A Trigram HMM-Based POS Tagger for Indian Languages

Kamal Sarkar[*] and Vivekananda Gayen

Computer Science & Engineering Department,
Jadavpur University,
Kolkata – 700 032, India
jukamal2001@yahoo.com, Vivek3gayen@gmail.com

Abstract. We present in this paper a trigram HMM-based (Hidden Markov Model) part-of-speech (POS) tagger for Indian languages, which will accept a raw text in an Indian language (typed in corresponding language font) to produce a POS tagged output. We implement the trigram POS Tagger from the scratch based on the second order Hidden Markov Model (HMM). For handling unknown words, we introduce a prefix analysis method and a word-type analysis method which are combined with the well known suffix analysis method for predicting the probable tags. Though our developed systems have been tested on the data for four Indian languages namely Bengali, Hindi, Marathi and Telugu, the developed system can be easily ported to a new language just by replacing the training file with the POS tagged data for the new language. Our developed trigram POS tagger has been compared to the bigram POS tagger defined as a baseline.

Keywords: Part-of-speech tagging, Second order Hidden Markov Model, Deleted interpolation, Indian Languages.

1 Introduction

Part-of-Speech (POS) tagging is the task of assigning grammatical categories (noun, verb, adjective etc.) to words in a natural language sentence [1]. POS tagging can be used in parsing, word sense disambiguation, information extraction, machine translation, question answering, chunking etc.

Since the most previous POS taggers [2] [3] [4] are experimented on datasets which are not publicly available, it poses various difficulties such as comparisons of the present works to the past works. So, our primary motivations are (1) to develop a POS tagger for Indian Languages, which can accept a raw text in one of Indian languages and (2) to report results after testing our developed POS tagger on a publicly available dataset named NLTK (Natural Language Toolkit) dataset to allow the researchers to easily compare their systems with our developed systems. The salient features of our developed POS tagger are as follows:

[*] Corresponding author.

- Our developed POS tagger uses prefix analysis, word-type analysis and suffix analysis methods for predicting the probable tags of an unknown word (the word which is not present in the training dataset).
- Our developed POS tagger can be used for POS tagging for multiple Indian Languages. Since our system has been developed on the Visual Basic platform, it has a good user interface to submit the file to be tagged and to get the tagged output in another file that can be directly redirected to other NLP applications.

A substantial amount of research work has already done in POS tagger developments for Indian languages [5]. The various supervised POS tagging methods for Bengali have been presented in [2][3] [4 [6 [7] [8]. The descriptions of Hindi POS taggers have been presented in [9][10]. A rule based approach to morphological analysis and POS tagging in Tamil Language via Projection and Induction Techniques has been presented in [11]. A SVM Based part-of-Speech tagger for Malayalam has been presented in [12]. The hybrid POS tagger for three Indian languages namely Hindi, Bengali and Telugu presented in [13] combines HMM based approach and rule based approach.

A TnT tagger version of a POS Tagger for three Indian Languages namely Hindi, Bengali and Telugu has been presented in [14]. The work presented in [14] uses only suffix analysis for handling unknown words.

Section 2 presents the background on HMM based POS tagging. The evaluation and results are presented in section 3.

2 HMM Based POS Tagging

A POS tagger based on Hidden Markov Model (HMM) [15] assigns the best sequence of tags to an entire sentence. Generally, the most probable tag sequence is assigned to each sentence following the Viterbi algorithm [16]. The task of Part of Speech (POS) tagging is to find the sequence of POS tags t_1^n that is optimal for a given word sequence, w_1^n. The tagging problem becomes equivalent to searching for $\operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n) P(t_1^n)$ (by the application of Bayes' law), that is, we need to compute:

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n) P(t_1^n) \tag{1}$$

Where: where t_1^n is a tag sequence and w_1^n is a word sequence, $P(t_1^n)$ is the prior probability of the tag sequence and $P(w_1^n|t_1^n)$ is the likelihood of the word sequence. Equation (1) is too hard to compute directly. HMM taggers make Markov assumption which states that the probability of a tag is dependent only on a small, fixed number of previous tags. A bigram tagger considers that the probability of a tag depends only on the previous tag. For our proposed trigram model, the probability of a tag depends on two previous tags and $P(t_1^n)$ is computed as:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}, t_{i-2}) \tag{2}$$

Depending on the assumption that the probability of a word appearing is dependent only on its own part-of-speech tag, $P(w_1^n|t_1^n)$ can be simplified to:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i\,|t_i) \qquad (3)$$

Plugging the above mentioned two equations (2) and (3) into (1) results in the following equation by which a bigram tagger estimates the most probable tag sequence:

$$\hat{t}_1^n = \mathrm{argmax}_{t_1^n}\;\; P(t_1^n\,|w_1^n) \approx \mathrm{argmax}_{t_1^n}\;\prod_{i=1}^{n} P(w_i\,|t_i)P(t_i|t_{i-1}) \qquad (4)$$

Where: the tag transition probabilities,$P(t_i|t_{i-1})$, represent the probability of a tag given the previous tag. The word likelihood probabilities, $P(w_i|t_i)$, represent the probability of a word given a tag.

Considering a special tag t_{n+1} to indicate the end sentence boundary and two special tags t_{-1} and t_0 at the starting boundary of the sentence and adding these three special tags to the tag set [1], gives the following equation for part of speech tagging:

$$\hat{t}_1^n = \mathrm{argmax}_{t_1^n} P(t_1^n\,|w_1^n) \approx \qquad (5)$$
$$\mathrm{argmax}_{\hat{t}_1^n}[\prod_{i=1}^{n} P(w_i\,|t_i)P(t_i|t_{i-1},t_{i-2})]P(t_{n+1}\,|t_n)$$

The equation (5) is still computationally expensive because we need to consider all possible tag sequence of length *n.* So, dynamic programming approach is used to compute the equation (5).

At the training phase of HMM based POS tagging, observation probability matrix and tag transition probability matrix are created.

2.1 Computing Tag Transition Probabilities

As we can see from the equation (4) to find the most likely tag sequence for a sentence (considered as an observation sequence), we need to compute two kinds of probabilities: tag transition probabilities and word likelihoods or observation probabilities.

Our developed trigram HMM tagger requires to compute tag trigram probability, P$(t_i|t_{i-1},t_{i-2})$, which is computed by the maximum likelihood estimate from tag trigram counts. To overcome the data sparseness problem, tag trigram probability is smoothed based on the bigram and unigram probabilities using the following equation:

$$P(t_i|t_{i-1},t_{i-2}) = \lambda_1\hat{P}((t_i|t_{i-1},t_{i-2}) + \lambda_2\hat{P}((t_i|t_{i-1}) + \lambda_3\hat{P}(t_i) \qquad (6)$$

$\hat{P}((t_i|t_{i-1},t_{i-2})$, $\hat{P}((t_i|t_{i-1})\; and \hat{P}(t_i)$ are the maximum likelihood estimates from counts for tag trigram, tag bigram and tag unigram respectively:

$$\hat{P}((t_i|t_{i-1},t_{i-2}) = \frac{C(t_{i-2},t_{i-1},t_i)}{C(t_{i-2},t_{i-1})}, \qquad \hat{P}((t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}, \qquad \hat{P}(t_i) = \frac{C(t_i)}{N}$$

Where: $C(t_{i-2},t_{i-1},t_i)$ indicates the count of the tag sequence $<t_{i-2},\, t_{i-1},\, t_i>$ and $\lambda_1,\, \lambda_2,\, \lambda_3$ ($\lambda_1 + \lambda_2 + \lambda_3 = 1$) are the weights for the maximum likelihood estimates of trigram, bigram and unigram tag probabilities respectively computed based on corpus statistics. The values of the parameters: $\lambda_1, \lambda_2,\, \lambda_3$ are estimated using a smoothing technique called the deleted interpolation proposed in [1].

2.2 Computing Observation Probabilities

The observation probability of a word is computed using the following equation:

$$P(w|t) = \frac{C(w,t)}{C(t)} \quad (7)$$

Using a equation (7), a observation probability matrix is created where each row is labeled with a word and each column is labeled with a tag. Each cell of the matrix contains the probability of a word given a tag. The observation probability of a word *w* given tag *t* may be zero when the pair <*w, t*> is not present in the training corpus although the word may be present in the corpus along with some tag other than *t*. In this case, to avoid sparseness of data, maximum negative value (negative infinity) is assigned to the cell.

2.3 Viterbi Decoding

The task of a decoder is to find the best hidden state sequence given an input HMM and a sequence of observations.

The Viterbi algorithm is the most common decoding algorithm used for HMM based part-of-speech tagging. This is a standard application of the classic dynamic programming algorithm [15]. The Viterbi algorithm that we use, takes as input a single HMM and a set of observed words $O = (o_1 o_2 o_3 \ldots o_t)$ and returns the most probable state sequence, $Q = (q_1 q_2 q_3 \ldots q_t)$, together with its probability.

Given a tag transition probability matrix and the observation probability matrix, Viterbi decoding (used at the testing phase) accepts an untagged text document in Indian language and finds the most likely tag sequence for each sentence in the input document.

We have used the Viterbi algorithm presented in [16] for finding the most likely tag sequence for a given sentence. One of the important problems to apply Viterbi decoding algorithm is how to handle unknown words in the input test sentence. The unknown words are the words which are not present in the training set and hence their observation probabilities are not known. To handle this problem, we estimate the observation probability of an unknown word by analyzing prefix, word-type and suffix of the unknown word.

2.4 Handling Unknown Words

For unknown words, observation probabilities are not available in the observation probability matrix which is created using the equation (7). We estimate the observation probability of an unknown word in the following ways:

Prefix analysis is done first for estimating the observation probability of an unknown word, if the prefix analysis fails, word-type analysis is done and when both fail, suffix analysis is done.

Prefix Analysis. The observation probability of an unknown word is estimated based on its prefix analysis. This is based on the hypothesis that a slightly modified form of an unknown word may be present in the training data whereas the unknown word itself is absent in the training data, for example, when X is the plural form of a common noun and Y is the singular form of the same common noun, X may be present in the training data, but Y may not be present in the training data. For such cases, we may estimate observation probability of an unknown word by matching carefully the unknown word with the closest known words (whose observation probability is known). To do so, an unknown word U is aligned from left to right with a known word K and if the match between U and K exceeds some threshold value, the observation probability of K is taken as that of U. For predicting the probable tags for an unknown word, some rules are framed based on heuristics that use prefix information constrained by the length of the unknown word. The minimum allowable length L_{min} of U is set to 4(in terms of number of characters), that is, an unknown word whose length L_u is less than L_{min} would not be assigned the observation probability through prefix analysis because the prefix of the word is too short to be confident about it. Where the length of the unknown word is greater than or equal to L_{min}, the allowable prefix mismatch between U and K is varied. When the length of U is relatively shorter, the maximum prefix mismatch between U and K that we allow is set to only one character (relatively tough matching). But, when U is relatively long, the criteria of prefix matching is little bit relaxed. However, we have devised three conditions for setting the observation probability of U based on its length and prefix matching between U and K, that is, the observation probability of U is set to that of K when any one of the following conditions holds:

1. $L_{min} \leq L_u \leq L_{mid1}$ and if U and K differ only in the last character
2. $L_{mid1} < L_u \leq L_{mid2}$ and if U and K differ in the last one or two characters
3. $L_u > L_{mid2}$ and if U and K differ in the last one or two or three characters

For the best results, we set $L_{min} = 4$, $L_{mid1} = 8$, $L_{mid2} = 12$.

Word-Type Analysis. The word-types identified by the surface level information of some unknown words help to narrow down the choices of the probable tags for those words. For example, the 4 digit number, hyphenated word etc. We have identified a list of word-types (instead of individual words) based on surface level features by which we can predict the nature of the word and search the table which contains precomputed information such as the word type, possible tags along with the observation probabilities(probability of a word-type given a tag). The table contains the probability, P (w-type| t_i) where w-type is a word type and t is a tag. P(w-type| t_i) is used in place of $P(w_i|t_i)$ which is required for the HMM based tagging(as specified in the equation (4)). Data sparseness problem is handled in the similar way presented in section 2 in the context of computing observation probabilities of individual words. We have identified the various types of words: four digit number, number started with a digit and ended with an alphabet, hyphenated word, the word fully numeric but the number of digits is not four.

The word-type analysis method functions in the following way:

For an unknown word, its word type is determined by analyzing its surface level features and if its word type matches with any of the word-type pre-computed and stored in the table, the observation probability retrieved from the table is set to the observation probability of the unknown word.

Suffix Analysis. The observation probabilities of unknown words are decided according to the suffix of a word. We find the observation probabilities of unknown words using suffix analysis of all rare words (frequency <=3) in the corpus since unknown words are infrequent and using suffixes of infrequent words in the lexicon is a better approximation for unknown words [1]. The term suffix as used in this context means "a sequence of characters occurring at the end of a word" which is not necessarily a linguistically meaningful suffix. The maximum length of suffix is set to 10 for which we get the best results on our training corpus. The probability of a tag given a suffix of length i is computed as: P(t |suffix-of-len(i)). These probabilities are smoothed using successively shorter and shorter suffixes [1]. This can be formulated in recursive way as:

$$P(t|\text{ suffix-of-len(i)}) = \frac{\hat{P}(t|\text{ suffix−of−len(i)}) + \theta_i\, P(t|\text{ suffix−of−len (i−1)})}{1+\theta_i} \quad . (8)$$

Where: $\hat{p}$ is the maximum likelihood probability based on the count of <tag, suffix> pair in all rare words (frequency <=3) in the corpus.

All θ_i are set to the standard deviation of the unconditioned maximum likelihood probabilities ($\hat{p}(t_i)$) of the tags in the training corpus [1]. P(t| suffix-of-len(i)) gives an estimate of $P(t_i|w_i)$. But for HMM based tagging we need to compute the likelihood $P(w_i|t_i)$ which is computed from $P(t_i|w_i)$ using Bayesian inversion that uses Bayes rule and prior $P(t_i)$.

3 Evaluation and Results

3.1 Evaluation

Accuracy of tagging is computed as the ratio of number of matched tags to the total number of tags with duplicates:

$$\text{Accuracy}=\frac{number\ of\ matched\ tags}{total\ number\ of\ tags\ in\ the\ testing\ corpus} \times 100\%$$

3.2 Results

We test our developed POS tagger on NLTK (Natural Language Toolkit) dataset[1] which are freely available for download. The NLTK POS tagged corpora for Indian Languages, downloaded from NLTK website consists of POS tagged text in four different languages namely Bengali, Hindi, Marathi and Telugu. The POS tagged data for Bengali contains a total of 895 Bengali (typed in Unicode) sentences tagged using

[1] `http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml`

26 POS tags. Similarly, the POS tagged data for Hindi contains a total of 539 sentences tagged using 25 tags, the POS tagged data for Marathi contains 1196 sentences tagged using 27 tags and the POS tagged data for Telugu contains 994 sentences tagged using 24 tags.

For system evaluation, 10-fold cross validation using NLTK data set is done and the results obtained after running the system on 10 different folds are averaged to find the final results for the system.

We consider bigram tagger as the baseline tagger to which our trigram tagger is compared. In bigram tagger, when transition probabilities are computed, only one previous tag is considered whereas the trigram tagger considers two previous tags while computing the transition probabilities.

To judge the effectiveness of our introduced new features, prefix analysis and word-type analysis for handling unknown words, two versions of a trigram tagger are developed: version 1 is a trigram tagger that considers only suffix analysis for handling unknown words [17] and version 2 is a trigram tagger that considers prefix, word-type and suffix analysis for handing unknown words. For each language, the performances of these two versions are also compared. Table 1 shows the performances of our developed POS tagger for four Indian Languages namely Bengali, Hindi, Marathi and Telugu.

Table 1. Performance comparisons of the POS tagger for Bengali, Hindi, Marathi and Telugu

	Bengali	Hindi	Marathi	Telugu
Systems	*Accuracy (%)*	*Accuracy (%)*	*Accuracy (%)*	*Accuracy (%)*
Trigram tagger version 2	79.65	84.80	84.00	80.00
Trigram tagger version 1	78.68	83.79	83.16	79.00
Bigram tagger	74.33	79.59	79.29	74.79

The table shows that the trigram POS tagger performs better than the bigram POS tagger for all four languages. It is also evident from the table that, for each of four languages considered in our experiments, the trigram tagger that uses prefix, word-type and suffix analysis for handing unknown words performs better than the trigram tagger that uses only suffix analysis for handing unknown words.

4 Conclusion

This paper describes a trigram HMM based POS tagger for Indian Languages namely Bengali, Hindi, Marathi and Telugu. The POS tagger has been developed using Visual Basic platform so that a suitable user interface can be designed for accepting the input in Unicode from the novice users. The system has been designed in such a way that only changing the training corpus in a file can make the system portable to a new Indian language.

References

1. Brants, T.: TnT – “A statistical part-of-speech tagger”. In: Proc. of the 6th Applied NLP Conference, pp. 224–231 (2000)
2. Dandapat, S., Sarkar, S., Basu, A.: Automatic part-of-speech tagging for bengali: an approach for morphologically rich languages in a poor scenario. In: Proceedings of the Association for Computational Linguistic, pp. 221–224 (2007)
3. Ekbal, A., et al.: Bengali part of speech tagging using conditional random field. In: Proceedings of the 7th International Symposium of Natural Language Processing (SNLP 2007), Pattaya, Thailand, December 13-15, pp. 131–136 (2007)
4. Ekbal, A., Bandyopadhyay, S.: Part of speech tagging in bengali using support vector machine. In: IEEE International Conference on Information Technology, ICIT 2008, pp. 106–111 (2008)
5. Kumar, D., Josan, G.S.: Part of speech taggers for morphologically rich indian languages: a survey. International Journal of Computer Applications (0975-8887) 6(5) (2010)
6. Ali, H.: An unsupervised parts-of-speech tagger for the bangla language, Department of Computer Science, University of British Columbia (2010)
7. Chakrabarti, D.: Layered parts of speech tagging for bangla, Language in Indian. Special Volume: Problems of Parsing in Indian Languages (May 2001), `http://www.languageinindia.com`
8. Antony, P.J., Soman, K.P.: Parts of speech tagging for Indian languages: a literature survey. International Journal of Computer Applications (0975-8887) 34(8) (November 2011)
9. Shrivastava, M., Bhattacharyya, P.: Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay. In: Proceeding of the ICON (2008)
10. Ray, P.R., Harish, V., Sarkar, S., Basu, A.: Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi, Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur, INDIA 721302, `http://www.mla.iitkgp.ernet.in/papers/hindipostagging.pdf`
11. Selvam, M., Natarajan, A.M.: Improvement of Rule Based Morphological Analysis and POS Tagging in Tamil Language via Projection and Induction Techniques. International Journal of Computers 3(4) (2009)
12. Antony, P.J., Santhanu, P.M., Soman, K.P.: SVM Based Parts Speech Tagger for Malayalam. In: International Conference on-Recent Trends in Information, Telecommunication and Computing, ITC 2010 (2010)
13. Pattabhi, R.K.R.T., Vijay Sundar Ram, R., Vijayakrishna, R., Sobha, L.: A Text Chunker and Hybrid POS Tagger for Indian Languages, AU-KBC Research Centre. MIT Campus, Anna University, Chromepet, Chennai (2007)
14. Rao, D., Yarowsky, D.: Part of Speech Tagging and Shallow Parsing of Indian Languages, Department of Computer Science, Johns Hopkins University, USA, The Proceedings of the Workshop on Shallow Parsing in South Asian Languages (2007), `http://shiva.iiit.ac.in/SPSAL2007/final/iitmcsa.pdf`
15. Jurafsky, D., Martin, J.H.: Speech and Language Processing An Intoduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Preason Education Series (2002)
16. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transaction on Information Theory IT-13(2), 260–269 (1967)
17. Sarkar, K., Gayen, V.: A Practical Part-of-Speech Tagger for Bengali. In: Third International Conference on Emerging Applications of Information Technology (EAIT 2012) (accepted, 2012)