Hindi POS Tagger Using Naive Stemming : Harnessing Morphological Information Without Extensive Linguistic Knowledge

Manish Shrivastava
Department of
Computer Science and Engineering,
Indian Institute of Technology
Bombay
manshri@cse.iitb.ac.in

Pushpak Bhattacharyya
Department of
Computer Science and Engineering,
Indian Institute of Technology
Bombay
pb@cse.iitb.ac.in

Abstract

Part of Speech tagging for Indian Languages in general and Hindi in particular is not a very widely explored territory. There have been many attempts at developing a good POS tagger for Hindi, but the morphological complexity of the language makes it a hard nut to crack. Some of the best taggers available for Indian Languages employ hybrids of machine learning or stochastic methods and linguistic knowledge. Though, the results achieved using such methods are good, there practicability for other inflective Indian Languages is reduced due to their heavy dependence on linguistic knowledge. Even though taggers can achieve very good results if provided good morphological information, the cost of creating these resources renders such methods impractical.

In this paper, we present a simple HMM based POS tagger, which employs a naive(longest suffix matching) stemmer as a pre-processor to achieve reasonably good accuracy of 93.12%. This method does not require any linguistic resource apart from a list of possible suffixes for the language. This list can be easily created using existing machine learning techniques. The aim of this method is to demonstrate that even without employing tools like morphological analyzer or resources like a pre-compiled structured lexicon, it is possible to harness the morphological richness of Indian Languages.

1 Introduction

Part of Speech tagging is the one of the most basic problems of NLP. It is the process of assigning correct part of speech to each word of a given input text depending on the context. The task belongs to a larger set of problems, namely, sequence labelling problems. Some of the other tasks which belong to this set are Speech Recognition, Optical Character recognition, Chunking *etc.* All these problems deal with assigning labels to discreet components of the input. A variety of methods have been tried for POS tagging over the years. The common methods employed for POS tagging of western languages include machine learning techniques like Transformation-Based Error-Driven learning(Brill, 1992), decision trees (Black et al., 1992),Hidden Markov Models (Cutting et al., 1992), maximum entropy methods (Ratnaparakhi, 1996) *etc.* Hybrid taggers have also been tried using both stochastic and rule-based approaches, such as CLAWS (Garside and Smith, 1997).

Though there are obviously many approaches to POS tagging, tagging of Indian Languages still poses a challenge. This is due to the morphological richness of Indian Languages. Morphologically rich languages typically have more than one morpheme in a word usually fused together. This renders fixed context stochastic methods useless(Samuelsson et al., 1997). POS tagging of some morphologically rich languages has been attempted earlier using hand-crafted rules and stochastic tagging methods(Hajic et al., 2001; Tlili-Guiassa, 2006; Uchimoto et al., 2001;

[0]Proceedings of ICON-2008: 6th International Conference on Natural Language Processing, Macmillan Publishers, India. Also accessible from http://ltrc.iiit.ac.in/proceedings/ICON-2008

Oflazer and Kuruoz, 1994). These systems typically use large corpora with detailed morphological analysis for the purpose of POS tagging. It is seen that neither rule-based nor stochastic methods have been sufficient for POS tagging of morphologically rich languages as rule based methods require expert linguistic knowledge and stochastic methods need very large corpora to be effective.

1.1 POS tagging of Hindi

In recent years a lot of work has gone into the POS tagging of Indian Languages, specifically, Hindi. Typically, stochastic methods have been combined with linguistic resources to achieve reasonably good results. The known works in POS tagging of Hindi and, more generally, Indian Languages are (Ray et al., 2003; Bharati et al., 1995; Dandapat et al., 2007; Dandapat et al., 2004; Singh et al., 2006). All these methods are either rule based or work using some combination of rule based and stochastic techniques. One common factor in all these approaches is the extensive use of detailed morphological analysis either for preliminary tagging (Singh et al., 2006; Ray et al., 2003; Bharati et al., 1995) or for restricting a stochastic model(Dandapat et al., 2004; Dandapat et al., 2007; N. et al., 2006). These are attempts to compensate for the failures of stochastic models by utilising the morphological richness of a language. These approaches make it obvious that harnessing morphology is crucial to good performance of POS taggers. But, the cost associated with developing a good morphological analyzer takes away some of the allure of these approaches.

In this paper, we present a simple POS tagger based on Hidden Markov Models(HMM) for the task of POS tagging. We attempt to utilize the morphological richness of the languages without resorting to complex and expensive analysis.

2 "Exploding" Input

The core idea of this approach is to "explode" the input in order to increase the length of the input and to reduce the number of unique types encountered during learning. This in turn increases the probability score of the correct choice while simultaneously decreasing the ambiguity of the choices at each stage. This also decreases data sparsity brought on by new morphological forms for known base words. For example, if we assume that the following sentence is seen in the training data:

घर	का	खाना	अच्छा	लगता	है .
house	gen	food	good	feel (Habitual)	present
English:					
Home	food	feels	good.		

And, the following sentence is found in the test data:

कई	घरों	के	खाने	कि	खुशबू
many	houses	gen	food (obl)	gen	smell
आ	रही	थी.			
come	ing (fem)	past (fem)			
English:	Smell	of	food	is	coming
from	many	houses.			

Further, if the word घरों has never been encountered in the training data then the model would treat the word as an unknown during testing. Here, no human annotator would commit a mistake even if घरों is never seen before. Just by knowing the morphology of nouns a human can predict that घरों is a noun(plural) and not a new, unknown word. The same facts apply to the word खाना .

We can see that the only problem in identifying the form घरों is the suffix which resulted in a new form which was never seen. If we can just remove the suffix we will be left with an underlying form which is common to both sentences and hence, observed during learning. One method of doing this would be to remove all inflections from all words of the data leaving just the base form. That is, The sentences would be written as:

घर	का	खाना	अच्छा	लग	है .
house	gen	food	good	feel(base)	present

कई	घर	के	खाना	कि	खुशबू
many	houses	gen	food(base)	gen	smell
आ	रह	था.			
come	ing	past			

While this method would solve the problem of sparsity due to multiple types, it also loses all the information contained in the suffixes. We know that a suffix contains a very good indication of the category of a word as the category suffix are usually either unique or can occur in no more than a few categories reducing the ambiguity for the accompanying stem. Thus it is essential that the suffix be preserved and used for further disambiguation.

The most favorable method of splitting would be to find the exact suffix and root form from the word. Once we have these two parts of the word they can be treated as separate tokens. That is, for the above sentences the best representation would be :

घर	का	खाना	अच्छा	लग	ता
house	gen	food	good	feel(base)	Habitual
है					
present					

And,

कई	घर	ओं	के	खाना	ए
many	house	Plural	gen	food(base)	obl
कि	खुशबू	आ	रह	ई	था
gen	smell	come	ing	fem	past
ई.					
fem					

Unfortunately, this requires quite precise stemming which is hard to achieve in practice. Also, the words here are in root forms which can only be arrived at by using a lexicon for cross-validation. This processing would require a rule based stemmer system which would again make us rely on extensive linguistic resources, which is something that we want to avoid. Thus, we need to rely on a stemming which is simpler but effective. In our efforts, we found that a simple longest suffix removal works reasonably well.

2.1 Longest Suffix Splitting

In case of a simple stemming, the result is a stem and a probable suffix. We need a method where the result should be consistent for both the testing and training phases. During training the tag associated with the word can at times disambiguate between multiple possible suffixes. But, we cannot rely on tag information because that would not be available at testing stage. We realized that the quest for a consistent stemming scheme ends by providing a simple list of all possible suffixes in the language can be used for splitting resulting in a crude and not very linguistically sound stemming. Though, this approach lack linguistic strength, it works very well for our purposes.

Assuming that { आ , ओं , ई , ता , ए } are in the list of suffixes, the sentences above will look like:

घर	का	खान्	आ	अच्छा	लग
house	gen	food	'A'	good	feel(base)
ता	है				
Habitual	present				

And,

कई	घर	ओं	के	खान्	ए
many	house	Plural	gen	food(base)	obl
कि	खुशबू	आ	रह	ई	थ
gen	smell	come	ing	fem	past
ई.					
fem					

This form becomes the new input sequence for HMM. Suffix list for a language is not very hard to create. For most languages, this list is readily available. Though, we used a manually created list of all possible suffixes for the purpose of stemming, it is possible to learn these suffixes using suitable machine learning methods (Goldsmith, 2001), thus, making this a very feasible method for quick and easy morphology infusion in a stochastic method. It can be seen that खाना incorrectly stemmed. Naive stemming will result in such errors but, we show later that this compromise is worth the results in most cases.

2.2 Suffix Tags

After this stemming and exploding of input, the exploded inflected tokens result in 2 tokens in the new corpus : the stem and the suffix. The next problem is that of assigning tags to the newly introduced symbols of the input i.e. the suffixes. For example, घरों_NN would result in घर which can be tagged NN and ओं which needs to be tagged. This can be done in four possible ways:

1. To assign category tags which are indicative of the category of the original inflected word but not exactly same. For example, in case of घरों, घर is tagged NN whereas ओं is tagged SNN.

2. To assign individual tags to each suffix that we encounter, preferably the suffix itself can be repeated as the tag. For example, tag for ओं would be ओं represented as ओं_ओं.

3. To assign tags, which are indicative of category as well as the suffix. Such as, ओं_Sओं. This turns out to be the same as the second method.

4. To assign exactly the same tag as the inflected word. This is not a good idea as it does not distinguish between word and suffix.

The experiments were carried out using methods 1 and 2. There are very few noticeable differences, but both approaches have their pros and cons. The first approach does not permit the use of actual suffix during generative training. It gives only the category of the word that the suffix belonged to. This affects the tagging of surrounding words as some of these words might require the actual suffix for disambiguation(for example, NST occasionally requires noun suffixes). Method 2 does not give category information again causing similar problems.

3 Why HMM?

HMM is a commonly used generative stochastic method regularly used in NL, Speech and Image Processing domains. The allure of HMM is its malleability and the ability to perform well if trained on a data closely resembling the test data. By malleability we mean the ability to modify a model. HMMs are very simple stochastic models and present themselves with ease to modifications. The various uses to which HMM has been put and their versatility is clearly visible in (Vergyri et al., 2004; Duh and Kirchoff, 2004; Duh, 2004; Brants, 2000; Connell, 1996; Rabiner, 1989; Fraser and Dimitriadis, 1994). (Vergyri et al., 2004; Duh and Kirchoff, 2004; Duh, 2004) show that an HMM can be effectively modified to brilliant results. TNT (Brants, 2000) is a very effective POS tagger for English and German with accuracy and speed matching the best systems currently available in the world. The applications to Speech, OCR and time series forcasting are presented in (Connell, 1996; Rabiner, 1989; Fraser and Dimitriadis, 1994). This gives enough ground to consider HMM for a possible candidate for the task of POS tagging Indian Languages using morphological features.

3.1 Discriminative Vs Generative Debate

As mentioned above, HMM is a generative stochastic model. Generative models learn a joint probability $p(x, y)$, where x is the observation and y is the label, and use the Bayes rule to compute $p(y|x)$. This is done by modelling $p(x|y)$ to make the prediction choosing the most likely label y. Discriminative models on the other hand learn the $p(y|x)$ directly from the input. The reasons cited for the more popular use of Discriminative models is ”One should solve the problem(computing $p(y|x)$ directly and never solve more general problem(modelling $p(x|y)$) as an intermediate step.”

There is a also a debate on how much modification can an HMM undergo. As mentioned above an HMM consists of two parameters $p(x|y)$ and $p(y)$, for including any restrictions in the form of observations, the number of parameters of the form $p(x|y)$ would increase thereby making the system rely more and more on the prior $p(y)$. This fact results in the unusability to HMMs for creation of complex models.

Be this as it may, Generative models have some advantages over Discriminative models in restricted cases. Some restrictions on the sort of distributions the generative model learns have been shown to improve the accuracy of classification over and above that of discriminatory classifiers. Here, the intuition is that knowledge restricts the size of the hypothesis space leading to better performance. Whereas, Discriminative methods do not allow any prior knowledge to be included apart from features. The importance of these feature cannot be pre-defined and is learned directly. Generative classifiers are a natural way to include domain knowledge,leading some researchers to propose a hybrid of the two(Tong and Koller, 2000). Another advantage of HMMs or generative models is that they perform better than Discriminative models with less training data and when the training data closely resembles the test data(Ng and Jordan, 2002).

4 Standard HMM

Hidden Markov Models (Rabiner, 1989) are simple three tuple models described as $\lambda(\Pi, A, B)$, where,

- Π = Initial Probabilities
- A = Transition Probabilities
- B = Emission Probabilities

For a given input sequence W=(w_1,w_2,. . .,w_n) we wish to determine a tag sequence T=(t_1,t_2,. . .,t_n) such that $P(W,T)$ is maximized. This probability term when broken down using chain rule results in a term implausible to compute.

$$P(W,T) = \Pi_i^N \left[P(w_i|t_{1,i}, w_{1,i-1})P(t_i|t_{1,i-1}, w_{1,i-1})\right]$$

This term is restricted by HMM using two simplifying assumptions:

- Word w_i depends only on the current tag(lexical independence).

- Tag t_i depends on previous K tags(Markov Property).

This results in a much more tractable form of the term $P(W,T)$. Thus, for inferencing with HMM, we primarily try to maximize,

$$P(W,T) = \Pi_i^N \left[P(w_i|t_i)P(t_i|t_{i-1},t_{i-2})\right] \quad (1)$$

Where,

- W is the word sequence
- T is the tag sequence
- w_i is the word at i^{th} position
- t_i is the tag at i^{th} position
- N is the length of the sequence

5 Exploded Input Model

The HMM remains the same as the standard HMM as all the required changes are made to the training and testing data at a pre-processing stage explained in the section 2. The approach makes use of simple splitting of words to lengthen the input to HMM by providing the base word and the suffix as separate observations. For a given sentence (w_1,w_2,. . .,w_n), we get a sequence of (r,s) pair for each inflected word resulting in a sequence of 2n length in the worst case of every word being inflected. The new input sequence for our model is thus, (r_1,s_1,r_2,s_2,. . .,r_n,s_n). The model is modified only in the input and output symbol set. The input set S is replaced by S_E and the output set T is replaced by T_E where

- S_E = $R \cup M$; R is the stem set and M is the set of suffixes
- T_E = $T \cup T_s$; T_s is the set of suffix tags and T is the Tag set

This approach leads to good accuracy for Hindi without resorting to detailed morphology analysis of input which would be required in the case of (Singh et al., 2006).

6 Evaluation

The corpus used for the training and testing purposes contains 66900 words. This data was ‘exploded’ resulting in a new corpus of 81751 tokens which was divided into 80% and 20% parts. The test set contains 13500 words which resulted in an exploded test set of 16000 tokens(stem and suffix tokens). The accuracy is calculated after imploding the output considering the assigned tag of the stem as the correct tag. This data was sourced from various domains including news, tourism and fiction. The tagset is the Indian Language tagset developed by IL-ILMT consortium. The following sections report the result after a four-fold cross-validation. This setup was used to evaluate a standard HMM as well as the Exploded Input-HMM (EIHMM). The implementations were developed in-house.

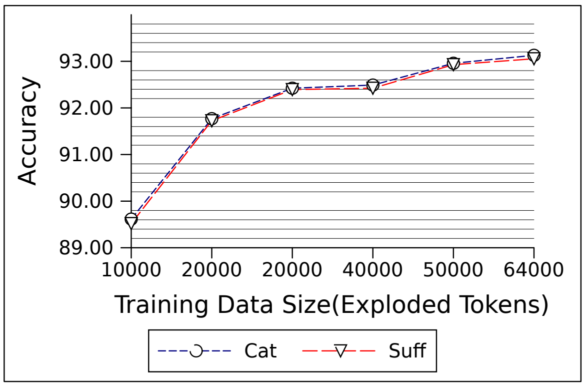

Figure 1: Training Curves for both EIHMM methods

	HMM	EI-HMM CatTags	EI-HMM SuffTags
Accuracy	83.26	**93.12**	93.05

Table 1: Comparison between HMM and EI-HMM

7 Results

The comparison of the results for standard HMM and the two model variations presented in section 2.2 are presented in Table 1. Figure 1 presents the training curve for both the methods. As expected, there is a regular increase in accuracy as the training corpus increases. But, the major advantage of these methods is the significant accuracy gain over plain HMM.

Per POS accuracy charts for both the methods in comparison to standard HMM results are shown in Figure 2 and Figure 3 respectively. It is clear from these graphs that the performance of Exploding Input HMM far outperforms standard HMM. Significant improvements are seen in case of inflected categories such as Verbs, Verb

Auxiliaries, Adjectives and, oddly, Ordinal numbers, Cardinal numbers and Quantifier. Contrary to expectations Noun(NN, NNC) accuracy does not pick up a lot. This effect is traced back to the fact that most rare nouns usually occur in their root forms. There are cases of unknowns such as मोम्बत्तियां(candles), where the suffix इयां helped disambiguate the word. But, such cases are very rare. It is hoped that as the number of unknowns and specifically number of inflected nouns in unknowns increase, the effect would be more prominently visible in noun accuracies. Currently only 11% of the words were unknown and less than 3% were found to be inflected. The number of unknowns might increase if a model is subjected to test data which is not of the same domain as the training data.

We see a significant increase in Verb and VAUX accuracy. This is due to the highly inflective verb morphology of Hindi. A common error made by HMM in Verb Group was to tag some main verbs(VMs) as VAUX or vice-versa. HMM regularly makes an error when dealing with copula verb forms (है, था etc.), tagging them as VAUX. This is because these forms occur more frequently as VAUX then as VMs. That there are usually three or more forms (था, थी, थे) does not help the case. Stemming reduces this form to (थ), distributing the probability of (थ) forms more evenly accross VM and VAUX. Stemming also helps identifying verbs in inflected forms which were not seen in training data. This is a common phenomenon as verbs inflect for Gender, Number, Person, Tense, Aspect or Mood. This means that the same verb or verb auxiliary might be seen in a different form. This makes the case stronger for utilizing stemming in case of verbs and as seen in Figures 2 and 3, it delivers the results too.

Improvements in NNP and NNPC were contrary to expected results. We were able to trace the reason for this increase to the transition probability distributions. Standard HMM tagger tags most NNPs and NNPCs as NNs. This is because words occuring as NNPs are usually unknowns and they are as likely to be followed by case-markers (NSTs) as regular Nouns. This makes them a good candidate for Noun category based on context. Thus, while maximizing the product term for HMM a NN-NST transition was chosen more often than a NNP-NST transition. This was a slight but regular imbalance that plagued HMM. The cure to this came unexpectedly with EIHMM where NN-NST transition probability is lowered as some of the weight is taken by SNN-NST or S(suffix)-NST probability. Whereas, NNP-NST probability distribution remains largely the same. This resulted in lower errors in case of NNPs.

QC does not seem very important class and truely the number QCs is small compared to VMs and NNPs. But, the improvements in their accuracies demonstrate the ability of the modified models quite convincingly. The words in QO can be all characters(such as पांचवां(fifth)) or can be a combination of digits and characters (5वां(5th)). In the second form almost all words are unknowns and its only the suffix that identifies them. There can be heuristics to handle such forms but in the current setup they are not necessary.

8 Conclusion

The over all performance of this approach is better than a simple stochastic method. But, it cannot hold a candle to methods using detailed morphological analysis and linguistic resources. The results presented may not be very impressive if compared to methods similar to one presented in (Singh et al., 2006), but, they prove that a simple stochastic method can be easily modified and used for improving performance by harnessing morphology in the simplest manner possible. In this paper, our aim was to demonstrate a method which can give good performance without relying on extensive linguistic knowledge.

The methods presented can be improved further by restricting stemming of closed category words so as to reduce unwanted stemming induced errors. Also, for closed category words the states of the HMM can be restricted as demonstrated in (Dandapat et al., 2004) by learning a smaller set of possible states from the training corpus. Similarly, efforts can be made to learn possible suffixes and their paradigms using methods similar to (Goldsmith, 2001).

9 Acknowledgement

The support of the Ministry of Information Technology, India, is gratefully acknowledged. The authors would also like to acknowledge the help and support from Ms. Smriti Singh, Ms. Vinaya and Mr. Rajendra Tripathi.

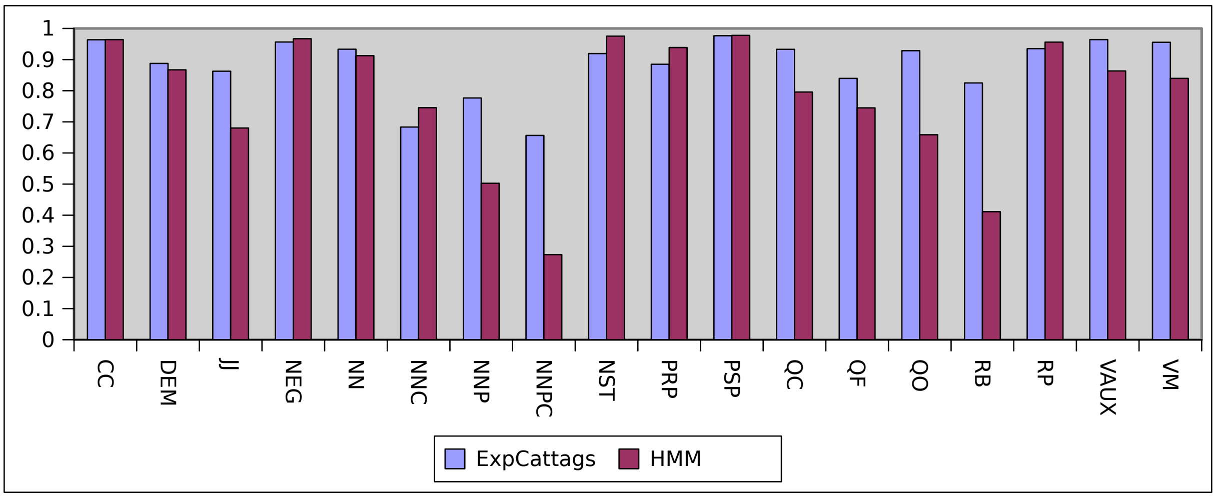

Figure 2: Comparison of Per-POS accuracy of HMM and EI-HMM with Naive Stemming- Category Tags

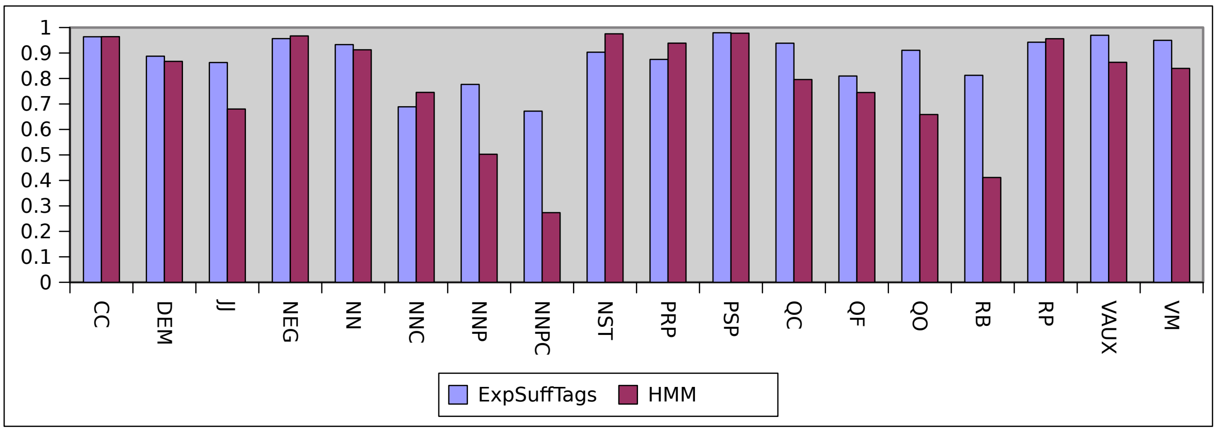

Figure 3: Comparison of Per-POS accuracy of HMM and EI-HMM with Naive Stemming- Suffix Tags

References

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing – A Paninian Perspective*. Prentice-Hall India.

Ezra Black, Fred Jelinek, John Lafferty, Robert Mercer, and Salim Roukos. 1992. Decision tree models applied to the labeling of text with parts-of-speech. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 117–121, Morristown, NJ, USA. Association for Computational Linguistics.

T. Brants. 2000. Tnt – a statistical part-of-speech tagger.

Eric Brill. 1992. A simple rule based part of speech tagger. *Proceedings of the DARPA Speech and Natural Language Workshop*.

S. Connell. 1996. A comparison of hidden markov model features for the recognition of cursive handwriting, May. Master's Thesis, Dept. of Computer Science, Michigan State University.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*.

Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. 2004. A hybrid model for part-of-speech tagging and its application to bengali. In *International Conference on Computational Intelligence*, pages 169–172.

Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. 2007. Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 221–224, Prague, Czech Republic, June. Association for Computational Linguistics.

Kevin Duh and Katarin Kirchoff. 2004. Pos tagging of dialectal arabic: A minimally supervised approach.

Kevin Duh. 2004. Jointly labeling multiple sequences: A factorial hmm approach.

Andrew M. Fraser and Alexis Dimitriadis. 1994. Forecasting probability densities by using hidden markov models with mixed states. In Andreas S. Weigend and Neil A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley.

R. Garside and N. Smith. 1997. A hybrid grammatical tagger: Claws4. In *R. Garside, G. Leech, A. McEnery (eds.) Corpus annotation: Linguistic information from computer text corpora*, pages 102–121. Longman.

John A. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

J. Hajic, P. Krbec, P. Kveton, K. Oliva, and V. Petkevic. 2001. A case study in czech tagging. *In Proceedings of the 39th Annual Meeting of the ACL 2001*.

Kumar N., Anikel Dalal, Uma Sawant, and Sandeep Shelke. 2006. Hindi part-of-speech tagging and chunking : A maximum entropy approach. *NLPAI Machine Learning Contest*.

A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes.

K. Oflazer and I. Kuruoz. 1994. Tagging and morphological disambiguation of turkish text. *In Proceedings of the 4 ACL Conference on Applied Natural Language Processing Conference 1994*.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

A. Ratnaparakhi. 1996. A maximum entropy part-of-speech tagger. *EMNLP*.

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 ICON 2003*.

Christer Samuelsson, Lucent Technologies, and Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 246–253. ACL.

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. 2006. Morphological richness offsets resource demand – experiences in constructing a pos tagger for hindi. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 779–786, Sydney, Australia, July. Association for Computational Linguistics.

Y. Tlili-Guiassa. 2006. Hybrid method for tagging arabic text. *Journal of Computer Science 2 (3): 245-248, 2006*.

Simon Tong and Daphne Koller. 2000. Restricted bayes optimal classifiers. In *AAAI/IAAI*, pages 658–664.

K. Uchimoto, S. Sekine, and H. Isahara. 2001. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. *In Proceedings of the Conference on EMNLP 2001*.

Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. 2004. Morphology-based language modeling for arabic speech recognition.