Tribhuvan University

Institute of Science and Technology

Literature Review Research

On

PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE

Submitted to

Central Department of Computer Science and Information Technology

Tribhuvan University, Kirtipur

Kathmandu, Nepal

In the partial fulfilment of the requirement for Master's Degree in Computer Science and Information Technology (M. Sc. CSIT), Third Semester

Submitted by

Raju Shrestha

Roll No. 48/2079

Jan, 2025

Tribhuvan University

Institute of Science and Technology

SUPERVISOR'S RECOMMENDATION

This is to certify that Mr. Raju Shrestha has submitted the literature review report on the topic **"PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE"** for the partial fulfilment of Master's of Science in Computer Science and Information Technology, Third semester. I hereby, declare that this literature review report has been approved.

…….

Supervisor

Asst. Prof. Bikash Balami

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

This is to certify that the literature review report prepared by Mr. Raju Shrestha entitled **"PARTS OF SPEECH TAGGER FOR NEPALI TEXT USING SUPPORT VECTOR MACHINE"** in partial fulfilment of the requirements for the degree of Master's of Science 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.

Evaluation Committee

……..……..	……...……
Asst. Prof. Sarbin Sayami	Asst. Prof. Bikash Balami
(H.O.D)	(Supervisor)
Central Department of Computer Science	Central Department of Computer Science
and Information Technology	and Information Technology

……

(Internal)

ACKNOWLEDGEMENT

The success and final outcome of this report required a lot of guidance and assistance from many people and I am very fortunate to have got this all along the completion. I am very glad to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed supervisor **Asst. Prof. Bikash Balami,** Central Department of Computer Science and Information Technology for his valuable supervision, guidance, encouragement, and support for completing this paper.

I am also thankful to **Asst. Prof. Sarbin Sayami**, HOD of Central Department of Computer Science and Information Technology for his constant support throughout the period. Furthermore, with immense pleasure, I submit by deepest gratitude to the Central Department of Computer Science and Information Technology, Tribhuvan University, and all the faculty members of CDCSIT for providing the platform to explore the knowledge of interest. At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly.

Raju Shrestha (48/2079)

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 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).

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN: Artificial Neural Network

Bi-LSTM: Bi-directional Long Short Term Memory

CNN: Convolution Neural Network

CSS: Cascading Style Sheet

GRNN: General Regression Neural Network

GRU: Gated Recurrent Unit

Bi-GRU: Bidirectional Gated Recurrent Unit

LSTM: Long Short Term Memory

Bi-LSTM: Bidirectional Long Short Term Memory

HMM: Hidden Markov Model

HTML: Hyper Text Markup Language

MNN: Multilayer Neural Network

NLP: Natural Language Processing

NLG: Natural Language Generation

NLU: Natural Language Understanding

NLTK: Natural Language Toolkit

POS: Parts of Speech

RBF: Radial Basis Function

RNN: Recurrent Neural Network

Bi-RNN: Bidirectional Recurrent Neural Network

SVC: Support Vector Classifier

SVM: Support Vector Machine

TnT: Trigrams 'n' Tags

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,[1] and has a diverse application domain including speech recognition, speech synthesis, grammar checker, machine translation, information retrieval and extraction etc. In this work, the tagging refers to the process of assigning part of speech (POS) tag to a word. The computer programs designed to automatically assign the POS tag to a word in natural language text, are called taggers. The outline of process is shown as in figure.

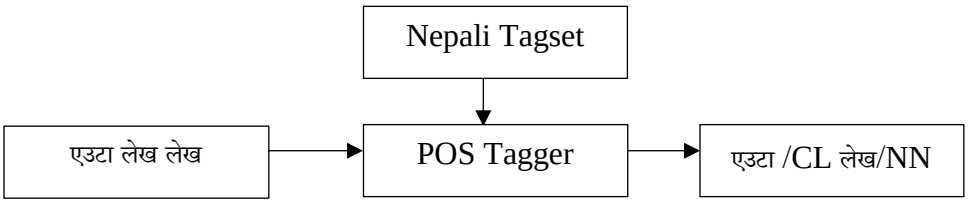

Figure 1: POS Tagger Example

The work on automatic part of speech tagging started in early 1960s. Klein and Simmon's rule based POS tagger can be considered as the first automatic tagging system. Since rule based approaches needs more sophisticated rules to capture the language knowledge, later on the data driven approaches were developed and recently machine learning and deep learning approaches are being developed [1].

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.

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 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.

This application 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 on it.

Parts of Speech Tagger for Nepali Text is quite challenging task due to complexity of the language. Limited research have been carried out in this domain. This work proposed a supervised Machine Learning based framework for assigning the POS tag for Nepali text into different category.

1.3 Objective

The main objectives of this literature review report is to assign the parts of speech (POS) tags automatically to each words for an input text or sentence based on its definition and context.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Natural Language Processing (NLP) is a dynamic field of artificial intelligence (AI) that focuses on enabling computers to understand, interpret, and generate human language. . Human language consists of words and sentences in natural form and NLP is used to extract the information. NLP processes or analyzes written or spoken language. Natural Language Processing is the analysis of linguistic data in the form of text as documents or sentences using computational methods. It helps to represent the unstructured text into structured text data using computational linguistics. NLP research try to collect information on how human understand and use different natural languages that helps machine understand and manipulate natural languages.

Natural language processing encompasses a wide range of tasks, 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.

NLP algorithms employ techniques from machine learning, linguistics, and computer science to bridge the gap between human communication and computational systems. With its rapid advancements and real-world applications across industries like healthcare, finance, and customer service, NLP plays a vital role in re-shaping how humans interact with machines and access information, making it a vital area of study and innovation in today's digital age.

Natural language processing is the field of machine learning which is concerned with training machines to understand and generate results like humans is done by using natural languages. It has 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. NLU is used in the automated reasoning, machine translation, question answering, news gathering, text categorization, voice activation, archiving, and large scale content analysis. Natural Language

Generation (NLG) is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation. NLG is complementary to natural-language understanding. The system needs to make decisions about how to put a representation into words.

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.

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.

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.

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.

The Natural Language Processing in Nepali language has not a long history as NLP in Nepali text become started from early 2000s AD. But the works are in high number now a days. The Nepali NLP works are broadly done on three categories namely Rule based, Machine Learning based and Deep Learning based.

2.2 Literature Review

2.2.1 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 [3] 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
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 ,तँ#	PTN
Non-honorific second person possessive pronoun with masculine agreement	*tero*	तेरो	PTNKM

2.2.2 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. Klein and Simmon's rule based POS tagger can be considered as the first automatic tagging system. Since rule based approaches needs more sophisticated rules to capture the language knowledge, later on the data driven approaches were developed and recently machine learning other deep learning approaches are being developed [2].

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	83%	20	No	No
2	Rule Based (Using Transformation Rules)	95% -97%	Unknown	Yes	No
3	Hidden Markov Model (TnT tagger)	96.7%	Unknown	Yes	Yes
4	Support Vector Machine (SVMtool)	96.7	1230	Yes	No
5	Neural Network Based	97.7%	Unknown	Yes	No
6	Decision Tree Based	97%	300	Yes	No

2.2.3 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 [4].

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	99.97	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. The network is trained using 5373 patterns (Nepali

words) and corresponding tags. 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 [5].

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

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 the Viterbi technique performs better than HMM [6].

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 POS Tagged Nepali Corpus dataset with 43 tags was used for modelling the POS tagger. A sequence to sequence approach was followed to model the problem with various deep learning algorithms such as RNN, LSTM, GRU, and their bi-directional variants. Bi-directional versions of RNN, LSTM and GRU achieved the maximum performance scores with binary cross entropy as the loss function. The accuracy of the system also increases with the increase in the size of word embedding vector. When the loss function was changed from categorical cross entropy to binary cross entropy, the accuracy score, precision score, recall score and F1 score improved significantly. This is due to the fact that binary cross entropy generally works well with multiclass-multilabel classification and in this problem, target values are a set of vectors instead of a single value. Bi-directional versions of RNN, LSTM, and GRU performs better than their unidirectional counterpart [7].

Table 6: Results of Nepali POS Taggers with Deep Learning Algorithms [7]

Hidden size	**Embedding size**	**Algorithm**	**Loss function**	**Accuracy**	**Precision**	**Recall**	**F1-score**
32	100	RNN	Categorical cross entropy	90.41	0.9	0.9	0.9
32	300	RNN	Categorical cross entropy	91.68	0.91	0.92	0.91
32	100	LSTM	Categorical cross entropy	90.41	0.91	0.91	0.91
32	300	LSTM	Categorical cross entropy	91.74	0.91	0.92	0.91
32	100	GRU	Categorical cross entropy	90.63	0.9	0.91	0.9
32	300	GRU	Categorical cross entropy	91.66	0.91	0.92	0.91

32	100	Bi-RNN	Categorical cross entropy	92.11	0.92	0.92	0.92
32	300	Bi-RNN	Categorical cross entropy	92.71	0.93	0.93	0.92
32	100	Bi-LSTM	Categorical cross entropy	91.98	0.91	0.92	0.92
32	300	Bi-LSTM	Categorical cross entropy	92.66	0.93	0.92	0.92
32	100	Bi-GRU	Categorical cross entropy	92.19	0.92	0.92	0.92
32	300	Bi-GRU	Categorical cross entropy	92.71	0.93	0.93	0.92
32	100	RNN	Binary cross entropy	99.76	0.89	0.9	0.89
32	300	RNN	Binary cross entropy	99.83	0.91	0.92	0.91
32	100	LSTM	Binary cross entropy	99.75	0.88	0.89	0.88
32	300	LSTM	Binary cross entropy	99.83	0.91	0.92	0.91
32	100	GRU	Binary cross entropy	99.77	0.89	0.9	0.89

32	300	GRU	Binary cross entropy	99.83	0.91	0.92	0.91
32	100	Bi-RNN	Binary cross entropy	99.83	0.91	0.92	0.91
32	300	Bi-RNN	Binary cross entropy	99.84	0.93	0.93	0.92
32	100	Bi-LSTM	Binary cross entropy	99.82	0.91	0.92	0.91
32	300	Bi-LSTM	Binary cross entropy	99.85	0.93	0.93	0.92
32	100	Bi-GRU	Binary cross entropy	99.83	0.91	0.92	0.91
32	300	Bi-GRU	Binary cross entropy	99.84	0.93	0.93	0.92

POS tagging of Nepali Text was carried out using simple RNN, LSTM, GRU and Bi-directional LSTM in a Nepali tagged corpus of tag size 40. This paper focuses on implementing and comparing different deep learning based POS tagger for Nepali such as Simple Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-directional Long Short Term Memory (Bi-LSTM). The accuracy obtained for simple RNN, LSTM, GRU and Bi-directional LSTM was 96.84%, 96.48%, 96.86% and 97.27% respectively. Therefore, Bi-directional LSTM outperformed all other three variants of RNN [8].

The proposed SVM based tagger has been compared with the existing TnT tagger for the Nepali language [1]. 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 clearly shows that proposed SVM Tagger performed better than the TnT tagger for Nepali text.

Table 7: 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 [9] for the Bengali language which is also morphologically rich and shows the outstanding performance.

Hence, SVM based POS tagger 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.

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.

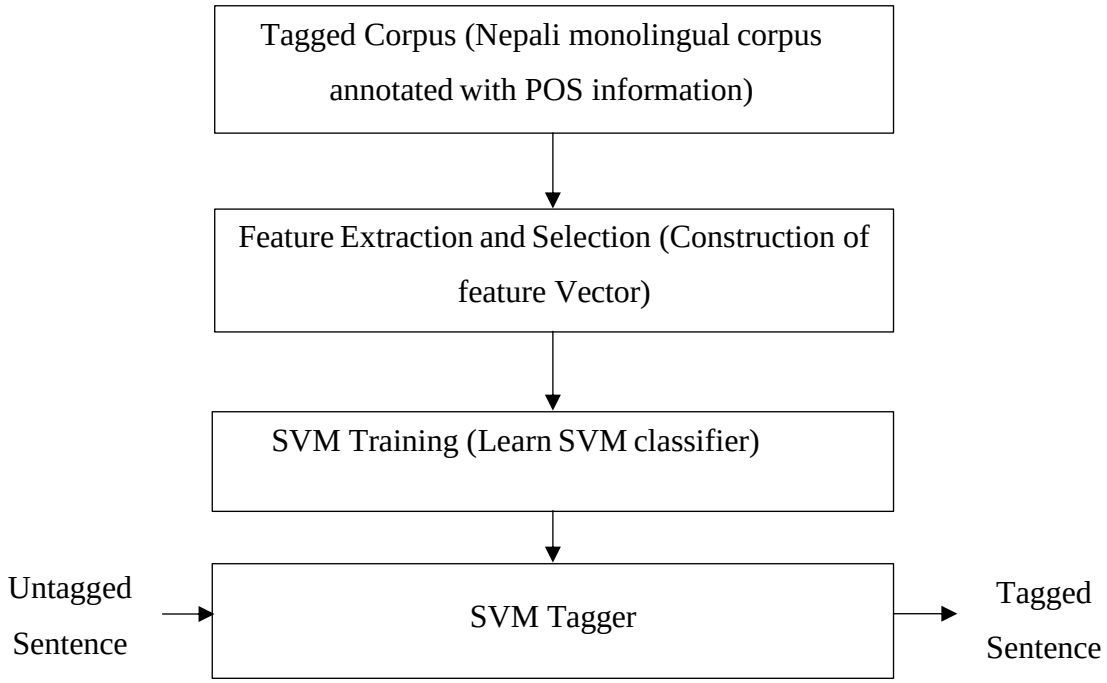

Figure 2: Flow Diagram of Parts of Speech Tagger for Nepali Text

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 algorithm 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.

3.1 Data Preparation

3.1.1 Data Collection

The Nepali Monolingual Written Corpus contains 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 tag, the sentences in the paragraph are enclosed with tag, and the words in the sentences are enclosed with 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="1">
  <w ctag="DDX">ऊनी</w>
  <w ctag="NN">गलैंचा</w>
  <w ctag="NN">निकासी</w>
  <w ctag="II">मा</w>
  <w ctag="MM">५२.</w>
  <w ctag="MM">५</w>
  <w ctag="NN">प्रतिशत</w>
  <w ctag="IE">ले</w>
  <w ctag="NN">वृद्धि</w>
</s>
```

Figure 3: Nepali POS Dataset Sample

3.1.2 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.

3.1.3 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 8: 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:

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

3.2 Algorithms Studied and Implemented

3.2.1 Support Vector Machine Algorithm

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:

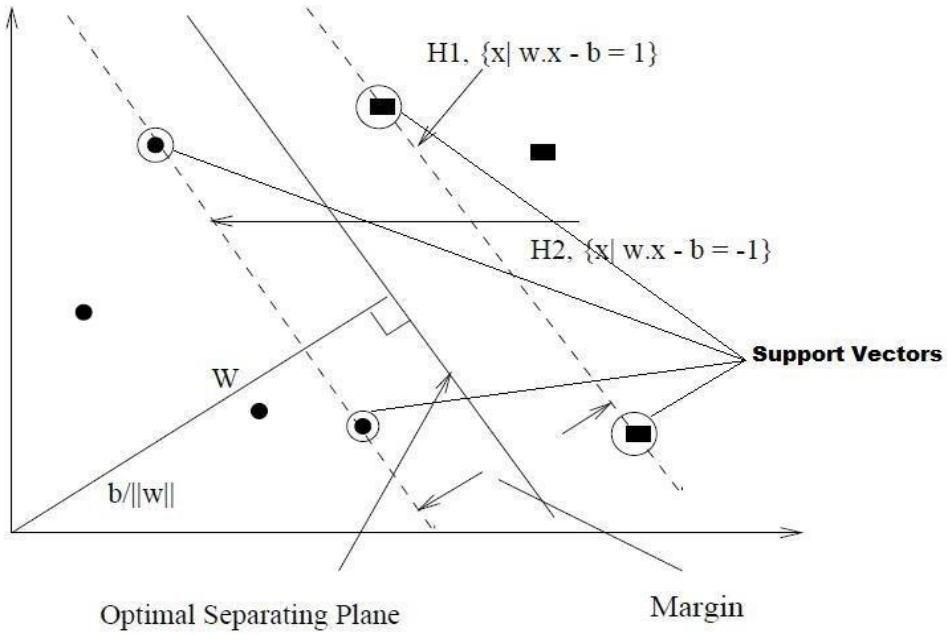

Figure 4: Support Vector Machine

The equation for general hyperplane can be written as:

$$w.x - b = 0$$

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:

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

$$\text{Subject to constraints } i\ (w.\ xi - b) \geq 1$$

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.

$$yi\ (w.\ xi - b) = 1$$

$$yi\ (w.\ xi - b) = -1$$

These two equation 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

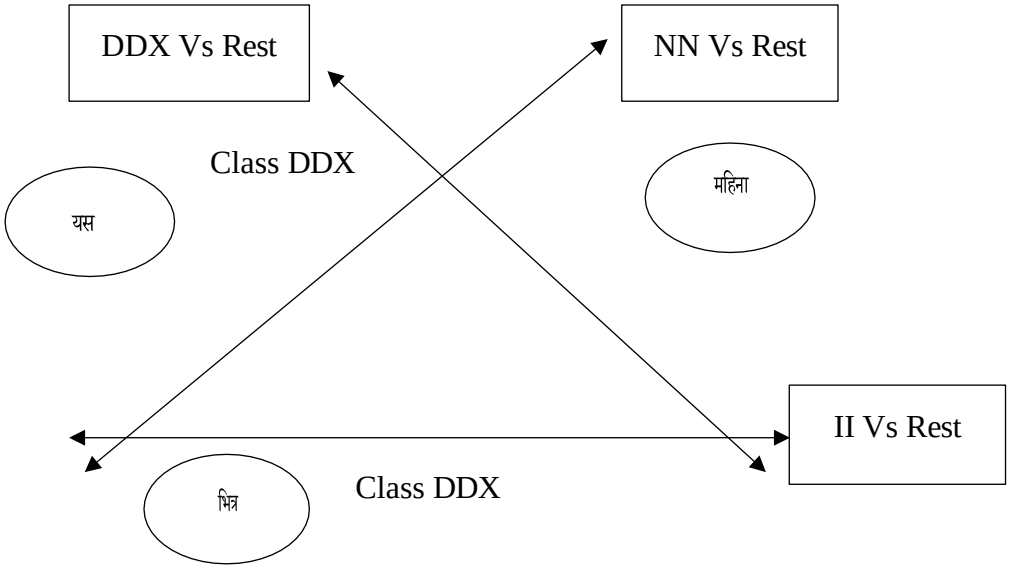

Figure 5: One Vs Rest Classification

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Tools and Technologies Used

This section describes the tools and technologies used for parts of speech tagger for Nepali text using SVM algorithm. The model is implemented in Python programming language, flask (Python web framework) is used for developing web application and VS Code is used as an IDE. Tools like HTML, CSS, Bootstrap and JavaScript are also used for developing front-end application. Python is a high-level, general purpose programming language. It has a great support for libraries for implementing machine learning algorithm. The libraries used for this work are as follows:

- **Pandas:** Pandas is an open source Python library that provides fast, powerful, flexible, high performance, easy to use data structures and data analysis tools. It also allows various data manipulation operations.
- **NumPy:** NumPy is a general purpose array processing Python library used for array variable and numerical computing.
- **Matplotlib:** Matplotlib is a plotting library for 2D graphics in python programming language.
- **Scikit learn:** Scikit-learn is the simple and efficient tools for predictive data analysis. It provides the tools for classification, regression, clustering, preprocessing and model selection.
- **Natural Language Toolkit (NLTK):** NLTK is a toolkit used for NLP in Python. It can be used to performing different NLP task such as tokenization, stop words removal, stemming, lemmatization, parse tree generation, parts of speech (POS) tagging etc.

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. 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 (Python framework).

4.3 Description of Major Function

The major function 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 = '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 9: 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 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.

4.4.1 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/NP सीमा/NN मा/II बढी/NN सबल/NN सुरक्षा/YF राष्ट्रिय/JX जनगणना/NN को/IKM महत्व/NN भारत/NP र/CC नेपाल/NP को/IKM वार्ता/NN विवाद/NN को/IKM जाल/NN मा/II राष्ट्र/NN बैंक/NN मा/II कर्मचारी/NN निलम्बित/YF त्रिभुवन/DDX 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/NN 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 86.15% was ob tained on the test data for the lemmatized test data.

4.4.2 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 वार्ताको लागि नडलकेको छलफल नभएको के सी प्रतिनिधि डा. अधिषेकराज सिंहले बताए ।

डा. केसी १४ दिनदेखि त्रिवि शिक्षण अस्पताल हातामा आमरण अनशन गरिरहेको 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 वार्ताको/NN लागि/II नडाकेको/VI छलफल/NN नभएको/VI के सी/NP प्रतिनिधि/NN डा./FB अधिषेकराज/NP सिंह/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 गरिहेका/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: सरकार/NN र/CC डा./JX गोविन्द/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 अधिषेकराज/NP सिंह/NP बताए/VVYN1 ।/YF डा./FB के सी/JX १४/NP दिनदेखि/NN त्रिवि/NN शिक्षण/NN अस्पताल/NN हातामा/JX आमरन/JX अनशण/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 /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.

4.5 Model Evaluation

After training, the model's performance was evaluated on the test dataset. Metrics such as accuracy, precision, recall and f1-score were calculated from the confusion matrix to assess its effectiveness in POS tagging.

```
# Calculate accuracy
from sklearn.feature_extraction import DictVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline

clf = Pipeline([
            ('vectorizer', DictVectorizer(sparse=False)),
            ('classifier', LinearSVC())
        ])
clf.fit(X[:50000],y[:50000])

y_predict = (clf.predict(X_test))
print(test_sen)
print(y_predict)
accuracy = clf.score(X_test, y_test)
print("Accuracy:", accuracy * 100, '%')
```

Figure 6: Model Prediction

```python
# Calculate macro/micro precision, recall & f1-score
from sklearn.metrics import precision_score, recall_score, f1_score

precision_macro = precision_score(Y_test, y_predicted, average='macro')
precision_micro = precision_score(Y_test, y_predicted, average='micro')

recall_macro = recall_score(Y_test, y_predicted, average='macro')
recall_micro = recall_score(Y_test, y_predicted, average='micro')

f1_macro = f1_score(Y_test, y_predicted, average='macro')
f1_micro = f1_score(Y_test, y_predicted, average='micro')

# Print the results
print("Macro Precision Score :",  precision_macro*100, '%')
print("Micro Precision Score :", precision_micro*100, '%')

print("Macro Recall Score :", recall_macro*100, '%')
print("Micro Recall Score :", recall_micro*100, '%')

print("Macro F1 Score :", f1_macro*100, '%')
print("Micro F1 Score :", f1_micro*100, '%')
```

Figure 7: Model Evaluation

CHAPTER 5: RESULT AND FINDINGS

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:

- 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.

5.1 Evaluation Metrics

Confusion Metrics: The confusion metrics provides a detailed breakdown of the model's predictions, showing the number of true positive, true negative, false positive, and false negatives.

The confusion matrix results provide a detailed view of the classification performance of the model. The confusion matrix is a table that summarizes how successful the classification model performs on labeled dataset. It provides the correct and incorrect classification for each class. One axis of the confusion matrix is predicted, and the other axis is the actual label.

Predicted	CC	DDX	IH	II	IKM	JX	NN	NP	VN	YF
Actual										
CC	3	0	0	0	0	0	0	0	0	0
DDX	0	0	0	0	0	0	1	0	0	0
IH	0	0	2	0	0	0	0	0	0	0
II	0	0	0	6	0	0	0	0	0	0
IKM	0	0	0	0	11	0	0	0	0	0
JX	0	0	0	0	0	2	2	0	0	1
NN	0	0	0	0	0	0	26	0	0	3
NP	0	1	0	0	0	0	0	3	0	0
VN	0	0	0	0	0	0	0	0	2	1
YF	0	0	0	0	0	0	0	0	0	1

Figure 8: Confusion Matrix

Accuracy: Since we are working on a classification problem of assigning the different POS tag to the Nepali text or word. Accuracy will be a good evaluator for the models. This is most commonly use metric to evaluate how well the model predicts. Accuracy is the ratio of the number of correct predictions made against the total number of prediction made.

$$Accuracy = \frac{(True\ Positive + True\ Negative)}{Total\ Number\ of\ Samples}$$

Precision: Precision calculates the ratio of correctly predicted positive instances to the total number of instances predicted as positive by the model. It assesses how well the model performs when it predicts a positive class.

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)}$$

Recall: Recall measures the model's ability to correctly identify positive instances (true positives) out of all the actual positive instances. It is also known as Sensitivity or True Positive Rate.

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)}$$

F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, especially when you want to find a single metric to evaluate your model's performance.

$$F1 - Score = \frac{2 * Precision * Recall)}{(Precision + Recall)}$$

5.2 Model Performance

The performance of Parts of Speech Tagger for Nepali Text Using Support Vector Machine algorithm is measured. These metrics indicate that the model achieved a quit low accuracy in correctly assigning the different POS tags. The precision score reflects the model's ability to correctly assign the POS tags. The recall score indicates its ability to classify all the actual positive tag present in the dataset. F1 score indicates the model's ability to achieve both accurate positive tag and comprehensive capturing of actual positive tag.

- Accuracy: 86.15 %
- Precision (Macro): 80.63 %
- Precision (Micro): 86.15 %
- Recall (Macro): 77.13 %
- Recall (Micro): 86.15 %
- F1-Score (Macro): 74.10 %
- F1-Score (Micro): 86.15 %

CHAPTER 6: CONCLUSION AND LIMITATIONS

6.1 Conclusion

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 86.15% 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. In future, other deep learning based techniques can be used to improve the performance of tagger.

6.2 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. 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 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. K. Bal and P. Shrestha, "Reports on Computational Grammar," *Madan Puraskar Pustakalaya, Patan Dhoka, Lalitpur, Kathmandu.*

[2] T. B. Shahi, T. N. Dhamala and B. Balami, "Support Vector Machines based Part of Speech Tagging for Nepali Text," *International Journal of Computer Applications,* vol. 70–No.24, no. May 2013.

[3] A. Hardie, R. R. Lohani, B. N. Regmi and Y. P. Yadava, "A morphosyntactic categorisation scheme for the automated analysis of Nepali," in *Annual Review of South Asian Languages and Linguistics*, De Gruyter Mouton, 2009.

[4] A. Yajnik, "ANN Based POS Tagging For Nepali Text," *nternational Journal on Natural Language Computing (IJNLC),* vol. 7, no. June 2018.

[5] A. Yajnik, "GENERAL REGRESSION NEURAL Newtork Based POS Tagging For Nepali Text," *Department of mathematics, Sikkim Manipal University, Sikkim , India,,* no. July 2018.

[6] A. Yajnik, "Part of Speech Tagging Using Statistical Approach for Nepali Text," *International Journal of Cognitive and Language Sciences,* vol. 11, 2017.

[7] P. Greeshma, P. Jyothsna, K. Shahina and B. Premjith, "A Deep Learning Approach for Part-of-Speech Tagging in Nepali Language," in *International Conference on Advances in Computing, Communications and Informatics, IEEE,* 2018.

[8] S. Sayami, T. B. Shahi and S. Shakya, "Nepali POS Tagging using Deep Learning Approaches," *EasyChair,* no. 1 December 2019.

[9] B. Prasain, L. Khatiwada, B. K. Bal and P. Shrestha, "Part-of-speech Tagset for Nepali," *Madan Puraskar Pustakalaya,* 2008.

APPENDIX

Figure 9: User Interface of Nepali POS Tagger

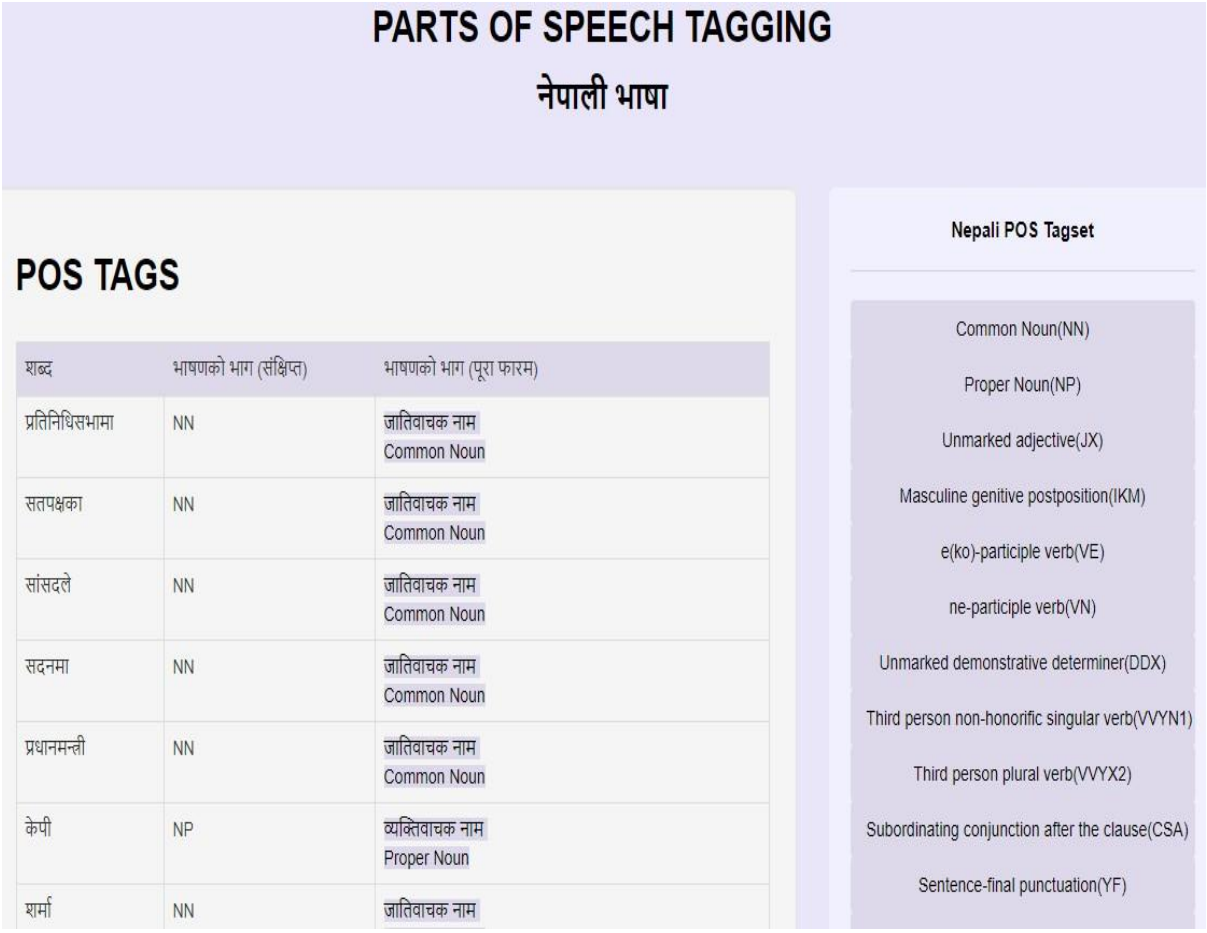

Figure 10: Output Result Displayed by the System

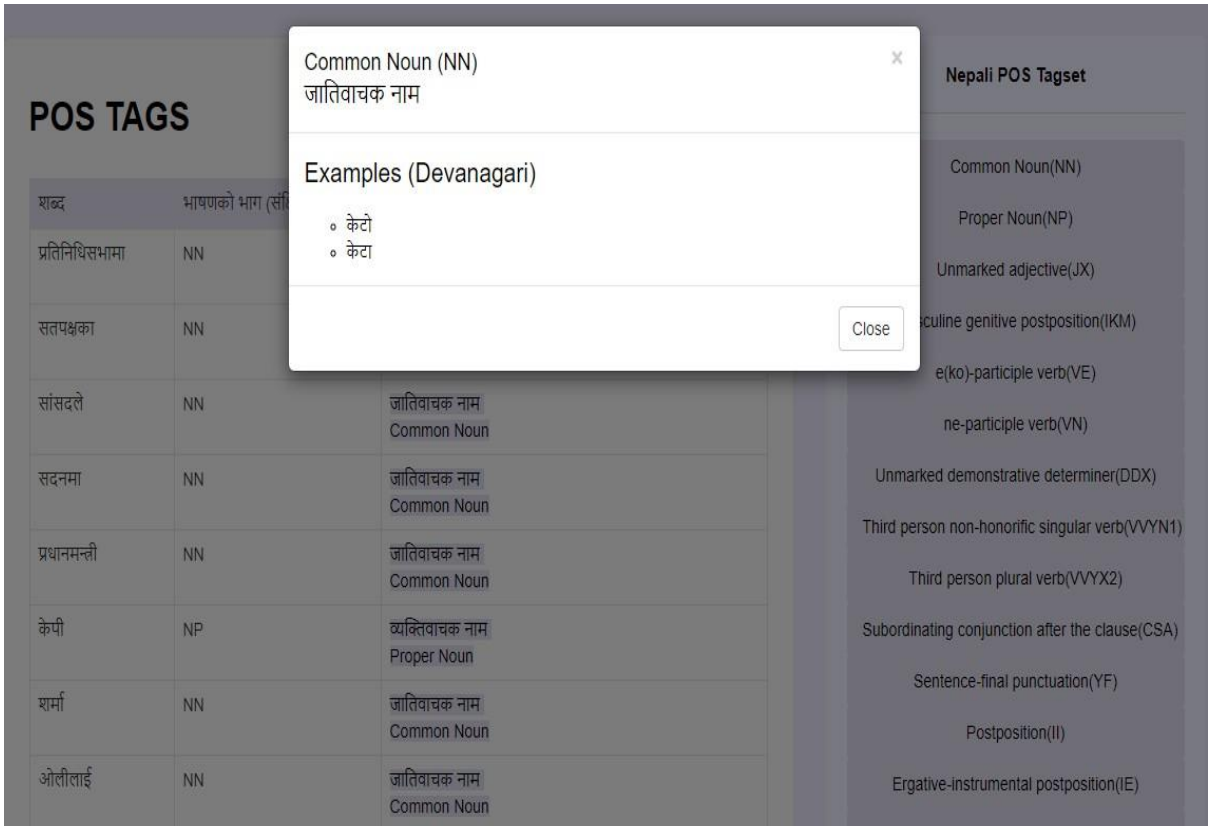

Figure 11: Tag-set Example Displayed by the System

Table 10: 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 ,हाम्रा	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*	तपाईं,हजुर	PTH
High-honorific unspecified-person pronoun	*yahaa~, wahaa~*	यहाँ ,वहाँ	PXH
Royal-honorific unspecified-person 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	अको	DGM
Feminine general determine-pronoun	arkii	अकी	DGF
Other agreement general determiner-pronoun	arkaa	अर्का	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	गदो ,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	गदै ,नगर्दै	VDX
e(ko)-participle verb	gae (as in gae saal), gare (as in garejati or gareko)	गरे	VE
ne-participle verb::	garne, nagarne	गने ,नगर्ने ,	VN
Sequential participle-converb	garera, gariikana, garii, nagarera, nagariikana, nagarii	गरेर ,गरीकन गरी,नगरेर, नगरीकन ,नगरी	VQ
Command-form verb, non-honorific	gar, jaa	जार ,जा	VCN
Command-form verb, mid-honorific	gara, jaau, jaao	गर ,जाऊ ,जाओ	VCM
Command-form verb, high-honorific	garnos, jaanos	गर्नोस ,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 ,गर्नेछु	VVMX1

First person plural verb	garyau~, garthyau~, garenau~, chau~, hau~, garnechau~	गर्‌यौ ,गथ्यौं,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 medial-honorific singular) verb	garyau, garthyau, garenau, chau, hau, garnechau	गर्‌यौ ,गथ्यौं ,गरेनौ ,छौ ,हौ ,गर्नेछौ	VVTX2
Third person non-honorific singular verb	garyo, garthyo, garena, cha, ho, garnecha	गयौ ,गथ्यौं ,गरेन ,छ ,हो ,गर्नेछ	VVYN1
Third person plural (or medial-honorific singular) verb	gare, garthe, garenan, chan, hun, garnechan	गरे ,गर्‌थे ,गरेनन् ,छन् ,हुन् ,गर्नेछन्	VVYX2
Feminine second person non-honorific singular verb	garlis, ches, garthis	गर्लिस ,छेस् ,गर्‌थिस्	VVTN1F
Feminine second person non-honorific singular verb	garthyau, chyau	गथ्यौं ,छयौ	VVTM1F
Feminine third person medial-honorific singular verb	garina, garii, che, garthii	गरिन, गरी, छे, गर्थी	VVYN1F
Feminine third person medial-honorific singular verb	garin, garthin, garinan, chin	गरिन्, गर्थिन्, गरिनन्, छिन	VVYM1F
First person singular optative verb	jaau~, garu~	जाऊँ, गरुँ	VOMX1
First person plural optative verb	jaaau~, garau~	जाऔ, गरौं	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	जाओस्, गरोस्	VOYX1
Third person plural (or medial honorific singular) optative verb	jaauun, garuun	जाऊन्, गरुन्	VOYX2
Adverb	raamrarii, ekdam, chiTo	राम्ररी, एकदम, छिटो	RR
Demonstrative adverb	yataa, utaa, tyataa; ahile, ahilyai, yahii~, yasarii, aba	यता, उता, त्यता,अहिले, अहिल्यै, यहीं, यसरी, अब	RD
Interrogative adverb	kataa, kahaa~, kahile, kasarii	कता, कहा, कहिले, कसरी	RK
Relative adverb	jataa, jahaa~, jahile, jasarii, jaba	जता, जहा, जहिले, जसरी, जब	RJ
Postposition	agaaDi, pachaaDi, baaTa, dwaaraa, maa,maathi, saath,	अगाडि, पछाडि, बाट, द्वारा, मा, माथि	II
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 ,दोस्रो ,तेस्री ,चौथी	MOF
Other-agreement ordinal number	*pahilaa, dosraa, tesraa, cauthaa*	पहिला ,दोस्रा ,तेस्रा ,चौथा	MOO
Unmarked ordinal number	*paa~cau~*	पाँचौ	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*	*(#)(*वट ,#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,यदी ,यद्यपि ,किनकि	CSB

Particle	nai, caahi~, pani, hai, ra, re, kyaare, ho, khai	नै,चाहिं,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*	मेरै	PMXKX
Non-honorific second person possessive pronoun without agreement	*terai*	तेरै	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