CHAPTER 1

INTRODUCTION

1.1 Introduction

Natural language processing (NLP) is meant for any attempt that helps the machine to understand the natural language (spoken or written) and to generate the natural language. Thus NLP consist 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. In this pipeline of NLP, the lexical analysis is early step and the accuracy of this module is significant to other following modules. Lexical analysis mainly consists of recognition of lexicon of the language and this is closely related with syntactic analysis such as parsing. At this level, one important task is assigning part of speech tag to an individual word that conveys the grammatical meaning of words and help to reduce the much work of parsing early in the process.

Tagging in its broad sense is the process of assigning any label to a linguistic unit. The linguistic unit may be word, phrase, sentence etc. in this dissertation 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 word in natural language text are called taggers. The outline of process is shown as in figure 1.1.

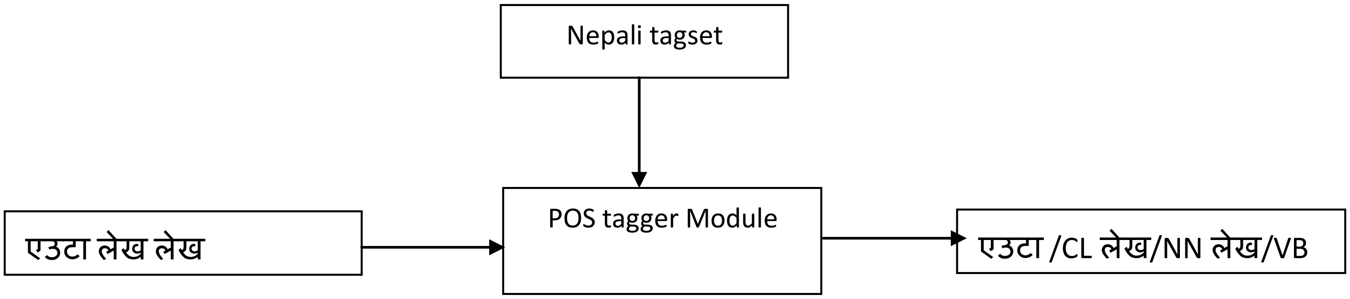

Figure 1.1: POS Tagging Example

The tags or tagset are defined in advance by the language expert and are usually covers the all POS category of language [20].

1.1.1 The Ambiguity Problem

The natural language is ambiguous in nature. Ambiguity appears in each level of language processing pipeline and it represent on of the most difficult problem in language understanding domain[17].Part of speech ambiguity in lexical analysis, semantic ambiguity in polysemic word, syntactic ambiguity in parsing, word choice selection ambiguity in machine translation etc are some representative examples of ambiguity problem in NLP. In order to resolve ambiguity, it is necessary to disambiguate two or more syntactically, semantically or structurally distinct linguistic unit depending upon the surrounding context. For example, the sentences taken from [14]

He will race the car.
When will the race end? ………………………….Example (1.1)

To understand these sentences, first their POS tag ambiguity must be resolved. The words 'race' may take Verb (VB) or NN (Noun) as POS tag. Here, the ambiguity can completely disambiguate using their contexts as:

He will race/VERB the car.
When will the race/NOUN end?

1.1.2 POS Tagging Problem

Part-of-speech tagging is the process of assigning to each word in an input text a proper morpho syntactic tag or part of speech tag in its context of appearance [5]. In most case, the ambiguous word can be disambiguate completely using the adequate context as in above example 1.1.The word 'race' would be disambiguate as noun because its previous word "the" is determiner, ambiguity is resolved by simply looking previous tag. But it is not sufficient to disambiguate the word by such simple context and may require much more language knowledge. The most challenging problem in POS tagging disambiguation is to determine the proper context and adequate features.

There are a wide variety of applications of part-of- speech tagging software and tagged text. These include information retrieval, word processor spelling and grammar-checking, speech processing, handwriting recognition, machine translation, production of corpus-based dictionaries and grammars, and applications in the teaching of foreign languages and knowledge of grammar [16]. The performance of these tasks depends upon the performance of tagger. So they need the fast, accurate, portable and trainable tagger.

1.1.3 General Approach for POS Tagging

The general representation of POS tagging process is shown in figure 1.2. The two main components are language model learning and disambiguation algorithms or tagging algorithms. These two components often related and found to be embedded in single tagger description.

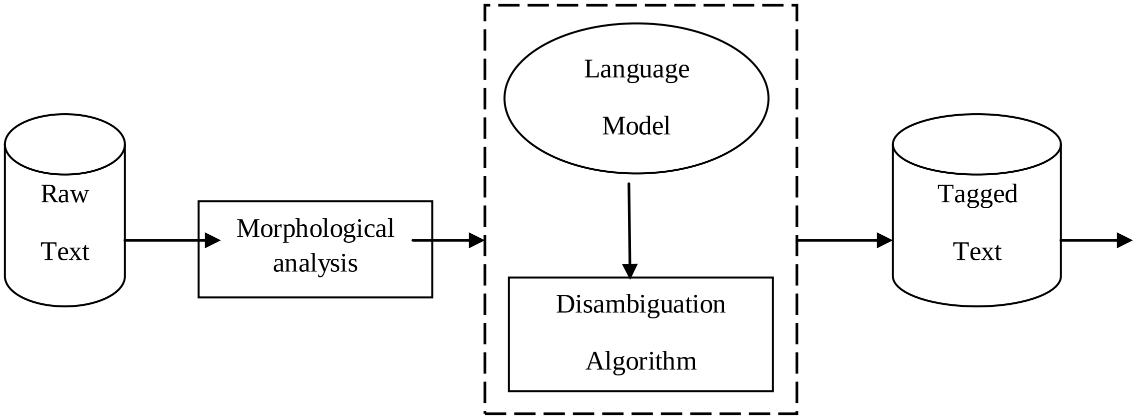

Figure 1.2: Phase of POS Tagging [10]

1.1.3.1 Tokenization and Analysis

In tokenization, the tokenizer, also called “Lexer” or “Scanner” which takes the raw source text and breaks it into the reserved words, constants, identifier and symbols that are defined in the language. These tokens so found are collected and assigns the possible tags to each tokens which usually involves the ambiguity. This assignment may be by simple lexicon look up or morphological analysis. The lexicon is usually extracted form pre tagged corpora. This lexicon is referred as dictionary in this dissertation work. This phase prepare the list of word with their possible POS tags.

1.1.3.2 Disambiguation

The so described previous stages such as tokenization and analysis are straightforward but the most challenging task in part-of-speech tagging is disambiguation. The disambiguation technique makes use of some kind of knowledge to reduce the possible POS tags for a lexical unit to few or a unique tag. This knowledge comes from different sources and in different representation which is known as language model.

Since the POS tagging disambiguation select the one tag for a word among the possible tags of that word in lexicon or given by morphological analysis, it can be recast as classification problem [19]. Classification is the supervised machine learning which attempt to classify the raw input into a predefined class. In case of POS tagging problem, the POS tags are predefined and they act as class and the word to which it is to be disambiguated is here to classify using some language knowledge learned during the training of classifier. In this dissertation work, the support vector machine [24] with one versus rest classification will be implemented for Nepali POS tagging.

1.2 Motivation

Nepali is morphologically rich language [21] and to build a language model for such language, one has to consider many features. The POS tagging approaches like rule based and hidden Markov model can not handle much features. The support vector machine based POS tagger has been implemented in [7] for a Bengali language which is also morphologically rich and shown the outstanding performance. In [7] rich feature set has been used to model the language characteristic. In [8] SVM based tagger was proposed which is efficient, portable, scalable and trainable. Support vector machine (SVM) are recently developed supervised learning method having good performance and generalization [24]. SVM has been successfully applied in text classification and shown that SVM can handle large features and is resist of overfitting [13].

Using many features, we can make a strong language model that can be used for POS disambiguation.

1.3 Objectives

Since the POS tagger should be accurate, fast, portable and trainable. The main objective of this dissertation is to build such a Nepali POS tagger based on support vector machine learning framework. The other motives of this study are as follows:

- Since the POS tagging is a prototyping problem in NLP, the technique developed for this can applied for other NLP problem as well.
- The tagger build in this dissertation will be compared with the existing state of art tagger.

1.4 Organization of Thesis

The rest of this thesis is organized as: chapter 2 gives a brief discussion of basic concept related to this work, chapter 3 is a survey of the major existing taggers, chapter 4 details the implementation of the Support vector Machine based tagging algorithm, chapter 5 presents the tagging and tagger-making results, and chapter 6 concludes the thesis, summarizing its achievements and further recomendations.

CHAPTER 2

BACKGROUND AND PROBLEM DEFINITION

2.1 Background

2.1.1 Natural Language Processing

Natural Language Processing (NLP) has been developed in 1960 as a subfield of Artificial Intelligence and Linguistics [14]. The aim of NLP is studying problems in the automatic generation and understanding of natural language. A Natural Language is any of the languages naturally used by humans, *i.e* .not an artificial or machine language such as a programming language like C language, Java, Perl etc.

NLP is a convenient description for all attempts to use computers to process natural language. NLP is also an area of artificial intelligence research that attempts to reproduce the human interpretation of language for computer system processing. The ultimate goal of NLP is to determine a system of language, words, relations, and conceptual information that can be used by computer logic to implement artificial language interpretation. NLP includes anything a computer needs to understand natural language (written or spoken) and also generate the natural language. To build computational natural language systems, we need Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLG systems convert information from computer databases into normal-sounding human language, and NLU systems convert samples of human language into more representation that are easier for computer programs to manipulate. Some of important levels of NLP are as follows:

Phonological Analysis: Phonology is the study of sound system in a language. The minimal unit of sound system is the phoneme which is capable of distinguishing the meanings in the words. The phonemes combine to form a higher level unit called syllable and syllables combine to form the words. Therefore, the organization of the sounds in a language exhibits the linguistic as well as computational challenges for its analysis. This level deals with the interpretation of speech sounds within and across words. There are, in fact, three types of rules used in phonological analysis: 1) phonetic rules – for sounds within words; 2) phonemic rules – for variations of pronunciation when words are spoken together, and; 3) prosodic rules – for fluctuation in stress and intonation across a sentence. In an NLP system that accepts spoken input, the sound waves are analyzed and encoded into a digitized signal for interpretation by various rules or by comparison to the particular language model being utilized.

Morphological Analysis: This level deals with the componential nature of words, which are composed of morphemes – the smallest units of semantic meaning. For example, the word preregistration can be morphologically analyzed into three separate morphemes: the prefix pre, the root ‘registra’, and the suffix ‘-tion’. Since the meaning of each morpheme remains the same across words, humans can break down an unknown word into its constituent morphemes in order to understand its meaning. Similarly, an NLP system can recognize the meaning conveyed by each morpheme in order to gain and represent meaning. For example, adding the suffix ‘-ed’ to a verb, conveys that the action of the verb took place in the past. This is a key piece of meaning, and in fact, is frequently only evidenced in a text by the use of the -ed morpheme. Typically, a natural language processor knows how to understand multiple forms of a word *i.e.* its plural and singular, for example, *ghar (घर)* ‘house’ *ghar-haru (घरहरु.)* ‘house-s’. From structural point of view, the words can be simple, complex and compound. For example, *ghar* ‘house’, *ghar-haru* ‘house-Plural’, *ghar-ghar-ai* ‘each house’.

Lexical Analysis: At this level, humans, as well as NLP systems, interpret the meaning of individual words. Several types of processing contribute to word-level understanding – the first of these being assignment of a single part-of-speech (POS) tag to each word. In this processing, words that can function as more than one part-of-speech are assigned the most probable part-of speech tag based on the context in which they occur. The lexical level may require a lexicon, and the particular approach taken by an NLP system will determine whether a lexicon will be utilized, as well as the nature and extent of information that is encoded in the lexicon.

Syntactic Analysis: Syntactic analysis uses the results of morphological analysis and lexical analysis to build a structural description of the sentence. The goal of this process, called parsing, is to convert the flat list of words that forms the sentence into a structure that defines the units that are represented by that flat list. The important thing here is that a flat list of words has been converted into a hierarchical structure and that the structures correspond to meaning units when semantic analysis is performed.

Semantic Analysis: It derives an absolute (dictionary definition) meaning from context; it determines the possible meaning of a sentence in a context .The structures created by the syntactic analyzer are assigned meaning. Thus, a mapping is made between individual words into appropriate objects in the knowledge base or data base. It must create the correct structure s to correspond to the way the meaning of the individual words combine with each other. The structures for which no such mapping is possible are rejected.
Example: the sentence "colorless green ideas…….." would be rejected as it has no such semantic mapping, because colorless and green make no sense.

Discourse Integration: The meaning of an individual sentence may depend on the sentences that precede it and may influence the meaning of the sentences that follow it.
Example: the meaning of word "it" in the sentence, "you wanted it" depends on the previous discourse context.

Pragmatic Analysis: It derives knowledge from external commonsense information; it means understanding the purposeful use of language in situations, particularly those aspects of language which require world knowledge.
Example: If someone says "the door is open" then it is necessary to know which door "the door" refers to; here it is necessary to know what the intention of the speaker: could be a pure statement of fact, could be an explanation of how the cat got in, or could be a request to the person addressed to close the door.

2.1.2 Major Application of Natural Language Processing

NLP is having a very important place in our day-to-day life due to its large natural language applications. By means of these NLP applications the user can interact with computers in their own mother tongue by means of a keyword and a screen. The few NLP processes are:

- Part-of-speech tagging
- Information retrieval
- Machine translation
- Named entity recognition

- Natural language generation
- Question answering
- Spoken dialogue system
- Text simplification
- Text to speech
- Speech recognition etc.

2.1.3 Computational Linguistics

Computational linguistics is the scientific study of language (i.e. statistical and/or rule-based modeling of natural language) from a computational perspective. Traditionally, computational linguistics was usually performed by computer scientists who had specialized in the application of computers to the processing of a natural language. Computational linguists often work as members of interdisciplinary teams, including linguists (specifically trained in linguistics), language experts (persons with some level of ability in the languages relevant to a given project), and computer scientists. In general, computational linguistics draws upon the involvement of linguists, computer scientists, and experts in artificial intelligence, mathematicians, logicians, cognitive scientists, cognitive psychologists, psycholinguists, anthropologists and neuroscientists, amongst others. Some of the areas of research that are studied by computational linguistics include:

- Computational complexity of natural language, largely modeled on automata theory, with the application of context-sensitive grammar.
- Computational semantics comprises defining suitable logics for linguistic meaning representation, automatically constructing them and reasoning with them.
- Computer-aided corpus linguistics.
- Design of parsers or chunkers for natural languages.
- Design of taggers like POS-taggers.
- Machine translation.

2.1.4 Corpus linguistics

Corpus linguistics is now seen as the study of linguistic phenomena through large collections of machine-readable texts: **corpora**. These are used within a number of research areas going from the descriptive study of the syntax of a language to language learning. Corpus linguistics has developed considerably in the last decades due to the great possibilities offered by the processing of natural language by computers having large storage capacity. The availability of computers and machine-readable text has made it possible to get data quickly and easily and also to have this data presented in a format suitable for analysis. Corpus linguistics is, however, not the same as mainly obtaining language data through the use of computers. Corpus linguistics is the study and analysis of data obtained from a corpus. The main task of the corpus linguist is not to find the data but to analyze it [10]. Computers are useful, and sometimes indispensable, tools used in this process.

2.1.5 Machine learning

It is a recent field of artificial intelligence (AI) which aim to make a machine able to learn as human learns the things. Marvin Minsky (1986) defined learning as *"it is making useful change in the working of our mind"*. Machine learning exists in various forms: supervised learning, unsupervised learning, semi supervised or minimally supervised learning, reinforcement learning etc. In its basic form, machine learn the knowledge form some sources and then generalize that knowledge for other instances.

2.1.5.1 Classification

Given the example data $\{(x_{i,}y_i), i=1\dots n\}$,where the x_i is input vector and the y_i is its associated label or class. Then the classification task is to learn the discriminate function

$$y=f(x),$$

which correctly classify the example data and optimized so that it will make minimal error on the classification of unseen data.

If the label 'y' is not discrete as above, then this task is called regression. Based on these examples (x_i, y_i), one is particularly interested to *predict* the answer for other cases before they are explicitly observed. Hence, learning is not only a question of remembering but also of generalization to unseen cases.

2.1.5.2 Support Vector Machine (SVM)

This is the supervised machine learning approach that can be used for both classification and regression. In their basic form, SVM construct the hyperplane in input space that correctly separate the example data into two classes. This hyperplane can be used to make the prediction of class for unseen data. The hyperplane exist for the linearly separable data [4].
This can be illustrated with figure

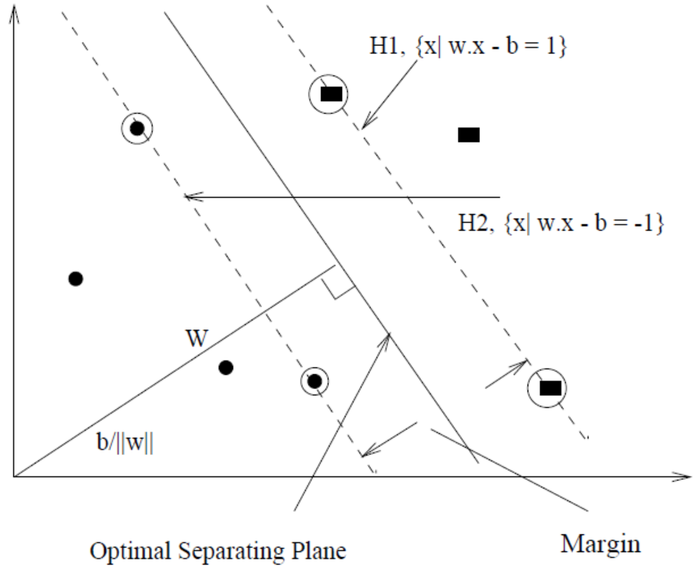

Figure 2.1: Support Vector Machine

The equation for general hyperplane can be written as

$$w.x - b = 0 \qquad \text{(Equation 2.1)}$$

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….n and y_i ∈(+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 hyperplans that can separate the training data correctly but the SVM find one hyperplane that maximize the margin between two supporting hyperplanes. 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

Minimize f= $\frac{|w|^2}{2}$ ……………………………(Equation 2.2)

Subject to constraints $y_i(w.x_i - b) \geq 1$

This can be solved by the variant of quadratic programming technique [13]

2.1.5.2.1 Kernel Trick

To deal with nonlinear separation, the same formulation and techniques as for the linear case are still used. We only transform the input data into another space (usually of a much higher dimension) so that a linear decision boundary can separate positive and negative examples in the transformed space. The transformed space is called the feature space. The original data space is called the input space [4].

The basic idea is to map the data in the input space *X* to a feature space *F* via a nonlinear mapping "ϕ",

$$\phi : X \to F$$
$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

After the mapping, the original training data set {(x_1, y_1), (x_2, y_2), …, (x_r, y_r)} becomes:

{(ϕ(x_1), y_1), (ϕ(x_2), y_2), …, (ϕ(x_r), y_r)}

Then perform linear separation in this feature space. Geometric interpretation is shown in figure 2.2.

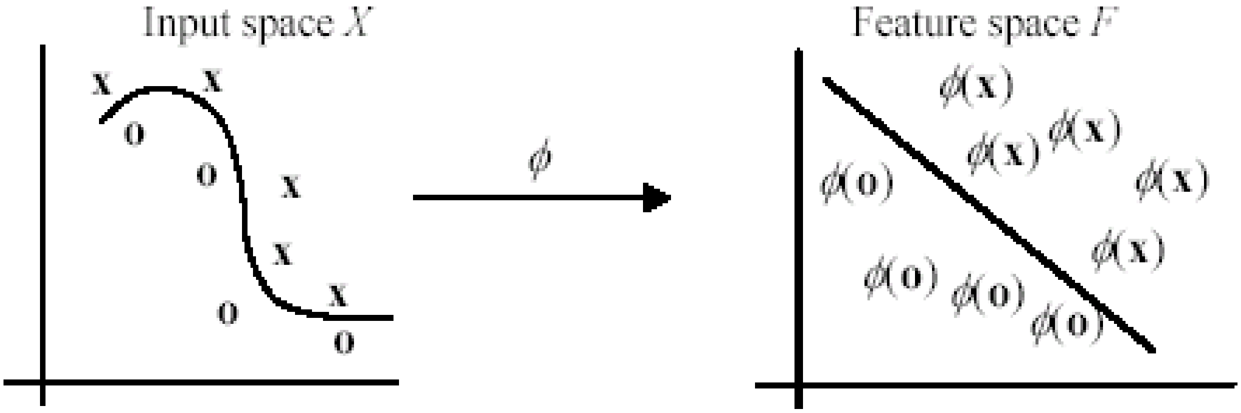

Figure 2.2: Feature Mapping [4]

The potential problem with this explicit data transformation and then applying the linear SVM is that it may suffer from the curse of dimensionality [4]. The number of dimensions in the feature space can be huge with some useful transformations even with reasonable numbers of attributes in the input space. This makes it computationally infeasible to handle. Fortunately, explicit transformation is not needed. In SVM, this is done through the use of kernel functions, denoted by K,

$$K(\mathrm{x}, \mathrm{z}) = \langle \phi(\mathrm{x}) \cdot \phi(\mathrm{z})\rangle$$

For example let us take Polynomial kernel

$$K(\mathrm{x}, \mathrm{z}) = \langle \mathrm{x} \cdot \mathrm{z}\rangle^d$$

Let us compute the kernel with degree $d = 2$ in a 2-dimensional space: $\mathrm{x} = (x_1, x_2)$ and $\mathrm{z} = (z_1, z_2)$.

This shows that the kernel $\langle \mathrm{x} \cdot \mathrm{z}\rangle^2$ is a dot product in a transformed feature space.

$$\begin{aligned}\langle \mathbf{x} \cdot \mathbf{z}\rangle^2 &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1{}^2 z_1{}^2 + 2x_1 z_1 x_2 z_2 + x_2{}^2 z_2{}^2 \\ &= \langle (x_1{}^2, x_2{}^2, \sqrt{2}x_1 x_2) \cdot (z_1{}^2, z_2{}^2, \sqrt{2}z_1 z_2)\rangle \\ &= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z})\rangle,\end{aligned}$$

2.1.6 Optimization

Many situations arise in machine learning where we would like to optimize the value of some function. It turns out that in the general case, finding the global optimum of a function can be a very difficult task. However, for a special class of optimization problems, known as convex optimization problems [13], we can efficiently find the global solution in many cases. Here, “efficiently” has a both practical and theoretical connotation: it means that we can solve many real-world problems in a reasonable amount of time, and it means that theoretically we can solve problems in time that depends only polynomially on the problem size.

A convex optimization problem is an optimization problem of the form minimize $f(x)$

subject to $x \in C$

where f is a convex function, C is a convex set, and x is the optimization variable. A linearly constrained optimization problem with a quadratic objective function is called a quadratic program (QP). The general quadratic program can be written as

Minimize $f(\mathbf{x}) = \mathbf{cx} + 1/2\ \mathbf{x}^T\mathbf{Q}\ \mathbf{x}$

Subject to $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$

where **c** is an *n*-dimensional row vector describing the coefficients of the linear terms in the objective function, and **Q** is an ($n \times n$) symmetric matrix describing the coefficients of the quadratic terms. If a constant term exists it is dropped from the model. As in linear programming, the decision variables are denoted by the *n*-dimensional column vector **x**, and the constraints are defined by an ($m \times n$) **A** matrix and an *m*-dimensional column vector **b** of right-hand-side coefficients. We assume that a feasible solution exists and that the constraint region is bounded.

2.2 Problem Statement

A Nepali sentence can be expressed as

$S=(w_{1,}\ w_{2,}\ldots\ldots\ldots\ldots\ldots.w_n)$

Where w_i is the i^{th} Nepali word in sentence S. The POS tagging will assign a tag to each word and then after the POS tagging, result will be

$T=(t_{1,}t_2,\ldots\ldots\ldots\ldots\ldots\ldots t_n)$

Where t_i is the tag assigned to word w_i. Our goal is to determine correct tag for each word in a given sentence. The categories indicated by POS tags are defined in advance so the POS tagging problem is equivalent to classification problem. Hence they are capable of being handled by machine learning methods. In this work, Support Vector machine with one versus rest multi classification approach will be applied to solve the problem.

CHAPTER 3

LITERATURE REVIEW

3.1 Existing Corpus Review

A corpus is valuable resources in Natural Language Processing. There existence in correct form makes the NLP a more fruitful process. The most well known corpora for English are probably the Brown Corpus and the PenTreeBank corpus. The Brown Corpus contains over a million words of American English and it was tagged in 1979 using the TAGGIT [23] tagger. Nowadays, corpora tend to be much larger, and are compiled mainly through projects and initiatives such as the Linguistic Data Consortium (LDC), the Consortium for Lexical Research (RLC) etc. These associations provide corpora as the Wall Street Journal (WSJ, 300 million words of American English), the Hansard Corpus (bilingual corpus containing 6 years of Canadian Parliament sessions), The Lancaster Spoken English Corpus (SEC) etc. Although most corpora limit their annotation level to part of speech tags, some other higher level annotations and constitute an important source of knowledge for those researching in NLP. For instance, PennTreebank corpus is an example of syntactically analyzed corpora (called Treebanks), which contains 3 million words from the WSJ corpus. Until few years ago, the existing corpora were all of the English language. Nevertheless, the success and applicability of corpus in linguistics as well as in NLP, has raised a wide interest and caused its quick extension to other languages. The following list (not exhaustive) provides some examples of available corpora of languages other than English.

1. Spanish: The LexEsp corpus which contains 5.5 million morph syntactically tagged words, the corpus of the Real Academia Espanola, which contains 200 million tagged and lemmatized words.

2. German: The NEGRA[3] corpus from the Saarland University, which contains German newspaper texts with syntactic annotation.

3. French: The `Tresor de la Langue Francaise' (TLF) which contains 150 million words of written French. Swedish: The `Bank of Swedish' corpus and other materials collected by the Department of Swedish of the University of Goteborg.

4. Nepali: The Nepali National Corpus (NNC) from NELRALEC (Nepali Language Resources and Localization for Education and Communication) project, which contain 14 milion Nepali words. It is 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 tagset with 112 tags). This data then served as the basis for the training of an automatic tagger. The Nepali English parallel corpus annotated with 43 pos tag developed at Madan Puraskar Pustakalaya (MPP) contains nearly 88000 words [20].

3.2 Nepali Tagset Review

NELRALAC tagset is the fist work in developing Nepali tagset which consist of 112 tags. This tagset has been compiled with reference to widely published grammars of Nepali. This tagset was used to tag (Nepali National Corpus) NNC manually and semi manually. With large tagset, as in [20] showed that error rates of annotation could be much higher when the size of the tagset was a big one, the reason primarily being the chances of assigning incorrect tags to the words out of confusion while manually annotating the training data itself.

It was with such motivations that a smaller sized POS Tagset was later on developed that consist of just 43 tags. While developing the tagset, maximum care has been taken to ensure that this minimalist approach does not unnecessarily eliminate the unavoidable lexical categories of the language. The design of this Nepali POS Tagset was inspired by the PENN Treebank POS Tagset. Hence, whenever possible, the same naming convention has been used as in the case of the Penn Treebank Tagset. In this dissertation work, this tagset is used as resource and referred as Nepali POS tagset.

3.2.1 Specification of Nepali POS Tagset

In this section, the development of the POS tagset for Nepali will be discussed in brief. One of the crucial issues that needs to be subtly addressed while designing a POS tagset is its' size. Generally, the assumption is –"the smaller the tagset, the greater the accuracy". However, in saying so, the compulsory categories evident in the language would not be missed and at the same time also not necessarily increase the size of the tagset whenever economy can be maintained. Hence, a middle ground has been adopted while designing a POS tagset for Nepali.

The tagset for Nepali currently includes 43 tags and covers almost all the grammatical categories in the Nepali language. By the reference of Penn Treebank Tagset, the tagset of the Nepali was designed [21]. The grammatical categories person, number and gender are found distinctively in pronouns and adjectival and verbal inflections. In this tagset, these distinctions have not been considered. That is both masculine and feminine pronouns get the same tag. Similarly, adjectives inflected for masculine and feminine references receive the same tag.

3.2.2 Description of Nepali Tagset

In this dissertation, the total of 43 tags described in [21] are used for part-of-speech tagging of Nepali text and covers almost all the grammatical categories in Nepali language. Basically, the tagset was guided by the general principle - “The smaller the size of the tagset, the more accurate a tagger” [21]. The short description of tags in [21] is given in the following table 1. It contains the main 21 categories and 43 POS tags.

Category	POS Tag ID No	POS Name	POS Tag	Example
Noun	1	Common Noun	NN	हरिले भाइ/NN लाई किताब दियो
	2	Proper Noun	NNP	हरि/NNP ले सरिता/NNP लाई किताब दियो
Pronoun	3	Personal Pronoun	PP	तिमि/PP ले भात खायौ?
	4	Possessive Pronoun	PP$	यो मेरो/PP$ घर हो
	5	Reflexive Pronoun	PPR	आफू/PPR त भात खाइन्छ
	6	Marked Demonstrative	DM	अर्की/DM केटी, यस्ता/DM केटा.

	7	Unmarked Demonstrative	DUM	त्यो/DUM मेरो साथी.
Verb	8	Finite Verb	VBF	श्यामले भात खायो/VBF
	9	Auxiliary Verb	VBX	राम भात खादै थियो/VBX
	10	Verb Infinitive	VBI	हरि खाना खान/VBI घर गयो
	11	Prospective Participle	VBNE	काम गर्ने/ VBNE मान्छेलाई बोलाऊ
	12	Aspectual Participle	VBKO	तिमीले दिएको/ VBKO किताब राम्रो छ
	13	Other Participle Verb	VBO	ऊ काम सकेर/VBO घर गयो
Adjective	14	Normal/Unmarked	JJ	असल/JJ केटी टीभी हेर्दैछे
	15	Marked Adjective	JJM	त्यो केटी धेरै राम्री/JJM छे
	16	Degree Adjective	JJD	उच्चतम/JJD बिन्दु
Adverb	17	Manner Adverb	RBM	ऊ ढिलो/RBM हिड्छ
	18	Other Adverb	RBO	यहाँ/RBO बस
Intensifier	19	Intensifier	INTF	ऊ धेरै/INTF चलाख छ
Postpositions	20	Le-Postposition	PLE	हरि-ले/PLE भाइलाई कूट्यो

	21	Lai-Postposition	PLAI	भाइ-लाई/PLAI कूट्यो
	22	Ko-Postposition	PKO	राम-की/PKO बहीनी चलाख छे
	23	Other Postpositions	POP	किताब टेबुल माथि/POP छ
Conjunction	24	Coordinating	CC	म खाजा खान्छु र/CC अफिस जान्छु
	25	Subordinating Conjunction	CS	किनभने, यद्धपि, भन्दा, यदि
Interjection	26	Interjection	UH	आहा!/UH कस्तो राम्रो बगैचा!
Number	27	Cardinal Number	CD	पाँच/CD जना मान्छे दौडदैछन
	28	Ordinal Number	OD	हरि परीक्षामा पाँचौँ/OD भयो
Plural Marker	29	Plural Marker हरु	HRU	गाई-हरु/HRU आए
Question Word	30	Question Word	QW	तिमी किन/QW यहाँ?
Classifier	31	Classifier	CL	दशजना/CL मान्छे आउदैछन्
Particle	32	Particle	RP	खै/RP मैले त/RP तिम्रो कुरा बुझिन
Determiner	33	Determiner	DT	त्यो /DT केटो मेरो साथी हो
Unknown Word	34	Unknown Word	UNW	उसलेउत्तरमा नेकोम्प्रेनास/UNW

				भन्यो
Foreign Word	35	Foreign Word	FW	उसले मध्यान्नमा भेटेर पनि गुड/FW मर्निङ/FW भन्यो
Punctuation	36	Sentence Final	YF	?/YF,
	37	Sentence Medieval	YM	,/YM, ;/YM, :/YM,
	38	Quotation	YQ	‘/YQ, “/YQ
	39	Brackets	YB	() {} []/YB
Header List	40	Header List	ALPH	क)/ALPH, अ/ALPH,
Symbol	41	Symbol	SYM	२/CD %/SYM ब्याज
Abbreviation	42	Abbreviation	FB	.म.पु.पु./FB, मपुपु/FB
NULL	43	NULL	NULL	Noisy symbol, control character etc.

Table 3.1: Description of Nepali Tagset

3.3 A Review of POS Tagging Approaches

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

The work on automatic part of speech tagging started in early 1960s [11]. Klein and Simmon's rule based POS tagger can be considered as the first automatic tagging system [15]. Since rule based approaches needs more sophisticated rules to capture the language knowledge, later on the data driven approaches were developed as [5] and recently machine learning approaches are being developed [3, 8,17]. In the following sections, the some of the related tagger that has been implemented for English and other language with their performance will be reported and subsequently, the tagger available for Nepali language are also mentioned.

3.3.1 Linguistic Tagger

The tagger based on rules engineered by linguist is referred as linguistic tagger. The most representative of such pioneer taggers was TAGGIT [23], which was used for an initial tagging of the Brown Corpus. These tagger model the language learning through the help of rules that may vary from hundreds to thousand in number. Each rule contains instructions for an operation to be performed, and a context describing where that rule should be applied. The operation specified by the matching rule will be performed to alter the list of tags associated with an ambiguously-tagged word so that one or more potential tags are eliminated from consideration and reduces the ambiguity. In both sentences below, initially run would be tagged as a verb:

*The **run** lasted thirty minutes.*

*We **run** three miles every day.*

Now the rules for disambiguation are there to choose the correct POS tag. The following table shows sample template that is used in Brill's rule tagger in table 3.2.

Rule	Expalnation
Change(A,B, prevTag(C)	Change tag A to Tag B if previous tag is C
Change(A,B, NextTag(C)	Change tag A to Tag B if the following tag is C

Table 3.2: Rule Template in Brill Tagger

After applying one of these rule, the word ‘run’ will be tagged as noun in first sentence and as verb in second sentence.

Rule based approaches are often associated with parsing. For example, the program of Harris [11] was a parser and Klein and Simmons [15] describe rule-based tagging as a preliminary stage to an eventual parsing process. However, there is no necessary link between rule-based approaches and parsing, as demonstrated by Green and Rubin [23], who’s tagging was not associated particularly with a parser.

The earliest work on rule based tagging was by Klein and Simmons and Greene and Rubin [15, 23], which was begun before any other methodology had been developed, and marked the first attempt to solve the problem of automated POS disambiguation.

The accuracy reported by the first rule-based linguistic English tagger was slightly below 80% [3]. It is tedious and requires the language expert to write the contextual rule and then it is computatiolly expensive to match the rule.

3.3.2 Probabilistic Taggers Using Markov Models

The basic idea of probabilistic or stochastic approaches consist of building statistical model of language and use it to disambiguate the word sequence by assigning the most probable tag sequence using maximum likelihood. The model is build on the basis of statistical information concerning the frequency with which sequences of tags occurs. This information is gathered from long stretches of running text usually form training corpus. For instance, acquiring frequency statistics on a tagged corpus of English, a system might discover that the tag for a subject pronoun is followed by the tag for a verb 70% of the time, the tag for an adverb 29% of the time, and the tag for a noun 1% of the time. If that system, during the course of tagging, then encounters a word following a subject pronoun that was ambiguously tagged as either noun or verb, it can use its statistical knowledge to deduce that the verb tag is most likely to be correct [10].

In practice, a model as primitive as the example here would be incapable of handling long sequences of ambiguous tokens and would be unlikely to perform particularly well.

The most representative model of this category is hidden Markov model (HMM), which can be represented as in [22], with five elements.

1. The number of distinct states (N) in a model. We denote the individual state as S={s_1,s_2,…s_n}. In case of Part-of-speech tagging, N is the number of tags in the tagset {T} that will be used by the system. Each tag in the tagset corresponds to one state in the HMM.

2. The number of distinct output symbols (M) in the HMM. We denote the individual symbol as V = {v1,v2,…..vm}. For Part-of-Speech tagging, M is the number of words in the lexicon of the system.

3. The state transition probabilities A = {a_{ij}}. The probability a_{ij} is the probability that the process will move from state i to state j in one transition.. In part-of-speech tagging, the states represent the tags, so a_{ij} is the probability that the model will move from tag t_i to t_j (1 <= i,j <= N) - in other words, the probability that tag t_j follows t_i. This probability can be estimated using data from a training corpus.

*4. T*he observation symbol probability distribution, *B = {$b_j(k)$}.* The probability $b_j(k)$ is the probability that the k^{th} output symbol will be emitted when the model is in state j. For part-of-speech tagging, this is the probability that the word w_k will be emitted when the system is at tag t_j (i.e., $P(w_k|t_j)$ where,1<= j <= N & 1 <= k <= M *)*. This probability can be estimated using data from a training corpus.

5. The initial state distribution ,π = {π_i},. π_i is the probability that the model will start in state i where, 1<= i <= N. For part-of-speech tagging, this is the probability that the sentence will begin with tag t_i.

When using an HMM to perform POS tagging, the aim is to determine the most likely tag (states) sequence that generates the words of a sentences (the sequence of output symbols). In other words, HMM taggers choose the tag sequence that maximizes the following formula

$$\hat{T} = \left(t_1, t_2, \ldots\ldots. t_n\right)^* = \arg\max_T \prod_{i=1}^{n} P(t_i \mid t_{i-1}, t_{i-2}, \ldots.. t_{n-1}) * P(w_i \mid t_i)$$

Where the first part is n-gram and second part is most frequent tag. The Markov assumption is that the "probability of a word depend one its own tag and the probability of current tag depends upon n-previous tag" [18].

Now the most probable tag sequences can be searched using Viterbi algorithm [22].

3.3.2.1 Variation on HMM

In the first order Markov model, the state transition probability of a particular tag *ti* depends on the previous one tag in the sequence. The symbol emission and state transition probabilities are estimated directly from the labeled training data as follows.

Contextual probability or state transition probability:

$$P(t_i|t_{i-1}) = \frac{C(t_i, t_{i-1})}{C(t_{i-1})} \text{......................(Equation 3.1).}$$

And lexical probability or symbol emission probability:

$$P(w_i|t_i) = \frac{C(w_i, t_i)}{C(t_i)} \text{...........................(Equation 3.2).}$$

This is also called Bi-gram HMM model since it takes two tag pair in account for modeling the language context [5].

For the second order HMM, the transition probability is calculated as

$$P(t_i|t_{i-1},t_{i-2}) = \frac{C(t_i, t_{i-1}, t_{i-2})}{C(t_{i-1}, t_{i-2})} \text{.........................(Equation 3.3)}$$

And lexical probability is calculated with the Equation 3.2.

Where C( ) denotes the number of occurrence in the labeled training data. As we are dealing with a small labeled corpora (FinalNepaliCorpus), it is often possible that C( ) will become zero. To cope with the above situation, state transition probabilities are smoothed and symbol emission probabilities are estimated for handling unknown words that are not in the labeled corpora.

TnT [2] tagger is a widely used tri gram tagger which is based on second order HMM. The other consideration taken in TnT are the methods of smoothing and interpolation and suffix analysis to handle the unknown words.

The essential difference between using bigrams and trigrams is that the latter has *S* times as many parameters to be estimated, where *S* is the size of the tagset. This requires more training data, since the average frequency of each trigram is so much lower than the average frequency of the equivalent bigram; [5] suggest that as a rule of thumb, “the training set needs to be large enough to contain on average ten instances of each type of tag sequence that occurs”. However, there does not appear to be any general agreement about whether calculating Markov model path probabilities using trigrams produces a notable improvement over bigrams, or on whether the improvement is worth the necessary extra data.

On the one hand, several studies into part-of-speech disambiguation algorithms have made use of trigrams. Merialdo [18] uses to compare training on tagged and untagged text, is a trigram tagger. Merialdo [18] suggest that “trigram models are usually superior to bigram ones” with the provision that sufficient training data must be available. On the other hand, other studies have questioned the use of trigrams. [6] test a bigram and a trigram version of their Markov model tagger and reports that the trigram model has a lower error rate but is slower at processing time.

However, in general, the performances reported for the taggers discussed in this section ranges between 95% and 97% for English.

3.3.1 Neural Network Based Taggers

A neural network is a learning system whose architecture consists of two or more interconnected layers of processing units. Each unit may be activated or not activated. The activations of the bottom layer correspond to the input to the system, and the activations of the top layer indicate the output. The optional intermediate layers are called “hidden” because while they contribute to the processing, they do not connect to the world outside the system at all. The activation of each unit propagates to other units via that links connecting them. The parameters of the system are the weights given to the links, and the activation values of the units. Thus, the activation of non-input units depends upon the activations of the units to which they are connected and the weightings of the links connecting them. The training of a neural network consists of iteratively adjusting the weights of the connections and the activation values to produce a system which produces the correct output for the training data.

Prior to their application to part-of-speech disambiguation, neural networks had been successfully used in speech recognition. The subsequent systems using neural networks include Nakamura’s NETgram system and the Net-Tagger system of Schmid [24].

The input to a neural network disambiguation system is the ambiguously tagged word, and some amount of context. For each word examined by the network, there is a set of units in the input layer equal in number to the number of tags in the tagset. The input units corresponding to the tag or tags marked up on the word examined are activated. The output layer then indicates the tag the system has chosen. The amount of context used varies greatly between different systems. In [24], they used as their input the four unambiguously tagged words preceding the target word and one ambiguously tagged word following it. Nakamura’s Netgram system allows the scope of the system to be varied, taking one, two or three words before the target word as the context, but *no* words after it. [24] also experimented with different context scopes, reporting that three words before the target word and two words afterwards were optimal (to wit, using more context did not improve the system, using less only worsened it slightly). It can thus be seen that in general, neural networks use a wider context than a Markov model can, based on bigram or trigram transition probabilities. This is because the number of parameters that must be estimated for a Markov model is equal to the size of the tagset raised to the power of the length of the sequences examined. Thus, the amount of training data needed to accurately estimate (say) six-gram probabilities for a tagset of (say) 100 – giving one trillion transition probabilities –would be astronomical [24]. For a neural network, the number of parameters is much smaller, since they are what Benello, Mackie and Anderson refer to as “complex conditional probabilities” rather than “simple first-order transition statistics”. Therefore wider contexts can realistically be taken into account. However, neural network taggers still use surface-level linguistic information only. They do not use any knowledge of syntactic structure.

3.3.2 Decision Tree Induction Based Taggers

Decision tree is a classification approaches which construct the tree in top down manner using the attribute the data satisfies.

Statistical decision tree to solve the tagging problem is used in [17]. Using 96% of WSJ of Penn Treebank as the training data, the authors group all multi-tag words into 253 ambiguity classes, such as class *JJ-VBG* (e.g. *amusing*, *exciting*, etc.) and class *JJ-VBD-VBN* (e.g. *amused*, *excited*, *surprised*, etc.). The tagging problem thus becomes a classification problem. They selected the following attributes: the third tag to the left of the word in question (t_{-3}), the second tag to the left (t_{-2}), the first tag to the left (t_{-1}), the first tag to the right (t_{+1}), and the second tag to the right (t_{+2}), and the word's spelling features, such as capitalization, containing digits, etc. In other words, the best tag of a word is decided by it contextual and spelling characteristics.

They tested their tagger on the remaining 4% of WSJ and the reported accuracy was 97%.

3.3.3 Support Vector Machine based taggers

SVM is binary classification method which can be extended for multiclass classification method. SVM do the classification of data into two classes with a hyperpalne which is drawn by maximizing the margin between the example data nearest to hyperplane. Since POS tagging is multiclass classification, SVM are extended for such problems using One-against-all method.

The basic idea of SVM is to separate the instances into two class by a hyperplane. the POS tagging is done by training a classifier for each POS tag and then the word to be classified is given to all classifer to classify this word either the word has that tag or not. Now the highest valued classes is taken as the final POS tag to that word. This strategy is called “one versus rest” [20]. This is computationally cost to train each POS tag with the one vs all technique but in case of POS tagging, the different heuristic can be used to enhance the process [8].

Support Vector Machines (SVM) has been used for POS tagging with simplicity and efficiency. Nakagawa in [20], first used the SVM based machine learning technique for POS tagging. The main disadvantage of the system was low efficiency (running speed of 20 words per second was reported). Further, Gimenez and Marquez [8] in their work proposed a SVM based POS tagging technique which is 60 times faster than the earlier one. The tagger also significantly outperforms the TNT tagger [8].

Support vector machines have two advantages over other models: They can easily handle high dimensional spaces i.e. large number of features and they are usually more resistant to over fitting.

3.3.4 Nepali Language Taggers

After giving short description of previous work on POS tagging for English and other resource rich language, in this section, the work done in Nepali language related with POS tagging and NLP are reviewed.

The *Unitag*[1] has been developed or customized for Nepali language and was used for semi automatic tagging of Nepali National Corpus under the NERLAC project. The tagset used is NERLAC tag set with 112 tags. *Unitag* was originally developed for Urdu language by hardie et [10].It consists of a powerful morphological and lexical analysis system, and twin disambiguation modules, one based on hand-written rules and the other using a probabilistic system based on a Markov model. After tagging, the corpus was manually reviewed and the correction was done. Since the tagset used was which large, it introduced the more error in tagging.

In [1] , the TnT has been used as POS tagger with the 43 tags and training corpus of mideum siz as one of the pipelined module for computational grammar analyzer.

First order Markov model has been implemented in [12] which use the same POS tagset as in [1] and reports the good accuracy (91%) for known word.

3.4 Measurement of the performance of taggers

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. So these four criteria are not necessarily mentioned in their literature.

[1] Unitag is originally developed for Urdu and later customized for Nepali by Hardie et.al at Bhashasanchar project.

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:

$$\text{Accuracy} = \frac{correct\ no\ of\ token\,tag\ pair\ occurance}{number\ of\ correct\ \ token\,tag\ pair\ that\ is\ possible}$$

The efficiency of tagger is measured in terms of number of word it can tag per seconds. Since the accuracy is major concern in POS tagging, this measurement is not much used in performance comparison of tagger.

The other two criteria: portable and trainable are related to language independent issue. The first criterion says that whether a particular tagger can be used for other language than the language for which it is developed with out or with minimal modification. “Trainable” refers to whether the tagger can improve itself or produce a new tagger if given a pre-tagged corpus by users.

The following *Table 3.3* show the comparison of different tagger reviewed in this chapter based upon these four criteria.

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

Table 3.3: Comparison of Existing Tagger for English

CHAPTER 4

IMPLEMENTATION

4.1 Support Vector Machine Algorithm

The optimization problem for SVM in its basic form is

Minimize f= $\frac{|w|^2}{2}$ ……………………….(Equation 4.1)

Subject to constraints $y_i(w.x_i - b) \geq 1$[as in chapter 2]

The equivalent dual formulation of this problem can be written as

Min $_{w\,b}$ max $_{\alpha}$ $\frac{1}{2}||W||^2 - \sum_j \alpha_j [y_j(< x_j.w > +b) - 1]$

Subject to $\alpha_j \geq 0$

Where α's are lagrangian multipliers.

With some simplification, the equations can be written as

Min $_{w,b}$ Max $_{\alpha}$ $\frac{1}{2}||W||^2 - \sum_j \alpha_j [y_j(< x_j.w > +b)] + \sum_j \alpha_j$ …………..(Equation 4.2)

Subject to $\alpha_j \geq 0$

Wishing to minimize both w and b while maximizing α's leaves us to determine the saddle points. The saddle points [4] correspond to those values where the rate of change equals to zero. This is done by differentiating the Lagrangian-primal (Lp) equation (4.2), with respect to w and b and setting their derivatives to zero:

$\frac{\delta L}{\delta w} = 0\, w \Rightarrow w - \sum_j \ \alpha_j y_j \, X_i \, = 0$………………………(Equation 4.3)

$w = \sum \alpha_j \; y_j X_i$ ……………………………………….(Equations 4.4)

$\frac{\delta L}{b} = 0 \Rightarrow -\sum_j \alpha_j y_j$=0 ……………………….(Equations 4.5)

$\sum_j \alpha_j y_j \; = 0$ ………………………(Equations 4.6)

Putting the value of (4.4) and (4.6) in above equation (4.2), we have

$$max_\alpha \quad -\frac{1}{2}\sum_j \alpha_j y_j X_j \sum_j \alpha_j \; y_j X_j + \sum_j \alpha_j$$

Equal to

$$max_\alpha \sum_j \alpha_j - \frac{1}{2}\sum_j \alpha_j y_j X_j \; \alpha_i y_i X_i$$

Now the optimization problem becomes

$$max_\alpha \;\; L = \quad \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} \alpha_j y_j X_j \; \alpha_i y_i X_i$$

$$subject\ to \quad \sum_j \alpha_j y_j \; = 0$$

$$\alpha \geq 0$$

This is the quadratic optimization problem and can be solved using the decomposition algorithm as in [13].Decomposition algorithm breaks the whole optimization problem in to smaller sets and solves each set iteratively.

4.2 Problem Setting

POS tagging is a multi classification problem since in natural language there exist more than two tags. As an instance, Nepali language has 43 tags defined to cover all grammatical categories. Number of tags is the number of classes. Since SVM are binary classifier so binarization of problem must be performed before apply them to POS tagging. [8] has suggested the one vs rest binarization of problem. i.e. a SVM is trained for each POS tag in order to distinguish this class and the rest. When tagging the word, the most confident prediction among the all binary SVM is selected. Hence the support vector machine used in this dissertation work is in fact the implementation of support vector machine with one versus rest method as explained below.

4.2.1 One Vs. Rest Binarization of Multiclass Classification

This strategy is based on idea of building one classifier per class. To train N different binary classifiers, each one 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

एउटा /CL लेख/NN लेख/VB

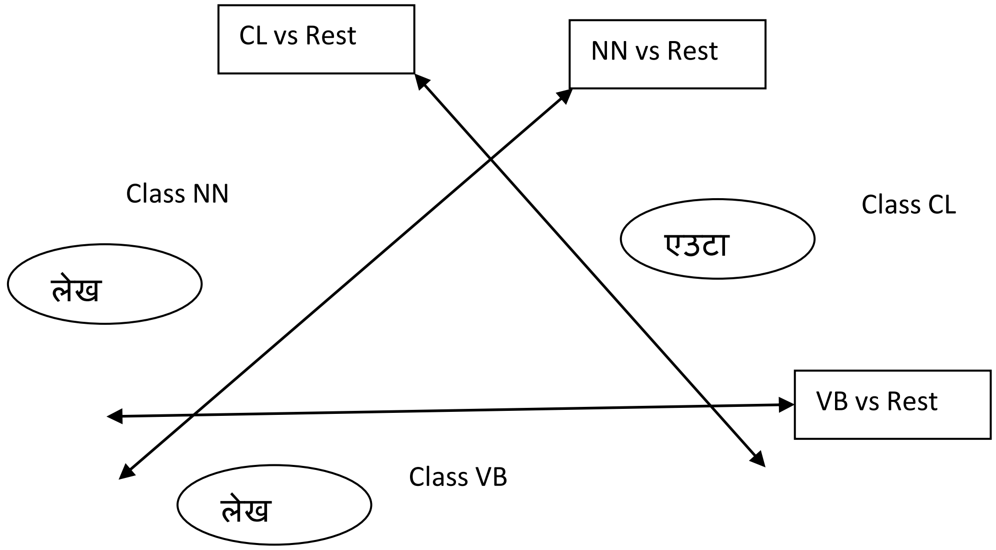

Figure 4.1: One Vs. Rest Classification Example

4.2.2 The Overall System Flowchart

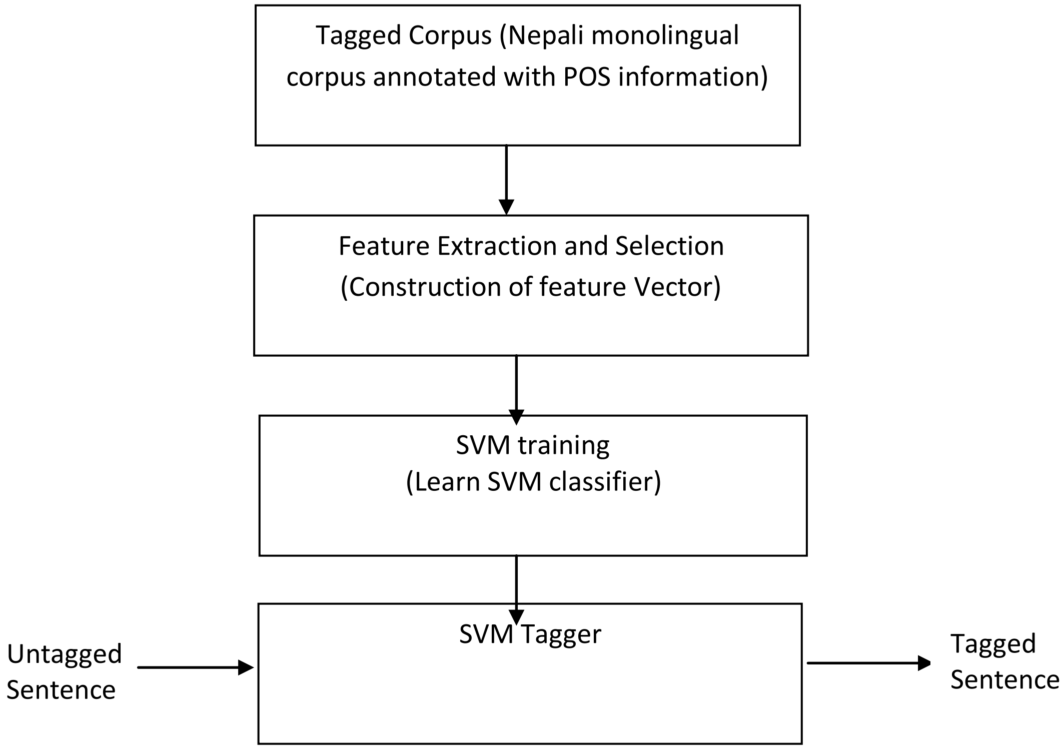
Tagged Corpus (Nepali monolingual corpus annotated with POS information)
Feature Extraction and Selection (Construction of feature Vector)
SVM training (Learn SVM classifier)
SVM Tagger
Untagged Sentence
Tagged Sentence

Figure 4.2: Implementation Model

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

4.2.3 Dictionary

A dictionary is extracted from the training corpus which contains all possible tags of each word.

A dictionary entry has the following format

< word > < N occurrences > < N possible tags > 1{< tag(i) > < N occurrences(i)}N

Example:

```
भने 4 3 VBNE 1 VBO 1, VBF 2
इन्जिनियरिङ 13 2 JJ 1 NN 10
अन्त्य 54 3 RBO 1 JJ 1 NN 8
```

4.2.4 Feature Set

The features used in the experiment are tabulated in the table 4.1.

Word Feature	w_{-3}, w_{-2}, w_{-1}, w_0, w_1, w_2, w_3
POS Feature	p_{-3}, p_{-2}, p_{-1}
Word Bigrams	$(w_{-3},w_{-2}),(w_{-2},w_{-1}),(w_{-1},w_0),(w_0,w_1),(w_1,w_2)$
POS Bigrams	(p_{-3}, p_{-2}) (p_{-2}, p_{-1})
Word Trigram	(w_{-3},w_{-2},w_{-1}), (w_{-2},w_{-1},w_0), (w_{-1},w_0,w_1), (w_0,w_1,w_2), (w_1,w_2,w_3)
Ambiguity Classes	a_0, a_1, a_2,a_3
Maybe's	m_0, m_1, m_2, m_3

Table 4.1: Description of Feature Set

Feature Vector Construction

प्यालेस्टाइनी<JJ> ओलम्पिक<NN> कमिटि<NN>का<PKO> एक<CD> अधिकारी<NN>ले<PLE> **भने<VBNE>** कमिटि<NN>ले<PLE> सब<JJ>भन्दा<VBO> पहिले<PLE> १९७९<CD> मा<POP> सदस्यता<NN>का<PKO> लागि<POP> निवेदन<NN> दिएको<VBKO> थियो<VBX>

The dictionary entry for target word "**भने**" is:

भने 126 4 VBKO 1 VBNE 1 VBO 1 VBF 4

Some of the features along with their ids for the target word **भने /VBNE** are:

w(-3)_is_एक 71

w(-2)_is_अधिकारी 72

w(-1)_is_ ले 73

w(0)_is_**भने 37**

w(1)_is_कमिटि 74

w(2)_is_ले 75

w(3)_is_सब 76

p(-3)_is_CD 77

p(-2)_is_ NN 78

p(-1)_is_PLE 79

a(0)_is_VBKO-VBNE-VBO-VBF 42

m(0)_may_be VBNE 43

m(0)_may_be_VBO 7

m(0)_may_be_VBF 45

a(1)_is_NN-VBO 80

m(1)_may_be_NN 81

m(1)_may_be_VBO 82

…………

…………..

The feature vector for target word **भने is**

+1 71:1 72:1: 73:1 37:1 74:1 75:1 76:1 77:1 78:1 79:1 42:1 43:1 7:1 43:1 7:1 45:1 80:1 81:1 82:1………

Feature Filtering

The feature space can be kept in a convenient size. Smaller models allow for a higher efficiency. In the experiment, the total 88135 features are used. Certain features that appear less then certain threshold values are discarded and the accuracy is not even in the attainable level. The features that appear less than once are discarded by default. The bigram feature appears less than 10, trigram feature that appears less than 5 and unigram feature that appears 50 are discarded and the overall accuracy is same as for by default.

4.2.5 Algorithm for Training (Algorithm 1)

```
INPUT:  Formatted Train-file

OUTPUT:  SVM models learned for all part of speech tags

POS[ ] = {the set of possible part of speech tags}

//Create a different example file corresponding to each POS

 1.  for each example word w tagged as $t_i$ in the train-file do

                Extract features, codify features and create a feature vector $f_i$

                T={the set of possible tags for w}

                Use $f_i$ as a positive example for $t_i$

                Use $f_i$ as a negative example for all $t_j$ in {T-$t_i$}

      end for

 2.  Learn SVM model for each example file

         for each $p_i$ in POS do

                Run the "svm_learn " on the examples corresponding to $p_i$

         end for

 3.  return the SVM models learned for each part of speech
```

Enhancement in Training

However, not all training examples have been considered for all classes. Instead, a dictionary is extracted from the training corpus with all possible tags for each word, and when considering the occurrence of a training word w tagged as t_i, this example is used as a positive example for class t_i and a negative example for all other t_j classes appearing as possible tags for w in the dictionary. In this way, the generation of excessive (and irrelevant) negative examples can be avoided, and training step can be made faster.

4.2.6 Tagging Algorithm (Algorithm 2)

```
INPUT: Formatted test file, Learned SVM models
OUTPUT: Part-of-speech tagged test file
1.  initialize  score = 0
2.  for each target word w in the test file do
        a)  Extract features and create a feature vector for w
        b)  Possible pos[] = possible tags of w in the dictionary
        c)  for each $p_i$ in possible pos do
                    ans = svm classify with pi SVM model
                    if ans > score then
                        score=ans
                        target pos=$p_i$
                    end if
            end for
        Assign target pos to w
    end for
3.  return the Part-of-speech tagged test file
```

4.2.7 Sample Input Output of Different Taggers

Input

सुश्री हाग एलियान्टी को भूमिका खेल्नुहुन्छ । रोल्स-रोयस मोटरकार इन्कर्पोरिटिड ले १९९० मा संयुक्त राज्य अमेरिका मा यस को बिक्री १२०० मा स्थिर हुने अपेक्षा राखेको बतायो । पछिल्लो वर्ष सुबिधा सम्पन्न गाडी निर्माता ले संयुक्त राज्य अमेरिका मा १२१४ वटा कार हरू बिक्रि गर्यो । अध्यक्ष तथा प्रमुख कार्यकारी अधिकृत होवार्डमोशर ले उहाँ ले बेलायत तथा युरोप र सुदूर पूर्वी बजार हरू मा सुबिधा सम्पन्न गाडी निर्माता हरू कालागि संवृद्धि को आशा गरेको बताउनुभयो । बेल औद्योगिक संस्था ले यस को त्रैमासिक लाभांश मा एक सेयर मा ७ सेन्ट बाट १० सेन्ट मा बढायो |

Output

TnT Tagger

सुश्री/NN हाग/NNP एलियान्टी/NNP को/PKO भूमिका/NN खेल्नुहुन्छ/VBF ।/YF रोल्स-रोयस/NNP मोटरकार /NN इन्कर्पोरेटिड/NN ले/PLE १९९०/CD मा/POP संयुक्त/JJ राज्य NN/ अमेरिका/NNP मा/POP यस/DUM को/PKO बिक्री/NN १२००/CD मा/POP स्थिर/JJ हुने/VBNE अपेक्षा/NN राखेको/VBKO बतायो/VBF ।/YF पछिल्लो/JJM वर्ष/NN सुबिधा/NN सम्पन्न/JJ गाडी/NN निर्माता/NN ले/PLE संयुक्त/JJ राज्य/NN अमेरिका/NNP मा/POP १२१४/CD वटा/CL कार/NN हरू/HRU बिक्रि/NN गर्यो/VBF ।/YF अध्यक्ष/NN तथा/CC प्रमुख/JJ कार्यकारी/NN अधिकृत/NN होवार्डमोशर/NN ले/PLE उहाँ/PP ले/PLE बेलायत/NNP तथा/CC युरोप/NNP र/CC सुदूर/JJ पूर्वी/JJ बजार/NN हरू/HRU मा/POP सुबिधा/NN सम्पन्न/JJ गाडी/NN निर्माता/NN हरू/HRU **कालागि/VBO** संवृद्धि/NN को PKO/आशा NN/गरेको VBKO/बताउनुभयो VBX/।/YF बेल/NNP औद्योगिक/JJ संस्था/NN ले/PLE यस/DUM को/PKO त्रैमासिक/JJ लाभांश/NN मा/POP एक/CD सेयर/NN मा/POP ७/CD सेन्ट/NNP बाट/POP १०/CD सेन्ट/NNP मा/POP बढायो/VBF ।/YF

SVM Tagger

सुश्री/NN हाग/NNP एलियान्टी/NNP को/PKO भूमिका/NN खेल्नुहुन्छ/VBF ।/YF रोल्स-रोयस/NNP मोटरकार/NN इन्कर्पोरेटिड/NN ले/PLE १९९०/CD मा/POP संयुक्त/JJ राज्य/NN अमेरिका/NNP मा/POP यस/ DUM को/PKO बिक्री/NN १२००/CD मा/POP स्थिर/JJ हुने/VBNE अपेक्षा /NN राखेको/VBKO बतायो/VBF ।/ YF पछिल्लो/JJM वर्ष/NN सुबिधा/NN सम्पन्न/JJ गाडी/NN निर्माता/NN ले/PLE संयुक्त/JJ राज्य/NN अमेरिका/NNP मा/POP १२१४/CD वटा/CL कार/NN हरू/HRU बिक्रि/NN गर्यो/VBF ।/YF अध्यक्ष/NN तथा/CC प्रमुख/JJ कार्यकारी/NN अधिकृत/NN होवार्डमोशर/NNP ले/PLE उहाँ/PP ले/PLE बेलायत/NNP तथा /CC युरोप/NNP र/CC सुदूर/JJ पूर्वी/JJ बजार/NN हरू/HRU मा/POP सुबिधा/NN सम्पन्न/JJ गाडी/NN निर्माता/NN हरू/HRU **कालागि/POP** संवृद्धि/NN को/PKO आशा/NN गरेको/VBKO बताउनुभयो/VBX ।/YF बेल/NNP औद्योगिक/JJ संस्था/NN ले/PLE यस/DUM को/PKO त्रैमासिक/JJ लाभांश/NN मा/POP एक/CD सेयर/NN मा/POP ७/CD सेन्ट/NNP बाट/POP १०/CD सेन्ट/NNP मा/ OP बढायो/VBF ।/YF

CHAPTER 5

TESTING AND ANALYSIS

5.1 Nepali Tagged Corpus Data Statistics

The tagged Nepali corpus named FinalNepaliCorpus which contains 111475 Unicode Nepali words is used for training and testing. This corpus is manually tagged with the 43 tags as described in [21]. This is a text file with the “word <tag>” as entry in file. For example:

६१<CD> वर्षीय<JJ> पियरे<NNP> भिन्केन<NNP> नोभेम्बर<NNP> २९<CD> बाट<POP> सल्लाहकार<NN>को<PKO> रूप<NN> मा<POP> सञ्चालक<NN> समिति<NN>मा<POP> आउनुहुनेछ<VBX> ।<YF>

5.2 The Dictionary Data Statistics

The dictionary used in the experiment contains the following statistics about ambiguous and unambiguous words. Here the word represents the unique word found in the Nepali Corpus.

Unambiguous words	11676
Ambiguous words	792
Total words in dictionary	12468

Table 5.1: The Word Distribution on Dictionary

In comparisons to wall street journal (WSJ) corpus which contains 44526 (5985 ambiguous + 38541 unambiguous) unique words, it is smaller which limit the performance of the tagger we have implemented then the same tagger for English.

5.3 Test Data Analysis

Test data is prepared form the original corpus. It consists of 10775 tokens from the original corpus. This part of data is not used during training period. Since the count of tokens should be in whole number, some consideration has made about the percentage of testing data to make it whole number. The test data contains total of 10775 randomly selected tokens out of which 82% are unambiguous, 13% are unknown tokens and 5% are of ambiguous. This is shown in the following pie chart.

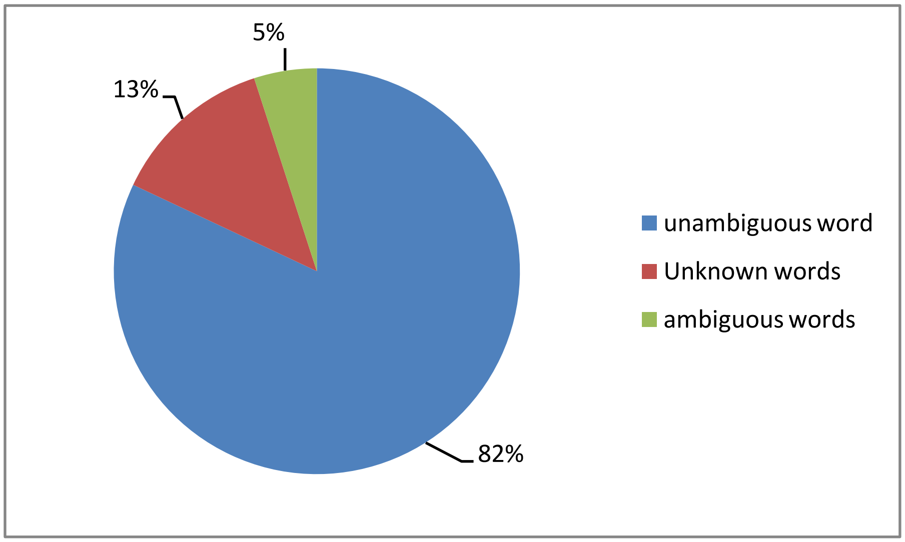

Figure 5.1: Statistics of sample test

Accuracy measurement of tagger

The accuracy is measured with matching two file: one is test file and another is manually tagged test gold file. The correctly tagged tokens are those which match in both files and the remaining tokens are tagged incorrectly. For this a matcher program is written which sequentially reads the both file and match the line by line and increment the count if both line matched in both files.

On the test file of 11147 *words, the overall accuracy is calculated as*

$$accuracy = \frac{Total\,word\,correctly\,tagged}{total\,word\,in\,test\,file} = \frac{10050}{10775} = 93.27\%$$

The detail comparison of tagger performance with ambiguous and unambiguous word is tabulated below (In table 5.2).

	No of tokens	Accuracy	Error
ambiguous word	538	490/538 (91.07%)	48/538(5.88)
Unambiguous words	8819	8290/8819(94.01%)	265/8819(1.07%)
Unknown words	1418	1270/1418(89.56)	141/558(9.94%)

Table 5.2: Unknown and Known word accuracy

Validation and Evaluation

Cross validation technique is used to validate the measured accuracy of tagger. The general k-cross validation is a technique which divides the whole corpus into ten parts and nine part(90%) of data is used for training and remaining one part is used for testing. The process is repeated for ten times taking each of ten parts as testing instances.

Here the 10-cross validation is adapted in which the whole corpus is divided into 10 portions sequentially and in each iteration of program, the 9 folds are used for training and the remaining 1 fold is used for testing.

And in other respect, the learning nature of tagger is evaluated with the different size of training data. The size of training data is gradually increased and the performance of tagger is observed. The result so found is presented in the table 5.3.

Training data size (in tokens)	Accurcy(%)	
	SVM	HMM
10000	71%	61%
20000	79%	69%
40000	85%	72%
80000	90%	90%
100000	92%	91%

Table 5.3: The tagging accuracy for different training size

The corresponding learning curve is presented below

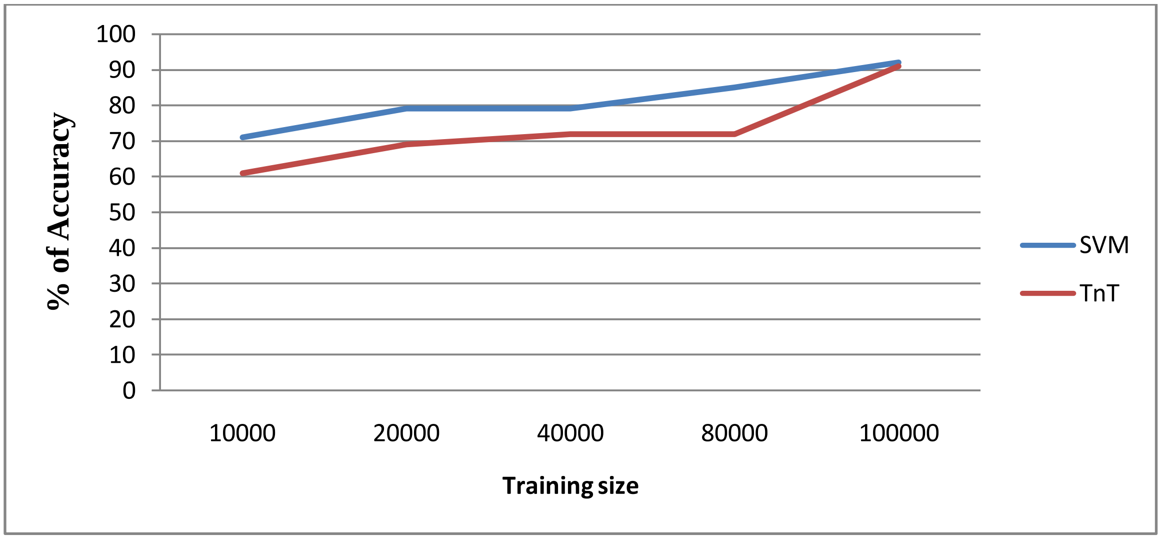

Figure 4.2: Learning Curve for Tagging Nepali

The learning curve shows the gradual increment in accuracy for the large size of training data for the HMM tagger and it performs not very well for the small set of training data since the bigram probabilities are not found sufficiently in the case of small size data and most of the case it becomes zero. But the SVM tagger does not depends upon the probabilities rather it depends upon the features extracted from the training and testing data. So it performs very well for the small training data size as well as for large data size.

5.4 Results

The following table shows the performance of different tagger.

	Accuracy		
Taggers	**Known words**	**Unknown word**	**Over all**
TnT	92%	56%	74%
SVM	96.48%	90.06%	93.27%

Table 5.4: Overall Result of Different Taggers

The overall accuracy is slightly better for SVM then other two taggers as shown table 5.4. This is because of rich pattern set provided for SVM. TnT perform well for known word but it gives lower accuracy for unknown word, it is because there is no mechanism to deal with unknown word for Nepali language.

CHAPTER 6

CONCLUSION AND FURTHER RECOMMENDATION

6.1 Conclusion

The Support Vector Machine (SVM) based part-of-speech tagger is built which assigns the most appropriate part-of-speech tag to each of word of the Nepali text. In this work, the study has gone through the empirical analysis of the performance of the tagger for morphologically rich and order free language like Nepali. It is observed from the experiment that the tagger based on SVM outperform the other two taggers: First order HMM and TnT. The SVM based tagger incorporate diverse set of feature such as word features, POS features and other features, while the other tagger rely on only local features such as current word and one or two previous word. Here, during the development of the model, the impact of the size of training data and test data on the performance was observed. From tests made for various sizes of training data, it had shown that the performance of the tagger depend upon the size of the training data, as well as number of tokens that are present and absent in the training data. Here, in this dissertation, the average overall accuracy of this tagger for morphologically rich and order free language –Nepali is 93.27%.

6.2 Further Recommendation

In this dissertations, the SVM based POS tagger is built which uses the dictionary as a primary resources. This dictionary is collected from the FinalNepaliCorpus which contains only 11147 unique words. The performance of tagger is dependent on this dictionary and so in future; such dictionary may be built using the information on news sources and available Nepali raw text with the help of morphological analyzer and part of speech acquisitions techniques.

The limitation of the SVM tagger built is the speed. It is found to be slow in training than other tagger. Since the SVM based POS tagger uses the different set of features to construct the feature vectors, the empirical analysis to find the optimal set of features may be the future work which may concentrate on speed optimization of tagger.

REFERENCES

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

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

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

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

[5] 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.

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

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

[8] 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.

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

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

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

[12] 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).

[13] 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.

[14] D. Jurafsky, J.H. Martin, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, University of Colorado, Boulder,(Pearson Education, 2000), pp 343.

[15] S. Klein, RF. Simmons, *A Computational Approach to Grammatical Coding of English Words,* Journal of the Association for computing machinery (JACM), 1963 pp 334-347.

[16] G. Leech, and N. Smith, *The Use of Tagging*. In: van Halteren (1999a), 1999.

[17] L. M`arquez, H. Rodr´ıguez, J. Carmona, and J. Montolio. *Improving POS Tagging Using Machine-Learning Techniques.* In: Proceedings of EMNLP/VLC,1999.

[18] B. Merialdo, *Tagging English Text with a Probabilistic Model*, Computational Linguistics, 20 (2), 1994.

[19] Masaki Murata, Qing Ma, Hitoshi Isahara, *Comparision of Three Machine Learning Method for Thai Part of Speech Tagging,* In: Proceeding of ACM Transactions on Asian Language Information Processing (TALIP-2002), 2002.

[20] T. Nakagawa, T. Kudoh, and Y. Matsumoto, *Unknown Word Guessing and Part-of-Speech Tagging using Support Vector Machines*, In: Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, 2001.

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

[22] Lawrence R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* In: Proceeding of the IEEE, 1989.

[23] GM. Rubin, BB Greene, *Automatic grammatical tagging of English*. Providence, Rhode Island: Brown University Department of Linguistics, 1971.

[24] Helmut Schmid, *part of speech tagging with neural network,* Institute of computational linguistic Germany

[25] Valdimir Vapnik, Corinna Cortes, *Support vector networks*, machine learning 20, 273-297, Kunwer Acedemic Publisher 1995.

Bibliography

Ethem Alpaydın, Introduction to Machine Learning, (Second Edition, 2010, The MIT Press, Cambridge, Massachusetts London, England).

Lutz Hamel, Knowledge Discovery with Support Vector Machine (John Wiley and Sons Inc., New Jersey, USA, 2009)

APPENDIX

SAMPLE PROGRAM CODE

1. Source program that implement the Algorithm for tagging (Perl source code)

```
#read the test file into the testexamples array
$i=0;
open (FH, '<', $testfile) or die "Cannot open '$testfile': $!";
while(<FH>){
        chomp;
        ($testexamples[$i]{word})=(/(\S+)/);
        $i++;
        }
close FH;
#build the feature vector for each test example.
my $ntestexamples= scalar (@testexamples);
my $k=0;
for (my $i=0; $i < $ntestexamples; $i++){
#add word features in the hash
   my $dictindex=search_dictionary($testexamples[$i]{word});
  #print "testexamples[$i]{word} => $testexamples[$i]{word}\n\n";
  if($dictindex == -1){
      print "Couldn't find dicionary entry for the word $testexamples[$i]{word} ...skipping the
    instance\n";
     next;
    }
# evaluate for all part of speech tags and assign the most confident tag
```

```
My    @possiblePOS=extract_possible_tags($dictionary[$dictindex]{tags},
                                                  dictionary[$dictindex]{ntags});
# extract features of the ith instance and returns a feature vector
my @features = extract_features($i,"test",\@testexamples);
my $testinstance="0 ";
my $m=0;
my %FEATURES;
my @uniquefeatures;

foreach my $f (@features){
$FEATURES{$f} = "$f";
}

foreach my $f (keys %FEATURES) {
                $uniquefeatures[$m++]= $FEATURES{$f};
                }

$testinstance="$testinstance".join( ":1 ",sort { $a <=> $b }(@uniquefeatures)).":1";
print "testexamples[$i]{word} => $testexamples[$i]{word}\n\n";
print join(" ",@features);
print "\n\n";

$testinstance="$testinstance".join( ":1 ",sort { $a <=> $b }(@features)).":1";
my $testinstancefile="tmp.test";
   open (FH, '>', $testinstancefile) or die "Cannot open '$testinstancefile': $!";
```

```
  print FH "$testinstance\n";
 close FH;
my $confidenttag;
my $score=-100;
foreach my $ppos (@possiblePOS){
            my $modelsubstring=$ppos;
            if($ppos !~ /[A-Z]+|(a-z)+/){
            $modelsubstring="$sTag{$ppos}";
            print "modelsubstring => $modelsubstring\n\n";
}

my $modelfile="$modeldir"."$model.$modelsubstring.$svmext";
print "$modelfile";
system("svm_classify -v 0 $testinstancefile $modelfile") == 0 or die "system failed: $?";
open (FH, '<', "svm_predictions") or die "Cannot open 'svm_predictions': $!";
while(<FH>){
chomp;
            if( $_ > $score){
                        $score=$_;
                        $confidenttag=$ppos;
                        }
            }
close FH;
}
$testexamples[$i]{pos}=$confidenttag;
```

```
print "word => $testexamples[$i]{word}\t pos => $testexamples[$i]{pos}\n\n";

}

        open(FH, '>', 'outputtagedfile.txt');

        for (my $l=0; $l < $ntestexamples; $l++){

                print FH "$testexamples[$l]{word} $testexamples[$l]{pos}\n";

                }

        close(FH);
```