# COMPARISON OF DIFFERENT POS TAGGING TECHNIQUES FOR SOME SOUTH ASIAN LANGUAGES

A Thesis

Submitted to the Department of Computer Science and Engineering of

BRAC University

by

Fahim Muhammad Hasan

Student ID: 03101057

In Partial Fulfillment of the

Requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

December 2006


BRAC University, Dhaka, Bangladesh

# Declaration

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole nor in part, has been previously submitted for any degree.

| Signature of Supervisor | Signature of Co-Supervisor | Signature of Author |
|---|---|---|
| --- | --- | --- |
| Dr. Mumit Khan | Naushad UzZaman | Fahim Muhammad Hasan |

# Acknowledgments

I would like to thank my thesis supervisor, Dr. Mumit Khan and co-supervisor, Naushad UzZaman, for their guidance and ever helpful comments on my work. I also thank my teachers at BRAC University, my family and friends.

# Abstract

There are different approaches to the problem of assigning a part of speech (POS) tag to each word of a natural language sentence. We present a comparison of the different approaches of POS tagging for the Bangla language and two other South Asian languages, as well as the baseline performances of different POS tagging techniques for the English language. The most widely used methods for English are the statistical methods i.e. n-gram based tagging or Hidden Markov Model (HMM) based tagging, the rule based or transformation based methods i.e. Brill's tagger. Subsequent researches add various modifications to these basic approaches to improve the performance of the taggers for English. Here, we present an elaborate review of previous work in the area with the focus on South Asian Languages such as Hindi and Bangla. We experiment with Brill's transformation based tagger and the supervised HMM based tagger without modifications for added improvement in accuracy, on English using training corpora of different sizes from the Brown corpus. We also compare the performances of these taggers on three South Asian languages with the focus on Bangla using two different tagsets and corpora of different sizes, which reveals that Brill's transformation based tagger performs considerably well for South Asian languages. We also check the baseline performances of the taggers for English and try to conclude how these approaches might perform if we use a considerable amount of annotated training corpus.

# Table Of Content

# List of Tables

# List of Figures

# Background

## 1.1 Introduction

Parts of speech (POS) tagging is one of the most well studied problems in the field of Natural Language Processing (NLP). Different approaches have already been tried to automate the task for English and other western languages. Though Bangla (or Bengali) is one of the top ten most spoken languages in the world [1] and is spoken by more than 200 million people, it still lacks significant research efforts in the area of NLP.

This thesis discusses the different techniques for POS tagging for western languages as well as the South Asian languages. It displays the analyses of performance of well known tagging methods for the western languages using corpora of varying sizes. It compares the performance of some South Asian languages using the same techniques with that of the western languages and attempts to suggest which technique might be better for the South Asian languages. It concludes with a discussion on some improvement techniques that could be added to a baseline tagger to improve its performance.

## 1.2 Parts of Speech Tagging

Parts of speech (POS) tagging means assigning grammatical classes i.e. appropriate parts of speech tags to each word in a natural language sentence. Assigning a POS tag to each word of an un-annotated text by hand is very time consuming, which results in the existence of various approaches to automate the job. So automated POS tagging is a technique to automate the annotation process of lexical categories. The process takes a word or a sentence as input,

assigns a POS tag to the word or to each word in the sentence, and produces the tagged text as output.

POS tags are also known as word classes, morphological classes, or lexical tags. The significance of these is the large amount of information they give about a word and its neighbors. POS tagging can be used in Text to Speech (TTS) applications, information retrieval, parsing, information extraction, linguistic research for corpora, [2, 3] and also can be used as an intermediate step for higher level NLP tasks such as parsing, semantics analysis, translation, and many more [4], which makes POS tagging a necessary function for advanced NLP applications in Bangla or any other language.

POS tagging has its importance in subsequent processing of text such as parsing, as it makes the processing easier by attaching a class to a word [5], and it is also widely used for linguistic text analysis. POS tagging is used as an early stage of text analysis in many applications such as subcategory acquisition, text to speech synthesis and alignment of parallel corpora [6]. It is essential for being a component of many applications in natural language processing and related domains [3], and it is also used to build the knowledge base of a Natural Language Analyzer [7].

POS tagging is useful for syntactic parsing as taggers reduce ambiguity from the parser's input sentence, which makes parsing faster by making the computational problem smaller, and the result less ambiguous. It also resolves some ambiguities that are not addressed by the syntactic parser’s language model. With the rule-based taggers, the parser's output becomes less ambiguous without a considerable penalty to recognition rate and at the same time, parsing speed increases to some extent [8].

In the following sections, we start by giving a overview of some of the widely used POS tagging models. Then we discuss about some of the

work already completed in this field, focusing on the South-Asian languages such as Bangla and Hindi. We continue by describing the annotated training corpora, the tagsets and the methodologies we used for our experiments and analyze the results of the experiments that we did for English, Bangla and two other South-Asian languages. We also try to compare the performances of the taggers on English and Bangla using tagsets and corpora of similar sizes. We conclude by describing some additional works and modifications that we wish to do in future on the POS tagging models for Bangla, to improve the performance of the tagging models to a significant extent.

## 1.3 Classification

There are different approaches for POS tagging. The following figure demonstrates different POS tagging models.

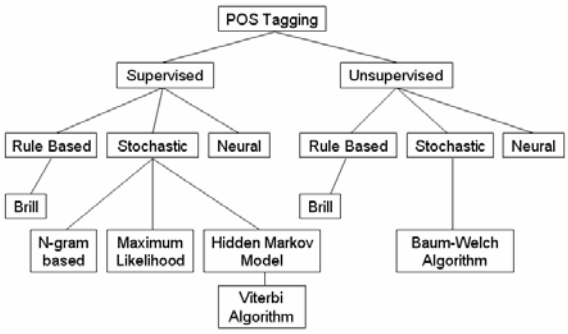


Figure 1: Classification of POS tagging models

### 1.3.1. Supervised POS Tagging

The supervised POS tagging models require a pre-tagged corpora which is used for training to learn information about the tagset,

word-tag frequencies, rule sets etc [10]. The performance of the models generally increase with the increase in size of this corpora.

### 1.3.2. Unsupervised POS Tagging

Unlike the supervised models, the unsupervised POS tagging models do not require a pre-tagged corpora. Instead, they use advanced computational methods like the Baum-Welch algorithm to automatically induce tagsets, transformation rules etc. Based on the information, they either calculate the probabilistic information needed by the stochastic taggers or induce the contextual rules needed by rule-based systems or transformation based systems [9, 10].

Both the supervised and unsupervised POS tagging models can be of the following types.

### 1.3.3. Rule Based / Transformation Based

The rule based POS tagging models apply a set of hand written rules and use contextual information to assign POS tags to words. These rules are often known as context frame rules. For example, a context frame rule might say something like: *"If an ambiguous/unknown word X is preceded by a Determiner and followed by a Noun, tag it as an Adjective."* On the other hand, the transformation based approaches use a pre-defined set of handcrafted rules as well as automatically induced rules that are generated during training.

Morphology is a linguistic term which means how words are built up from smaller units of meaning known as morphemes [3]. In addition to contextual information, morphological information is also used by some models to aid in the disambiguation process. One such rule might be: "*If*

*an ambiguous/unknown word ends in a suffix -ing and is preceded by a Verb, label it a Verb"*.

Some models also use information about capitalization and punctuation, the usefulness of which are largely dependent on the language being tagged.

In general, the rule based tagging models usually require supervised training i.e. pre-annotated corpora. But recently, good amount of work has been done to automatically induce the transformation rules. One approach to automatic rule induction is to run an untagged text through a tagging model and get the initial output. A human then goes through the output of this first phase and corrects any erroneously tagged words by hand. This tagged text is then submitted to the tagger, which learns correction rules by comparing the two sets of data. Several iterations of this process are sometimes necessary before the tagging model can achieve considerable performance. [9]

The transformation based approach is similar to the rule based approach in the sense that it depends on a set of rules for tagging. It initially assigns tags to words based on a stochastic method e.g. the tag with highest frequency for a particular word is assigned to that word. Then it applies the set of rules on the initially tagged data to generate final output. It also learns new rules while applying the already learnt rule, and can save the new rules if they seem effective i.e. improve the performance of the model.

### 1.3.4. Stochastic

A stochastic approach includes frequency, probability or statistics. The simplest stochastic approach finds out the most frequently used tag for a specific word in the annotated training data and uses this information to tag that word in the unannotated text. The problem with

this approach is that it can come up with sequences of tags for sentences that are not acceptable according to the grammar rules of a language.

An alternative to the word frequency approach is known as the n-gram approach that calculates the probability of a given sequence of tags. It determines the best tag for a word by calculating the probability that it occurs with the n previous tags, where the value of n is set to 1,2 or 3 for practical purposes. These are known as the Unigram, Bigram and Trigram models. The most common algorithm for implementing an n-gram approach for tagging new text is known as the Viterbi Algorithm [11], which is a search algorithm that avoids the polynomial expansion of a breadth first search by trimming the search tree at each level using the best m Maximum Likelihood Estimates (MLE) where m represents the number of tags of the following word [4].

There are different models that can be used for stochastic POS tagging, some of which are described below.

### 1.3.5. Conditional Random Fields

A Conditional Random Field (CRF) is a framework of probabilistic model to segment and label a sequence of data. A conditional model specifies the probabilities of possible label sequences given an observation sequence. The conditional probability of the label sequence can depend on arbitrary, non-independent features of the observation sequence. The probability of a transition between labels may depend not only on the current observation, but also on past and future observations [10]. The CRF model calculates the probability based on some features, which might include the suffix of the current word, the tags of previous and next words, the actual previous and next words etc. [9]

### 1.3.6. Hidden Markov Model

A Hidden Markov Model (HMM) consists of:

N: The number of states
M: Total number of distinct observation symbols
T: Length of observation sequence
$i_t$: The state at time t
$V = \{v_1, v_2 \ldots v_M\}$: The discrete set of possible observation symbols
$\pi = \{ \pi_i \}$ The probability of being in state i at the beginning of the experiment
$A = \{a_{ij}\}$ where $a_{ij} = P(i_{t+1}=j \mid i_t = i)$is the probability of being in state j at time t+1 given that we were in state i at time t.
$B = \{b_j(k)\}$ where $b_j(k) = P(v_k \text{ at } t \mid i_t = j)$ is the probability of observing the symbol $v_k$ given that we are in state j.
$O_t$: The observation symbol at time t.
$\lambda = (A, B, \pi)$ is the notation to denote an HMM [11, 12]

There are three basic problems that the HMM must solve to be used for any practical purpose. They are as follows:

1. Given the observation sequence $O = O_1O_2 \ldots O_T$ , and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O| \lambda)$, the probability of the observation sequences, given the model?

2. Given the observation sequence $O = O_1O_2 \ldots O_T$ , and the model $\lambda$, how do we choose a corresponding state sequence $V = v_1, v_2 \ldots v_M$, which is optimal in some meaningful sense (i.e., best “explains” the observations)?

3. How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O| \lambda)$?

For supervised POS tagging, we solve problem 2, in which we attempt to uncover the hidden part of the model, i.e. to find the “correct” state sequence [13].

For POS tagging, HMM is used to model the joint probability distribution P(word, tag). The generation process uses a probabilistic Finite State Machine (FSM).

The states of HMM correspond to the tags, it has an alphabet which consists of the set of words, the transition probabilities $P(Tag_i|Tag_{i-1})$ and the emission probabilities $P(Word_i|Tag_i)$

In HMM, for a given (word, tag) pair we have the probability:

$$P(w, t) = \prod_i P(Tag_i|Tag_{i-1}) * P(Word_i|Tag_i)$$

HMM is called Hidden as for a word sequence, we cannot determine the exact sequence of tags that generated this word sequence. It is called Markov as it is based on the Markovian assumption that the current tag depends only on the previous n tags. (n=1 or 2 defines the first or second order)

The HMM model trains on annotated corpora to find out the transition and emission probabilities. For a sequence of words w, HMM determines the sequence of tags t using the formula: t = argmax P(w, t)

The computation of this formula is very expensive as all possible tag sequences are required to be checked in order to find the sequence that maximizes the probability. So a dynamic programming approach known as the Viterbi Algorithm is used to find the optimal tag sequence [14]. We describe the operation of HMM for POS tagging in more details in the subsequent sections.

### 1.3.7. Maximum Entropy Model

The Maximum Entropy Model (MEM) is based on the principle of Maximum Entropy, which states that when choosing between a number of different probabilistic models for a set of data, the most valid model is the one which makes fewest arbitrary assumptions about the nature of the data [3].

The probability model for MEM is defined over (H, X, T), where H is the set of possible word and tag contexts or “histories”, and T is the set of allowable tags. The model's probability of a history h together with a tag t is defined as:

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

where π is a normalization constant, $\{a_1, ..., a_k\}$ are the positive model parameters, $\{f_1, . . , f_k\}$ are known as “features”, where fj(h, t) is in {0, 1} and each parameter $a_j$ corresponds to a feature $f_j$.

Given a sequence of words $\{w_l , . . . , w_n\}$ and tags $\{t_l,..., t_n\}$ as training data, $h_i$ is defined as the history available when predicting $t_i$. The parameters $\{a_1, …, a_k\}$ are then chosen to maximize the likelihood of the training data p [10], using the following formula:

$$L(p) = \prod_{i=1}^{n} p(h_i, t_i) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h_i,t_i)}$$

### 1.3.8. Memory Based Learning

The Memory Based Learning (MBL) Model takes tagged data as input, and produces a lexicon and memory based POS tags as output. MBL consists of two components, one is a memory based learning

component, and the other is a similarity based performance component. The learning component is called memory based as it memorizes examples while training. The performance component matches the similarity of the input with the output of the learning component to produce the actual output of the system [10].

The different models described above have their own advantages and disadvantages, however, they all face one difficulty, which is to assign a tag to an unknown word which the tagger has not seen previously i.e. the word was not present in the training corpora. Different tagging models use different methods to get around this problem. The rule based taggers use certain rules to specially handle unknown or ambiguous words. But the stochastic taggers have no way to calculate the probabilities of an unknown word beforehand. So to solve the problem, the taggers of this category calculate the probability that a suffix of an unknown word occurs with a particular tag. If HMM is used, the probability that a word containing that suffix occurs with a particular tag in the given sequence is calculated. An alternate approach is to assign a set of default tags to unknown words. The default tags typically consist of the open classes, that are word classes, which freely admit new words and are readily modified by morphological processes [3], examples are Noun, Verb, Adjective, Adverb etc. The tagger then disambiguates using the probabilities that those tags occur at the end of the n-gram in question. A third approach is to calculate the probability that each tag in the tagset occurs at the end of the n-gram, and to select the path with the highest probability. This, however, is not the optimal solution if the size of the tag set is large [9].

# Chapter I: Previous Work

Considerable amount of work has already been done in the field of POS tagging for English. Different approaches like the rule based approach, the stochastic approach and the transformation based learning approach along with modifications have been tried and implemented. However, if we look at the same scenario for South-Asian languages such as Bangla and Hindi, we find out that not much work has been done. The main reason for this is the unavailability of a considerable amount of annotated corpora of sound quality, on which the tagging models could train to generate rules for the rule based and transformation based models and probability distributions for the stochastic models [15].

In the following sections, we describe some POS tagging models that have been implemented for English along with their performances. Then we have a look at the models implemented for Hindi, Bangla and some other South-Asian Languages.

## 2.1 Rule Based and Transformation Based Approaches

The most widely used rule based or transformation based approach for POS tagging is the linguistically motivated model by Brill [16, 17], that initially assigns POS tags to words based on the most likely tags and later changes the tags using rules that could be predefined or learnt while the tagger is processing the annotated training corpora. We explore the Brill tagger in depth in the subsequent sections.

## 2.2 Stochastic Approaches

Most of the stochastic approaches are based on first or second order HMM and generally achieve good performance in POS tagging.

The supervised approaches use a training corpus on which they apply statistical models to find the most likely tag for a word [18, 19, 20, 21]. Some of the taggers based on stochastic approaches are as follows:

Trigrams N Tags (TNT) is a stochastic HMM tagger based on trigram analysis which uses a suffix analysis technique based on properties of words like suffices in the training corpora, to estimate lexical probabilities for unknown words that have the same suffices.

A bootstrapping method for training is described in [22], which use a small annotated corpora to train the initial model. This initial model is then used to tag more text, which are corrected manually before re-training the model using the corrected text. This is the method that we used to enlarge the size of the Prothom-Alo corpus [23].

Most of the supervised approaches require a large pre-annotated corpora which might not be available for new languages. This is where the unsupervised models come in. There have been several works that have utilized unsupervised learning for training a HMM for POS tagging. The most widely known is the Baum-Welch algorithm [24], that can be used to train a HMM from unannotated data.

A HMM based POS tagger is described in [20]. This model uses a lexicon and an unannotated corpus. The whole system consists of three modules, that are the tokenizer, the trainer and the tagger. The task of the tokenizer is to identify a set of tags, also known as an ambiguity class, for each word. The training module uses the Baum-Welch algorithm on a sequence of ambiguity classes on a large corpus to train an HMM. The tagging module buffers sequence of ambiguity classes between sentence boundaries. These sequences are disambiguated by computing the maximal path through the HMM using the Viterbi algorithm.

An approach to improve the estimation for unsupervised tagging is described in [25]. It states that a simple HMM model can display very good performance for unsupervised tagging by improving aspects of standard Baum-Welch estimation. According to them, one such improvement is to use word similarities to smooth the lexical tag to word probability estimates, which avoids over-fitting the lexical model. Another improvement is to limit the model to preserve a specified marginal distribution over the hidden tags, which avoids over-fitting the tag to tag transition model.

## 2.3 Other Approaches

An language independent approach is reported in [26], that achieves 77% accuracy on the Brown corpus. This approach does not require a pre-annotated corpus, lexicon or language specific information for determining tags. Instead, it uses six language features along with information on language universals to tag syntactic clusters with corresponding POS tags. These six features are called the openness, affixation, numeracy, optionality, binding, and word order of the language and are extracted from the language clusters. According to the authors, a preliminary evaluation of this process on the Brown corpus shows that the system can accurately tag most of the major categories of POS. for English

A recurrent neural network based POS tagging approach is reported in [27], which trains a discrete time recurrent neural network to predict the ambiguity class of the next word. In this approach, the information about POS tagging is stored in the states of this neural network.

To test the tagger, a 14276 entry lexicon was built from the first 20 sections of the 24 data sets of the Penn Treebank corpora, corresponding to the Wall Street Journal. 95% coverage was found by

discarding all words appearing less than 4 times. No guessers were used in the experiment. The experiments performed to compute error rates on this text displayed performance similar to a HMM based unsupervised model that uses the Baum Welch estimation algorithm for training.

Another different approach [28] is used in a POS tagger for the German language. This tagger is known as the tree tagger. To correctly tag text, the tagger builds a decision tree where the nodes are tags of previous words. These nodes are used to determine the current node i.e. the tag of the current word. Along with this, the tagger also uses a suffix tree to improve its performance. Figure shows a small part of a sample decision tree built by the tree tagger.

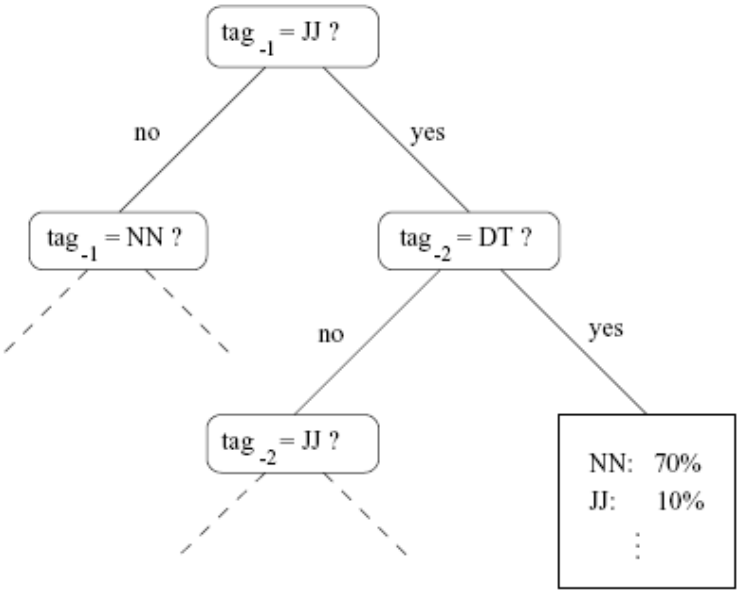


Figure 2: A Sample decision tree (Partially drawn) [28]

## 2.4 Hindi and Other South-Asian Languages

### 2.4.1. Stochastic Approaches

We have found that most of the research on POS tagging on the South-Asian languages has been done using stochastic tagging models like HMM, MEM etc. A POS tagging approach based on Maximum Entropy Markov Model using supervised training is reported in [29]. This model trains using a pre-tagged corpora and uses a feature set to predict the tag for a word. The feature set consists of POS tagging features, context based features, word features, dictionary features and corpus based features. The tagger reports an accuracy of 89.34% on the development data of the NLPAI Machine Learning Competition 2006. To attain such performance, the tagger uses a pre-annotated training corpus consisting of around 35,000 words annotated with a tagset consisting of 29 different POS tags.

As described in [30], a HMM based POS tagger was developed which demonstrated 85.42% accuracy. The tagger was also trained on a pre-annotated corpus consisting of 40956 tokens. The tagger was tested on a annotated corpus having 5967 tokens. The tagger demonstrated 79.12% accuracy when tested on an un-annotated test set consisting of 5129 tokens.

Yet another HMM based tagger is described in [31], reporting a performance of 76.49% accuracy on training and test data having about 25000 and 6000 words, respectively. This tagger uses HMM in combination with probability models of certain contextual features for POS tagging.

Finally, as reported in [5], a stochastic model based on Conditional Random Fields (CRF) is developed which demonstrates a performance of 77.48% accuracy. This is the baseline performance of the tagger

when trained on 21,000 words. When the tagger is trained on the same amount of data with the best feature set, the accuracy improves to 82.67%. But the size and type of the testing data is not mentioned in this paper, which can improve or deteriorate the performance of the tagger to a great extent.

### 2.4.2. Hybrid Approaches

In [32], the authors report a hybrid tagger for Hindi that runs on two phases to POS tag input text. In the first phase, the HMM based TnT tagger is run on the untagged text to perform the initial tagging. During this phase, a set of transformation rules is induced which are used later. In the second phase, the set of transformation rules learnt earlier is used on the initially tagged text to correct error created in the first phase. However, the performance of this tagger is not as good as the other taggers reported for Hindi. It uses a corpus of 35,000 words annotated with 26 tags, and the resulting accuracy is 79.66% using the TnT tagger. The authors suggest that the low score could be the result of the sparseness of the training data. The use of the set of transformation rules in post processing improves the overall accuracy to 80.74%.

For the Telegu language, [10] reports the performances of various approaches of POS tagging. Here the pre-annotated training corpora are the training data released for the NLPAI Machine Learning Competition 2006, consisting of 27336 words. The size of the testing data used is around 5662 tokens. Using the above data, the HMM based approach demonstrates an accuracy of 82.47% whereas the MEM based approach displays 82.27% which are very similar. The Memory Based Learning approach has 75.75% accuracy, and finally, the Conditional Random Fields based approach exhibits 75.11% accuracy.

For the Tamil language, a tagger is reported in [33], that uses a suffix stripper before performing the actual tagging to improve the accuracy. The suffix stripper uses a list of suffices, pronouns, adjectives and adverbs to remove the suffices from words. A simple block diagram of the suffix stripper is included below.

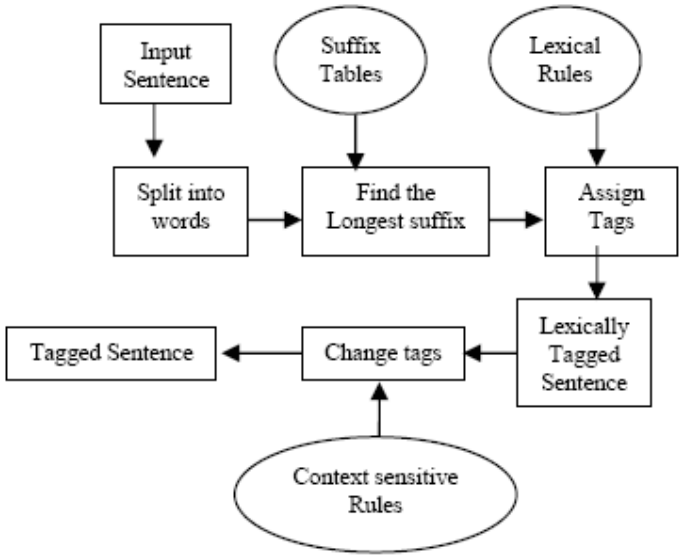


Figure 3: Block diagram of suffix stripper [33]

The input format for the tagger is one sentence per line in which each word is separated by a white space. On the input text, the tagger runs the following algorithm to remove suffices and then to complete the tagging.

*1. Split the sentence in to words.*

*2. For each word,*

*2.1. find the longest suffix at the end*

*2.2. find the table number of the suffix and eliminate the suffix from the word*

*2.3. Go to 2.1 until the word length is 2.*

*3. Using the combination of suffixes and the rules, apply the lexical rules and assign the category.*

*4. For each sentence,*

*4.1. Apply the context sensitive rules on the unknown words.*

*4.2. Apply the context sensitive rules on the wrongly tagged words.*

*4.3. If no context rule applies for any unknown words, tag it as noun.*

## 2.5 Bangla

### 2.5.1. Rule Based and Transformation Based Approaches

A rule based POS tagger for Bangla is reported in [34], but only the rules for Noun and Adjective are showed. No review or comparison with established work on POS tagging is done, neither is the presence of any performance analysis report in the paper, which makes uncertain whether the approach is worthwhile or not.

Furthermore, the work described here can be thought as more of a morphological analyzer than a POS tagger. A morphological analyzer indeed provides some POS tag information, but we need more fine-grained tags from a POS tagger.

The tagset here consists of only 9 tags which is very small compared to renowned tagsets. A POS tagger works as an intermediate tool or component for many advanced NLP applications as described earlier, but with a tagset consisting of only 9 tags, the output of the POS tagger can only be used in restricted applications. For English, most widely used tagsets include the Brown tagset [35] consisting of 87 distinct tags, and Penn Treebank's tagset [36], consisting of (36 + 12 = 48) tags.

These show the necessity of a large tagset to use the POS tagging information in other applications.

### 2.5.2. Stochastic Approaches

Notable work on POS tagging has been reported in [37] for Indian Bangla. Here, a HMM based approach is used for tagging Bangla which is a combination of both supervised and unsupervised learning for training a Bigram based HMM. It also uses a morphological analyzer before tagging that takes a word as input and gives all possible POS tags for the word. This restricts the set of possible tags for a given word to possibly increase the performance of the tagger.

To test this tagger, a tagset of 27 tags and a training corpus consisting of 3085 sentences, approximately 41,000 words have been used. The tagger can train in two ways, unsupervised and supervised. For unsupervised learning, the tagger uses the HMM trained from supervised learning as the initial model. The Baum-Welch estimation algorithm is then used to re-estimate the parameters of the model. For this, a fixed set of 11,000 unlabeled sentences with approximately 100,000 words taken from CIIL corpus is used. As reported in the paper, the baseline model demonstrates performance of 69.11% accuracy, which improves to 89.65% using morphological analyzer and semi-supervised learning.

Another paper in [38] uses a suffix based tree tagger, influenced by [28], but this is also more of morphological analyzer than POS tagger. Here the authors also mention about the n-gram based tagging, but do not describe how to combine both. This paper also lacks any review or comparison with established work on POS tagging, instead it only proposes a rule-based technique. The paper also does not show any

performance analysis of the proposed work, and it also uses a small tagset with 9 tags.

### 2.5.3. Hybrid Approaches

A hybrid POS tagger for Bangla based on HMM is described in [6] that tags using three methods. The first method uses only supervised learning, the second one uses a partially supervised learning and decodes the best tag sequence without using morphological analyzer to restrict the possible tags. The third method also uses partially supervised learning and decodes the best tag sequence with using morphological analyzer to restrict the possible tags.

To test the performance of this tagger, the authors used a tagset of 40 different tags, along with an annotated corpus consisting of 500 tagged sentences for supervised learning. For unsupervised learning, un-annotated data of 50,000 words was used for re-estimating parameter.

As reported, the tagger demonstrates impressive performance of 96.28% accuracy when run on the third method. The performance on method 1 is 64.31% and that one method 2 is 67.6%

The tagger was also tested on random sentences of 1003 words from the CIIL corpus, which were more complex than the training data and these were tagged manually. This resulted in some reduction in accuracy. The performance of method 3 in this case was 84.37% while that of method 1 and 2 were 59.93% and 61.79%, respectively.

In [39], the authors describe a tagset of 84 tags and a tagging program for Bangla using Oracle 8i, Visual Basic 6.0 and GistSDK ActiveX Controls Version 2.7. In this paper, the user interface of the tagger is given much importance and is described in detail. However, nothing is

mentioned about the training or testing corpora as well as tagging performance of the tagger.

# Chapter II: Methodology

In this chapter, we describe the theories behind the working of n-gram and HMM based taggers as well as the Brill tagger. We also discuss the experiments that we did for several languages. We describe the results of the experiments in the next chapter.

As discussed in the previous chapter, notable work has already been done for English as well as Indian Bangla. The results of the works on Indian Bangla might suggest how the different techniques for POS tagging should perform on Bangladesh Bangla. But we cannot say anything decisively as, at present, we do not have training corpora of considerable amount developed using Bangladesh Bangla. So we focused mainly on Bangladeshi Bangla and the baseline taggers without using advanced techniques to see how they perform for similar cases in comparison to other languages. Here we describe the resources that we used for the experiments.

## 3.1 Corpora

For correctly POS tagging, training the tagger well is very important, which requires the use of well annotated corpora. Annotation of corpora can be done at various levels which include POS, phrase or clause level, dependency level etc. For English we used some of the genres of the Brown corpus from NLTK [40], while for Bangla we used two different corpora. The first one was the annotated Prothom-Alo corpus, currently under development at the Center for Research on Bangla Language Processing (CRBLP), BRAC University, Dhaka, Bangladesh, consisting of around 6000 words from a Bangladeshi Newspaper Prothom-Alo [23]. The other was the combination of the training and development corpora provided for [41]. For all the

experiments that we did, our test sets had been disjoint from the training corpora.

As not much work on Bangladeshi Bangla has been done, we had to start with the unannotated Prothom-Alo corpus. We took a small part of it and manually tagged it. We used the Python language which, at present has some issues representing the two Bangla characters 'ঃ' and 'অ'. We had to take care of it by replacing each occurrence of these with the flag characters 't' and 'o', respectively. After fixing the problem, we structured the corpus in the Brown corpus format. This became our initial corpus. We used a bootstrapping method to enlarge its size. We ran some unannotated data through the n-gram, HMM, and Brill taggers and took the output of the tagger that performed best. Then we manually corrected the generated output and added it to our annotated training corpus. In the next iteration, the taggers were trained using this corpus. We repeated this process several times to get a corpus of around 6000 tagged tokens. Later we also used the training and development data provided for the SPSAL contests 2006 [41]. We experimented with the corpora for all the three languages Hindi, Bangla and Telegu provided by the authority, but before that, we again had to fix the aforementioned problem. Then we converted the data from the given SSF format [42] to the format used by the Brown corpus in NLTK [40].

## 3.2 Tagsets

Apart from a corpora, a well chosen tagset is also important. According to [15], for deciding upon a tagset, we should consider the following properties:

1. Fineness Vs coarseness

   When choosing the tagset for a POS tagger, we have to decide whether the tags will allow for precise distinction of the various

features of POS of the language i.e. whether features like plurality, gender and other information should also be available or whether the tagger would only provide the different lexical categories.

2. Syntactic function Vs lexical category
   The lexical category of a word can be different than the POS of the word in a sentence, and the tagset should be able to represent both.

   e.g. uttara (North) - noun (lexical category)
   uttara bhArata me bhArI varRA HuI. - adjective (syntactic category)
   ("north" "India" "in" "lots" "rain" "happened")

3. New tags Vs tags from a standard tagger
   It has to be decided whether an existing tagset should be used, or a new tagset should be applied according to the specifics of the language on which the tagger will work.

For English, we used the Brown tagset [35], while for Bangla we used our bangla tagset [43], which is a two level tagset. The first level is the high-level tagset for Bangla, which consists of only (12+2 = 14) tags (Noun, Adjective, Cardinal, Ordinal, Fractional, Pronoun, Indeclinable, Verb, Post Positions, Quantifiers, Adverb, Punctuation, Abbreviation and Others). The second level is more fine-grained with 41 tags. Most of our experiments are based on the level 2 tagset (41 tags). However, we also experimented several cases with the level 1 tagset (14 tags). We also used the 26 tags tagset in [15], for experimenting with Bangla, Hindi and Telegu.

Apart from the corpora and the tagsets, we used the Natural Language Toolkit (NLTK) [40], which is a set of computational linguistics and NLP program modules, annotated corpora and tutorials supporting research

and teaching for the Python language. NLTK allows various NLP tasks by providing implementation of various algorithms such as the Brill tagger, HMM based POS tagger, n-gram based taggers etc. For our experiments, we used the parts of the Unigram, Bigram, Brill and the HMM tagging modules of NLTK.

## 3.3 Taggers

### 3.3.1 Unigram and Bigram Taggers

The Unigram tagger (n-gram, n = 1) is a simple statistical tagging algorithm. For each token, it assigns the tag that is most likely for that token. For example, it will assign the tag ‘adj’ to any occurrence of the word ‘frequent’, since ‘frequent’ is used as an adjective (e.g. a frequent word) more often than it is used as a verb (e.g. I frequent this cafe).

Before a Unigram Tagger can be used to tag data, it must be trained on a training corpus. It uses the corpus to determine which tags are most common for each word. We used the default tagger that assigns ‘NP’ to all words that it encounters, as the back-off tagger for the Unigram Tagger, meaning the Unigram Tagger will pass any word not encountered in the training data, to the back-off tagger to tag as ‘NP’. We assigned the default tag to be ‘NP’ as most of the unknown words are members of open word classes and commonly are Proper Nouns (NP).

The Bigram tagger works in exactly the same way as the Unigram Tagger, the only difference is that it considers the context when assigning a tag to the current word. When training, it creates a frequency distribution describing the frequencies with which, each word is tagged in different contexts. The context consists of the word to be tagged and

the tag of the previous word. When tagging, the tagger uses the frequency distribution to tag words by assigning each word, the tag with the maximum frequency given the context. For our case, when a context is encountered for which no data has been learnt, the tagger backs off to the Unigram tagger.

We compared the Unigram and Bigram taggers to the more advanced taggers like HMM and Brill. We also used the Unigram tagger as the pre-tagger of the Brill tagger. We used the unigram and n-gram (specifying n=2 for creating a Bigram model) tagging modules of NLTK [40] to create Unigram and Bigram taggers and do the tagging. We found that for a small corpus of Bangla, both the taggers tag with similar results. They are also extremely fast compared to the other taggers.

### 3.3.2 HMM

As all other stochastic taggers, the task of HMM based taggers are very simple, i.e. to find the most likely tag for a word or a sequence of words. Unlike other taggers, HMM usually tags one sentence at a time. Given the sentence, it chooses the tag sequence that maximizes the following formula:

P (word | tag) * P (tag | previous n tags)

The HMM approach is different than the other POS tagging approaches in the sense that it considers the best combination of tags for a sequence of words, whereas the other tagging methods greedily tag one word at a time, without regard to the optimal combination. [10]

In the HMM based tagging approach, we have the following entities:

$\{w_1, w_2, \ldots, w_w\}$ is a set of words
$\{t_1, t_2, \ldots, t_T\}$ is a set of POS tags

$W_{1,n} = W_1 W_2 \ldots Wn$ is a sentence of n words
$T_{1,n} = T_1 T_2 \ldots T_n$ is a sequence of n POS tags

As each of the words $W_i$ can take any of the words in $\{w_1, w_2, \ldots, w_w\}$ as its value, we denote the value of $W_i$ by $w_i$ and a particular sequence of values for $W_{i,j\ (i<=\ j)}$ by $w_{i,j}$. Similarly, we denote the value of $T_i$ by t*i* and a particular sequence of values for $T_{i,j\ (i<=j)}$ by $t_{i,j}$. Then the probability $Pr(t_{1,n}, w_{l,n})$ using the following formula can be used to find the most likely sequence of POS tags for a given sequence of' words.

$$\Pr(t_{1,n}, w_{1,n}) \approx \prod_{i=1}^{n} \left( \begin{array}{l} \Pr(t_i \mid t_{i-K,i-1}) \\ \times \Pr(w_i \mid t_i) \end{array} \right)$$

In HMM, the probability of the current tag $t_i$ depends on only the previous k tags $t_{i-k,i-1}$ and the probability of' the current word $w_i$ depends on only the current tag $t_i$. [9, 10]

Bangla, unlike English and some other European languages, is a free word order language meaning that the words in a sentence can change their order but still keep the sentence meaningful. [6] gives an example of this which is as follows:

Consider the simple English sentence I/PRP eat/VB rice/NN

The possible Bengali equivalents of this sentence could be one of the following:

Ami/NN bhAta NN khAi/VB (I rice eat)
Ami/NN khAi/VB bhAta/NN (I eat rice)
bhAta NN Ami/NN khAi/VB (Rice I eat )
bhAta NN khAi/VB Ami/NN (Rice eat I )
khAi/VB Ami/NN bhAta NN ( Eat I rice )
khAi/VB bhAta NN Ami/NN ( Eat rice I )

Using linguistic rules for a free word order language is more troublesome than a language that is not so. For languages like this, [6] suggests that the HMM based approach is more appropriate than other approaches for POS tagging.

We used the HMM tagger of NLTK [40] on parts of the Brown corpus as well as the Bangla corpora to test its performance. We started from a small size and increased the size of the corpus to find out how the performance improves with the increase in size of training tokens. We've noticed that for Bangla HMM performs with similar results even when the size of the tagset changes. This can be observed in the results given in the next chapter. We've also experimented with merging or disjointing the training and testing data sets to find out how the performance of HMM changes with data that have been seen previously.

### 3.3.3 Brill's Tagger

The stochastic taggers have high accuracy and are very fast to tag after having been trained. But a common drawback for all stochastic taggers is the size. A stochastic nth order tagger using back-off may store huge tables containing n-gram entries and large sparse arrays having millions of entries. So these taggers are not a very good choice if they have to be deployed on mobile computing devices which have relatively small storage space and computation power. This is where the rule or transformation based taggers are useful.

The Brill tagger is a transformation based tagger that performs very well but uses only a tiny fraction of the space required by the nth-order stochastic taggers [16, 17]. The general idea of the tagger is very simple. It uses a set of rules to tag data. Then it checks the tagged data for potential errors and corrects those. In the same time it may learn some new rules. Then it uses these new rules to again tag the corrected

data. This process continues until a threshold in improvement in each pass has been reached.

The process of Brill tagging is usually explained by analogy with painting. Suppose we were painting a tree, with all its details of boughs, branches, twigs and leaves, against a uniform sky-blue background. Instead of painting the tree first then trying to paint blue in the gaps, it is simpler to paint the whole canvas blue, then "correct" the tree section by over-painting the blue background.

In the same fashion we might paint the trunk a uniform brown before going back to over-paint further details with a fine brush. Brill tagging uses the same idea: get the bulk of the painting right with broad brush strokes, then fix up the details. As time goes on, successively finer brushes are used, and the scale of the changes becomes arbitrarily small. The decision of when to stop is somewhat arbitrary [2].

The Brill tagging model works in two phases. In the first phase, the tagger tags the input tokens with their most likely tag. This is usually done using a Unigram tagging model. Then in the second phase, a set of transformation rules are applied to the tagged data [16]. An improvement to this technique is described in [17], where unannotated text is passed through the initial state annotator at first. The initial state annotator can range in complexity from assigning random structure to assigning the output of a sophisticated manually created annotator. After getting the output from the initial state annotator, it is compared to the *truth* as specified in a manually annotated corpus. In this stage, transformation rules are applied to the output of the initial state annotator so that it resembles the truth better. After that, a greedy learning algorithm is applied. At each iteration of learning, the transformation is found whose application results in the highest score and the transformation is then added to the ordered transformation list and the training corpus is updated by applying the learned transformation.

After completing the learning stage, unannotated text is tagged by applying the initial state annotator to it and applying each of the learned transformations, in order. A diagram of the whole system is included below [16, 17].

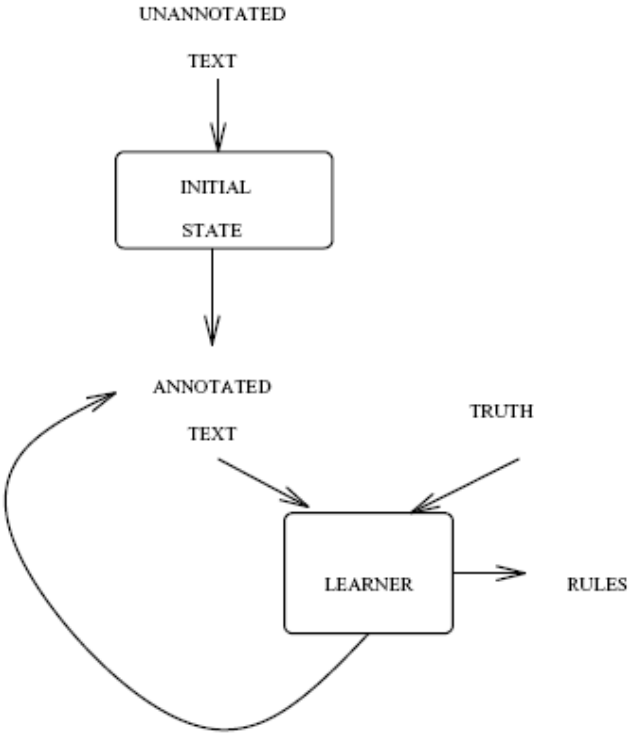


Figure 4: Diagram of improved Brill tagger [17]

## 3.4 Tagging Example

Here we include the outputs of the taggers on two sample Bangla sentences. The training corpus was the Prothom-Alo corpus using 4484 tokens

### 3.6.1. Untagged Text

1. দ্বিতীয় বিশ্বযুদ্ধে মিত্র বাহিনীর নেতা ব্রিটিশ প্রধানমন্ত্রী উইন্সটন চার্চিলকে গত সপ্তাহের শুরুতে টপকে বেয়ার এ তালিকায় স্থান লাভ করেন ।

2. তবে তিনি যদি আবার নির্বাচন করেন এবং জয়ী হন তাহলে হয়তো এ রেকর্ডও ভাঙতে পারবেন ।

### 3.6.2. Tagged output

#### Using level 2 tagset (Full tagset: 41 tags)

**Brill:**

1. দ্বিতীয়/NC বিশ্বযুদ্ধে/NC মিত্র/NC বাহিনীর/NC নেতা/NC ব্রিটিশ/ADJ প্রধানমন্ত্রী/NC উইন্সটন/NP চার্চিলকে/NP গত/ADJ সপ্তাহের/NC শুরুতে/ADVT টপকে/NP বেয়ার/NP এ/DP তালিকায়/NC স্থান/NC লাভ/NC করেন/VF । /PUNSF

2. তবে/INDO তিনি/PP যদি/INDO আবার/ADVM নির্বাচন/NC করেন/VF এবং/CONJC জয়ী/NC হন/VE তাহলে/INDO হয়তো/OTHER এ/DP রেকর্ডও/NC ভাঙতে/NC পারবেন/VF । /PUNSF

**Unigram:**

1. দ্বিতীয়/NP বিশ্বযুদ্ধে/NP মিত্র/NP বাহিনীর/NC নেতা/NC ব্রিটিশ/ADJ প্রধানমন্ত্রী/NC উইন্সটন/NP চার্চিলকে/NP গত/ADJ সপ্তাহের/NC শুরুতে/ADVT টপকে/NP বেয়ার/NP এ/DP তালিকায়/NC স্থান/NC লাভ/NP করেন/VF । /PUNSF

2. তবে/INDO তিনি/PP যদি/INDO আবার/ADVM নির্বাচন/NC করেন/VF এবং/CONJC জয়ী/NP হন/VE তাহলে/INDO হয়তো/OTHER এ/DP রেকর্ডও/NP ভাঙতে/NP পারবেন/NP । /PUNSF

**HMM:**

1. দ্বিতীয়/DP বিশ্বযুদ্ধে/NC মিত্র/NC বাহিনীর/NC নেতা/NC ব্রিটিশ/ADJ প্রধানমন্ত্রী/NC উইন্সটন/NP চার্চিলকে/NP গত/ADJ সপ্তাহের/NC শুরুতে/ADVT টপকে/ADVT বেয়ার/NP এ/NP তালিকায়/NC স্থান/NC লাভ/NC করেন/VF । /PUNSF

2. তবে/INDO তিনি/PP যদি/INDO আবার/ADVM নির্বাচন/NC করেন/VF এবং/CONJC জয়ী/NC হন/VF তাহলে/PUNSF হয়তো/OTHER এ/DP রেকর্ডও/NC ভাঙতে/VNF পারবেন/VF । /PUNSF

**Using Level 1 Tagset (Reduced tagset: 14 tags)**

**Brill:**

1. দ্বিতীয়/NN বিশ্বযুদ্ধে/NN মিত্র/NN বাহিনীর/NN নেতা/NN ব্রিটিশ/ADJ প্রধানমন্ত্রী/NN উইন্সটন/NN চার্চিলকে/NN গত/ADJ সপ্তাহের/NN শুরুতে/ADV টপকে/NN বেয়ার/NN এ/PN তালিকায়/NN স্থান/NN লাভ/NN করেন/VB । /PUNC

2. তবে/IND তিনি/PN যদি/IND আবার/ADV নির্বাচন/NN করেন/VB এবং/IND জয়ী/NN হন/VB তাহলে/IND হয়তো/OTHER এ/PN রেকর্ডও/NN ভাঙতে/VB পারবেন/VB । /PUNC

**Unigram:**

1. দ্বিতীয়/NN বিশ্বযুদ্ধে/NN মিত্র/NN বাহিনীর/NN নেতা/NN ব্রিটিশ/ADJ প্রধানমন্ত্রী/NN উইন্সটন/NN চার্চিলকে/NN গত/ADJ সপ্তাহের/NN শুরুতে/ADV টপকে/NN বেয়ার/NN এ/PN তালিকায়/NN স্থান/NN লাভ/NN করেন/VB । /PUNC

2. তবে/IND তিনি/PN যদি/IND আবার/ADV নির্বাচন/NN করেন/VB এবং/IND জয়ী/NN হন/VB তাহলে/IND হয়তো/OTHER এ/PN রেকর্ডও/NN ভাঙতে/VB পারবেন/NN । /PUNC

**HMM:**

1. দ্বিতীয়/PN বিশ্বযুদ্ধে/NN মিত্র/NN বাহিনীর/NN নেতা/NN ব্রিটিশ/ADJ প্রধানমন্ত্রী/NN উইন্সটন/NN চার্চিলকে/NN গত/ADJ সপ্তাহের/NN শুরুতে/ADV টপকে/ADV বেয়ার/NN এ/NN তালিকায়/NN স্থান/NN লাভ/NN করেন/VB । /PUNC

2. তবে/IND তিনি/PN যদি/IND আবার/ADV নির্বাচন/NN করেন/VB এবং/IND জয়ী/NN হন/VB তাহলে/IND হয়তো/OTHER এ/PN রেকর্ডও/NN ভাঙতে/VB পারবেন/VB । /PUNC

# Chapter III: Results

Different types of POS tagging models have been implemented for English and other western languages, which perform well over the 90% mark. On the contrary, only a small amount of work has been done for Bangla and some other South Asian languages. From the results of the SPSAL machine learning contest 2006 [41], we find out the performance of various taggers to be 60%-78% for Bangla, 62%-79% for Hindi and 53%-78% for Telegu. As shown later in the chapter, our baseline tagging models perform well in these intervals for the development data provided, using no special tools like Morphological Analyzers and others.

The results of our experiments are shown below in the forms of tables and graphs.

## 4.1 Bangla - Prothom Alo Corpus and Level 1 Tagset

| Tokens | HMM Accuracy | Unigram Accuracy | Brill Accuracy |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 60 | 15.4 | 51.2 | 50.4 |
| 104 | 18 | 51.1 | 44.6 |
| 503 | 34.2 | 60.7 | 56.3 |
| 1011 | 42.3 | 64.2 | 62.6 |
| 2023 | 45.8 | 69.1 | 67.8 |
| 3016 | 49.4 | 70.1 | 70.9 |
| 4484 | 45.6 | 71.2 | 71.3 |

Table 1: Performance of POS Taggers for Bangla [Test data: 85 sentences, 1000 tokens from the (Prothom-Alo) corpus; Tagset: Level 1 Tagset (14 Tags)]

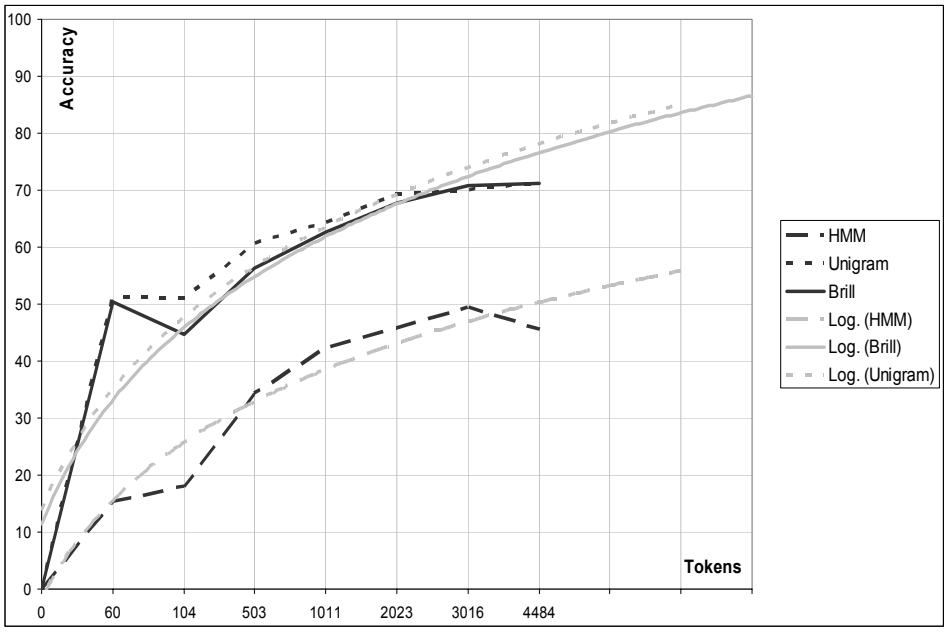


Figure 5: Performance of POS Taggers for Bangla [Test data: 85 sentences, 1000 tokens from the (Prothom-Alo) corpus; Tagset: Level 1 Tagset (14 Tags)]

## 4.2 Bangla - Prothom Alo Corpus and Level 2 Tagset

| Tokens | HMM Accuracy | Unigram Accuracy | Brill Accuracy |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 60 | 19.7 | 17.2 | 38.7 |
| 104 | 18.1 | 17.4 | 26.2 |
| 503 | 28.8 | 26.1 | 46.1 |
| 1011 | 32.8 | 30 | 51.1 |
| 2023 | 40.1 | 36.7 | 49.4 |
| 3016 | 44.5 | 39.1 | 51.9 |
| 4484 | 46.9 | 42.2 | 54.9 |

Table 2: Performance of POS Taggers for Bangla [Test data: 85 sentences, 1000 tokens from the (Prothom-Alo) corpus; Tagset: Level 2 Tagset (41 Tags)]

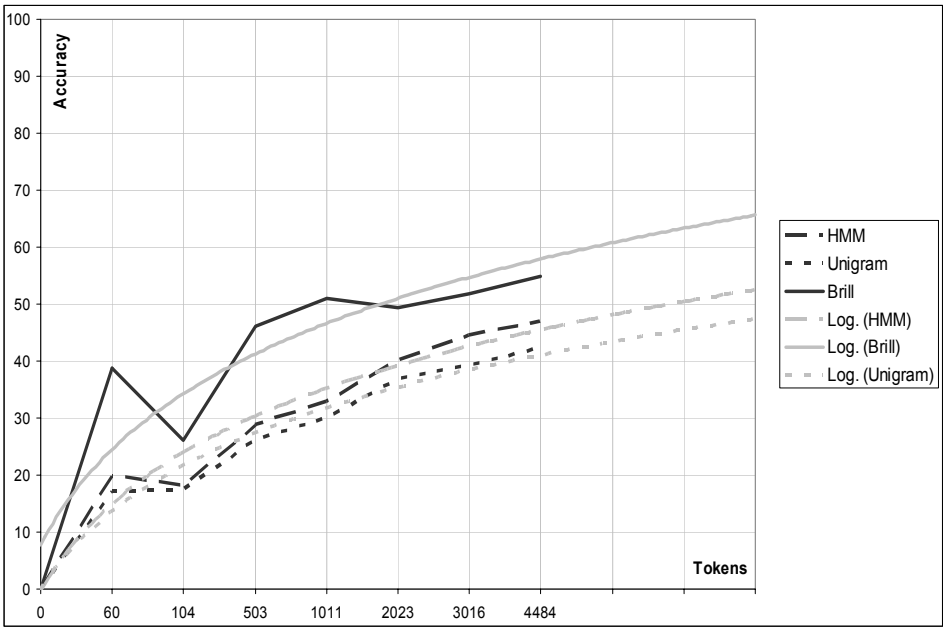


Figure 6: Performance of POS Taggers for Bangla [Test data: 85 sentences, 1000 tokens from the (Prothom-Alo) corpus; Tagset: Level 2 Tagset (41 Tags)]

## 4.3 English - Brown Corpus and Tagset

| Tokens | HMM Accuracy | Unigram Accuracy | Brill Accuracy |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 65 | 36.9 | 28.7 | 33.6 |
| 134 | 44.2 | 34 | 42.9 |
| 523 | 53.4 | 41.6 | 53.7 |
| 1006 | 62 | 47.7 | 58.3 |
| 2007 | 66.8 | 52.4 | 62.9 |
| 3003 | 68.2 | 55.1 | 66.1 |
| 4042 | 70 | 57.2 | 67.5 |
| 5032 | 71.5 | 59.2 | 70.2 |
| 6008 | 71.9 | 60.8 | 71.4 |
| 7032 | 74.5 | 61.5 | 71.8 |
| 8010 | 74.8 | 62.1 | 72.4 |
| 9029 | 76.8 | 63.5 | 74.5 |
| 10006 | 77.5 | 65.2 | 75.2 |
| 20011 | 80.9 | 69.5 | 79.8 |
| 30017 | 83.1 | 71.7 | 78.8 |
| 40044 | 84.7 | 73.3 | 79.8 |
| 50001 | 84.6 | 74.4 | 80.4 |
| 60022 | 85.3 | 75.2 | 80.8 |

| | | | |
|---|---|---|---|
| 70026 | 86.3 | 75.8 | 81 |
| 80036 | 87.1 | 77.1 | 81.6 |
| 90000 | 87.8 | 78.1 | 82.4 |
| 100057 | 87.5 | 78.9 | 83.4 |
| 200043 | 91.7 | 83 | 86.8 |
| 300359 | 89.5 | 84.2 | 87.3 |
| 400017 | 89.7 | 84.8 | 88.5 |
| 500049 | 90.3 | 85.6 | |
| 600070 | 90 | 85.9 | |
| 700119 | 90.3 | 86.1 | |
| 800031 | 90.2 | 86.2 | |
| 900073 | 90.3 | 86.6 | |
| 1000107 | 90.3 | 86.5 | |

Table 3: Performance of POS Taggers for English [Test data: 22 sentences, 1008 tokens from the Brown corpus; Tagset: Brown Tagset]

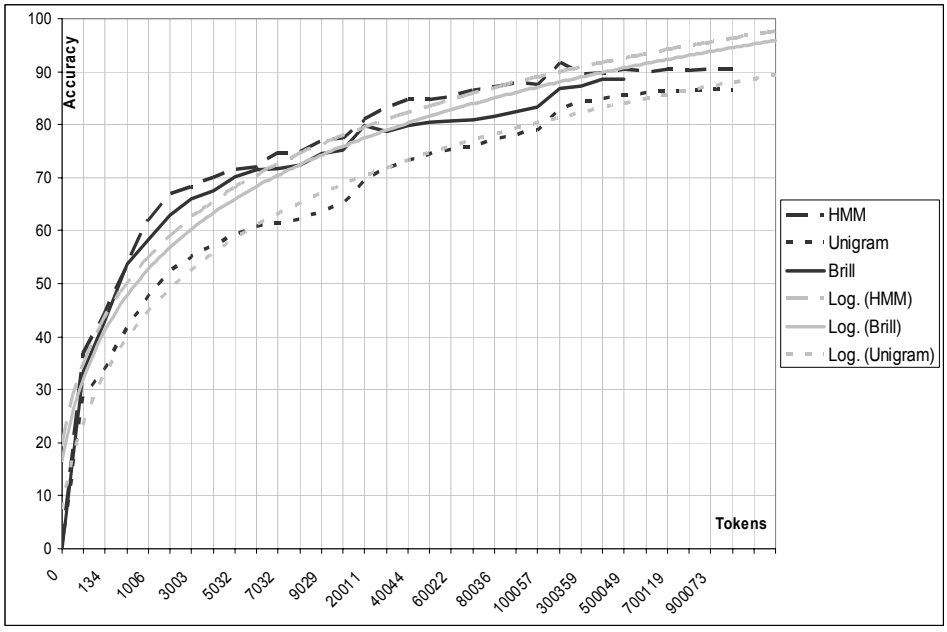


Figure 7: Performance of POS Taggers for English [Test data: 22 sentences, 1008 tokens from the Brown corpus; Tagset: Brown Tagset]

Apart from these experiments, where we incremented the size of the corpora in small increments to plot the performance, we also experimented with taggers on Bangla, Hindi and Telegu where we

trained the taggers using the training data as well as the development data provided for [41] and tested the performance using the testing data for the same.

## 4.4 Hindi - SPSAL Corpus and Tagset

| **Test data: 209 sentences, 4924 tokens from the SPSAL test corpus** | | | | | |
|---|---|---|---|---|---|
| **Sentences** | **Tokens** | **HMM Accuracy** | **Unigram Accuracy** | **Bigram Accuracy** | **Brill Accuracy** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 60 | 36 | 18 | Insufficient data | 37.6 |
| 7 | 113 | 32.2 | 23.8 | Insufficient data | 43.6 |
| 12 | 201 | 30.6 | 27.6 | Insufficient data | 46.7 |
| 21 | 415 | 39.8 | 35.8 | 35.8 | 53.8 |
| 30 | 607 | 43.6 | 37.6 | 37.7 | 56.2 |
| 38 | 826 | 50.5 | 40.3 | 40.5 | 60.3 |
| 43 | 1039 | 53.3 | 41.9 | 42.1 | 59.7 |
| 85 | 2017 | 57.8 | 46 | 46.4 | 61.8 |
| 182 | 4031 | 61.9 | 49.2 | 49.3 | 64.9 |
| 259 | 6017 | 62.8 | 50.9 | 51 | 68.8 |
| 362 | 8009 | 64.4 | 52 | 52.3 | 69.4 |
| 450 | 10001 | 64.4 | 52.7 | 53.1 | 69.1 |
| 535 | 12003 | 65.7 | 54.1 | 54.5 | 69.6 |
| 619 | 14011 | 66.3 | 54.5 | 54.9 | 69.7 |
| 698 | 16020 | 67.3 | 55.5 | 55.5 | 70.6 |
| 784 | 18019 | 67.3 | 55.8 | 56.2 | 70.6 |
| 865 | 20004 | 68 | 56.9 | 57.1 | 70.7 |
| 934 | 22010 | 67.5 | 57 | 57.3 | 70.8 |
| 1007 | 24030 | 68.6 | 57.7 | 55.7 | 71.1 |
| 1125 | 26005 | 68.5 | 58.4 | 57.5 | 71.3 |
| 1135 | 26148 | 68.5 | 58.5 | 57.5 | 71.5 |

Table 4: Performance of POS Taggers for Hindi [Test data and Tagset source: [41]]

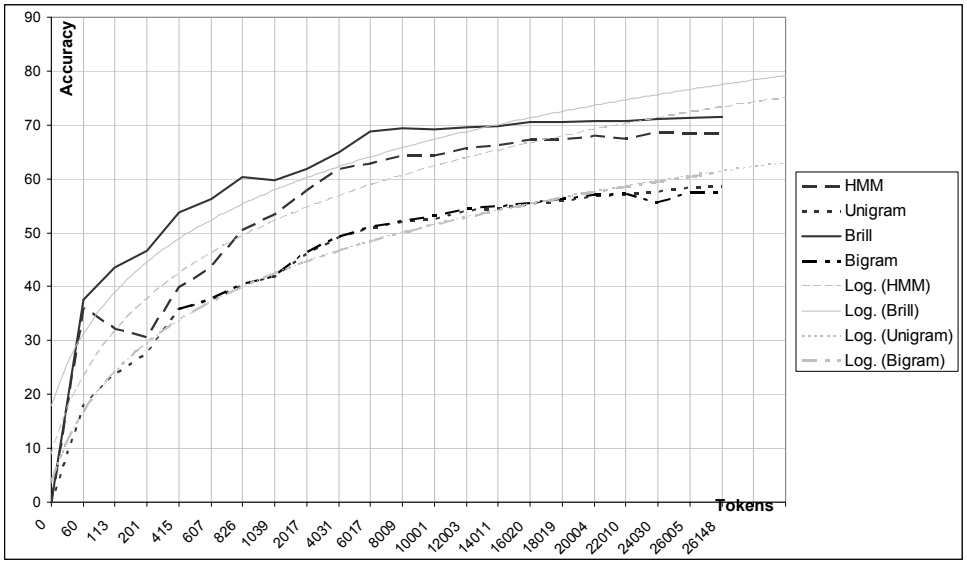


Figure 8: Performance of POS Taggers for Hindi [Test data and Tagset source: [41]]

## 4.5 Telegu - SPSAL Corpus and Tagset

| **Test data: 415 sentences, 5193 tokens from the SPSAL test corpus** | | | | | |
|---|---|---|---|---|---|
| | | **HMM** | **Unigram** | **Bigram** | **Brill** |
| **Sentences** | **Tokens** | **Accuracy** | **Accuracy** | **Accuracy** | **Accuracy** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 50 | 28.4 | 15.6 | Insufficient data | 45.7 |
| 9 | 102 | 28.1 | 16.4 | Insufficient data | 47.7 |
| 23 | 202 | 32.1 | 16.9 | Insufficient data | 48 |
| 54 | 401 | 30.8 | 18 | Insufficient data | 49.2 |
| 87 | 612 | 29.6 | 18.3 | 18.3 | 49.1 |
| 107 | 811 | 30.9 | 18.8 | 18.8 | 49.6 |
| 131 | 1004 | 31.7 | 19.1 | 19.1 | 38.2 |
| 248 | 2010 | 32.8 | 23.4 | 23.4 | 53.5 |
| 421 | 4001 | 42.6 | 28.1 | 28.2 | 57.9 |
| 605 | 6007 | 48 | 31.7 | 31.7 | 60.4 |
| 783 | 8002 | 51.1 | 34.9 | 34.5 | 62.6 |
| 994 | 10018 | 53 | 37.4 | 37.2 | 63.9 |
| 1192 | 12000 | 53.6 | 38.8 | 38.3 | 64.6 |
| 1409 | 14010 | 53.3 | 38.8 | 38.7 | 64.4 |
| 1626 | 16005 | 53.9 | 39.6 | 39.2 | 65 |
| 1842 | 18004 | 53.7 | 40.1 | 39.7 | 65.1 |
| 2048 | 20012 | 54.9 | 40.4 | 40.2 | 65.1 |
| 2184 | 22013 | 54.8 | 41.5 | 41 | 65.8 |
| 2335 | 24002 | 55.6 | 41.6 | 41 | 65.8 |

| | | | | | |
|---|---|---|---|---|---|
| 2485 | 26025 | 55.9 | 41.9 | 41.3 | 66 |
| 2655 | 27511 | 56.6 | 42.8 | 42.2 | 66.9 |

Table 5: Performance of POS Taggers for Telegu [Test data and Tagset source: [41]]

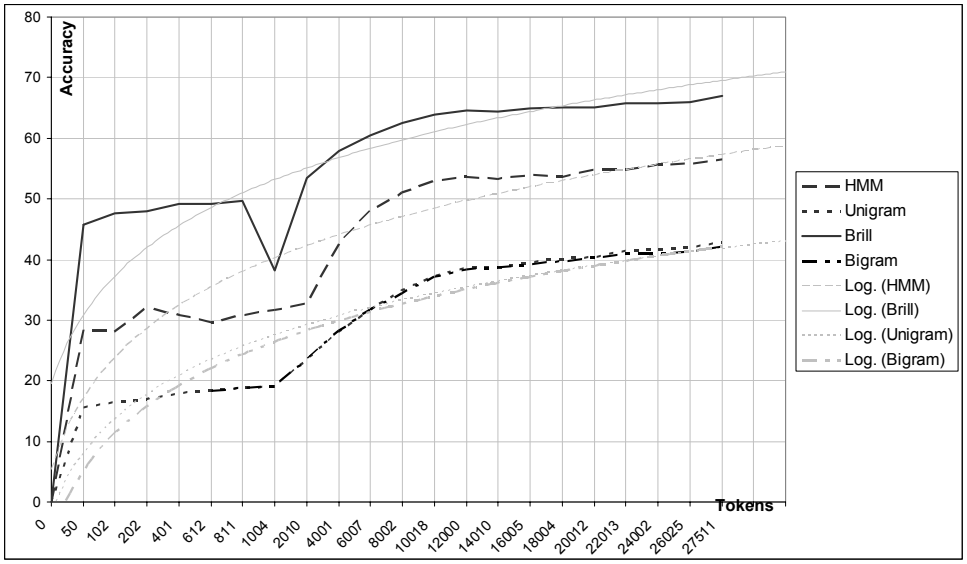


Figure 9: Performance of POS Taggers for Telegu [Test data and Tagset source: [41]]

## 4.6 Bangla - SPSAL Corpus and Tagset

| **Test data: 400 sentences, 5225 tokens from the SPSAL test corpus** | | | | | |
|---|---|---|---|---|---|
| | | **HMM** | **Unigram** | **Bigram** | **Brill** |
| **Sentences** | **Tokens** | **Accuracy** | **Accuracy** | **Accuracy** | **Accuracy** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 51 | 14.3 | 14 | Insufficient data | 35.6 |
| 13 | 108 | 20.7 | 17.9 | Insufficient data | 39.6 |
| 21 | 206 | 26.5 | 19.3 | 19.3 | 40.9 |
| 37 | 405 | 30.7 | 21.8 | 21.8 | 42.7 |
| 53 | 605 | 32.7 | 24.1 | 24.1 | 45.4 |
| 69 | 807 | 36.4 | 27.7 | 27.7 | 48.6 |
| 87 | 1002 | 39.3 | 28.6 | 28.6 | 50.2 |
| 173 | 2004 | 44.3 | 36 | 36 | 55.8 |
| 304 | 4003 | 49.7 | 42.4 | 41.9 | 61.3 |
| 398 | 6036 | 49.8 | 45.6 | 45.3 | 63.8 |
| 532 | 8026 | 53.6 | 48.1 | 47.9 | 64.7 |
| 677 | 10001 | 54.3 | 49.8 | 49.5 | 65.6 |

| | | | | | |
|---|---|---|---|---|---|
| 846 | 12006 | 56.7 | 51.7 | 51.1 | 66.4 |
| 960 | 14027 | 57.5 | 52.9 | 51.7 | 67.2 |
| 1130 | 16000 | 58.6 | 53.9 | 52.6 | 68.2 |
| 1301 | 18006 | 60.5 | 54.5 | 53 | 68.7 |
| 1427 | 20001 | 61.9 | 55.8 | 54.4 | 69.1 |
| 1535 | 22014 | 62.4 | 56.2 | 54.7 | 68.3 |
| 1656 | 24001 | 63.3 | 56.7 | 55.2 | 68.4 |
| 1786 | 25426 | 63.6 | 56.9 | 55.5 | 69.6 |

Table 6: Performance of POS Taggers for Bangla [Test data and Tagset source: [41]]

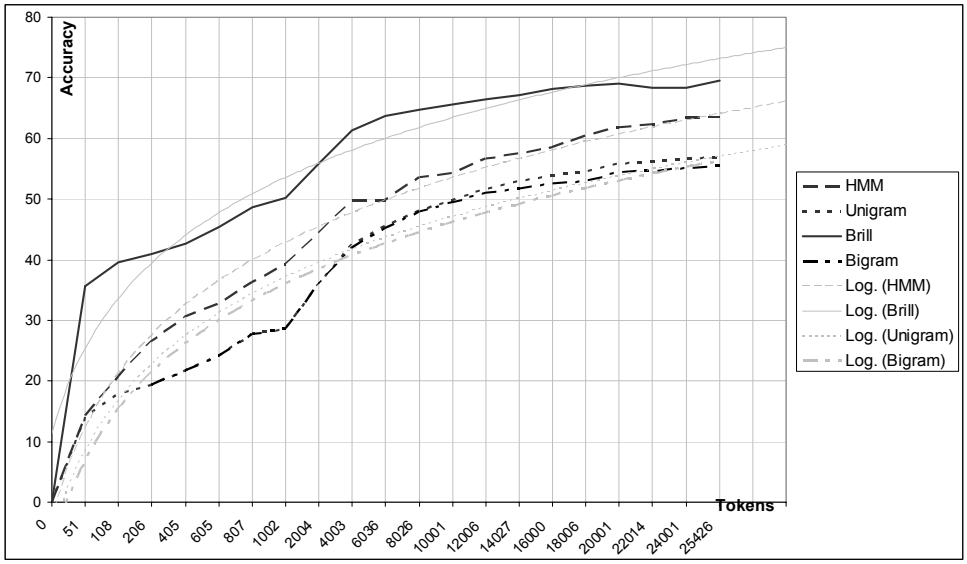


Figure 10: Performance of POS Taggers for Bangla [Test data and Tagset source: [41]]

We also experimented merging the development data with the training data and using this for training.

## 4.7 Bangla - SPSAL Corpus and Tagset (Merged)

| **Test data: 340 sentences, 5029 tokens** | | | | | |
|---|---|---|---|---|---|
| **Sentences** | **Tokens** | **HMM Accuracy** | **Unigram Accuracy** | **Bigram Accuracy** | **Brill Accuracy** |
| 1785 | 25426 | 92.9 | 74.4 | 73.2 | 83 |

Table 7: Performance of POS Taggers for Bangla on merged training and testing data [Test data and Tagset source: [41]]

The resulting high accuracy gain of the HMM model (62.7 to 92.9) once more reveals that stochastic models are far superior to any other when the knowledge about unknown words is available.

# Chapter IV: Analysis Of Results

We have experimented with basic tagging models which report performance in the 96%+ range for English, but for our cases they performed in the 50%-70% range for South Asian Languages and around 90% for English using the Brown tagset [35] and genres from the Brown Corpus taken from NLTK [40]**.** The reason behind this is in the cases where performance was reported in the 95% range, very large corpora were employed for training the model.

For Bangla, we did not have any annotated corpus available, and the reason of very low performance of Bangla on our cases is mostly due to the small corpus size and sparseness of training data, which makes it very difficult for stochastic taggers to create probability distribution to hold transitions between different states [44].

We have compared the performance of English and Bangla as well as two other South Asian languages, using same corpus size and showed that the performance is similar in some cases. So if we can extend the corpus size of South Asian languages then we will probably be able to get similar performance for these languages as English.

Within this limited corpus, our experiments suggested that for the three South Asian languages Bangla, Hindi and Telegu, with limited tagged corpus, Brill's tagger performs better than HMM based tagger and n-gram based Unigram and Bigram taggers. Researchers, who want to implement a tagger for a language with limited language resources, i.e. annotated corpora of large size, can try Brill's tagger or any other rule based tagger for their languages too.

# Chapter V: Future Work

We compared different POS tagging models like n-gram, HMM and Brill's transformation based techniques for three South Asian languages (Bangla, Hindi and Telegu). At present with the training corpus with a size of around 20000 words of a single domain we get a performance of over 90% when the test set is extracted from the training corpus, and we get a performance of over 70% if the test set is taken randomly from any other source. If we can increase the training corpus size covering most of the domains then we might get a recognizable performance of 95%+ for Bangla too.

From [20], we find that a tagger should have the following qualities to be of any practical purpose.

Robust: Text corpora may contain ungrammatical constructions, isolated phrases (titles), nonlinguistic data (tables) as well as unknown words. The tagger should be able to deal with these.

Efficient: The tagger should be efficient in the sense that it should be able to train fast on newly available corpora and text genres. It should be able to train in a relatively short time on large corpora, and it should also perform in time linear complexity on the number of words to tag.

Accurate: The tagger should assign the best possible POS tag for every word in the text to tag.

Tunable: The tagger should be able to avoid erroneous tagging by accepting a priori hints.

To improve the performances of tagging models, these features could be implemented.

In [15], we find information about a method that augments a probabilistic tagger with a handcrafted procedure to pre-tag problematic idioms. As stated there, the procedure improves the accuracy of the tagger by 3%, resulting in a total of 95%-96% accuracy.

[45] reports that Finite State Transducers (FST) can be used to represent the transformation rules used by the Brill tagger, which improves the running time of the tagger Also, to decrease the training time on corpora of large sizes, Directed Acyclic Graphs (DAG) could be used to represent the corpora.

[46] describes a new approach that suggests the use of Dynamic Bayesian Network instead of HMM that performs with similar accuracy.

A suffix stripper as described in the earlier chapters could be implemented that might prove useful for Bangla and experiments could be done with the tagset as it can improve the performance of the tagger [3], to some extent.

More experiments could be performed on the South Asian languages to find out whether some specific POS tagging model or modification to a specific model performs better than others for a specific language. The unsupervised approaches are left out of the present discussion, mainly because of their high requirements of computational power and slow speed to train. But for languages, in which resources are limited, unsupervised POS tagging models [47, 48, 49] are very good options. These models could also be experimented with for Bangla or other South Asian languages.

The Baum Welch re-estimation algorithm is widely used for unsupervised training of POS taggers. [50] describes three patterns of behavior in Baum Welch re-estimation. These are:

Classical: A general trend of rising accuracy on each iteration, with any falls in accuracy being local. It indicates that the model is converging towards an optimum which is better than its starting point.

Initial maximum: Highest accuracy on the first iteration, and falling thereafter. In this case the initial model is of better quality than BW can achieve. That is, while BW will converge on an optimum, the notion of optimality is with respect to the HMM rather than to the linguistic judgements about correct tagging.

Early maximum: Rising accuracy for a small number of iterations (2-4), and then falling as in initial maximum.

Using these patterns, the paper describes some guidelines for unsupervised training of HMM models. These are:

1. If a hand-tagged training corpus is available, use it . If the test and training corpora are near identical, do not use BW re-estimation; otherwise use for a small number of iterations.

2. If no such training corpus is available, but a lexicon with at least relative frequency data is available, use BW re-estimation for a small number of iterations.

3. If neither training corpus nor lexicon are available, use BW re-estimation with standard convergence tests such as perplexity. Without a lexicon, some initial biasing of the transitions is needed if good results are to be obtained.

The next step could be to find out whether these patterns are present in South Asian Languages and also, whether the above mentioned guidelines are applicable for these languages as well. There are some other state of the art POS tagging techniques, which could also be tried out for Bangla.

In another study we have seen that in case of n-gram based POS tagging, backward n-gram (considers next words), performs better than usual forward n-gram (considers previous words), Based upon this observation, further experiments can be carried out to determine whether the feature is worthwhile to implement in a tagging model.

Finally, a hybrid solution for POS tagging in Bangla can be proposed that can be used in other advanced NLP applications, which might use a combination of the techniques mentioned earlier to achieve a significant gain in performance and performs with very good accuracy as English or other western languages in all domains.

# References

[1] The Summer Institute for Linguistics (SIL) Ethnologue Survey (1999).

[2] Daniel Jurafsky and James H. Martin, “Chapter 8: Word Classes and Part-Of-Speech Tagging, Speech and Language Processing”, Prentice Hall, 2000.

[3] Andrew MacKinlay, “The Effects of Part-of-Speech Tagsets on Tagger Performance”, Undergraduate Thesis, University of Melbourne, 2005.

[4] Yair Halevi, “Part of Speech Tagging”, Seminar in Natural Language Processing and Computational Linguistics (Prof. Nachum Dershowitz), School of Computer Science, Tel Aviv University, Israel, April 2006.

[5] Himanshu Agrawal and Anirudh Mani, “Part of Speech Tagging and Chunking with Conditional Random Fields”, In Proceedings of the NLPAI Machine Learning 2006 Competition.

[6] Sandipan Dandapat, Sudeshna Sarkar and Anupam Basu, “A Hybrid Model for Part-of-Speech Tagging and its Application to Bengali”, International Journal Of Information Technology Volume 1 Number 4 2004.

[7] Atro Voutilainen, “A Syntax Base POS Analyzer”, University of Helsinki, Finland.

[8] Atro Voutilainen, “Does tagging help parsing? A Case Study On Finite State Parsing”, University of Helsinki, Finland.

[9] Linda Van Guilder, “Automated Part of Speech Tagging: A Brief Overview”, Handout for LING361, Fall 1995, Georgetown University.

[10] Karthik Kumar G, Sudheer K, Avinesh Pvs, “Comparative Study of Various Machine Learning Methods For Telugu Part of Speech Tagging”, In Proceedings of the NLPAI Machine Learning 2006 Competition.

[11] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, Chapter 8, “Spoken Language Processing: A Guide to Theory, Algorithm and System Development”. Prentice Hall, 2001.

[12] Rakesh Dugad, U.B.Desai, “A Tutorial on Hidden Markov Models”, IIT Bombay.

[13] Lawrence R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, Proceedings of the IEEE, Vol. 77, No. 2, February 1989.

[14] Manoj Kumar C, “Stochastic Models for POS Tagging”, IIT Bombay.

[15] “A Part of Speech Tagger for Indian Languages (POS tagger)”, In Guidelines of the Workshop on Shallow Parsing in South Asian Languages (SPSAL) 2007.

[16] Eric Brill, “A simple Rule-Based Part-of-Speech Tagger”, In Proceedings Of The Third Conference On Applied Natural Language Processing, Trento, Italy, Pages: 152 - 155, 1992.

[17] Eric Brill, “Some Advances in Transformation Based Part of Speech Tagging”, In Proceedings Of The Twelfth National Conference On Artificial Intelligence (vol. 1), Seattle, Washington, United States, Pages: 722 – 727, 1994.

[18] L. Bahl and R. L. Mercer, “Part-Of-Speech Assignment By A Statistical Decision Algorithm”, IEEE International Symposium on Information Theory, pages: 88 - 89, 1976.

[19] K. W. Church, “A Stochastic Parts Program And Noun Phrase Parser For Unrestricted Test”, Proceeding of the Second Conference on Applied Natural Language Processing, pages: 136 - 143, 1988.

[20] D. Cutting, J. Kupiec, J. Pederson and P. Sibun, “A Practical Part-Of-Speech Tagger”, Proceedings of the Third Conference on Applied Natural Language Processing, pages: 133 - 140, ACL, Trento, Italy, 1992.

[21] S. J. DeRose, “Grammatical Category Disambiguation By Statistical Optimization”, Computational Linguistics, 14 (1), 1988.

[22] A. M. Deroualt and B. Merialdo, “Natural Language Modeling For Phoneme-To-Text Transposition”, IEEE transactions on Pattern Analysis and Machine Intelligence, 1986.

[23] Bangla Newspaper, Prothom-Alo. Online version available online at: http://www.prothom-alo.net

[24] L. E. Baum, “An Inequality And Associated Maximization Technique In Statistical Estimation On Probabilistic Functions Of A Markov Process”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 22, Issue: 4 Page: 371-377, April 2000.

[25] Qin Iris Wang Dale Schuurmans, “Improved Estimation for Unsupervised Part-of-Speech Tagging”, University of Alberta, Canada.

[26] Patrick Schone and Daniel Jurafsky, “Language-independent Induction of Part of Speech Class Labels Using Only Language Universals”, In Workshop at IJCAI-2001, Seattle, WA., August 2001.

[27] Juan Antonio P´erez-Ortiz and Mikel L. Forcada, “Part-of-Speech Tagging with Recurrent Neural Networks”, Universitat d’Alacant, Spain.

[28] Helmut Schmid, “Improvements in POS Tagging with an Application to German”, Technical Report, Universitat Stuttgart, Germany.

[29] Aniket Dalal, Kumar Nagaraj, Uma Sawant and Sandeep Shelke, “Hindi Part-of-Speech Tagging and Chunking : A Maximum Entropy Approach”, In Proceeding of theNLPAI Machine Learning 2006 Competition.

[30] Sivaji Bandyopadhay, Asif Ekbal and Debasish Halder, “HMM based POS Tagger and Rule-based Chunker for Bengali”, In Proceeding of the NLPAI Machine Learning 2006 Competition.

[31] Sankaran Baskaran, “Hindi POS Tagging and Chunking”, In Proceedings of the NLPAI Machine Learning 2006 Competition.

[32] Pranjal Awasthi, Delip Rao and Balaraman Ravindran, “Part Of Speech Tagging and Chunking with HMM and CRF”, In Proceedings of the NLPAI Machine Learning 2006 Competition.

[33] Arulmozhi. P, Sobha. L, Kumara Shanmugam. B., “Parts of Speech Tagger for Tamil”, Symposium on Inidan Morphology, Phonology and Language Engineering, March 19-21, 2004 IIT Khadagpur, 55-57, India.

[34] International Conference on Computer and Information Technology (ICCIT), Bangladesh, 2004.

[35] The Brown Tagset, available online at: http://www.scs.leeds.ac.uk/amalgam/tagsets/brown.html

[36] Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz, “Building a Large Annotated Corpus of English: The Penn Treebank”, Computational Linguistics Journal, Volume 19, Number 2, Pages: 313-330, 1994. Available online at: http://www.ldc.upenn.edu/Catalog/docs/treebank2/cl93.html

[37] Sandipan Dandapat, Sudeshna Sarkar, “Part of Speech Tagging for Bengali with Hidden Markov Model”, In Proceedings of the NLPAI Machine Learning 2006 Competition.

[38] International Conference on Computer and Information Technology (ICCIT), Bangladesh, 2003.

[39] Goutam Kumar Saha, Amiya Baran Saha and Sudipto Debnath, “Computer Assisted Bangla Words POS Tagging”, Proceedings of the International Symposium on Machine Translation NLP & TSS (iSTRANS-2004), New Delhi 2004.

[40] NLTK, The Natural Language Toolkit, available online at: http://nltk.sourceforge.net/index.html

[41] Workshop on Shallow Parsing on South Asian Languages (SPSAL) 2007, Twentieth International Joint Conferences on Artificial Intelligence, Hyderabad, India.

[42] Akshar Bharati, Rejeev Sangal and Dipti M Sharma, “Shakti Analyser: SSF Representation”, IIT Hyderabad.

[43] Bangla POS Tagset used in our Bangla POS tagger, available online at http://www.naushadzaman.com/bangla_tagset.pdf

[44] Scott M. Thede and Mary P. Harpe, “A Second-Order Hidden Markov Model for Part-of-Speech Tagging”, Purdue University.

[45] Emmanuel Roche and Yves Schabes, “Deterministic POS Tagging with Finite State Transducers (FST)”, Computational Linguistics Volume 21 , Issue 2 (June 1995), Pages: 227 – 253. 1995.

[46] Virginia Savova and Leonid Peshkin, “Part-of-Speech Tagging with Minimal Lexicalization”, Johns Hopkins University, Massachusetts Institute of Technology.

[47] Robbert Prins and Gertjan van Noord, “Unsupervised Pos-Tagging Improves Parsing Accuracy And Parsing Efficiency”, In Proceedings of the International Workshop on Parsing Technologies, 2001.

[48] Mihai Pop, “Unsupervised Part-of-speech Tagging”, Department of Computer Science, Johns Hopkins University, 1996.

[49] Eric Brill, “Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging”, In Proceeding of The Natural Language Processing Using Very Large Corpora, Boston, MA, 1997.

[50] David Elworthy, “Does Baum-Welch Re-estimation :Help Taggers?”, In Proceedings of the fourth conference on Applied natural language processing, Pages: 53 – 58. 1994.

# Appendix

## Bangla POS Tagset used in our experiments

## Level 1 – High Level Tags

| No. | Tag Name | Short Name | Example |
|---|---|---|---|
| 1 | Proper Noun | NN | মতিউর |
| | Common Noun | | পানি, গরু |
| | Verbal Noun | | করানো, পড়ানো |
| 2 | Adjective | ADJ | লাল, গরম |
| 3 | Cardinal (Det) | CAR | এক, দুই |
| 4 | Ordinal (Det) | ORD | প্রথম, দ্বাদশ |
| 5 | Fractional (Det) | FRA | তৃতীয়াংশ |
| 6 | Personal Pronoun | PN | আমি, তুমি |
| | Demonstrative Pronoun | | এ, এরা, ওরা |
| | Indefinite Pronoun | | কেউ |
| | Relative Pronoun | | যে, যিনি, যা |
| | Reflexive Pronoun | | আপনি, নিজ |
| | Inclusive Pronoun | | সব, সকল, উভয় |
| | Reciprocal Pronoun | | আপনা–আপনি |
| 7 | Connective Conjunction | IND | ও, এবং, আর |
| | Adversative Conjunction | | বা, অথবা |
| | Interjection | | বাহ!, ওহ! |
| | Vocatives | | ওগো, ওরে |
| | Other Indeclinables | | যদি, তবে, সুতরাং, বটে, কিন্তু |
| | Assertive Particle | | হ্যা |
| | Negative Particle | | না, নাই |
| | Question Particle | | কি, কে, কীনো |
| | Onomatopes | | টনটন, কনকন |
| 8 | Finite verb | VB | করি, খেলি |
| | Non-finite verb | | করতে, খেলতে |
| | Causative verb | | করাই, খেলাই |
| | Verb Imperatives | | করো, খেলো |
| | Negative Verb | | যাইনি, করিনি |
| | Existential Verb | | থাকে, হয়, আছে |
| 9 | PostPositions | POSTP | থেকে, দিয়ে, সংগে, সাথে |
| 10 | Quantifiers | QUAN | কিছু, এতো, অনেক, খুব, আরো |
| 11 | Temporal Adverb | ADV | আজ, কাল, সর্বদা, ক্রমশ |
| | Spatial Adverb | | নিচে, উপড়ে |
| | Adverb of Manner | | আস্তে, দ্রুত |

| | | | |
|---|---|---|---|
| 12 | Sentence-Final Punctuation | PUNC | \|, ?, ! |
| | Quote | | ", " |
| | Parenthesis | | ( ) {} [] |
| | Mid-sentence Punctuation | | , ; : |
| | Other Punctuation | | %. |
| 13 | Abbreviation | ABB | মোঃ, ডাঃ |
| 14 | Others | OTHER | |

## Level 2 – Detailed Tags

| No. | Tag Name | Short Name | Example |
|---|---|---|---|
| 1 | Proper Noun | NP | মতিউর |
| 2 | Common Noun | NC | পানি, গরু |
| 3 | Verbal Noun | NV | করানো, পড়ানো |
| 4 | Adjective | ADJ | লাল, গরম |
| 5 | Cardinal (Det) | CAR | এক, দুই |
| 6 | Ordinal (Det) | ORD | প্রথম, দ্বাদশ |
| 7 | Fractional (Det) | FRA | তৃতীয়াংশ |
| 8 | Personal Pronoun | PP | আমি, তুমি |
| 9 | Demonstrative Pronoun | DP | এ, এরা, ওরা |
| 10 | Indefinite Pronoun | IP | কেউ |
| 11 | Relative Pronoun | RP | যে, যিনি, যা |
| 12 | Reflexive Pronoun | REFP | আপনি, নিজ |
| 13 | Inclusive Pronoun | INCP | সব, সকল, উভয় |
| 14 | Reciprocal Pronoun | RECP | আপনা–আপনি |
| 15 | Connective Conjunction | CONJC | ও, এবং, আর |
| 16 | Adversative Conjunction | CONJA | বা, অথবা |
| 17 | Interjection | INTJ | বাহ!, ওহ! |
| 18 | Vocatives | VOC | ওগো, ওরে |
| 19 | Other Indeclinables | INDO | যদি, তবে, সুতরাং, বটে, কিন্তু |
| 20 | Assertive Particle | PRTA | হ্যা |
| 21 | Negative Particle | PRTN | না, নাই |
| 22 | Question Particle | PRTQ | কি, কে, কীনো |
| 23 | Onomatopes | ONO | টনটন, কনকন |
| 24 | Finite verb | VF | করি, খেলি |
| 25 | Non-finite verb | VNF | করতে, খেলতে |
| 26 | Causative verb | VC | করাই, খেলাই |
| 27 | Verb Imperatives | IMP | করো, খেলো |
| 28 | Negative Verb | VN | যাইনি, করিনি |
| 29 | Existential Verb | VE | থাকে, হয়, আছে |

| | | | |
|---|---|---|---|
| 30 | PostPositions | POSTP | থেকে, দিয়ে, সংগে, সাথে |
| 31 | Quantifiers | QUAN | কিছু, এতো, অনেক, খুব, আরো |
| 32 | Temporal Adverb | ADVT | আজ, কাল, সর্বদা, ক্রমশ |
| 33 | Spatial Adverb | ADVS | নিচে, উপড়ে |
| 34 | Adverb of Manner | ADVM | আস্তে, দ্রুত |
| 35 | Sentence-Final Punctuation | PUNSF | \|, ?, ! |
| 36 | Quote | PUNQ | ", " |
| 37 | Parenthesis | PUNPAR | ( ) {} [] |
| 38 | Mid-sentence Punctuation | PUNMS | , ; : |
| 39 | Other Punctuation | PUNO | %. |
| 40 | Abbreviation | ABB | মোঃ, ডাঃ |
| 41 | Others | OTHER | |