An Online Semi Automated Part of Speech Tagging Technique Applied To Assamese

A thesis submitted for the award of the degree of

Doctor of Philosophy

by

Pallav Kumar Dutta

Under the supervision of

Prof. Gautam Barua

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM, INDIA

December, 2013

S T A T E M E N T

I do hereby declare that the thesis entitled **"An Online Semi Automated Part of Speech Tagging Technique Applied to Assamese"**, *is a bonafide work based on the investigation results carried out by me in the department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India, under the supervision of* **Prof. Gautam Barua**.

Due Acknowledgements have been made to the other researchers, whenever the work described is based on on their findings or results.

Pallav Kumar Dutta

December, 2013

C E R T I F I C A T E

This is to certify that the work contained in this thesis entitled "**An Online Semi Automated Part of Speech Tagging Technique Applied to Assamese**" submitted by **Pallav Kumar Dutta**, *a research student in the department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for the award of the degree of Doctor of Philosophy, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute. The results embodied in this thesis have not been submitted to any other university for the award of an degree or diploma.*

Gautam Barua
Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Assam- 781039, India.

December, 2013
Guwahati

Acknowledgements

Before going deep into the report, I wish to express my sincere gratitude to my advisor, Prof. Gautam Barua who inspired me with the idea of this project and has been a constant source of motivation. I will remain ever grateful to him for his guidance. He has given me the time whenever I needed his input and guidance despite his busy schedule.

I am indebted to the members of the doctoral committee for their valuable suggestions.

I would like to thank the faculty members and technical staff of the department of Computer Science & Engineering, specially Prof. Sukumar Nandi for his constructive critism and Dr Shakuntala Mahanta, Asst Prof, department of Humanities and Social Sciences, IIT Guwahati.

I appreciate all the staff of the Computer & Communication Centre, IIT Guwahati, Ms Sugandha Kaur and Aleendra Brahma of RCILTS project for helping me out in every possible way.

I express my thanks to all the persons who had co-operated with me to carry out the different experiments and provided the valuable suggestions for the interfaces and helped me during the creation of corpora.

Lastly, I am obligated to my parents and family for their unweavering support and encouragments and bearing with me during these long years of the research work.

Abstract

Developing annotated tagged corpora for a language with limited electronic resources can be very demanding. Although Assamese is a language spoken by about 15 million people in the Indian state of Assam as a first language, the development of electronic resources for the language has been lagging behind other Indian languages. Also, there has not been much work done in POS tagging for Assamese.

In order to fill this gap, we have designed a POS Tagger for Assamese. Our approach is to use a combination of methods to try and get good results. Further, we amortise the manual intervention over the period of tagging rather than doing all manual work at the beginning. This allows us to quickly start the tagging system. But it also means that what we have is a semi-automatic tagger and not an automatic tagger. Our method requires only native speakers' intervention in stages other than the beginning making the system amenable to some form of "crowd-sourcing" with a few experts for moderation. This will enable our system to create very large tagged corpora in the language.

We first create a knowledge base using a number of methods. This knowledge base is then used to automatically tag sentences in the language. This tagging uses a combination of stemming, application of a few grammatical rules, and a bigram tagger. The tagged sentences are then shown to non-expert native speakers for verification and correction. Before starting the actual tagging process, the knowledge base was tuned by examining the results on small data sets using experts instead of native speakers. The design of a user friendly interface plays an important role in reducing the time taken by native speakers in their examination.

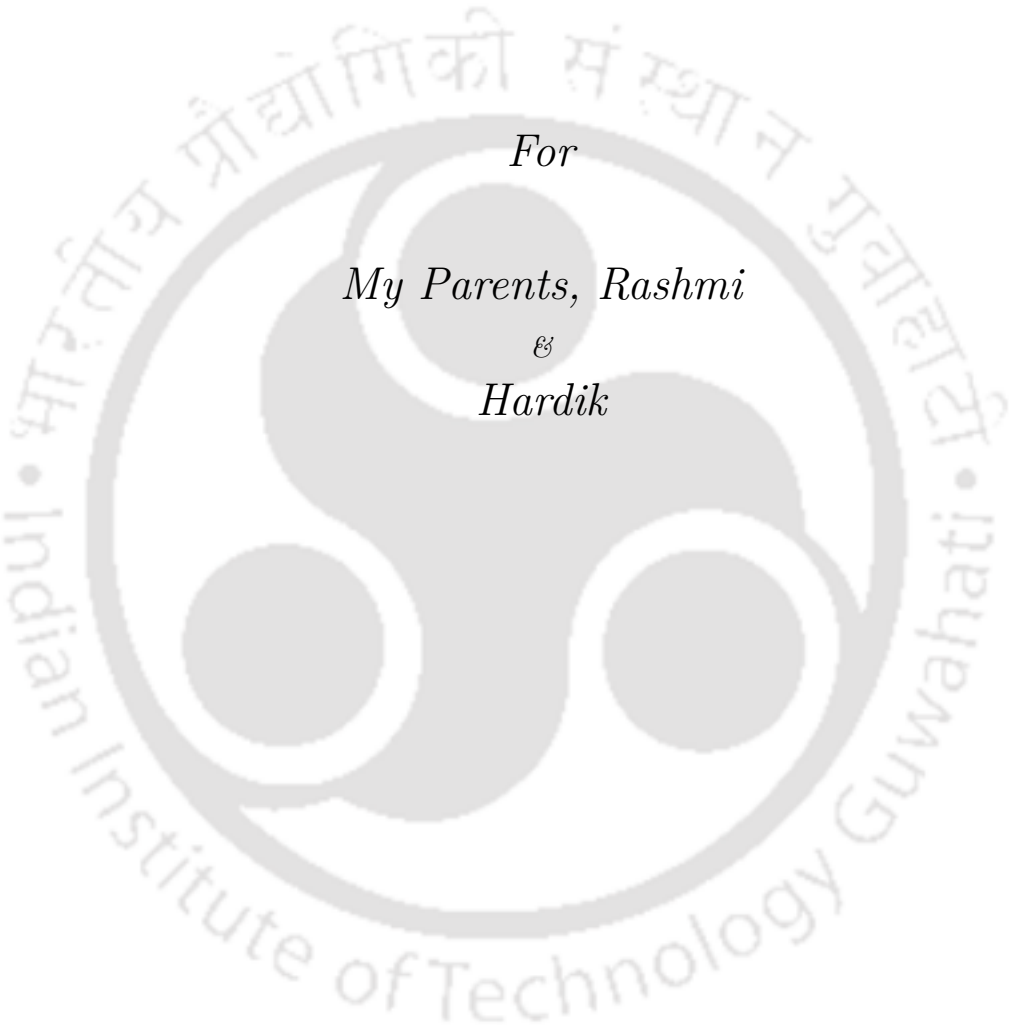
Given a base tagged corpus, a database of tagged words was created. Tables containing root words, prefixes, and suffixes were created. Grammar specific rules

were added to handle exceptions. A computer programme was developed which tags new text by looking up these tables. Ambiguities in the output are then removed by the use of a bigram table. This table was created using the same base tagged corpus. At the final stage, a native, non-expert speaker examines the machine tagged output and makes corrections as deemed fit.

The accuracy of the scheme has been estimated by portions of the tagged corpus being further checked by an expert. The errors found by the expert are used to estimate the accuracy of the main process.

An online Assamese tagged corpus of more than 5,50,000 words has been generated and is available for research purposes. Another 2,50,000 words have already been tagged and it is proposed to use crowd sourcing on the net for the native users' stage.

The tagging of 5,50,000 words yielded an accuracy of 96.27%. The fidelity (where some of the tagging may be coarse) obtained was 98.39%. Even without human intervention, the system accuracy obtained was 95.91%, and with coarse tagging allowed, the system accuracy rose to 98.25%. These results compare with the best results reported in the literature for any language. These results are better than any result report so far for Indian languages.



For

My Parents, Rashmi

&

Hardik

Bibliographic Notes

Portions of the dissertation have appeared elsewhere.

Chapter 3 "Assamese Language" is based on the documents published by Resource Centre for Indian Language Technology Solutions (RCILTS), Indian Institute of Technology Guwahati and Language Information Service (LIS) India, CIIL, Mysore, on Assamese Language ([66], [67], [68]).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Part of Speech (POS) tagging assigns one of the parts of speech to a given word of a sentence based on both its definition, as well as its context. POS tagging is a basic step in analyzing sentences of a language. It is one of the most studied problems in the field of Natural Language Processing which is an interdisciplinary field of Linguistics and Computer Science to process natural language text so that a computer can understand the text and also to generate text in a language via translations or some other means. Most of the techniques requires a large corpus of text in the language to be analysed. So the first step has been to create large corpora in different languages. The first major corpus of English was the Brown Corpus ([3]), developed in the year 1960. Lancaster-Oslo-Bergen Corpus (LOB) ([4]) was compiled in 1970s as a British counterpart to the Brown Corpus. The British National Corpus ([5]) is a hundred million word text corpus created during 1991-94. The German Reference Corpus ([6]) was first created in 1964 and currently comprises more than 4.0 billion word tokens.

POS tagging helps in the creation of annotated corpora, where the major annotations are the parts of speech. Such corpora can then be used for information retrieval, text-to-speech conversion, word sense disambiguation, machine translation, etc. It has been observed that sometimes the morpho-syntactic behavior of a word is lost after POS tagging, but this can be reduced by properly choosing the

tag set for POS tagging of the language. Developing annotated tagged corpora for a language can be done by humans or it can be done through software. Human tagging is time consuming and is also prone to error. It requires highly skilled persons of the specific language. Tagging using software results in a lot of errors. To reduce errors, the software needs to be trained. Unfortunately this training itself requires a lot of time of language experts. The attempt therefore is to speed up the tagging process, while at the same time keeping errors at a low level.

1.1 Part of Speech Tagger

"A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'" ([2]). Automatically assigning a proper POS tag to a word in a sentence is a challenging task and majority of the research has been done in languages like English and other western languages.

There are different POS taggers available for different languages based on different approaches and models. CLAWS ([7]), the Constituent Likelihood Automatic Word-tagging System for English, has been continuously developed since the early 1980s. The latest version of the tagger, CLAWS4, was used to POS tag the British National Corpus (BNC). CLAWS recently developed a web based service for POS tagging which is in a testing phase. The English Stanford tagger was first released in 2004. This model also contains Arabic, Chinese and Arabic trained models in addition to English. Centre for Language Technology at the University of Copenhagen have developed an online POS tagger based on Brill tagging ([8]). Cognitive Computation Group, University of Illinois has also published an on line POS tagger ([9]).

1.2 Different Approaches of POS tagging

There are different approaches and models for POS tagging, and these can be broadly categorised as Supervised and Unsupervised. Supervised POS tagging models require previously tagged corpora which are used for training. So, the performance increases with the increase of the size of the tagged corpora. The Unsupervised approach does not require a pre-tagged corpus. This approach uses various algorithms to determine the tag set.

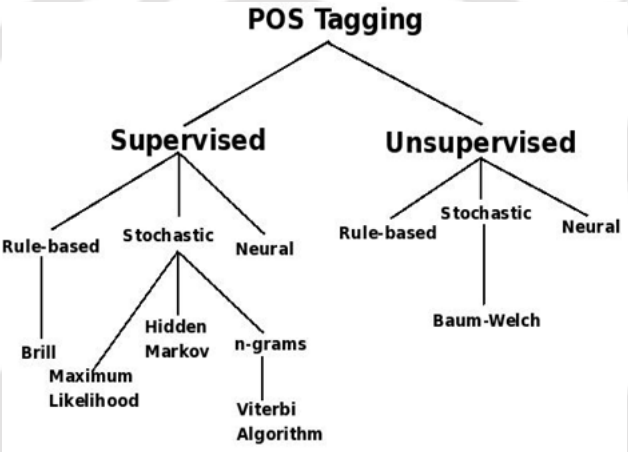


Figure 1.1: Different Approaches of POS Tagging

Both of them again can be divided into the following types: rule-based approach and statistical /stochastic approach.

1.2.1 Rule-based approach

Rule-based approaches are early approaches to POS tagging. The rule based approaches use contextual information and disambiguation rules to assign tags to unknown or ambiguous words. These rules are often known as context frame rules. Many rule based taggers use morphological information in the disambiguation process. Capitalization and punctuation information has also been used for rule based taggers for languages like German. Rule based taggers require pre-annotated corpora. Automatic rule induction technique has also been tested. In

this approach the untagged text is run through a tagger and the initial output is manually verified to correct the erroneously tagged words. The modified text is then again submitted to the tagger and the tagger learns the correction rules by comparing the two data sets. In this approach several iterations of the process may be required to achieve considerable performance. Brill Tagger ([10]), developed by Eric Brill is the most widely used rule-based tagger. A snapshot of the brill tagger using NLTK is shown in chapter 4.

1.2.2 Stochastic approach

The stochastic approach mainly uses the frequency of occurence as the probability of occurrence to select the most likely tag. The tag most frequently encountered for a word in the training corpus is assigned to an ambiguous instance of that word ([11]). Another alternative approach to word frequency is known as the n-gram approach. In this approach the probability of contiguous sequence of n tags is calculated and the best tag for a word is assigned by calculating the probability that it occurs with the n previous tags. Unigram model ignores the context whereas the Bigram and Trigram models add one and two words of context respectively. In case of bigrams, the conditional probability of a token given the preceding token can be calculated as

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1},W_n)}{P(W_{n-1})}$$

That is, the probability P() of a token W_n given the preceding token W_{n-1} is equal to the probability of their bigram, or the co-occurrence of the two tokens $P(W_{n-1},W_n)$, divided by the probability of the preceding token.

In n-gram models the number of parameters required grows exponentially with the number of words of prior context. As n increases, the power (expressiveness) of an n-gram model increases, but the ability to estimate accurate parameters

from sparse data decreases. Many times local context does not provide the most useful predictive clues, which instead are provided by long-distance dependencies. In this approach one incorrect tagging choice might have cascading effects.

A number of statistical approaches are available like Hidden Markov Model (HMM), Maximum Entropy Model (MEM), Conditional Random Field (CRF), Memory Based learning (MBL) etc.

Hidden Markov Model

A Markov model is a finite-state machine. Each state has two probability distributions: the probability of emitting a symbol and probability of moving to a particular state. From one state, the Markov model emits a symbol and then moves to another state. The objective of a Markov model is to the find optimal sequence of tags T = {t_1,t_2, t_3,...t_n} for the word sequence W = {w_1,w_2, w_3,...w_n}. That is, it is to find the most probable tag sequence for a word sequence. The joint probability P(T,W) of the tag sequence and the word sequence is decomposed into a product of conditional probabilities. It is assumed that the next part of speech depends only on the k preceding POS tags and that the next word depends only on its part of speech. These assumptions are called Markov assumptions and the resulting statistical model is HMM. Each state of the HMM model corresponds to a k-tupel <t_1,t_2, ...t_k> of POS tags. The value of k used in POS tagging are usually 1 or 2 (bigram or trigram tagger).

In HMM the state transitions are not observable. HMM taggers require only a lexicon and untagged text for training a tagger. Hidden Markov Models aim to make a language model automatically with little effort. Disambiguation is done by assigning the more probable tag. One can calculate the probability that the model produced the sequence, as well as which state sequence was most likely to have produced the observations.

The best POS tag sequence for the word sequence W is efficiently computed by the Viterbi algorithm. It is a dynamic programming algorithm for finding the most likely sequence of hidden states that result in a sequence of observed events.

Baum-Welch algorithm is also used to train a Markov model where manually annotated corpus is not used to train the model.

Maximum Entropy Model

The maximum Entropy Model (ME) is based on the principle of maximum Entropy. It is also known as the log-linear model. These taggers use more information like information about surrounding words and less number of parameters compared to HMM taggers. To determine the POS of a word in a given sentence involves many factors. Assume that random variable x represents these factors and random variable y represents the POS, and then p(y|x) represents the probability of a word with POS y in the given sentence given the factors x. The probability p(y|x) in the ME model is required to satisfy some constraints and must maximize the entropy defined as following:

$$H(p) = -\sum_{x,y} p(y|x) \log p(y|x)$$

Here constraints are all the known cases and can be represented by

$$f_i(x, y) = \begin{cases} 1, & \text{if } \{x,y\} \text{ satisfying restrictions} \\ 0, & \text{else} \end{cases}$$

$f_i(x,y)$ is called the feature of the ME model. These features described the relations between the contextual factors x with the POS y.

HUANG Heyan and ZHANG Xiaofei showed that ME-based taggers perform better than HMM taggers. While comparing HMM and ME taggers on the same corpus, they have reported that POS tagging error rate is reduced by 54.25% in close test and 40.56% in open test over the HMM-based baseline, and synchronously an accuracy of 98.01% in closed test and 95.56% in open test are obtained ([12]).

Conditional Random Fields

Conditional models are used to label a novel observation sequence x by selecting the label sequence y that maximizes the conditional probability p(y|x). The conditional nature of such models means that no effort is wasted on modeling the observations and one is free from having to make unwarranted independence assumptions about these sequences; arbitrary attributes of the observation data may be captured by the model, without the modeler having to worry about how these attributes are related.

Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting sequential data, based on a conditional approach. A CRF is a form of undirected graphical model that defines a single log-linear distribution over label sequences given a particular observation sequence. The primary advantage of CRFs over hidden Markov models is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. Additionally, CRFs avoid the label bias problem, a weakness exhibited by maximum entropy Markov models (MEMMs) and other conditional Markov models based on directed graphical models. CRFs outperform both MEMMs and HMMs on a number of real-world sequence labeling tasks ([13]).

Memory Based Learning

Memory-based learning is founded on the hypothesis that performance in cognitive tasks relies rather on reasoning on the basis of similarity of new situations to stored representations of earlier experiences, than on the application of mental rules abstracted from those earlier experiences. An MBL system contains two components: a learning component which is memory-based, and which is sometimes called 'lazy' as memory storage is done without abstraction or restructuring, and a performance component which does similarity-based classification. The MBT tagger generator takes an annotated corpus as input, and produces a lexicon and memory-based POS tagger as output. During tagging, each word in the text to be tagged is looked up in the lexicon. If it is found, its lexical representation is retrieved and its context is determined, and the resulting pattern is disambiguated using extrapolation from nearest neighbours in the known words case base. When a word is not found in the lexicon, its lexical representation is computed on the basis of its form, its context is determined, and the resulting pattern is disambiguated using extrapolation from nearest neighbors in the unknown words case base. In each case, the output is a best guess of the category for the word in its current context.

Jakub Zavrel, Walter Daelemans showed that MBT performs at state-of-the-art levels with an accuracy of 97%, providing better generalization accuracy than two widely-used methods: trigram tagging and transformation-based tagging ([14]).

Memory Based Learning technique is one of the methods of Classification-Based approaches of POS tagging. Other classification algorithms applied to POS tagging are neural networks, support vector machines (SVMs), decision trees, the Window algorithm etc ([15]).

1.3 The Problem & Our Approach

A large annotated tagged corpus for a language with limited electronic resources can be very demanding but it is a great source of NLP research for that language. The whole process requires a lot of human input, time and is also prone to error. The process requires highly skilled persons of the specific language. Very few corpora of Assamese are publicly available and most of them are motivated by specific research agenda. Not much work has been done in the natural language processing field for Assamese language and other North Eastern Indian languages. There is hardly any publicly available corpus or any other NLP tools on the internet for these languages.

In order to fill this gap, we have tried to develop a POS Tagger for Assamese language as most of the other NLP activities depend on POS tagging. Language experts manually handled stemming, morphological analysis and tagging of words in a base corpus and words so tagged were stored in a database which also contains a list of affixes, a list of root words, and other such information. The list of root words was obtained from a dictionary of words that was manually created earlier. This database is then used as a resource in the second stage where a program finds the root word of a given word to be tagged. It checks if the word is in the base corpus or if the root word is in the list of root words. If so, then the appropriate tag is given. If no tag can be given, then a default tag is given. Some words may get more than one tag in this stage. So the next stage uses a bigram frequency table (built from the base corpus) to disambiguate such cases. Exceptions are dealt with using a set of language specific rules developed by experts. In the third stage, a body of native speakers who are not experts in linguistics, check the tagged corpus and make corrections. So our approach is a mixture of the various approaches described in the literature, with the use of native speakers in the last stage allowing us to use “crowd sourcing” techniques to speed up the tagging process.

While this use of humans to assist in the tagging may seem that it is not going to be of much use, we have observed that providing simple aids to humans improves their productivity tremendously. Our system therefore lays stress on developing a good user interface.

1.3.1 Dissertation Outline

The dissertation consists of the following chapters.

Chapter 2: This chapter reviews the various approaches and methodologies adopted for POS tagging in Indian languages and a few other languages. The various stemmers developed for Assamese have also been discussed. This chapter also throws light on the crowd sourcing approach implemented on some resource-scarce languages to develop language resources.

Chapter 3: This chapter provides a brief overview of the Assamese language, its features and uniqueness, phonology, morphology and grammar etc. It also describes the Assamese corpus already available, and the tagset used for tagging in our various experiments.

Chapter 4: In this chapter we have compared some basic probabilistic POS tagging techniques with the Assamese language corpus using the Natural Language Toolkit (NLTK). We have tested the n-gram, Backoff and Brill tagger modules with parts of our corpus as input and tabulated the results of the experiments.

Chapter 5: This chapter describes the creation of the knowledge base for our approach. This chapter describes about the approach that we have adopted, the types of suffixation, the suffix striping method to generate the root word from a word and some computational linguistics grammatical rules for the Assamese language etc. The various database tables used in the approach have also been

discussed.

Chapter 6: This chapter describes the implementation details of our approach-the evaluation procedure, algorithms, tuning of the tagger, the User Interface developed for the native users for providing verification.

Chapter 7: This chapter describes the analysis of various experimental results-correctness and quality of the newly generated tagged corpus, annotation time analysis etc.

Chapter 8: This chapter summarizes the work describes in the thesis and provides direction for future work in the area.

Chapter 2

POS Tagging: Different Languages & Approaches

In this chapter we describe the major approaches used for POS tagging different languages. In section 2.1 we have listed the various approaches used in the English language. Following this we have mentioned methods for some more foreign languages like Hungarian, Russian etc.

In section 2.3 we have mentioned different works carried out on Indian languages.

2.1 English Language

The most widely experimented language for POS tagging in the world is English. A corpus is the basic requirement for POS tagging and the Brown Corpus created in mid 1960s for English is one of the most widely used (1 million words from 500 different texts). There are many more large corpora in English. The largest is the British Natioanl Corpus (100 million words). Brown corpus has been tagged over many years. In the mid 1980s, European researchers began to use the Hidden Markov Model for POS tagging. CLAWS tagger is one of the ear-

liest tagger developed based on HMM. Later versions of the tagger have achieved 96-97% accuracy on the BNC corpus. Brants, Thorsten developed the TnT tagger based on HMM and achieved 96.46% accuracy on all tokens ([16]).

Dynamic programming algorithms were used for POS tagging around 1987 using the Brown corpus. The dynamic programming methods were similar to the Viterbi algorithm, used in other fields, and thus people started using this algorithm in POS tagging. Eric Brill developed the rule based Brill tagger and showed that the approach performs as well as the statistical techniques.

Machine learning methods have also been tested for better accuracy. SVM (Support Vector Machine)-based tagger developed at TALP Research center NLP group achieves an accuracy of 97.2% for English on the Wall Street Journal (WSJ – 30 million words – overlaps the PTB corpus) corpus ([17]). Conditional Random Fields POS tagger (CRFTagger) developed by Xuan-Hieu Phan achieved 97% accuracy on the WSJ corpus ([18]).

Kristina Toutanova, Dan Klein, Christopher D. Manning and Yoram Singer achieved 97.24% accuracy on the PTB corpus using previous and next tag contexts through cyclic dependency network and use of lexical features ([19]).

Yoshimasa Tsuruoka, Yusuke Miyao, Jun'ichi Kazama have shown that combining history based models with lookahead approach can increase the accuracy of POS taggers and have achieved around 97.19% ([20]).

Helmut Schmid has proposed a new probabilistic tagging method using a decision tree. A Tree Tagger has been implemented with an accuracy of 96.36% on PTB data. To avoid the problems of estimating the transition probabilities using HMM based taggers, a decision tree has been used in this approach ([21]).

2.1.1 Issues with English Corpora

The corpora available in English language are mainly domain specific. The accuracy becomes very high when the trained and test sets are of the same domain. But when tested in different domains the accuracy decreases drastically. Lois Boggess et al. described that a small domain specific corpus of size only 20,000 tokens might suffice to outperform a tagger trained on one of the large publicly available corpora ([22]).

Though English language has the largest corpus, no POS tagging method, trained on such large coproa, could ever achieve 100% accuracy. Human annotators have to be used to manually post process the tagged corpus to get the highest level of accuracy.

The morphological features of English language are restricted by tagsets like the Penn Treebank tagset used in various supervised learning approaches for English ([23]). This leads to major problems during the creation of parallel annotated corpora of morphologically rich languages.

2.2 Other Foreign Languages

We have found some more approaches that have been applied to languages like Hungarian, Myanmarese, Indonesian, German & Russian languages.

Péter Halácsy, Andras Kornái et. al. had compared the performance of several POS tagging architectures for the Hungarian language and presented a hybrid tagging architecture that incorporates a weighted morphological analyzer (WMA) in the maximum entropy framework. The output of WMA module is pruned by the Viterbi algorithm ([24]).

Phyu Huninn Myint, Tin Myat Htwe and Ni Lar Thein describe a two phase bigram POS tagger using supervised learning approach for Myanmarese. They have used a pre tagged corpus trained with a HMM model using the Baum-Welch algorithm and then used the Viterbi tagging algorithm to find the most likely tag sequences. Then they have applied lexical rules to normalize some words and tags in order to produce accurate and finer tags ([25]).

Stéphane Bressan, Lily Suryana Indradjaja have proposed few methods for fully automatic acquisition of the knowledge necessary for POS tagging of Indonesian corpora using n-gram method and sentence splitting, clustering and extended Schutze's approach. They have also evaluated the methods using the Brown corpus and the Indonesian language corpus ([26]).

Bocharov V. V., Alexeeva S. V., Granovsky D. V., Protopopova E. V., Stepanova M. E. and Surikov A. V. have developed a one million word corpus of Russian texts in the OpenCorpora project using crowd-sourcing ([27]). They have used native speakers with no linguistic knowledge for morphological annotation of each word of a sentence using a set of simple annotation questions leading to a decision tree and quality was estimated by disagreement rate among several annotators.

Helmut Schmid modified the tree tagger based on HMM approach for German using suffix and prefix lexicons, equivalent class probabilities and additional word-tag pairs from an automatically tagged corpus and achieved 97.53% accuracy ([28]).

2.3 POS tagging and Indian Languages

India is a country where almost every state has its own language. Each of the languages has specific characteristics. Hence the Natural Language Processing field is a hot topic of research in India. Due to the non-availability of large anno-

tated corpora, not much work has been carried out in different Indian languages. The Indian languages are morphologically rich languages and generating a standard tag set framework for POS tagging is very difficult. Due to the different structure of sentences and grammatical rules, designing a common full tagset for all Indian languages is also very difficult. So, language specific tags have to be incorporated in the common tagset for each language.

Dinesh Kumar and Gurpreet Singh Josan have described different approaches used for languages like Hindi, Bengali, Malayalam, Punjabi and Telugu ([29]). Shambhavi B. R. and Dr. Ramakanth Kumar P. have surveyed different POS taggers developed for eight Indian Languages: Hindi, Bengali, Tamil, Telugu, Gujarati, Malayalam, Manipuri and Assamese ([30]). Similarly Antony P J and Dr. Soman K P have also given detailed literature survey of various POS taggers developed for Indian languages like Hindi, Bengali, Punjabi, Tamil, Telugu and Kannada ([31]).

IIIT Hyderabad and Baskaran et al. tried to design a common POS tagset framework for Indian languages. Due to the different structure of sentences and grammatical rules, designing a common full tagset for all Indian languages is difficult and their effort has not received wide acceptance. But using their set, researchers have added language specific tags for the language they were analysing ([32]).

2.3.1 Hindi Language

The maximum research work has been done on Hindi in India. Due to the non availability of sufficient large corpus, initial attempts were made to develop POS tagger using morphological rules of Hindi.

Shrivastava et al. used Stemmer, a morphological analyzer for Hindi prior to

developing a rule-based POS tagger. Based on the linguistic knowledge, a dictionary was built up with root words and affixes. The whole process required a lot of human effort and time and a huge data set of the language ([33]).

Pradipta Ranjan Ray, Harish V, Sudeshna Sarkar, A Basu describe a POS tagger based on lexical sequence constraints in Hindi ([34]) based on a morphological analyzer and lexical rules.

IIIT Hyderabad initiated the POS tagging and chunking contest, NLPAI ML in 2006 and SPSAL workshop in 2007. During these two contests, various methods of POS tagging have been developed for Indian languages.

Hidden Markov Model (HMM) based approach for Hindi was studied by Sankaran Bhaskaran. Manish Shrivastava and Pushpak Bhattacharyya, and they have developed a simple POS tagger for Hindi based on HMM ([35]).

Aniket Dalal, Kumar Nagraj et al. developed a POS tagger for Hindi using Maximum Entropy Markov model along with lexical and morphological fearures of Hindi language and achieved 85.59% per word accuracy. This method results in 87.04% accuracy when tested with the IIIT Hyderabad corpus ([36]).

The Conditional Random Field (CRF) approach was first introduced by Agrawal with an accuracy of 79.13% ([37]) and by Rao et al. ([38]) in NLPAI, IIIT Hyderabad.

Avinesh PVS et al. have described the combination of CRF and Transformation Based Learning (TBL) method for Telugu, Hindi and Bengali with accuracy 77.37%, 78.66% and 76.08% respectively ([39]).

Yarowsky and Delip Rao show that suffix-based POS tagging helps in improv-

ing accuracy on out-of-vocabulary words in absence of morphological information and TnT tagger performs best for Hindi, Bengali and Telugu ([40]).

Pradeep Varma D, Rakesh M and Ratna Sanyal had developed a language independent POS tagger based on HMM and tested it for English, Hindi and Urdu languages ([41]).

Sandipan Dandapat, Priyanka Biswas Monojit Choudhury and Kalika Bali had designed and developed a generic annotation tool for POS tagging using Cyclic Dependency Network as an initial annotator and described a case for Bangla and Hindi languages ([42]).

The Linguistic Data Consortium for Indian Languages (LDC-IL) has created monolingual corpora for 23 Indian languages. Hindi corpus consists of 35 million words, the largest among all the Indian languages ([43]).

The Indian languages Corpora Initiative (ILCI) consortium has been building parallel annotated corpora of 600,000 sentences (50,000 in each languages) for 12 Indian languages including English in the health and tourism domain with Hindi as the source language ([44], [45]).

2.3.2 Bengali /Bangla Language

Bengali or Bangla is an eastern Indo-Aryan language with similarities to Assamese. The script is almost the same for both the languages and the same unicode chart is used.

Asif Ekbal and S. Bandyopadhyay had developed a POS tagger based on HMM and Support Vector Machine (SVM) for Bengali ([46]).

HMM and ME based tagger for Bengali has been described by Dandapat et al. ([47]). The developed tagger is also available in internet for free.

Fahim Hasan et. al. compared the Unigram, Bigram, HMM and Brill's POS tagger for Bengali language ([48]). It was observed that due to the lack of trained data results are not very high compared to English.

Fahim Muhammad Hassan et al. of BRAC University have compared the different POS tagging techniques for Bangla, Hindi and Telugu showing that accuracy increases for most of the taggers with the increase of trained corpus data using the NLTK toolkit ([49]).

Gautam Kumar Saha described another method for Bengali using a parser in machine translation for identifying parts of speech in a sentence ([50]).

2.3.3 Other Indian Languages

We have taken an overview of different approaches of POS tagging for Dravidian family of languages like Tamil, Telugu, Kannada and Malayalam. It was observed that mostly rule based, HMM, SVM, ME approaches are tested for these languages.

CRF method has been applied for Gujarati and an accuracy of 92% was achieved ([51]).

A rule based system was used for the Punjabi language ([31]). A morphological analyzer and tagger has been developed for Manipuri based on stemming and suffix stripping method ([52]).

Jyoti Singh, Nisheeth Joshi and Iti Mathur developed a POS tagger for Marathi

language using trigram approach and achieved accuracy of 91.63%. To resolve the ambiguity, they have considered the context instead of taking single words ([53]).

Braja Gopal Patra, Khumbar Debbarma, Dipankar Das and Sivaji Bandyopadhyay have developed a POS tagger for the Kokborok language using rule based and supervised methods. In the rule based approach they have used a morphological analyzer. CRF & SVM methods were used in the supervised model and achieved 69%, 81.67% and 84.46% respectively ([54]).

2.3.4 Assamese Language

Although Assamese is a language spoken by about 15 million people in the Indian state of Assam as a first language, the development of electronic resources for the language has been lagging behind other Indian languages. Assamese language is morphologically rich and relatively free word order language like other Indian languages. It is also agglutinative in nature.

The Assamese language monolingual corpus created by LDC-IL consists of 8.3 million words. But this corpus is not freely downloadable. Though few organizations like CDAC, CIIL etc have Assamese language corpora, and some are tagged, details are not available as these works are motivated by specific research agenda and the corpora are not in the public domain. In fact no Unicode supported POS tagged Assamese language corpus is available in the public domain.

The Resource Centre for Indian Language Technology Solution (RCILTS) group, IIT Guwahati ([55]) has created an annotated parallel corpora for Assamese and Bodo languages by manually tagging the sentences using the Sanchay tagging software developed by IIIT Hyderabad. The Assamese corpus developed in this project has been used in this work.

Other than our work, only Utpal Sharma, Navanath Saharia et al. have designed a POS tagger for the Assamese language using the HMM and viterbi algorithms ([56]). They have used morphological analysis for unknown word tagging and created a tagset consisting of 172 tags for the language. The POS tagger claims an accuracy of 85.64%. The corpus generated for the approach is a non-Unicode corpus and collected from Assamese news papers. It is not in the public domain.

Mizanur Rahman, Sufal Das and Utpal Sharma have also described a context free grammar (CFG) based Assamese sentence parsing technique for simple sentences ([57]). They have shown that due to the free word order and morpho-sysnactic characteristics, it is difficult to parse even simple Assamese sentences.

Navanath Saharia, Utpal Sharma et. al. have also developed a CFG grammar to generate the parse tree for simple Assamese sentences ([58]).

Utpal Sharma, Jugal Kalita et. al. have been trying to develop a stemmer for Assamese for some time and have developed various methods. In one approach they have suggested a method to find out the new root words in a corpus using a suffix list. They presume that a valid root word is likely to occur at different places in the corpus with different suffixes. In this approach the probability of occurrence of a root word increases with the number of its occurrences in distinct cases. But this approach of finding of root word does not perform well with a small corpus ([59]).

Navanath Saharia, Utpal Sharma et al. have used suffix striping method in EMILLE corpus of size 123753 words to generate the stem words. They have manually created a rule engine to generate all possible suffix sequences that can be attached after a root word and used the suffix stripping method to find root words by matching the stripped suffix in this list. In this approach they have achieved

an accuracy of 61%. To increase the accuracy they have created a root-word list with most frequent and exceptional root words of size approximately 20,000 and matched words against this list. If the word is found in the list then the word is marked as root else again suffix stripping is done and matched in the root word list. In this approach they have achieved an accuracy of 82% due to the use of the root-list ([60]). It seems that the efficiency has increased due to the root-list.

Navanath Saharia Kishori M Konwar et. al. have developed a stemmer using a HMM based algorithm to resolve ambiguities in single letter suffixes and achieved an accuracy of 92% ([61]). They have defined two states: morphologically inflected and morphologically not inflected. Then they identify the states for each words of the corpus and calculate the transition and emission probabilities from the training corpus. They have used 2000 words of EMILLE corpus as the training corpus and 1542 words as test corpus. This approach is a modified approach of morfessor, a unifying framework for four morphology learning models developed by Mathias Creutz and Krista Lagus ([62]). This is a statistical approach of root word generation and the selection of training set used to train the model effects the efficiency of the system. On the other hand this method does not describe anything about the prefix stripping method.

It has been observed that mostly various supervised approaches have been tested for different Indian languages. Initially, emphasis was given to rule based approaches based on morphological characteristics of the languages. One of the main reasons for this was the non availability of large corpus of various Indian languages. It has also to be noted that various machine learning approaches have been applied to different languages. Some hybrid approaches has also been tested for some languages.

2.3.5 Crowd Sourcing and Semi-Automatic Techniques

With the popularity of Amazon Mechanical Turk ([63]), crowdsourcing becomes a cost effective and efficient way of developing various language resources and tools within a short time frame specially for resource-scarce languages.

Matt Post, Chris Callison-Burch and Miles Osborne have developed parallel corpora among English and six Indian languages: Bengali, Hindi, Malayalam, Tamil, Telugu and Urdu for machine translation research using MTurk ([64]).

Anoop Kunchukuttan, Shourya Roy, Pratik Patel, Kushal Ladha, Somya Gupta, Mitesh Khapra and Pushpak Bhattacharyya used the crowdsourcing approach for generation of large scale translation resources for Judiciary domain for Indian languages ([65]).

Christopher D. Manning in his paper describes that semi automated methods of POS tagging is the way to increase the performance beyond 97% ([81]).

Milan Sečujski, Vlado Delić have developed a semi automatic software tool for POS tagging and sentence accentuation for the Serbian language ([82]). They have used an algorithm based on dynamic programming along with a large dictionary containing more than 80,000 lemmas. The dictionary also consists of the inflected forms of words as separate entries and accent type and position. The software tool first tokenizes the input text and then annotates the tokens along with accentuation patterns. Then the annotated text is corrected manually using the software. The accuracy of the POS tagging system was found to be 87.2% and correct accentuation assignment rate was 97.2%.

Pavel Kveton and Karel Oliva have developed a semi automatic method for detection of errors in POS tagged corpora creating the incorrect bigrams from the tagged corpus using a statistical tagger ([83]).

Mohamed Attia et al. had developed an Arabic text annotation tool called Fassieh via a sophisticated interactive GUI application which can perform multiple linguistic analysis like POS tagging, Arabic morphological analysis, full phonetic transcription, lexical semantic analysis etc. They factorized the whole Arabic morphology into no more than 7800 morphemes and around 1100 inflection and derivation rules. The software allows the annotators to directly add syntactic diacritics whenever missing. The ambiguity of the output of factorization process was handled by the statistical disambiguation methods ([84]).

Semi-automatic methods have been mostly implemented for image annotation ([85]), text mining ([86]) and semantic annotation of the web documents to make the semantic web. Nadzeya Kiyavitskaya, Nicola Zeni et al. have have applied software design recovery techniques to semantic annotation of documents ([87]) whereas Bertrand Sereno et al. have described a strategy to support the semantic annotation in the context of the Scholarly Ontologies project ([88]).

Tomohiro YAMASAKI, Hiromi WAKAKI and Masaru SUZUKI made a framework to develop POS taggers and POS lexicons of some target languages using information of other source languages. They built POS taggers for English, Spanish and Esperanto manually and used as the source languages to build the POS taggers of target languages (Portuguese and Indonesian) semi automatically applying the machine learning using CRF and POS lexicons using the parallel corpora ([89]).

In Indian languages, A. Das and S. Bandyopadhyay adopted a number of automatic, semi-automatic and manual validations and evaluation methodologies to measure the coverage and credibility of the developed SentiWordNet(s) for Bengali, Hindi and Telegu ([90]).

Chapter 3

Assamese Language

Assam is situated in the northeast of India with an area of 78,438.00 square kilometers, comprising 27 districts. The inhabitants of Assam are a broad racial intermixture of Mongolian, Indo-Burmese, Indo-Iranian and Aryan origin. The hilly tracks of Assam are mostly inhabited by tribes of Mongolian origin. This broad racial intermixture is the native of the state of Assam, whose main language is "Asomiya" or "Assamese" which is also the state language of Assam. Assamese or Asomiya (অসমীয়া /Axomiya/) is also used to define people on a cultural or ethnic basis.

3.1 Description of Assamese

Assamese is the principal language of the state. The Assamese language is the easternmost member of the Indo-European family. As a first language it is spoken by over 15 million people and including those who speak it as a second language, a total of 20 million speak Assamese primarily in Assam and in parts of the neighboring states. Immigrants from Assam have carried the language with them to other parts of the world.

Assamese is written using the Assamese script, similar to that of Bengali except

the symbols for ৰ/ra/ and ৱ/wa/, and highly resembles the Devanagiri script of Hindi, Sanskrit and other related Indic languages. As such it is a syllabary script and is written from left to right and top to bottom, in the same manner as English. The Assamese alphabets have consonant letters, independent vowel letters, dependent vowel signs (matras), punctuation and numerals from zero to nine. The total number of graphemes in the Assamese Script is fifty two. Out of these eleven are vowels and rest forty one are consonants. There are equivalent matras for all the vowels. These matras can exist above, below, to the right or to the left of the consonant to which it is applied. There are thirty three pure consonant letters in the Assamese alphabet and each letter represents a single sound with an inherent vowel, the short vowel অ/a/. Assamese spelling is not always phonetically based. Current Assamese spelling practices are based on Sanskrit spelling, as introduced in the second Assamese dictionary *Hemkosh* which was written in the middle of the 19th century. A large number of ligatures are possible since potentially all the consonants can combine with one another. Also, the combination of three and rarely four consonants are also possible without their intervening vowels. Vowels can either be independent or dependent upon a consonant or a consonant cluster.

3.2 Phonological Features

The Assamese phoneme inventory consists of eight vowels and twenty one consonants and ten diphthongs depending upon the analysis. The vowel sounds in Assamese language occur in all the three positions, namely word initially, medially and finally.

A unique feature of the Assamese language is a total absence of any retroflex sounds. Instead the language has a whole series of alveolar sounds, which include oral and nasal stops, fricatives, laterals, approximants, flaps and trills, unlike other Indo-Aryan and Dravidian languages.

Another striking phonological feature of the Assamese language is the extensive use of velar nasal ঙ/ŋ/. In other New Indo Aryan languages ঙ/ŋ/ is always attached to a homorganic sound like /g/. In contrast it is always used singly in Assamese.

The voiceless velar fricative /x/ (or h) is a distinct characteristic of Assamese language which is not to be found in any language in the entire country. It is similar to the velar sound in German of Europe. It may be an Indo- European feature, which has been preserved in Assamese. It is an important phoneme in the language. The derivation of the velar fricative from the coronal sibilant /s/ is evident in the name of the language in Assamese; some Assamese prefer to write (Oxomiya) or (Ôxômiya) instead of (Asomiya) or (Asamiya) to reflect the sound change.

3.3 Morphology and Grammar

A subject and a predicate form a sentence or it may be said that Subject, Object and Verb are the essential items of a sentence in Assamese. The basic word order of the Assamese language is SOV (Subject-Object-Verb). But it is a free word order language and hence SVO, OSV, OVS, VOS and VSO can realize the same meaning. Out of them OSV is more frequently found in the native speakers' speech. There are some exceptions, where sometimes a sentence is completed by only one item of a sentence stated above. Such exceptions occur in emotional description and conversational language. The roots of the subject, verb and object of a sentence are generally noun, pronoun, adjective, preposition, verbs etc. Assamese sentences can be classified as simple, complex and compound sentences based on the structure of the sentences. Following are few examples of the sentence types.

Simple Sentence:

সি কামটো কৰিব [ɦi kamtʊ kɔribɔ] *''He will do the work."*
ɦi (He, 3SG(M).NOM) kam (work) -tʊ(-CLS) kɔr (do) –ib (-FUT) -ɔ (-PERS)

Complex Sentence:
সি কৈছে যে সি কামটো কৰিব [ɦi kɔise ze ɦi kamtʊ kɔribɔ] *"He said that he'll do the work."*
ɦi (He, 3SG(M).NOM) kɔ (say) –is (-ASP) –e(-PERS) ze(that, 3SG(m).NOM) ɦi (He, 3SG(M).NOM) kam (work) -tʊ(-CLS) kɔr (do) –ib (-FUT) -ɔ (-PERS)

Compound Sentence:
সি কৈছে যে সি কামটো কৰিব আৰু যাব [ɦi kɔise ze ɦi kamtʊ kɔribɔ aru zabɔ] *"He said that he'll do the work and go"*
ɦi (He, 3SG(M).NOM) kɔ (say) –is (-ASP) –e(-PERS) ze(that, 3SG(m).NOM) ɦi (He, 3SG(M).NOM) kam (work) -tʊ(-CLS) kɔr (do) –ib (-FUT) -ɔ (-PERS) aru (and) za (go) –b(-FUT) -ɔ (-PERS)

Nouns in Assamese are mostly monosyllabic such as আই /ai/ meaning *mother* as well as di-syllabic such as দেউতা /deuta/ meaning father. However derived nouns such as পিতাক /pitak/ meaning *your father* etc are also frequently used.

Personal, demonstrative, interrogative, reflexive and indefinite pronouns are available in Assamese.

Nouns, pronouns are generally inflected for number, gender and case in Assamese. Assamese has two numbers, singular and plural. Plural numbers can be formed by adding suffixes to the singular forms of nouns [ৰামহঁত, /ramhot/ ram (ram) -hɔt(-PL) *'Ram and others'*], pronouns [সিহঁত, /xihɔt/ xi (he) -hɔt(-PL) *'They'*]. They can also be expressed by adding qualifying words, in which case no suffix is added [বহুত মানুহ, /bɔhut manuh/ bɔhut (many) manuh(man) *'Many men'*].

Assamese adjective is basically not inflected, but sometimes adjectives are inflected. When adjectives are inflected, then they take the noun form. Assamese adverbs are dependent on adjectives. When adjective words take verbal form then the word becomes an adverb. Assamese conjunctions and interjections are indeclinable.

In Assamese, gender is not grammatical. The distinction of sex may be expressed mainly by three ways – using separate noun bases for male and female [ভাই /bʰai/ *'brother'* and ভনী /bʰoni/ *'sister'*], using separate qualifying words before or after the common nouns [মতা মানুহ /mɔta manuh/ *'male'* and মাইকী মানুহ /maiki manuh/ *'female'*], using different suffixes. Assamese possesses the following feminine suffixes: ী /-i/ (নিলাজী /nilazi/ *'shameless woman'*), অনী /-ɔni/ (নাপিতনী /napitɔni/ *'female barber'*), নি /-ni/ (নাতীনি /natini/ *'granddaughter'*) and ওনী /-uni/ (চুৰুনী /suruni/ *'female thief'*). In many cases tatsama words puruh (Sanskrit purusa, male) and mohila 'female' are used before the terms belonging to a common gender [পুৰষ ৰাষ্ট্রপতি /puruh rastrɔpɔti/ *'male president'* or মহিলা ৰাষ্ট্রপতি /mohila rastrɔpɔti/ *'female president'*]. Sometimes sex is indicated by addition of enclitic definitive টো /tʊ/ for masculine and জনী /zoni/ for feminine after the nouns. There is no masculine suffix in Assamese but in some derived words formative suffix া /-a/ is used as masculine suffix. For example, খৰা-নাক /kʰɔra-nak/, meaning *flat-nose*; after this if we add া /-a/, it becomes খৰা-নকা /kʰɔra-nɔka/, meaning *'a man who has a flat nose'*. It may be changed to feminine gender after adding the feminine suffix ী /-i/, e.g. খৰা-নাকী /kʰɔra-naki/, meaning *'a woman who has a flat nose'*. There are some nouns in Assamese which always belong to the feminine gender; they have no masculine forms in opposition like দাউনী /dawɔni/ *'a reaper woman'*.

Assamese possesses some suffixes which are generally added to the masculine noun bases to indicate feminine gender. These suffixes are called feminine suffixes. The sex of a few words can be known only from the context like সখী /xɔkhi/ *'intimate friend'*. In writing to denote the feminine these are written with a long

ী /-i/. Some of the third personal pronouns have gender distinction in Assamese, e.g. সি /hi/ *'he'*, তাই /tai/ *'she'*, ই /e/ *'this man'*, এই /ei/ *'this woman'*.

There are only three persons in Assamese - first person, second person and third person. There are three ways of address for second and third person: inferior, normal, and honorific. So মই /mɔI/ meaning *'I'* and আমি /ami/ meaning *'we'* in first person. In second person we have তুমি /tumi/ meaning *'you'* (normal), তই /toi/ (inferior) and আপুনি /apuni/ (honorific). In third person we have, সি /xi/ meaning *'he'* (inferior and normal, male), তাই /tai/ (inferior and normal, feminine) and তেখেত /tekhet/, তওঁ /teo/ meaning "he" (honorific).

In Assamese, some post positional words or affixes are added to nouns, adjectives and numerals and these have the sense of the definite article "the" and after attachment to a word such a definitive becomes a part of the word [eg.: মানুহ+টো = মানুহটো /manuh+tu = manuhtu/ meaning *'the man'* or ধুনীয়া+টোৰপৰা = ধুনীয়াটোৰপৰা /dhuniya + turpɔra = dhuniyaturpɔra/ meaning *'from the beautiful'*]. Assamese has a special feature to indicate relationships. Different affixes are used to indicate the relationship of someone with first, second and third persons [eg.: মোৰ মা /mur maa/ (1st person, মা +0) meaning 'my mother', তোমাৰ মাৰা /tomar maara/ or তোৰ মাৰ /Tur maar/ (2nd person, মা +ৰা/ৰ) meaning 'your mother' and তাৰ মাক /tar maak/ (3rd person, মা +ক) meaning 'his mother'].

The case relationship in Assamese sentences are indicated by using the case endings as suffixes directly to the nouns and pronouns or with some post positional words, after which case endings are added to indicate the case relation. There are six cases with distinct case markers available in Assamese namely nominative (ে /-e/), accusative (ক /-k/), locative (ত /-t/), allative (পৰা /-pora/), instrumental (ৰে, দ্বাৰা /-re, -dwara/) and genitive (ৰ /-r). The case markers are always added at the end of the words. It is interesting to note that when the subject has an interrogative verb, nominative case ending is zero [eg.: মধু (মধু+ 0) গ'ল /madhu

গ‍ʊl/ meaning 'Madhu went']. When the verb is transitive, the subject text is nominative case ending [eg.: মধুৱে (মধু+ৱে) ভাত খালে /modhuwe (modhu+e) bhat khale/ meaning 'Madhu ate rice'].

Verbs in Assamese subject themselves to various kinds of classification. From a wider point of view, verbs could be classified into two categories. They are main verbs and auxiliary verbs. Main verbs can again be divided into simple and derived verbs. Verb roots themselves are considered as simple or primary verb stems. They are verb stems in their own right because sometimes they are meaningful on their own, without being inflected for any markers. Such cases are found with the present imperative verb form for 2nd person inferior [eg.: খা /kha/ meaning 'you eat' or যা /za/ meaning 'you go']. However, for other tenses, aspects and persons, the verb stems take regular inflection [eg.:মই খাওঁ (খা+ও) /moi khau (kha+u)/ meaning 'I eat']. Derived stems are formed by adding different suffixes to the simple stem. This morphological process is termed as derivation. Derivational morphemes generate new words from existing ones, thereby widening the scope of expression [eg.: বহ /bɔh/ meaning *'sit'*+ ওৱা/-uwa/ = বহোৱা/bɔh-ʊwa/ meaning *'cause to sit'*]. Derived stems do not fulfill any syntactic function but they meet only morphological and semantic necessities. This is true of Assamese derivatives also.

Assamese verbs manifest a three-tier distinction, viz, present, past and future. They are marked by suffixation. Tense markers are directly suffixed to the verb stems if there does not involve any aspect markers. However, if there is any aspect marker, it invariably precedes the tense marker. The tense markers are not terminal suffixes: person markers invariably are added to them. The tense marker changes depending on the person. While the past and the future tenses in Assamese are marked by suffixation, the present tense is marked by the absence of any overt suffixation which is generally expressed as ø. There is no tense marker for the present tense [eg.: খা /kha/ meaning *'eat'*+ ও/ʊ/ = খাও /kha ʊ/ ; verb stem

+ ø + person marker(s)].

The past tense in Assamese is marked by ল /-l/ which has another allomorph ইল /-il/. -l occurs after vowel ending stems and -il after consonant ending stems. In the absence of aspect markers, past tense makers are directly suffixed to the verb stems and the person markers are thereafter added to them [eg.: খা /kha/ meaning *'eat'*+ ল /-l/ + ও/ʊ/ = খালো /khalʊ/ ; verb stem + -l/-il + person marker(s)].

The future tense in Assamese is marked by ব /-b/ which has three allomorphs ইব /-ib/, ম /-m/ and ইম /-im/. -b and -ib occur with 2nd and 3rd persons while -m and -im occur with 1st person. Of these, -b and -m occur with vowel ending verb stems while -ib and -im occur with consonant ending verb stems [eg.: খা /kha/ meaning *'eat'*+ ম /-m/ = খাম /kham/ ;Verb stem + -b/-ib/-m/-im + person marker(s)].

Assamese verbs manifest aspectual contrasts. Three aspects could be identified in Assamese verbs. They are habitual or indefinite, perfective or completive and continuative or progressive. The aspect markers are directly added to the verb stem; tense marker, if any, is suffixed to the aspect markers. In the absence of a tense marker, the person marker is directly added to the aspect marker. The aspect markers have no allomorphs. The habitual aspect denotes the habitual occurrence of the action suggested by the verb. This aspect is relevant to all the tenses, viz, present, past and future. It is marked by ø. The perfective or completive aspect denotes the completion of the action of the verb. This is clearly marked by a bound morpheme ইছ /-is/. It could express completion of action in both present and past tenses. However, Assamese does not manifest this aspect for future tense verbs. Progressive aspect suggests that the action suggested by the main verb is in progression. This can be realized generally for present as well as past tenses and sometimes for future tense verbs too. The verbs in the progressive aspect are constructed periphrastically. A bound morpheme ই /-i/ is suffixed

to the main verb stem.

3.4 Grammatical Features

The use of personal markers with regard to the use of various kinship terms in reference to the age and rank of both the speaker and listener is a unique feature of the Assamese language. For instance তোমাৰ দেউতাৰা /tumar deutara/ *'your father'* Here, ৰা /ra/ is the personal deictic or marker.

The process of negation of verbs in Assamese is another feature which clearly demarcates it from the rest of the Indo- Aryan and other Dravidian languages. In Assamese ন /n/ is pre-fixed to the verb followed by a vowel which is the exact copy of the vowel of the first syllable of the verb, as in নালাগে /nalage/ meaning 'do not want' (1st, 2nd, 3rd person). The various negative markers in Assamese are ন /n/, নো /no/, না /na/, নে /ne/ and নি /ni/ etc.

The use of the plural suffixes is another feature of Assamese. For instance, all the bound forms such as হঁত /hɔt/ , বোৰ /bur/ , বিলাক /bilak/ , মখা /mokha/ , জাক /zak/, সকল /xɔkɔl/ etc denote plurality and are suffixed to a noun or a pronoun.

The extensive use of classifiers is another feature of the Assamese language. For almost everything or every shape the language uses a different classifier. The classifiers are also combined with all types of nouns and numerals occurring in the language resulting in combinations of the following type of grammatical constructions.

এ /ɛ/+ জন /zon/ + মানুহ /manuh/ (numeral + classifier + noun) (one man)

মানুহ /manuh/ + এ /ɛ/+ জন /zon/ (noun + numeral + classifier) (one man)

The Assamese language has its own numerals from zero to nine, punctuation

marks, localized data like calendar, time, months of the year etc.

3.5 Assamese Unicode based corpus

A corpus is one of the basic requirements of a POS tagger. Initially we did not have any Assamese Unicode based corpus. We had a very small non-unicode glyph based text corpus of around a thousand words. We designed a converter to convert the non-unicode texts to Unicode text. As part of a sponsored project, Resource Centre for Indian Language Technology Solutions (RCILTS), Unicode based corpus (corpus C1) of about 16,264 sentences (about 2,50,000 words) was keyed in. This corpus was tagged by language experts using Sanchay software developed at IIIT Hyderabad (IIITH) ([69]). This annotated corpus was used to create the knowledge base for POS tagging. Details are given in later chapters and all the copora used are described in Annexure 1.

3.6 Tagset for Assamese

Tagset is another basic requirement of a POS tagger. Designing a tagset covering the morpho-syntactic details of the language is a very difficult task. A common tagset for Indian languages provide advantages like flexibility, cross-linguistic compatibility and reusability. To achieve this. Sankaran et. al. had developed a common POS tagset framework for Indian Languages (IL-POSTS) ([70]). IL-POSTS offers broad guidelines for users to define their own tagset for a particular language and/ or a specific application. IIIT Hyderabad has also developed a common tagset to cover all Indian languages ([71]). It is mostly based on the Penn tagset which is a standard tagset for English and only a few changes and modifications were made which, however, are not enough to capture the rich linguistic features of Indian languages. With the motivation to have less number of tags, this tagset has adopted a very coarse structure in linguistic analysis, leading

to a very flat structure capturing only coarse-level categories. While it may be useful for translations in limited domains of text, this tagset needs to be enhanced to capture the nuances of a particular language.

Several POS tagsets have been designed by a number of research groups working on Indian Languages though very few are available publicly (IIIT-tagset, AU-KBC Tamil tagset). However, as each of these tagsets have been motivated by specific research agenda, they differ considerably in terms of morpho-syntactic categories and features, tag definitions, level of granularity, annotation guidelines etc. Moreover, some of the tagsets (e.g., the AU-KBC Tamil tagset) are language specific and do not scale across other Indian languages. This has led to a situation where despite strong commonalities between the languages addressed, resources cannot be shared due to incompatibility of tagsets. This is detrimental to the development of language technology for Indian Languages which already suffers from a lack of adequate resources in terms of data and tools.

It has been observed during the project, RCILTS (phase II), that the tagset available in Sanchay software(version 0.1) developed by IIIT Hyderabad for Indian languages cannot fully describe some of the basic morpho-syntactic behavior of Assamese. Assamese has rich linguistic features and idiosyncrasies which clearly were not taken into account while planning the structure of the tagset. Having gone through the analysis of linguistic data, we have come across many issues that must be dealt with utmost care to avoid a flat structure representation of the language. Considering all the above facts, we have modified the meaning of some of Sanchay tags (tagset containing 27 tags) and have added 7 new tags. We are therefore using a tagset containing 34 tags as shown in *Table 3.1* in *Page 38*, for all our experiments.

Table 3.1 List of used tagset

Sl. No.	Sanchay Tagset	Our Tagset	Meaning
1	CC	CC	Conjunction
2	INTF	INTF	Intensifier
3	JJ	JJ	Adjective
4	JVB	JVB	Adjective in Kriya Mula
5	NEG	NEG	Negative
6	NLOC	NLOC	Noun Location
7	NN	NN	Noun
8	NNC	NNC	Noun Common
9	NNP	NNP	Noun Proper
10	NNPC	NNPC	Compound Proper Nouns
11	NVB	NVB	Noun in Kriya Mula
12	PREP	PREP	Preposition
13	PRP	PRP	Pronoun
14	PUNC	PUNC	Punctuation
15	QF	QF	Quantifier
16	QFNUM	QFNUM	Number Quantifier
17	QW	QW	Question Word
18	RB	RB	Adverb
19	RBVB	RBVB	Adverb in Kriya Mula
20	RP	RP	Particle
21	SYM	SYM	Special (Symbol)
22	UH	UH	Interjection word
23	VAUX	VAUX	Auxiliary Verb
24	VFM	VFM	Verb Finite Main
25	VJJ	VJJ	Verb Non-Finite Adjectival
26	VNN	VNN	Verb Non-Finite Nominal
27	VRB	VRB	Verb Non-Finite Adverbial
28		VAUXN	Negative Verb Auxiliary
29		VNF	Non-Finite Verb
30		VNFN	Negative Non-Finite Verb
31		VJJN	Negative Verb Non-Finite Adjectival
32		VNNN	Negative Verb Non-Finite Nominal
33		VFMN	Negative Verb Finite Main
34		VRBN	Negative Verb Non-Finite Adverbial

Chapter 4

Assessment of Probabilistic Approaches

We have observed that mostly probabilistic methods have been used to implement automatic POS taggers in various Indian languages. So we decided to evaluate the efficacy of some basic probabilistic POS tagging techniques for the Assamese language. The NLTK toolkit ([72]) consists of various in-built modules of probabilistic tagging and it is open source software. Four Indian POS tagged corpus namely Hindi, Bangla, Marathi and Telugu are freely available with the toolkit. We have evaluated the n-gram, Backoff, Brill and HMM probabilistic POS tagging techniques for Assamese using the NLTK toolkit. In order to consider using a probabilistic method as part of our tagging process, we tried to find out the relationship between the size of trained corpus and the corresponding changes in accuracy of POS tagging. We took the RCILTS developed generic corpus containing 1867 sentences with 20414 tokens to test the modules (this is a subset of the C1 corpus, C1-G). We also took 73 sentences with 1240 tokens from the tourism corpus developed in the project (another subset: C1-T) to test the accuracy between inter domain corpora. When both the trained and test sets are from the same domain (either from C1-G or from C1-T) we term that as intra domain and when the trained and test sets are from different corpus we term that as inter domain.

We used the open source Natural Language Toolkit, NLTK, version 0.9.9, for the evaluation of various taggers. The toolkit was developed at the University of Pennsylvania in 2001. It is a suite of libraries and programs for symbols and statistical natural language processing written in the python programming language([73]). It covers symbolic and statistical natural language processing and is interfaced to annotated corpora. A tagged corpus has to be fed as input to form a training set. Decisions are then taken based on the contents of this training set. We first describe briefly each of the methods that we used. It may be noted that even though the Kit has a Hidden Markov Model based tagger, this was not used as the Kit does not have any tools for automatically generating the transition and emission probabilities from training data.

4.1 Default Tagger

As the name explains, this tagger assigns the same tag to each word irrespective of context. The default tagger can help to improve the robustness of a language processing system. In general the performance of the tagger is very poor but plays an important rule to establish an important baseline for tagger performance while combined with other taggers. The user can choose the most likely tag to be used in the default tagger as the default tag for a corpus. Words in a corpus are assigned tags and the words that have not been tagged before are given the default tag. We are using noun, NN, as default tag for our corpora.

4.2 n-gram Tagger

An n-gram tagger uses a stochastic approach as described in Section 1.2.2, and is a simple form of tagging. Depending upon the value of n, the number of terms used, taggers can be named as unigram (n=1), bigram (n=2) and trigram (n=3).

A Unigram tagger assigns that tag which has been assigned the most to that word in the training set. Since a Unigram tagger always considers the current token, the context of the word has nothing to do with the tag.

A bigram tagger considers the tag already given to the previous word and chooses that tag for the current word which occurs most frequently with the previous word and its allotted tag, in the training set. The efficiency of a bigram tagger depends upon the training set. It will tag efficiently until it finds an unseen word. Whenever it encounters a new word, it tags that word as None and there after the efficiency decreases drastically. This happens because it never sees a None tag on the previous words in the training set.

A trigram tagger is like the bigram tagger except that it considers the previous two tags in addition to the current word instead of only one preceding tag. The efficiency of the tagger depends upon the training set.

4.3 Backoff Tagger

As n gets larger in n-gram taggers, the context specificity increases and as a result the performance of the tagger decreases drastically. This is known as the sparse data problem, and is quite pervasive in NLP. As a consequence, there is a trade-off between the accuracy and the coverage of the results.

The toolkit provides a facility to combine the results of n-gram taggers and the default tagger to address the trade-off between accuracy and coverage. The combination may either happen with one n-gram and a default tagger or multiple n-gram taggers and the default tagger. The combined tagger is termed as Back-off tagger. Starting with the trigram tagger, it backs-off to the bigram tagger if the word sequence is not found in the trigram list. Similarly, it backs-off to the unigram tagger if the bigram tagging fails. If there is a failure at the Unigram

level also, then the default tagger is used, which has been set to NN for our corpora.

The backoff tagger has to specify during the initialization of the tagger so that training can take advantage of the backoff tagger. Thus, if the bigram tagger would assign the same tag as its unigram backoff tagger in a certain context, the bigram tagger discards the training instance. This keeps the bigram tagger model as small as possible.

4.4 Brill Tagger

A Brill tagger is also included. As described in Section 1.2.1, it is a rule based tagging scheme. Though this tagger requires training data, it does not count the observations but compiles a list of transformational correction rules. The n-gram taggers are not context sensitive but a Brill tagger is context sensitive and it generates the transformation rules based on the contexts of the training set. *Fig 4.1* of *Page 43* shows a snapshot of the execution of a Brill tagger using NLTK.

4.5 Evaluation of the Techniques

In order to evaluate the models, we divided the generic corpus annotated data (C1-G) into different sets, namely the trained and test sets to test the intra domain performance. If the test set is too small, then the evaluation may not be accurate. However, making the test set larger usually means making the training set smaller, which can have a significant impact on performance if a limited amount of annotated data is available. To resolve this problem we have performed multiple evaluations on different test sets, and then combined the scores from those evaluations. This technique is known as cross-validation. In particular, we subdivided the original corpus into equally five subsets called folds with approximately 4082 tokens in each fold. For each of these folds, we trained the model using all

```
    Found 28114 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
  29  33   4  16  | VNF -> VFM if the tag of the following word is
                  |   'PUNC'
  21  22   1   0  | VFM -> VNF if the tag of the following word is
                  |   'VAUX'
  18  19   1   0  | JVB -> JJ if the tag of the following word is 'NN'
  16  16   0   4  | JVB -> JJ if the tag of the following word is
                  |   'PUNC'
  13  15   2   1  | VFM -> VAUX if the text of the preceding word is
                  |   '\xe0\xa6\xa6\xe0\xa7\x87\xe0\xa6\x96\xe0\xa6\xb
                  |   e'
  12  12   0   0  | VFM -> VAUX if the text of the preceding word is
                  |   '\xe0\xa6\x95\xe0\xa7\xb0\xe0\xa6\xbe'
  12  16   4   4  | VJJ -> VNF if the text of word i+2 is
                  |   '\xe0\xa5\xa4'
  11  21  10   4  | JJ -> JVB if the tag of the following word is
                  |   'VNF'
  11  12   1   7  | NVB -> NN if the tag of the following word is 'NN'
  11  13   2   6  | VAUX -> VFM if the tag of the preceding word is
                  |   'NN'
>>> brill_tagger.evaluate(train_sents)
0.9121198568872988
>>> brill_tagger.evaluate(test_sents)
0.51433691756272404
>>> brill_tagger.evaluate((assamese_sents)*0.5)
```

Figure 4.1: Snapshot of Brill tagger using NLTK

of the data except the data in that fold, and then tested that model on the fold. For example in one instance we have taken 80% of the corpus (four folds) as the trained set and the remaining 20% (one fold) is used as the test set. Similarly to test the inter domain performance, we have divided the tourism corpus (C1-T) into five folds with approximately 250 tokens in each fold and used as them as test sets against the generic corpus train sets. Even though the individual folds might be too small to give accurate evaluation scores on their own, the combined evaluation score is based on a large amount of data, and is therefore quite reliable. The advantage of using cross-validation is that it allows us to examine how widely the performance varies across different training sets.

The table *Table 4.1* in *Page 44* describes the results of various models tested in inter and intra domain corpora. We have displayed the minimum and maximum values that we got during the evaluation of different train/test folds. For example the minimum and maximum values for Trigram tagger for inter domain test folds

varies between 30-50%.

Table 4.1 Results of various taggers

	Iter Domain		Intra Domain	
	Min	Max	Min	Max
Unigram	50%	51%	31%	34%
Bigram	40%	52%	30%	30%
Trigram	30%	50%	23%	29%
Unigram Backoff	47%	49%	66%	68%
Bigram Backoff	47%	48%	67%	68%
Trigram Backoff	47%	48%	67%	69%
Brill	12%	76%	61%	68%

4.6 Analysis

We have observed from the results of the various taggers that the efficiency increases with the increase of the training data for Unigram, Backoff and Brill taggers. Sometimes the efficiency of the taggers were drastically reduced in inter domain corpora because no word-tag combinations of the test corpus were trained in the training corpus. Variations in efficiency have also been observed with different corpora. It was observed from the table that the scores of all the five training sets are not varying much implying that the scores are fairly accurate.

Due to the sparse data problem, the bigram and trigram tagger results comparatively more deviations in minimum and maximum scores in inter domain corpus compared to intra domain corpus. As soon as these two taggers encounter a new word, they are unable to assign a tag to the new word. It cannot tag the following word even if it was seen during training, simply because it never saw it during training with a *None* tag on the previous word. Consequently, the tagger fails to tag the rest of the sentence. Its overall accuracy score is very low.

As has been reported by earlier researchers, a tagger based on any one of these methods is unlikely to provide good performance. Our analysis has given a similar indication. However, during our experimentation and analysis, we realised that

a n-gram tagger could be used as one of the steps in the tagging process. As we describe in later chapters, we are using a bigram tagger to try and eliminate ambiguities left over from an earlier rule based tagging step. This has yielded good results as we describe in later chapters.

Chapter 5

Knowledge Base Creation

We have seen that no single approach to automatic POS tagging gives a good percentage of accurate tags. This is not only true of Assamese (where limited work has been done as mentioned above), but of almost all languages. We have also seen that to get good accuracy, a lot of manual work has to be done before the tagger can be used. This includes tasks such as creating a tagged training set of sufficient size, doing morphological analysis of the language, creating a stemmer, building a dictionary, etc.

5.1 Our Approach

Our approach is to use a combination of methods to try and get good results. Further, we amortise the manual intervention over the period of tagging rather than doing all manual work at the beginning. This allows us to quickly start the tagging system. But it also means that what we have is a semi-automatic tagger and not an automatic tagger. Our method requires only native speakers' intervention in stages other than the beginning making the system amenable to some form of "crowd-sourcing" with a few experts for moderation, and this will enable our system to create very large tagged corpora in the language. The flow chart, *Fig 5.1* of *Page 48*, depicts our approach.

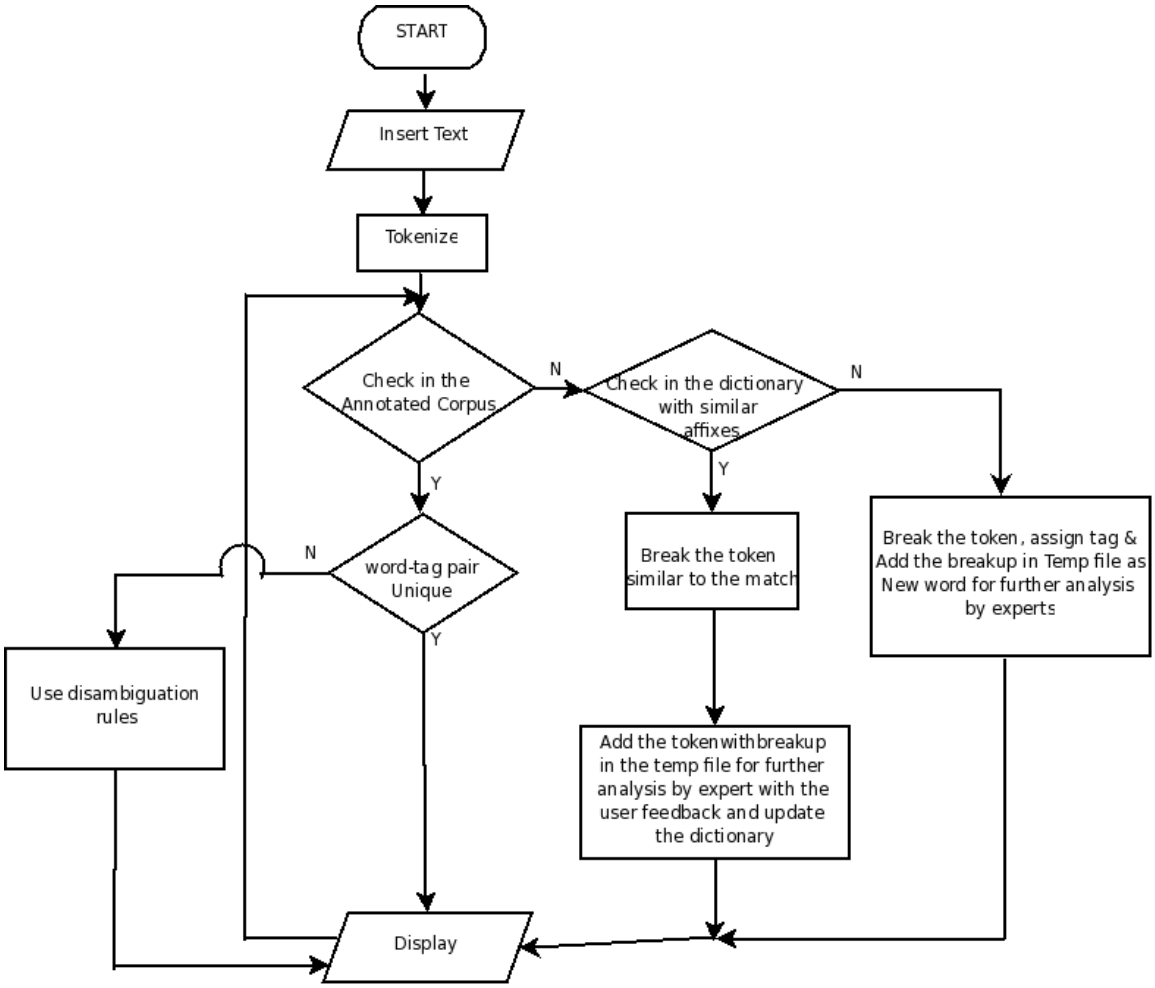


Figure 5.1: Flow Chart of our Approach

We first create a knowledge base using a number of methods which we describe in detail in this chapter. This knowledge base is then used to automatically tag sentences in the language. This tagging uses a combination of stemming, application of a few grammatical rules, and using a bigram tagger. The tagged sentences are then shown to non-expert native speakers for verification and correction. Before starting the actual tagging process, the knowledge base was tuned by examining the results on small data sets using experts instead of native speakers. The design of a user friendly interface plays an important role in reducing the time taken by native speakers in their examination. All these details are described in

the next chapter.

5.2 Initial Resources

We had the tagged corpus C1 (already mentioned in chapter 3), of about 16,264 sentences (about 2,50,000 words). This corpus is divided into three parts: two generic corpora - C1-GA, consisting 7451 sentences (around 87,639 words) & C1-GB consisting 1671 sentences (around 25223 words). The third corpus based on tourism related information (C1-T). We had a dictionary containing root words, from which verb root words could be, and were used as described later. Then we had two untagged corpora, C2 (48,950 sentences with 5,54,054 words) and C3 (443 sentences with 7734 words) which were used in the tagging process. Another very small generic corpus C4 of 37 sentences containing around 490 words was also used in the process.

We have generated a frequency list of the words of the corpus. In computational linguistics, a frequency list is a sorted list of words (word types) together with their frequency of occurrences in a given corpus. We have generated two frequency lists namely frequency list of words *Fig 5.2* of *Page 50* and frequency list of word-tag combination *Fig 5.3* of *Page 50* using the C1 corpus. From the frequency list of words we got the most common words in the corpus and from the word-tag frequency list we got the most common word-tag combinations. Using both the lists, we can find out the words with multiple tags based on the context. These information are useful while framing language specific rules.

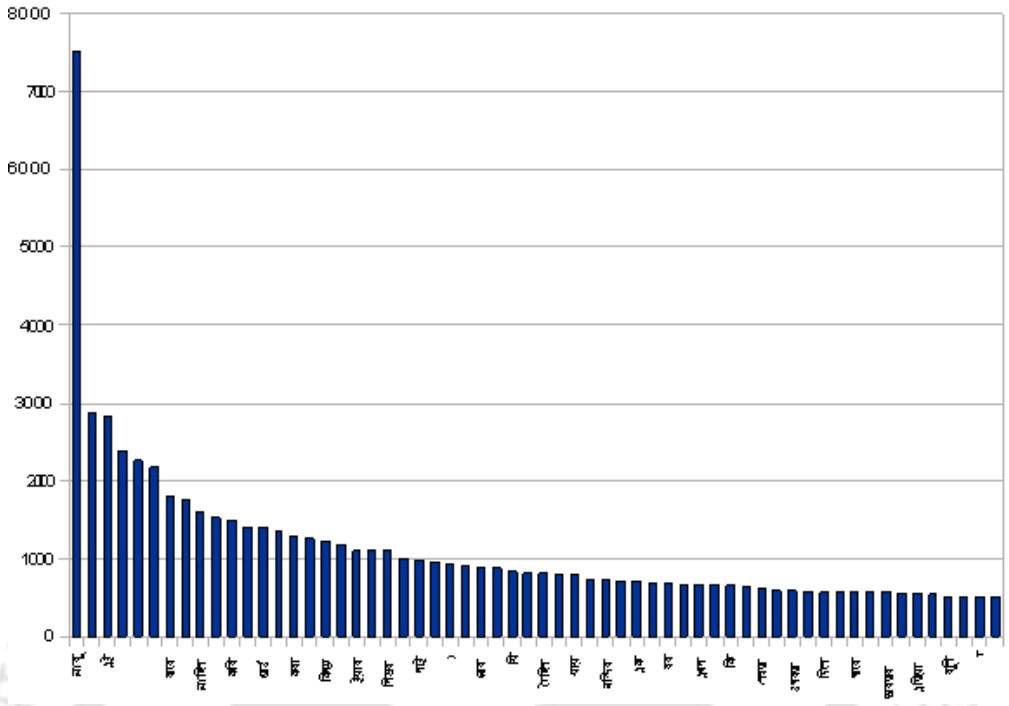


Figure 5.2: Frequency list of word (upto freq 500)

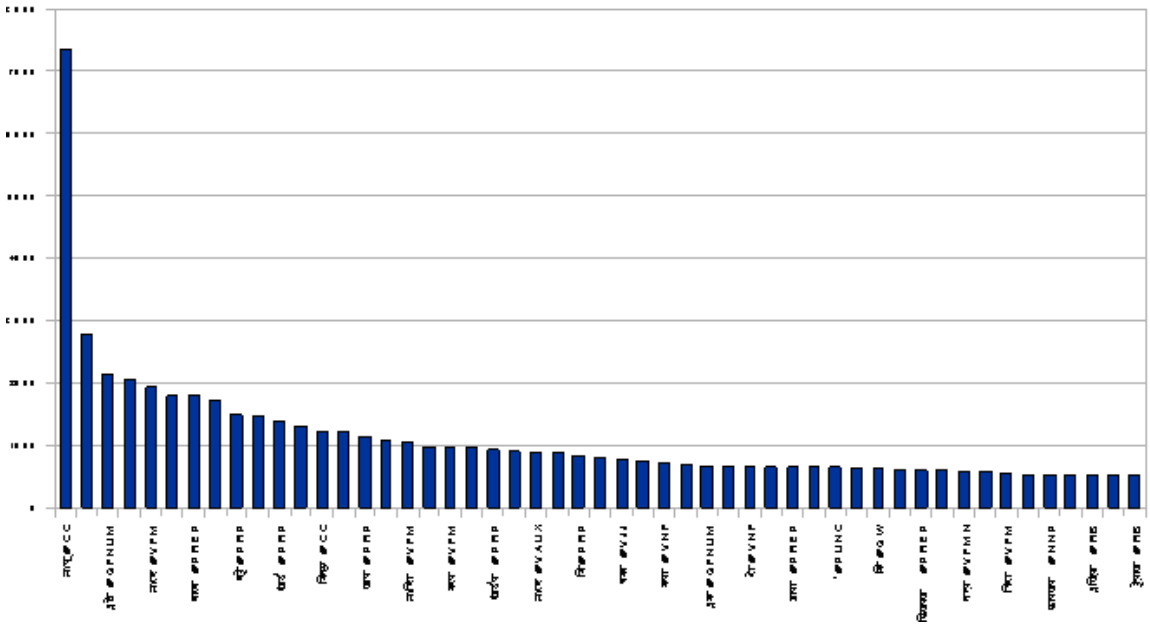


Figure 5.3: Frequency list of word-tag (upto freq 500)

The following table *Table 5.1* in *Page 51* is the list of the most common 20 words or the highest frequency words found in the corpus. আৰু /aru/ meaning

'and' is the most frequent word found in the corpus. This means that Assamese language sentences contain frequent conjunct forms.

Table 5.1 Word - Frequency table

Word	Frequency
আৰু	7520
আছে	2862
এই	2836
পৰা	2391
এটা	2181
বাবে	1813
হৈ	1746
আছিল	1605
মই	1519
কৰি	1486
হয়	1404
তেওঁ	1401
সেই	1343
কৰা	1280
যে	1269
কিন্তু	1222
তাৰ	1174
ইয়াৰ	1101
কথা	1097
নিজৰ	1094

We have stored all the unique word-tag pairs of the C1-GA corpus into a table named the "submitted table".

We have also assigned bigram values to two consecutive tags of the C1 corpus based on the frequency of availability of those two tag pairs in the corpus. We have taken out the tag sequences from the corpus and generated the bigram values for each pair of tags and saved them in the "bigram table". Since we have taken the tag sequences from a tagged corpus created by linguistic experts, possibility of false positive tag sequences with bigram values are very less.

We handle stemming and morphological analysis of words in a base corpus us-

ing the a priori knowledge of language experts. We took a very small portion of the generic Unicode corpus (C6) of around 2,500 words and manually analyzed each word. Words in this corpus were analyzed, broken down into root words, affixes (prefixes and/or suffixes), quantifiers and classifiers etc by a team of experts. Each word is analyzed in detail, so that all the lexical and grammatical information are successfully identified. In this word analysis process we also gather the lexical and grammatical rules which will be used for the automatic tagging process. Words so analyzed are stored in a database. Also, using a priori knowledge, the "prefix" and "suffix" tables were created. In contrast to other approaches, the base corpus is relatively small and so the effort required to create the database is small. This database is then used as a resource for the later stages.

5.3 Creation of Root Table

Root word creation from a word is an important issue from the point of morphology and corpus linguistics. The left out form of a word after removing the derivational and inflectional affixes is known as the root word. Root word generation techniques are mainly used to increase the efficiency in information retrieval systems by increasing the recall value (Recall is the fraction of the documents that are relevant to the query that are successfully retrieved) and in spell checkers to choose the correct word reference form.

In Assamese, nouns are inflated for case and noun roots are followed by some inflectional suffix (Classifier- Singular/Plural, Feminine, Case, Degree) or an emphatic suffix, or a post position (*Fig 5.4* of *Page 53*).

Similarly, verbs are either finite or infinite and inflected for tense, person and number. The morphological composition of verb finite main (VFM) can be depicted as below (*Fig 5.5* of *Page 53*).

Also due to the derived, inflected and concatenation forms of suffixation a

NUMERAL PREFIX	QUANTIFIER	ROOT	TYPE	PERSONAL MARKER	CLASSIFIER SUFFIX	FEMININE SUFFIX	CASE SUFFIX	DEGREE SUFFIX	POST POSITION	EMPHATIC SUFFIX
NUM এ-, দু-	QF ইমান-	type here	NC NP NLOC NM NA NCL	PM -রা	CL.SG -জন -জনী -গরাকী -টা -টো -টি -খন -জোড়া, etc. CL.PL -হঁত -রোর -বঁলাক -মথা -জাক -সকল -গাল	DAT -লে	NOM -ে -ই ACC -ক GEN -র LOC -ত DAT - লে	COMP -কে	POSTP পরা	EMP -হে -ও -ই -েই -টো -ে

Figure 5.4: Internal Structure of Noun Phrase

NEG	ROOT	TYPE	CAUSATIVE SUFFIX	ASPECT	TENSE	PERSON	IMPERATIVE	MOOD
NEG ন- নো- না- নে- নি-	type here	V	CAUS -া -ুলা -োরা -িয়া	IPFV -িছ	PST -ল -িল	1 -ও	IMP	CTF -হঁতেন -হয়
					FUT -ব -িব	2 -ে	-ক	
						2 -া		
						2 -ব		
						3 -ে	-ক	

Figure 5.5: Internal Structure of Verb Finite Main

number of words can be formed from a single root word. Instead of storing all the variants of a word in the database, we can store only the root word in our root database. And using our exhaustive list of affixes we can generate the variants. In this approach we can not only reduce the database length but also the time complexity of POS tagging. Also, if some new words are encountered with these affixes, then the POS tagger can break the new word based on the existing root words and affixes.

We have applied the a priori knowledge of language experts and the affix (either a prefix or a suffix or sometimes both, if available in a word) stripping method

with some language specific rules for creation of the root words of the Assamese language. To begin with, a small set of root words were entered manually by experts. The root words found during the manual analysis of a small 2,500 words corpus were stored in the "root table". This resulted in 155 root words (100 Nouns, 20 Adverbs, 15 Adjectives and 20 Verbs root words) being inserted in the table. We have inserted around 1700 more root words from the dictionary created during RCILTS phase-I resulting in a total of 1855 entries. The dictionary contains mainly root words with the grammatical categories like Noun, Adverb, Adjective and Pronoun etc. But each word of the Verb category contains about 20 variants, inflected forms of root verb word based on tense, person, and number. Since we want to keep only the root words on this table, we have added only the root verb words. We have inserted the root words from the dictionary as these resources are already available in the dictionary and will definitely affect the POS tagger performance upto some extent. Using this base table, we applied the algorithm described in the next chapter to create new root words. Such newly generated words were put in a temporary table for further verification by experts and for each word a decision was taken whether to enter the new word into the root table or not. Thus the process of generating the root table was a semi-automatic process and was much faster than a purely manual process.

5.4 Generation of Prefix and Suffix Tables

In general, a word can be formed with the concatenation of prefix(es), root word(s) and suffix(es). We have created an exhaustive list of suffixes and prefixes along with the most suitable grammatical category for each affix using the a priori knowledge of experts and by consulting books on Assamese grammar ([74], [75], [76], [77], [78]). Assigning a grammatical category to affixes is not necessary, but we did it to make the generated resource more meaningful and useful for other NLP activities.

Suffixation is a major morphological phenomenon. Derived, inflected and concatenative forms of suffixation are available in Assamese (*Fig 5.6* of *Page 55*). We have designed a rule for suffixes, S, in such a way that it can accommodate all these. Also depending upon the top down or bottom up approach S may be either sS or Ss in concatenative forms.

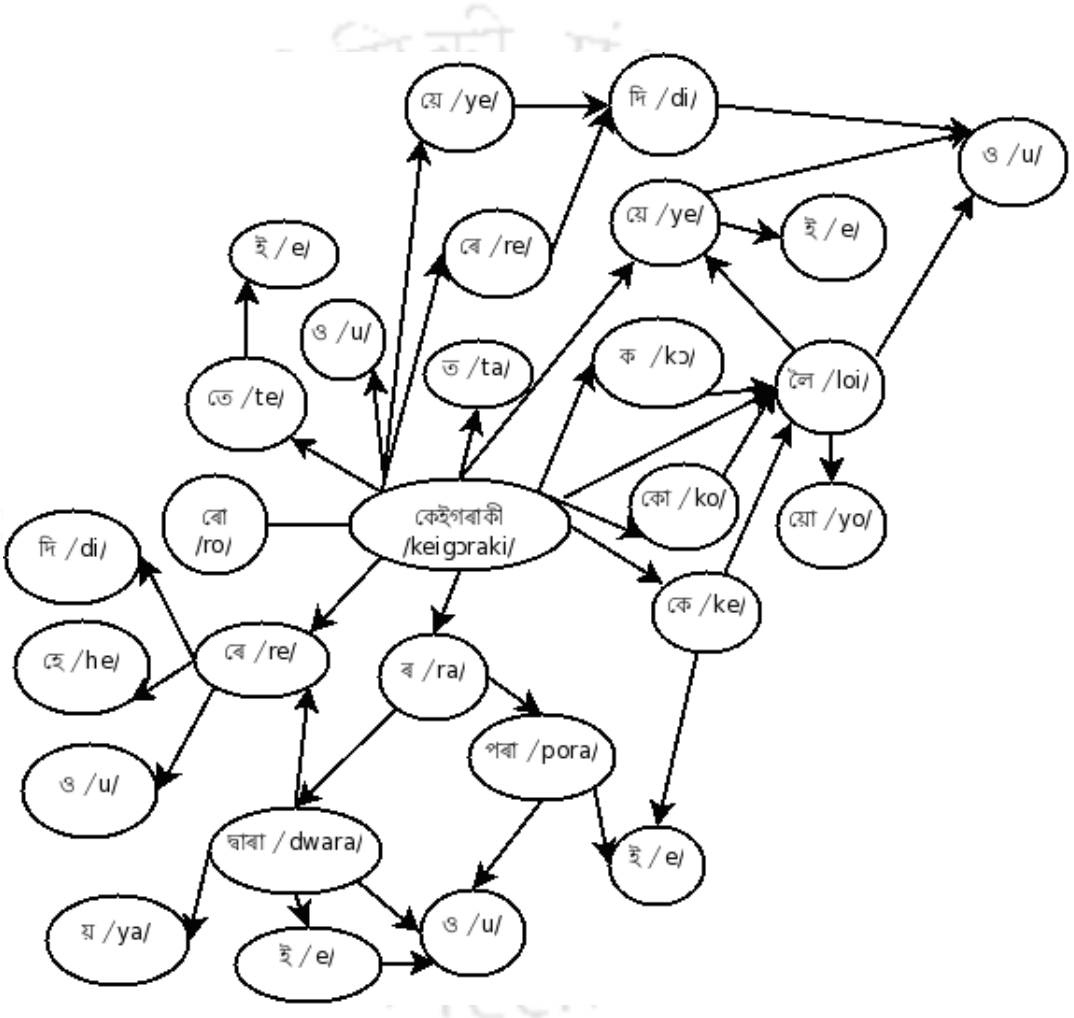


Figure 5.6: Pattern of suffixation

We can write the grammar as

W → PRS | PR|RS|R|PS

P → Numeral_Prefix| Quantifier|Sanskrit_Prefix|অ /a/|
আ /aa/|উ /u/|অনা/ana/ |আও /aau/|নি /ni/|ɛ

R → rR|Rr| r|ɛ

S → sS|Ss| s|ɛ

Numeral_Prefix → এ/a/ |দু/du/ |ɛ

Quantifier → ইমান/eman/ |ɛ

Where

W: Word P: Prefix non-terminal R: Root non-terminal

S:suffix non-terminal s: suffix r: root word ɛ: epsilion

Suffixation is an iterative process. The word মানুহকেইজনৰপৰাও /manuh-keijɔn-ɔr-pɔra-u/ *"from all the men also"* can be annotated in the following way:

মানুহকেইজন - ৰ - পৰা - ও

Man PL.CLASSIFIER - POSSESSIVE - from(POST POSITION) - EMPHATIC

The word মানুহকেইজনৰপৰাও /manuhkeijɔnɔrpɔrau/ is a result of suffixation. Using the grammar, we can derive the same word as below:

W → RS	[মানুহকেইজনৰপৰাও]	/manuhkeijɔnɔrpɔrau/
W → rS	[মানুহ-কেইজনৰপৰাও]	/manuh-keijɔnɔrpɔrau/
W → rsS	[মানুহকেইজন-ৰপৰাও]	/manuhkeijɔn-ɔrpɔrau/
W → rssS	[মানুহকেইজনৰ-পৰাও]	/manuhkeijɔnɔr-pɔrau/
W → rsssS	[মানুহকেইজনৰপৰা-ও]	/manuhkeijɔnɔpɔra-u/
W → rssssS	[মানুহকেইজনৰপৰাও]	/manuhkeijɔnɔpɔrau/
W → rsssssɛ	[মানুহকেইজনৰপৰাও-ɛ]	/manuhkeijɔnɔpɔrau-ɛ/
W → rssss	[মানুহকেইজনৰপৰাও]	/manuhkeijɔnɔpɔrau/

Assamese has a limited number of prefixes including the basic twenty prefixes of Sanskrit origin. Some more prefixes of Assamese are অ /a/, আ /aa/, উ /u/, অনা/ana/, আও /aau/ (modified from of অৱ /aawa/), and negation prefix নি /ni/ and

its variants like নো, না, নে, নি (/no/, /na/, /ne/, /ni/) etc. In the Noun form, there exists another two types of prefixes- namely Numerical (eg. এ /a/,দু /du/) and Quantifier (eg. ইমান /iman/).

5.5 Tuning of the Tagger

We decided to tune the tagger by observing its functioning on small data sets. Based on our observations, we added the "proper noun" and "common noun" tables into the database. We used four small sets of texts set-A, set-B (D1 corpus), set-C (D2 corpus) and set-D (D3 corpus) to check the output of the POS tagger developed by tagging one set at a time. Set-A (initial 200 words from corpus C1-GA) consists of sentences which have already been tagged by experts and inserted into the database. The sentences of set-B and set-C consist of sentences collected from various sources. Set-D consists of sentences which are not included in the database but were manually tagged by language experts. The details are shown in the following table table *Table 5.2* in *Page 58*.

Table 5.2 Output of the tagger for four text sets

Sets	Size of Sets (in words)	Step-I output			Step-II output (without human intervention in step-I)	
		Words with Single tag	Words with Multiple tags	Unknown words with Single tag	Errors (in words)	Accuracy
Set-A	200	200	0	0	0	100%
Set-B	195	78	48	69	17	91.28%
Set-C	83	31	24	28	15	81.92%
Set-D	212	93	54	65	74	65.09%

5.5.1 Confusion Matrix calculation

A confusion matrix is a table where each cell [i,j] indicates how often label j was predicted when the correct label was i. Thus, the diagonal entries (i.e., cells [i,j]) indicate labels that were correctly predicted, and the off-diagonal entries indicate errors. The confusion matrix, indicating the common errors, of the above four sets are shown in the table below, *Table 5.3* in *Page 59*, *Table 5.4* in *Page 59* and *Table 5.5* in *Page 60*. Since the Set-A was analyzed and tagged by experts, we did not create the confusion matrix for the set.

Table 5.3 Confusion Matrix for set-B

	NN	NNP	CC	RB	VAUXN	JJ	QF	RP	VFM
NN	X	3 (1.53%)				1 (0.5%)			
NNP	8 (4.1%)	X							
CC			X				1 (0.5%)		
RB				X				1 (0.5%)	
VAUXN					X				1 (0.5%)
JJ		1 (0.5%)				X			
QF							X		
RP								X	
VFM									X

Table 5.4 Confusion Matrix for set-C

	NN	NNP	JJ	VNF	VNN	VJJN	QF	VNNN	NNC	NNPC
NN	X									
NNP	3 3.6%	X X							1 1.2%	
JJ	5 6.02%		X X							
VNF				X X	2 2.4%					1 1.2%
VNN					X					
VJJN						X X		1 1.2%		
QF	1 1.2%						X X			
VNNN								X		
NNC									X	
NNPC				1 1.2%						X

Table 5.5 Confusion Matrix for set-D (smaller form of the matrix to reduce the matrix size)

	NN	NNP	NVB	NNC	JVB	PRN	VNF	VFM	VJJ	PREP
NN			1 0.47%	4 1.88%						
NNPC	10 4.71%	1 0.47%								
NNP	29 13.67%									
JJ	3 1.41%				3 1.41%	1 0.47%				
NVB	8 3.77%				1 0.47%		1 0.47%			
RB	1 0.47%									
VJJ	1 0.47%									
PREP	1 0.47%									
NNC	1 0.47%									
VNF								1 0.47%	2 0.94%	
JVB									1 0.47%	
CC										1 0.47%

We have observed that most of the errors have occurred for the tags Proper Nouns (NNP), Common Nouns (NNC) and Compound Proper Nouns (NNPC). All those words were tagged as Noun (NN) by the tagger. Few of the words were tagged as NN for JJ. Also, we have observed that few of the words were tagged as Verbs for Nouns as those words contain verb suffixes.

In Assamese (can be said to hold for most Indian languages also), there is no specific marker for proper nouns in orthographic conventions. Also, all the words which occur as proper nouns can also occur as common nouns denoting a lexical meaning. Proper nouns can be correctly tagged by manual tagging by looking at the context of the word, but this is difficult for a machine. For the same reason, we have observed from the above three matrices that NNPC and NNP words were tagged as NN by the tagger. But once those words are tagged as NNP or NNPC manually and saved in the respective databases, the errors get reduced drasti-

cally. Some of the NNP and NNPC words which are in the databases already were tagged properly as NNP or NNPC.

To reduce this Proper noun- Noun conflict (NNPC, NNP and NN tag conflict), we have created a new database table named proper_noun table and inserted a few popular names of people/ brands etc. The outline of the proper noun algorithm is as below:

Algorithm 1 function (Proper_Noun)

```
for each sentence do
  count=number of tokens
  for (i=0, i<count;i++) do
    search in the proper_noun table for each token
  end for
  if consecutive two or more tokens exists then
    assign tag NNPC to (n-1) tokens and tag NNP to n^th token
  end if
end for
```

Since the Proper noun – Noun conflict was maximum in set-D, we have inserted the texts of the set once again into the tagger after populating the proper noun table and found much better results this time and the confusion matrix is as follows in table *Table 5.6* of *Page 62*.

It has also been observed that (only once in the three sets) due to the very low bigram value between a pair of tags in the bigram table, words were assigned wrong tags by the tagger.

Table 5.6 Confusion Matrix D after reducing Proper noun- Noun conflict

	NN	NNP	NVB	NNC	JVB	PRN	VNF	VFM	VJJ	PREP
NN			1 0.47%	4 1.88%						
NNPC	3 1.41%	1 0.47%								
NNP	9 4.24%									
JJ	3 1.41%				3 1.41%	1 0.47%				
NVB	8 3.77%				1 0.47%		1 0.47%			
RB	1 0.47%									
VJJ	1 0.47%									
PREP	1 0.47%									
NNC	1 0.47%									
VNF								1 0.47%	2 0.94%	
JVB									1 0.47%	
CC										1 0.47%

5.5.2 Reducing Compound Common Noun – Noun Conflict

During our testing with the tagger, it was found that in Assamese, clustering of two or more words can result in a Compound Common Noun (NNC & NN) (eg. ঘিলা পিঠা /ghila pitha/ etc). But the tagger tagged them as NN. Hence we have adopted the same procedure like proper noun case to resolve the compound common noun-noun conflict.

We have created another table, common_noun table, in our database and the outline of the algorithm is as below:

Algorithm 2 function (Common_Noun)

```
for each sentence do
  count=number of tokens
  for (i=0, i<count;i++) do
    search in the common_noun table for each token
  end for
  if consecutive two or more tokens exists then
    assign tag NNC to (n-1) tokens and tag NN to n^th token
  end if
end for
```

5.6 Database Tables

The knowledge base created for the POS tagger is stored in the database tables. A summary of the tables in the database is given below. There are eight tables. The first three tables – prefix, suffix and root tables were generated from the a priori knowledge of the language. The bigram table was created from the base tagged corpus. The proper noun and common noun tables were created to resolve the conflict between proper and compound proper nouns, and between noun and compound common noun words. These six tables are the basic tables consisting of language specific information. The output table stores the output data of the tagger for further analysis by language experts and the submitted table stores the expert verified word tag pairs.

Prefix table:

The prefix table contains the basic Assamese language prefixes discussed in section 5.4. The table contains 26 prefixes.

Suffix table:

We have created a list of suffixes containing 1092 entries in the table together with the relevant tags. We have collected the derivational, inflectional, emphatic and concatenative forms of suffixes and tagged all the derivational, inflectional or emphatic suffixes for nouns as NN and verb inflected suffixes as V. Some of the

inflected suffixes were found for pronouns also and these were marked as PRP in the suffix table.

Root table:

The root table initially contains around 1885 root words of the language along with the best possible tag(s) for each word. Out of this, 155 words were inserted by experts by manually analysing some text. The rest 1700 words came from the dictionary already available with us. Few hundred more root words were inserted by the experts during verification of the tagged corpus generated by the POS tagger during the experiment making thecurrent root word table's size to be around 2200.

Bigram table:

This table contains the bigram values of two corresponding tags. The tag pairs were taken from the C1 corpus of 2,50,000 words. The table contains around 1900 entries.

Proper Noun table:

This table contains around 30 commonly used names of persons and brands as discussed in section 5.5.1 to reduce the proper noun- noun conflict.

Common Noun table:

This table contains common and compound common nouns as discussed in section 5.5.2 to reduce compound common noun-noun conflict. The table is empty at this stage.

Besides the above tables which were created manually, the following tables are also in the database.

Submitted table:

This table contains word-tag pairs of the tagged corpus created by language experts. The table contains 42,681 words and their corresponding tags. It is used to check whether the words of the newly inserted text for POS tagging has already been tagged or not to reduce the analysis time of the tagger. The table was created by taking the C1-GA corpus whose text was from a novel. The C1-GA corpus had about 87,639 words, and 42,681 remained after duplicates were removed.

Output table:

This table contains the words from the input texts which are not available in the root or submitted table with the system assigned tags. The table also contains the word-tag combinations for the words which were defined by the users if the user does not agree with the system defined tags for some words during the native user verification process using the User Interface described in the next chapter. The data of this table will be analyzed by the language experts in later phases and will be added to the submitted and root table after verification. The table is empty at this stage.

Though we are using eight tables in our approach, only four, namely prefix, suffix, root and bigram tables are the prime tables. Our approach can start working once these four tables are ready. We are using the other four tables to increase the efficiency of our approach. It has to be noted that mostly the a priori knowledge of the language has been used for the generation of the knowledge base of the POS tagger. Hence the creation of the base requires comparatively less involvement of cost and time. However, the availability of the C1 tagged corpus played an important part in improving the accuracy as the bigram table and the submitted tables got populated. The root table's size was increased by using an existing dictionary.

If more dictionaries become available in future, the root table's size can be easily increased. As the next chapter shows, the size of the root table has a positive impact on accuracy of the tagging process. The number of entries in these tables can increase over time as the experts can update those tables using the expert verification interface described in the next chapter. Since only language experts can update the database tables, the possibilities of erroneous entries in the tables are low.

Chapter 6

Implementation Details

We are trying to develop a complete system starting from text submission, word-frequency calculation of the submitted text and POS tagging with native user's verification and language expert's cross validation.

6.1 Support for other languages

We have found that English and/or Hindi language words and numerals are inserted into some of the input texts, mainly in news papers, web sites and school text books in between Assamese sentences. So, we have taken a small step to tag those words automatically during the tagging process by labeling them as foreign words or quantifiers.

6.2 Root Word Generation Algorithm

To find the root word of a word, the word is stripped from left-to-right. i.e. from starting to ending of the word. Initially it will check in the root database for availability. If the inserted word is found in the root database, it will display that. If no match is found then we try to find the prefixes first from the prefix database. If any prefix is available in the inserted word, we remove that prefix from the word.

Concatenation of prefixes is not as common as it is for suffixes. We have found concatenation of a maximum of two prefixes for a few words. After removing the prefixes, we try to find out the longest matching suffix available in the remaining word. If any match is found in the suffix list we remove that part from the word. The leftover part is considered as the root word of the inserted word. The newly generated root word is then checked against the root database for further correctness. If the newly generated root word is present in the database then the whole word is stored in a different database called the output table with the corresponding tag. Else, the generated root word is stored along with its prefix(es), suffix(es) and the corresponding tag in the output table for further verification.

We also try to store the preferred grammatical category of the newly generated root words based on its affixes. But a single suffix may be concatenated with multiple root words of different grammatical categories. For example the two words মানুহৰ/manuhor/ meaning "*male human*" and তাইৰ/tair/ meaning "*her*" have the same suffix ৰ/r/ while the root word মানুহ/manuh/ is noun and তাই/tai/ is pronoun. So, based on the root word and suffix category, the probable category (ies) of the newly generated root word is decided.

Figure *Fig 6.1* of *Page 69* gives the flowchart of the algorithm. As can be expected, there are many exceptions. So a set of rules were created to capture these exceptions and these are applied before it is decided that the result is a new root word. The rules are described in the next section.

It has to be mentioned that this system is highly dependent on the prefix and suffix databases. Hence, we are allowing the addition of new affixes to these two databases by language experts whenever new affixes are encountered by them during analysis (see below).

A root word may also be the concatenation of another root word and suf-

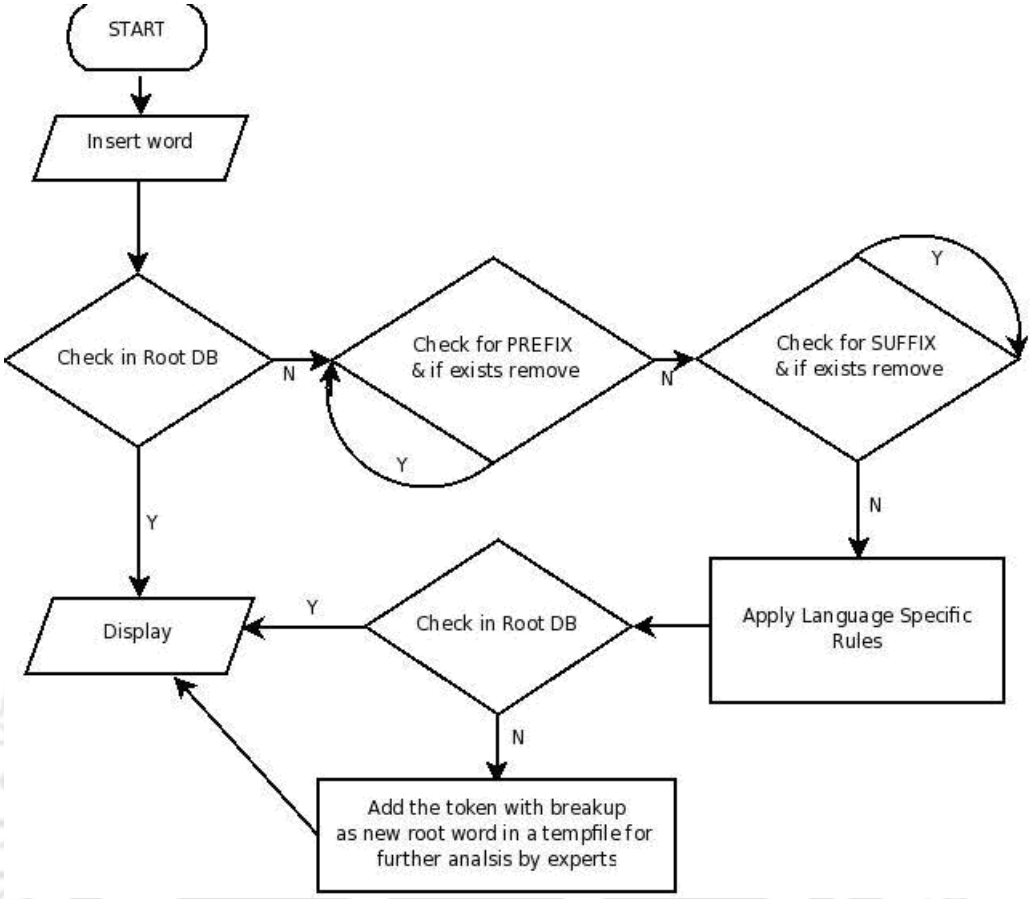


Figure 6.1: Flow chart of root word generation

fixes. For example suffix "ই" /i/ can be added after the root word "খেত" /khet/ to generate a new word "খেতি" /kheti/. Another suffix, "য়ক" /yak/ can be added to the word "খেতি" /kheti/ to form another word "খেতিয়ক" /khetiyak/. All these three words have individual and specific meanings and hence need to be considered as three different root words. The system is unable to find such root words if the other root word already exists in the database.

The root word generation technique described in these sections is also used during the tagging process itself. From an untagged word, a root word is determined by this technique, and based on the root word that is obtained, a tag is given to the word. Other details of the tagging process are described in the next chapter.

6.3 Exception Rules

During the evaluation of the words of the corpus, we have encountered some exceptions and similarities between various words. Based on these observations and a priori knowledge, we have created some language specific rules to generate correct root words. It has to be mentioned that these rules are formed based on the computational point of view and sometimes may not be seem logical if viewed as formal grammatical rules.

We have found that some words are formed with the combination of prefixes and suffixes only. In most of the cases these are the quantifiers like "এটা" /eta/ meaning one. We have added special rules for such cases.

Also there are hyphenated words which are written using a hyphen in between. Looking at the characteristics of such words we have tried to break the second word by affix stripping. An example word is ঢলং-পলংকৈ [ঢলং-পলং+কৈ] /dholong-pongkoi : dholong-polong+koi/. Similarly, in Assamese, suffixes are added to the second word of the duplicate words. We have analyzed the second word by affix stripping method like the hyphenated words, লাহে লাহেকৈ [লাহে লাহে+কৈ] /lahe lahekoi: lahe lahe+koi,meaning slowly-slowly/.

Consider the following example. The four words নালাগে /nalage/, নহৰু /naharu/, নাহৰ /nahar/ and নিমাত /nimat/ when analysed, resulted in the following root words *Table 6.1* in *Page 70*.

Table 6.1 Word analysis based on rules

Word	Prefix	Root Word	Suffix	Our Output
নালাগে /nalage/	না /na/	লাগ /lag/	ে /ae kar/	না + লাগ + ে
নহৰু /naharu/	ন /n/	হৰু /haru/	–	নহৰু
নাহৰ /nahar/	না /na/	হৰ /hara/	–	নাহৰ
নিমাত /nimat/	নি /ni/	মাত /mat/	–	নিমাত

All the four words have started with the negative prefixes না /na/, ন /n/ and

নি /ni/. But all the root words generated after the affix striping method are not correct. There is no word in Assamese like "হৰু" /haru/. And the word "হৰ" /hara/ may have different grammatical forms. To resolve this we have added the following grammatical rules. A negation will be treated as a prefix only if the root word formed after stripping the prefix is a verb. The second rule is that a word is said to have a negated prefix only if the 2nd and 4th characters of the word are the same (for example নালাগে/nalage/, নিদিওঁ /nidiu/, নেদেখিম /nedekhim/ & নুসুধিবা /nusudhiba/. As in all natural languages, there are exceptions. We have found three exceptions against these rules and the words are নিমাত /nimat/,নিভাজ /nibhaj/ and নিলাজ /nilaj/. These three words have been added in the root database by the experts. The word নালাগে /nalage/ satisfies both the rules. The word নহৰু /naharu/ satisfies only the second rule and "হৰু" /haru/ is not a root word. So, the output is the word only i.e. নহৰু /naharu/. Similarly the word নাহৰ /nahar/ does not satisfy the second rule but the word "হৰ" /hara/ has two grammatical categories Noun and Verb. Since the second rule was not satisfied, হৰ /hara/ is considered as Noun for the word and output is নাহৰ /nahar/ only. Since the word নিমাত /nimat/ is found in the root database, the output is the word itself.

In Assamese there are few root words like যা/ja/ which changes its form with the change of tense and person. The root যা/ja/ changes to গ/go/ whenever the suffix ইলোঁ/ilo/ is added, যা+ইলোঁ =গ'লো /ja+ilo= golu/. Similarly গৈছিল= যা + ইছিল /goisil =ja+ isil/. For such roots we have considered the base form (গ'ল /gol/ and গৈছ /goiso/) as the root word together with the original root word যা/ja/. We have implemented two rules based on character manipulation to resolve such cases. But these rules are primary root word specific. One such rule for primary root word যা/ja/ is that the initial ই /i/ of the suffix is removed and depending upon the tense and person যা/ja/ changes to either গ/go/ or গৈ /goi/.

Some root words seem to contain some combination of suffixes. But if we remove those considering general rules then the remaining word doesn't contain any

meaning. Like পাবলৈ (পা +বলৈ) /paboloi (pa+boloi/),বিদ্যালয়ে (বিদ্যালয়+ে) /bidyaloye (bidyaloy+e/ etc. There are suffixes like াবলৈ /aaboloi/ and য়ে /ye/. But if we remove the said suffixes from the above two words, then the remaining words প /p/, বিদ্যাল /bidyal/ can't be considered as root words. Hence we have added some specific rules for such cases.

6.4 Experimental Results of Root Word Generation Technique

We inserted around 30 most common root verb words like খা/kha/, পা/pa/, যা/ja/ etc in the root table, besides those already available in the root table. This gave 1885 root words in the root database. The number of suffixes which can be associated with verbs is comparatively more in the suffix table. Looking at the above facts, we have taken out the 1213 words tagged as VFM from the C1 corpus and gave them as input words into the root word generation process to check the efficiency of the process. The accuracy was around 89%. That means that the process correctly generated 1079 root words. We have observed that 11% errors occurred (134 words wrongly analyzed) due to the non availability of the root words in the root database, affixes in the affix databases and implementation of lesser language specific rules in the program.

We have also tested with around 1986 non verb words (NN, JJ, CC etc). The initial accuracy was around 68%. Most of the erroneous words were ending with matras like া /a-kar/, ে /e-kar/, ি /i-kar/,ী /dirghai- kar/,ো /o-kar/ etc. These word end markers are available in the suffix list as these are used as suffixes also. So, such words were wrongly split. In the initial stage this problem was very high due to the non- availability of those words in the root database. We have also used words from different internet resources to test the accuracy of the generated system.

The root database contains only a single instance of the root word with its relevant grammatical category(ies). We have observed that we can form around 44 different variants of words from a single verb root word based on tense, person and number including the negation. For nouns and pronouns this value is much higher.

The correctness increases with the increase in the size of the root database. We have also found words formed by combining two or more root words in Assamese. Our system could not generate the corrected root words for such words.

The computational linguistic rules provide an added advantage to the system. Though the rules are language dependent, they increase the efficiency of the root word generator. Due to these rules, the size of the root table decreases because without these rules each and every exceptional word with all their forms has to be included into the root table.

6.5 Evaluation Procedure

After creating and populating the database tables, our proposed system is ready for the operation. The developed system can accept either a single sentence or group of sentences as input data. We have considered the punctuation marks – full stop (।) /dari/ and question mark (?) /prasnabudhak/ as the sentence end markers. The developed tagger analyzes the input text in two steps. The third step is used for native users' verification and experts re-validation. Here we are discussing the details of each step along with the outline of the algorithms.

Step-I:

The input text consists of multiple Assamese sentences. The text is first separated into individual sentences by looking at the sentence end marker. In general,

sentences are formed with multiple individual words with a blank space between two consecutive words. Every individual word is separated at the blank space in each sentence. We term this process as tokenization of sentences and each individual word are termed as tokens. First, the token is checked to find out if it is a number or a foreign word, and if so, appropriate tags are assigned. Then, proper noun and common noun tables are searched for a match and action taken if a match is found. Then the token is searched for in the submitted table and then, if not available in that table, in the root word table for the corresponding tags. If it is found in any of these tables, the tags stored in these tables are assigned to the token. Tokens may have single or multiple tags. If the input token is unavailable in these tables, then the root generation method described in section 6.2 is applied to generate a root word. If the generated root word is found in the root word table then the tag stored there is assigned to the token. Only one tag will be assigned, as unlike the submitted table, there is only one entry for a word in the root word table. If the generated root word is not found in the root word table, then the default tag is assigned which is the Noun (NN) tag. The pseudo-code *Algorithm 3* of *Page 75* below describes the process.

Algorithm 3 Basic Algorithm

```
while !(end of input data) do
  Separate each sentence at the sentence end marker
  for each sentence do
    break each sentence at " " from left to right to make each word as token
    for each token do
      Search in the submitted and root table for corresponding tag(s)
      if exists then
        Output=Output+ word/tag(s)
      else
        Check for digits, foreign words(FW)
        if digit then
          Output=Output+ word/QFNUM
        else if FW then
          Output=Output+ word/FW
        else
          remove prefixes(P) & suffixes(S) if available
          search in the root table for the remaining word (A)
          [function (prif_strip) & function (sufx_strip)]
          if exists then
            re-form the word(W=P+A+S) and assign tag based on P,A and
            S and language specific rules
            Output=Output+ word/tag
          else
            /* tokens neither available in db nor analyzed by the system are
            tagged as NN*/
            Output=Output+ word/NN
            insert the word/NN pair in the output table
          end if
        end if
      end if
    end for
  end for
end while
```

Algorithm 4 function (prif_strip)

```
/*discarding a character from right side of the token on every iteration*/
len= strlen(token)
while (len>=0) do
    read the token from left to right and match the longest prefix available
    len--
end while
```

Algorithm 5 function (sufx_strip)

```
/*discarding a character from left side of the token on every iteration*/
len= strlen(token)
j = 0
while (j<=len) do
    read the token from left to right and match the maximum length suffix avail-
    able
    j++
end while
```

A snapshot of the first step output is shown below in Fig 6.2 of *Page 77*:

The red coloured (without underlined) words indicate that the words are available in the databases. These words with multiple tags mean that those words may occur in multiple contexts in a sentence and hence multiple instances of those words are available in the databases. The blue coloured (underlined) words indicate the unknown words and are analyzed by the system and assigned a tag based on the analysis.

Step-II:

It has been observed in step-I that some of the words which are available in the databases were tagged with multiple tags (e.g.: কৰা/kora/##VJJ/VNF/VNN). To reduce this ambiguity we have adopted the bigram approach with a little modification. We have applied the bigram method in the output text of the step-I of the tagger. The bigram approach takes a single sentence from the output text (of step-I) at a time and tokenizes the sentence at blank spaces and processes the tokens from left to right. In this approach, the previous word ($word_i$) will always

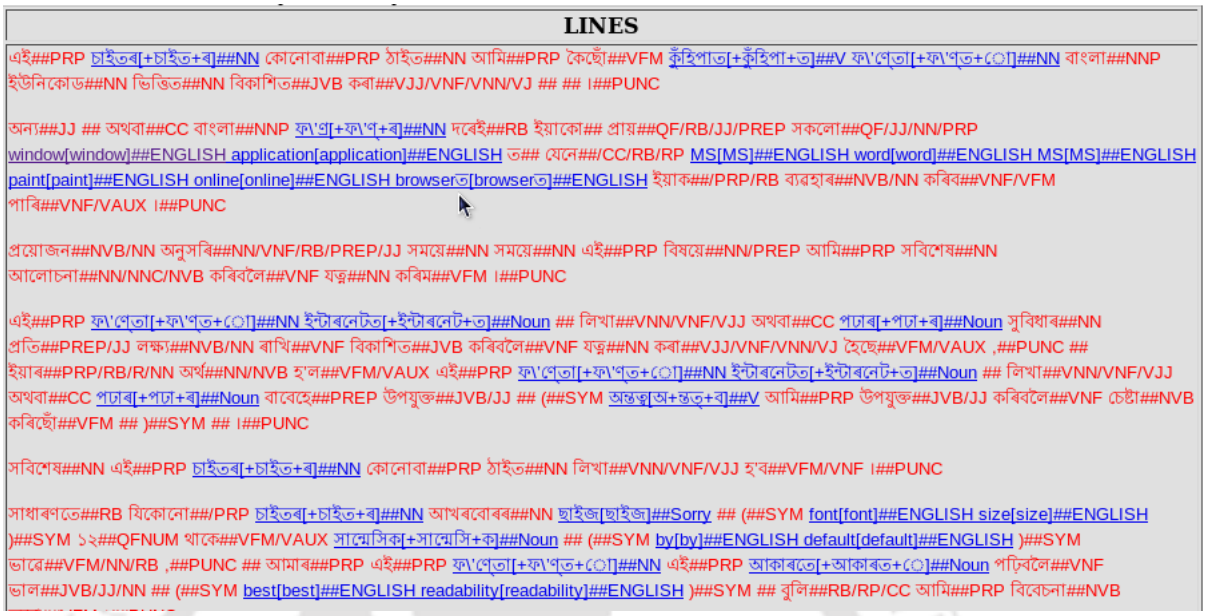


Figure 6.2: Snapshot of the Step-I

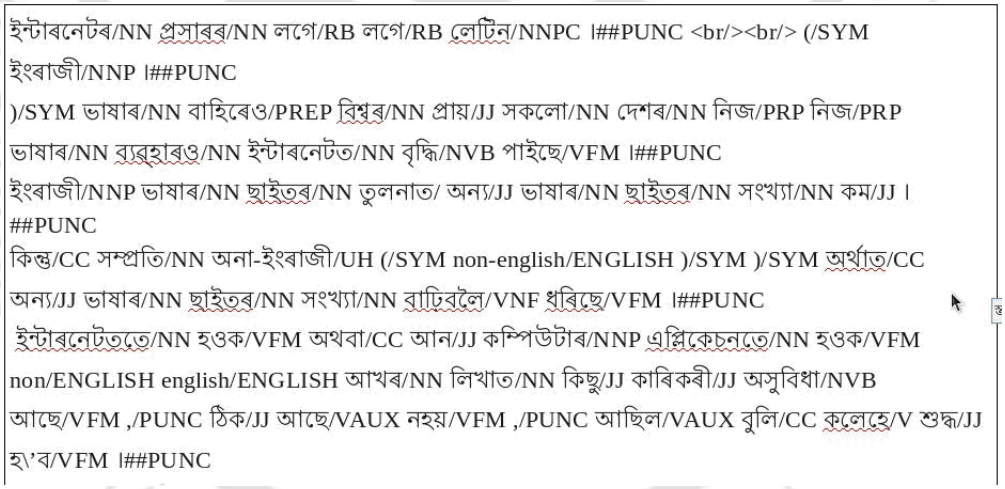
.

contain a single tag and we try to find out the best suitable tag for the immediate next word,$word_{i+1}$, which may contain multiple tags $tag_{i+1,1}/tag_{i+1,2}/tag_{i+1,3}$ etc based on the bigram values of the tag pairs $(tag_i\ ,\ tag_{i+1,1})$, $(tag_i\ ,tag_{i+1,2})$ and $(tag_i\ ,tag_{i+1,3})$. If the immediate next word contains only one tag, then the $word_{i+1}\#tag_{i+1}$ combination will be kept and the $word_{i+1}\#tag_{i+1}$ pair will be assigned as $word_i\ \#tag_i$ and same process will follow. The out line of the bigram function is described in *Algorithm 6* of *Page 78*.

A snapshot of the second step output is shown below in Fig 6.3 of *Page 78*:

Algorithm 6 Bigram

while !(end of input data) **do**
  for each sentence **do**
    tokenize the sentence for each $word_i$ / tag_{ij} from left to right
    if sizeof(tag_{ij}) of $word_i$ =1 **then**
      Output= $word_i$ / tag_{ij}
    else
      for each of tag_{ij} of $word_i$ **do**
        for each of tag_{mn} of $word_{i+1}$ **do**
          (t_{1max} ,t_{2max})=find max bigram value pair of tags (tag_{ij}, tag_{mn})
        end for
      end for
      Output= Output+ $word_i$ / t_{1max} + $word_{i+1}$ /t_{2max}
    end if
  end for
end while


ইন্টাৰনেটৰ/NN প্ৰসাৰৰ/NN লগে/RB লগে/RB লেটিন/NNPC ।##PUNC

 (/SYM
ইংৰাজী/NNP ।##PUNC
)/SYM ভাষাৰ/NN বাহিৰেও/PREP বিশ্বৰ/NN প্ৰায়/JJ সকলো/NN দেশৰ/NN নিজ/PRP নিজ/PRP
ভাষাৰ/NN ব্যৱহাৰও/NN ইন্টাৰনেটত/NN বৃদ্ধি/NVB পাইছে/VFM ।##PUNC
ইংৰাজী/NNP ভাষাৰ/NN ছাইতৰ/NN তুলনাত/ অন্য/JJ ভাষাৰ/NN ছাইতৰ/NN সংখ্যা/NN কম/JJ ।
##PUNC
কিন্তু/CC সম্প্ৰতি/NN অনা-ইংৰাজী/UH (/SYM non-english/ENGLISH)/SYM)/SYM অৰ্থাত/CC
অন্য/JJ ভাষাৰ/NN ছাইতৰ/NN সংখ্যা/NN বাঢ়িবলৈ/VNF ধৰিছে/VFM ।##PUNC
ইন্টাৰনেটতে/NN হওক/VFM অথবা/CC আন/JJ কম্পিউটাৰ/NNP এপ্লিকেচনতে/NN হওক/VFM
non/ENGLISH english/ENGLISH আখৰ/NN লিখাত/NN কিছু/JJ কাৰিকৰী/JJ অসুবিধা/NVB
আছে/VFM ,/PUNC ঠিক/JJ আছে/VAUX নহয়/VFM ,/PUNC আছিল/VAUX বুলি/CC কুলেহে/V শুদ্ধ/JJ
হ\'ব/VFM ।##PUNC

Figure 6.3: Snapshot of the Step-II

.

Step-III:

The output of step-II is then put up for native users' verification. The basic minimum criterion of the native user is to be able to read the Assamese language. We have created an interface for registration of users (`http://www.iitg.ernet.in/cc/web/pkdutta/rcilts/`) . We have provided the tagged text of step-II to only the registered users for further verification. The registered users can verify the tagged output using the user interface which has been described in section

6.4. We are restricting unauthorized access of the tagged corpus to minimize user conflicts, tagging verification of the same parts of text by multiple users and to reduce the junk tagging of tagged output data. Though we are using a limited crowd sourcing technique, we are hopeful that the system will help us getting corrected annotated corpus in less time.

6.6 Development of User Interface

In this interface, users after logging in, can change the tags of any word/tag combinations that the user thinks are not suitable for that word in a particular sentence of the output of step-II. Users can select multiple words in a sequence to tag all the selected words as proper or common noun. We have provided this feature because of the appearance pattern of the proper noun and compound common nouns. Most of the time proper and compound proper nouns have appeared in multiple words in a sentence. Using this feature user can tag multiple word sequences in a single click of the mouse button reducing the time requirement. A snapshot of the user interface is shown below in fig *Fig 6.4* of *Page 80* & *Fig 6.5* of *Page 81*.

The user has the provision of logging out of a current session so that the user can continue the work in the next session. There is a provision of timeout from the current session if the user remains inactive for a specific interval of time determined by the admin. This is done to calculate accurately the time requirement and the number of word-tag pair(s) modified by any user in a particular session or in total. After logout or timeout, the user is redirected to a page which displays the following information- the total number of word-tag pair(s) changed, the total time taken by the user in that session and the changed word(s) with old and new tag(s) in that session (*Fig 6.6* of *Page 81*).

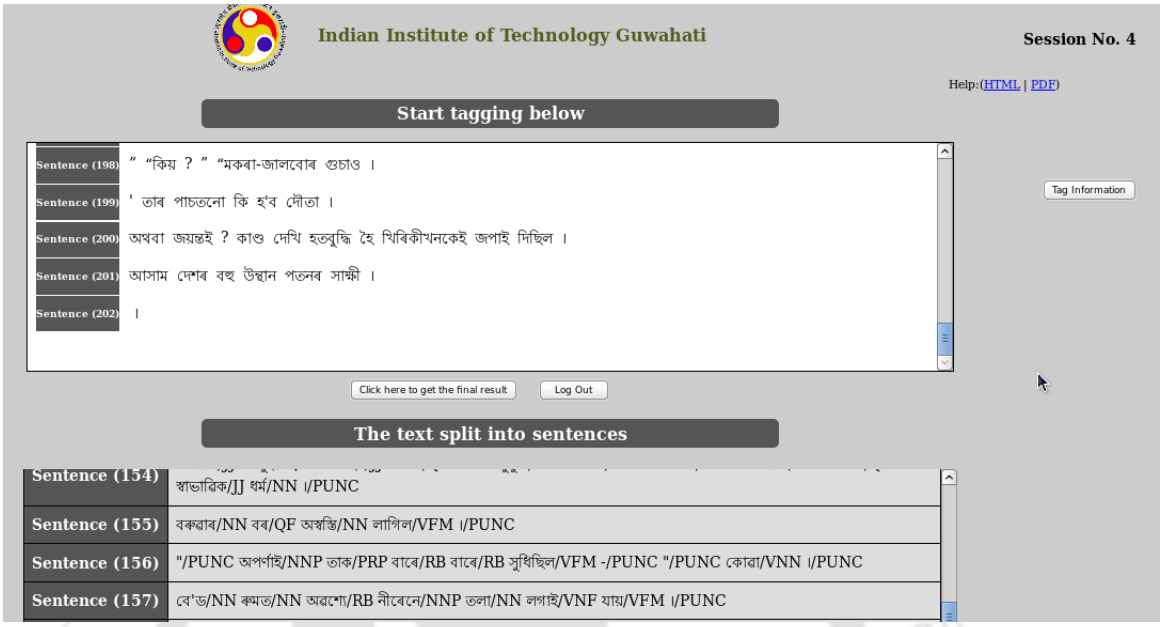


Figure 6.4: Snapshot of User Interface

Whenever the user completes the modifications of the full text, the "Complete tagging" button is enabled and clicking that button the user can finally submit the modified text. After the completion of the task, the user is not allowed to log into the page again. All the user modified word/tag combinations along with time taken to update tags are stored in a temporary database.

There is an administrator /language expert page where an expert can view/ modify the user modified word-tag pairs. After the expert intervention, words with modified tags are stored in the main database tables for future use. We have created this expert intervention as we assume that all the native users are not language experts and any incorrect word-tag combination in the main database can lead to erroneous tagging of words. The following snapshot shows the expert view of a user modified word-tag pairs. The expert can click in any of the words for further modification (*Fig 6.7* of *Page 82*).

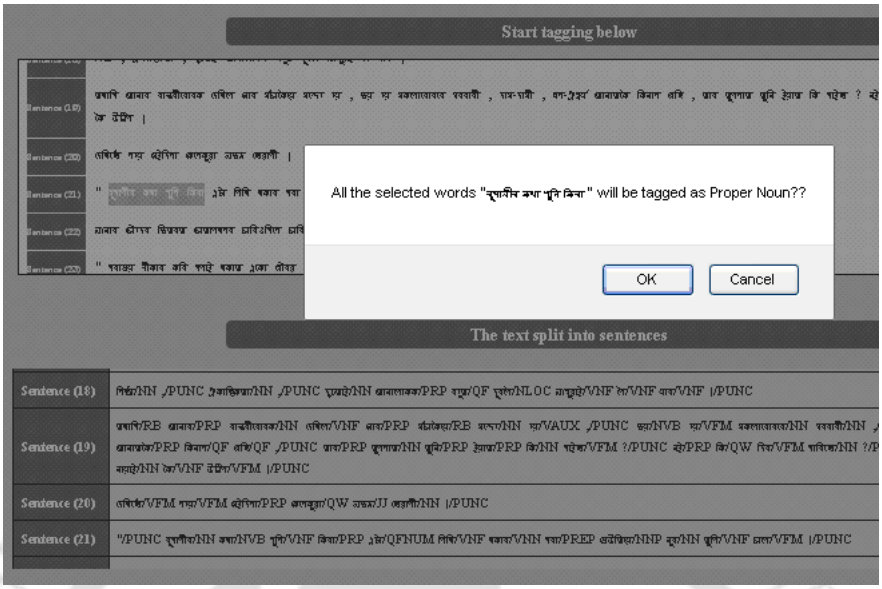


Figure 6.5: Snapshot of multiple word tagging

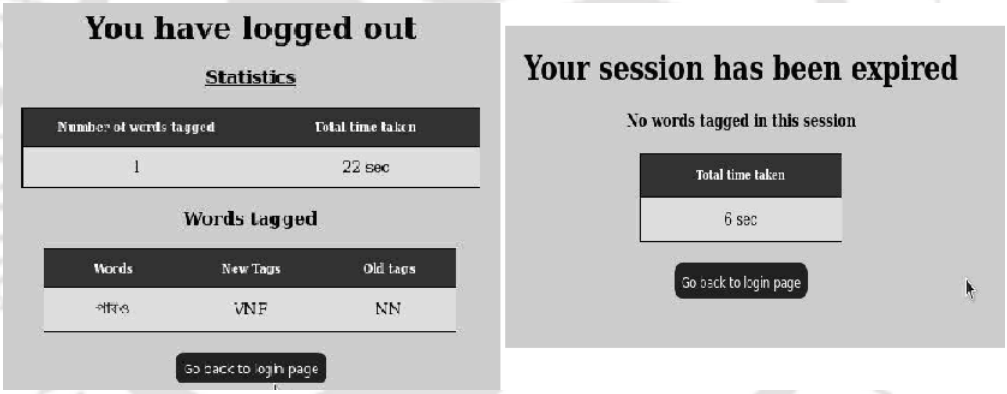


Figure 6.6: Logout/ Timeout screenshot

The language experts after login will get the following page (*Fig 6.8* of *Page 82*) where the expert can change the user assigned tag and put some extra information about the word (optional- root word, prefix, suffix etc) which will help for better analysis and create a better database of root words. It is to be noted that the intervention by a language expert at this stage is not necessary for the tagging process. If a language expert looks at the temporary table, he or she can increase the size of the knowledge base by copying the tagged words to the submitted table.

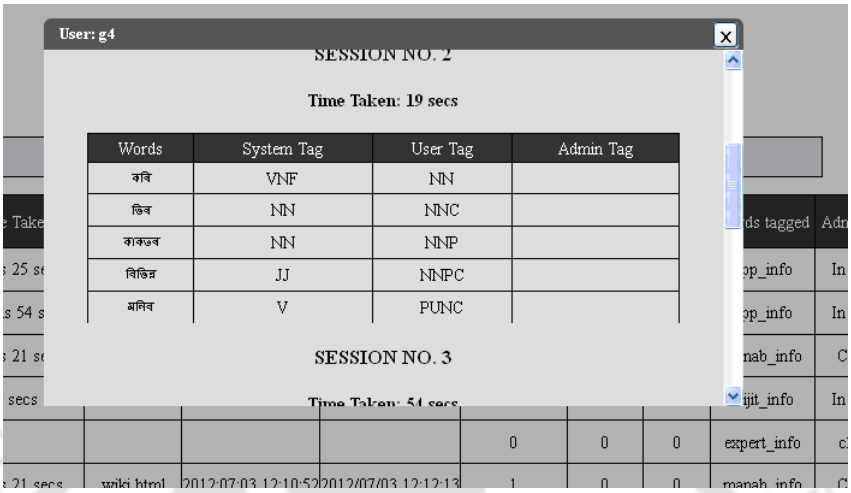


Figure 6.7: Screenshot of Expert view

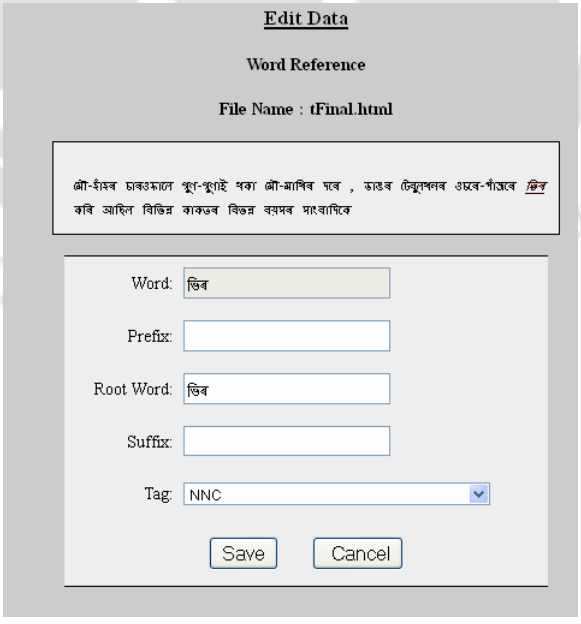


Figure 6.8: Expert Page for editing

Chapter 7

Experimental Results

In the previous chapter, we have discussed about the procedure of automatic tagging, the native users' verification on the output of the tagger and the database tables used in the process. The users play a major role in identifying the compound proper nouns, compound common nouns and proper nouns in submitted texts and also to reduce the conflict among these tags. The time required by native speakers for providing verification is a major part of the developed POS tagging process and the efficiency reduces if the time requirement is more. Considering all these factors, we have developed a user friendly, easy to use, User Interface (UI) for native users' verification to expedite the verification and modification process of the tagged output of the semi automated tagger after the second step.

We have populated the database tables with the respective information and data gathered from the analysis of language experts and apriori knowledge of the language. The prefix table contains the prefixes of Assamese language which are limited in number. The suffix table contains an exhaustive list of suffixes ranging from single character suffix to longer concatenative form of suffixes along with the best possible tags. Initially this table was filled up with the most common and various available suffixes. Then some more were inserted after the manual analysis of the 2500 corpus by the experts.

The root table contains root words of the language along with the best possible tag(s) for each word. We are keeping two tag fields for each word – the first tag field contains the best possible most common tag of the word and the second tag field contains the second best possible tag for the word. Except few words the second tag field is blank for most of the words. This indicates that the root words with two tags have the higher probability of ambiguous instances.

We have created the bigram table by inserting the bigram values of each pair of consecutive tags, tag_i and tag_{i+1} of words $word_i$ and $word_{i+1}$ of the sentences available in the tagged corpus C1. Since the corpus was tagged by the linguistic experts, we presume that the wrong tag pair is not available in the corpus.

To reduce the word analysis time using the prefix, suffix and root table we have created another table named submitted table. This table contains the expert verified word-tag pairs from the tagged corpus. Initially this table contains only the word-tag pairs of C1 corpus. In later phases more word-tag pairs were inserted from the tagged corpus generated from the experiment after verification by the experts.

The proper noun and common noun tables were initially empty. During the confusion matrix calculation we have inserted few entries in the proper noun table and more entries were inserted after the native users' verification and validation by experts from the newly generated tagged corpus by the POS tagger.

The output table was empty before the experiment. This table contains all the new word-tag pairs unavailable in either root or submitted table along with the native user modified tags so that the experts can verify those new and modified word-tag pairs.

7.1 Experimental Results using the User Interface

We have created an untagged corpus C2 containing 48,950 sentences (around 5,54,054 words) for testing the POS tagger. We have created the corpus from various short stories, novels and web articles written by different writers. We have divided the corpus into seven different sets of texts based on writers. We did this to differentiate the tagged output of different writers and to check the variations of the tagged corpus. As is to be expected, we have observed that the writing patterns of different writers are different and sometimes some specific word combinations are used by some writers.

The seven untagged text sets were tagged using the POS tagger. Six of the seven tagged sets were supplied to six native users to modify/ update the word-tag combinations as per their knowledge using the user interface (UI) that we have developed. The number of words/ tag pairs changed and time taken by the users were noted by the system and shown in table *Table 7.2* in *Page 87*. Set-III, being the smallest set consisting of only 19982 words, was manually corrected by the experts and hence further details were not recorded. The following table describes all the details. We have used a PC with only 2GB RAM and AMD phenom 8750 tri core processor during the system tagging process. The system time can therefore be reduced drastically by using a higher configuration server. The system time is therefore not of significance.

Table 7.1 Experimental results - I

Corpus	Number of Sentences	Number of words	Tags changed by users,	Time taken by users T_{User}, (hh:mm)	Time taken by system T_{System} (Minutes)
Set-I	21137	223261	759	06:45	477.27
Set-II	4418	67510	185	05:47	174.03
Set-III	1518	19982	0	—	52.3
Set-IV	10336	115933	283	05:28	182.7
Set-V	5681	57227	300	03:38	147.16
Set-VI	3435	32915	109	02:10	31.46
Set-VI	2425	37226	345	02:36	88.22
Total	**48950**	**5,54,054**	**1981**	**25:44**	**1153.14(= 19:20 hrs)**

It has been observed that most of the users have changed mainly the Proper nouns, Compound Proper nouns, and Compound Common Nouns in the corpus assigned to them. The details are given in the following table *Table 7.2* in *Page 87*.

Table 7.2 Native Users' Changes

Native Users' changes					
NEW	OLD	FREQ	NEW	OLD	FREQ
NNPC	NN	311	QFNUM	NN	2
NNP	NN	289	UH	V	2
NN	V	245	VAUX	V	2
JJ	NN	117	JJ	QF	1
NNP	V	97	JJ	UH	1
NNC	NN	86	NN	NN	1
CC	QF	37	NN	NVB	1
NNPC	NNP	36	NNP	JJ	1
NN	NNP	27	NNP	VNF	1
NNP	NNPC	22	NNPC	NLOC	1
NN	NNPC	19	NNPC	PRP	1
VFM	V	17	NNPC	PUNC	1
VAUXN	NEG	12	NNPC	SYM	1
NN	NNC	11	NVB	VNF	1
QF	JJ	11	PRP	NN	1
QF	V	8	RB	V	1
NNP	NNP	6	SYM	NNP	1
JJ	V	3	UH	CC	1
NNPC	NNPC	3			
QW	NN	3	UH	VNF	1
NN	JJ	2	VFM	NN	1
NNP	QFNUM	2	VFMN	V	1
PRP	V	2	VNF	JJ	1
			VNF	PRP	1

It was found that 381 words were entered in the proper noun and 208 words were entered in the common noun tables. Around 267 words that were coarse tagged as Verb (V)- the tag used by the system to tag words which were found to be verbs - but not available in the root word table, were changed by the users to finer tags like VNF/VFM/VNFN/VFMN etc. It has been observed that around 831 tags (approx. 60%) were changed to finer tags (eg. changing NN to NNPC/NNP/NNC or V to VFM etc) and around 561 tags (approx. 40%) were errors which were corrected by the users. We are using the term *coarse tag* to indicate the word-tag pairs that are tagged by the system based on the inflectional suffixes attached to the word when the root word generated by the system from the input word is

not available in the root table. Similarly the term *finer tag* indicates the word-tag pairs that are initially coarse tagged by the system and which are modified by the native or expert users to more accurate and meaningful tags of the initially tagged group (for example changing from NN to NNPC/NNP/NNC etc or changing from V to VFM/VNF/ VFMN/VNFN etc). Some input words are wrongly tagged based on the suffix(es) of the word in the context of the sentence or wrongly analyzed by the system and assigned some wrong tag. Such cases are *errors*.

7.2 Correctness of the newly generated tagged corpus

We used the expertise of language experts at this final stage of our evaluation approach to check the correctness of the tagging. We created a new, tagged database of 39711 sentences after the native speakers' verification. Then we randomly chose a small corpus of 500 sentences containing 5276 words from this new database. This tagged random corpus was provided to language experts to check the correctness of the corpus and our developed interface have noted down the time required by the experts to modify wrongly tagged word-tag pairs of the corpus, the word-tag pairs that has been changed by the experts and the total number of word-tag pair(s) changed after native speakers' verification. It was found that 197 tags were changed by the expert in 2 hours 21 minutes (T_{Expert}).

The 500 sentences were chosen uniformly randomly from the tagged corpus. Since different sentences may have different complexity, we needed to make sure that our sample accurately represented the mix of sentences in the original corpus. It is safe to say that short sentences are usually simple, and complexities usually arise in long sentences. So we mapped the distribution of sentences according to length for both the original corpus and the sample. Figure (Fig 7.1 of *Page 89*) shows the result. It is seen that the distribution of sentences according to legth

is very similar and this has given us more confidence to assert that the results of the smaple will apply to the entire tagged corpus.

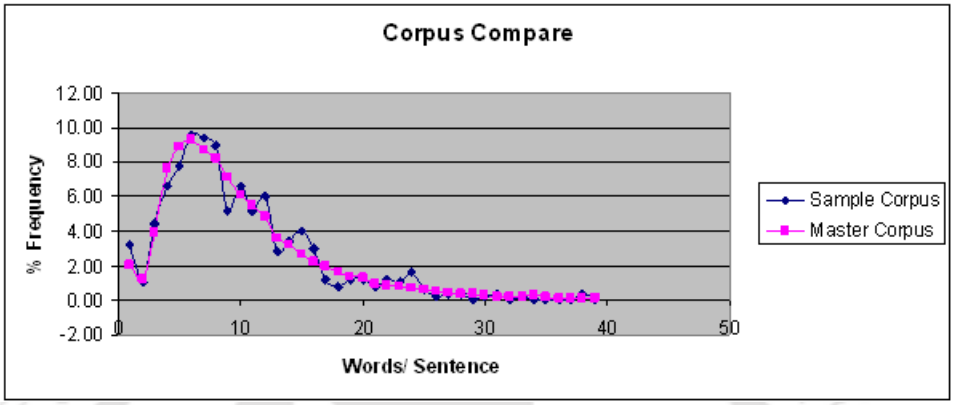


Figure 7.1: Corpus Compare diagram

We have created another small corpus C3 (443 sentences containing 7734 words). These sentences were collected from web sites. We have created this new corpus to recheck the performance of native users using the User Interface in a different contextual environment. Also, this corpus is of different domain than the previous corpus or the RCILTS corpus that we have used to train our database. Thus we have checked whether the inter domain effect prevails in our system or not.

This was fed to the tagger to generate a tagged corpus. Then the system generated tagged corpus was provided to a native user who changed 81 tags in 56 minutes and then the user changed tagged corpus was provided to an expert. The expert changed 121 tags in 1 hr 17 minutes in the user verified tagged corpus. It has been observed that all the changed tags were not wrongly tagged by the system. Some words were coarse tagged by the system. The experts have changed those tags to finer tags. We have calculated the tags changed by the users and experts, percentage of finer tagging by the experts, percentage of error in tagging, quality and fidelity of the new tagged corpus. We define the following terms described in the table *Table 7.3* in *Page 90*. The Tagger includes the three steps

described in chapter 6 and so includes the native users' step. The System Tagger refers to what the software alone has achieved without the native users' inputs.

Table 7.3 Term Defination

Term	Defination
Fidelity	100 – (percentage of words wrongly tagged by the Tagger)
Accuracy	Fidelity – (percentage of words tagged coarsely by the Tagger)
System Accuracy	100 – (percentage of tags changed by native users) – (the percentage of tags changed by experts)
System Fidelity	100 - (percentage of error tags changed by native users) – (the percentage of error tags changed by experts)

So *Fidelity* is the percentage of words correctly tagged by the Tagger, but with some tags possibly coarsely tagged. *Accuracy* is the percentage of words correctly tagged by the Tagger with coarse tags considered as errors. The *System Accuracy* is the percentage of words tagged correctly by the System Tagger with coarse tags considered as errors. Finally, the *System Fidelity* is the percentage of words correctly tagged by the System Tagger, but with some tags possibly coarsely tagged.

The detail results of the experiments are shown in table *Table 7.4* in *Page 91*. As can be seen, the fidelity is between 97-99% and the accuracy is between 96-98%. The System Quality is between 96-97% and the coarse system quality is between 98-99%. These results compare with the best results reported in the literature by automatic taggers, and they are better than any results reported for any Indian language.

Table 7.4 Results Table

	500sentence corpus/ 5276 words	443 sentence corpus/ 7734 words
% changed by Users	19/5276=0.36% (taken average of 1981/554054 = 0.36% from table 7.1)	81/7734= 1.05%
%error tags by Users	0.14% (40% of all changes: see table 7.2)	0.42%
%Finer tags by Users	0.22%	0.63%
Tags changed by Experts	197	121
% changed by Experts	197/5276 = 3.73%	121/7734=1.56%
% Error tags by Experts	1.61%	0.81%
% Finer tags by Experts	2.12%	0.74%
Fidelity = 100- % Error	98.39%	99.19%
Accuracy = Fidelity - %Finer	96.27%	98.35%
System Accuracy = 100 - %changed by Expert - % changed by user	100 – 3.73% - 0.36% =95.91%	97.39%
System Fidelity = (100 - % error by Users - % Error by Experts)	100- .14% -1.61% = = 98.25%	100 - 0.42% -0.81% = 98.77%
Expert scanning rate to achieve 100% accuracy	5276/(141/60) = 2245 words / hour	7734 / (77/60) = 6027 words / hour

7.3 Comparison of Annotation time with Manual Tagging

The time required for annotation can be measured by keeping track of the time taken by annotators while tagging a sentence. We have compared the annotation time taken by the system to generate a tagged corpus followed by native users' review, with the time taken by linguistic experts to tag a corpus using Sanchay software.

It has to be mentioned that the usability of the software and expertise of the user on the tagging software is a major factor of time analysis. It was found that the speed of tagging verification was comparatively slow at the initial stages and as the users got accustomed with the system the speed increased. So we discounted the "learning" time of users to get more realistic results.

We have calculated the number of words tagged by linguists per hour using the Sanchay software developed by IIIT Hyderabad. The linguists were well accustomed with the software during the tagging of the RCILTS corpora. We took two linguistic experts with three different set of corpora varying from common frequently used simple word sentences to longer conjugated sentences with less frequently used words and it was found that around 58 words were tagged by the linguistic experts at their best speed. The following figure,(Fig 7.2 of *Page 92*), depicts the number of words tagged by the experts in the three different corpora.

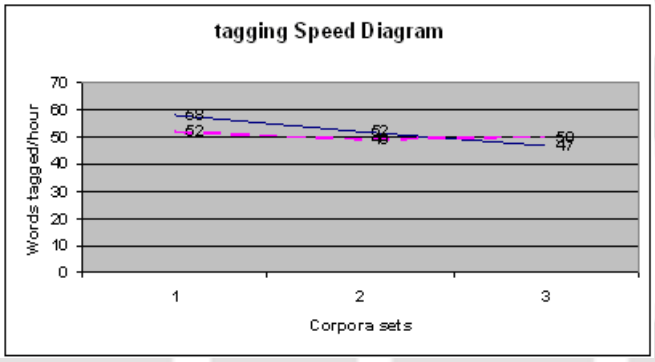


Figure 7.2: Tagging speed diagram

We have added the total time required by the system and time required by the native speakers to update the word-tag pairs to calculate the total time (T_{Total} = T_{System} +T_{User}) required by our approach. Then we divide the total words of the corpus by the total time (T_{Total}) [554054/45=12312]. It was observed that around 12312 words were tagged per hour by our approach, which is more than 200 times faster than expert's manual tagging system.

We have also compared the annotation time after adding the expert's verification time (T_{Expert}) with the total time (T_{Total}). We have calculated the probable maximum time requirement by the expert's to verify the correctness of the full corpus of 554054 words based on the time required by the experts to check the random corpus of 5276 words and the expert time (T_{Expert}) requirement comes up

as 247 hours [(554054/5276)*141/60]. We have added the value of T_{Expert} with total time ($T_{Total} = T_{User} + T_{Expert}$) and found that around 2029 words were tagged per hour by our approach, which is more than 34 times (2029/ 58) faster than expert's manual tagging system. So even if we include the phase of verification by experts, our tagging system is much faster.

7.4 Analysis of the results

We have observed from above two sections that the applied POS tagging approach yields better results than many of the standard statistical approaches discussed in chapter-II. The involvement of native users has been providing an added advantage to the system. The base of our approach is the a priori knowledge of the language experts. Hence almost all the major database table entries are accurate and the probability of containing erroneous information is almost nil. We have calculated the bigram values between the pair of tags from the RCILTS corpus which is manually tagged by the linguistic experts. Hence the probability of wrong tag pair bigram values is negligible. The output of the step-I of the POS tagger may assign multiple tags to a single word creating an ambiguity problem. To remove this, we have applied bigram approach in step-II into the output of step-I. We have applied this approach per sentence basis reducing the unseen word problem of bigram taggers. Since we are using NN as the default tag, all the words of a sentence were tagged with only one tag. The bigram approach has played an important role in removing the ambiguous tags of words. Words which contain more than one tag are considered as ambiguous and after analysis of the output of step-I of C2 corpus, it was found that around 1,43,000 words (approx 26%) were ambiguous. The step-II has reduced the ambiguity based on the bigram table and thus it has contributed to the increase in the accuracy of the tagger.

The proper and common nouns available in the submitted table are accurately tagged in step-I. The presence of compound proper nouns and common nouns is

then checked per sentence in the output of step-I using the proper and compound noun databases and if found then the word sequence is tagged as per compound proper and common noun. If the input text contains some new compound or proper nouns which are not available in the respective database tables, those were tagged as per the bigram approach. The native speakers have accurately tagged the newly available proper, common, compound proper and compound common nouns in the sentences using the user interface. Since we have provided sentences to each native user, they can easily find out the compound proper and common nouns based on the context. Hence the ambiguity based on context has also been removed. But in case of the statistical approaches the compound proper and common nouns also drastically reduces the accuracy. The databases are updated only after the expert verification of the native user's input. Hence the probability of a wrong tag entry into the databases is drastically reduced. The experts need only view the words in which a native user has a disagreement with the system tag. This reduces the involvement time of experts also.

It has also been observed that the inter and intra domain effect is negligible in the approach whose effect is very high in stochastic approaches as we have observed during the use of NLTK.

Due to the above mentioned reasons, the POS tagger is performing better than the statistical methods. In statistical methods, the training set plays a major role. By using a hybrid method, we have obtained better results, with no single parameter becoming critical in determining the accuracy of results. We have used a fairly large knowledge base and quite a bit of work was put into analysing suffixes, and in framing language rules to deal with exceptions. However, this was a one time effort, and our results indicate that we can obtain similar accuracies on more copora without any significant addition to the knowledge base.

We have tagged another corpus, C4, of 2,50,000 words using the knowledge

base and the programme. Native User Verification is proposed with crowd sourcing on the net. We are on the look out for more corpora on the net, as creating a corpus by typing in text is an expensive and slow process.

Chapter 8

Conclusion and Future Work

In the thesis we have proposed a method to create a large Assamese language tagged corpus in a short time with small intervention of linguistic experts. The contributions of the thesis are summarized as below.

8.1 Contributions

Given a base tagged corpus, a database of tagged words was created. Tables containing root words, prefixes, and suffixes were created. Grammar specific rules were added to handle exceptions. A computer programme was developed which tags new text by looking up these tables. Ambiguities in the output are then removed by the use of a bigram table. This table was created using the same base tagged corpus. At the final stage, a native, non-expert speaker examines the machine tagged output and makes corrections as deemed fit.

The accuracy of the scheme has been estimated by portions of the tagged corpus being further checked by an expert. The errors found by the expert are used to estimate the accuracy of the main process.

A new approach has been tried during the root word generation technique by

trying to store the preferred grammatical category of the newly generated root words based on its affixes.

An online Assamese tagged corpus of more than 550,000 words has been generated and is available for research purposes. Another 250,000 words have already been tagged and it is proposed to use crowd sourcing on the net for the native users' stage.

The tagging of 5,50,000 words yielded an accuracy of 96.27%. The fidelity (where some of the tagging may be coarse) obtained was 98.39%. Even without human intervention, the system accuracy obtained was 95.91%, and with coarse tagging allowed, the system accuracy rose to 98.25%. These results compare with the best results reported in the literature for any language. These results are better than any result report so far for Indian languages. We started out with the premise that some human intervention is necessary to obtain acceptable accuracies. While the results show that accuracy does increase when native users are used, the system by itself has done very well. The improvement is only from 95.91% to 96.27%.

When our scheme is compared with manual tagging, the time taken is two orders of magnitude less even with native users' intervention (about 200 times faster). We are discounting the time taken to create the knowledge base as that is a one-time exercise. The interesting result we obtained was that even if we include the time taken by experts to verify and correct the output of our tagger, the time taken is still much less than a fully manual tagging system by experts using the Sanchay software to do the tagging. The tagging rate is 34 times faster. This is because the expert is looking at already tagged text most of which is accurate. But the need for a completely machine based system becomes clear when we consider the actual times rather than the rate or the percentage. To verify the entire 250,000 word corpus that we tagged, will take 247 hours of experts' time.

Experiments showed that the main contribution of native users is the identification of proper nouns (simple and compound) and compound common nouns. If a reader reads an untagged corpus and populates the proper noun tables by identifying the proper nouns in the corpus, we may need very little native speaker assistance and the speed of this last stage may go up further. This is proposed as future work.

The community of native speakers can contribute towards the development of the Assamese language through the various user interfaces developed during the process. The tool is available online so that we can tap the large community of native speakers to contribute towards the development of a tagged corpus of Assamese through crowd sourcing.

8.2 Language Independent approach

This semi automated POS tagging approach can be used for any other morphologically rich language or languages where suffixation plays a major role. We have partially worked with the Bodo Language. We have generated a small affix list and a list of few root words. Suffixation plays a major role in Bodo. Concatenative form of suffixes are much more in Bodo than in Assamese. The same word grammar developed for Assamese can be applied to Bodo. Figure*Fig 8.1*of *Page 100* describes the suffixation patterns for Bodo.

We have also created the spell checker for Bodo lanuage to test the correctness of the root words.

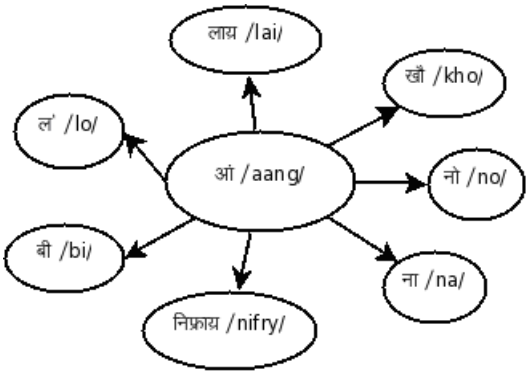


Figure 8.1: Suffixation pattern for Bodo

8.3 Future Work

The work described in this thesis is available in the internet so that using crowd sourcing we can create a larger (annotated) corpus for Assamese. We will explore the use of pre-populating the proper noun table by a reading of the corpus to be tagged and see the effect on the tagging process. We are in the process of applying this technique to another language of the region, Bodo. The corpus generated during the research work is also available in internet for other users who wish to work on the Assamese language. A lot of computational linguistic work has yet to be done for Assamese. With this large annotated tagged corpus of Assamese, further NLP related research can be carried out.

Appendix A

Appendix

A.1 List of Corpora

Detail List of Corpora used in the thesis:

Sl. No.	Domain/ Corpus	Source	Sentences	Words	Remarks
A1	Generic/ C1-GA	Assamese book "Kaikoi Aane Ghatisil" [Original English "It Happened Tomorrow"]. It is a collection of scientific fiction stories.	7,451	87,639	Used in submitted table. First 200 words were used in set-A (ref: Table 5.2, sec 5.5)
A2	Generic/ /C1-GB	Assamese children story Book "Budhi Aair Sadhu"	1,671	25,223	
A3	Tourism / C1-T	Assamese translation of IITB Tourism Hindi corpus	7,142	1,37,353	
A	**C1**	**A1+A2+A3**	**16,264**	**2,50,215**	Used for bigram value calculation
B1	Generic/ Set-I	Assamese Novel: Dhanya Nara Tanu Bhal	21,137	2,23,261	This corpus was tagged using
B2	Generic/ Set-II	Assamese Novel: Ashimat Jar Heral Sima	4,418	67,510	
B3	Generic/	Assamese Novel:	1,518	19,982	

	Set-III	Anuradhar Desh			
B4	Generic/ Set-IV	Assamese Novel: Rupawali Noir Sonowali Ghat	10,336	11,5933	the developed
B5	Generic/ Set-V	Assamese Novel: Papiya Tarar Sadhu	5,681	57,227	POS tagger (ref: Table 7.1, sec 7.1)
B6	Generic/ Set-VI	Assamese Novel: Silabhadrar Galpa Ghat	3,435	32,915	
B7	Generic/ Set-VII	as.wikipedia.org	2,425	37,226	
B	**Generic/ C2**	**B1+B2+B3+B4+ B5+B6+B7**	**48,950**	**5,54,054**	
C	**Farming/ C3**	**http://sadik2crs.blogspot .in/2013_02_01_archive. html**	**443**	**7,734**	Used to verify the results of the developed tagger (ref: Table-7.2, sec 7.2)
D1	Generic/ Set-B	Assamese Novel: Jiwanar Batot	20	195	Used during the tuning of the tagger (ref. Table- 5.2, sec. 5.5)
D2	Generic/ Set-C	Blogspot.com	8	83	
D3	News/ Set-D	Page-3, 25th Nov, 2008, Dainik Janambhumi	9	212	
D	**C4**	**D1+D2+D3**	**37**	**490**	
E1	Generic	Assamese eBook: Bakhar	298	4,847	Collected from AssamKart.com
E2	Generic	Assamese eBook: Miri Jiyari	2,178	24,338	
E3	Generic	as.wikipedia.org	599	7,922	
E4	History	as.wikipedia.org	1,577	56,392	Assam History
E5	Legends	as.wikipedia.org	1,600	61,150	Life of various Assamese great people
E6	Tourism	as.wikipedia.org + Gauhati University	10,336	95,558	Part of this was collected from

					Gauhati University
E	**C5**	**E1+E2+E3+E4+E5+E6**	**16,588**	**2,50,007**	
F	**C6**	**Assamese Novel: Jiwanar Batot**	**279**	**2,500**	Used for manual analysis

A.2 Some of the language specific rules that have been implemented in the POS tagger

```
# The following code segment checks whether the generated
root word is available in the root table or not. If
exists and contains any of the mentioned suffixes, then
the tag of the root word will be assigned to the word.
Else the word will be tagged with the default tag, NN.

        $rows = mysql_fetch_assoc($result);
        $s=$rows['sfx'];
        $tag=$rows['tag'];
        $tagSfx=$rows['tag'];

if ( (strcmp($s,"র" ) == 0) OR  (strcmp($s,"ই" ) == 0) OR
   (strcmp($s,"ুরা" ) == 0) OR  (strcmp($s, "োরা") == 0) OR
   (strcmp($s, "াতে") == 0) OR (strcmp($s, "ার") == 0) OR
   (strcmp($s, "ি") == 0) OR (strcmp($s,"ী") == 0) OR
   (strcmp($s, "া") == 0) OR (strcmp($s, "ে") == 0) OR
   (strcmp($s, "ো") == 0) OR (strcmp($s, "ও") == 0)OR
   (strcmp($s, "ক") == 0)OR (strcmp($s, "ে") == 0)OR
   (strcmp($s, "ো") == 0)OR (strcmp($s, "ত") == 0)OR
   (strcmp($s, "ই") == 0)OR (strcmp($s, "তে") == 0)) OR
   (strcmp($s,"িল" ) == 0) OR (strcmp($s,"লত" ) == 0) OR

{

 $query20="SELECT * FROM `root` where rootw='$word'";
 $result20 = mysql_query($query20) or die('Error, query
                failed-20. '.mysql_query());
    if(mysql_num_rows($result20)!=0){
                $row20= mysql_fetch_assoc($result20);
                $tag=$row20[tag];
    } else{ $tag="NN";}
}

# If the word does not contain any prefix and/or suffix
and is not available in the root table, then tag the word
with default tag, NN and insert the word as a new word in
the output table for further analysis by native users' or
experts.
```

```
if ( ($p=="") && ($a=="") && ($s=="") && ($digit == 0 )
&& ($english == 0) &&($symbol == 0) &&($hindi == 0) ) {
        $w= " ".$word."[".$word."]##"."Sorry";
        $init1[$index][1]="NN";
        $init1[$index][2]="1";

 $query101="insert into output select '$word','NULL',
    '$word', 'NULL','NN', 'NULL', 'YES' from dual where
     not exists ( select * from output where
     word='$word')";
 $result101 = mysql_query($query101) or die('Error, query
              failed-5. '.mysql_query());}


# If there is no root word and only prefix and suffix is
available in the word, then the new root word will be the
concatenation of prefix and suffix. Like এটা /eta/ 'one'
or দুটা /duta/ 'two' etc

if((($p=="এ") || ($p=="দু")) && (($s=="টা")&& ($a==""))){
        $w= $p.$s;
        $p="";
        $s="";
}


# The following rules will act whenever there is no
suffix available in the word, i.e. the word contains only
prefix and root word.


    A1. If the whole word is available in the root table
    then the corresponding tag available in the table
    will be assigned to the word.

     $a=$word;
     $query1="SELECT * FROM `root` where rootw='$a'";
     $result1 = mysql_query($query1) or die('Error,
                query failed-6. '.mysql_query());
```

```
  if(mysql_num_rows($result1)!=0) {
     $row1= mysql_fetch_assoc($result1);
     $a= "$row1[rootw]";
    $tag=$row1[tag];
    $x=$p.$a;

    A2. If the system generated root word is like “কাৰ”
       /kar/, “হৰ” /har/ etc and prefix contains any of
       the following prefixes, then concatenating the
       prefix and root word the new word will be formed
       and this new word will be searched in the root
       table, and if it exists, corresponding tag will
       be assigned or else default tag will be assigned.


if ((($tag=="NN") ||($tag=="N")) && (($p=="ন") || ($p=="নো"
    ) || ($p=="নু" )|| ($p=="না") || ($p=="নি") || ($p=="নে") ||
    ($p=="বি") || ($p=="উস")|| ($p=="অষি")|| ($p=="পৰা")) )

          {  $a= $p.$a; $p="";
             $tag=$row1[tag]; }

else { if (($p=="অ") && ($tag=="V" )) { $tag="NN";}
else { if ($p=="এ"){$tag="NN"; }
else $tag="$tag";


# The following rule will act whenever there is no prefix
available in the word, i.e. the word contains only root
word and suffix. And after suffix striping if the root
word length becomes 1,the root word length is 1.

$len=strlen(utf8_decode($s));
if (($len >1) && (strlen(utf8_decode($a))==1) &&
    (strcmp($a,”ক”) !=0)) {
         $t= mb_substr("$s",0,1,"UTF-8");
          $sp= mb_substr("$s",1,$len,"UTF-8");
          $at=$a.$t;
           $a=$a.$t;
           $x=$p.$at.$sp;
```

```
     $s=$sp;

After reformation of the root word and suffix, the newly
formed words are searched in the databases.

# To check whether a word with variants of negated
prefixes is really a negated verb or not.

    $wl= strlen(utf8_decode($word));
    $s1=mb_substr("$word ",1,1,"UTF-8");
    $s2=mb_substr("$word ",2,1,"UTF-8");
    If (((strcmp($s1,$s2)!=0)&&((($p=="ন") || ($p=="নো" )
    || ($p=="নু" )|| ($p=="না") || ($p=="নি") || ($p=="নে")) ) {

        $word=$p.$a.$x;
        $tag="NN";
    }
    Else {$tag="V";}


//Another rule for some exceptional cases like “বিদ্যালয়ে”

    $wl= strlen(utf8_decode($word));
    $sl= strlen(utf8_decode($s));
    if ( $sl ==2)
        {
        $s1=mb_substr("$s",0,1,"UTF-8");
        $s2=mb_substr("$s",1,2,"UTF-8");
        $res= ($wl-$sl);
        $wst=mb_substr("$word",$res,1,"UTF-8");
        if (($wst == $s1) && ($s1 !="ল") && (($s1!="ি" )
        &&($s1!="ী") )){
            $anw = $a.$s1;  $snw=$s2;
        }

    if ($anw !=""){
    $w= " ".$x."[".$p."+".$anw."+".$snw."]##"."$tag";}
    else {$w= " ".$x."[".$p."+".$a."+".$s."]##"."$tag";
    }
```

Bibliography

[1] http://en.wikipedia.org/wiki/Part-of-speech_tagging

[2] http://nlp.stanford.edu/software/tagger.shtml

[3] http://en.wikipedia.org/wiki/Brown_Corpus

[4] http://en.wikipedia.org/wiki/Lancaster-Oslo-Bergen_Corpus

[5] www.natcorp.ox.ac.uk/

[6] http://en.wikipedia.org/wiki/German_Reference_Corpus

[7] http://ucrel.lancs.ac.uk/claws/

[8] http://cst.dk/online/pos_tagger/uk/

[9] http://cogcomp.cs.illinois.edu/demo/pos/?id=4

[10] Eric Brill: *A simple rule-based part of speech tagger*, Proceedings of the Third Conference on Applied Natural Language Processing, 1992, 152-155.

[11] http://ccl.pku.edu.cn/doubtfire/NLP/Lexical_Analysis/Word_Segmentation_Tagging/POS_Tagging_Overview/POS%20Tagging%20Overview.htm

[12] HUANG Heyan, ZHANG Xiaofei: *Part-of-Speech Tagger Based on Maximum Entropy Model*, 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Beijing,8-11 August,2009, pp. 26-29.

[13] Hanna M. Wallach: *Conditional Random Fields: An Introduction*, University of Pennsylvania CIS Technical Report MS-CIS-04-21, 2004.

[14] Jakub Zavrel, Walter Daelemans: *Recent Advances in Memory-Based Part-of-Speech Tagging*, In VI Simposio Internacional de Comunicacion Social, 1999, pp. 590-597.

[15] http://www.coli.uni-saarland.de/ schulte/Teaching/ESSLLI-06/Referenzen/Tagging/schmid-hsk-tag.pdf.

[16] http://www.coli.uni-saarland.de/ thorsten/tnt/.

[17] http://www.lsi.upc.edu/ nlp/SVMTool/

[18] http://crftagger.sourceforge.net/

[19] Kristina Toutanova, Dan Klein, Christopher D. Manning, Yoram Singer: *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*, In Proceedings of North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL),2003, pp. 252-259.

[20] Yoshimasa Tsuruoka, Yusuke Miyao, Jun'ichi Kazama: *Learning with Lookahead: Can History-Based Models Rival Globally Optimized Models?*, Proceedings of the Fifteenth Conference on Computational Natural Language Learning, Portland, USA, 23–24 June 2011, pp. 238–246.

[21] Helmut Schmid: *Probabilistic Part-of-Speech Tagging Using Decision Trees*, Revised version of a paper which was presented at the International Conference on New Methods in Language Processing, Manchester, UK, 1994.

[22] Lois Boggess, Janna S. Hamaker, Richard Duncan, Lee Klimek, Yufeng Wu, Yu Zeng: *A comparison of Part of Speech Taggers in the Task of Changing to a New Domain*, In Proceedings of International Conference on Information Intelligence and Systems, 1999, pp. 574 - 578.

[23] Elliott Franco Drabek, David Yarowsky: *Induction of Fine-grained Part-of-Speech Taggers via Classifier Combination and Crosslingual Projection*, Proceedings of the ACL Workshop on Building and Using Parallel Texts, Ann Arbor, June 2005, pp. 49-56.

[24] Péter Halácsy, Andras Kornái, Casba Oravecz, Viktor Trón, Dániel Varga: *Usning a morphological analyzer in high precision POS tagging of Hungarian*, In Proceedings of LREC 2006, pp. 2245-2248.

[25] Phyu Huninn Myint, Tin Myat Htwe and Ni Lar Thein: *Bigram Part-of-Speech Tagger for Myanmar Language*, In Proceedings of International Conference in Information Communication and Management (IPCSIT),2011. pp. 147-152.

[26] Stéphane Bressan, Lily Suryana Indradjaja: *Part-of-Speech tagging Without Training*, In Proceedings of IFIP International Conference (INTELLCOMM),Bangkok, 2004, PP 112-119.

[27] Bocharov V. V., Alexeeva S. V., Granovsky D. V., Protopopova E. V., Stepanova M. E., Surikov A. V.: *Crowdsourcing morphological annotation*, In Proceedings of Dialogue, Bekasovo, May 29- June 2, 2013.

[28] Helmut Schmid: *Improvements in part-of-SpeechTagging With An Application To German*, In Proceedings of the ACL SIGDAT-Workshop, Dublin, 1995, pp. 47-50.

[29] Dinesh Kumar and Gurpreet Singh Josan: *Part of Speech Taggers for Morphologically Rich Indian Languages: A Survey*, International Journal of Computer Applications 6(5), September 2010, pp. 1–9.

[30] Shambhavi B. R. and Dr Ramakanth Kumar P: *Current state of the Art POS tagging for Indian Languages - A Study*, International Journal of Computer Engineering and Technology, May- June 2010, pp. 250-260.

[31] Anthony P J and Dr Soman K P : *Parts of Speech tagging for Indian Languages: a Literature Survey*, International Journal of Computer Applications 34(8), November 2011.

[32] Baskaran, S, Kalika Bali, Tanmoy Bhattacharya, Pushpak Bhattacharyy, Girish Nath Jha, Rajendran S, Saravanan K, Sobha L, Subbarao K V.: *Designing a common POS-Tagset Framework for Indian Languages*, In Proceedings of the 6th Workshop on Asian Language Resources (ALR 6), Hyderabad,2008, pp. 89

[33] Shrivastava, Agrawal, Mohapatra, Singh and Battacharya : *Morphology based Natural Language Processing tool for Indian languages*, 4th Annual Inter Re-

search Institute Student Seminar in Computer Science (IRISS05), IIT Kanpur, April 2005.

[34] Pradipta Ranjan Ray, Harish V, Sudeshna Sarkar, A Basu: *Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi*, 1st International Conference on Natural Language Processing (ICON 2003), Mysore, 2003.

[35] Manish Shrivastava, Pushpak Bhattacharyya: *Hindi POS Tagger Using Naive Stemming : Harnessing Morphological Information Without Extensive Linguistic Knowledge*, ICON, 2008.

[36] A. Dalal, K. Nagaraj, U. Swant, S. Shelke and P. Bhattacharyya: *Building Feature Rich POS Tagger for Morphologically Rich Languages: Experience in Hindi*, ICON, 2007.

[37] Himanshu Agrawal: *POS Tagging and Chunking for Indian Languages*, In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages, Hyderabad, 2007.

[38] Pranjal Awasthi, Delip Rao, B Ravindran: *Part of Speech Tagging and Chunking with HMM and CRF*, In Proceedings of of NLPAI Machin Learning, 2006.

[39] Avinesh PVS and Karthik G: *Part-Of-Speech Tagging and Chunking Using Conditional Random Fields and Transformation Based Learning*, In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages, Hyderabad, 2007.

[40] Delip Rao, David Yarowsky: *Parts of Speech tagging and Shallow Parsing for Indian Languages*, In Proceedings of the workshop on Shallow Parsing for South Asian Languages, at the International Joint Conference in Artificial Intelligence (IJCAI), 2007.

[41] Pradeep Varma D, Rakesh M, Ratna Sanyal: *HMM-based Language-independent POS Tagger*, in Proceedings of the 3rd Indian International Conference on Artificial Intelligence, Pune, India, December 17-19, 2007, pp. 1924-1935.

[42] Sandipan Dandapat, Priyanka Biswas Monojit Choudhury, kalika Bali: *Complex Linguistic Annotation- No Easy Way Out! A case from Bangla and Hindi POS Labeling Tasks*, In Proceedings of the 3rd Linguistic Annotation Workshop (ACL-IJCNLP), Suntec, Singapore, 2009), pp-10-18.

[43] http://www.ldcil.org/resourcesTextCorp.aspx

[44] http://www.lrec-conf.org/proceedings/lrec2010/pdf/874_Paper.pdf

[45] http://sanskrit.jnu.ac.in/ilci/index.jsp

[46] Asif Ekbal and S. Bandyopadhyay: *Web-based Bengali news Corpus for Lexicon Development and POS Tagging*, POLIBITS, ISSN 18709044 (37), 2008, pp. 20-29.

[47] Sandipan Dandapat, Sudeshna Sarkar, Anupam Basu: *Automatic Part-of-Speech Tagging for Bengali: An Approach for. Morphologically Rich Languages in a Poor Resource Scenario*, In Proceedings of the ACL 2007 Demo and Poster Sessions, June 2007, pages 221-224.

[48] Fahim Hasan, N UzZaman, Mumit Khan: *Comparision of different POS Tagging techniques (N-Gram,HMM and Brillś tagger) for Bangla*, In Proceedings of ICS2E, 2006.

[49] Fahim Hasan, N UzZaman, Mumit Khan: *Comparision of Unigram, Bigram, HMM and Brill's POS Tagging Approaches for some South Asian Languages*, In Proceedings of Conference on Language and Technology (CLT07), Pakistan, August 7-11, (2007).

[50] Gautam Kumar Saha: *Parsing Bengali text- an intelligent Approach*, ACM Ubiquity, Vol. 7 Issue 13, April, ACM Press, USA, 2006.

[51] Chirag Patel and Karthik Gali: *Part-Of-Speech Tagging for Gujarati Using Conditional Random Fields*, In Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, Hyderabad, January 2008, pp. 117–122.

[52] Sirajul Islam Choudhury, L. Sarbajit Singh, S. Borgohain and P. K. Das: *Morphological Analyzer for Manipuri: Design and Implementation*, In Pro-

ceedings of Asian Applied Computing Conference,Kathmandu, Nepal, 2004, pp. 123-129.

[53] Jyoti Singh, Nisheeth Joshi, Iti Mathur: *Part of Speech Tagging of Marathi Text Using Trigram Method* , In Proceedings of International Journal of Advanced Information Technology (IJAIT) Vol. 3, No.2, April, 2013.

[54] Braja Gopal Patra, Khumbar Debbarma, Dipankar Das, Sivaji Bandyopadhyay: *Part of Speech (POS) Tagger for Kokborok* , In Proceedings of COLING (Posters), Mumbai, December 2012, pp. 923–932.

[55] http://www.iitg.ernet.in/rcilts/

[56] Navanath Saharia, Dhrubajyoti Das, Utpal Sharma, Jugal Kalita: *Part of Speech Tagger for Assamese Text*, In Proceedings of the ACL-IJCNLP Conference, 2009, pp. 33-36.

[57] Mizanur Rahman,Sufal Das, Utpal Sharma: *Parsing of part-of-speech tagged Assamese Texts*, In Proceedings of the International Journal of Computer Science Issues, Vol. 6, No.1, 2009.

[58] Navanath Saharia, Utpal Sharma, Jugal Kalita: *A First Step Towards Parsing of Assamese Text*, Languages of India, Special Vol: Problems of Parsing in Indian Languages, May 2011.

[59] Utpal Sharma, Jugal Kalita, Rajib Das: *Root Word Stemming by Multiple Evidence from Corpus*, In Proceedings of 6th International Conference on Computational Intelligence and Natural Computing (CINC), North Carolina, 2003.

[60] Navanath Saharia, Utpal Sharma, Jugal Kalita: *Analysis and Evaluation of Stemming Algorithms: A case study with Assamese*, In Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2012.

[61] Navanath Saharia, Kishori M Konwar, Utpal Sharma, Jugal K Kalita: *An improved stemming approach using HMM for a highly inflectional language*,

In Proceedings of 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), Samos, Greece, March 24-30, 2013, pp. 164-173 .

[62] Mathias Creutz, Krista Lagus, Sami Virpioja: *Unsupervised Morphology Induction Using Morfessor*, In Proceedings of 5th International Workshop on Finite-State Methods and Natural Language Processing, (FSMNLP), Helsinki, Finland, September 1-2, 2005, pp. 300-301.

[63] https://www.mturk.com/mturk/

[64] Matt Post, Chris Callison-Burch and Miles Osborne: *Constructing Parallel Corpora for Six Indian Languages via Crowdsourcing*, In Proceedings of the Seventh Workshop on Statistical Machine Translation (WMT), 2012, pp. 401-409.

[65] Anoop Kunchukuttan, Shourya Roy, Pratik Patel, Kushal Ladha, Somya Gupta, Mitesh Khapra and Pushpak Bhattacharyya: *Experiences in Resource Generation for Machine Translation through Crowdsourcing*, In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC), 2012.

[66] http://www.iitg.ernet.in/rcilts/assamese.html

[67] http://www.iitg.ernet.in/rcilts/phaseI/newassamesedesign.pdf

[68] http://www.lisindia.net/Assamese/Assamese.html

[69] http://ltrc.iiit.ac.in/nlpai_contest07/Sanchay/

[70] Baskaran, S, Kalika Bali, Tanmoy Bhattacharya, Pushpak Bhattacharyy, Monojit Choudhury, Girish Nath Jha, Rajendran S, Saravanan K, Sobha L, Subbarao K V.: *A Common Parts-of-Speech Tagset Framework for Indian Languages*, In Proceedings of the International Conference on Language Resources and Evaluation, (LREC), 26 May - 1 June 2008, Morocco, 2008.

[71] http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf

[72] http://www.nltk.org

[73] Nitin Madnani: *Getting Started on Natural Language Processing with Python*, ACM Crossroads Student Magazine, 13(4):5, 2007.

[74] Ms Lilabati Saikia Bora: *Asamiya Bhasar Ruptattva- First Edition*, Published by M/s Banalata, Panbazar Guwahati-1, January 2006,

[75] Satyanath Borah: *Bahal Vyakaran*, Published by B.C. Barua on behalf of Gopal Barua Agency, S.B Road, Guwahati-1, April 2006 Ed.

[76] Prof. G. C. Goswami: *Asamiya Vyakaran Pravesh*, Published by M/s Bina Library, Guwahati-1, Second Edition, April 2003.

[77] Prof. G. C. Goswami: *Asamiya Vyakaranar Maulik Vicar*, Published by M/s Bina Library, Guwahati-1, Sixth Edition, April 2008.

[78] Prof. P. N. Dutta Baruah: *Adhunik Bhasabignan Paricaya*, Published by Banalata, Dibrugarh-1, July 2006.

[79] Atro Voutilainen, Lluís Padró: *Developing a hybrid NP parser*, In Proceedings of the 5th Applied Natural Language Processing Conference, March 31 - April 3, 1997, pp. 80-87.

[80] Akshar Bharati, Dipti Misra Sharma, Lakshmi Bai, Rajeev Sangal: *AnnCorra : Annotating Corpora Guidelines For POS And Chunk Annotation For Indian Languages* , Language Technologies Research Centre IIIT, Hyderabad, 2006.

[81] Christopher D. Manning: *Part-of-speech tagging from 97% to 100%: is it time for some linguistics?*, In Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I, pp. 171-189.

[82] Milan Sečujski, Vlado Delić: *A Software Tool for Semi-Automatic Part-of-Speech Tagging and Sentence Accentuation in Serbian Language*, In Proceedings Fifth Slovenian and First International Language Technologies Conference, Slovenia, 2006.

[83] Pavel Kveton and Karel Oliva: *(Semi-)Automatic Detection Of Errors In PoS-Tagged Corpora*, In Proceedings International Conference on Computational Linguistics, 2002.

[84] Mohamed Attia, A. A. Rashwan and Mohamed A. S. A. A. Al-Badrashiny: *Fassieh, a Semi-Automatic Visual Interactive Tool for Morphological, PoS-Tags, Phonetic, and Semantic Annotation of Arabic Text Corpora*, IEEE Transactions on Audio, Speech, and Language Processing, Volume 17 Issue 5, July 2009, pp. 916-925.

[85] Liu Wenyin, Susan Dumais, Yanfeng Sun, Hongjiang Zhang, Mary Czerwinski and Brent Field: *Semi-Automatic Image Annotation*, In Proceedings of INTERACT2001, 8th IFIP TC 13 Conference on Human-Computer Interaction, 2001, pp.326-333.

[86] Nadzeya Kiyavitskaya, Nicola Zeni, Luisa Mich, James R. Cordy and John Mylopoulos: *Text Mining Through Semi Automatic Annotation*, In Proceedings 6th International Conference, PAKM 2006, Vienna, Austria, pp. 143-154.

[87] Nadzeya Kiyavitskaya, Nicola Zeni, James R. Cordy , Luisa Mich, and John Mylopoulos: *Semi-Automatic Semantic Annotations for Web Documents*, In Proceedings 2nd Italian Semantic Web Workshop, SWAP 2005, pp. 14-15.

[88] Bertrand Sereno, Simon Buckingham and Enrico Motta: *Semi-Automatic Annotation of Contested Knowledge on the World Wide Web*, In Proceedings International World Wide Web Conference, 2004, pp. 276-277.

[89] Tomohiro YAMASAKI, Hiromi WAKAKI and Masaru SUZUKI: *The Semi-Automatic Construction of Part-Of-Speech Taggers for Specific Languages by Statistical Methods*, In Proceedings of 2nd Workshop on South and Southeast Asian Natural Language Processing, Thailand, 2011, pp. 23-29.

[90] A. Das and S. Bandyopadhyay: *SentiWordNet for Indian Languages*, In the 8th Workshop on Asian Language Resources (ALR), COLING August, 2010, Beijing, China Pages 56-63.