IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

A Study on Different Part of Speech (POS) Tagging Approaches in Assamese Language

Bipul Roy[1], Bipul Syam Purkayastha[2]

Scientist B, NIELIT, Itanagar Centre, Arunachal Pradesh, India [1]

Professor, Department of Computer Science, Assam University, Assam, India [2]

Abstract: Syntactic parsing is a necessary task which is required for Natural Language Processing (NLP) applications including Part of Speech (POS) tagger. For the development and enrichment of languages, part of speech tagging plays a very crucial role. Part of speech tagging, especially for the regional Indian languages can give an international and world-wide approach. For a regional language like Assamese which is Assam's official language, part of speech tagging has become very much essential for the overall flourishment of the language. The linguistic experts have developed different types of POS tagging approaches like Rule based, Stochastic based, Neural Network based approaches, etc. Here in this paper our aim is to briefly overview the computational works that has been done till date by the linguists in the field of POS tagging of Assamese language.

Keywords: Syntatic Parsing, POS Tagging, Assamese, Stochastic.

I. INTRODUCTION

The Indian Constitution Recognized language Assamese is an Eastern Indo-Aryan Language spoken by around 32 million people in the Indian states of Assam,
Meghalaya, Arunachal Pradesh and also spoken in Bangladesh and Bhutan partially. But , unfortunately, despite such a widespread, well used and morphological richness, a very less work has been done so far in terms of formal computational study of Assamese language like natural language processing. Natural language processing is the skill of a computer program to understand human language as it is spoken. NLP is a process of developing a system that can read text and translate between one human language and another. Part of speech tagging is an important tool for processing natural languages and the work on part-of-speech (POS) tagging has begun in the early 1960s [6]. It is one of the simplest as well as most stable and statistical model for many NLP applications. POS tagging is an initial stage of information extraction, summarization, retrieval, machine translation, speech conversion [6]. Here in this paper, we are going to briefly overview the Parsing algorithms & POS tagging approaches that have been done till date to Assamese language.

II. BACKGROUND THEORY

A. What is Part of Speech Tagging?

The technique of assigning an appropriate part of speech tag for each word in an input sentence of a language is called Part of Speech Tagging. It is commonly referred to as POS tagging. Part of speech includes nouns, verbs, adjectives, pronouns, conjunctions and their sub-categories [2, 10].

Example:
Word: Bird, Tag: Noun
Word: Sing, Tag: verb
Word: Melodious, Tag: Adjective

Note that some words can have more than one tag associated with. For example, the word "play" can be a noun or verb depending on the context.

B. Part of Speech Tagger

Part of Speech tagger or POS tagger is a tagging program in NLP. Taggers use several kinds of information, dictionaries, lexicons, rules and so on. Dictionaries have a category or categories of particular words, i.e. a word may belong to more than one category. For example, the word "study" is both noun and verb. Taggers use probabilistic information to solve such ambiguity.

There are mainly two types of taggers, viz. Rule-based taggers and Stochastic taggers. Rule-based taggers use hand written rules to distinguish the tag ambiguity. Stochastic taggers are either HMM based, choosing the tag sequence which maximizes the product of word likelihood and tag sequence probability, or Transformation based, using decision trees or maximum entropy models to combine probabilistic features. Ideally a typical tagger should be robust, efficient, accurate, tunable and reusable. In reality taggers either definitely identify the tag for the given word or make the best guess based on the available information. As the natural language is complex, it is sometimes difficult for the taggers to make accurate decisions about tags. So occasional errors in tagging are not taken as a major roadblock to NLP research.

C. Tagset

Tagset is the set of tags from which the tagger is supposed to choose to attach to the relevant word. Every tagger will be given a standard tagset. The tagset may be coarse such as N (Noun), V (Verb), ADJ (adjective), ADV (Adverb), PREP (Preposition), CONJ (Conjunction) or fine-grained such as NNOM (Noun-Nominative), NSOC (Noun-Sociative), VFIN (Verb finite) , VNFIN (Verb Nonfinite) and so on. Most of the taggers use only fine grained tagset.

C. Architecture of POS Tagger

1. Tokenization
The given text is divided into tokens so that they can be used for further analysis. The tokens may be words, punctuation marks, and utterance boundaries.

2. Ambiguity look-up
This is to use lexicon and a guessor for unknown words. While the lexicon provides a list of word forms and their likely part of speech, guessors analyze unknown tokens. Compiler or interpreter, lexicon and guessor make what is known as lexical analyzer. A lexical analyzer is a program which breaks a text into lexemes (tokens).

3. Ambiguity Resolution
It is a property of linguistic expressions. If an expression (word/phrase/sentence) has more than one interpretation we can refer it as ambiguous. The process to remove the ambiguity of words in a given context is called disambiguation. Disambiguation is based on information about word such as the probability of the word. For example, the word "power" is more likely used as a noun than as a verb. Disambiguation is also based on contextual information or word/tag sequences. For example, the model might prefer noun analyses over verb analyses if the preceding word is a preposition or article. Disambiguation is the most difficult problem in tagging. The ambiguity which is identified in the tagging module is resolved using the grammar rules. Sometimes, the ambiguity of a word can get reduced when it appears in the context of other words.

E. Applications of POS Tagger
The POS tagger can be used as a pre-processor. Text indexing and retrieval uses POS information. Speech processing uses POS tags to decide the pronunciation. POS tagger is used for making tagged corpora.

III. POS TAGGING TECHNIQUES

A. Rule-based POS Tagging
Rule-based part-of-speech tagging is the oldest approach that uses hand-written rules for tagging. Rule based tagger depends on dictionary or lexicon to get possible tags for each word to be tagged. Hand-written rules are used to identify the correct tag when a word has more than one possible tag. Disambiguation is done by analyzing the linguistic features of the word, its preceding word, its following word and other aspects. For example, if the preceding word is article then the word in question must be a noun. This information is coded in the form of rules.

B. What is Markov Model?
Markov models extract linguistic knowledge automatically from the large corpora and do POS tagging. Markov models are alternatives for laborious and time-consuming manual tagging.
A Markov model is nothing but 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 Markov model is to find optimal sequence of tags T = {t1, t2, t3,…tn} for the word sequence W = {w1,w2,w3,…wn}. That is to find the most probable tag sequence for a word sequence.
If we assume the probability of a tag depends only on one previous tag, then the model developed is called bigram model. Each state in the bigram model corresponds to a POS tag. The probability of moving from one POS state to another can be represented as $P(t_i|t_j)$. The probability of word being emitted from a particular tag state can be represented as $P(w_i|t_j)$. Assume that the sentence, "The sun shines" is to be tagged. Obviously, the word, "The" is determiner, so can be annotated with tag, say Det, "sun" is noun so the tag can be N, and "shines" is a verb so the tag can be V. So we get the tagged sentence as
The|Det sun|N shines|V
Given this model, P(Det N V | The sun shines) is estimated as
P(Det | START) * P(N | Det) * P(V | N) * P(The | Det) * P(sun | N) * P(shiness | V)
This is how to derive probabilities required for the Markov model.

C. Viterbi Algorithm/ Hidden Markov Models (HMM) in POS tagging
The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models.
Hidden Markov Models (HMM) are so called because 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 more probable tag. For example, the word "help" will be tagged as a noun rather than verb if it comes after an article. This is because the probability of noun is much more than verb in this context.
In an HMM, we know only the probabilistic function of the state sequence. In the beginning of tagging process, some initial tag probabilities are assigned to the HMM. Then in each training cycle, this initial setting is refined using the Baum-Welch re-estimation algorithm.

D. Transformation-based Learning
1. What is Transformation-Based Learning?
Transformation-based learning (TBL) is a rule-based algorithm for automatic tagging of parts-of-speech to the given text. TBL transforms one state to another using transformation rules in order to find the suitable tag for each word. TBL allows us to have linguistic knowledge in a readable form. It extracts linguistic information automatically from corpora. The outcome of TBL is an ordered sequence of transformations of the form as shown below.
Tagi->Tagj in context C
A typical transformation-based learner has an initial state annotator, a set of transformations and an objective function.

2. Initial Annotator
It is a program to assign tags to each and every word in the given text. It may be one that assigns tags randomly or a Markov model tagger. Usually it assigns every word with its most likely tag as indicated in the training corpus. For example, "walk" would be initially labelled as a verb.

3. Transformations
The learner is given allowable transformation types. A tag may change from X to Y if the previous word is W, the previous tag is ti and the following tag is tj, or the tag two before is ti and the following word is W. Consider the following sentence, The sun shines. A typical TBL tagger (or Brill Tagger) can easily identify that "sun" is noun if it is given the rule, if the previous tag is an article and the following tag is a verb.

4. How transformation based learning works?
Transformation based learning (TBL) usually starts with some simple solution to the problem. Then it runs through cycles. At each cycle, the transformation which gives more benefit is chosen and applied to the problem. The algorithm stops when the selected transformations do not add more value or there are no more transformations to be selected. This is like painting a wall with background color first, then paint different color in each block as per its shape or so. TBL is best suitable for classification tasks.
In TBL, accuracy is generally considered as the objective function. So in each training cycle, the tagger finds the transformations that greatly reduce the errors in the training set. This transformation is then added to the transformation list and applied to the training corpus. At the end of the training, the tagger is run by first tagging the fresh text with initial-state annotator, then applying each transformation in order wherever it can apply.

5 . Advantages of Transformation Based Learning
- Small set of simple rules that are sufficient for tagging is learned.
- As the learned rules are easy to understand development and debugging are made easier.
- Interlacing of machine-learned and human-generated rules reduce the complexity in tagging.
- Transformation list can be compiled into finite-state machine resulting in a very fast tagger. A TBL tagger can be even ten times faster than the fastest Markov-model tagger.
- TBL is less rigid in what cues it uses to disambiguate a particular word. Still it can choose appropriate cues.

6. Disadvantages of Transformation Based Learning
- TBL does not provide tag probabilities.
- Training time is often intolerably long, especially on the large corpora which are very common in Natural Language Processing.

IV. PARSING

A. Introduction of Parsing
Parsing is another important aspect utilized in conjunction with part-of-speech tagging to identify and understand natural language sentences. With parsing, when given an input sentence and a grammar, it can be determined whether the grammar can generate the sentence. Parsing can be described, at least in this context, as "the process of analyzing a string of words to uncover its phrase structure, according to the rules of the grammar" [1, 3, 8, 11]. In other words, part-of-speech tagging can be viewed as a necessary subtask of parsing, as the tagging rules occur as part of the lexicon. The goal of parsing is to find all possible permutations that contain all words in the given input while abiding by the rules of the grammar to create a sentence; currently two main strategies exist to do so.
A top-down parsing strategy begins with the knowledge that the input is a sentence, then attempts to create all possible permutations that can be derived from this interpretation and check the results against the original input to find the proper formatting. A bottom-up parsing strategy starts with the input and applies all possible rules to attempt to generate the base property.

Parsing the sentence would convert the sentence into a tree whose leaves will hold POS tags (which correspond to words in the sentence), but the rest of the tree would tell you how exactly these words are joining together to make the overall sentence.

B. Earley's Parsing Algorithm
In computer science, the Earley parser is an algorithm for parsing strings that belong to a given context-free language, though (depending on the variant) it may suffer problems with certain nullable grammars. The algorithm, named after its inventor, Jay Earley, is a chart parser that uses dynamic programming; it is mainly used for parsing in computational linguistics. It was first introduced in his dissertation in 1968.
The task of the parser is essential to determine if and how the grammar of a pre-existing sentence can be determined. This can be done essentially in two ways, Top-down Parsing and Bottom- up Parsing.
Earley's algorithm is a top-down dynamic programming algorithm [11]. We use Earley's dot notation: given a production X → xy, the notation X → x • y represents a condition in which x has already been parsed and y is expected.
For every input position (which represents a position between tokens), the parser generates an ordered state set. Each state is a tuple (X → x • y, i), consisting of

- the production currently being matched (X → x y);
- our current position in that production (represented by the dot);
- the position i in the input at which the matching of this production began: the origin position

The state set at input position k is called S(k). The parser is seeded with S(0), consisting of only the top-level rule. The parser then iteratively operates in three stages: prediction, scanning, and completion.

- Prediction: For every state in S(k) of the form (X → x • Y y, j) (where j is the origin position as above), add (Y → • z, k) to S(k) for every production in the grammar with Y on the left-hand side (Y → z).

- Scanning: If a is the next symbol in the input stream, for every state in S(k) of the form (X → x • a y, j), add (X → x a • y, j) to S(k+1).
- Completion: For every state in S(k) of the form (X → z •, j), find states in S(j) of the form (Y → x • X y, i) and add (Y → x X • y, i) to S(k).

V. LITERATURE STUDY

Navanath Saharia et al. in 2009 [5] have used HMM and the Viterbi Algorithm in the Assamese text corpus (Corpus Asm) of nearly 3,00,000 words from the online version of the Assamese daily News paper “Asomiya Pratidin” where nearly 10,000 words of this corpus were manually tagged by them for training. The tagset used by them have 172 tags which was larger in size with compared to the other Indian languages’ tagsets. They have obtained an average tagging accuracy of 87%. According to their report, the HMM based experiments on various Indian languages, they have obtained the best accuracy level so far. Moreover, for the improvement of the system’s accuracy, they have proposed some additional works like

- the size of the manually tagged part of the corpus will have to be increased.
- a suitable procedure for handling unknown proper nouns will have to be developed.
- If this system can be expanded to trigrams or even n-grams using a larger training corpus.

Rahman, Mirzanur and et al. in 2009 [4, 7] have developed a context free grammar (CFG) for simple Assamese sentences. In this work they have considered only limited number of sentences for developing rules and only seven main tags are used. They have analyzed the issues that arise in parsing Assamese sentences and produce an algorithm to solve those issues. They produced a technique to check that grammatical structure of the sentences in Assamese text and made grammar rules by analyzing the structure of Assamese sentences. Their Parsing program can find the grammatical error, if any, in the Assamese sentences. If there is no error, their program can generate the parse tree for the input Assamese sentence. Their algorithm is a modification of Earley’s Parsing Algorithm and they found the algorithm simple and efficient but the accuracy rate is not mentioned.

Navanath Saharia et al. in 2011 [7, 9] described a parsing criterion for Assamese text. They have discussed some salient features of Assamese syntax and the issues that simple syntactic frameworks can not tackle. They have also described the practical analysis of Assamese sentences from a computational perspective. This approach can be used to parse the simple sentences with multiple noun, adjective, adverb clause. They have defined a context free grammar (CFG) to parse simple Assamese sentences like “মই কিতাপ পঢ়িলোঁ” that is any type of simple sentences where object is prior to verb. But the main drawback of this approach is that it can also generate a parse tree for a sentence which is semantically wrong. Again they have also found that if the noun is attached with any type of suffix, then the defined CFG can easily generate synatically and semantically correct parse tree. Also to generate parse tree for the sentences which can not be obtained using their CFG, they have applied Chu-Liu-Edmond’s maximum spanning tree algorithms. They have achieved an accuracy of 78.82% in this particular parsing approach.

TABLE I Literature Survey

Sl. No	**Paper name (Year)**	**Publication details and Author name**	**Language**	**Method/Algorithm /Tool**	**Accuracy**	**Corpus /Dataset**
1	Part of Speech tagger for Assamese Text (2009)	In Proceedings of the ACL IJCNLP 2009 Conference, Short Papers, Suntec, Singapore, Pp. 33-36(2009) (Navanath Saharia et al.)	Assamese	Hidden Markov Model/Viterbi Approach	86.89%	Ten thousand Assamese words (10,000)
2	Parsing of part-of-speech tagged Assamese Texts (2009)	IJCSI International Journal of Computer Science Issues, Vol. 6, No. 1, 2009 (Rahman, Mirzanur et al.)	Assamese	Earley’s Parsing Algorithm	Earley’s algorithm is simple and effective	Assamese sentences
3	A First Step Towards Parsing of Assamese Text (2011)	Special Volume: Problems of Parsing in Indian Languages (Navanath Saharia et al.)	Assamese	Rule Based	78.82%	ICON 2009 datasets

VI. CONCLUSION AND FUTURE WORK

Here in this paper, we have presented a brief study on the different POS tagging approaches on Assamese language with their performances. We also discussed briefly some of the existing approaches used to develop parsers for Assamese language. We found in this study that all of the three (03) NLP approaches are efficient and satisfactory, but only for the simple Assamese sentences. So, in this regard much work has to be done to handle complex Assamese sentences with different structures. Because of relatively free word order characteristics and various ambiguous words, POS tagging of Assamese language is relatively tough work. The added difficulty in Assamese language POS tagging is of unavailibity of annotated corpora and predefined tagset which is beyond public access. Our future work is to create annotated corpora and an efficient Syntactic Analyzer by considering the agglutinative and morphological rich features of Assamese language to donate our bit of contribution to the resource poor Assamese language.

REFERENCES

[1] Joakim, Nivre (2009), Parsing Indian languages with maltparser, Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing : 12-18.
[2] Patil, H.B., Patil, A.S. Pawar, B.V.: Part-of-Speech Tagger for Marathi Language using Limited Training Corpora 2014 in International Journal of Computer Applications (0975 – 8887) Recent Advances in Information Technology.
[3] Bharati, Akshar, Gupta, Mridul, Yadav, Vineet, Gali, Karthik and Misra Sharma, Dipti (2009) : Simple parser for Indian languages in a dependency framework, Proceedings of the Third Linguistic Annotation Workshop. Association for Computational Linguistics.
[4] Rahman, Mirzanur, Das, Sufal and Sharma, Utpal (2009): Parsing of part-of-speech tagged Assamese Texts, IJCSI International Journal of Computer Science Issues, Vol. 6, No. 1.
[5] Saharia, Navanath., Das, Dhrubajyoti ,Sharma, Utpal., Kalita, Jugal.: Part of Speech Tagger for Assamese Text: In Proceedings of the ACL IJCNLP 2009 Conference Short Papers,Suntec, Singapore, Pp. 33-36 (2009).
[6] Rathod, Shubhangi, Govilkar, Sharvari (2015), Survey of various POS tagging techniques for Indian regional languages, International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2525-2529
[7] Makwana, Monika T., Vegda ,Deepak C.(2015), Survey: Natural Languages Parsing for Indian Languages, Computer Science, Computation and Language, Cornell University Library.
[8] Chatterji, Sanjay, Sonare, Praveen, Sarkar, Sudheshna and Roy, Debashree (2009), Grammar Driven Rules for Hybrid Bengali Dependency Parsing, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009
[9] Saharia, Navanath ,Sharma, Utpal, and Kalita, Jugal (2011) A First Step Towards Parsing of Assamese Text, Special Volume: Problems of Parsing in Indian Languages
[10] http://language.worldofcomputing.net/pos-tagging/parts-of-speech-tagging.html
[11] Pandey, Rakesh, Pande, Nihar Ranjan, Dhami, H. S. : Parsing of Kumauni Language Sentences after Modifying Earley's Algorithm Information Systems for Indian Languages, Volume 139 of the series Communications in Computer and Information Science pp 165-173