Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 604–611

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Intelligent Part of Speech tagger for Hindi

Devashish Dutta[a], Subhanu Halder[b], Tirthankar Gayen[a*]

[a]*Jawaharlal Nehru University, New Delhi -110067, India*
[b]*IIT Delhi, New Delhi - 110016, India*

Abstract

English Part of Speech like noun, verb, adverb, adjective, pronoun, preposition, interjection, conjunction is somewhat similar in Hindi but not exactly the same. Hindi grammar has different Part of Speech (POS) based on its morphological features and the occurrence of a word/lexeme in a sentence. The existing techniques used in English language for POS tagging may not work properly for Indian language like Hindi. It is because the grammatical structure of the relatively free word order language like Hindi differs from English. Stochastic taggers may not give good performance as morphological information is not taken into account. The available Hindi word corpora usually have less frequency for individual tags. As a result, a larger size corpus having diversity in the type of sentences can provide better results. But, even after using smoothing techniques most these taggers fail to provide correct results in the presence of unknown words. Considering these aspects, this paper proposes an Intelligent POS tagger for Hindi language based on VITERBI and K-Nearest Neighbour, capable of providing more accurate results than VITERBI in the presence of unknown words.

Keywords: Part of Speech; Hindi; Emission probabilities; Transition probabilities; K-Nearest Neighbour

1. Introduction

In Part of Speech (POS) tagging, a word in a sentence is assigned a POS based on its definition and context. A fundamental stage in analyzing sentences in a language is POS tagging. It can be considered as one of the most researched issues in the domain of Natural Language Processing (NLP). Computer science and linguistics work together in this interdisciplinary field of NLP to accomplish a variety of tasks like Machine Translation, Text Summarization, Text Extraction, Text Classification, Sentiment Analysis, Speech Recognition, etc. Since POS tags are usually needed for analyzing sentences, they are extensively used in many NLP applications. In NLP, they are usually applied to punctuations as well as words. For punctuations it assumes tokenization of quotation marks, commas, etc. and disambiguation of periods inside words (for e.g., et al., etc.) from end-of-sentence periods. A tagging

10.1016/j.procs.2023.01.042

algorithm receives a series of tokenized words with a tagset as input and provides a sequence of tags (usually one tag for each token) as output. Words are usually ambiguous, hence a word may have more than one possible Part-of-Speech. India is a country where almost every state has its own language. Each of these languages has specific characteristics. Due to the non-availability of large annotated corpora, there hasn't been much study done in the various Indian languages. The morphologically rich languages of India are producing a common tagset framework for POS tagging is very difficult. Due to different structure of sentences and grammatical rules, designing a common full tagset for all Indian languages is also very difficult. So, language-specific tags may need to be incorporated in the common tagset. Kumar and Josan [1] have described different approaches used for languages like Malayalam, Bengali Hindi, Punjabi and Telugu. POS taggers created for Bengali, Hindi, Malayalam, Assamese, Telugu, Manipuri, Tamil, and Gujarati have been surveyed by Shambhavi and Ramakanth [2]. Similarly, Antony and Soman [3] have also given a detailed literature survey of various POS taggers created for Bengali, Punjabi, Hindi, Tamil, Telugu and Kannada. Baskaran et al. [4] aimed to create a POS tagset architecture that was universal to Indian languages. Due to different structure of sentences and grammatical rules, designing a common full tagset for all Indian languages is difficult and their efforts have not received wide acceptance. But using their set, researchers have added languages specific tag for languages they were analyzing.

Hindi is commonly spoken and written language in various parts of Northern India. English POS is somewhat similar in Hindi but not exactly the same. Hindi grammar has different POS based on its morphological features and the occurrence of a word in a sentence. The existing techniques used in the English language for POS tagging may not work properly for Indian Languages like Hindi. It is because the grammatical structure of the relatively free word order language like Hindi differs from English. Stochastic taggers may not give a good performance as morphological information is not taken into account. The available Hindi word corpora usually have less frequency for individual tags. As a result, a larger size corpus having diversity in the type of sentences can provide better results. Consequently, the creation of an appropriate POS tagger for Hindi is required. But, even after using smoothing techniques most these taggers fail to provide correct results in the presence of unknown words. Since, Hindi is widely spoken in various parts of India, this article is focused on improving the accuracy of POS tagging for the Indian language Hindi in order to obtain more accurate results in various applications like Sentiment Analysis, Machine Translation using Hindi, Word Sense Disambiguation, Named Entity Recognition, etc.

2. Related work

Most of the research work concerned with POS tagging for Hindi is done in India. But, due to the non-availability of sufficiently large corpus, initial attempts were made to develop POS tagger using morphological rules of Hindi. Shrivastava et al. [5] used Stemmer, a morphological analyzer for Hindi prior to developing a rule-based POS tagger. Based on linguistic knowledge, a dictionary was built up with root words and affixes. The whole process required a lot of human effort, time and a huge data set for the language. Ray et al. [6] described a POS tagger based on lexical sequence constraints in Hindi using morphological analyzer rules. The contest on POS tagging and chunking, NLPAI ML and SPSAL workshop were started by IIIT Hyderabad in 2006 and 2007 respectively [7]. Numerous POS tagging techniques for Indian languages have been developed throughout these two contests. Shrivastava and Bhattacharyya [8] developed a Hidden Markov Model (HMM)-based technique for Hindi and created a straightforward POS tagger for Hindi using HMM. With the use of Maximum Entropy Markov Model (MEMM) along with the lexical and morphological characteristics of Hindi languages, Dalal et al. [9] created a POS tagger for Hindi and attained 85.59% per word accuracy. This method results in 87.04% accuracy when tested with the IIIT Hyderabad corpus. The Conditional Random Field (CRF) approach was used by Himanshu Agrawal [10] to achieve an accuracy of 79.13%. Awasthi et al. [7] also used CRF in NLPAI, IIIT Hyderabad. Avinesh and Karthik [11] have described the combination of CRF and Transformation Based Learning (TBL) method for Bengali, Hindi and Telugu with an accuracy of 77.37%, 78.66% and 76.08% respectively. Rao and Yarowsky [12] showed that the accuracy of out-of-vocabulary words in the absence of morphological information can be improved by suffix-based POS tagging. They also showed that TnT tagger performs best for Hindi, Bengali and Telugu. Varma et al. [13] have developed a language independent POS tagger based on HMM and tested it for English and Hindi languages. Dandapat et al. [14] have designed and developed a genetic annotation tool for POS tagging using Cyclic Dependency Network as an initial annotator and described a case for Bangla and Hindi. Hindi corpus consists of 35 million words, the largest among all the Indian languages [11].

The Indian Languages Corpora Initiative (ILCI) consortium has been building parallel annotated corpora of 600,000 sentences (50,000 in each language) for 12 Indian languages which includes English in the health and tourism domain with Hindi as the source language [15 - 17].

The survey reveals that there hasn't been much work done on Hindi language PoS tagging. The exiting work for Hindi suffers from inaccuracy. Also it is found that there is lack of suitable dataset (corpus) having all possible diversities in the types of words/sentences. The existing techniques used for POS tagging of English language sentences may not work properly for the Indian language Hindi. It is because the grammatical structure of the relatively free word order language like Hindi differs from English. Stochastic taggers may not give good performance as morphological information is not taken into account. The available Hindi word corpora usually have less frequency for individual tags. As a result a larger size corpus having diversity in the type of sentences can provide better results. But, even after applying the smoothing techniques most these taggers fail to provide correct results in the presence of unknown words. Considering these aspects, the objective is to develop a suitable PoS tagger for Hindi language capable of providing more accurate results in the presence of unknown words.

3. Proposed approach

Hindi grammar has POS based on its morphological features and the occurrence of a word in a sentence. The existing techniques used in English language for POS tagging may not work properly for the Indian Language Hindi. It is because the grammatical structure of the relatively free word order language like Hindi differs from English. Stochastic taggers may not give good performance as morphological information is not taken into account. The available Hindi word corpora usually have less frequency for individual tags. As a result a larger size corpus having diversity in the type of sentences can provide better results. But, even after using smoothing techniques most of these taggers fail to provide correct results in the presence of unknown words. Consider various Hindi phases and their English equivalents (shown in Table 1).

Table 1. Some Hindi phrases with their English equivalents

Hindi text	English equivalent
यह एक फेंटाबुलोस बंगला है	This is a fantabulous bunglow
यह एक भूत बंगला है	This is a ghost bunglow
यह एक बड़ा बंगला है	This is a big bunglow
यह एक छोटा बंगला है	This is a small bunglow
यह भाषा बंगला है	This language is Bangla

In Table 1, words like फेंटाबुलोस (fantabulous) may not be found in the tagged Hindi corpora. Thus, considering these aspects, a suitable POS tagger for Hindi language capable of providing more accurate results in the presence of unknown words is developed. In order to find the POS tag for the unknown word Intelligent POS Tagger (IPT) algorithm is proposed. This algorithm is based on K-Nearest Neighbour (KNN) and HMM VITERBI [18] algorithms. The proposed algorithm IPT uses both Laplace smoothing and KNN to find the emission probabilities of the unknown word. Following are the details of the algorithm IPT.

Algorithm Intelligent POS Tagger (IPT)

Require: Tagged Dataset, D_t
Set of all tags, S_T
Test Dataset, D_{test}
Vocabulary, V

1: **Initialize** D_t, S_T, D_{test}, V

2: for every tag $s \in S_T$

2.1: $q(s|u) \leftarrow \frac{c(u\ s)+\lambda}{c(u)+\lambda|S_T|}, \forall u \in S_T$ ///$|S_T|$ represents the total number of tags in S_T

// u, s represents tags

// c(z) represents the count value of the sequence z

2.2: **for** every word $x \in V$ // V represents vocabulary

2.2.1: $e(x|s) \leftarrow \frac{c(s \rightarrow x)+\lambda}{c(s)+\lambda|S_T|}$ // λ is basically a real value between 0 and 1 (discounting factor)

// s→x represents the word ***x*** emitted for the tag ***s***

2.3: **endfor**

3: **endfor**

4: call VITERBI($|D_{test}|$, $|S_T|$)

5: for every word x in D_{test}

5.1: **if** $e(x|s) = \frac{\lambda}{c(s)+\lambda|S_T|}$, $\forall s \in S_T$ then

5.1.1: euclid [ ] ← 99999999

5.1.2: find the previous word x_p and the next word x_n

5.1.3: find all sequences of words x_p x' x_n in the training dataset D_t and add { T_{xp}, $e(x_p|T_{xp})$, T_{xn}, $e(x_n|T_{xn})$, $T_{x'}$, $e(x'|T_{x'})$ } in the set KNN_t

5.1.4: **for** every entry ***i*** in the set KNN_t

5.1.4.1: **if** ($T_{ip} = T_p$) and ($T_{in} = T_n$) then

5.1.4.1.1: find the Euclidean distance $euclid[i] \leftarrow \sqrt[2]{((e(x_{tp}|T_{xtp}) - e(x_p|T_p))^2 + (e(x_{tn}|T_{xtn}) - e(x_n|T_n))^2}$

5.1.4.2: **endif**

5.1.5: **endfor**

5.1.6: Sort all the entries in KNN_t in non decreasing order of their Euclidean distance

5.1.7: From the first ***k*** entries of the sorted KNN_t find the particular tag value of $T_{x'}$ which is repeated maximum number of times and assign the record (entry) corresponding to that tag value with the minimum Euclidean distance to Z

5.1.8: From the record ***Z*** assign $T_{x'}$ and $e(x'|T_{x'})$ to T and e(x|T) respectively

5.1.9: Add e(x|T) to the emission probability list

5.2: **endif**

6: **endfor**

7: **return**

In this algorithm, the emission and transition probabilities using training dataset (Tagged Dataset, D_t) (using steps 2 to 3) are first found. To acquire the POS tags for each word in the D_{test} the VITERBI algorithm [18] is invoked in step 4. In the Test Dataset, D_{test}, the unknown word x is found (if the condition in step 5.1 is satisfied) and then for every non-consecutive unknown word (x) the previous word (x_p) and the next word (x_n) of the unknown word in the sequence is found. Then all sequences of words x_p x' x_n in the training dataset D_t are found and { T_{xp}, $e(x_p|T_{xp})$, T_{xn}, $e(x_n|T_{xn})$, $T_{x'}$, $e(x'|T_{x'})$ } is added in the set KNN_t (step 5.1.3). For every entry in the set KNN_t if the previous tag (T_{ip}) and the next tag (T_{in}) of an entry **i** matches with the previous tag (T_p) and next tag (T_n) of the unknown word ***x*** then the Euclidean distance (using step 5.1.4.1.1) is calculated and the result is stored in the array *euclid*. Further, according to their Euclidean distance, all of the items in KNN_t are arranged in a non-decreasing order. From the first ***k*** entries of the sorted KNN_t find the particular tag value of $T_{x'}$ which is repeated maximum number of times and assign the record (entry) corresponding to that tag value with the minimum Euclidean distance to Z (step 4.1.7).

Further, from the record $\boldsymbol{Z}$, the $T_{x'}$ and $e(x'|T_{x'})$ values are assigned to T and e(x|T) respectively and e(x|T) is added to the emission probability list. Steps 5 to 6 are repeated for every word in D_{test}. Once this is done, the VITERBI algorithm [18] can be used to obtain the POS tags for all the words in the D_{test}. Following are the details of the VITERBI algorithm.

Algorithm VITERBI (T, N)

Require: Observations length, **T**
State-graph length, **N**

// Steps related to VITERBI algorithm for finding the optimal sequence

Step 1: construct a matrix vtb[N,T] of path probabilities;
Step 2: **for** every state ***s*** from ***1*** to ***N***
Step 2.1: $vtb[s,1] \leftarrow \pi_s * b_s(o_1)$; // $\boldsymbol{\pi_s}$ represents the initial probability of getting into state ***s***
Step 2.2: backptr[s,1]←0 ;
Step 3: **endfor**
Step 4: **for** ***t*** = ***2*** to ***T*** // ***t*** represents a time step
Step 4.1: **for** ***s*** = ***1*** to ***N*** // ***s*** represents a state

Step 4.1.1: $vtb[s,t] \leftarrow \max_{s'=1}^{N} vtb[s', t-1] * a_{s',s} * b_s(o_t)$;
// $a_{x,y}$ represents the probability of the change of state from ***x*** to ***y***
// $b_s(o_t)$ represents the emission probability of the observation $\boldsymbol{o_t}$ at state ***s*** and time step **t**

Step 4.1.2: $backptr[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} vtb[s', t-1] * a_{s',s} * b_s(o_t)$;

Step 4.2: **endfor**
Step 5: **endfor**
Step 6: $bpptr \leftarrow \operatorname{argmax}_{s=1}^{N} vtb[s,T]$;
Step 7: $bpprob \leftarrow \max_{s=1}^{N} vtb[s,T]$; // assigning the best path probability
//Follow the *backptr* matrix from state *bpptr* to states further back in time by tracing the path.

This algorithm provides paths through various states in HMM state graph of length ***N*** where the observation sequence of length ***T*** is assigned the maximum likelihood.

4. Implementation results and discussion

The implementation was made on an Intel(R) Core(TM) i5-1035G1, 4 Core(s), 8 Logical Processor(s) CPU @ 1.00 GHz, 1190 MHz, Windows 10 system using Python 3.9. Initially, the proposed algorithm (IPT) and VITERBI were implemented using Python 3.9 and their accuracies were tested using the tagged/labelled dataset of Hindi sentences from Hindi (original) sections of the universal dependencies corpus [19, 20]. 30% of the dataset was utilized for testing, while the rest 70% is used for training. A snapshot of the used dataset viewed using Notepad editor is shown in Fig. 1.

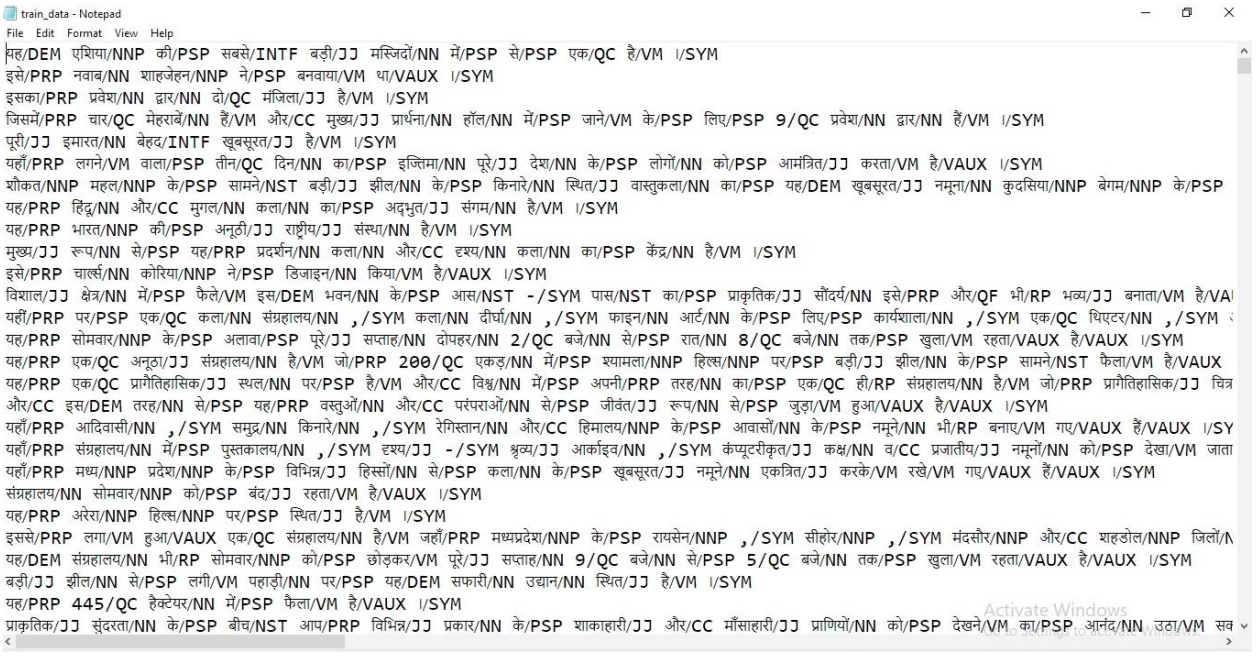

Fig. 1. A screenshot of the used dataset viewed using Notepad

Later on additional sentences from various sources [21] with words not present in the training dataset (unknown words) were incorporated manually in the test dataset. These additional sentences were tagged by hand annotations. Using this technique 25 different tagged datasets were generated. These 25 datasets were tested using VITERBI and the proposed IPT. Using each of the 25 datasets the accuracy values were obtained for VITERBI and IPT. Fig. 2 shows the plots of the obtained accuracy values.

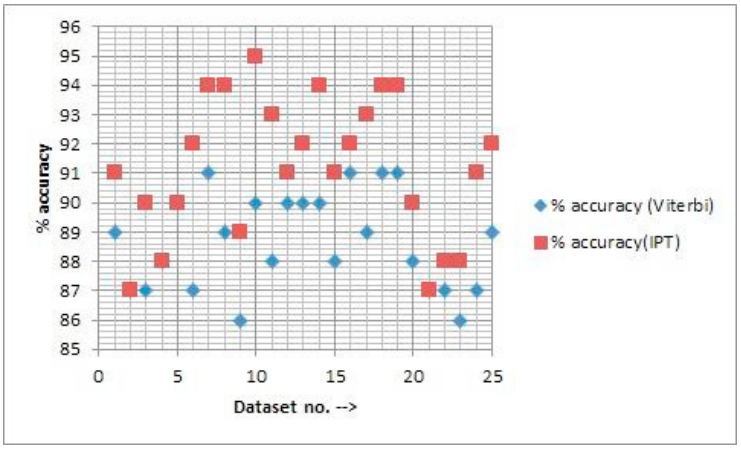

Fig. 2. Plots of the obtained accuracy values.

Using the accuracy values obtained from VITERBI and IPT a regression analysis was performed and the results obtained were summarized in Table 2.

Table 2. Regression results obtained from the accuracy values provided by VITERBI and IPT

Regression Results	
Standard Error	1.89832837
Multiple R	0.66511994
R Square	0.44238453
Adjusted R Square	0.41814038
Observations	25

ANOVA

	df	**SS**	**Significance F**
Regression	1	65.75603651	0.000286094
Residual	23	82.88396349	
Total	24	148.64	

	t Stat	**Coefficients**	**P-value**	**Standard Error**	**Upper 95%**	**Lower 95%**
Intercept	-0.04942	-1.05215124	0.961013	21.29070547	42.9910287	-45.09533114
% accuracy (VITERBI)	4.271658	1.03520209	0.000286	0.242341986	1.53652468	0.533879493

From Table 2, it is found that the regression coefficient r (Multiple R) = 0.67 and $p < 0.001$. This indicates that there is a positive relationship between the percentage accuracy obtained from VITERBI and the percentage accuracy obtained from the proposed IPT. Thus, the proposed IPT seems to provide better accuracy in the presence of unknown words as compared to VITERBI. Although a considerable amount of research and implementations were made using VITERBI for POS tagging of various languages like English, Gujarati [24], Indonesian [25], Hindi, Marathi [26], Malayalam [27], etc. the accuracy of the results become less with the use of unknown words in the test dataset. Thus, replacing the VITERBI algorithm used in these approaches with the suggested IPT can significantly improve the findings' accuracy.

5. Conclusion and scope for future work

POS tagging finds applications in various Natural Language Processing (NLP) application like text summarization [22], Word Sense Disambiguation (WSD), Sentiment Analysis, Named Entity Recognition NER), etc. [23]. Hindi is commonly spoken and written language in various parts of Northern India. Hindi grammar has POS based on its morphological features and occurrence of a word/lexeme in a sentence. Thus, the existing techniques used in English language for POS tagging may not work properly for Indian Languages like Hindi. It is because the grammatical structure of the relatively free word order language like Hindi differs from English. Stochastic taggers may not give good performance as morphological information is not taken into account. The available Hindi word corpora usually have less frequency for individual tags. As a result, a larger size corpus having diversity in the type of sentences can provide better results. But, even after using smoothing techniques most these taggers fail to provide correct results in the presence of unknown words. Considering these aspects, a suitable POS tagger for Hindi language is developed based on VITERBI and KNN capable of providing more accurate results than VITERBI in the presence of unknown words. Presently, the proposed IPT can handle only non-consecutive unknown words. Efforts can be made to improve IPT for handling more than one consecutive unknown word. Nonetheless, the proposed IPT can be considered as an improvement over VITERBI to provide more accurate results in the presence of unknown words.

References

[1] Kumar D and Josan G S (2010) "Part of Speech Taggers for Morphologically Rich Indian Languages: A Survey" *International Journal of Computer Application* **6(5)**: 1-9.

[2] Shambhavi B R. and Ramakanth P K (2010) "Current state of the Art POS tagging for Indian Languages – A Study" *International Journal of Computer Engineering and Technology*:250-260.

[3] Anthony P J and Soman K P (2011) "Part of Speech tagging for Indian Languages: a Literature Survey" *International Journal of Computer Applications* **34(8)**:22-29.

[4] Baskaran S, Bali K, Bhattacharya T, Bhattacharyya P, Jha G N, Rajendran S, Saravanan K, Sobha L and Subbarao K V. (2008) "Designing a common POS- Tagset Framework for Indian Languages" In Proceedings of the 6th Workshop on Asian Language Resources (ALR 6), Hyderabad.

[5] Shrivastava M, Agrawal N, Mohapatra B, Singh S and Bhattacharyya P (2004) "Morphology based Natural Language Processing tool for Indian languages" Available at http://www.cse.iitb.ac.in

[6] Ray P R, Harish V, Sarkar S and Basu A (2003) "Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi" In Proceedings of the 1st International Conference on Natural Language Processing (ICON 2003), Mysore, India.

[7] Awasthi P, Rao D and Ravindran B (2006) "Part of Speech Tagging and Chunking with HMM and CRF" In Proceedings of NLPAI Machine Learning Contest 2006, India.

[8] Shrivastava M and Bhattacharyya P (2008) "Hindi POS Tagger Using Naïve Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge" In Proceedings of 6th International Conference on Natural Language Processing, ICON 2008, India.

[9] Dalal A, Nagaraj K, Swant U, Shelke S and Bhattacharyya P (2007) "Building Feature Rich POS Tagger for Morphologically Rich Languages: Experience in Hindi" In Proceedings of 5th International Conference on Natural Language Processing, ICON 2007, India.

[10] Agrawal H (2007) "POS Tagging and Chunking for Indian Languages" In Proceeding of IJCAI Workshop on Shallow Parsing for South Asian Languages, Hyderabad, India.

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

[12] Rao D, Yarowsky D (2007) "Part of Speech tagging and Shallow Parsing for Indian Languages" In Proceedings of the workshop on Shallow Parsing for South Asian Languages at the International Joint Conference in Artificial Intelligence, India.

[13] Verma D P, Rakesh M, Sanyal R (2007) "HMM-based Language-independent POS Tagger" In Proceedings of the 3rd Indian International Conference on Artificial Intelligence, Pune, India.

[14] Dandapat S, Biswas P, Choudhury M, Bali K (2009) "Complex Linguistic Annotation- No Easy Way Out! A case from Bangla and Hindi POS Labeling Tasks" In Proceeding of the 3rd Linguistic Annotation Workshop (ACL-IJCNLP), Suntec, Singapore.

[15] http://www.idcil.org/resourcesTextCorp.aspx

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

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

[18] Jurafsky D and Martin J H (2009) "Speech and Language Processing" 2nd Edition, Prentice-Hall, Inc., USA.

[19] https://github.com/gayatri-01/POS-Tagging-in-Hindi-Document

[20] http://universaldependencies.org.

[21] https://paperswithcode.com/datasets?lang=hindi

[22] Mishra R and Gayen T (2018) "Automatic lossless summarization of news articles with abstract meaning representation" Procedia Computer Science, **135**:178-185, Elsevier.

[23] Arumugam R and Shanmugamani R (2018) "Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications", Packt Publishing.

[24] Prajapati M and Yajnik A (2019) "POS Tagging of Gujarati Text using VITERBI and SVM" International Journal of Computer Applications. **181(43)**: 32-35.

[25] Cahyani D. E and Vindiyanto M, J (2019) "Indonesian Part of Speech Tagging Using Hidden Markov Model – Ngram & Viterbi" In Proceedings of 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Indonesia.

[26] Khedkar V and Shah P (2021) "Viterbi based Parts of Speech tagging for Hindi and Marathi" International Journal of Advanced Research in Engineering and Technology, **12(1)**: 753-759.

[27] Jayan J P and Rajeev R R (2011) "Parts Of Speech Tagger and Chunker for Malayalam: Statistical Approach" Computer Engineering and Intelligent Systems, **2(3)**: 6-10.