Building Feature Rich POS Tagger for Morphologically Rich Languages: Experiences in Hindi

Aniket Dalal
CSE department
IIT Bombay
Mumbai

Kumar Nagaraj
CSE department
IIT Bombay
Mumbai

Uma Sawant
CSE department
IIT Bombay
Mumbai

Sandeep Shelke
CSE department
IIT Bombay
Mumbai

Pushpak Bhattacharyya
CSE department
IIT Bombay
Mumbai

(aniketd,kumar,uma,sandy,pb) @cse.iitb.ac.in

Abstract

In this paper we present a statistical part-of-speech(POS) tagger for a morphologically rich language: Hindi. To the best of our knowledge, our tagger achieves the highest reported tagging accuracy for Hindi. Our tagger employs the maximum entropy Markov model with a rich set of features capturing the lexical and morphological characteristics of the language. The feature set was arrived at after an exhaustive analysis of an annotated corpus. The morphological aspects are addressed by features based on information retrieved from a lexicon generated from the corpus, a dictionary of the Hindi language and a stemmer. The system was evaluated over a corpus of 15,562 words developed at IIT Bombay. We performed 4-fold cross validation on the data, and our system achieved the best accuracy of 94.89% and an average accuracy of 94.38%. Our work shows that linguistic features play a critical role in overcoming the limitations of the baseline statistical model for morphologically rich languages.

1 Introduction

A POS tagger assigns appropriate part-of-speech categories (e.g., noun, verb, adverb etc.) to unseen text. Such a tagger is required for many applications, such as word sense disambiguation, parsing etc. Part-of-speech tagging has been studied extensively in the past two decades (section 2 discusses related work). The fundamental problem in POS tagging task stems from the fact that a word can take different lexical categories depending on its context. The tagger has to resolve this ambiguity and determine the best tag sequence for a sentence.

Most of the work in tagging is concentrated on corpus-rich languages like English. In this paper we deal with POS tagging for Hindi, the national language of India and ranking 4^{th} in the world in terms of the population size speaking it. Though not as rich as English in terms of annotated corpora, Hindi is morphologically rich - which is the motivation for the work reported in the paper.

The morphological richness of Hindi language increases the complexity of tagging. Hindi is a free word order language, which implies that no fixed order is imposed on the word sequence. This creates difficulties for a statistical tagger as many permutations of the same string are possible. Additionally, by combining various morphemes, several words can be generated, which may not be present in the reference resources.

Various approaches to POS tagging have been studied so far, which can be divided in two broad categories, namely, rule based and statistical. Considering tagging as a stochastic process, we can build a statistical model to predict tag sequences. A statistical model learns required probability distributions from the training data and applies them to the unseen text. Our approach uses an exponential model known as the Maximum Entropy Markov Model(MEMM) (Ratnaparkhi, 1996).

2 Related work

There have been many implementations of POS tagger using machine learning techniques, mainly for corpus-rich languages like English. Such as, transformation-based error-driven learning based tagger (Brill, 1995) and maximum entropy Markov model based tagger (Ratnaparkhi, 1996).

A POS tagger for English based on probabilistic triclass model was developed in (Merialdo, 1994). (Brants, 2000) proposed TnT, a statistical POS tagger based on Markov models with a smoothing technique and methods to handle unknown words. (Nakagawa et al., 2001) presents a method to predict POS tags of unknown English words as a post-processing of POS tagging using Support Vector Machines (SVMs), which can handle a large number of features.

Another approach for POS tagging is based on incorporating a set of linguistic rules in the tagger. A comparison (Samuelsson and Voutilainen, 1997) between stochastic tagger and tagger built with hand-coded linguistic rules shows that for the same amount of remaining ambiguity, the error rate of the statistical tagger is one order of magnitude greater than that of the rule-based one. Some implementations combine the statistical approach with the rule-based, to build a hybrid POS tagger. Such a tagger was constructed by (Kuba et al., 2004) for Hungarian, which shares many difficulties such as free word order, with Hindi. Ezeiza and others (Ezeiza et al., 1998) built a hybrid tagger for agglutinative languages.

There has been some previous work towards building a Hindi POS tagger, such as the partial POS tagger discussed in (Ray et al., 2003). Shrivastava et al. propose harnessing morphological characteristics of Hindi for POS tagging in (Shrivastava et al., 2005). This was further enhanced in (Singh et al., 2006), which suggests a methodology that makes use of detailed morphological analysis and lexicon lookup for tagging. The results are further improved by applying disambiguation rules learnt from modestly sized corpora.

Hindi, unlike English, belongs to the category of inflectionally rich languages which suffer from data sparseness problem. Hajič (Hajič, 2000) argues strongly in favor of use of an independent morphological dictionary over collecting more annotated data. Hajič also proposes to further enrich some of the best taggers available by making use of the dictionary information. Uchimoto et al. (Uchimoto et al., 2001) describe a morphological analysis method based on a maximum entropy model. This method uses a model that not only consults a dictionary with large amount of lexical information but also identifies unknown words by learning certain characteristics.

3 Methodology

We treat POS tagging as a stochastic sequence labelling task, in which, given an input sequence of words $W = w_1 w_2 ... w_n$, the task is to construct a label sequence $T = t_1 t_2 ... t_n$, where t_i belongs to the set of POS tags. The label sequence T generated by the model is the one which has highest probability among all the possible label sequences for the input word sequence W, that is

$$T = argmax \left\{ Pr(T^{'}|W) \right\} \quad (1)$$

where $T^{'}$ belongs to list of possible label sequences. We employ a feature driven, exponential model based learner for tagging. The underlying model is maximum entropy Markov model (MEMM). The general formulation of MEMM model is given as (Berger et al., 1996):

$$p(t|c) = \frac{1}{Z} exp \sum_{i=1}^{n} \lambda_i f_i(c,t) \quad (2)$$

where Z is the normalization factor and $p(t|c)$ is the probability of tag t being assigned for a context c. Also, $f_i(c,t)$ is a binary valued feature function on the event (c,t). A set of such feature functions are defined to capture relevant aspects of the language. The model parameters λ_i's are determined through Generalized Iterative Scaling (GIS) (Darroch and Ratcliff, 1972) algorithm.

The system architecture is shown in figure 1. The dotted part, which includes the learner and the tagger, is the heart of the system. Note that the system incorporates training time information (through training data) as well as prior belief (through dictionary[1]). The learner puts together all this information and generates the model. This model is then used by the tagger to tag the raw data file.

3.1 Feature functions

A crucial aspect of feature based probabilistic modelling is to identify the appropriate facts about the data. We have developed a rich set of features capturing lexical and morphological characteristics of the language. The feature set was arrived at after an exhaustive analysis of an annotated corpus. The morphological aspects of the language are addressed by features based on information retrieved from dictionary, lexicon and stemmer.

[1]The dictionary contains only paradigmatic and categorical information as explained in (Shrivastava et al., 2005)

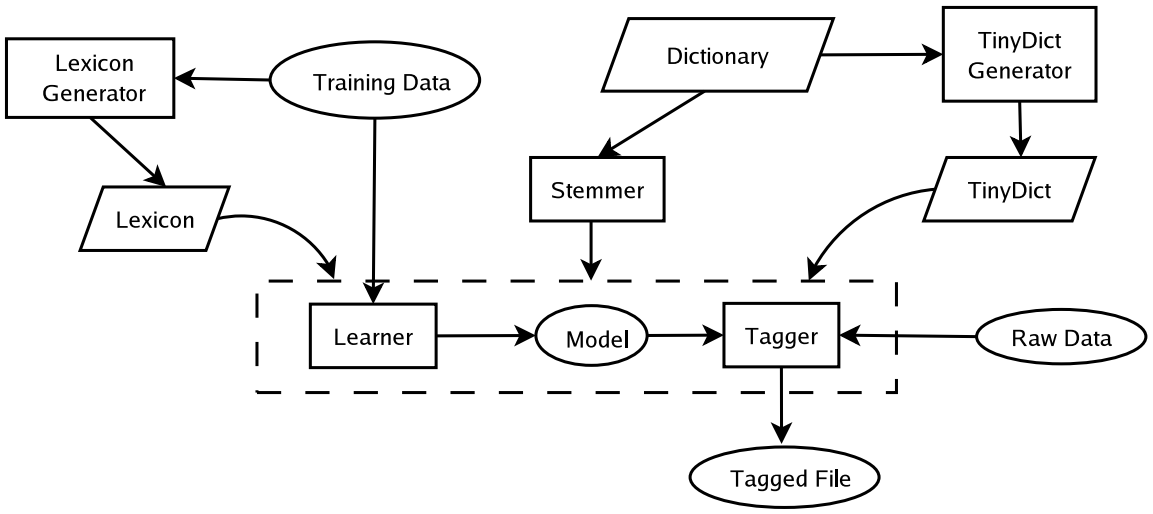

Figure 1: System architecture

Contextual features[2]

Sense disambiguation has been a longstanding problem in computational linguistics. In most of the cases the ambiguity can be resolved using the context of the usage. Consider an example Hindi statement
आज सोने का भाव क्या है?
aaja sone kaa Bhaava kyaa hai?
today gold-of price what is?
What is the price of gold today?

The word सोने [*sone*] can take two forms, noun (gold) and verb (sleep). The ambiguity between the two forms can be resolved only when word भाव [*Bhaava*] (price) is encountered. To resolve such kind of ambiguities we define a feature set within a context window. The size of the context window is determined based on empirical observations.

For a context window $c =< t_{i-1}, w_i, w_{i+1} >$, the context based feature templates are

$$f_{prevTag}(c,t) = \delta(t_j,t).\delta(t_k,t_{i-1}) \quad (3)$$
$$f_{word}(c,t) = \delta(t_j,t).\delta(w,w_i) \quad (4)$$
$$f_{nextWord}(c,t) = \delta(t_j,t).\delta(w,w_{i+1}) \quad (5)$$

for all $t_j, t_k \in T$ and $w \in W$. Here, w_i is the word at i^{th} position, t_i is its tag, T is the tagset, W is the set of all words and δ is the Kronecker delta function.

[2]In our model, contextual features defi ne baseline system. In table 4, baseline system is tagger with just contextual features.

Morphological features

Another classical problem in computational linguistics is tagging of unseen words. These are set of words which are not observed in the training data and hence there are no context based events within the model to facilitate correct tagging. Our system uses a stemmer, a module which uses the dictionary and outputs the list of suffixes for a given word. We use the presence of suffixes as a morphological feature. An example is the suffix ना [*naa*] (roughly a gerundial marker). Words having ना [*naa*] as suffix belong to the verb class, for example, तैरना [*tairanaa*] (swimming), भागना [*Bhaaganaa*] (running), चलना [*chalanaa*] (walking).

Let S be the set of all possible suffixes. For every context window c, the suffix based feature is defined as :

$$f_{suf}(c,t) = \delta(t_i,t).\delta(s,suf(w_i)) \;\; \forall\, s \in S \quad (6)$$

where, $suf(w_i)$ is a suffix for the current word within an event (c,t).

Lexical features

English letters and numerals are frequently used in Hindi texts to represent the information of things like year, quantity etc. In cases like 20वी सदी मै [*biisavii sadii mai*] (in 20^{th} century) the Hindi suffix gets attached to English digits to form a word. Words like ”IBM”, ”ISRO”, ”IIT” are used in their original form in Hindi texts. We take care of such possibilities where the data is not necessarily clean by defining features that detect anomalies in the text. We also add a feature

for dealing with special symbols and punctuation characters. Mathematically, the feature functions for capturing these properties (English characters, special characters) can be represented as

$$f_{prop_a}(c,t) = \delta(t_i,t).\delta(prop_a(c), true)$$

where $prop_a(c)$ is a function that returns true if the property a holds in the context c.

Categorical features

Our approach extensively uses the lexical properties of words in feature functions. This is achieved by collecting categorical information from the dictionary. It is known that parts-of-speech for a word is restricted to a limited set of tags. For example, word आम [*aama*] has one of the two possible POS categories, noun (mango) and adjective (common). We use this restricted set of POS categories for a word as a feature. This boosts the probability of assigning a POS tag belonging to the restricted category list as tag for the word. This feature is crucial for unseen words where there is no explicit bias for a word in the built model and we produce an artificial bias with the help of limited tag set. A special case of this feature is when the restricted category list has exactly one POS tag, which implies that the word would be tagged with that particular tag with very high probability.

More formally, the feature functions based on the dictionary are

- Can the word occur with a particular tag according to the dictionary?

$$f_{tagset}(c,t) = in(tagset_{dict}(w_i), t) \quad (7)$$

  where, $tagset_{dict}(w_i)$ is the set of tags for w_i according to the dictionary and $in(l,b)$ is true if $b \in l$.

- Does the word have a single possible tag according to the dictionary?

$$f_{singleTag}(c,t) = \delta(|tagset_{dict}(w_i)|, 1) \quad (8)$$

- Does the word have a single possible tag of *proper noun* according to the dictionary? This feature is the conjunction of the previous two feature functions with proper noun as the tag.

Compound features

The lexicon is generated from the training data, and it contains a detailed account of the observed facts. An extensive analysis and understanding of the language structure enabled us to come up with a rule based feature set which helps in improving the performance of the tagger, especially for proper nouns. These rules constitute compound features as they are based on information from multiple resources. Following are the rules applied, in order:

- Is the word absent in the lexicon (unseen word)?

$$f_{unseen}(c,t) = \delta(isSeen(w_i), false) \quad (9)$$

  where, $isSeen(w_i)$ returns true if w_i is in lexicon.

- Can the unseen word occur as proper noun according to the dictionary or is the unseen word unknown to the dictionary? This feature function is conjunction of feature of equation 7 with proper noun as the tag and feature 9.

- Did the word occur as proper noun in the lexicon?

$$f_{lexPPN}(c,t) = lex_{PPN}(w_i) \quad (10)$$

  where $lex_{PPN}(w_i)$ is true if the proper noun flag is set for w_i in lexicon.

- Did the word occur as proper noun in the lexicon and it is also a proper noun according to the dictionary or unknown to the dictionary? This feature function is true if feature 10 is true and the feature 7 is true with proper noun as the tag.

- Did the word never appear with proper noun tag in the training corpus and either the word can occur as a proper noun as per the dictionary or the word is unknown to the dictionary? This compound feature is conjunction of feature function 7 with proper noun as the tag and negation of feature 10.

4 Experimental setup and results

In this section, we outline our experimental setup and discuss the effect of the feature functions on the system performance.

4.1 Data set

Data for our experiments was taken from Hindi news corpus of BBC[3] and manually tagged at IIT Bombay. This data set consisted of 15562 words tagged with 27 different POS tags. Although the data set is of moderate size, care was taken to ensure that data was not limited to a particular domain by adding news items from wide range of topics. This data was spread across four files and each file had approximately same number of words. We performed four fold cross validation on this data set.

4.2 Preprocessing

Our system processes data in two phases. In the first phase, resources necessary for the tagging phase are generated. The generated resources include the list of unique words in the training corpus, called *lexicon*, and a restricted dictionary called *TinyDict*. A lexicon generator is run on the training corpus to create the lexicon. In the lexicon, along with the word, a flag is stored to indicate occurrence of the word as a proper noun in the training corpus. If a word appears with the tag proper noun at least once in the training corpus, then this flag is true. The purpose of lexicon is two fold: (a) it serves as a list of seen proper nouns, and (b) it serves as an indicator for seen words, so that the information from the restricted dictionary can be utilized for unseen words.

For every word in the corpus, TinyDict stores information about the list of possible POS tags according to the dictionary. Another resource that is generated in preprocessing phase is the list of suffixes for all words using stemmer [4]. To avoid the duplication in resources and processing, the list of suffixes is appended to the POS tags in TinyDict. In other words, along with the list of possible POS tags for a word, TinyDict also stores suffixes of that word.

Note that the lexicon stores only those words that appear in the training corpus, whereas TinyDict has information about both training and test corpus. This does not violate the basic rules of tagging as TinyDict is a summarization of relevant information from the dictionary, and the dictionary is for the whole language.

Tables 1 and 2 show excerpts from the dictionary and the lexicon, respectively.

[3]BBC Hindi news at http://www.bbc.co.uk/hindi/

[4]The Hindi stemmer was developed by IIT Bombay

Word	Suffixes	POS Categories	Root
बनाता	ता	verb	बना
शिया		proper-noun, adj	शिया
जुलूस		noun	जुलूस
पाएगा	एगा	verb-aux, verb	पा
एक		cardinal, noun	एक

Table 1: Dictionary

Word	Proper Noun flag
भारत	TRUE
मै	FALSE
पुलिस	FALSE
कहना	FALSE
उमा	TRUE

Table 2: Lexicon

4.3 Context window

The best context window was determined empirically. Our initial context window consisted of two words on either side of the current word, POS tags for the previous two words and the combination of these POS tags. The best per word tagging accuracy of the tagger for this context window without any other feature function was 77.73%. We experimented with the context window by trying different combinations of surrounding words and their POS tags. The best result of 85.59% was obtained with the context window consisting of the POS tag of the previous word, the current word and the next word. Best per word tagging accuracies along with corresponding sentence accuracies[5] are reported in table 3. In the table, we use $word_{i-2}^{i+2}$ to mean all words in the sequence $word_{i-2}$ to $word_{i+2}$, with i as the index for current word being tagged. Similar notation is followed for tags and (tag_{i-2}, tag_{i-1}) stands for combination of the tags tag_{i-2} and tag_{i-1}.

4.4 Influence of feature functions

The best per word tagging accuracy that can be obtained using appropriate context window is 85.59%. We call this as the *baseline* tagger. As can be expected, the addition of linguistic features boosts the performance of the baseline system. Improvement in performance with the addition of each feature function is summarized in table 4. The addition of TinyDict suggested pos-

[5]Sentence accuracy is the ratio of number of complete sentences tagged correctly to the number of sentences tagged.

Context window	Per word accuracy	Sentence accuracy
$(tag_{i-2}, tag_{i-1}), tag_{i-2}^{i-1}, word_{i-2}^{i+2}$	77.73	3.91
$(tag_{i-2}, tag_{i-1}), tag_{i-2}^{i-1}, word_{i-2}^{i+1}$	79.85	2.79
$(tag_{i-2}, tag_{i-1}), tag_{i-2}^{i-1}, word_{i-1}^{i+1}$	83.17	4.47
$(tag_{i-2}, tag_{i-1}), tag_{i-1}, word_{i-1}^{i+1}$	84.06	5.59
$(tag_{i-2}, tag_{i-1}), tag_{i-1}, word_{i}^{i+1}$	84.71	7.82
$tag_{i-1}, word_{i-1}^{i+1}$	84.86	7.26
$tag_{i-1}, word_{i-1}^{i}$	83.52	8.94
$\mathbf{tag_{i-1}, word_{i}^{i+1}}$	**85.59**	**8.94**
$word_{i}^{i+1}$	82.18	5.03
$word_i$	77.27	5.59

Table 3: Different context windows and corresponding results

Feature function	Per word accuracy	Sentence accuracy
Baseline	85.59	8.94
Morphological	86.57	11.18
Lexical	86.95	11.18
Categorical	94.39	33.52
Compound	94.89	38.54

Table 4: Performance gain with addition of feature functions

Data set	Per word accuracy	Sentence accuracy	Unseen word accuracy
Set1	94.89	37.99	92.32
Set2	94.26	35.00	92.23
Set3	94.65	34.29	92.50
Set4	93.73	33.33	93.25

Table 5: Results of four fold cross validation.

sible POS tags as feature greatly improves the accuracy of the system. This is because, for unseen words the information in TinyDict aids in determining the set of possible POS tags.

4.5 Implementation

Our POS tagger was developed in Java[6] and uses the maxent[7] package for maximum entropy model. This package employs generalized iterative scaling (GIS) algorithm to estimate the model parameters. The number of iterations for GIS is configurable and we ran the algorithm for 100 iterations. During the tagging phase, beam search algorithm is employed to find the most promising tag sequence with a beam width of 6. Typical execution times on an Intel Pentium 4 machine with linux are approximately 14.79 seconds for training and 2.30 seconds for tagging.

4.6 Results

We use two measures to evaluate the performance of our system, namely, per word tagging accuracy and sentence accuracy. Per word tagging accuracy is the ratio of number of words that are tagged correctly to the number of words present in the text. Sentence accuracy represents the percentage of sentences for which the tag sequence assigned by the system matches the true tag sequence. If all the words in a sentence are assigned correct tags, then the sentence is said to be correctly tagged. Sentence accuracy is the ratio of correctly tagged sentences to the number of sentences present in the

[6] Java at http://java.sun.com

[7] maxent package for Maximum Entropy Markov models at http://maxent.sourceforge.net/

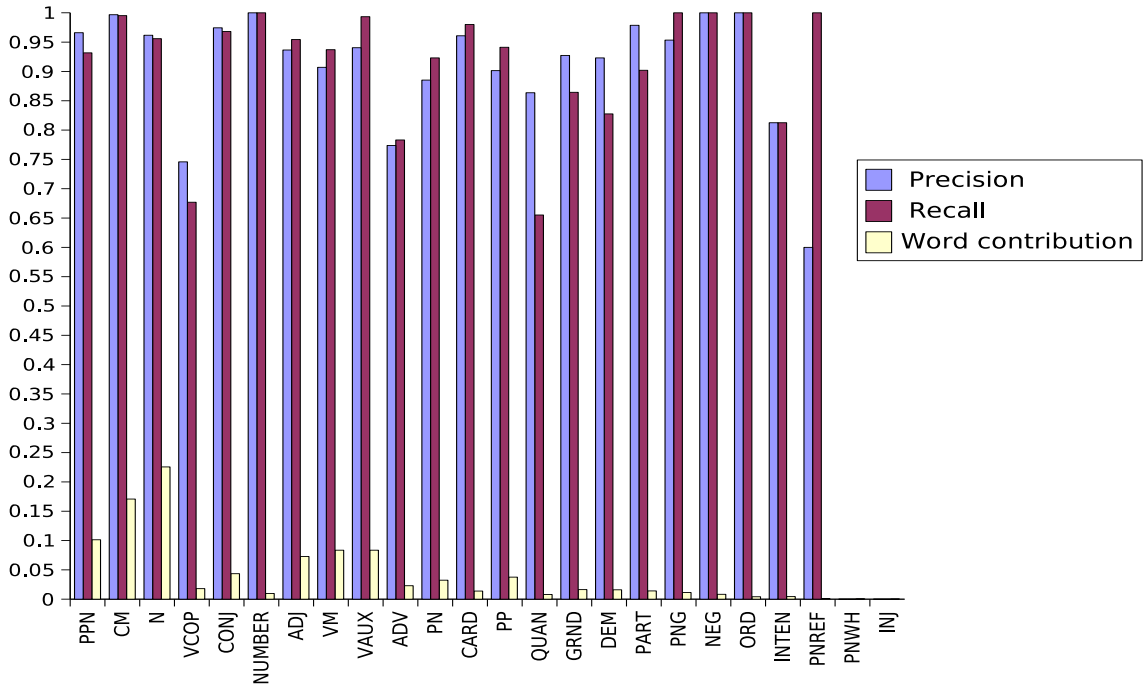

Figure 2: Per tag accuracy

text.

We performed 4-fold cross validation on the data. The results of 4-fold cross validation are provided in table 5. The best per word tagging accuracy of our system is 94.89% and the average per word tagging accuracy is 94.38%. The best and average sentence accuracies are 37.99% and 35.15%. To the best of our knowledge, these are the highest reported accuracies for Hindi. On the same data set and a similar tag set (Singh et al., 2006) reports a best accuracy of 93.45%.

4.7 Performance analysis

The graph in figure 2 shows precision, recall and total number of occurrences of individual tags. From the figure, it can be observed that precision and recall for categories like case marker (CM), negation (NEG), pronoun genitive (PNG) and conjuction (CONJ) are exceptionally good, as these are closed word list categories. In case of numbers (NUMBER) also the performance of the tagger is excellent, as they are handled by lexical features.

One of the main challanges in POS tagging is correct identification of proper nouns (PPN) and disambiguating them from nouns (N). Our tagger has considerably high precision and recall for both categories. This can be attributed to the set of compound features specifically designed for handling proper nouns.

In contrast, adverbs (ADV), quantifiers (QUAN) and intensifiers (INTEN) display low recall and average precision. This is due to the substantially less number of training instances for these categories. We observed that less then 2.4% of training instances are adverbs, wheras for Quantifiers and intensifiers the percentage of training instances is less then 1.

Precision and recall for verb-copula (VCOP) are low even though the training data has considerable number of instances. Our tagger tends to frequently misclassify VCOP's as verb-main (VM), because of ambiguity in words like है [*hai*] (is) , था [*tha*] (was) which can occur as VCOP as well as VM in similar sentence structure/context. The difference is largely semantic and it is hard to disambiguate at syntactic level.

4.8 Unseen words

To handle unseen words, information from lexicon and TinyDict are used as feature functions. A word is *unseen* if it is not present in training corpus. Equivalently, if a word is absent in lexicon, then it is unseen. A feature function is defined to capture the information that a word is unseen. Feature functions are also defined on the possibility of an unseen word occurring as proper noun accord-

ing to TinyDict. On an average 19% of test data consisted of unseen words. The best and average tagging accuracies for unseen words were 93.25% and 92.58%, respectively.

5 Case study : IIIT Hyderabad corpus

We conducted experiments on Hindi corpus provided as part of NLPAI-ML 2006 contest[8] by IIIT Hyderabad (IIITH). We outline the modifications made to the feature set for this corpus and the results in this section.

5.1 IIIT Hyderabad corpus : feature functions

The tagset of IIITH corpus consisted of 29 different POS tags. These POS tags were considerably different from the POS tags of IITB corpus. Although our approach is largely independent of corpus and its tagset, the mismatch in tags of TinyDict and the IIITH tagset resulted in reduced performance. To overcome this, for every word, we appended the information stored in the lexicon with its POS tags. Specifically, the list of unique words in the training data and the set of POS tags with which that word appears in the training data are stored in the lexicon for IIITH corpus. This set of POS tags are used in feature functions in place of the possible POS tags provided by TinyDict.

IIITH tagset has tags to represent the notion of *kriyamuls*[9]. The root of the next word plays a crucial role in identifying such kriyamuls. A feature function is defined to capture this relation on the root of the next word.

5.2 IIIT Hyderabad corpus : results

The development corpus for the task was provided by contest organizers. We conducted experiments with a 75-25 split for training and test data in our experiments. The results were averaged out across different runs, each time randomly picking training and test data. The best POS tagging accuracy of the system in these runs was found to be 89.34% and the least accuracy was 87.04%. The average accuracy over 10 runs was 88.4%. In the final round of the contest, our system had the highest POS tagging accuracy for Hindi and second highest among all languages.

[8]NLPAI Machine Learning Contest 2006, http://ltrc.iiit.net/nlpai_contest06/proceedings.php

[9]As defined by the contest organizers, kriyamul is a verb formed by combining a noun or an adjective or an adverb with a helping verb.

6 Discussions

In this paper we have showed that contextual, morphological and lexical feature of a language when used judiciously can deliver high performance for a morphologically rich language like Hindi. We have also discussed the exact nature of various features and their role in boosting the tagging accuracy for stochastic exponential-model-based tagger. Our system reached the best accuracy of 94.89% and an average accuracy of 94.38%.

We have developed a stochastic tagger to which morphological and linguistic features can be easily augmented through resources like stemmer, dictionary and lexicon. Our methods have distinctive advantage over other pure stochastic and rule based linguistic systems as it provides a simplistic approach for embedding the linguistic properties within a stochastic model. Rule based systems are strongly coupled with the language specific properties and the associated tag set, whereas pure stochastic systems fail to capture language specific peculiarities. Our method overcomes the shortcoming of both the approaches and can be easily extended to other morphologically rich languages just by building language appropriate resources like lexicon and stemmer.

7 Acknowledgment

We would like to thank Manish Shrivastava for many helpful suggestions and comments.

References

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Thorsten Brants. 2000. Tnt – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, April.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480.

N. Ezeiza, I. Alegria, J. M. Arriola, R. Urizar, and I. Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual*

Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pages 379–384, San Francisco, California. Morgan Kaufmann Publishers.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 6th Applied Natural Language Processing and the 1st NAACL Conference*, pages 94–101.

András Kuba, András Hócza, and János Csirik. 2004. Pos tagging of hungarian with combined statistical and rule-based methods. In *Proceedings of the 7th International Conference on Text, Speech and Dialogue*, pages 113–120.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. 2001. Unknown word guessing and part-of-speech tagging using support vector machines. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 325–331, Tokyo, Japan.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.

P. R. Ray, V. Harish, A. Basu, and S. Sarkar. 2003. Part of speech tagging and local word grouping techniques for natural language parsing in hindi. In *Proceedings of the International Conference on Natural Language Processing (ICON 2003)*, Mysore.

Christer Samuelsson and Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 246–253, Morristown, NJ, USA. Association for Computational Linguistics.

M. Shrivastava, N. Agrawal, S. Singh, and P. Bhattacharya. 2005. Harnessing morphological analysis in pos tagging task. In *Proceedings of the International Conference on Natural Language Processing (ICON 05)*, December.

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. 2006. Morphological richness offsets resource poverty- an experience in building a pos tagger for hindi. In *Proceedings of Coling/ACL 2006*, Sydney, Australia, July.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 91–99.