DWIT COLLEGE

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Institute of Science and Technology

PARTS-OF-SPEECH TAGGER FOR NEPALI TEXT USING SVM

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

Asmita Subedi

July, 2017

DWIT College

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by ASMITA SUBEDI entitled **“PARTS-OF-SPEECH TAGGER FOR NEPALI TEXT USING SVM”** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

…………………………………………

Birodh Rijal

Lecturer

DWIT College

DWIT College

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

LETTER OF APPROVAL

This is to certify that this project prepared by ASMITA SUBEDI entitled **“PARTS-OF-SPEECH TAGGER FOR NEPALI TEXT USING SVM”** 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.

…………………………………… Birodh Rijal [Supervisor] Lecturer DWIT College	………………………………………… Hitesh Karki Chief Academic Officer DWIT College
………………………………………….. Dhiraj Kedar Pandey [External Examiner] IOST, Tribhuvan University	………………………………………….. Dr. Sunil Chaudhary [Internal Examiner] Head of R&D DWIT College

ACKNOWLEDGEMENT

I would like to thank Mr. Birodh Rijal, 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 am grateful to Dr. Bal Krishna Bal, Assistant Professor, Kathmandu University, for the linguistic support and data resources. He provided me an overview of the existing primary resources and applications in the field of Natural Language Processing (NLP) for the Nepali language. I am thankful to the Language Technology Kendra for providing me with the Nepali National Monolingual Written Corpus.

At last, my sincere thanks goes to my parents and members 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.

Asmita Subedi
TU Exam Roll no: 3019/070

STUDENT'S DECLARATION

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

... ...

Asmita Subedi

Date: July, 2017

ABSTRACT

Parts-of-Speech Tagger for Nepali Text using SVM is an application that assigns parts of speech like noun, pronoun, verb, adverb and other lexical tags to each word in Nepali text based on both its definition, as well as its context. The tagger is built using the Support Vector Machine learning framework that is trained with 80,000 lemmatized words from the Nepali National Monolingual Written Corpus.

The average accuracy of 88% and 72% was obtained for lemmatized text and unprocessed raw text tagging system respectively.

Keywords**:** Support Vector Machine, Natural Language Processing, Supervised Machine Learning, Parts-of Speech Tagger, Tag-set, Nepali National Monolingual Written Corpus.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

POS:	Parts of Speech
SVM:	Support Vector Machine
HMM:	Hidden Markov Model
HTML:	Hyper Text Markup Language
NLP:	Natural Language Processing
NLU:	Natural Language Understanding
NLG:	Natural Language Generation

CHAPTER 1: INTRODUCTION

1.1 Background

Natural language processing (NLP) is concerned with programming machines to understand and generate the natural language. Thus NLP consists of two components: natural language understanding (NLU) and natural language generation (NLG).

NLU consist of 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 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.

Part-of-Speech (POS) Tagging is a task of assigning the appropriate POS 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. It has diverse application domain including speech recognition, speech synthesis, grammar checker, machine translation, information retrieval and information extraction.

1.2 Problem Statement

Parts-of-Speech tagging is the initial and most significant step in Natural Language Processing. Further processing like chunking, parsing, translation, etc. are all dependent on POS tagging. It can be further used in application domains like speech recognition, machine translation, and information retrieval. While POS tagging is considered to be very crucial in NLP, manually assigning tags to a large corpus becomes work-intensive. Hence it is required to build computer programs (taggers) to automatically assign POS tags to words in natural language text.

Many taggers have been built for different language like English, Spanish, Hindi, etc. but there are only a few taggers for the Nepali language. The resources and applications in the Nepali language are very limited and there is few advancement in this field recently. Parts-of-Speech Tagger for Nepali Text using SVM is a web-based application which allows people to enter an input text and get the corresponding lexical tags for each word in the text. The module can then be further used for other NLP applications.

1.3 Objectives

1.3.1 General objective

To implement SVM algorithm for developing a training model from an annotated corpus so that it automatically assigns parts of speech tags to each word given an input text.

1.3.2 Specific objective

To assign parts-of speech tags to each word in a given sentence based on its definition and context.

1.4 Scope

Parts-of-Speech Tagger for Nepali Text can be used by linguistics, students, and teachers to learn as well as to determine the grammatical errors in the Nepali texts. Also, this module can be applied for building other NLP applications as well.

1.5 Limitation

a) Text pre-processing is not implemented.
b) The sample of trained words cannot represent all of the Nepali words.
c) Parts-of-Speech Tagger for Nepali Text using SVM is comparatively slower than other taggers.

1.6 Outline of Document

The report is organized as follows

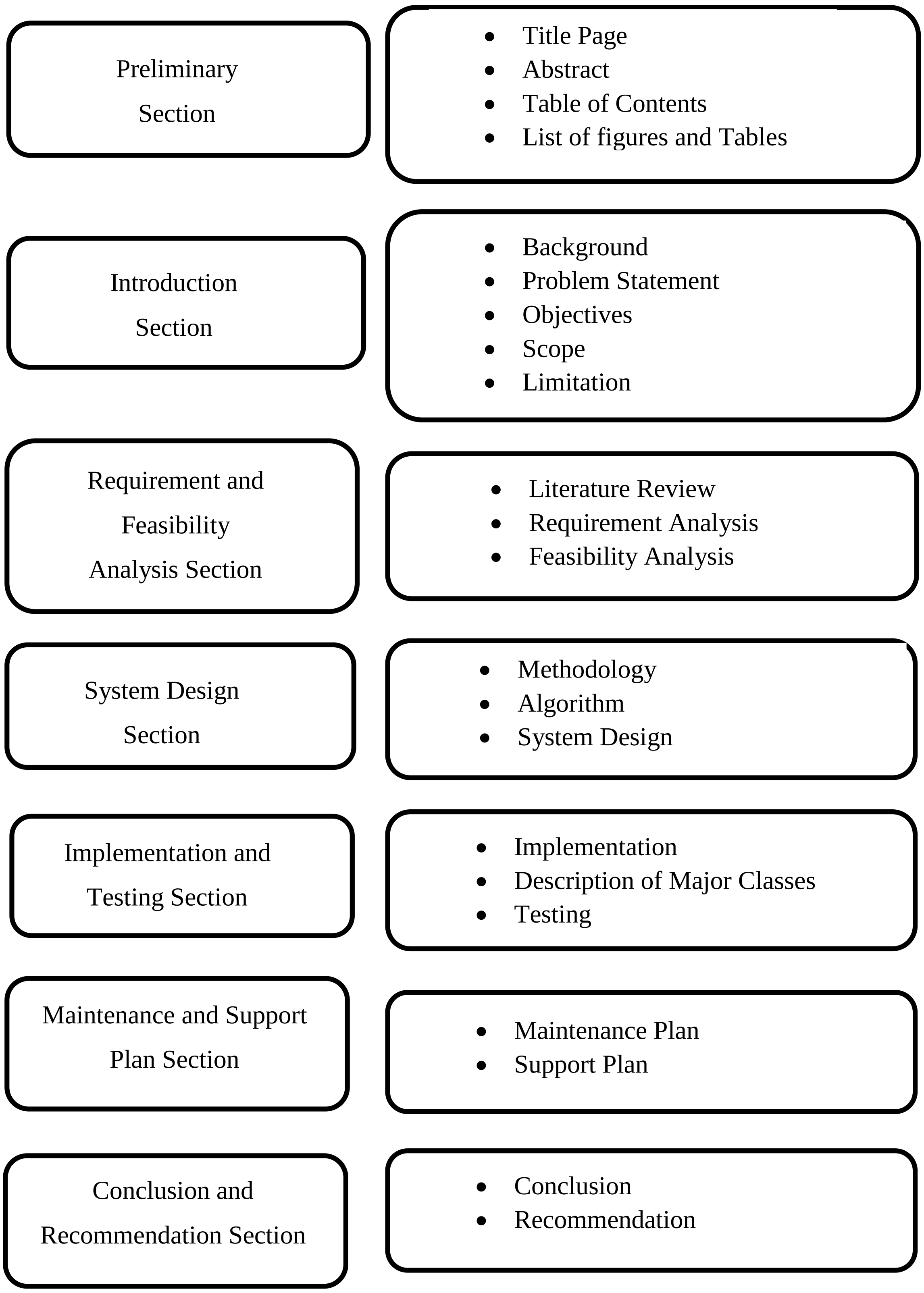


Figure 1- Outline of document

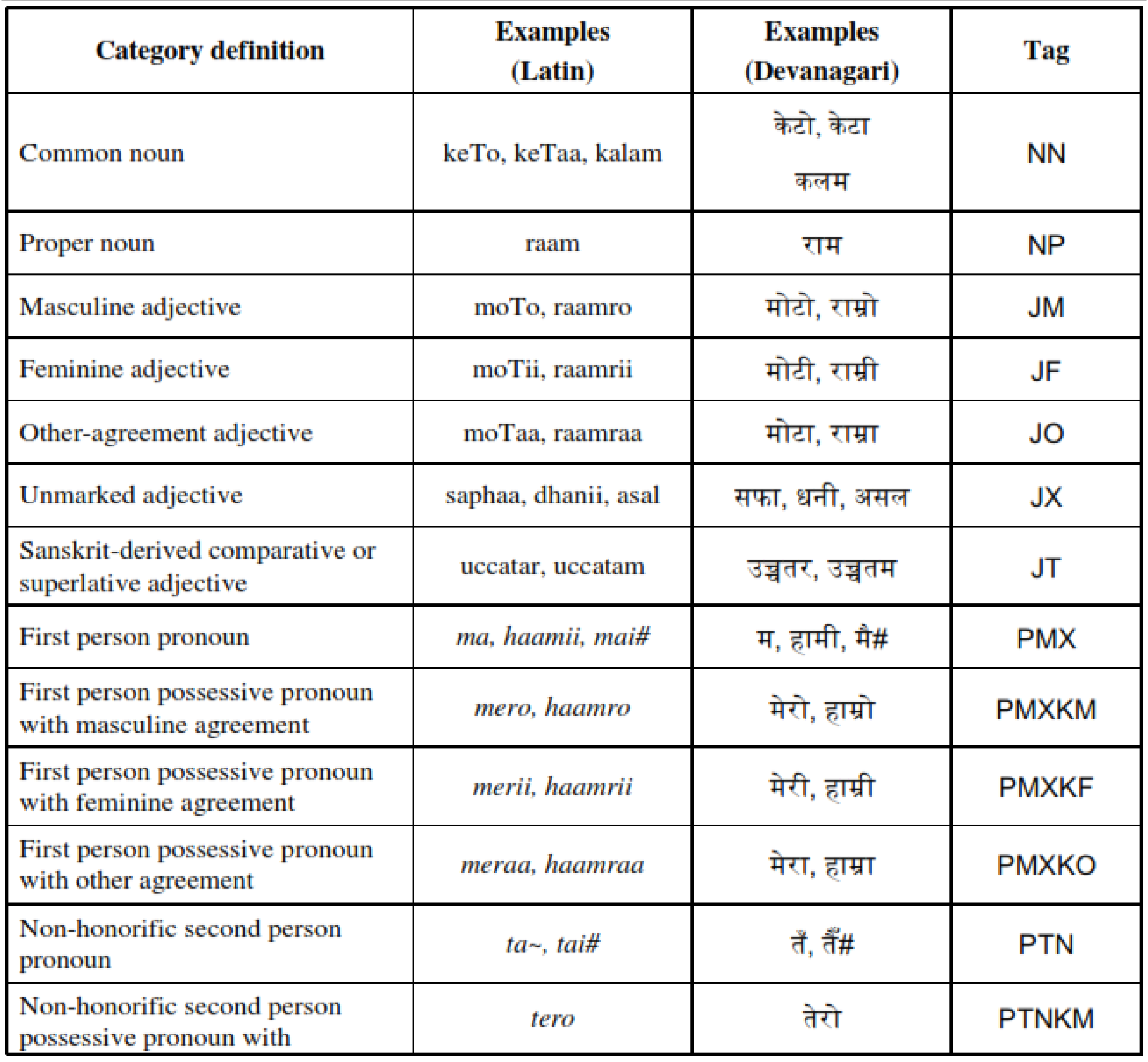
CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS

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 contain 14 million Nepali words. It consists of speech corpus, spoken corpus, core sample (CS), general collection, and parallel data.

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. In this project, this tag-set is used as the resource and referred as NNC 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. The short description of tags in [2] is given in the following figure 2. The detailed description of the 112 tag-set is at the Appendix Section.



Category definition	Examples (Latin)	Examples (Devanagari)	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, मैं#	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, हाम्रा	PMXKO
Non-honorific second person pronoun	*ta~, tai#*	तँ, तैँ#	PTN
Non-honorific second person possessive pronoun with	*tero*	तेरो	PTNKM

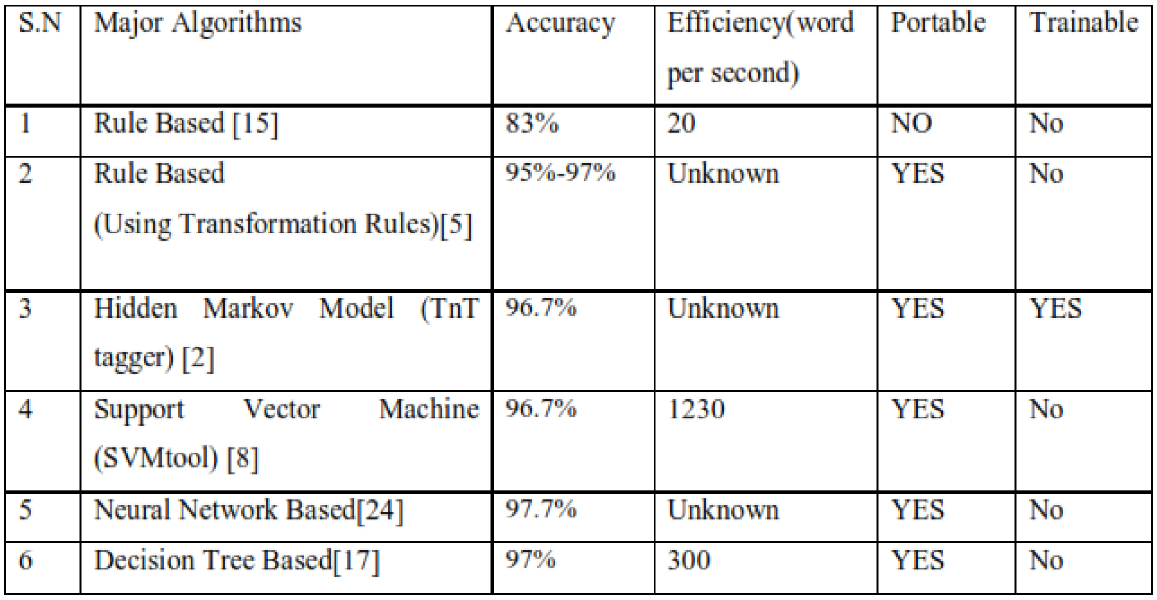
Figure 2 - NNC Tag-set

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 the automatic part of speech tagging started in the 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 need 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 Figure 3 shows the comparison of different tagger reviewed in the paper [2] based upon the four criteria – Accuracy, Efficiency, Portable, and Trainable.

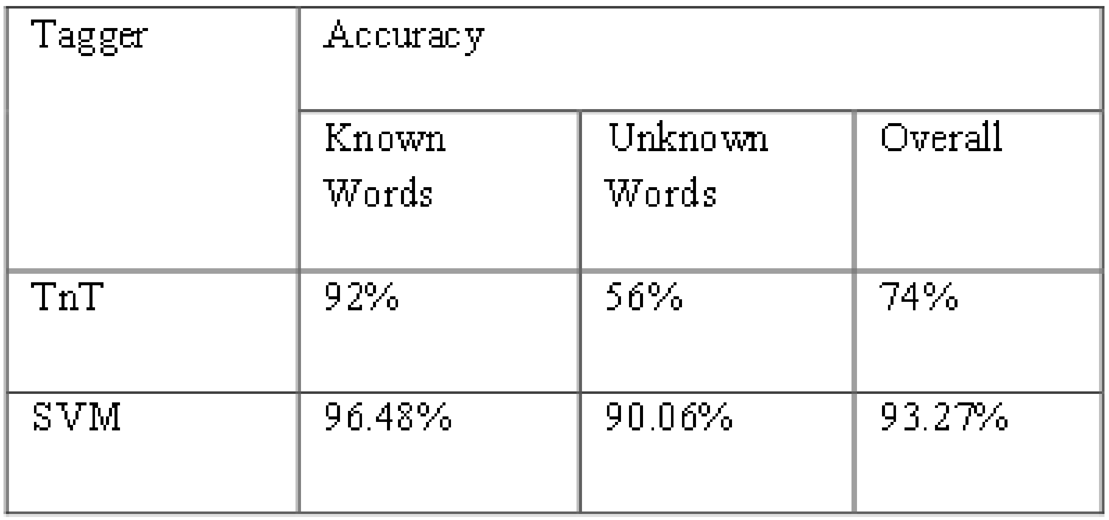


S.N	Major Algorithms	Accuracy	Efficiency(word per second)	Portable	Trainable
1	Rule Based [15]	83%	20	NO	No
2	Rule Based (Using Transformation Rules)[5]	95%-97%	Unknown	YES	No
3	Hidden Markov Model (TnT tagger) [2]	96.7%	Unknown	YES	YES
4	Support Vector Machine (SVMtool) [8]	96.7%	1230	YES	No
5	Neural Network Based[24]	97.7%	Unknown	YES	No
6	Decision Tree Based[17]	97%	300	YES	No

Figure 3 - Comparison of different taggers for English

2.1.2 Comparison of different taggers for Nepali Text

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 4 clearly shows that proposed SVM Tagger performed better than the already existing taggers for Nepali.



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

Figure 4 - Comparison of taggers for Nepali

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

Table 1 - Functional and non-functional requirements

Functional requirement	**Non-functional requirement**
Assign POS tags to each word in an input text.	Display word-tag pairs of the input text after classification.

Table 1 describes the basic functionality of Parts-of-Speech Tagger for Nepali Text which includes displaying word-tag pairs of the input sentence as output after the user enters paragraph or sentence in the textbox as an input.

2.3 Feasibility Analysis

2.3.1 Technical feasibility

Parts-of-Speech Tagger for Nepali Text is a web application that uses Python Web framework – Flask. It uses Flask, HTML, and Bootstrap CSS to design front end and Python as the back end. 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 are available and can be accessed freely, hence it was determined technically feasible.

2.3.2 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). Any user connected to the internet can access the application anytime to assign parts of speech tags to the input sentences or to further use the application for various NLP applications. Hence, Parts-of-Speech Tagger for Nepali Text was determined operationally feasible.

2.3.3 Schedule feasibility

Schedule of Parts-of-Speech Tagger for Nepali Text is as follows:

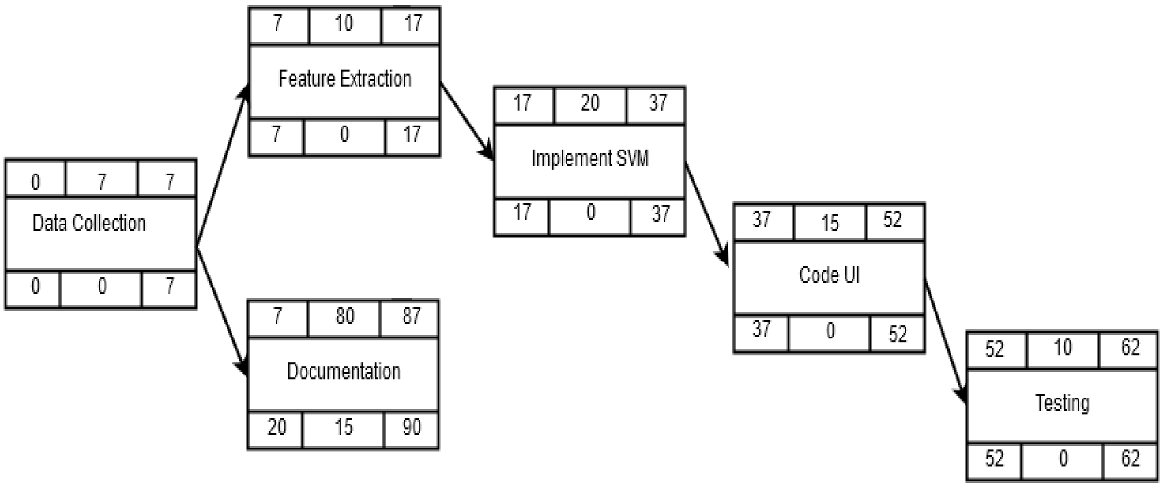


Figure 5 - Activity network diagram of Parts-of-Speech Tagger for Nepali Text

Figure 5 shows Activity Network Diagram of Parts-of-Speech Tagger for Nepali Text using SVM. The early finish time for the project is 87 days and the late finish time for the project is 90 days. The timeline above shows that the project can be completed within 15 weeks of a semester. Hence, Parts-of-Speech Tagger for Nepali text was determined to be feasible in terms of schedule.

CHAPTER 3: SYSTEM DESIGN

3.1 Methodology

Parts of Speech Tagging System has already been implemented for Bengali, Tamil and Odia languages, using SVM algorithm. The obtained result 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.

3.1.1 Data collection

Parts-of-Speech Tagger for Nepali Text uses Nepali National Monolingual Written Corpus with 13 million words for preparing the training model. The corpus consists written texts from 15 different genres with 2000 words each published between 1990 and 1992 and texts from a wide range of sources such as the internet webs, newspapers or books. The words in the corpus are lemmatized and tagged with appropriate POS tags.

3.1.2 Construction of Train & Test Set

Firstly, the XML based corpus is parsed to obtain a train set 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.

3.1.3 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 2 - Feature Set for Each Word

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

Table 2 shows the feature set that is extracted for each word in an input sentence.

3.1.4 Validation of model

Accuracy was computed for the validation of the model.

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

3.2 Algorithm

3.2.1 Support Vector Machine (SVM)

"Support Vector Machine" (SVM) is a 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 -

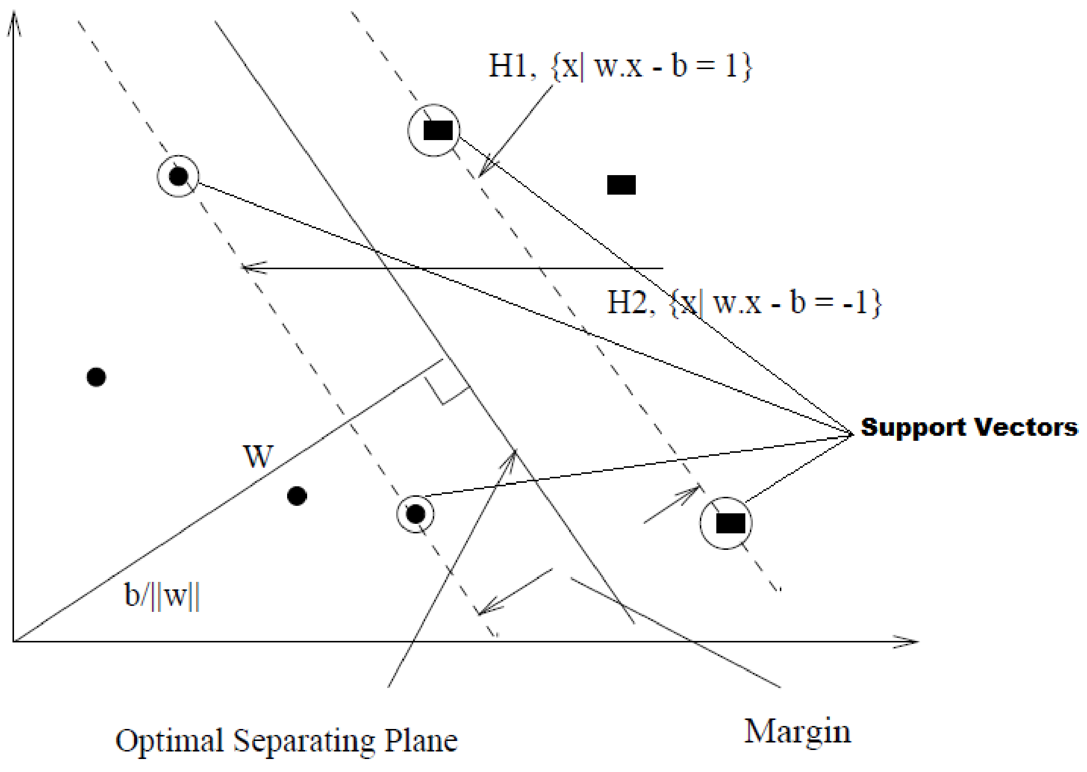


Figure 6 - Support vector machine

The equation for general hyperplane can be written as

$$w.\, x - b = 0 \qquad \text{(Equation 3.1)}$$

Where x is point vector, w is a weight vector and b is bias. The hyperplane should separate training data $\{(x_i, y_i), i=1\ldots n \text{ and } y_i \in (+1,-1)\}$ in such way that $y_i\,(w.\, x_i - b) \geq 1$. 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

$$\text{Minimize } f = \frac{|w|^2}{2} \tag{3.2}$$

$$\text{Subject to constraints } \quad y_i(w.\, x_i - b) \geq 1$$

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.

$$y_i(w.\, x_i - b) = 1 \tag{3.3}$$

$$y_i(w.\, x_i - b) = -1 \tag{3.4}$$

Equations 3.3 and 3.4 represent positive class support vectors and negative class support vectors simultaneously.

One Vs Rest 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

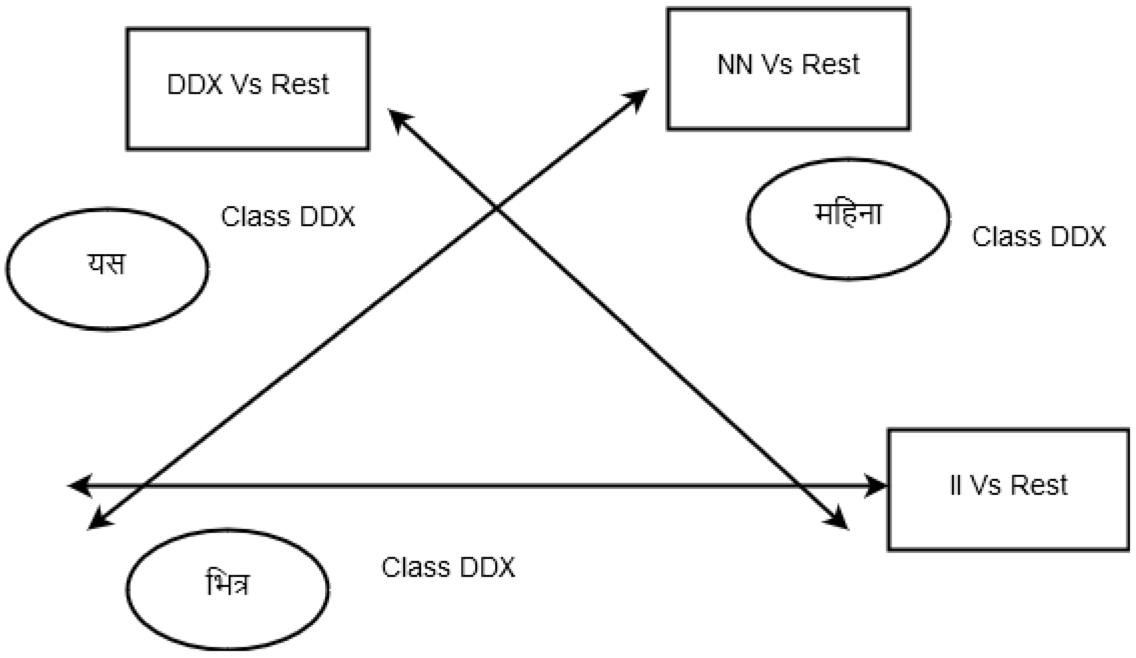


Figure 7 - One Vs. Rest Classification

3.3 System Design

3.3.1 Class diagram

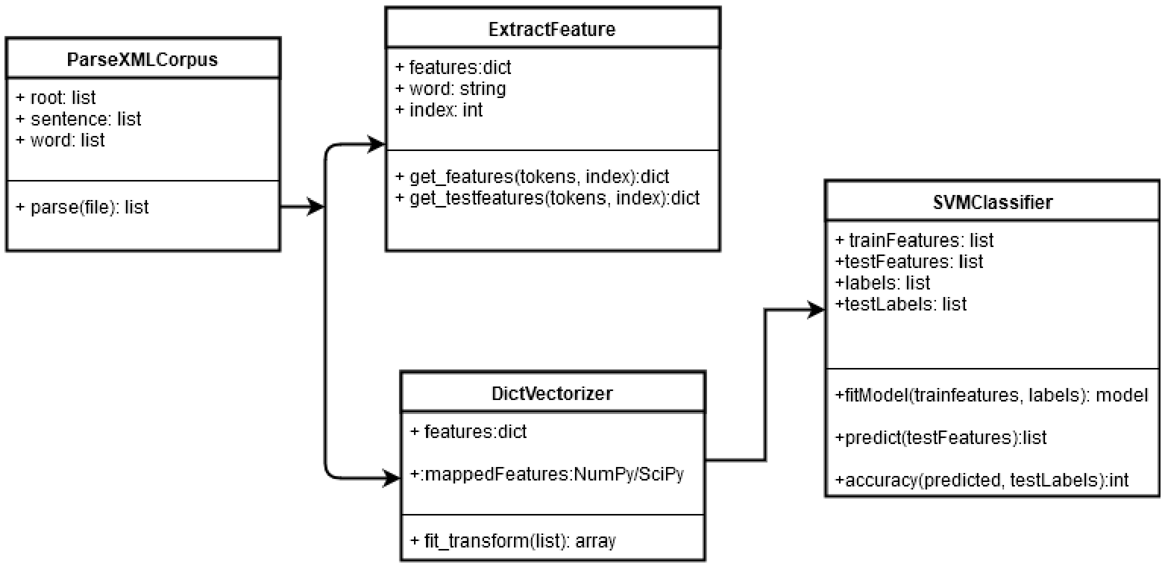


Figure 8 - Class Diagram of Parts-of-Speech Tagger for Nepali Text

Figure 8 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.

3.3.2 State diagram

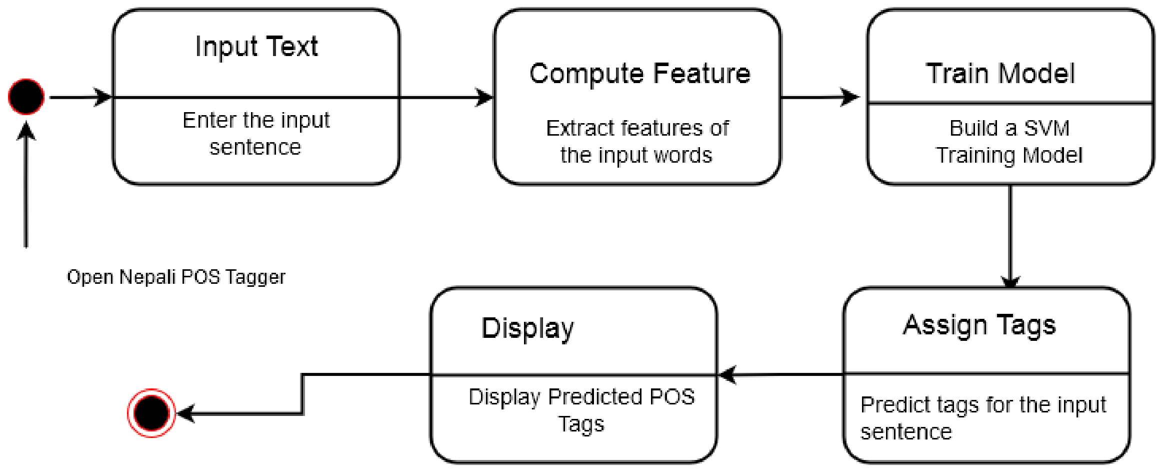

Input Text
Enter the input sentence
Compute Feature
Extract features of the input words
Train Model
Build a SVM Training Model
Open Nepali POS Tagger
Display
Display Predicted POS Tags
Assign Tags
Predict tags for the input sentence

Figure 9 - State Diagram of Parts-of-Speech Tagger for Nepali Text

Figure 9 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.

3.3.3 Sequence diagram

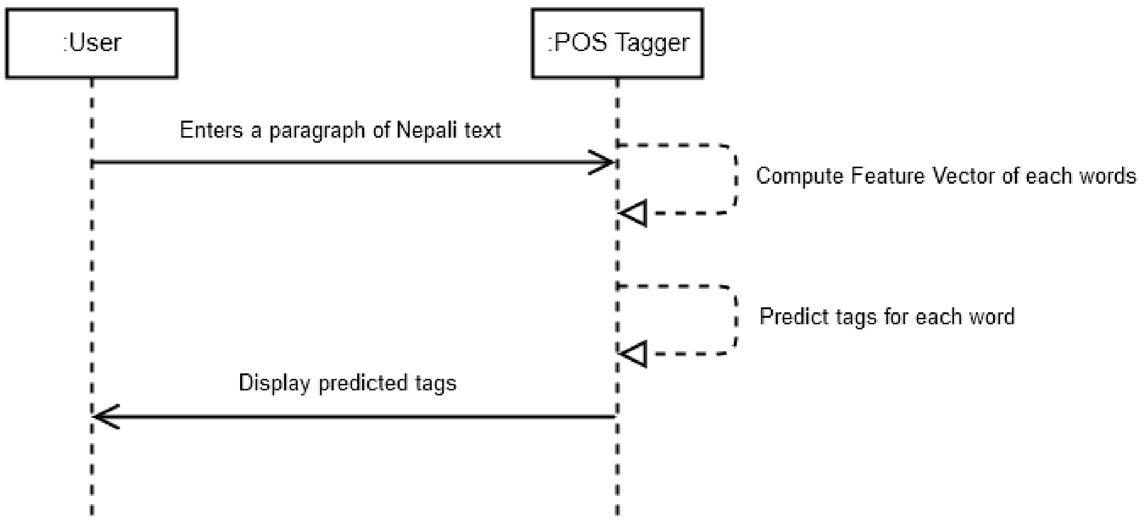

:User
:POS Tagger
Enters a paragraph of Nepali text
Compute Feature Vector of each words
Predict tags for each word
Display predicted tags

Figure 10 - Sequence Diagram of Parts-of-Speech Tagger for Nepali Text

Figure 10 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 TESTING

4.1 Implementation

User can access the application through a browser and see the interface. User can enter a paragraph of Nepali text as an input on the "Enter your text here" textbox available. The application calculates the POS tags for the input text after the user selects the "Submit" button. 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.

4.1.1 Tools used

CASE tools:

i. Draw.io

Client Side:

i. Flask
ii. HTML
iii. Bootstrap CSS

Server side:

i. Python

This section describes the technologies used to build Parts-of-Speech Tagger for Nepali Text. Parts-of-Speech Tagger for Nepali Text is a web application that uses Python to develop the back-end and Python Web framework – Flask, HTML and Bootstrap CSS to design the front-end.

All the algorithms for the application are written in Python classes. The first class used in Parts-of-Speech tagger is ParseXMLCorpus which 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.

4.2 Description of Major Methods

The major methods in the application are:

4.2.1 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 = 'H:\\FYP\\nepali pos\\nnc_updated_ah\\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.

4.2.2 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.

4.2.3 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.

4.2.3 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 -

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.

4.3 System Flow

The system flow for Parts-of-Speech Tagger for Nepali Text is shown in Figure 4.1.

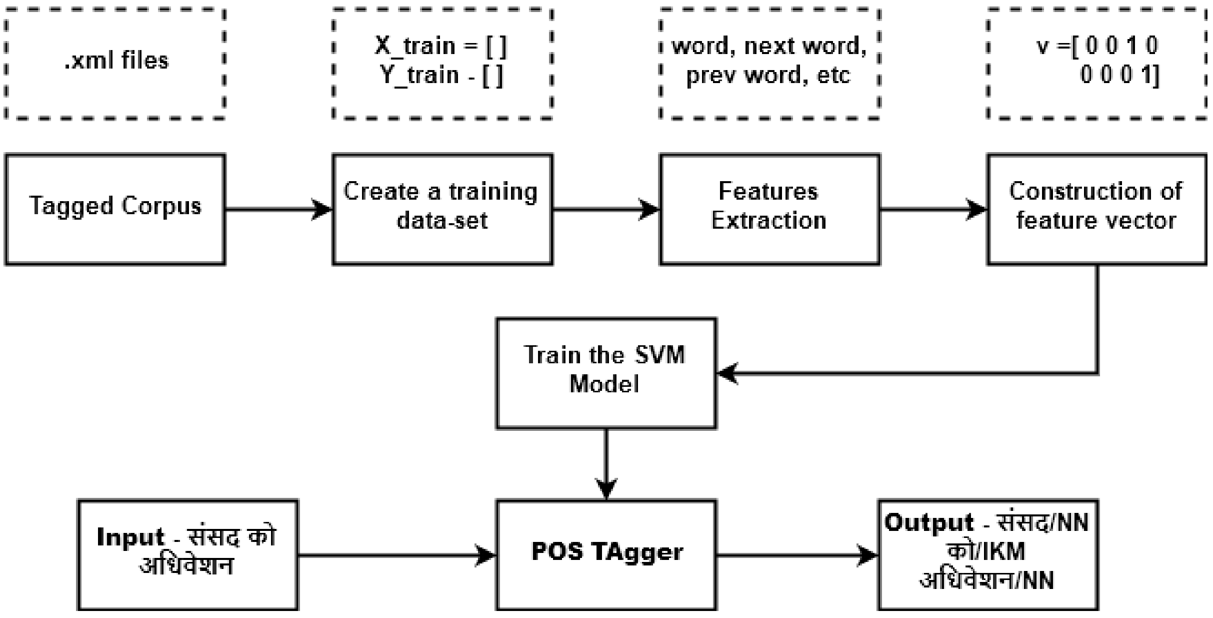


Figure 11 - System Flow of Nepali POS Tagger

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

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', 'पन्द्रह', 'MM', 'सदस्यीय', 'JX', 'मन्त्रि', 'NN', 'परिषद', 'NN', 'को', 'IKM', 'गठन', 'NN', 'गरी', 'VR', 'सक्नु', 'VI', 'भए', 'VE', 'पछि', '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': 'पन्द्रह', 'word': 'ले'}, {'prev-word': 'ले', 'next-word': 'सदस्यीय', 'word': 'पन्द्रह'}, {'prev-word': 'पन्द्रह', 'next-word': 'मन्त्रि', 'word': 'सदस्यीय'}, {'prev-word': 'सदस्यीय', 'next-word': 'परिषद', 'word': 'मन्त्रि'}, {'prev-word': 'मन्त्रि', 'next-word': 'को', 'word': 'परिषद'}, {'prev-word': 'परिषद', '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': 'भए'}, {'prev-word': 'भए', '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': 'हुन', 'word': 'केन्द्रित'}, {'prev-word':
'केन्द्रित', '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 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 ∈(+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 – ['संसद', 'को', 'अधिवेशन', 'आषाढ', 'को', 'शुरु', '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': 'अधिवेशन'}, {'next-word': 'शुरु', 'word': 'को', 'prev-word': 'आषाढ'}, {'next-word': 'मा', 'word': 'शुरु', 'prev-word': 'को'}, {'next-word': 'हुने', 'word': 'मा', 'prev-word': 'शुरु'}, {'next-word': 'राष्ट्रियसभा', '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', 'prev-word': '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-सोभियत', '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', 'prev-word': '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 $y_i\,(w.\,x_i - b) \geq 1$ 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 शुरु/NN मा/II हुने/VN राष्ट्रियसभा/NN को/IKM गठन/NN यै/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

4.4 Testing

Other than evaluating and validating the classifier's performance, different test cases were created in order to make sure that the system delivers the required output and functions properly. 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.

4.3.1 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', 'अधिवेशन', 'आषाढ', 'को', 'शुरु', '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', 'त्रिभुवन', 'विश्वविद्यालय', 'मा', 'राजिनामा', 'को', 'लहर', 'उपकुलपति', 'र', 'पदाधिकारी', 'हरु', 'को', 'नयाँ', 'नियुक्ति', 'हुने', '?', 'मुस्लिम', 'र', 'हिन्दु', 'को', 'दंगा', 'पार्टी', 'हरु', 'को', 'दिबालियापन', 'को', 'परिणाम']

Output: संसद/NN को/IKM अधिवेशन/NN आषाढ/NN को/IKM शुरु/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 विश्वविद्यालय/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.

4.3.2 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.

Input: सरकार र डा. गोविन्द केसी प्रतिनिधिबीच आइतबार वार्ता भएन । सरकारी पक्ष वार्ताको लागि नडाकेकाले छलफल नभएको केसी प्रतिनिघि डा. अभिषेकराज सिंहले बताए ।

डा. केसी १४ दिनदेखि त्रिवि शिक्षण अस्पताल हातामा आमरण अनशन गरिरहेका 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 केसी/NP प्रतिनिधिबीच/NN आइतबार/NP वार्ता/NN भएन/VI ।/YF सरकारी/JX पक्ष/NN वार्ताको/NN लागि/II नडाकेकाले/VI छलफल/NN नभएको/VI केसी/NP प्रतिनिघि/NN डा./FB अभिषेकराज/NP सिंहले/NP बताए/VVYN1 ।/YF

डा./FB केसी/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 औं/MOX आमरण/JX अनशन/NN हो/VVYN1 ।/YF

Tagger's Output: सरकार/NN र/CC डा./JX गोविन्द/NP केसी/NP प्रतिनिधिबीच/NN आइतबार/JX वार्ता/NN भएन/VVYN1 ।/YF सरकारी/NN पक्ष/NN वार्ताको/NN लागि/II नडाकेकाले/NN छलफल/NN नभएको/NN केसी/NN प्रतिनिघि/NN डा./FB अभिषेकराज/NP सिंहले/NP बताए/VVYN1 ।/YF

डा./FB केसी/JX १४/NP दिनदेखि/NN त्रिवि/NN शिक्षण/NN अस्पताल/NN हातामा/JX आमरण/JX अनशन/NN गरिरहेका/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 अनशन/NN हो/VVYN1 ।/YF

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

4.5 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: MAINTENANCE AND SUPPORT PLAN

5.1 Maintenance Plan

Parts-of-Speech Tagger for Nepali Text will implement corrective maintenance for resolving different bugs and errors that may occur when this project is made live. Perfective maintenance will be implemented for increasing efficiency of the tagger by optimizing various implementation methods. Preventive maintenance will be implemented to make sure that the tagger will not be harmed by hackers and security mechanism will be added.

5.2 Support Plan

Parts-of-Speech Tagger for Nepali Text will be presented to the respective authority of the Government for investment so that the Nepali language can be promoted and can be further used for developing other NLP related applications. For self-sustenance of the Parts-of-Speech Tagger, the community will be established which will collect new Nepali words from various sources and help to build an accurate Nepali language based corpus.

CHAPTER 6: CONCLUSION AND RECOMMENDATION

6.1 Conclusion

Parts-of-Speech Tagger for Nepali Text was successfully implemented using Support Vector Machine algorithm. The tagger built assigns the most appropriate part-of-speech tag to each word of the 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%. Hence, use of Tagger System that implements Support Vector Machine for Parts of Speech Tagging is not recommended and further improvements need to be made to be used for Nepali text.

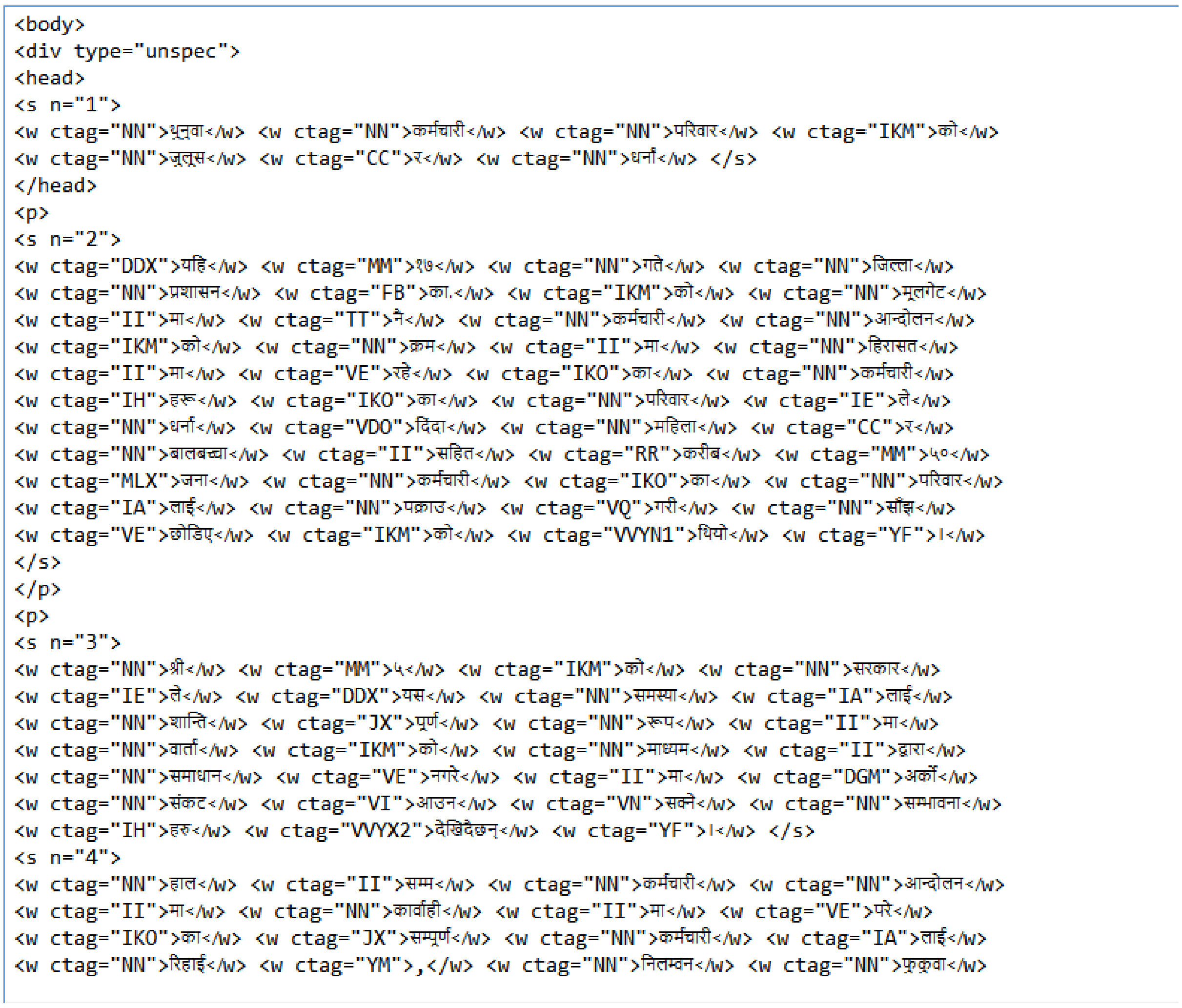
6.2 Recommendation

This project did not consider text preprocessing procedures like stop words removal, stemming, and lemmatization for parts-of-speech categorization. Hence, the use of Stemmer/Morphological Analyzer for text preprocessing and normalization is recommended to increase the accuracy of the Parts-of-Speech Tagger for Nepali text.

The other limitation of the parts-of-speech tagger built based on SVM is the speed. It is found to be slow in training than other tagger.

APPENDIX

```
<cesHeader version="2.1">
<fileDesc>
    <titleStmt>
        <h.title>NNC-CS: sample A16</h.title>
        <respStmt>
            <respType>Electronic version created by
            </respType>
            <respName>
                Nelralec / Bhasha Sanchar Project (भाषा सञ्चार)
            </respName>
        </respStmt>
        <respStmt>
            <respType>transcribed by</respType>
            <respName>सरिता दहाल तिम्सिना</respName>
        </respStmt>
    </titleStmt>
    <extent>
        <wordCount>2005</wordCount>
        <byteCount units="kb">137</byteCount>
    </extent>
[...]
    <sourceDesc>
        <biblStruct>
            <monogr>
                <h.title>जनमत अर्द्धसाप्ताहिक</h.title>
                <imprint>
                    <publisher>सुशिला चुके</publisher>
                    <pubDate>
                        २०४८-०४-२०; २०४८-०६-१४;
                        २०४८-०८-२६; २०४८-०९-२९
                    </pubDate>
                    <pubPlace>नेपालगन्ज</pubPlace>
                </imprint>
            </monogr>
        </biblStruct>
    </sourceDesc>
</fileDesc>
[...]
```



```
<body>
<div type="unspec">
<head>
<s n="1">
<w ctag="NN">धुन्वा</w> <w ctag="NN">कर्मचारी</w> <w ctag="NN">परिवार</w> <w ctag="IKM">को</w>
<w ctag="NN">जुलुस</w> <w ctag="CC">र</w> <w ctag="NN">धर्ना</w> </s>
</head>
<p>
<s n="2">
<w ctag="DDX">यहि</w> <w ctag="MM">१७</w> <w ctag="NN">गते</w> <w ctag="NN">जिल्ला</w>
<w ctag="NN">प्रशासन</w> <w ctag="FB">का.</w> <w ctag="IKM">को</w> <w ctag="NN">मूलगेट</w>
<w ctag="II">मा</w> <w ctag="TT">नै</w> <w ctag="NN">कर्मचारी</w> <w ctag="NN">आन्दोलन</w>
<w ctag="IKM">को</w> <w ctag="NN">क्रम</w> <w ctag="II">मा</w> <w ctag="NN">हिरासत</w>
<w ctag="II">मा</w> <w ctag="VE">रहे</w> <w ctag="IKO">का</w> <w ctag="NN">कर्मचारी</w>
<w ctag="IH">हरू</w> <w ctag="IKO">का</w> <w ctag="NN">परिवार</w> <w ctag="IE">ले</w>
<w ctag="NN">धर्ना</w> <w ctag="VDO">दिंदा</w> <w ctag="NN">महिला</w> <w ctag="CC">र</w>
<w ctag="NN">बालबच्चा</w> <w ctag="II">सहित</w> <w ctag="RR">करीब</w> <w ctag="MM">५०</w>
<w ctag="MLX">जना</w> <w ctag="NN">कर्मचारी</w> <w ctag="IKO">का</w> <w ctag="NN">परिवार</w>
<w ctag="IA">लाई</w> <w ctag="NN">पक्राउ</w> <w ctag="VQ">गरी</w> <w ctag="NN">साँझ</w>
<w ctag="VE">छोडिए</w> <w ctag="IKM">को</w> <w ctag="VVYN1">थियो</w> <w ctag="YF">।</w>
</s>
</p>
<p>
<s n="3">
<w ctag="NN">श्री</w> <w ctag="MM">५</w> <w ctag="IKM">को</w> <w ctag="NN">सरकार</w>
<w ctag="IE">ले</w> <w ctag="DDX">यस</w> <w ctag="NN">समस्या</w> <w ctag="IA">लाई</w>
<w ctag="NN">शान्ति</w> <w ctag="JX">पूर्ण</w> <w ctag="NN">रूप</w> <w ctag="II">मा</w>
<w ctag="NN">वार्ता</w> <w ctag="IKM">को</w> <w ctag="NN">माध्यम</w> <w ctag="II">द्वारा</w>
<w ctag="NN">समाधान</w> <w ctag="VE">नगरे</w> <w ctag="II">मा</w> <w ctag="DGM">अर्को</w>
<w ctag="NN">संकट</w> <w ctag="VI">आउन</w> <w ctag="VN">सक्ने</w> <w ctag="NN">सम्भावना</w>
<w ctag="IH">हरु</w> <w ctag="VVYX2">देखिदैछन्</w> <w ctag="YF">।</w> </s>
<s n="4">
<w ctag="NN">हाल</w> <w ctag="II">सम्म</w> <w ctag="NN">कर्मचारी</w> <w ctag="NN">आन्दोलन</w>
<w ctag="II">मा</w> <w ctag="NN">कार्वाही</w> <w ctag="II">मा</w> <w ctag="VE">परे</w>
<w ctag="IKO">का</w> <w ctag="JX">सम्पूर्ण</w> <w ctag="NN">कर्मचारी</w> <w ctag="IA">लाई</w>
<w ctag="NN">रिहाई</w> <w ctag="YM">,</w> <w ctag="NN">निलम्बन</w> <w ctag="NN">फुकुवा</w>
```

Figure – A basic format of Nepali National Monolingual Written Corpus

Category definition	Examples (Latin)	Examples (Devanagari)	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, हाम्रा	PMXKO
Non-honorific second person pronoun	*ta~, tai#*	तँ, तैँ#	PTN
Non-honorific second person possessive pronoun with	*tero*	तेरो	PTNKM

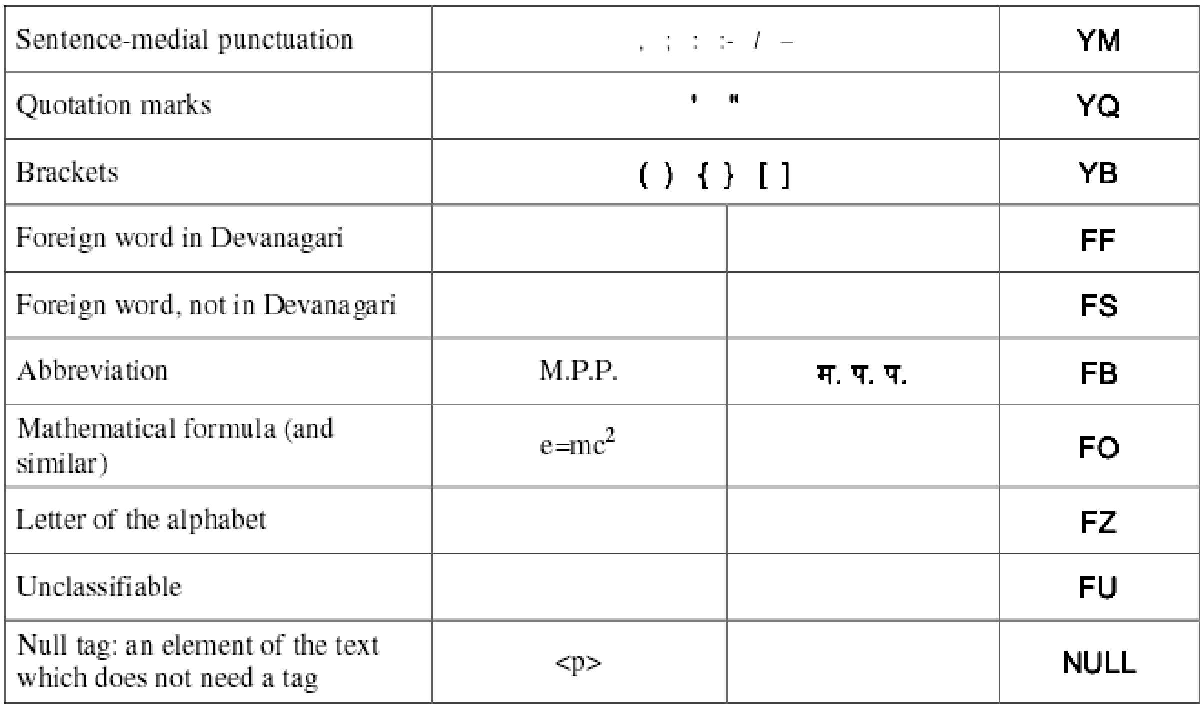
Non-honorific second person possessive pronoun with feminine agreement	*terii*	तरो	PTNKF
Non-honorific second person possessive pronoun with other agreement	*teraa*	तरा	PTNKO
Medial-honorific second person pronoun	*timii*	तिमो	PTM
Medial-honorific second person possessive pronoun with masculine agreement	*timro*	तिम्रो	PTMKM
Medial-honorific second person possessive pronoun with feminine agreement	*timrii*	तिम्रो	PTMKF
Medial-honorific second person possessive pronoun with other agreement	*timraa*	तिम्रा	PTMKO
High-honorific second person pronoun	*tapaai~, hajur*	तपाई, हजुर	PTH
High-honorific unspecified-person pronoun	*yahaa~, wahaa~*[7]	यहा, वहा	PXH
Royal-honorific unspecified-person pronoun	*sarkaar, mausuph*	सरकार, मौसफ	PXR
Reflexive pronoun	*aaphuu*	आफ़	PRF
Possessive reflexive pronoun with masculine agreement	*aaphno*	आफ्नो	PRFKM
Possessive reflexive pronoun with feminine agreement	*aaphnii*	आफ्नो	PRFKF
Possessive reflexive pronoun with other agreement	*aaphnaa*	आफ्ना	PRFKO
Masculine demonstrative determiner	*yasto, yatro*	यस्तो, यत्रो,	DDM
Feminine demonstrative determiner	*yastii, yatrii*	यस्तो, यत्रो	DDF
Other-agreement demonstrative determiner	*yastaa, yatraa*	यस्ता, यत्रा	DDO
Unmarked demonstrative determiner	*yo, yas#, yi, yin#, yinii, yati, yatti*	यो, यस#,यो, यिन#,	DDX

Masculine interrogative determiner	*kasto, katro*	कस्तो, कत्रो	DKM
Feminine interrogative determiner	*kastii, katrii*	कस्तो, कत्रो	DKF
Other-agreement interrogative determiner	*kastaa, katraa*	कस्ता, कत्रा	DKO
Unmarked interrogative determiner	*ko, kas#, ke, kun, kati*	को, कस#, क, कन, कति	DKX
Masculine relative determiner	*jasto, jatro*	जस्तो, जत्रो,	DJM
Feminine relative determiner	*jastii, jatrii*	जस्तो, जत्रो	DJF
Other-agreement relative determiner	*jastaa, jatraa*	जस्ता, जत्रा	DJO
Unmarked relative determiner	*jo, jas#, je, jati, josukai*	जो, जस#, ज, जति, जोसक	DJX
Masculine general determiner-pronoun	arko	अर्को	DGM
Feminine general determiner-pronoun	arkii	अर्को	DGF
Other-agreement general determiner-pronoun	arkaa	अका	DGO
Unmarked general determiner-pronoun	aruu	अरू	DGX
Infinitive verb	garnu, garna, garnaa, nagarnu, nagarna, nagarnaa	गन, गन, गना, नगन, नगन, नगना	VI
Masculine d-participle verb	gardo, nagardo	गर्दा, नगर्दा	VDM
Feminine d-participle verb	gardii, nagardii	गर्दा, नगर्दा	VDF
Other-agreement d-participle verb	gardaa, nagardaa	गदा, नगदा	VDO
Unmarked d-participle verb	gardai, nagardai	गद, नगद	VDX
e(ko)-participle verb	gae (as in gae saal), gare (as in garejati or gareko)	गर	VE
ne-participle verb::	garne, nagarne	गन, नगन	VN
Sequential participle-converb	garera, gariikana, garii, nagarera,	गरर, गरोकन, गरो,	VQ

Command-form verb, non-honorific	gar, jaa	गर, जा	VCN
Command-form verb, mid-honorific	gara, jaau, jaao	गर, जाऊ, जाओ	VCM
Command-form verb, high-honorific	garnos, jaanos	गर्नोस, जानोस	VCH
Subjunctive / conditional e-form verb	gare, nagare	गर, नगर	VS
i-form verb	gari	गरि	VR
First person singular verb	gare~, garthe~, garina~, chu, hu~, garnechu	गर, गथ ,गरिन, छ, हु, गनछ	VVMX1
First person plural verb	garyau~, garthyau~, garenau~, chau~, hau~, garnechau~	गर्यौं, गर्थ्यौं, गरनौ, छौ, हौ, गनछौ	VVMX2
Second person non-honorific singular verb	garis, garthis, garinas, chas, hos, garnechas	गरिस, गर्थिस, गरिनस, छस, होस, गनछस	VVTN1
Second person plural (or medial-honorific singular) verb	garyau, garthyau, garenau, chau, hau, garnechau	गर्यौं, गर्थ्यौं, गरनौ, छौ, हौ, गनछौ	VVTX2
Third person non-honorific singular verb	garyo, garthyo, garena, cha, ho, garnecha	गर्या, गर्थ्या, गरन, छ, हो, गनछ	VVYN1
Third person plural (or medial-honorific singular) verb	gare, garthe, garenan, chan, hun, garnechan	गर, गथ, गरनन, छन, हुन, गनछन	VVYX2
Feminine second person non-honorific singular verb	garlis, ches, garthis	गर्लिस, छस, गर्थिस	VVTN1F
Feminine second person non-honorific singular verb	garthyau, chyau	गर्थ्यौं, छ्यौ	VVTM1F
Feminine third person medial-honorific singular verb	garina, garii, che, garthii	गरिन, गरो, छ, गर्थो	VVYN1F
Feminine third person medial-honorific singular verb	garin, garthin, garinan, chin	गरिन, गर्थिन, गरिनन, छिन	VVYM1F
First person singular optative verb	jaau~, garu~	जाऊ, गरू	VOMX1
First person plural optative verb	jaaau~, garau~	जाऔं, गरौ	VOMX2

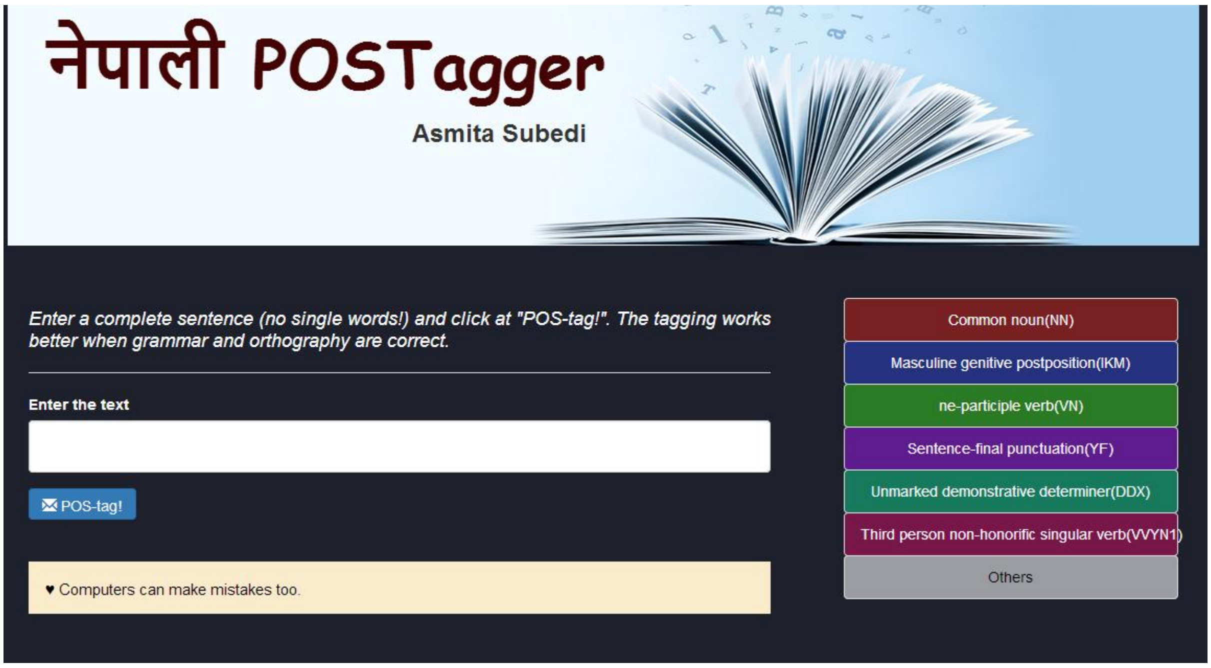
Second person non-honorific singular optative verb	gaes, gares	गएस, गरस्	VOTN1
Second person plural (or medial-honorific singular) optative verb	gae, gare	गए, गर	VOTX2
Third person non-honorific singular optative verb	jaaos, garos	जाओस, गरोस्	VOYN1
Third person plural (or medial-honorific singular) optative verb	jaauun, garuun	जाऊन, गरून्	VOYX2
Adverb	raamrarii, ekdam, chiTo	राम्ररो, एकदम, छिटो	RR
Demonstrative adverb	yataa, utaa, tyataa; ahile, ahilyai, yahii~, yasarii, aba	यता, उता, त्यता, अहिल, अहिल्य, यहों, यसरो, अब	RD
Interrogative adverb	kataa, kahaa~, kahile, kasarii	कता, कहा, कहिल	RK
Relative adverb	jataa, jahaa~, jahile, jasarii, jaba	जता, जहा, जहिल	RJ
Postposition	agaaDi, pachaaDi, baaTa, dwaaraa, maa, maathi, saath, puurvak, tira, tarpha, vasa, sanga, binaa	अगाडि, पछाडि, बाट, द्वारा, मा, माथि	II
Plural-collective postposition	*haruu*	हरू	IH
Ergative-instrumental postposition	*le*	ल	IE
Accusative-dative postposition	*laaii*	लाई	IA
Masculine genitive postposition	*ko*	को	IKM
Feminine genitive postposition	*kii*	को	IKF
Other-agreement genitive postposition	*kaa*	का	IKO
Cardinal number	*ek, eu#, yau#, dui, tin, caar, paa~c*	एक, दई, तोन, चार, पाच	MM
Masculine ordinal number	*pahilo, dosro, tesro, cautho*	पहिलो, दोस्रो, तस्रो, चौथो	MOM
Feminine ordinal number	*pahilii, dosrii, tesrii, cauthii*	पहिलो, दोस्रो, तस्रो,	MOF

Other-agreement ordinal number	*pahilaa, dosraa, tesraa, cauthaa*	पहिला, दोस्रा, तस्रा, चौथा	MOO
Unmarked ordinal number	*paa~cau~*	पाचौ	MOX
Masculine numeral classifier	*#To* [as in euTo]	#टो	MLM
Feminine numeral classifier	*(#)waTii, #Tii, #oTii, #auTii*	(#)वटो, #टो, #ओटो	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	भन, भनो, भनर	CSA
Subordinating conjunction appearing **before** the clause it subordinates	ki, yadi, yaddyapi, kinaki	कि, यदि, यद्यपि, किनकि	CSB
Particle	nai, caahi~, pani, hai, ra, re, kyaare, ho, khai	न, चाहिँ, पनि, ह, र, र, क्यार, हो, ख	TT
Question marker	*ke*	क	QQ
Interjection	oho, aahaa, hare	ओहो, आहा, हर	UU
Possessive reflexive pronoun without agreement	*merai*	मर	PMXKX
Non-honorific second person possessive pronoun without agreement	*terai*	तर	PTNKX
Medial-honorific second person possessive pronoun without agreement	*timrai*	तिम्र	PTMKX
Possessive reflexive pronoun without agreement	*aaphnai*	आफ्न	PRFKX
Unmarked genitive postposition	*kai*	क	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. प. प.	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

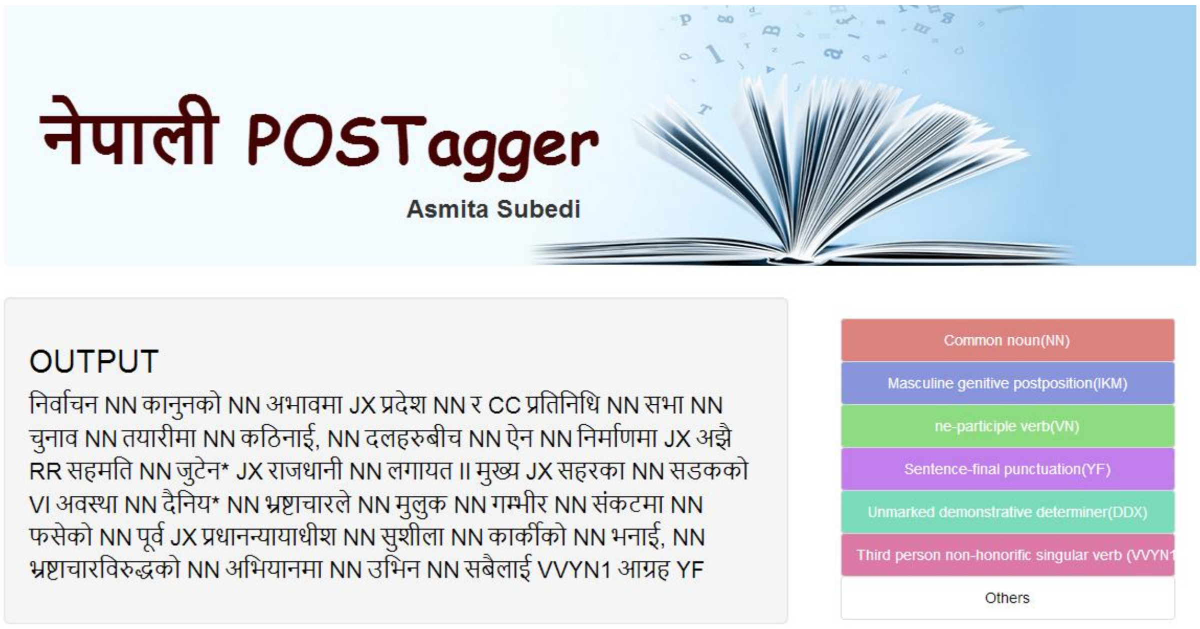
Figure: 112 Tag-set of Nepali National Monolingual Written Corpus


नेपाली POSTagger
Asmita Subedi
Enter the text
POS-tag!
Common noun(NN)
Masculine genitive postposition(IKM)
ne-participle verb(VN)
Sentence-final punctuation(YF)
Unmarked demonstrative determiner(DDX)
Third person non-honorific singular verb(VVYN1)
Others

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

♥ Computers can make mistakes too.

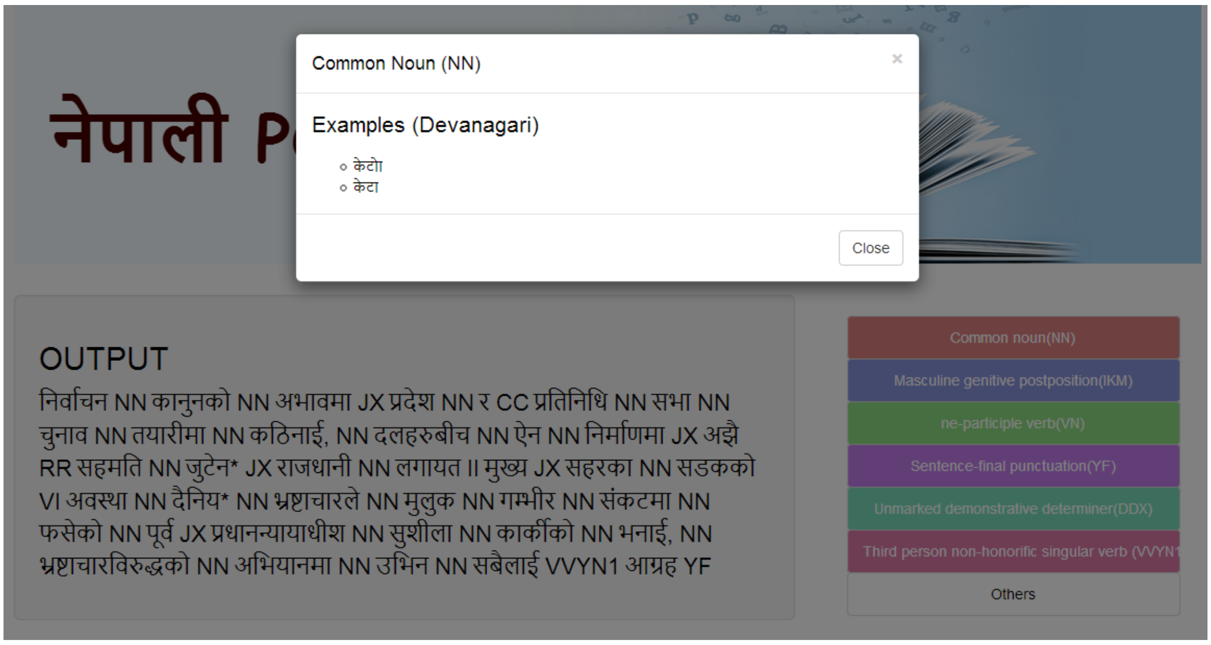
Figure – Landing Page of Nepali POS Tagger


नेपाली POSTagger
Asmita Subedi
Common noun(NN)
Masculine genitive postposition(IKM)
ne-participle verb(VN)
Sentence-final punctuation(YF)
Unmarked demonstrative determiner(DDX)
Third person non-honorific singular verb (VVYN1
Others

OUTPUT

निर्वाचन NN कानुनको NN अभावमा JX प्रदेश NN र CC प्रतिनिधि NN सभा NN चुनाव NN तयारीमा NN कठिनाई, NN दलहरुबीच NN ऐन NN निर्माणमा JX अझै RR सहमति NN जुटेन* JX राजधानी NN लगायत II मुख्य JX सहरका NN सडकको VI अवस्था NN दैनिय* NN भ्रष्टाचारले NN मुलुक NN गम्भीर NN संकटमा NN फसेको NN पूर्व JX प्रधानन्यायाधीश NN सुशीला NN कार्कीको NN भनाई, NN भ्रष्टाचारविरुद्धको NN अभियानमा NN उभिन NN सबैलाई VVYN1 आग्रह YF

Figure – Output Page of Nepali POS Tagger


नेपाली P
Common Noun (NN)
×
Close
Common noun(NN)
Masculine genitive postposition(IKM)
ne-participle verb(VN)
Sentence-final punctuation(YF)
Unmarked demonstrative determiner(DDX)
Third person non-honorific singular verb (VVYN1
Others

Examples (Devanagari)

- केटो
- केटा

OUTPUT

निर्वाचन NN कानुनको NN अभावमा JX प्रदेश NN र CC प्रतिनिधि NN सभा NN चुनाव NN तयारीमा NN कठिनाई, NN दलहरुबीच NN ऐन NN निर्माणमा JX अझै RR सहमति NN जुटेन* JX राजधानी NN लगायत II मुख्य JX सहरका NN सडकको VI अवस्था NN दैनिय* NN भ्रष्टाचारले NN मुलुक NN गम्भीर NN संकटमा NN फसेको NN पूर्व JX प्रधानन्यायाधीश NN सुशीला NN कार्कीको NN भनाई, NN भ्रष्टाचारविरुद्धको NN अभियानमा NN उभिन NN सबैलाई VVYN1 आग्रह YF

Figure – Modal that gives the explanation of the POS tags

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] A. 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.