DWIT COLLEGE

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Institute of Science and Technology

PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE

A PROJECT REPORT

Submitted to

Department of Computer Science and Information Technology

DWIT College

In partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Information Technology

Submitted by

Raju Shrestha

August, 2019

DWIT College
DEERWALK INSTITUTE OF TECHNOLOGY
Tribhuvan University

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by RAJU SHRESTHA entitled **"PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

…………………………………………

Mr. Bhas Raj Pathak

Lecturer

Deerwalk Institute of Technology

DWIT College

DWIT College

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

LETTER OF APPROVAL

This is to certify that this project prepared by RAJU SHRESTHA entitled **“PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE”** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…………………………………… Mr. Bhas Raj Pathak [Supervisor] Lecturer DWIT College	………………………………………… Mr. Hitesh Karki Chief Academic Officer DWIT College
………………………………………….. Dr. Subarna Shakya [External Examiner] Professor, Department of Electronics and Computer Engineering Pulchowk Campus, IOE, Tribhuvan University	………………………………………….. Dr. Sunil Chaudhary [Internal Examiner] HOD, Department of Computer Science DWIT College

ACKNOWLEDGMENT

I would like to express my gratitude to Mr. Bhas Raj Pathak, Lecturer, DWIT College for supervising this project and giving assistance when faced with problems. Without his guidance and persistent help, this project would not have been possible. I would like to thank him for providing technical help and providing deeper knowledge about the technology needed for this project. Without his help, I would be struggling to understand the theory needed and might have reached an incorrect conclusion.

I would like to thank Mr. Hitesh Karki, Chief Academic Officer, DWIT College, Dr. Sunil Chaudhary, HOD of DWIT College, and Mr. Ritu Raj Lamsal, Lecturer DWIT College for guiding me on various aspect of this project.

I would also like to give special thanks to Dr. Subarna Shakya, Professor, Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, Tribhuvan University, for appearing as our external examiner and providing us necessary guidance and feedback.

At last but not the least my sincere thanks goes to my parents and member of my family, who have always supported me and to all of my friends who directly or indirectly helped me to complete this project report.

Raju Shrestha

TU Exam Roll No.: 7461/072

Date: August 12, 2019

ABSTRACT

Parts of Speech Tagger for Nepali Text Using Support Vector Machine is an application that assigns the appropriate parts of speech like noun, pronoun, verb, adverb, adjective etc. and other lexical tags to each words written in Nepali language based on its definition as well as context. The parts of speech tagger is build using the supervise machine leaning algorithm namely Support Vector Machine. The model uses 14 million Nepali words and corpus consists of written text from 15 different genre with 2000 words each published between 1990 and 1992. The texts are from a wide range of sources such as internet webs, newspapers or books. And, the model is trained with 80,000 lemmatized words collected from the Nepali National Monolingual Written Corpus.

The Parts of Speech Tagger for Nepali Text has wide range of scope in research and NLP applications such as machine translation, speech recognition, speech synthesis, grammar checker, information retrieval and extraction.

Nepali is morphologically rich language and one has to consider many features to build the language model. The SVM based POS tagger construct the feature vectors for each input word and classify the word into one of the two classes (One Vs Rest).

The performance analysis includes different components such as known words, unknown words and size of the training data. The average accuracy obtained for lemmatized text and unprocessed raw text is 88% and 72% respectively.

Keywords: *Parts of Speech Tagger; Natural Language Processing; Support Vector Machine; Supervised Machine Learning*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

ANN:	Artificial Neural Network
CSS:	Cascading Style Sheet
GRNN:	General Regression Neural Network
HMM:	Hidden Markov Model
HTML:	Hyper Text Markup Language
NLG:	Natural Language Generation
NLP:	Natural Language Processing
NLU:	Natural Language Understanding
POS:	Parts of Speech
RBF:	Radial Basis Function
SVM:	Support Vector Machine
WBS:	Work Breakdown Structure
XML:	Extensible Markup Language

CHAPTER 1: INTRODUCTION

1.1 Overview

Parts of Speech (POS) Tagging is one of the widely researched topic in NLP. Parts of Speech tagging can be defined as the task of task of assigning the appropriate POS tag or lexical category to each word in a natural language sentence. It is an initial step in Natural Language Processing (NLP) and is useful for most NLP applications,[2] and has a diverse application domain including speech recognition, speech synthesis, grammar checker, machine translation, information retrieval and extraction etc.

The work on automatic part of speech tagging started in early 1960s [11]. Klein and Simmon's rule based POS tagger can be considered as the first automatic tagging system [15]. Since rule based approaches needs more sophisticated rules to capture the language knowledge, later on the data driven approaches were developed as [5] and recently machine learning approaches are being developed [3, 8, 10].

Nepali is morphologically rich language and one has to consider many features to build a language model for such language. The Support Vector Machine based POS tagger has been implemented in [9] for a Bengali language which is also morphologically rich and shown the outstanding performance. In [9] rich feature set has been used to model the language characteristic. The Support Vector Machine are recently developed supervised learning method having good performance and generalization. SVM has been applied in text classification and shown that it can handle large features and resist of overfitting [5].

Although, wide research is being carried out in Parts of Speech tagging for many languages, but only few research has been published for Nepali language.

1.2 Background

Language is a basic form of communication. Human beings use language to express ideas, to form thoughts, share feelings, to show will and activity. Knowledge of language is a core base developed from the concept of communication.

Natural Language Processing (NLP) is a diversified field of computational linguistics, [17] which is widely used is research and development of applications like Language Translation, Stemmer and Morphological Analyzer, Text to Speech translation, Parsing, POS tagging etc. Among those POS tagging is one of the core part of NLP which is used in other applications of computational linguistics. [17]

Natural Language Processing (NLP) is the field of machine learning which is concerned with programming machines to understand the machine and generate the natural language. Thus, NLP is consists of two components: Natural Language Understanding (NLU) and Natural Language Generation (NLG).

Natural Language Understanding (NLU) is interpretation of text in meaningful form that the user communicates and classifies it into proper intents. It involves mapping the given input into useful representations and analyzing different aspects of the language.

Natural Language Generation (NLG) is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

NLP consists of various steps such as phonological analysis, morphological analysis, lexical analysis, syntactic analysis, semantic analysis and pragmatic analysis.

Lexical analysis is the early step in NLP and involves identifying and analyzing the structure of words. It mainly consists of recognition of lexicon of the language. At this phase, assigning part of speech tag to each individual word in a sentence helps to convey the grammatical meaning of words and simplifies the tasks of parsing, chunking, text classification and more. Thus, the accuracy of this module is significant to the other following modules [2].

Phonological analysis is related to sound system of language. The minimum unit of sound is the phoneme which is capable of distinguishing the meaning in the words.

Morphological analysis deals with the componential nature of word, which are composed of morphemes – the smallest unit of semantic meaning. Since, the meaning of each morpheme remains the same across the words but human can break down an unknown word into its constituent form in order to understand its meaning. Similarly, the NLP system can recognize the meaning conveyed by each morpheme in order to represent the meaning of word. [2]

Syntactic analysis uses the result of morphological analysis and lexical analysis to build a structural description of the sentence. The goal of this process is to convert the flat list of words that forms the sentence into a structure that defines the units [2].

Semantic analysis derives an absolute meaning from context; it determines the possible meaning of a sentence in a context.

Pragmatic analysis derives knowledge from external commonsense information; it means understanding the purpose use of language in situations, partially those aspect of language which require world knowledge.

Parts of Speech Tagger for Nepali Text is a web-based application uses Support Vector Machine as a classifier. It allows users to enter text as input and the system will automatically assign the corresponding appropriate tag.

1.3 Problem Statement

Parts of Speech tagging is the initial and most significant step in Natural Language Processing. The processes like chunking, parsing, stemming, translation etc. are dependent on parts of speech. It can be further used in other application domains such as machine translation, speech recognition and speech synthesis, grammar checker, named entity recognition, question answering and information retrieval. It is considered as one of the core part of NLP, manually assigning tags to a large corpus becomes work-intensive. The performance of these task depends upon the performance of tagger. Hence, it is required to build a computer program (tagger) that automatically assign POS tags to a words in natural language and it should be fast, accurate, portable and trainable. But, the most challenging problem in POS tagging is to determine the proper context and features.

There are many POS tagging applications built for different languages like English, Spanish, Hindi etc. but only few taggers are available for Nepali Language. The resources and applications are limited for Nepali language and there is few advancement in this field recently.

Parts of Speech Tagger for Nepali Text Using Support Vector Machine is a web based application which allows user to enter the text as input and get the corresponding appropriate lexical tags for each word in the text. The module can be further used to other NLP applications since many processes are dependent it.

1.4 Objective of the Project

The object of this project is:

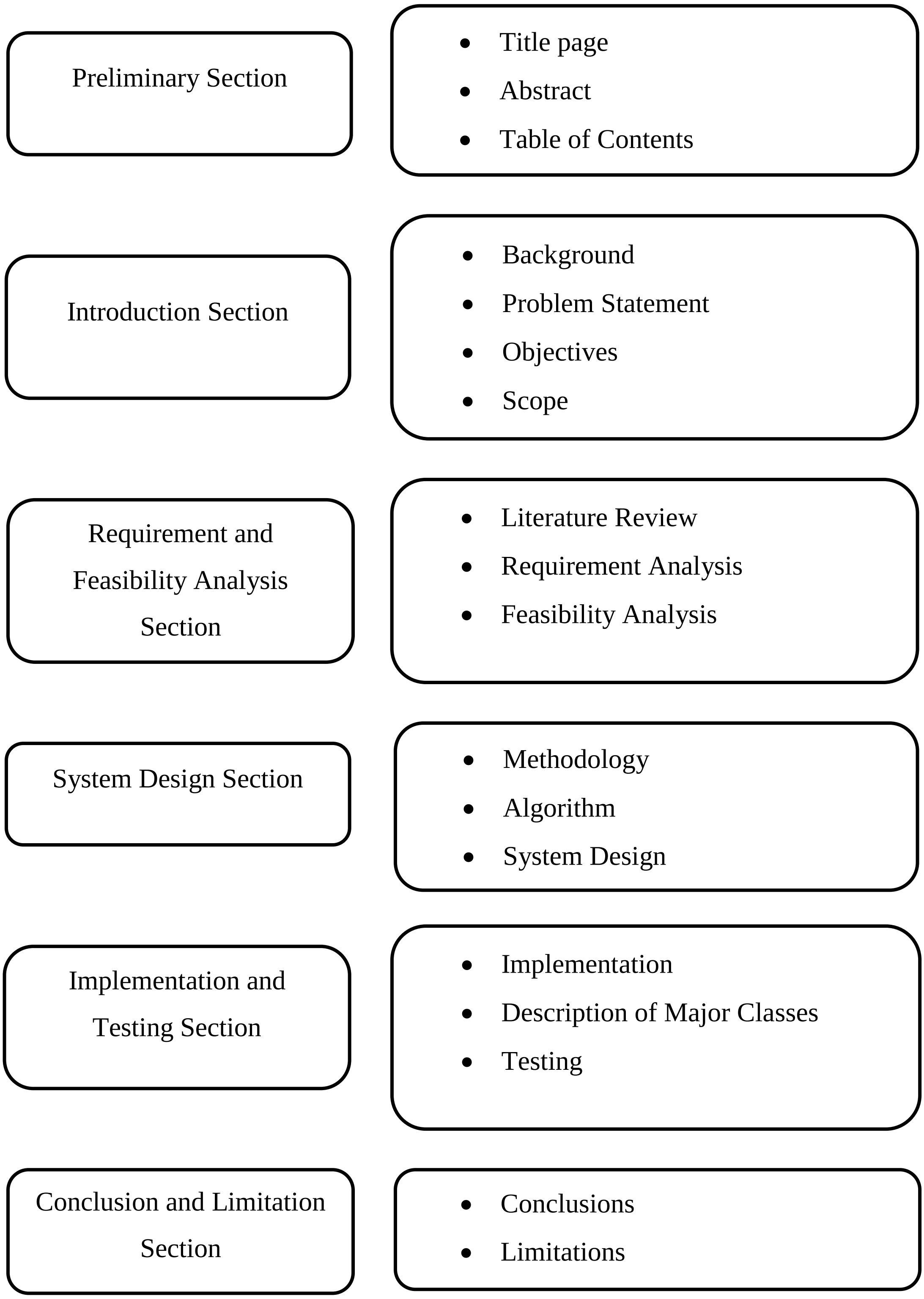
- To assign parts of speech tags automatically to each words for an input text or sentence based on its definition and context.

1.5 Scope of the Project

Parts of Speech Tagger for Nepali Text Using Support Vector Machine assigns the appropriate tags to each input text. It has wide range of scope in research and NLP applications such as machine translation, speech recognition and speech synthesis, grammar checker, named entity recognition, question answering, information retrieval and extraction. It can further used by linguistics, teachers and students in order to teach, learn as well as to determine the grammatical errors in the Nepali texts. In addition, this can be used for building other NLP applications as well.

1.6 Outline of the Report

The report is organized as follows:



Preliminary Section	• Title page • Abstract • Table of Contents
Introduction Section	• Background • Problem Statement • Objectives • Scope
Requirement and Feasibility Analysis Section	• Literature Review • Requirement Analysis • Feasibility Analysis
System Design Section	• Methodology • Algorithm • System Design
Implementation and Testing Section	• Implementation • Description of Major Classes • Testing
Conclusion and Limitation Section	• Conclusions • Limitations

Figure 1 - Outline of Report

CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS

2.1 Literature Review

2.1.1 Nepali Tag-set Review

The Nepali National Corpus (NNC) from NELRALEC (Nepali Language Resources and Localization for Education and communication) project, which contains 14 million Nepali words. It consists of speech corpus, spoken corpus, core sample (CS), general collection, and parallel data. It was first manually tagged some part (One hundred and sixty texts from the NNC–CS were annotated manually using this tag-set with 112 tags). This data then served as the basis for the training of an automatic tagger. [2]

The design of this Nepali POS Tag-set was inspired by the PENN Treebank POS Tag-set. Hence, whenever possible, the same naming convention has been used as in the case of the PENN Treebank Tag-set. NELRALAC tag-set is the first work in developing Nepali tag-set which consists of 112 tags. This tag-set has been compiled with reference to widely published grammars of Nepali. This tag-set was used to tag (Nepali National Corpus) NNC manually and semi manually. The short description of tags in [2] is given in the following Table 1. The detailed description of the 112 tag-set is at the Appendix Section.

Table 1- NNC Tag-sets

Category definition	Examples (Latin)	Examples (Devnagari)	Tag
Common noun	keTo, keTaa, kalam	केटो, केटा, कलम	NN
Proper noun	raam	राम	NP
Masculine adjective	moTo, raamro	मोटो ,राम्रो	JM

Category definition	**Examples** **(Latin)**	**Examples** **(Devnagari)**	**Tag**
Feminine adjective	moTii, raamrii	मोटी ,राम्री	JF
Other-agreement adjective	moTaa, raamraa	मोटा ,राम्रा	JO
Unmarked adjective	saphaa, dhanii, asal	सफा ,धनी ,असल	JX
Sanskrit-derived comparative or superlative adjective	uccatar, uccatam	उच्चतर ,उच्चतम	JT
First person pronoun	*ma, haamii, mai#*	म ,हामी ,मै#	PMX
First person possessive pronoun with masculine agreement	*mero, haamro*	मेरो ,हाम्रो	PMXKM
First person possessive pronoun with feminine agreement	*merii, haamrii*	मेरी ,हाम्री	PMXKF
First person possessive pronoun with other agreement	*meraa, haamraa*	मेरा ,हाम्री	PMXKO
Non-honorific second person pronoun	*ta~, tai#*	तँ ,तँ#	PTN
Non-honorific second person possessive pronoun with masculine agreement	*tero*	तेरो	PTNKM

2.1.2 Different Approaches for Building POS Tagger

A considerable amount of work has already been done in the field of POS tagging for English. Different approaches like the rule based approach, the stochastic approach and the transformation based learning approach along with modifications have been tried and implemented. However, if we look at the same scenario for South-Asian languages such as Bangla, Hindi, and Nepali, we find out that not much work has been done.

The work on automatic part of speech tagging started in early 1960s [11]. Klein and Simmon's rule based POS tagger can be considered as the first automatic tagging system [15]. Since rule based approaches needs more sophisticated rules to capture the language knowledge, later on the data driven approaches were developed as [5] and recently machine learning approaches are being developed [3, 8, 10].

The following Table 2 shows the comparison of different tagger reviewed in the paper [2] based upon the four criteria – Accuracy, Efficiency, Portable, and Trainable.

Table 2 – Comparison of Different Taggers for English

S.N	**Major Algorithms**	**Accuracy**	**Efficiency (word per second)**	**Portable**	**Trainable**
1	Rule Base [15]	83%	20	No	No
2	Rule Based (Using Transformation Rules) [5]	95% -97%	Unknown	Yes	No
3	Hidden Markov Model (TnT tagger) [2]	96.7%	Unknown	Yes	Yes
4	Support Vector Machine (SVMtool) [8]	96.7	1230	Yes	No

S.N	Major Algorithms	Accuracy	Efficiency (word per second)	Portable	Trainable
5	Neural Network Based [18]	97.7%	Unknown	Yes	No
6	Decision Tree Based [17]	97%	300	Yes	No

2.1.3 Comparison of Different Taggers for Nepali Text

The Neural Network based POS tagger namely Radial Basic Function Network (RBF), General Regression Neural Network (GRNN) and Feed Forward Neural Network is trained with 42,100 words and it is tested with 6000 words. The result shown in Table 3 clearly shows that the performance of all the three networks are very well for train set but in the case of test set except GRNN none other network performs efficiently. The fact that 5899 out of 6000 testing samples are tagged properly by GRNN. [17, 18]

Table 3 - Comparison of Neural Network Based Taggers for Nepali Text

S.N	Technique	Accuracy(%)in train set	Accuracy(%) in test set	Number of correctly tagged words in test set
1	RBF	100	25	1500
2	GRNN	100	98.32	5899
3	Feed forward Neural Network	99.76	26.65	1599

The General Regression Neural Network (GRNN) is probabilistic neural network technique that follows supervised learning, the network is assigned 123 features as an input layer and 41 neurons as an output layer for each word [18]. The network is trained using 5373 patterns (Nepali words) and corresponding tags. [18]

In the first phase, the same training set is validated, 4451 words out of 5373 are observed to be correct. Hence all together, the network has achieved 96.13% accuracy. In the second phase the network is tested on the words does not belong to the training set which contains 2500 Nepali words. The network has achieved 63.88% and 10.4% for correct identification and Group identification accuracy respectively, hence it becomes 74.28% total accuracy. [18]

The same sets are tested using Viterbi (traditional statistical technique). It is used for POS tagging for various languages.

The performance shown in Table 4 shows that GRNN outperforms better Viterbi [18].

Table 4 - Comparison of GRNN and Viterbi Based Taggers for Nepali Text

S.N	**Technique**	**Validation set**	**Accuracy (%)**	**Group Identification Accuracy (%)**	**Total Accuracy (%)**
1	GRNN	Training set (5373)	82.84	13.29	96.13
2	GRNN	Testing set (2500)	63.88	10.4	74.28
3	Viterbi	Training set (5373)	93	4.2	97.2
4	Viterbi	Testing set (2500)	37	3	40

The performance comparison between Hidden Markov Model (HMM), a probabilistic (non-deterministic) technique and Viterbi is shown in Table 5. The result shows that Viterbi technique performs better than HMM [16].

Table 5 - Comparison of HMM and Viterbi Based Taggers for Nepali Text

S.N	**Technique**	**No. of Mismatches**	**Accuracy**
1	HMM	3455	76.97
2	Viterbi	574	95.43

The proposed SVM based tagger has been compared with the existing TnT tagger for the Nepali language. [2] Accuracy of the tagger is the most important parameter to judge the quality of the tagger so the comparison of the taggers was done on the basis of accuracy only. The results shown in Figure 7 clearly shows that proposed SVM Tagger performed better than the already TnT tagger for Nepali.

Table 6 - Comparison of TnT and SVM Taggers for Nepali Text

Tagger	**Accuracy**		
	Known Words	Unknown Words	Overall Accuracy (%)
TnT	92%	56%	74%
SVM	96.48%	90.06%	93.27%

Nepali is a morphologically rich language which has many features. The POS tagging approaches like Rule based and Hidden Markov model cannot handle many features. The SVM based POS tagger has been implemented [1] for the Bengali language which is also morphologically rich and shows the outstanding performance.

SVM based tagger [1] was proposed which is efficient, portable, scalable and trainable. SVM has been successfully applied in text classification and shows that SVM can handle large features and is resist of overfitting [2].

2.2 Requirement Analysis

Functional and Non-Functional Requirements are as follows:

The basic functionality of Parts of Speech Tagger for Nepali Text which includes displaying the POS tag for each word of the input sentence or paragraph is shown in Table 7.

Table 7 - Functional and Non-Functional Requirements

Functional Requirement	Non-Functional Requirement
The system shall assign POS tags to each words for an input text or sentence	The system must display word with corresponding tag for the input text after classification.

2.3 Feasibility Analysis

After gathering of the required resources, whether the completion of the project with the gathered resource is feasible is checked using the following feasibility analysis.

2.3.1 Technical Feasibility

Parts of Speech Tagger for Nepali Text is a web application that uses Flask, which is a python-based framework. It uses HTML, Bootstrap CSS, JavaScript to design frontend and Python as the backend. It requires a server, client and Internet connection to function properly.

It supports both Windows and Linux platform for its operation. All of the technology required by Parts of Speech Tagger for Nepali Text can be accessed and available freely. Hence, it was determined technically feasible.

2.3.2 Economic Feasibility

The expenses incurred in the project are mostly indirect expenses. I used my personal devices to build the system and used personal internet subscription to do the research online. However, some direct expenses were incurred in the acquisition of printing the documentation.

Since this is an academic project, the incentive to complete the project was University credit score. Hence, considering the costs and benefits of completing the project, the project is deemed economically feasible.

2.3.3 Operational Feasibility

Parts of Speech Tagger for Nepali Text has a simple design and is easy to use. It uses two-tier architecture (i.e. Client and Server). It can be easily accessed by any user connected to internet, and can be used to assign parts of speech tags to the input sentences or to further use the application for various NLP applications. It can also be used in educational institutions in order to provide better grammar knowledge to students. Hence, Parts of Speech Tagger for Nepali Text was determined operationally feasible.

2.3.4 Schedule Feasibility

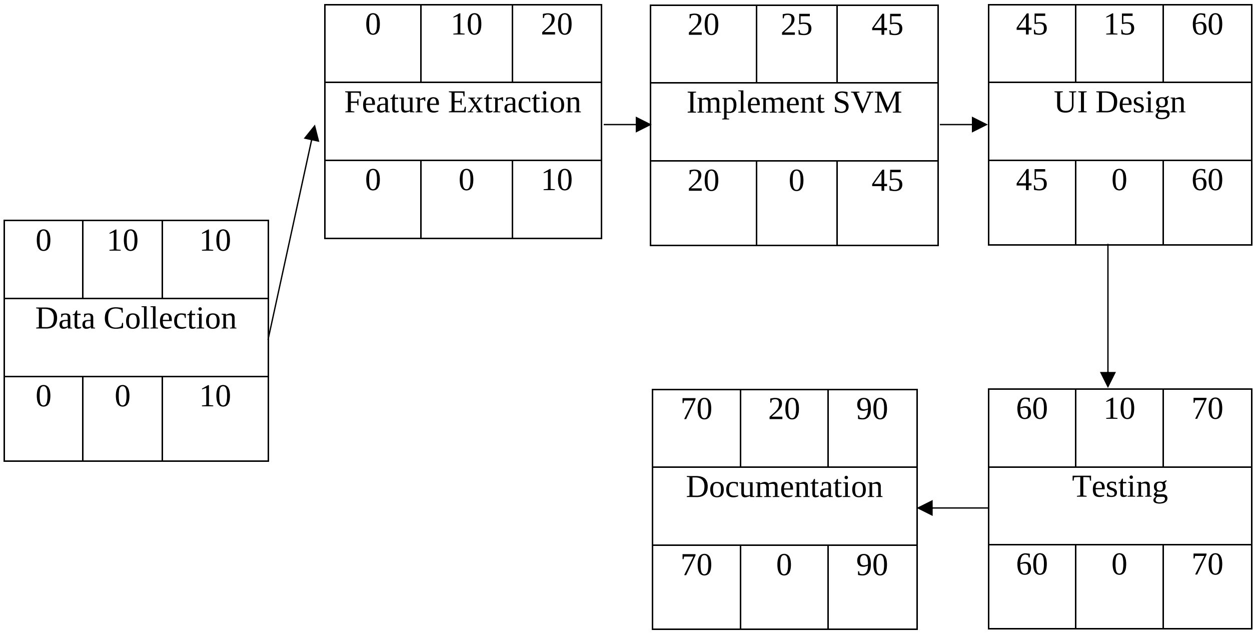
The schedule feasibility analysis is carried out using the CPM method. With CPM, critical tasks and interrelationship between tasks were identified which helped in planning that defines critical and non-critical tasks with the goal of preventing time-frame problems and process bottlenecks. The CPM analysis was carried out as follows:

First, the activity specification table with WBS is constructed as shown in the Table 8.

Table 8 - Activity Specification with WBS

Activity	**Time (days)**	**Predecessor**
Data Collection (A)	10	-
Research on previous works and algorithms to be implemented (B)	10	-
Feature Extraction (C)	10	A
Support Vector Machine Algorithm Implementation (D)	25	C
UI Design (E)	15	D
Testing (F)	10	D, E
Documentation (G)	20	F

Then, the identification of critical path and estimates for each activity are analyzed with the help of network diagram, which is based on the Table 8. The network diagram is shown in Figure 2.



Index

ES = Early Start, TF = Total Float, EF = Early Finish, LS = Late Finish,

FF = Free Float, LF = Late Finish, A = Activity, D = Duration

Figure 2 - Network Diagram to Identify Critical Path

Figure 2 shows Network Diagram of Parts of Speech Tagger for Nepali Text. As shown in the above figure, it is observed that the critical tasks are (A) Data Collection, (B) Research on previous works and algorithms to be implemented, (C) Feature Extraction, (D) Support Vector Machine algorithm implementation, (E) UI design, (F) Testing and (G) Documentation because it is clearly seen that the Early Finish (EF) and Late Finish (LF) of these activities are same. The total duration of the critical path is 90 days, which is in the project deadline range. Hence, this project is feasible in terms of the schedule since, the project is completed in time if the critical tasks are carried out within the specified tasks duration in Table 8.

CHAPTER 3: METHODOLOGY

Parts of Speech Tagging System has already been implemented for Hindi, English, Bengali, Tamil and Odia languages, using SVM algorithm. The obtained result for these language is satisfactory as the accuracy of SVM is high. Hence, Parts-of-Speech Tagger for Nepali Text also uses SVM algorithm for parts of speech tagging system for Nepali words.

3.1 Developmental Methodology

The waterfall development model was followed for this project because it is simple, easy to understand and to use it. Since, it is an individual project, it becomes easy to manage due to the rigidity of the model and each phase has specific deliverables and a review process. It was used because the model phases are processed and completed one at a time and these phases do not overlap. Also the requirements are well understood at the beginning of this project, so, the waterfall model is helpful in this.

3.2 Data Preparation

3.2.1 Data Collection

The Nepali Monolingual Written Corpus 14 million Nepali words. It consists of two corpus: core and general. The core corpus consists written texts from 15 different genres with 2000 words each published between 1990 and 1992 and the general corpus consists of written text collected from a wide range of sources such as the books, journals, magazines, newspapers and internet webs.

The written corpus is morphologically-annotated i.e. the text is divided into tokens. In this corpus, the tokens are appropriately-sized units for morphosyntactic analysis rather than orthographic tokens.

The written corpus is in XML format where a paragraph from the text is enclosed with <p> tag, the sentences in the paragraph are enclosed with <s> tag, and the words in the sentences are enclosed with <w> tag with its POS tag specified as the value of attribute ‘ctag’. There are a total of 112 POS tags in the corpus denoted by roman alphabetic symbols and an index is maintained for the symbols.

```
<s n="2">

     <w ctag="DDX">यस</w>

     <w ctag="NN"> अवधि</w>

     <w ctag="II"> मा</w>

     <w ctag="DKX">कसै</w>

     <w ctag="IE">ले</w>

     <w ctag="TT">पनि</w>

     <w ctag="DKX">कसै</w>

     <w ctag="JX">महत्वपूर्ण</w>

     <w ctag="NN">अवार्ड</w>

     <w ctag="VI">जित्न</w>

     <w ctag="VE">सके</w>

     <w ctag="IKM">को</w>

     <w ctag="VVYN1">थिएन</w>

     <w ctag="YF">।</w>

</s>
```

3.2.2 Construction of Train and Test Set

Firstly, the XML based corpus is parsed to obtain a train and test set of data. The words and their respective tags enclosed inside the <p>, <s>, <w>, <c> XML tags are extracted from the Nepali National Monolingual Written Corpus and a train dataset and test dataset are constructed. Such that,

Given, X = {w1, w2, ……,wn} , y = {t1, t2, …,tn}

Predict y_test = {?, ?,……. ?} for X_test = {w1, w2, …, wn}

Further, another test set is created from the raw unprocessed text, which is not lemmatized to test the accuracy of training model.

3.2.3 Feature Extraction

Feature extraction is the process of transforming arbitrary data, such as text or images, into numerical features usable for machine learning. The features are intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

In this project, Word feature of each word in a sentence is extracted to derive relevant information about that word and its POS tag. The extracted features of all the words in a sentence or paragraph are stored as lists of standard Python dictionary objects. The feature arrays are then converted to NumPy/SciPy representation using DictVectorizer to be further used by scikit-learn estimators.

Table 9 - Feature Set for Each Word

Feature-Set	Features
Word Features	Previous-previous word, Previous word, Word, Next word, Next-next word

The performance measurement for POS tagger is difficult in the sense that there is no universal agreed system for rating the performance of a tagger. The taggers are designed with different aspect in mind such as efficiency, accuracy, and portable.

Probably the most widely used system of assessing the performance of a tagger is to look at the percentage of tokens which are tagged correctly. This is the measurement of accuracy of tagger. The accuracy is a measure of how much of the information that the system returned is actually correct, and is also known as accuracy. Accuracy is defined as follows:

$$\text{Accuracy} = \frac{No\ of\ times\ (predicted\ tags == true\ value\ of\ tags)}{Total\ no\ of\ tokens} * 100 \qquad \text{(Equation 3.1)}$$

3.3 Algorithms Studied and Implemented

3.3.1 Support Vector Machine

Support Vector Machine (SVM) is the supervised machine learning approach that can be used for both classification and regression. The main objective of the Support Vector Machine is to find the best splitting boundary between data. In their basic form, SVM constructs the hyperplane in input space that correctly separates the example data into two classes. This hyperplane can be used to make the prediction of class for unseen data. The hyperplane exists for the linearly separable data. [4]

This can be illustrated with figure-

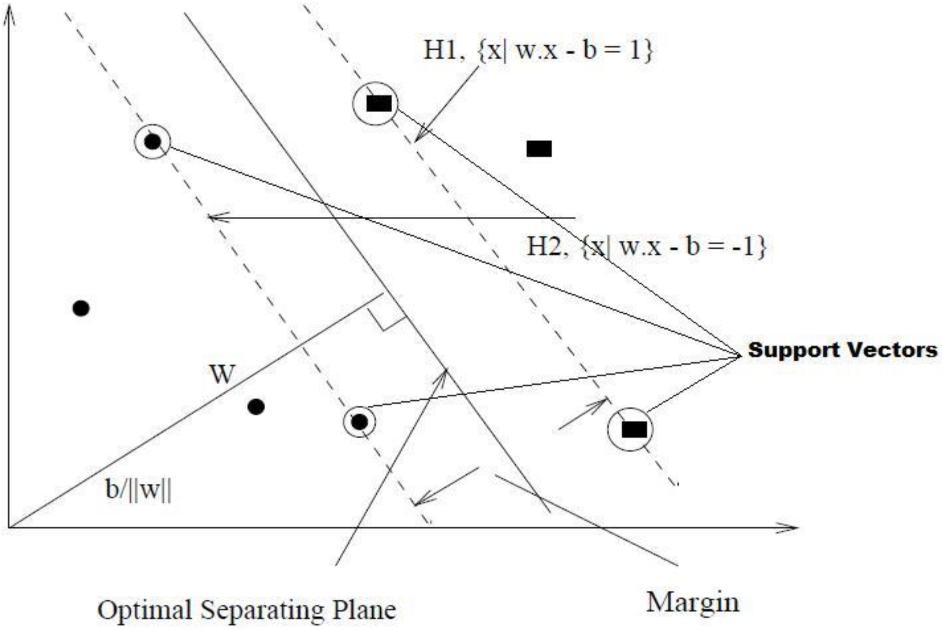


Figure 3 - Support Vector Machine

The equation for general hyperplane can be written as:

$$w.x - b = 0 \qquad \text{(Equation 3.2)}$$

Where x is the point vector, w is a weight vector and b is bias. The hyperplane should separate training data {(xi,yi), i = 1.….n and yi ∈(+1,-1)} in such way that $yi\ (w.\ xi - b) \geq 1$. The two plane H1 and H2 are supporting hyperplane. We can see that there exist so many hyperplanes that can separate the training data correctly but the SVM find one hyperplane that maximize the margin between two supporting hyperplanes often referred to as a decision boundary. It finds the w and b such that the distance (margin) between H1 and H2 is maximum. This can be formulated as optimization problem as:

$$\text{Minimize } f = \frac{|w|^2}{2} \qquad \text{(Equation 3.3)}$$

Subject to constraints $yi\ (w.\ xi - b) \geq 1$

Given that the optimal margin is determined by checking each and every data point against the condition stated above and we have the least norm of w, then the vectors of data points that lie on either of the hyperplanes become the **Support Vectors**. They are closest vector samples on the either sides of the hyperplanes.

$$y_i\ (w.\ x_i - b) = 1 \qquad \text{(Equation 3.4)}$$

$$yi\ (w.\ xi - b) = -1 \qquad \text{(Equation 3.5)}$$

Equations 3.4 and 3.5 represent positive class support vectors and negative class support vectors simultaneously.

One Vs Rest classification strategy is used for the binarization of Multiclass Classification. The idea here is to separate each group from the rest. To train N different binary classifiers, each classifier is trained to distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which outputs the largest (most positive) value is chosen.

This can be explained with an example- यस/DDX महिना/NN भित्र/II

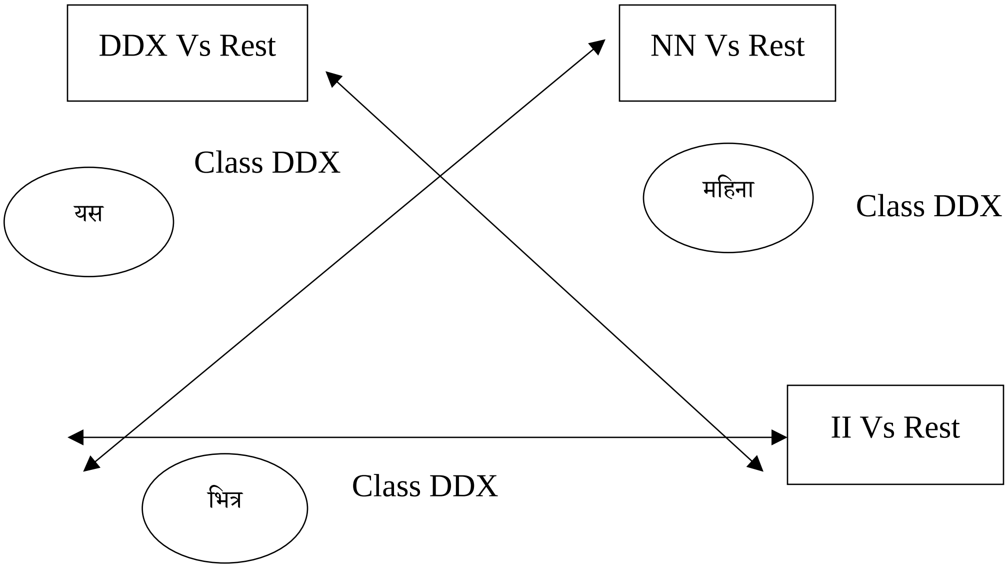


Figure 4 – One Vs Rest Classification

3.4 System Design

3.4.1 Flow Diagram

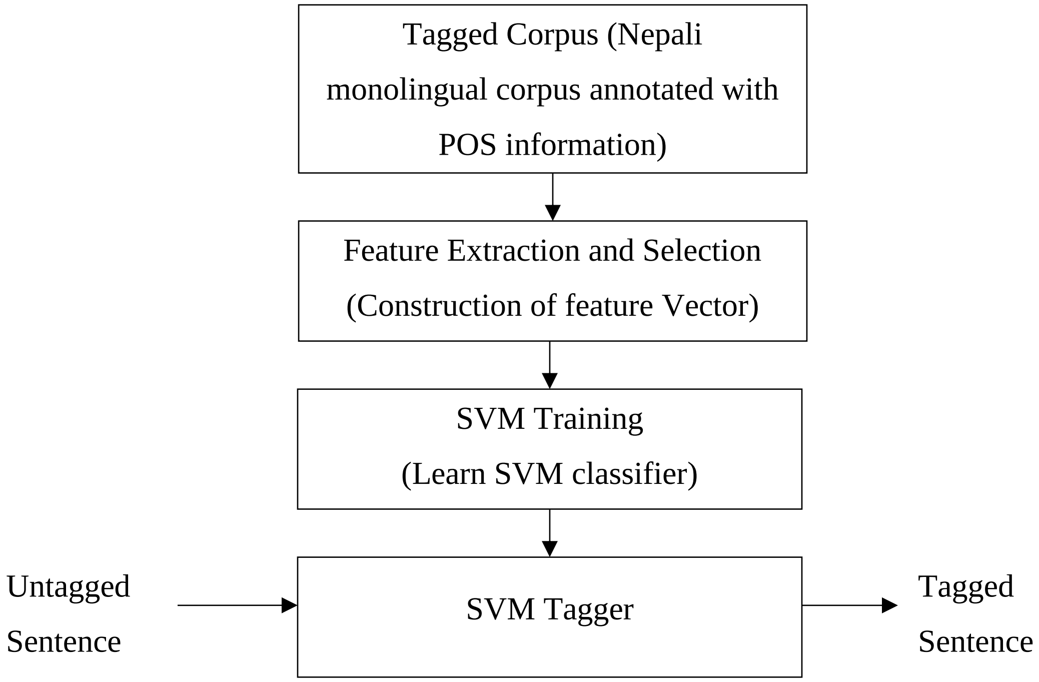

Tagged Corpus (Nepali monolingual corpus annotated with POS information)
Feature Extraction and Selection (Construction of feature Vector)
SVM Training (Learn SVM classifier)
Untagged Sentence
SVM Tagger
Tagged Sentence

Figure 5 - Flow Diagram of Parts of Speech Tagger for Nepali Text

The tagged Nepali corpus will be used as a training corpus from which the feature vectors are created for each word in the corpus. The SVM light [9] will be used to learn the model from these training vectors. Finally the Tagging algorithm will be implemented to perform the tagging of raw sentence in the input to produce the tagged output sentence.

3.4.2 Class Diagram

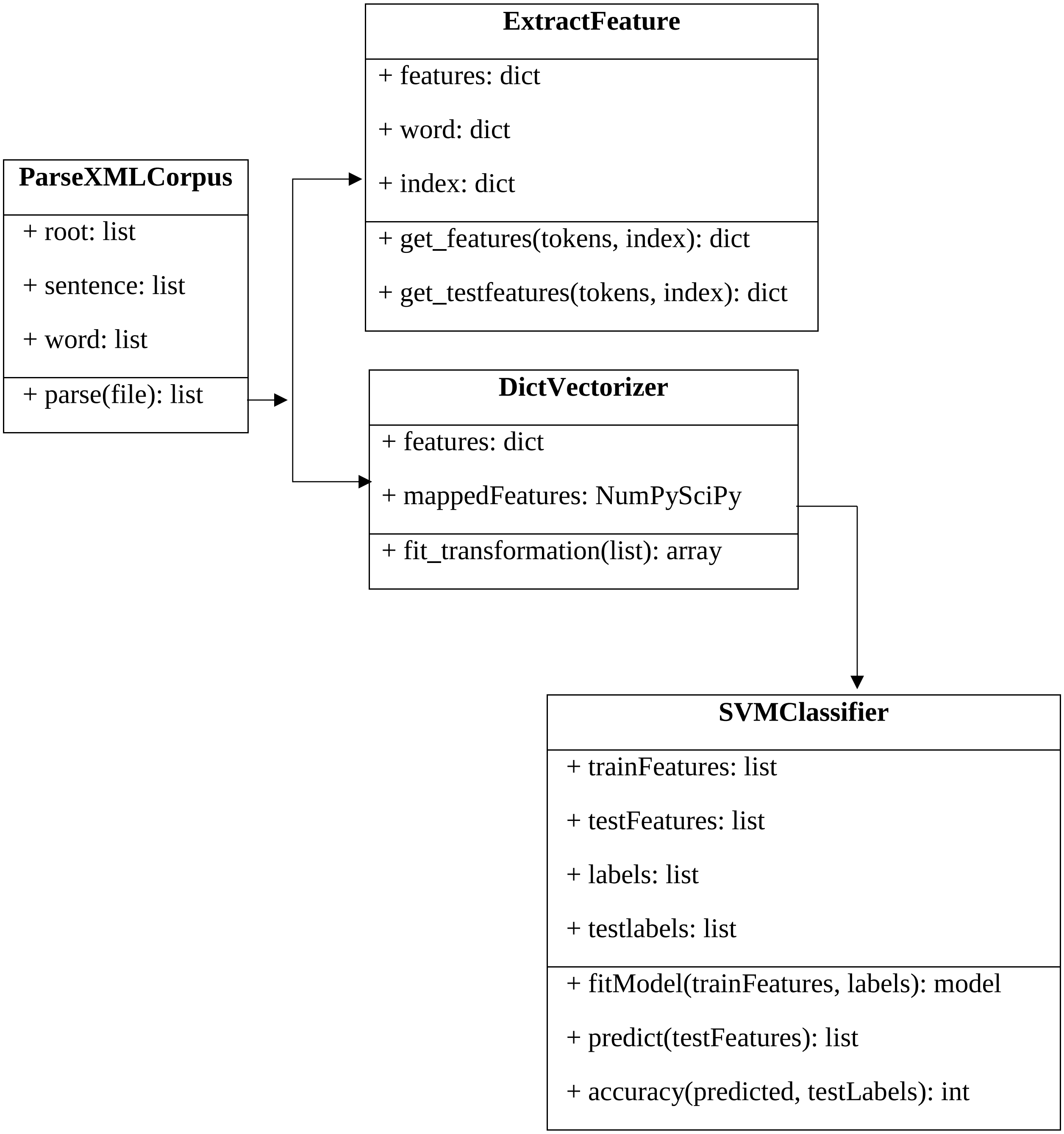


Figure 6 - Class Diagram of Parts of Speech Tagger for Nepali Text

Figure 6 explains the classes used in Parts-of-Speech Tagger for Nepali Text. There are four classes used in total, ParseXMLCorpus, ExtractFeature, DictVectorizer, and SVMClassifier. ParseXMLCorpus is the first class in the application. In this class, all the XML files of the corpus are parsed and each sentence are sent to the ExtractFeature class to calculate the features of words. DictVectorizer converts the feature arrays to NumPy/SciPy representation which are then sent to the SVMClassifier to predict the POS tag of each word in a new sentence.

3.4.3 State Diagram

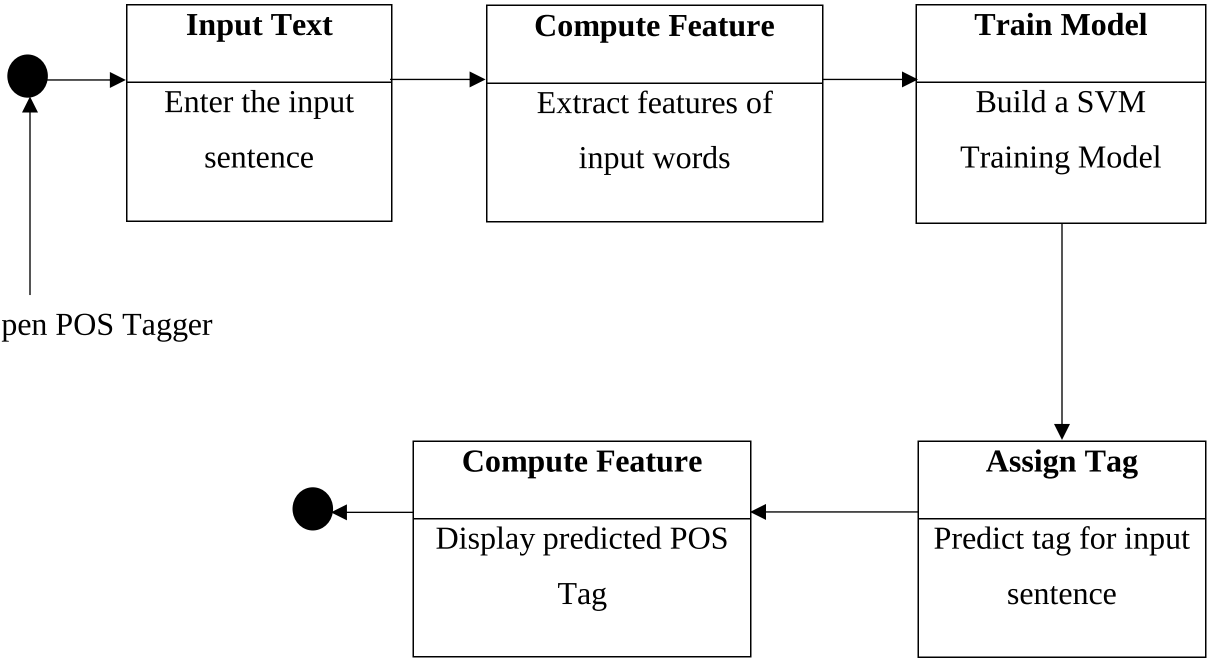

Input Text
Enter the input sentence
Compute Feature
Extract features of input words
Train Model
Build a SVM Training Model
Open POS Tagger
Compute Feature
Display predicted POS Tag
Assign Tag
Predict tag for input sentence

Figure 7 - State Diagram of Parts of Speech Tagger for Nepali Text

Figure 7 explains the different state of the system. First, the user opens Parts-of-Speech Tagger for Nepali Text, then enters either a paragraph or sentence in Nepali as in input to the system. The system then computes the features of each of the words in the sentence and fits into the trained SVM model. The model then predicts the tags for each of the words in the sentence. Finally, the predicted tags are displayed to the user as final output.

3.4.4 Sequence diagram

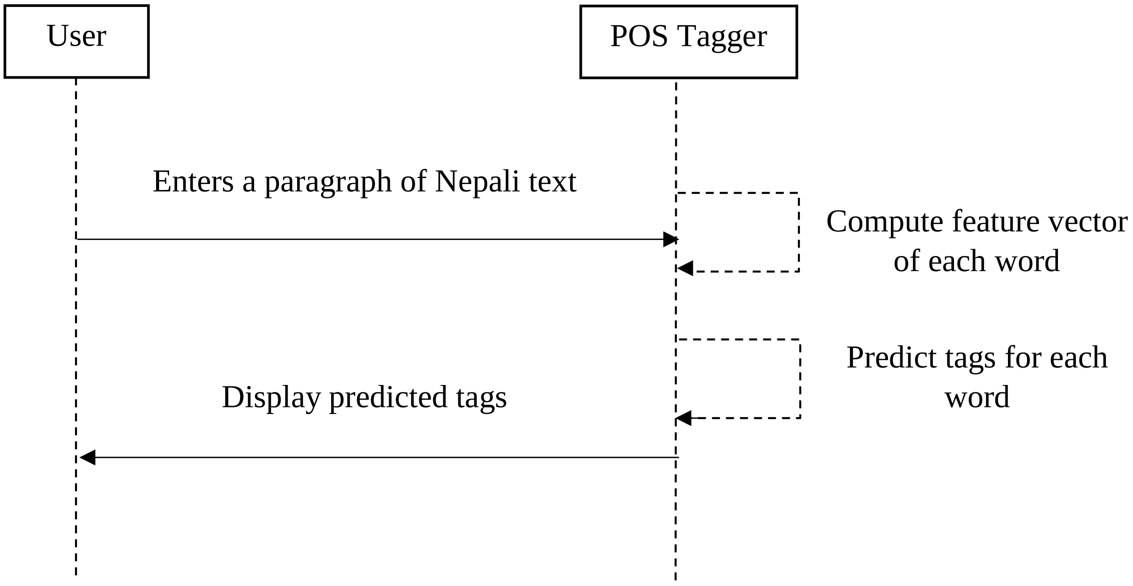

User
POS Tagger
Enters a paragraph of Nepali text
Compute feature vector of each word
Predict tags for each word
Display predicted tags

Figure 8 - Sequence Diagram of Parts of Speech Tagger for Nepali Text

Figure 8 explains the sequence of the Parts-of-Speech Tagger for Nepali Text. Initially, a user opens a browser and enters a sentence in the textbox available. Then the Parts-of-Speech Tagger extracts features of all the words in the sentence and sends it to the trained SVM model for prediction. After prediction, Parts-of-Speech Tagger then sends the assigned tags to the user as an output.

CHAPTER 4: IMPLEMENTATION AND EVALUATION

4.1 Tools and Technologies Used

This section describes the tools and technologies used in the project.

CASE tools:

- Draw.io

Client Side:

- HTML is used to structure the user interface and display the output results.
- Bootstrap CSS framework is used to beautifying the user interface.
- JavaScript is used to make HTTP requests to the server.

Server Side:

- Python programming language is used as the server-side programming language.
- Flask micro framework is used to implement a simple web user interface. It is used to start the web application and display the input field and the result.
- OS and NumPy are used for file handling and array processing functionalities respectively.

4.2 Implementation

The application can be access by user through a browser and see the interface. User can enter a paragraph of Nepali text as an input on the “Enter your text” textbox available. The application calculates the POS tags for the input text after the user selects the “Submit” button, [1]. Then the application displays input text along with their predicted tags on the interface. All of this is shown in Appendix.

When the user clicks Submit button, the input text is sent to the ExtractFeature function where feature vectors of each of the input words are created. The obtained feature vectors are then passed to the SVM trained model which predicts the corresponding tags for each word in the sentence. After classification, the predicted tags are displayed in the interface of the application.

All the algorithms for the application are written in python classes and functions. The class used in Parts of Speech Tagger for Nepali Text is ParseXMLCorpus parses the XML based Nepali National Monolingual Written Corpus to get the word-tag pair of each word in a sentence. The second class is getFeatures which extracts the word feature and POS feature of each word and stores in a feature array. DictVectorizer algorithm then transforms the lists of feature-value mappings to feature vectors. The feature vectors are then mapped in n-dimensional feature space and tags are predicted using the final class - LinearSVC. For the implementation of class LinearSVC, sklearn library is customized and used. The Web interface is designed using the Flask framework.

4.3 Description of Major Classes

The major classes in the application are:

4.3.1 ParseXMLCorpus

This is the main method which is used to parse the annotated corpus and design a training set for the application.

Input: It takes the path of the XML files of an annotated corpus stored in a directory.

Process: It then parses each XML file and stores the word-tag pair in Element Tree format.

```
Path = 'E:\\FYP\\nepali pos\\nnc_updated_ah\\cs\\a17.xml'
   X, y, X_test, y_test = [], [], [], []
     tree = ET.parse(path)
     root = tree.getroot()
     for data in root.findall('text'):
          for value in data:
            if (value.tag == 'group'):
              for body in value.findall('body'):
               for div in body.findall('div'):
                 for subdiv in div:
                   if (subdiv.tag == "head"):
                    for sentence in subdiv.findall('s'):
                      for s in sentence:
                        if (s.tag == "w"):
                           X.append(s.text)
                           y.append(s.attrib['ctag'])
```

Output: It provides the list of word-tag pairs in sentence by sentence order.

4.3.2 getFeatures

This method extracts the word features and POS features of each word in a sentence and stores in a dictionary object.

Input: It takes the list of all the words in a sentence and their indices.

Process: The word features of each word is calculated based on the indices of the words.

```
def get_wordFeatures(tokens, index):
  word = tokens[index]
  if index == 0:
    prevword = ''
    prevprevword = ''
  elif index == 1:
    prevword = tokens[index - 1]
    prevprevword = ''
  else:
    prevword = tokens[index-1]
    prevprevword = tokens[index - 2]
  if index == len(tokens) - 1:
    nextword = ''
    nextnextword = ''
  elif index == len(tokens) - 2:
    nextword = tokens[index + 1]
    nextnextword = ''
  else:
    nextword= tokens[index + 1]
    nextnextword = tokens[index + 2]
```

```
        return {
            'word': word,
            'next-word': nextword,
            'next-next-word': nextnextword,
            'prev-word': prevword,
            'prev-prev-word': prevprevword,
}
```

Output: It gives the dictionary-like object where feature names are mapped to feature values.

4.3.3 DictVectorizer

This method transforms those dictionary-like objects of features into Numpy arrays or scipy.sparse matrices for the further use with scikit-learn estimators.

Input: It takes dictionary-like object where feature names are mapped to feature values.

Process: It then converts those feature arrays to feature vectors.

```
v = DictVectorizer(sparse=False)
```

trainFeatures = [{'prev-word': '', 'next-word': 'निर्वाचन', 'word': 'आम'}, {'prev-word': 'आम', 'next-word': 'मा', 'word': 'निर्वाचन'}, {'prev-word': 'निर्वाचन', 'next-word': 'स्पष्ट', 'word': 'मा'}, {'prev-word': 'मा', 'next-word': 'बहुमत', 'word': 'स्पष्ट'}, {'prev-word': 'स्पष्ट', 'next-word': 'ल्याए', 'word': 'बहुमत'}, {'prev-word': 'बहुमत', 'next-word': 'को', 'word': 'ल्याए'}]

```
X = v.fit_transform(trainFeatures)
```

Output: It gives the Numpy arrays or scipy.sparse matrices of the feature vectors.

4.3.4 LinearSVC

Linear SVC (Support Vector Classifier) class is used to fit the data you provide and return a "best fit" hyperplane that divides, or categorizes, your data. After getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. It handles multiclass classification according to a one-vs-the-rest scheme.

Input: It takes either the Numpy arrays or scipy.sparse matrices as an input.

Process: The various functions to perform the processing with LinearSVC are:

Table 10 – Functions to Perform the LinearSVC

fit(X, y[, sample_weight])	Fit the model according to the given training data.
fit_transform(X[, y])	Fit to data, then transform it.
predict(X)	Predict class labels for samples in X.
score(X, y[, sample_weight]	Returns the mean accuracy on the given test data and labels.

```
X, y, X_test, y_test = [], [], [], []

clf = svm.SVC(kernel="linear")

clf.fit(X, y) # Use only the first 10K samples if you're
running it multiple times. It takes a fair bit

# Display Support Vectors

print(clf.support_vectors_)

# Display No of Support Vectors

print(clf.n_support_)
```

```
# Display Support vector indices

print(clf.support_)

# Predict the POS tags for the test set

y_predict = (clf.predict(X_test))

# Calculate the accuracy of the model

accuracy = ((clf.score(X_test, y_test)) * 100)
```

Output: A trained model which can be used to predict class labels based on the test features set.

4.4 System Flow

In this section implementation of the project with algorithms and their mathematical calculation have been presented in a sequential manner. A text is taken as input in order to find the corresponding POS tag.

The system flow is explained systematically using the given sample training set and testing set.

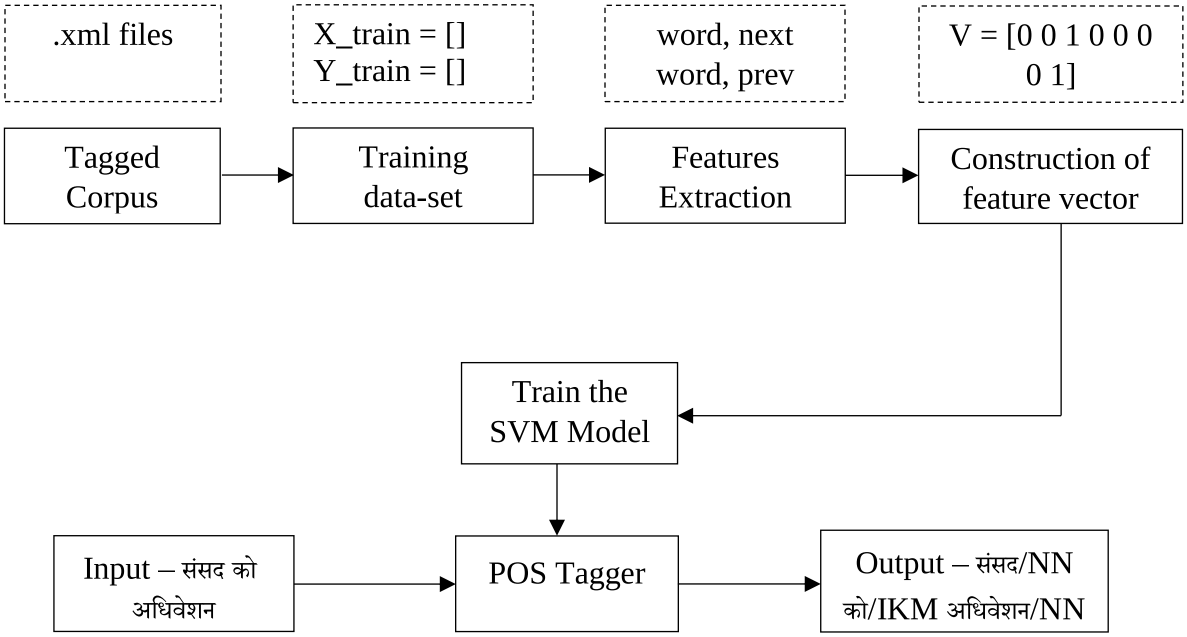


Figure 2 - System Flow Diagram of Nepali POS Tagger

First a training set is created using an annotated corpus NNC. The corpus contains the list of words and their corresponding POS tags. Let us consider the following as a training set.

Training Set – ['आम', 'JX', 'निर्वाचन', 'NN', 'मा', 'II', 'स्पष्ट', 'JX', 'बहुमत', 'NN', 'ल्याए', 'VE', 'को', 'IKM', 'नेपाली', 'NN', 'काँग्रेस', 'NN', 'को', 'IKM', 'संसदीय', 'JX', 'दल', 'NN', 'का', 'IKO', 'नेता', 'NN', 'को', 'IKM', 'रुप', 'NN', 'मा', 'II', 'श्री', 'NN', 'गिरीजा', 'NP', 'प्रसाद', 'NP', 'कोइराला', 'NP', 'ले', 'IE', 'पन्ध्र', 'MM', 'सदस्यीय', 'JX', 'मन्त्रि', 'NN', 'परिषद', 'NN', 'को', 'IKM', 'गठन', 'NN', 'गरी', 'VR', 'सक्नु', 'VI', 'भए', 'VE', 'पहि', 'II', 'मुलुक', 'NN', 'को', 'IKM', 'ध्यान', 'NN', 'राष्ट्रिय', 'JX', 'सभा', 'NN', 'को', 'IKM', 'गठन', 'NN', 'र', 'CC', 'त्यस', 'DDX', 'पछि', 'II', 'हुने', 'VN', 'संसद', 'NN', 'को', 'IKM', 'अधिवेशन', 'NN', 'तिर', 'II', 'केन्द्रित', 'JX', 'हुन', 'VI', 'थाले', 'VE', 'को', 'IKM', 'छ', 'VVYN1', '।', 'YF']

Next, the features are extracted from the Training Set to create a feature vector. As of now, only three features of each word are taken into consideration for simplification.

[{'prev-word': '', 'next-word': 'निर्वाचन', 'word': 'आम'}, {'prev-word': 'आम', 'next-word': 'मा', 'word': 'निर्वाचन'}, {'prev-word': 'निर्वाचन', 'next-word': 'स्पष्ट', 'word': 'मा'}, {'prev-word': 'मा', 'next-word': 'बहुमत', 'word': 'स्पष्ट'}, {'prev-word': 'स्पष्ट', 'next-word': 'ल्याए', 'word': 'बहुमत'}, {'prev-word': 'बहुमत', 'next word': 'को', 'word': 'ल्याए'}, {'prev-word': 'ल्याए', 'next-word': 'नेपाली', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'काँग्रेस', 'word': 'नेपाली'}, {'prev-word': 'नेपाली', 'next-word': 'को', 'word': 'काँग्रेस'}, {'prev word': 'काँग्रेस', 'next-word': 'संसदीय', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'दल', 'word': 'संसदीय'}, {'prev-word': 'संसदीय', 'next-word': 'का', 'word': 'दल'}, {'prev-word': 'दल', 'next-word': 'नेता', 'word': 'का'}, {'prev-word': 'का', 'next-word': 'को', 'word': 'नेता'}, {'prev-word': 'नेता', 'next-word': 'रुप', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'मा', 'word': 'रुप'}, {'prev-word': 'रुप', 'next-word': 'श्री', 'word': 'मा'}, {'prev-word': 'मा', 'next-word': 'गिरीजा', 'word': 'श्री'}, {'prev-word': 'श्री', 'next-word': 'प्रसाद', 'word': 'गिरीजा'}, {'prev-word': 'गिरीजा', 'next-word': 'कोइराला', 'word': 'प्रसाद'}, {'prev-word': 'प्रसाद', 'next word': 'ले', 'word': 'कोइराला'}, {'prev-word': 'कोइराला', 'next-word': 'पन्ध्र', 'word': 'ले'}, {'prev-word': 'ले', 'next-word': 'सदस्यीय', 'word': 'पन्ध्र'}, {'prev-word': 'पन्ध्र', 'next-word': 'मन्त्री', 'word': 'सदस्यीय'}, {'prev-word': 'सदस्यीय', 'next-word': 'परिषद', 'word': 'मन्त्री'}, {'prev-word': 'मन्त्री', 'next-word': 'को', 'word': 'पररषद'}, {'prev-word': 'पररषद', 'next-word': 'गठन', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'गरी', 'word': 'गठन'}, {'prev-word': 'गठन', 'next-word': 'सक्नु', 'word': 'गरी'}, {'prev-word': 'गरी', 'next word': 'भए', 'word': 'सक्नु'}, {'prev-word': 'सक्नु', 'next-word': 'पछि', 'word': 'थिए'}, {'prev-word': 'थिए', 'next-word': 'मुलुक', 'word': 'पछि'}, {'prev-word': 'पछि', 'next-word': 'को', 'word': 'मुलुक'}, {'prev-word': 'मुलुक', 'next-word': 'ध्यान', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'राष्ट्रिय', 'word': 'ध्यान'}, {'prev word': 'ध्यान', 'next-word': 'सभा', 'word': 'राष्ट्रिय'}, {'prev-word': 'राष्ट्रिय', 'next-word': 'को', 'word': 'सभा'}, {'prev-word': 'सभा', 'next-word': 'गठन', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'र', 'word': 'गठन'}, {'prev-word': 'गठन', 'next-word': 'त्यस', 'word': 'र'}, {'prev-word': 'र', 'next-word': 'पछि', 'word': 'त्यस'}, {'prev-word': 'त्यस', 'next-word': 'हुने', 'word': 'पछि'}, {'prev-word': 'पछि', 'next-word': 'संसद', 'word': 'हुने'}, {'prev-word': 'हुने', 'next-word': 'को', 'word': 'संसद'}, {'prev-word': 'संसद', 'next-word': 'अधिवेशन', 'word': 'को'}, {'prev-word': 'को', 'next-word': 'तिर', 'word': 'अधिवेशन'}, {'prev-word': 'अधिवेशन', 'next word': 'केन्द्रित', 'word': 'तिर'}, {'prev-word': 'तिर', 'next-word': 'गठन', 'word': 'केहन'}, {'prev-word':'केहन्द्रत', 'next-word': 'थाले', 'word': 'हुन'}, {'prev-word': 'हुन', 'next-word': 'को', 'word': 'थाले'}, {'prev

```
word': 'थाले', 'next-word': 'छ', 'word': 'को'}, {'prev-word': 'को', 'next-word': '।', 'word': 'छ'}, {'prev word':
'छ', 'next-word': '', 'word': '।'}]
```

The dictionary like object is then converted into Numpy arrays or scipy.sparse matrices of the feature vectors. The strings are mapped into Boolean values to be further used by learning estimators. Here, the class DictVectorizer is used.

```
[[ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  1.]]
```

The extracted features are then fitted on the learning estimator or a classification algorithm called LinearSVC(). In SVM each dimension is a feature, and all of the dimensions make up our featureset i.e. if no of features is 3, dimension of feature space also equals to 3 and so on.

Here using the fit(X, y[, sample_weight]) function calculates the separating hyperplane between the features in the input space. SVM finds one hyperplane that maximize the margin between two supporting hyperplanes out of many other hyperplanes. It finds the w and b such that the distance (margin) between H1 and H2 is maximum.

The selected hyperplane should separate training data $\{(x_i, y_i)$, $i=1....n$ and $y_i \in(+1,-1)\}$ such way that $y_i\,(w.\,x_i - b) \geq 1$.

The selected hyperplane is often referred to as a decision boundary. This decision boundary is used to separate the class of the new features in the input space.

Support Vectors are calculated in order to find an optimized separating hyperplane.

The total no of support vectors in the above training set and their indices are -

No of Support Vectors - [1 1 1 5 8 1 6 1 17 3 3 2 1 1 1 1]

Support vector indices - [39 40 21 2 16 31 41 46 6 9 14 26 33 37 44 50 12 0 3 10 23 35 47 22 1 4 7 8 11 13 15 17 24 25 27 32 34 36 38 43 45 18 19 20 5 30 49 29 48 42 28 51 52]

Since Nepali POS Tagging is multiclass classification, One Vs Rest strategy is used to binarize the multiclass problem. To train N different binary classifiers, each classifier is trained to distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which outputs the largest (most positive) value is chosen.

The LinearSVC algorithm learns the given word-tag pairs and tries to predict a class (POS tag) given a new word. Once the training model is built and a separating hyperplane is determined it can be used further for the prediction on a test set.

Now let’s see the prediction process using the given sample test sentence.

Testing Set – ['ससद ', 'को', 'अधिवेशन', 'आषाढ', 'को', 'शरु ', 'मा', 'हुने', 'राष्ट्रियसभा', 'को', 'गठन', 'यै', 'महिना', 'मा', 'भईसक्ने', 'चीन-सोभियत', 'सीमा', 'मा', 'बढी', 'सबल', 'सुरक्षा']

The features are calculated for each word in the test sentences using the similar procedure as in the training set.

Test features - [{'next-word': 'को', 'word': 'संसद ', 'prev-word': ''}, {'next-word': 'अधिवेशन', 'word': 'को', 'prev-word': 'संसद '}, {'next-word': 'आषाढ', 'word': 'अधिवेशन', 'prev-word': 'को'}, {'next-word': 'को', word': 'आषाढ', 'prev-word': 'अहधिवेशन'}, {'next-word': 'शरु ', 'word': 'को', 'prev-word': 'आषाढ'}, {'nextword': 'मा', 'word': 'शरु ', 'prev-word': 'को'}, {'next-word': 'हुने ', 'word': 'मा', 'prev-word': 'शरु'}, {'nextword': 'राष्ट्रियससभा', 'word': 'हुने', 'prev-word': 'मा'}, {'next-word': 'को', 'word': 'राष्ट्रियसभा', 'prev-word': 'हुने '}, {'next-word': 'गठन', 'word': 'को', 'prev-word': 'राष्ट्रियसभा'}, {'next-word': 'यै', 'word': 'गठन', 'prevword': 'को'}, {'next-word': 'महिना', 'word': 'यै', 'prev-word': 'गठन'}, {'next-word': 'मा', 'word': 'महिना', 'prev-word': 'यै'}, {'next-word': 'भईसक्ने', 'word': 'मा', 'prev-word': 'महिना'}, {'next-word': 'चीन-सोहियत', 'word': 'भईसक्ने', 'prev-word': 'मा'},{'next-word': 'सीमा', 'word': 'चीन-सोभियत', 'prev-word': 'भईसक्ने'}, {'next-word': 'मा', 'word': 'सीमा', 'prev-word': 'चीन-सोभियत'}, {'next-word': 'बढी', 'word': 'मा', 'prevword': 'सीमा'}, {'next-word': 'सबल', 'word': 'बढी', 'prev-word': 'मा'}, {'next-word': 'सुरक्षा', 'word': 'सबल', 'prev-word': 'बढी'}, {'next-word': '', 'word': 'सुरक्षा', 'prev-word': 'सबल'}]

The dictionary-like array object is then converted into the Numpy arrays or scipy.sparse matrices of the feature vectors.

```
[[ 0.  0.  0. ...,  0.  0.  0.]

 [ 0.  1.  0. ...,  0.  0.  0.]
 [ 0.  0.  1. ...,  0.  0.  0.]

 ...,

 [ 0.  0.  0. ...,  0.  0.  0.]

 [ 0.  0.  0. ...,  0.  0.  0.]

 [ 1.  0.  0. ...,  0.  1.  0.]]
```

The feature vector is then passed on the SVM for estimation of the input class or tags.

LinearSVM uses predict(X) function to predict class or labels for X number of test samples. The function predicts the value for each samples using the y (. –) ≥ 1 constraint. If score equals to 1 or greater than one, the sample belongs to a positive class while it belongs to a negative class (rest class in case of multiclass classification) if score equals to -1 or less than in a negative class.

The predicted tags for the given test set are –

संसद/NN को/IKM अधिवेशन/NN आषाढ/NN को/IKM शरु/NN मा/II हुने/VN राष्ट्रियसभा /NN को/IKM गठन/NN /DDX महिना/NN मा/II भईसक्ने/VN अनुमान/NN गरिन्छ/VVYN1 ,/YM यस/DDX महिना/NN भित्र/II राष्ट्रिय/JX सभा/NN को/IKM गठन/NN पनि/TT भइ/VR सक्ने/VN छ/VVYN1 |/YF

4.5 Testing

During testing process, different test cases were created other than evaluating and validating the classifier’s performance in order to make sure that the system functions properly and delivers the required output. These test cases ensured the validity and reliability of the entire system. Testing was performed on both lemmatized test data as well as raw unprocessed test data.

4.5.1 Testing in a lemmatized Test Data

The data in the Nepali National Monolingual Written Corpus is already lemmatized and annotated i.e. all the inflectional endings are already removed to return the base or dictionary form of a word, which is known as the lemma.

Input: ['संसद', 'को', 'अधिवशन', 'आषाढ', 'को', 'शरु ', 'मा', 'हुने', 'राष्ट्रियसभा', 'को', 'गठन', 'यै', 'महिना', 'मा', 'भईसक्ने', 'चीन-सोभियत', 'सीमा', 'मा', 'बढी', 'सबल', 'सुरक्षा', 'राष्ट्रिय', 'जनगणना', 'को', 'महत्व', 'भारत', 'र', 'नेपाल', 'को', 'वार्ता', 'विवाद', 'को', 'जाल', 'मा', 'राष्ट्र', 'बैंक', 'मा', 'कर्मचारी', 'निलम्बित',', 'विश्वविधालय', 'मा', 'राजिनामा', 'को', 'लहर', 'उपकुलपति', 'र', 'पदाधिकारी', 'हरु', 'को', 'नयाँ', 'नियुत्ति, 'हुने', '?', 'मुस्लिम', 'र', 'हिन्दु', 'को', 'दंगा', 'पार्टी', 'हरु', 'को', 'दिबालियापन', 'को', 'परिणाम']

Output: संसद /NN को/IKM अधिवेशन/NN आषाढ/NN को/IKM शरु /NN मा/II हुने/VN राष्ट्रियसभा/NN को/IKM गठन/NN यै/NN महिना/NN मा/II भईसक्ने/YF चीन-सोभियत/NN सीमा/NN मा/II बढी/JX सबल/NN सुरक्षा /YF राष्ट्रिय/NN जनगणना/NN को/IKM महत्व/YF भारत/NN र/CC नेपाल/NP को/IKM वार्ता/NN विवाद/NN को/IKM जाल/NN मा/YF राष्ट्र/NN बैंक/NN मा/II कर्मचारी/JX निलम्बित/YF त्रिभुवन /NN विश्वविधालय/NN मा/II राजिनामा/NN को/IKM लहर/YF उपकुलपति/NN र/CC पदाधिकारी/NN हरु/IH को/IKM नयाँ/JX नियुक्ति /NN हुने /VN ?/YF मुस्लिम /NN र/CC हिन्दु/NN को/IKM दंगा /YF पार्टी/NN हरु/IH को/IKM दिबालियापन/NN को/IKM परिणाम/YF

Testing was performed in the lemmatized test data and an average accuracy of 88% was obtained on the test data for the lemmatized test data.

4.5.2 Testing in Raw Unprocessed Test Data

The raw text available in the newspaper, books, etc. are not lemmatized and contains many different types of inflectional endings attached with the base words or dictionary form words. Testing was performed in the raw and unprocessed data from the online news site in order to evaluate the performance of the POS tagger.

The raw data was manually annotated consulting the grammatical rules and dictionary to assign accurate POS tags.

Further, the raw text was tagged using the proposed Nepali POS Tagger and its accuracy and efficiency was evaluated. [5]

Input: सरकार र डा. गोविन्द के सी प्रतिनिधिबीच आइतबार वार्ता भएन । सरकारी पक्ष र्वतवाको लागि नडाकेकोले छलफल नभएको के सी प्रतिनिधि डा. अभिषेकराज सिंहले बताए ।

डा. के सी १४ दिनदेखि त्रिवि शिक्षण अस्पताल हातामा आमरण अनशन गरिरहहेको छन् । चिकित्सा शिक्षा र सेवामा सुधारको लागि उस्ताउस्तै मागसहित यो उनको ११ औं आमरण अनशन हो।

Manually Annotated: सरकार/NN र/CC डा./FB गोविन्द/NP के सी/NP प्रतिनिधिबीच/NN आइतबार/NP वार्ता/NN भएन/VI ।/YF सरकारी/JX पक्ष/NN वर्ताको/NN लागी/II नडाकेको/VI छलफल/NN नभएको/VI के सी/NP प्रतिनिधि/NN डा./FB अभिषेकराज/NP सिंहले/NP बताए/VVYN1 ।/YF

डा./FB के सी/NP १४/MM दिनदेखि/NN त्रिवि/NP शिक्षण/NP अस्पताल/NN हातामा/NN आमरण/JX अनशन/NN गरिरहेका/VVMX2 छन्/VVYX2 ।/YF चिकित्सा/NP शिक्षा/NN र/CC सेवामा/JX सुधारको/NN लागि/II उस्ताउस्तै /RJ मागसहित/NN यो/DDX उनको/DDX ११/MM औ/MOX आमरन/JX अनशन/NN हो/VVYN1 ।/YF

Tagger's Output: सरका/NN र/CC डा./JX गोविन्द्र/NP के सी/NP प्रतिनिधिबीच/NN आइतबार/JX वार्ता/NN भएन/VVYN1 ।/YF सरकारी/NN पक्ष/NN वार्ताको/NN लागि/II नडाकेको/NN छलफल/NN नभएको/NN के सी/NN प्रतिनिधि/NN डा./FB अभिषेकराज/NP सिंह/NP बताए/VVYN1 ।/YF

डा./FB के सी/JX १४/NP दिनदखि/NN त्रिवि/NN शक्षण/NN अस्पताल/NN हातामा/JX आमरन/JX अनशण/NN गरिरहका/JX छन्/VVYX2 । ।/YF चिकित्सा/NN शिक्षा/NN र/CC सेवामा/JX सुधारको/NN लागि/II उस्ताउस्तै/JX मागसहित/NN यो/DDX उनको/NN ११/JX औं/JX आमरण/JX /N अनशन/N हो/VVYN1 ।/YF

Hence, the average accuracy of 72 % was obtained on the test data for the raw and unprocessed test data.

4.6 Result and Discussion

As per the result obtained in the testing phase, there lies drastic variation in the efficiency of tagger between the lemmatized test data and unprocessed raw test data.

Following are the important findings obtained from analyzing the morphosyntactic units in

Nepali National Monolingual Written Corpus [5].

- Postpositions *haruu, ko/kii/kaa, le, laaii and maa, baaTa, sanga, dekhi* are necessarily tokenized separately to the forms to which they are attached, and tagged separately.

- Gender distinction like *o/ii/aa distinction* is ignored on nouns, but not on adjectives, pronouns, adjectives, non-finite verbs.

- Nepali has a great range of verb inflections. If every distinction made in the verb system were to be indicated by a separate tag, then the tags for verbs would become entirely unmanageable.

These findings clearly direct to the differences between the lemmatized and unprocessed raw text.

As shown above in the tagger output for the raw unprocessed test data, the number of false predictions for JX – Unmarked Adjective tags is high, while it confuses between the Adjective and other forms of Speech tags. The presence and absence of the inflectional endings attached at the end of the words increase the complexity for the tagger.

Thus, the tagger performs better in case of lemmatized test data compared to unprocessed raw test data since it was trained using the lemmatized corpus.

CHAPTER 5: CONCLUSIONS AND LIMITATIONS

5.1 Conclusions

Parts of Speech Tagger for Nepali Text is an important task in the field of NLP. It is useful for many application domains such as Machine Translation, Speech Recognition and Speech Synthesis, Grammar Checker, Named Entity Recognition, Question Answering and Information Retrieval etc. The research works only considered English, Spanish, Hindi and other languages and a tagger developed for one language does not necessarily work for other languages. Some of the reasons for this is variation in language characteristics, vocab size, etc. Hence, the SVM based tagger is developed which assigns the most appropriate parts of speech tag for each word of Nepali text. From the testing done for lemmatized text the average accuracy obtained is 88% and for unprocessed raw text, the accuracy obtained is 72%. However, the SVM based POS Tagging System does not perform well on testing. Hence, it is not recommended and further improvements need to be made to be used for Nepali text.

5.2 Limitations

The limitations of the system are as follows:

- The different steps of text preprocessing like stop words removal, stemming and lemmatization are not considered.
- The train word sample cannot represent all the Nepali words.
- The SVM based POS tagger uses different sets of features to construct the features vectors. Hence, it is comparatively slower than other taggers.

REFERENCES

[1].B. Prasain, LP. Khatiwada, B.K. Bal, and P. Shrestha. “Part-of-speech Tagset for Nepali”, *Madan Puraskar Pustakalaya*, 2008.

[2].T. Bahadur Shahi, T. Nath Dhamala and B. Balami, "Support Vector Machines based Part of Speech Tagging for Nepali Text", *International Journal of Computer Applications*, vol. 70, no. 24, pp. 38-42, 2013.

[3].B.K. Bal, and P. Shrestha, *Reports on Computational Grammar* Madan PuraskarPustakalaya, Patan Dhoka, Lalitpur, Kathmandu.

[4].T. Brants, TnT -- A Statistical Part-of - Speech Tagger, *In: Proceedings of the 6th Applied NLP Conference, (ANLP-2000)*, 2000.

[5].E. Brill, A Simple Rule-Based Part of Speech Tagger. *In: Proceedings of the ThirdConference on Applied Natural Language Processing (ANLP-1992)*, Torento, 1992.

[6].N. Cristianini and J. S. Taylor, An Introduction to Support Vector Machines and Other Kernel- based Learning Methods, (Cambridge University Press 2002), pp 126.

[7].K. Church, A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *In:Proceedings of the Second Conference on Applied Natural Language Processing, ACL*, 1988.

[8].D. Cutting, J. Kupiec, J. Pederson and P. Sibun, A Practical Part-of-Speech Tagger, *n: Proceedings of the Third Conference on Applied Natural Language Processing, ACL,* 1992.

[9].A. Ekbal and S. Bandopadhya, Part of Speech Tagging in Bengali Using Support Vector Machine*, In: Proceeding of IEEE* 2008.

[10]. Jesus Giménez and Lluís Márquez , SVMTool: A General POS Tagger Generator Based on Support Vector Machines, *In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004).* Lisbon, Portugal, 2004.

[11].R. Garside G. Leech and G. Sampson, *CLAWS Word-tagging system*, 1987.

[12].Hardie, The Computational Analysis of Morphosyntactic Categories in Urdu, (PhD Thesis, Department of Linguistics and Modern English Language, Lancaster University, 2003)

[13].Z. Harris, *String Analysis of Sentence Structure,* The Hague: Mouton, 1962.

[14].M. R. Jaishi., Hidden Markov Model Based Probabilistic Part Of Speech Tagging For Nepali Text, (Masters Dissertation, Central Department of Computer Science and IT ,Tribhuvan University, Nepal).

[15].T. Joachims, *Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning,* B., Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[16].Archit Yajnik “Parts of Speech Tagging using Statistical Approach for Nepali Text” International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:11, No:1, 2017

[17].Archit Yajnik “ANN Based POS Tagging for Nepali Text” International Journal of Natural Language computing (IJNLC) Vol.7, No, 3, June 2018

[18].Archit Yajnik “General Regression Neural Network Based POS Tagging for Nepali Text” Department of mathematics, Sikkim Manipal University, Sikkim , India, 2018

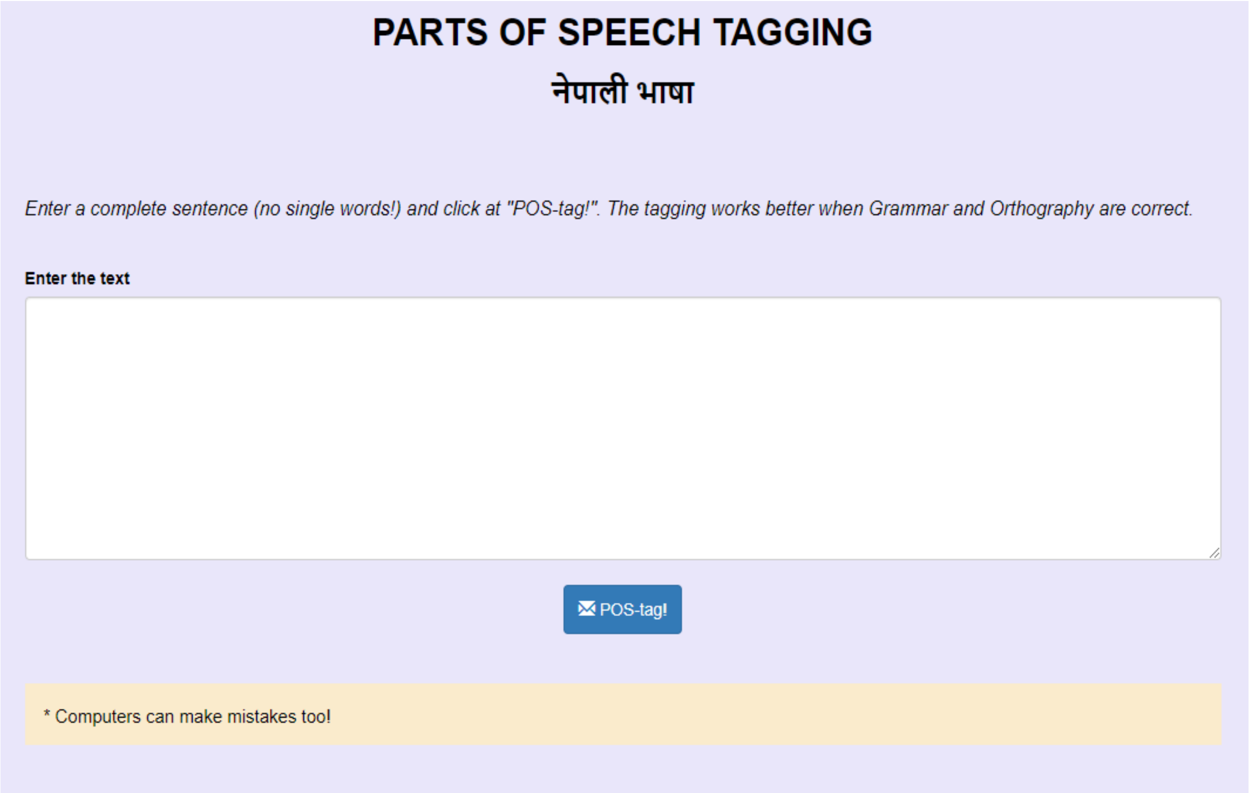
APPENDIX I

```
<cesHeader version="2.1">
  <fileDesc>
    <titleStmt>
      <h.title>NNC-CS: sample A17</h.title>
      <respStmt>
        <respType>Electronic version created by</respType>
        <respName>Nelralec / Bhasha Sanchar Project (भाषा सञ्चार)</respName>
      </respStmt>
      <respStmt>
        <respType>transcribed by</respType>
        <respName>लक्ष्मीप्रसाद भट्टराई</respName>
      </respStmt>
    </titleStmt>
    <extent>
      <wordCount>2036</wordCount>
      <byteCount units="kb">137</byteCount>
    </extent>

    <sourceDesc>
      <biblStruct>
        <analytic>
          <h.title>संसदको अधिवेशन आषाढको शुरुमा हुने राष्ट्रियसभाको गठन यै महिनामा भइसक्ने</h.title>
          <h.author>अज्ञात</h.author>
<h.title>चीन-सोभियत सीमामा बढी सबल सुरक्षा</h.title>
          <h.author>अज्ञात</h.author>
<h.title>राष्ट्रिय जनगणनाको महत्व</h.title>
```

```
           <h.author>अज्ञात</h.author>
<h.title>भारत र नेपालको वार्ता विवादको जालमा</h.title>
           <h.author>अज्ञात</h.author>
<h.title>राष्ट्रबैंकमा कर्मचारी निलम्बित</h.title>
           <h.author>अज्ञात</h.author>
<h.title>त्रिभुवन विश्वविद्यालयमा राजिनामाको लहर</h.title>
           <h.author>अज्ञात</h.author>
<h.title>मुस्लिम र हिन्दुको दंगा</h.title>
           <h.author>अज्ञात</h.author>
         </analytic>
         <monogr>
           <h.title>ज्योति साप्ताहिक</h.title>
           <imprint>
             <publisher>कुलदिप प्रसाद 'कुमुद'</publisher>
             <pubDate>२०४८-०२-२२; २०४८-०५-२६; २०४८-०४-१५; २०४८-०५-१२</pubDate>
             <pubPlace>जनकपुरधाम</pubPlace>
           </imprint>
         </monogr>
       </biblStruct>
     </sourceDesc>
   </fileDesc>

<body>
<div type="unspec">
<head>
<s n="1">
<w ctag="NN">संसद</w> <w ctag="IKM">को</w> <w ctag="NN">अधिवेशन</w>
<w ctag="NN">आषाढ</w> <w ctag="IKM">को</w> <w ctag="NN">शुरु</w>
<w ctag="II">मा</w> <w ctag="VN">हुने</w> <w ctag="NN">राष्ट्रियसभा</w>
```

```
<w ctag="IKM">को</w> <w ctag="NN">गठन</w>
<w ctag="DDX">यै</w> <w ctag="NN">महिना</w> <w ctag="II">मा</w>
<w ctag="VN">भईसक्ने</w>
</s>
</head>

<p>
<s n="2">
<w ctag="NP">काठमाडौ</w> <w ctag="YF">।</w>
</s>
</p>

<p>
<s n="3">
<w ctag="JX">आम</w> <w ctag="NN">निर्वाचन</w> <w ctag="II">मा</w>
<w ctag="JX">स्पष्ट</w>
<w ctag="NN">बहुमत</w> <w ctag="VE">ल्याए</w> <w ctag="IKM">को</w>
<w ctag="NN">नेपाली</w>
<w ctag="NN">काँग्रेस</w> <w ctag="IKM">को</w> <w ctag="JX">संसदीय</w>
<w ctag="NN">दल</w> <w ctag="IKO">का</w> <w ctag="NN">नेता</w>
<w ctag="IKM">को</w> <w ctag="NN">रुप</w> <w ctag="II">मा</w> <w ctag="NN">श्री</w>
<w ctag="NP">गिरीजा</w> <w ctag="NP">प्रसाद</w> <w ctag="NP">कोइराला</w>
<w ctag="IE">ले</w> <w ctag="MM">पन्द्रह</w> <w ctag="JX">सदस्यीय</w>
<w ctag="NN">मन्त्रि</w> <w ctag="NN">परिषद</w> <w ctag="IKM">को</w>
<w ctag="NN">गठन</w> <w ctag="VR">गरी</w>
<w ctag="VI">सक्नु</w> <w ctag="VE">भए</w> <w ctag="II">पछि</w>
<w ctag="NN">मुलुक</w> <w ctag="IKM">को</w> <w ctag="NN">ध्यान</w>
<w ctag="JX">राष्ट्रिय</w> <w ctag="NN">सभा</w> <w ctag="IKM">को</w>
<w ctag="NN">गठन</w> <w ctag="CC">र</w> <w ctag="DDX">त्यस</w>
```

```
<w ctag="II">पछि</w> <w ctag="VN">हुने</w> <w ctag="NN">संसद</w>
<w ctag="IKM">को</w> <w ctag="NN">अधिवेशन</w> <w ctag="II">तिर</w>
<w ctag="JX">केन्द्रित</w> <w ctag="VI">हुन</w> <w ctag="VE">थाले</w>
<w ctag="IKM">को</w> <w ctag="VVYN1">छ</w> <w ctag="YF">।</w>
 </s>
</p>
```



PARTS OF SPEECH TAGGING

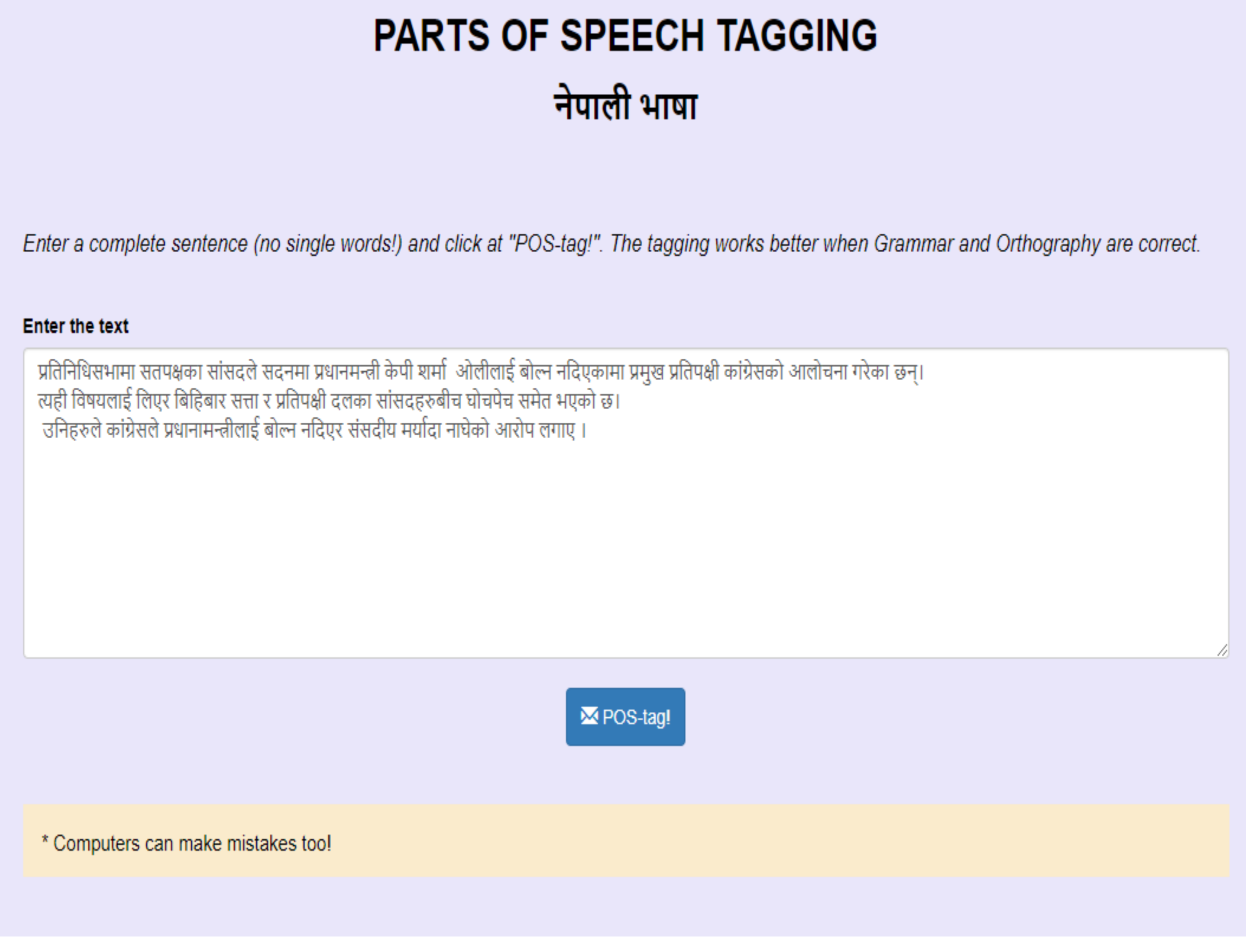
नेपाली भाषा

Enter a complete sentence (no single words!) and click at "POS-tag!". The tagging works better when Grammar and Orthography are correct.

Enter the text

* Computers can make mistakes too!

Figure 10 – User Interface of Nepali POS Tagger



PARTS OF SPEECH TAGGING

नेपाली भाषा

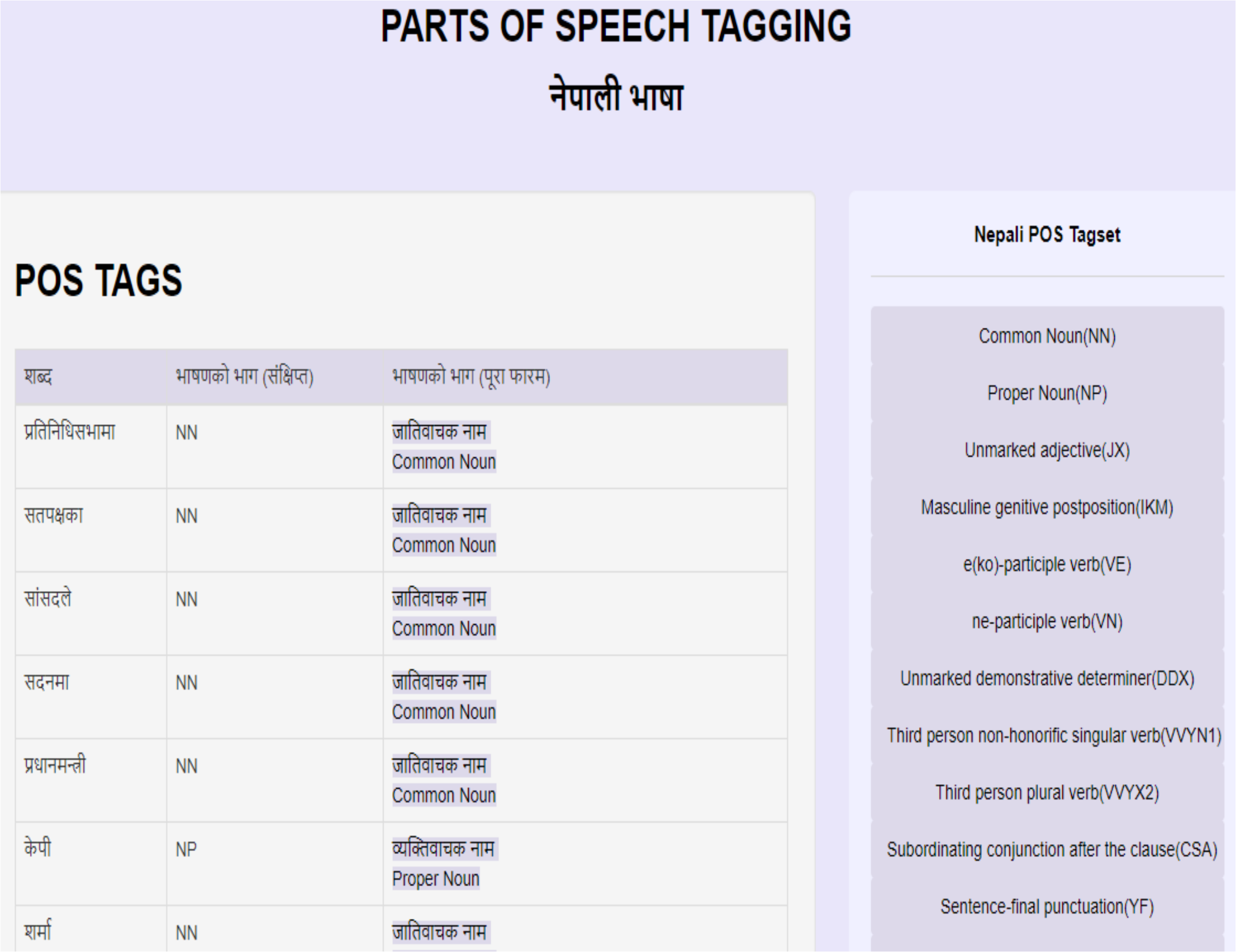
Enter a complete sentence (no single words!) and click at "POS-tag!". The tagging works better when Grammar and Orthography are correct.

Enter the text

प्रतिनिधिसभामा सतपक्षका सांसदले सदनमा प्रधानमन्त्री केपी शर्मा ओलीलाई बोल्न नदिएकामा प्रमुख प्रतिपक्षी कांग्रेसको आलोचना गरेका छन्।
त्यही विषयलाई लिएर बिहिबार सत्ता र प्रतिपक्षी दलका सांसदहरुबीच घोचपेच समेत भएको छ।
 उनिहरुले कांग्रेसले प्रधानामन्त्रीलाई बोल्न नदिएर संसदीय मर्यादा नाघेको आरोप लगाए ।

* Computers can make mistakes too!

Figure 11 – User Interface with an Input Text



PARTS OF SPEECH TAGGING

नेपाली भाषा

POS TAGS

शब्द	भाषणको भाग (संक्षिप्त)	भाषणको भाग (पूरा फारम)
प्रतिनिधिसभामा	NN	जातिवाचक नाम Common Noun
सतपक्षका	NN	जातिवाचक नाम Common Noun
सांसदले	NN	जातिवाचक नाम Common Noun
सदनमा	NN	जातिवाचक नाम Common Noun
प्रधानमन्त्री	NN	जातिवाचक नाम Common Noun
केपी	NP	व्यक्तिवाचक नाम Proper Noun
शर्मा	NN	जातिवाचक नाम

Nepali POS Tagset

Common Noun(NN)

Proper Noun(NP)

Unmarked adjective(JX)

Masculine genitive postposition(IKM)

e(ko)-participle verb(VE)

ne-participle verb(VN)

Unmarked demonstrative determiner(DDX)

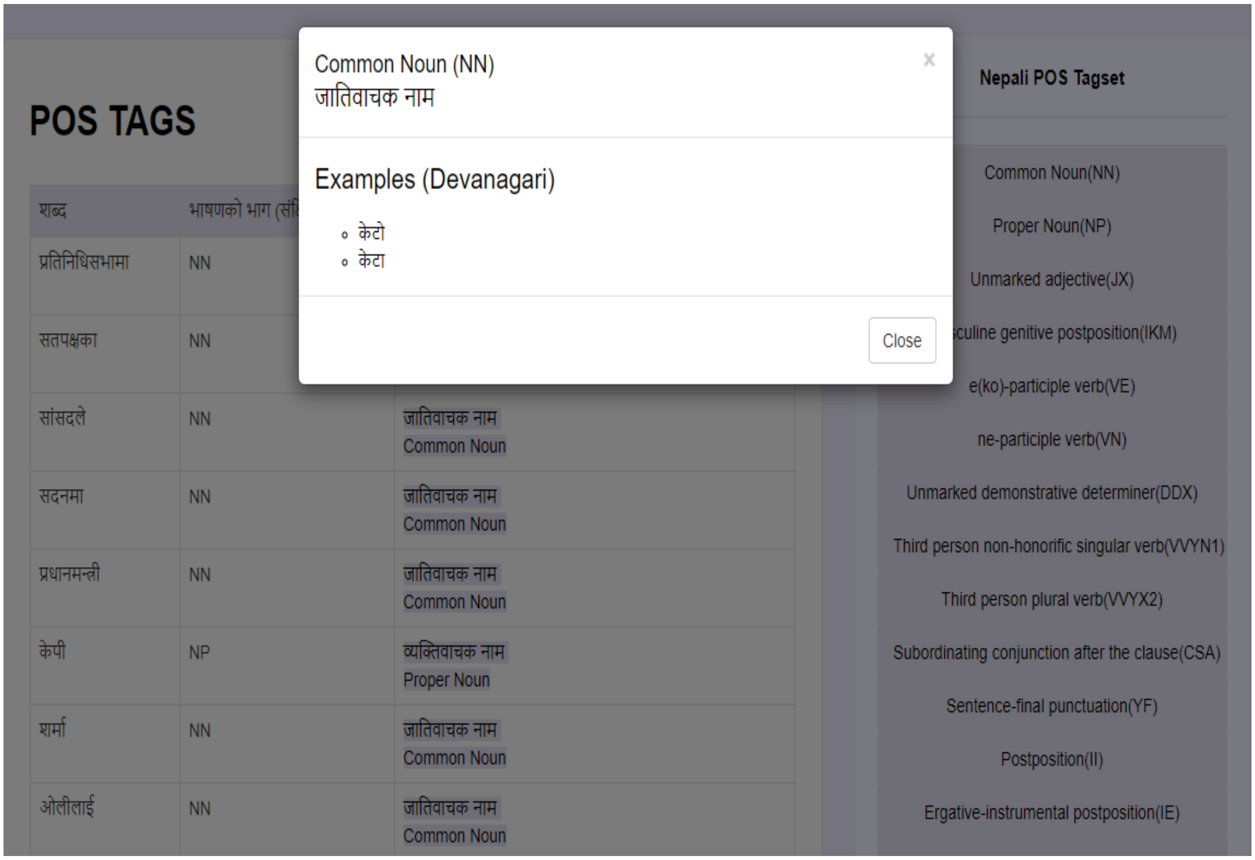
Third person non-honorific singular verb(VVYN1)

Third person plural verb(VVYX2)

Subordinating conjunction after the clause(CSA)

Sentence-final punctuation(YF)

Figure 12 – Output Result Displayed by the System



Common Noun (NN)
जातिवाचक नाम

Examples (Devanagari)

- केटो
- केटा

POS TAGS

शब्द	भाषणको भाग (सं	
प्रतिनिधिसभामा	NN	
सतपक्षका	NN	
सांसदले	NN	जातिवाचक नाम Common Noun
सदनमा	NN	जातिवाचक नाम Common Noun
प्रधानमन्त्री	NN	जातिवाचक नाम Common Noun
केपी	NP	व्यक्तिवाचक नाम Proper Noun
शर्मा	NN	जातिवाचक नाम Common Noun
ओलीलाई	NN	जातिवाचक नाम Common Noun

Nepali POS Tagset

- Common Noun(NN)
- Proper Noun(NP)
- Unmarked adjective(JX)
- culine genitive postposition(IKM)
- e(ko)-participle verb(VE)
- ne-participle verb(VN)
- Unmarked demonstrative determiner(DDX)
- Third person non-honorific singular verb(VVYN1)
- Third person plural verb(VVYX2)
- Subordinating conjunction after the clause(CSA)
- Sentence-final punctuation(YF)
- Postposition(II)
- Ergative-instrumental postposition(IE)

Figure 13 – Tag-set Example Displayed by the System

Table 11 - 112 Tag-set of Nepali National Monolingual Written Corpus

Category definition	**Examples** **(Latin)**	**Examples** **(Devnagari)**	**Tag**
Common noun	keTo, keTaa, kalam	केटो, केटा,कलम	NN
Proper noun	raam	राम	NP
Masculine adjective	moTo, raamro	मोटो ,राम्रो	JM
Feminine adjective	moTii, raamrii	मोटी ,राम्री	JF
Other-agreement adjective	moTaa, raamraa	मोटा ,राम्रा	JO
Unmarked adjective	saphaa, dhanii, asal	सफा ,धनी ,असल	JX
Sanskrit-derived comparative or superlative adjective	uccatar, uccatam	उच्चतर ,उच्चतम	JT
First person pronoun	*ma, haamii, mai#*	म ,हामी ,मै#	PMX
First person possessive pronoun with masculine agreement	*mero, haamro*	मेरो ,हाम्रो	PMXKM
First person possessive pronoun with feminine agreement	*merii, haamrii*	मेरी ,हाम्री	PMXKF
First person possessive pronoun with other agreement	*meraa, haamraa*	मेरा ,हाम्री	PMXKO
Non-honorific second person pronoun	*ta~, tai#*	तँ ,तँ#	PTN
Non-honorific second person possessive pronoun with masculine agreement	*tero*	तेरो	PTNKM

Non-honorific second person possessive pronoun with feminine agreement	*terii*	तेरी	PTNKF
Non-honorific second person possessive pronoun with other agreement	*teraa*	तेरा	PTNKO
Medial-honorific second person pronoun	*timrii*	तिम्री	PTM
Medial-honorific second person possessive pronoun with masculine agreement	*timro*	तिम्रो	PTMKM
Medial-honorific second person possessive pronoun with feminine agreement	*timrii*	तिम्री	PTMKF
Medial-honorific second person possessive pronoun with other agreement	*timraa*	तिम्रा	PTMKO
High-honorific second person pronoun	*tapaai~, hajur*	तपाईँ ,हजुर	PTH
High-honorific unspecifiedperson pronoun	*yahaa~, wahaa~ 7*	यहाँ ,वहाँ	PXH
Royal-honorific unspecifiedperson pronoun	*sarkaar, mausuph*	सरकार ,मौसुफ	PXR
Reflexive pronoun	*aaphuu*	आफू	PRF
Possessive reflexive pronoun with masculine agreement	*aaphno*	आफ्नो	PRFKM

Possessive reflexive pronoun with feminine agreement	*aaphnii*	आफ्नी	PRFKF
Possessive reflexive pronoun with other agreement	*aaphnaa*	आफ्ना	PRFKO
Masculine demonstrative determiner	*yasto, yatro*	यस्तो ,यत्रो	DDM
Feminine demonstrative determiner	*yastii, yatrii*	यस्ती ,यत्री	DDF
Other-agreement demonstrative determiner	*yastaa, yatraa*	यस्ता ,यत्रा	DDO
Unmarked demonstrative determiner	*yo, yas#, yi, yin#, yinii, yati, yatti*	यो ,यस# ,यी ,यिन#	DDX
Masculine interrogative determiner	*kasto, katro*	कस्तो ,कत्रो	DKM
Feminine interrogative determiner	*kastii, katrii*	कस्ती ,कत्री	DKF
Other-agreement interrogative determiner	*kastaa, katraa*	कस्ता ,कत्रा	DKO
Unmarked interrogative determiner	*ko, kas#, ke, kun, kati*	को ,कस ,#के ,कुन ,कति	DKX
Masculine relative determiner	*jasto, jatro*	जस्तो ,जत्रो	DJM
Feminine relative determiner	*jastii, jatrii*	जस्ती ,जत्री	DJF
Other-agreement relative determiner	*jastaa, jatraa*	जस्ता ,जत्रा	DJO
Unmarked relative determiner	*jo, jas#, je, jati, josukai*	जो ,जस ,#जे ,जति , जोसुकै	DJX
Masculine general determiner-pronoun	arko	अर्को	DGM

Feminine general determine-pronoun	arkii	अर्की	DGF
Other-agreement general determiner-pronoun	arkaa	हर्का	DGO
Unmarked general determiner-pronoun	haruu	हरू	DGX
Infinitive verb	garnu, garna, garnaa, nagarnu, nagarna, nagarnaa	गर्नु ,गर्न ,गर्ना ,नगर्नु , नगर्न ,नगर्ना	VI
Masculine d-participle verb	gardo, nagardo	गर्दो ,नगर्दो	VDM
Feminine d-participle verb	gardii, nagardii	गर्दी ,नगर्दी	VDF
Other-agreement d-participle verb	gardaa, nagardaa	गर्दा ,नगर्दा	VDO
Unmarked d-participle verb	gardai, nagardai	गर्दै ,नगर्दै	VDX
e(ko)-participle verb	gae (as in gae saal), gare (as in garejati or gareko)	गरे	VE
ne-participle verb::	garne, nagarne	गर्ने ,नगर्ने ,	VN
Sequential participle-converb	garera, gariikana, garii, nagarera, nagariikana, nagarii	गरेर ,गरीकन ,गरी , नगरेर ,नगरीकन ,नगरी	VQ
Command-form verb, nonhonorific	gar, jaa	गर् ,जा	VCN
Command-form verb, midhonorific	gara, jaau, jaao	गर ,जाऊ ,जाओ	VCM
Command-form verb, highhonorific	garnos, jaanos	गर्नोस् ,जानोस्	VCH

Subjunctive / conditional e-form verb	gare, nagare	गरे ,नगरे	VS
i-form verb	gari	गरी	VR
First person singular verb	gare~, garthe~, garina~, chu, hu~, garnechu	गरे ,गर्थे ,गरिन ,छु, हुँ ,गरनेछु	VVMX1
First person plural verb	garyau~, garthyau~, garenau~, chau~, hau~, garnechau~	गयौं ,गर्थ्यौं,गरेनौ ,छौ ,हौ , गर्नेछौ	VVMX2
Second person non-honorific singular verb	garis, garthis, garinas, chas, hos, garnechas	गरिस् ,गर्थिस् ,गरिनस् , छस् ,होस् ,गर्नेछस्	VVTN1
Second person plural (or medialhonorific singular) verb	garyau, garthyau, garenau, chau, hau, garnechau	गर्यो ,गर्थ्यौं ,गरेनौ ,छौ , हौ ,गर्नेछौ	VVTX2
Third person non-honorific singular verb	garyo, garthyo, garena, cha, ho, garnecha	गयौं ,गर्थ्यो ,गरेन ,छ ,हो , गर्नेछ	VVYN1
Third person plural (or medialhonorific singular) verb	gare, garthe, garenan, chan, hun, garnechan	गरे ,गर्थे ,गरेनन् ,छन् ,हुन् , गर्नेछन्	VVYX2
Feminine second person nonhonorific singular verb	garlis, ches, garthis	गर्लिस ,छेस् ,गर्थिस्	VVTN1F
Feminine second person nonhonorific singular verb	garthyau, chyau	गथ्यौ ,छ्यौ	VVTM1F
Feminine third person medialhonorific singular verb	garina, garii, che, garthii	गरिन ,गरी ,छे ,गर्थी , गरिन् ,गरी ,छे ,गर्थी	VVYN1F

Feminine third person medialhonorific singular verb	garin, garthin, garinan, chin	गरिन ,गर्थिन ,गरिनन् ,छिन्	VVYM1F
First person singular optative verb	jaau~, garu~	जाऊँ ,गरूँ	VOMX1
First person plural optative verb	jaaau~, garau~	जाऔँ ,गरौँ	VOMX2
Second person non-honorific singular optative verb	gaes, gares	गएस् ,गरेस्	VOTN1
Second person plural (or medialhonorific singular) optative verb	gae, gare	गए ,गरे	VOTN2
Third person non-honorific singular optative verb	jaaos, garos	जाऔस् ,गरोस्	VOYX1
Third person plural (or medial honorific singular) optative verb	jaauun, garuun	जाऊन् ,गरून्	VOYX2
Adverb	raamrarii, ekdam, chiTo	राम्ररी ,एकदम ,छिटो	RR
Demonstrative adverb	yataa, utaa, tyataa; ahile, ahilyai, yahii~, yasarii, aba	यता ,उता ,त्यता ,अहिले , अहिल्यै ,यहिँ ,यसरी ,अब	RD
Interrogative adverb	kataa, kahaa~, kahile, kasarii	कता ,कहाँ ,कहिले	RK
Relative adverb	jataa, jahaa~, jahile, jasarii, jaba	जता ,जहाँ ,जहिले	RJ
Postposition	agaaDi, pachaaDi, baaTa, dwaaraa, maa, maathi, saath,	अगाडि ,पछाडि , बाट,द्वारा ,मा ,माथि	II

	puurvak, tira, tarpha, vasa, sanga, binaa		
Plural-collective postposition	*haruu*	हरू	IH
Ergative-instrumental postposition	*le*	ले	IE
Accusative-dative postposition	*laaii*	लाई	IA
Masculine genitive postposition	*ko*	को	IKM
Feminine genitive postposition	*kii*	की	IKF
Other-agreement genitive postposition	*kaa*	का	IKO
Cardinal number	*ek, eu#, yau#, dui, tin, caar, paa~c*	एक ,दुई ,तीन ,चार ,पाँच	MM
Masculine ordinal number	*pahilo, dosro, tesro, cautho*	पहिलो ,दोस्रो ,तेस्रो ,चौथो	MOM
Feminine ordinal number	*pahilii, dosrii, tesrii, cauthii*	पहिली ,दोस्री ,तेस्री ,चौथी	MOF
Other-agreement ordinal number	*pahilaa, dosraa, tesraa, cauthaa*	पहिला ,दोस्रा ,तेस्रा ,चौथा	MOO
Unmarked ordinal number	*paa~cau~*	पाँचौँ	MOX
Masculine numeral classifier	*#To [as in euTo]*	#टो	MLM
Feminine numeral classifier	*(#)waTii, #Tii, #oTii, #auTii*	(#)वटी# ,टी# ,ओटी	MLF
Other-agreement numeral classifier	*(#)waTaa, #Taa, #oTaa, #auTaa*	(#)वटा# ,टा# ,ओटा , #औटा	MLO
Unmarked numeral classifier	*(#)janaa*	(#)जना	MLX

Coordinating conjunction	ra, tathaa	र ,तथा	CC
Subordinating conjunction appearing after the clause it subordinates	bhanne, bhanii, bhanera	भने ,भनी ,भनेर	CSA
Subordinating conjunction appearing before the clause it subordinates	ki, yadi, yaddyapi, kinaki	कि ,यदि ,यधपि ,किनकि	CSB
Particle	nai, caahi~, pani, hai, ra, re, kyaare, ho, khai	नै ,चै ,पनि ,है ,र ,रे , क्यारे ,हो ,खै	TT
Question marker	ke	के	QQ
Interjection	oho, aahaa, hare	ओहो ,आहा ,हरे	UU
Possessive reflexive pronoun without agreement	*merai*	मैरे	PMXKX
Non-honorific second person possessive pronoun without agreement	*terai*	तैरे	PTNKX
Medial-honorific second person possessive pronoun without agreement	*timrai*	तिम्रै	PTMKX
Possessive reflexive pronoun without agreement	*aaphnai*	आफ्नै	PRFKX
Unmarked genitive postposition	*kai*	कै	IKX
Sentence-final punctuation	? ! . | ||		YF
Sentence-medial punctuation	; ,: :-- /		YM

Quotation marks	‘ “		YQ
Brackets	( ) { } [ ]		YB
Foreign word in Devanagari			FF
Foreign word, not in Devanagari			FS
Abbreviation	M.P.P.	म .पु .पु.	FB
Mathematical formula (and similar)	$e=mc^2$		FO
Letter of the alphabet			FZ
Unclassifiable			FU
Null tag: an element of the text which does not need a tag	<p>		NULL