . Accessed on 18th March 2023

[3] Flask, https://flask.palletsprojects.com/ . Accessed on 18th March 2023

[4] Nepali National Corpus, https://www.sketchengine.eu/nepali-national-corpus . Retrieved on 6th February 2023

[5] Nelralec Tagset , https://nepalinlp.com/nelralec-tagset-a-part-of-speech-tagset-for-nepali-language/ . Retrieved on 8th February 2023

[6] Shahi, Tej Bahadur, Tank Nath Dhamala, and Bikash Balami. "Support Vector Machines based part of speech tagging for Nepali text." International Journal of Computer Applications 70, no. 24 (2013).

[7] Yajnik, Archit. "Part of speech tagging using statistical approach for Nepali text." International Journal of Cognitive and Language Sciences 11, no. 1 (2017): 76-79.

[8] Yajnik, Archit. "ANN based pos tagging for nepali text." International Journal on Natural Language Computing 7.

[9] Prabha, Greeshma, P. V. Jyothsna, K. K. Shahina, B. Premjith, and K. P. Soman. "A deep learning approach for part-of-speech tagging in nepali language." In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1132-1136. IEEE, 2018.

[10] Fasttext Embeddings, https://fasttext.cc/docs/en/crawl-vectors.html . Retrieved on 15th March 2023

Nepali POS Tagger

Yumnam Nirmal

ORIGINALITY REPORT

7%	3%	5%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Chandigarh College of Engineering & Technology , CCET Student Paper	1%
2	Greeshma Prabha, P.V. Jyothsna, K.K. Shahina, B. Premjith, K.P. Soman. "A Deep Learning Approach for Part-of-Speech Tagging in Nepali Language", 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018 Publication	1%
3	Submitted to Universidad Europea de Madrid Student Paper	1%
4	Submitted to The Robert Gordon University Student Paper	1%
5	dokumen.pub Internet Source	<1%
6	Submitted to Coventry University Student Paper	<1%
7	www.sketchengine.eu Internet Source	<1%

8	Submitted to Liverpool John Moores University Student Paper	<1 %
9	simran Garg, Devang Chaturvedi, Tanya Jain, Anju Mishra, Anjali Kapoor. "Sentiment Analysis of Twitter Data using Machine Learning: A Case Study of SVM Algorithm", Research Square Platform LLC, 2023 Publication	<1 %
10	medinform.jmir.org Internet Source	<1 %
11	"Full Proceedings Printed", 2021 International Research Conference on Smart Computing and Systems Engineering (SCSE), 2021 Publication	<1 %
12	savioglobal.com Internet Source	<1 %
13	Submitted to Banaras Hindu University Student Paper	<1 %
14	Leran Wang, Md Shafaeat Hossain, Joshua Pulfrey, Lisa Lancor. "The effectiveness of zoom touchscreen gestures for authentication and identification and its changes over time", Computers & Security, 2021 Publication	<1 %
15	C H Ayishathahira, C Sreejith, C Raseek. "Combination of Neural Networks and	<1 %

	Conditional Random Fields for Efficient Resume Parsing", 2018 International CET Conference on Control, Communication, and Computing (IC4), 2018 Publication	
16	geoscience-meeting.ch Internet Source	<1 %
17	Hakimeh Fadaei. "Persian POS tagging using probabilistic morphological analysis", International Journal of Computer Applications in Technology, 2010 Publication	<1 %
18	pubmed.ncbi.nlm.nih.gov Internet Source	<1 %
19	www.coursehero.com Internet Source	<1 %
20	Jhansi Rani Challapalli, Nagaraju Devarakonda. "A novel approach for optimization of convolution neural network with hybrid particle swarm and grey wolf algorithm for classification of Indian classical dances", Knowledge and Information Systems, 2022 Publication	<1 %
21	Lecture Notes in Computer Science, 2005. Publication	<1 %

22	CHUN-HUI HE, CHONG ZHANG, SHENG-ZE HU, ZHEN TAN, HUI-MING ZHU, BIN GE. "Chinese News Text Classification Algorithm Based on Online Knowledge Extension and Convolutional Neural Network", 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, 2019 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		