Proceedings of the IJCAI – 2007

Workshop On

Shallow Parsing for South Asian Languages

(SPSAL-2007)

Hyderabad, India
http://shiva.iiit.ac.in/SPSAL2007/

8 January 2007

Organizers

Rajeev Sangal,
Sushma Bendre,
Dipti Misra Sharma,
Prashanth Reddy Mannem

Table of Contents

Preface

In the summer of 2006, NLP Association of India (NLPAI) and IIIT-Hyderabad conducted a machine learning contest on Part Of Speech (POS) tagging and chunking for Indian languages. The contest was successful in bringing together students and researchers from all over India to develop POS taggers and chunkers for Indian languages. A workshop was held along with National Workshop on Artificial Intelligence, 2006 in Mumbai, India where the participating teams presented their approaches.

NLPAI ML Contest 2006 was the first-of-the-kind contest held for Indian languages. The contest organizers provided a platform for researchers to work on a common problem by releasing annotated data in Hindi, Bengali and Telugu with the help of IIIT-Hyderabad. POS tagged data of approximately 20,000 words in Hindi, Bengali and Telugu and chunk annotated data for Hindi was released. The participants worked on developing systems for a language of their choice and a couple of teams even contributed some annotated data. However, there was no scope of comparing the approaches across languages since the teams worked on only one language.

Meanwhile, the POS and chunk tagsets were refined for Indian languages through various workshops and the need to hold a multi-lingual POS tagging and chunking contest, where the participants developed common approaches for the group of languages, grew. With this as the aim, Shallow Parsing for South Asian Languages (SPSAL) workshop was proposed to the IJCAI Workshop committee and on June 1, 2006, the proposal was accepted.

The contest was announced on June 30, 2006 and the registration was opened. Twenty teams registered by the end of registration on 1st August. On September 7th, training and development data of approximately 20,000 words and 5,000 words respectively was released to the participants. Both POS and chunk annotated data was released for Hindi, Bengali and Telugu using the IIIT-H tagset in Shakti Standard Format (SSF). Due to the lack of annotated data for other languages, the contest was only held for these three languages.

The first round of the contest required the participating teams to submit a 4 page paper on the approach used by them and the results of their approach on the development data of all the three languages. A total of 8 teams submitted papers in the first round of the contest. The program committee reviewed the papers based on the novelty of the approaches and also the results on the development data. All the teams made through to the second and final round of the contest. Un-annotated test data was released on 1st November, 2006 and the teams submitted the output of their systems on the test data of Hindi, Bengali and Telugu. The final evaluation results were announced on 7th November, 2006. Camera-ready papers were submitted on 13th November, 2006 with the final evaluation results of each team.

The proceedings of the IJCAI workshop on "Shallow Parsing for South Asian Languages" has the final report on the approaches followed by the teams along with the individual papers submitted by the teams. We hope that this is a part of a continuing series of contests for shallow parsing of South Asian languages and these contests will lead to advancement of technology of shallow parsing.

Prashanth Reddy Mannem
Dipti Misra Sharma
Sushma Bendre
Rajeev Sangal

Program Committee

Prof. Rajeev Sangal	IIIT-Hyderabad
Prof. Sushama Bendre	Birmingham University, UK University of Hyderabad, India
Prof. Pushpak Bhattacharya	IIT-Bombay
Prof. Sudeshna Sarkar	IIT-Kharagpur
Prof. Sobha L	Anna University
Dr. Dipti Misra Sharma	IIIT-Hyderabad
Dr.Vasudev Verma	IIIT-Hyderabad
Mr. Prashanth Reddy	IIIT-Hyderabad
Mr. Anil Kumar Singh	IIIT-Hyderabad
Mr. Prasad Pingali	IIIT-Hyderabad

Program Schedule 8 Jan 2007

9.15 - 9.30 – Opening

09.30 - 10.30 **Invited Talk by Yoshimasa Tsuruoka**

10.30 - 11.00 Sandipan Dandapat
Part of Speech Tagging and Chunking with Maximum Entropy Model

11.00 - 11.15 Tea break

11.15 - 11.45 G. M. Ravi Sastry, Sourish Chaudhuri and P. Nagender Reddy
A HMM Based Part-Of-Speech Tagger and Statistical Chunker for 3 Indian Languages

11.45 - 12.15 Sathish Chandra Pammi and Kishore Prahallad
POS Tagging and Chunking Using Decision Forests

12.15 - 12.45 Himanshu Agrawal
POS Tagging and Chunking for Indian Languages

12.45 - 13.45 Lunch break

13.45 - 14.45 **Invited Talk by Sabine Buchholz**

14.45 - 15.15 Avinesh PVS and Karthik G
Part-Of-Speech Tagging and Chunking Using Conditional Random Fields and Transformation Based Learning,

15.15 - 15.30 Tea break

15.30 - 16.00 Delip Rao and David Yarowsky
Part Of Speech tagging and Shallow Parsing for Indian Languages

16.00 - 16.30 Pattabhi R K Rao T, Vijay Sundar Ram R, Vijayakrishna R and Sobha L
A Text Chunker and Hybrid POS Tagger for Indian Languages

16.30 - 17.00 Asif Ekbal , Samiran Mandal and Sivaji Bandyopadhyay
POS Tagging Using HMM and Rule Based Chunking

Introduction to the Shallow Parsing Contest for South Asian Languages

Akshar Bharathi and Prashanth R. Mannem
Language Technologies Research Center
International Institute of Information Technology
Hyderabad, India 500032
prashanth@research.iiit.ac.in

Abstract

As part of the IJCAI workshop on "Shallow Parsing for South Asian Languages", a contest was held in which the participants trained and tested their shallow parsing systems for Hindi, Bengali and Telugu. This paper gives the complete account of the contest in terms of how the data for the three languages was released, the performances of the participating systems and an overview of the approaches followed for POS tagging and chunking. We finally give an analysis of the systems which gives insights to directions for future research on shallow parsing for South Asian languages.

Acknowledgement

Many thanks to Prof. Sushma Bendre and Prof. Rajeev Sangal, the organizers of the workshop, for the discussions, review and the constant support they provided in making this contest possible. We express our immense gratitude to Dr. Dipti Misra Sharma for her invaluable inputs and guidance all along. Special thanks to the Language Technologies Research Center at the International Institute of Information Technology, Hyderabad for making the annotated data for all the three languages, Hindi, Bengali and Telugu available. Credit also goes to Dr. Soma Paul and Ganga Bhavani for Bengali and Telugu annotated data respectively.

1 Introduction

NLP research around the world has taken giant leaps in the last decade with the advent of efficient machine learning algorithms and the creation of large annotated corpora for various languages. However, NLP research in Indian languages has mainly focused on the development of rule based systems due to the lack of annotated corpora. These rule based systems do not perform well for POS tagging and chunking for Indian languages.

Statistical NLP research in Indian languages can only be given a push by the creation of annotated corpus for Indian languages. What better way to start than POS tagging and chunking? It has been shown time and again that for any language processing tool, Part Of Speech tagging and chunking are the fundamental processing steps.

Part-of-speech tagging (POS tagging or POS) is the process of marking up each word in a text with a corresponding part of speech like noun, verb, adjective ,adverb etc ... , based both on its definition, as well as its context. The number of part of speech tags in a tagger may vary depending on the information one wants to capture. For example, in the sentence below, the POS tags are appended at the end of each word with an '_'.

Children_NNS are_VBP watching_VBG some_DT programmes_NNS on_IN telivision_NN

Chunking consists of dividing a text in syntactically correlated parts of words. Common chunk tags are NP, VP, PP. The chunk tags may also vary depending on the information one wants to capture.

[NP Children_NNS] (VP are_VBP watching_VBG) [NP some_DT programmes_NNS] [PP on_IN telivision_NN]

In this paper, we give a complete account of the contest along with approaches followed by the participating teams and insights into their work. Section 2 in this paper gives an idea of the previous research in POS tagging and chunking in Indian languages. Section 3 lists the task definition and the data format. Section 4 gives a performance analysis of the various approaches adopted by the participant for POS tagging and chunking. Finally, Section 5 gives possible directions for future research.

2 Previous Research

For English, there are many POS taggers , employing machine learning techniques such as Hidden Markov Models (Brants, 2000), transformation based error driven learning (Brill, 1995), decision trees (Black, 1992), maximum entropy methods (Ratnaparkhi, 1996), conditional random fields (Laffertey et al., 2001). Some of the techniques proposed for chunking in English are based on Support vector machines (Kudoh et al., 2001), Winnow (Zhang et al., 2002) . The POS taggers reach anywhere between 92-97 % accuracy and chunkers have reached approximately 94 % accuracy. However, these accuracies are aided by the availability of large annotated corpus for English.

As mentioned above, due to the lack of annotated corpora, previous research in POS tagging and chunking in Indian languages has mainly focused on rule based systems utilizing the morphological analysis of word-forms. A. Bharati et al. (1995) in their work on computational Paninian POS parser, described a technique where POS tagging is implicit and is merged with parsing phase. More recently, Smriti et al. (2006) proposed a pos tagger for Hindi which uses an annotated corpus (15,562 words collected from BBC Hindi News site), exhaustive morphological analysis backed by high-coverage lexicon and a decision tree based learning algorithm(CN2). They reach an accuracy of 93.45% for Hindi with a tagset of 23 POS tags.

For Bengali, (Sandipan et al., 2004) developed a corpus based semi-supervised learning algorithm for POS tagging based on HMMs. Their system uses a small tagged corpus (500 sentences) and a large unannotated corpus along with a Bengali morphological analyzer. When tested on a corpus of 100 sentences (1003 words), their system obtained an accuracy of 95%.

A. Singh et al. (2005) proposed a HMM based chunker for Hindi with an accuracy of 91.7 %. They used HMMs trained on four tag scheme (STRT, CNT, STP, STRT_STP) with POS tag information and converted it into two tag (STRT, CNT) scheme while testing for chunk boundary identification. They however used a rule based system for chunk label identification. Annotated data of 150,000 words was used for training and the chunker was tested on 20,000 words with POS tags which were manually annotated.

To the best of our knowledge, these are the reported works on POS tagging and chunking for Indian languages till the NLPAI Machine Learning Contest (2006) was held in the summer of 2006. For the contest, participants had to train on a set of training data for a chosen language provided by the contest organizers. The systems, thus trained, were to automatically mark POS and chunk information on the test data of the chosen language. Chunk annotated data wasn't released for Bengali and Telugu. Sandipan and Sudeshna (2006) achieved an accuracy of 84.34 % for Bengali POS tagging using semi-supervised learning combined with a Bengali morphological analyzer. A. Dalal et al. (2006) achieved accuracies of 82.22 % and 82.4 % for Hindi POS tagging and chunking respectively using maximum entropy models. Karthik et al. (2006) got 81.59 % accuracy for Telugu POS tagging using HMMs. Sivaji et al. (2006) came up with a rule based chunker for Bengali which gave an accuracy of 81.64 %. The training data for all the three languages contained approximately 20,000 words and the testing data had approximately 5000 words.

The experiences from organizing the NLPAI ML Contest prompted us to hold a contest for Shallow Parsing (POS tagging and chunking) where the participants will have to develop systems for POS tagging and chunking across Indian languages using the same learning technique. The next section formally defines the task and the data released as part of the contest.

Shakti Standard Format		
1	((	NP
1.1	Children	NNS
	))	
2	((	VG
2.1	are	VBP
2.2	watching	VBG
	))	
3	((	NP
3.1	some	DT
3.2	programmes	NNS
	))	
4	((	PP
4.1	on	IN
4.2	((	NP
4.2.1	television	NN
	))	

Table 1: An example sentence in the SSF Format.

3 Task definition and Data

The contest was conducted for three languages Hindi, Bengali and Telugu. It could not be conducted for other languages due to the lack of annotated data. Participants were provided with 20,000 words of training data, 5,000 words of development data and 5000 words of test data in all the three three languages.

Hindi and Bengali data was released in UTF-8 encoding whereas Telugu data was released in wx notation. While the training data contained three fields (address, word/group, tag), testing data only had 2 fields (address and word). Participants' systems predicted the tags and chunk groups for the words in each sentence. The evaluation metric for POS tagging is POS-Tagging-Accuracy which is the percentage of tokens with correct POS tag. The evaluation metric for chunking is Chunking-Accuracy which is the percentage of tokens with correct chunk boundary (denoted by 'B' for begin of a chunk, 'I' for inside a chunk and 'O' for outside a chunk) and the correct chunk tag.

The data was released in Shakti Standard Format (A. Bharati et al., 2005). SSF Format is used to represent the different kinds of linguistic analysis , as well as different levels of analysis (both constituent level analysis and feature-structure level analysis). Table 1 gives the SSF Format for the example sentence in Section 1. As shown in the figure, there are four columns in SSF format, separated by a tab. Each line represents a word or a group (except for lines with '))' which only indicate the end of a group). For each group, the symbol used is '((' . Each line has 4 fields. The word or group is in the second field, with part of speech tag in the third field. The first field is mainly for human readability, and stores the tree or forest address of each word or group. The fourth field holds the feature information. Morphological information, grammatical roles, semantic information etc.. can be listed as features in the fourth field. However, for the current task, the fourth field remains empty. Sentences in the corpus are separated by a blank line.

All the languages were annotated with the IIIT POS and chunk tagset. The tagset has a total of 24 POS tags namely CC,VRB, NNP, QFNUM, VJJ, RP, QF, JJ, PRP, NN, VFM, NNPC, INTF, RB, VAUX, NNC, PREP, SYM, VNN, QW, NEG, UH, NLOC. Bengali has a few additional tags QT, INF, JJC and RBC. The chunk tagset has 7 labels namely CCP, RBP, JJP, VG, BLK, NP and NEGP. The reader is requested to refer to A. Bharati (2006a) and A. Bharati (2006b) for additional information regarding the tagsets.

4 Approaches

The training data provided to the participants is small (20,000 words approx. in each language) considering the number of tags and the morphological richness of Indian languages. Table 2 shows the number of words in training and test data for all the languages. Note the percentage of unseen words in the test data for each language. Approximately, one third of the test words are unseen in Hindi and Bengali whereas it is alarmingly high (almost half) in Telugu. It is due to the agglutinative nature of the language where a root can combine with other word or suffix to form a larger word-form. This accounts for data sparsity for statistical techniques and is a major problem. It rightfully deserved the maximum attention from the participants in their approaches.

Eight teams submitted their systems for evaluation. Most teams used different approaches for the two tasks, POS tagging and chunking. Following

Lang.	*Train*	*Test*	*Unseen*	*Percent*
Hindi	21470	4924	1532	31.11%
Bengali	20397	5225	1817	34.77%
Telugu	21415	5193	2454	47.26%

Table 2: Number of words in Train and Test data along with the number of unseen words in the Test data with percentages.

section has the analysis of the approaches used by various teams.

4.1 POS Tagging

Participants tried out a wide range of learning techniques starting from Naive Bayes, HMMs, Decision Trees to Maximum Entropy Model and Conditional Random Fields. None of the the teams tried rule based approaches for POS tagging. Most teams tried out a number of learning techniques for comparison before choosing the best. Table 4 lists the POS tagging accuracies achieved by the teams for Bengali, Hindi and Telugu.

Team names used in Table 4 and subsequent tables are the first three characters of the first author of the respective participating team.

(Pattabhi et al., 2007) → *Pat, (Satish and Kishore, 2007)* → Sat , *(Rao and Yarowsky, 2007)* → Rao, *(Himanshu, 2007)* → Him , *(Asif et al., 2007)* → Asi , *(Sandipan, 2007)* → San , *(Ravi et al., 2007)* → Rav , *(Avinesh and Karthik, 2007)* → Avi

Team	*Bengali*	*Hindi*	*Telugu*	**Avg**
Pat	72.17	76.34	53.17	67.22
Sat	60.08	69.35	77.20	68.88
Rao	69.07	73.90	72.38	71.78
Him	76.00	62.35	77.16	71.84
Asi	73.17	76.87	67.69	72.57
San	77.61	75.69	74.47	75.92
Rav	74.58	78.35	75.27	76.06
Avi	76.08	78.66	77.37	77.37
Avg	72.35	73.93	71.83	72.71

Table 4: POS tagging accuracies of the teams for Hindi, Bengali and Telugu.

Previous research has shown that HMM based POS tagging is simple to implement and efficient too for a variety of languages. Four out of the eight teams (Pattabhi et al., 2007; Ravi et al., 2007; Rao and Yarowsky, 2007; Asif et al., 2007) used HMMs for POS tagging. However, there are variations in the way these teams handled the data sparsity issues. Instead of smoothing, Pattabhi et al. (2007) used linguistic rules to tag words for which the emission or transition probabilities are low or zero. Asif et al. (2007) too didn't use smoothing. For unknown words, the emission probability is replaced by the probability of the suffix for a specific POS tag. Additionally, they also used Named Entity Recogniser (trained on the same data) and a lexicon with basic POS information (like noun, verb, adjective etc ...). This lexicon is only used for Bengali. Rao and Yarowsky (2007) tried four systems, Naive Bayes Classifier, A suffix based Naive Bayes Classifier which uses suffix info for unseen words and HMM tools Tnt (Brants, 2000) and QTag (Tufis and Mason, 1998). They found TnT to be performing the best among the four approaches. Ravi et al. (2007) too used TnT for POS tagging. However, there is a considerable difference in the POS tagging accuracies of Ravi et al. (2007) and Rao and Yarowsky (2007) even though both used TnT.

Sandipan (2007) used Maximum Entropy Model for POS tagging with contextual features covering a word window of 1 and suffix (with length ≤ 4) and prefix (with length ≤ 4) information. For Bengali, his system is assisted by a morphological analyser which is used to restrict the output of the tagger to the set of tags given by the analyser. This system performs the best for Bengali POS tagging.

Himanshu (2007) developed a CRF based approach with a feature set including a word window of 2, suffixes (last chars ≤ 4), word length and flag indicating special symbols. To handle data sparsity, he built a *knowledge database* by picking word & tag pairs which are tagged with high confidence by the initial model over a raw corpus of 150,000 words. The output of the tagger for a word is then restricted to the set of tags listed in the knowledge database and the training data. This is similar to work done for Bengali by Sandipan (2007) with the difference being that instead of a manually built morphological analyzer, a machine built knowledge database is used to restrict the outcomes of the tagger. This system worked well for Bengli and Telugu. However, it didn't perform well for Hindi.

POS Tagging			
$Team$	$LearningAlgo$	$LanguageResources$	$Features$
Pat	HMM and rule based tagging.	-	Lexical features
Rav	HMM based tagging using TnT.	-	Lexical features
Rao	HMM based tagging using TnT.	-	Lexical features
Avi	CRF learning followed by TBL	Morphological Analysers	word window of 6, root word, suffix, prefix, all possible tags for the word length of the word
Asi	Hybrid HMMs with $P(w_i\|t_i)$ replaced by $P(w_i\|t_i, t_{i-1})$ supplemented by suffix info. Also uses NER.	Bengali lexicon annotated with basic POS tags	Lexical features
Him	Conditional Random Fields	Unannotated corpus	word window of 2 last chars ≤ 4, word length
San	Maximum entropy Model	Morphological Analyser for Bengali	word window of 1 $\|\text{suffix}\| \leq 4$ and $\|\text{prefix}\| \leq 4$
San	Decision Forests	-	features based on Syllables, Phonemes and Onset-Vowel-Code

Table 3: Summary of the approaches followed by the participants for POS tagging.

Avinesh and Karthik (2007) proposed a two level training approach where Transformation Based Learning (TBL) was applied on top of a CRF based model. The CRF model used exhaustive contextual information with a window size of 6, 6 and 4 for Hindi, Bengali and Telugu respectively, morphological information like root word, all possible categories, suffixes and prefixes. They even used a flag to indicate words with length $\leq$ 3 to take into account the large number of functional words in Indian languages. Their system got the highest POS tagging accuracy for Hindi and Telugu.

Satish and Kishore (2007) used Decision Forests for POS tagging with some innovative features based on subwords (syllables, phonemes and Onset-Vocal-Code for syllables) for Indian Languages. The novel features they used are interesting and quite intuitive since in Indian languages, the subwords are an important source of information to determine the category of the word. For Telugu, the system worked with reasonable accuracy whereas for Hindi and Bengali the results don't look promising. It would be interesting to know the performance of CRF based systems using these novel subword based features rather than using *last-n* characters as suffixes.

The average POS tagging accuracy of all the systems for Bengali, Hindi and Telugu are 72.35 % , 73.93 % and 71.83 % respectively.

4.2 Chunking

Chunking is comparatively easier for Indian languages than POS tagging. The teams used the output of their own POS tagger as input to the chunker. So, the noise from the earlier tagging stage effected the chunking performance. Table 6 lists the chunking performance of all the teams for Bengali, Hindi and Telugu.

For chunking, an even wider variety of learning techniques were used as compared to POS Tagging. It is interesting to note that none of the teams used HMMs for chunking. All the teams divided the chunking task into chunk boundary identification and chunk labeling. A rule based system was proposed by Asif et al. (2007). However, no details of the linguistic rules used in the system are given. This rule based system worked well for Bengali and Hindi. Satish and Kishore (2007) used the same De-

Chunking			
$Team$	$LearningAlgo$	$LanguageResources$	$Features$
Pat	Transformational Based Learning..	-	Lexical features
Rav	Learning Chunk Pattern Templates	-	Lexical features
Rao	Naive Bayes Classifier	-	Lexical features
Avi	HMMs for chunk boundary identification CRFs for chunk labeling	-	Word & POS combinations
Asi	Rule Based System	-	-
Him	Conditional Random Fields	-	POS and chunk tags for a word window of 2
San	Maximum entropy Model	-	POS and chunk tags for a word window of 2
San	Decision Forests	-	POS tags for a word window of 2

Table 5: Summary of the approches followed by the participants for chunking.

$Team$	$Bengali$	$Hindi$	$Telugu$	**Average**
Pat	65.28	73.80	50.38	63.15
Rao	65.47	70.67	61.41	65.85
Asi	80.63	71.65	53.15	68.47
Rav	67.52	69.98	68.32	68.60
Sat	70.99	69.92	74.74	71.88
San	80.59	74.92	68.59	74.70
Him	82.72	72.87	79.13	78.24
Avi	82.74	80.97	79.15	80.95
Avg	74.49	73.09	66.85	71.48

Table 6: Chunking accuracies of the teams for Hindi, Bengali and Telugu.

cision Forests for chunking too with features from words within a window size of 2. They used the two tag scheme (A. Singh et al., 2005) for training. Pattabhi et al. (2007) used the TBL although they used HMM based technique for POS tagging. Sandipan (2007) used the Maximum Entropy Model for POS tagging and the same he used for chunking. As features, he used only contextual POS tags and not the context words.

Ravi et al. (2007) proposed learning chunk patter templates based on the 4 parameters *Chunk-POS combinations*, *Chunk Structure*, *Chunk Label-POStag of the following word* and *POS tag of the following word*. They use CKY algorithm to get the best Chunk sequence for a sentence. Rao and Yarowsky (2007) used CRF training with 4 tag schema (A. Singh et al., 2005). They took a special note of the punctuations occurring in the sentence which adds noise and act as a deterrent for the learning of clear syntactic pattern. So, they purge them out of the chunk and create a dummy chunk with label *BLK*. They are later added to the previous chunk as a post processing step. This improved the accuracy considerably. However, they submitted a Naive Bayes Classifier based chunker for the final evaluation.

Avinesh and Karthik (2007) used HMMs for chunk boundary identification and CRFs for chunk labeling. This is the only team which used different learning techniques for the two stages in chunking. They got the best results for chunking for all the three languages. Himanshu (2007) divided the task into three stages. In the first stage BL (Boundary-Label) prediction is done. In the second stage the chunk boundaries are predicted using the chunk label information from the first stage. In the final stage, the chunk labels are re-predicted. This is a composite chunking task followed by a two stage chunking. His system came as a close second to Avinesh and Karthik (2007)'s systems for Bengali and Telugu.

The average chunk accuracy of all the systems for Hindi, Bengali and Telugu are 73.09 % , 74.49 % and 66.85 % respectively. This also supports the fact that Telugu is richer and hence harder to learn than Hindi and Bengali. Table 6 also shows that systems which have been trained with richer features (Avinesh and Karthik, 2007; Himanshu, 2007; Sandi-

pan, 2007) performed better than the ones using only lexical information (Ravi et al., 2007; Pattabhi et al., 2007; Rao and Yarowsky, 2007). The only rule based system (Asif et al., 2007) worked well for Bengali, since the rules were prepared for Bengali. The average chunk accuracies on development data with gold-standard POS tags was greater thatn 90%.

5 Conclusion and Future Work

Participants have come up with efficient POS tagging and chunking algorithms for Bengali, Hindi and Telugu with limited annotated data. Various strategies have been proposed to counter the data sparsity problem. Among these, using morphological information proves to be the most effective. In case of the unavailability of morphological analyzers, suffix and prefix information can also be used with good results. It should also be noted that TnT which doesn't use any of the above information performs well enough with better smoothing techniques. Highest POS tagging accuracies of 77.61 % , 78.66 % and 77.37 % were achieved for Bengali, Hindi and Telugu respectively. Similarly, highest chunk accuracies of 82.74 %, 80.97 % and 79.15% were achieved for Bengali, Hindi and Telugu respectively. This was possible with the use of morphologically rich features coupled with efficient learning techniques. However, there is a lot of scope for improvement in the way unseen words are handled. Due to the agglutinative nature of Indian languages, morphological analysers play a greater role in POS taggging as is evident from the results. Developing automatic morph analysers for Indian languages is one area where future research can look into.

References

Himanshu Agrawal 2007. POS Tagging and Chunking for Indian Languages. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Sivaji Bandyopadhyay, Asif Ekbal, Debasish Halder. 2006. HMM based POS Tagger and Rule-based Chunker for Bengali. In *Proceedings of NLPAI Machine Learning Workshop on "Part Of Speech and Chunking for Indian Languages"-2006*,

A. Bharati. 2006a. *Part-Of-Speech tagger for Indian languages*, http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf .

A. Bharati. 2006b. *Tagging scheme for Chunking*, http://shiva.iiit.ac.in/SPSAL2007/chunking-guidelines-1.2.pdf .

A. Bharati , R. Sangal and Dipti M Sharma. 2005. *Shakti Analyser: SSF Representation*, http://shiva.iiit.ac.in/SPSAL2007/ssf-analysis-representation.pdf .

A. Bharati , V. Chaitanya and R. Sangal. 1995. *Natural Language Processing: A Paninian Perspective*, Prentice Hall India.

E. Black et al. 1992. Decision tree models applied to the labeling of text with parts of speech. In *Darpa Workshop on Speech and Natural Language-1992*,

E. Brill. 1995. Transformation based error driven learning and Natural Language Processing: A case study in Part Of Speech tagging. In *Computational Linguistics, 1995*,

T. Brants. 2000. TnT - A Statistical Part Of Speech Tagger. In *Proceedings of 6th Applied NLP Conference-2000*,

NLPAI Machine Learning Contest. 2006. POS tagging and chunking for Indian languages. *http://ltrc.iiit.net/nlpai_contest06/proceedings.php*,

Aniket Dalal, Kumar Nagaraj, Uma Sawant, Sandeep Shelke. 2006. Hindi Part-of-Speech Tagging and Chunking : A Maximum Entropy Approach. In *Proceedings of NLPAI Machine Learning Workshop on "Part Of Speech and Chunking for Indian Languages"-2006*,

Sandipan Dandapat, Sudeshna Sarkar, Anupam Basu. 2004. A Hybrid Model for Part-of-Speech Tagging and its Application to Bengali. In *Proceedings of International Conference on Computational Intelligence-2004*,

Sandipan Dandapat, Sudeshna Sarkar. 2006. Part of Speech Tagging for Bengali with Hidden Markov Model. In *Proceedings of NLPAI Machine Learning Workshop on "Part Of Speech and Chunking for Indian Languages"-2006*,

Sandipan Dandapat. 2007. Part Of Specch Tagging and Chunking with Maximum Entropy Model. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Asif Ekbal, Samiran Mandal and Sivaji Bandhyopadhyay. 2007. POS Tagging using HMM and Rule Based Chunking. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Karthik Kumar G, Sudheer K, Avinesh PVS. 2006. Comparative study of various Machine Learning methods For Telugu Part of Speech tagging. In *Proceedings of NLPAI Machine Learning Workshop on "Part Of Speech and Chunking for Indian Languages"-2006*,

Taku Kudoh and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proceedings of NAACL-2001*,

John Laffertey, A. McCallum and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML-2001*,

Satish C. Pammi and Kishore Prahallad. 2007. POS Tagging and Chunking Using Decision Forests. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Avinesh PVS, 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"*,

Delip Rao and David Yarowsky. 2007. Part Of Speech Tagging and Shallow Parsing of Indian Languages. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Pattabhi R K Rao, Vijay S R R, Vijaykrishna R and Sobha L. 2007. A Text Chunker and Hybrid POS Tagger for Indian Languages. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

A. Ratnaparkhi 1996. A maximum entropy Part Of Speech tagger. In *Proceedings of EMNLP-1996*,

G. M. Ravi Sastry, Sourish Chaudhari and P. Nagender Reddy. 2007. A HMM Based Part-Of-Speech Tagger and Statistical Chunker for 3 Indian Languages. In *Proceedings of IJCAI Workshop on "Shallow Parsing for South Asian Languages"*,

Akshay Singh, S M Bendre, Rajeev Sangal. 2005. HMM Based Chunker for Hindi. In *Proceedings of International Joint Conference on NLP-2005*,

Smriti Singh , Kuhoo Gupta , Manish Shrivastava and Pushpak Bhattacharya. 2006. Morphological Richness Offsets Resource Demand - Experiences in Constructing a POS Tagger for Hindi. In *Proceedings of ACL-2006*,

Dan Tufis and Oliver Mason. 1998. Tagging Romanian Texts: A Case Study for QTag, A Language Independent Probabilistic Tagger. In *Proceedings of LREC-1998*.

Tong Zhang, Fred Damerau and David Johnson. 2002. Text Chunking based on a Generalization of Winnow. In *Journal of Machine Learning Research, volume 2 (March), 2002*,

A Text Chunker and Hybrid POS Tagger for Indian Languages

Pattabhi R K Rao T, Vijay Sundar Ram R, Vijayakrishna R and Sobha L
AU-KBC Research Centre,
MIT Campus, Anna University,
Chromepet, Chennai
{pattabhi, sundar, vijayakrishna, sobha}@au-kbc.org

Abstract

Part-of-Speech (POS) tagging can be described as a task of doing automatic annotation of syntactic categories for each word in a text document. This paper presents a generic hybrid POS tagger for Indian languages. Indian languages are relatively free word order, morphologically productive and agglutinative languages. In this hybrid implementation we have used combination of statistical approach (HMM) and rule based approach. Here we have used the tagset developed by IIIT, Hyderabad consisting of 26 tags. This paper also presents a transformational-based learning (TBL) approach for text chunking. In this technique of chunking, a single base rule (or a few base rules) is provided to the system, and the other rules are learned by system itself during the training phase for reorganization of the chunks. Here we have worked for three of the Indian languages namely Hindi, Bengali and Telugu. The corpus used for training was provided by SPSAL workshop. The results obtained varied for each language, but are encouraging.

1 Introduction

Part-of-speech (POS) tagging is the task of labeling each word in a sentence with its appropriate syntactic category called part of speech. There are two factors determining the syntactic category of a word (a) the words lexical probability (e.g. without context, *bank* is more probably a noun than a verb) (b) the words contextual probability (e.g. after a noun or a pronoun, bank is more a verb than a noun, as in "I bank with HSBC "). Hence it disambiguates the part of speech of a word when it occurs in different contexts.

Chunking is the task of identifying and labeling different types of phrases such as Noun phrase (NP), Verb phrase (VG), Adjectival phrase (JJP) etc in a sentence. This is done after POS tagging.

Both POS tagging and chunking are important for all language processing activities. These two tasks are considered as important preprocessing activities. This helps in doing deep parsing of text. Also helps in developing Information extraction systems, semantic processing etc.

The paper's schema is as follows. The second section discusses about the related work in this area of research. Section three discusses hybrid POS tagger. In section four we discuss the TBL based text chunker. Section five discusses the results of POS tagger. Section six concludes the paper.

2 Related Work

2.1 Part-of-Speech Tagger

The POS taggers developed using statistical method are based on probability measures. The probabilities are calculated using unigram, bigram, trigram, and n-gram methods (Charniak, 1993). For many morphologically rich languages like Japanese and Arabic, viterbi algorithm is used for tagging and disambiguation (Kazama et. al, 2001; KHOJA, 2001). The Arabic POS tagger achieved an accuracy of around 90% when disambiguating ambiguous words. But unknown words are tagged as nouns because the tagged training corpus had 67% of nouns (KHOJA, 2001).

The rule-based taggers use lexical and context sensitive rules in getting the correct tag. Among the rule-based approaches the one that achieved the best accuracy is 97.5% (Brill, 1994). The rule-based taggers are developed for European languages.

In Indian languages a rule-based POS tagger for Tamil was developed and tested (Arulmozhi et. al 2004). There are nearly 90 lexical rules and nearly 7 context sensitive rules, which are hand crafted. The precision of the system reported was 92%. This system gives only the major tags and the sub tags are overlooked while evaluation. Tagging becomes difficult when there is no inflection in the word. When the sub tags are considered the performance is reduced.

A hybrid POS tagger for Tamil using HMM technique and a rule based system and had a precision of 97.2%. Using statistical approach alone for POS tagging has shown promising precision but low recall. When combined shows good precision as well as recall. (Arulmozhi et al, 2006). Here the rule based system is used for smoothing the HMM system. In this paper we have used hybrid approach for POS tagging.

2.2 Text Chunking

Several methods have come up for this task. Church's stochastic noun phrase tagging was one of the first, where the corpus frequencies were used to determine the noun phrase boundaries (Church, 1988). Abney did partial parsing using finite state cascades, where the finite state cascade has sequence of levels. Phrase at one level is built on the phrase at the previous level without any recursion (Abney, 1996).

The technique of Transformation Based Learning (TBL) was used by Eric Brill for the task of part-of-speech tagging (Brill, 1994), but also for prepositional phrase attachment disambiguation (Brill and Resnik, 1994), and assigning unlabeled binary-branching tree structure to sentences (Brill, 1993b). Because transformation-based learning uses pattern-action rules based on selected features of the local context, it is helpful for the values being predicted to also be encoded locally. Ramshaw and Marcus used transformation based learning for recognizing the noun chunks and other text chunking for the English language (Ramshaw, 1995). In Indian languages, TBL was used for Tamil text chunking and had a precision of 97.4% (Sobha et al, 2006).

In this paper we have used TBL method for text chunking.

3 The Hybrid POS Tagger

A Hidden Markov Model (HMM) is a five-tuple $H = (\Sigma, Q, q_0, A, B)$ where:

Σ is a finite observation alphabet;
Q is a finite set of states;
$q0 \in Q$ is the distinguished initial state;
$A : Q \times Q \rightarrow [0..1]$ is a probability distribution on state transitions:

A (q_1, q_2) is the probability of a transition to state q_2 from state q_1; and
$B : Q \times \Sigma [0..1]$ is a probability distribution on state symbol emissions :

B(q, a) is the probability of observing the symbol a when in state q.

The probability distributions A and B are estimated from the tagged training corpus. In POS tagging task the input sentence is the observed part, which is a sequence of words. Here part-of-speech tags are represented as states of the model. This states are hidden and we need to estimate state sequence for a given word sequence. Here we use Viterbi algorithm for efficiently computing the state sequence, which is most likely to generate the observed word sequence.

When statistical method like HMM is used we face the problem of word sparseness or data sparseness. In a language there are always new words, which may not be there in the training corpus however large, the training corpus is. This we term it as lexical item sparseness. In such cases we find the value of emission probability distribution, B for unknown words to be zero. But this can be handled by taking into account, only the transition probability distribution, A. But due to sparseness of data it is found in some instances that the values of A also to be zero. This may be called as structural sparseness. Whenever, both the probability distributions give value of zero, there we use the linguistic rules (context sensitive rules and lexical rules) to tag the words. Hence for this purpose a rule based system is used. This rule based system consists of context sensitive rules which can be applied across the Indian languages and lexical rules specific to dravidian languages. Hence in this system, instead of using any statistical methods for smoothing we are using linguistic rules for smoothing. It is found that statistical methods of smoothing do not yield a very significant increase in the results for part-of-speech tagging task, but whereas use of linguistic rules had yielded significant increase in the results.

It's observed that the certain tags occur in low frequency in the training corpus. We identify such tags by calculating the frequency of occurrence each tag of the tagset in the training

corpus. The tags whose frequency of occurrence in the training corpus doesn't cross the minimum threshold are taken as low frequency tags. The HMM system would not tag such tags correctly. For such tags we use the rule based system for tagging correctly. Here we have taken the threshold to be 30%. The criteria for threshold can be arrived at, by performing several experiments.

4 The Text Chunker

Transformation-based learning is a method where the different transformations or changes in the structure of the base rule are learned from the training corpus. Transformation-based learning starts with a training corpus, where the linguistic feature are tagged, a base rule for predicting the initial values of the features and rule template for the possible transformational rules. The patterns are generated from the training corpus. From the patterns the rules are learned by considering the neighboring word and the part-of-speech tags. The rules learned by the system are scored, by comparing with the tagged features of the training corpus.

The process starts with the comparison of the patterns of tagged data with the base rule. The patterns that do not fall under the base rule are grouped together. Using each pattern in the group the possible rules are generated. These rule candidates formed are scored against the trained data. The rules that cross a threshold in score are taken as rule that are to be learned.

This entire learning process is then repeated on the tagged corpus: deriving candidate rules, scoring them, and selecting one, which is greater than the threshold score. This process is iterated, to generate rules. The predictions of the model on new text are determined by beginning with the base rule and then applying the rules learned

5 Results

The hybrid POS tagger was used for three Indian languages namely Hindi, Bengali and Telugu. The results for all the three languages are shown below in Table 1. Also evaluation results of Text chunker are shown in Table 2 and Table 3. In Table 2 the evaluation results shown are when the text chunker is applied on the gold standard POS tagged input text, which means there are no errors in the POS tagger. The results shown in Table 3 are obtained when the input text to the text chunker is POS tagged text obtained by executing our hybrid POS tagger. Table 3 shows the combined results of POS tagger and Text chunker.

Language	No. Of Words	Words tagged	Correctly Tagged	Precision %	Recall %
Hindi	5677	5677	4515	79.5	79.5
Bengali	5029	5029	3742	74.4	74.4
Telugu	6098	6098	3547	58.2	58.2

Table 1. Results of the POS tagger System

Language	Total No. Of Phrases	Phrases Chunked	Correctly Chunked	Precision %	Recall %
Hindi	2710	2726	2245	82.35	82.84
Bengali	3298	3246	2954	91.0	89.57
Telugu	3550	3571	2983	83.53	84.03

Table 2. Results of the Text Chunker with gold standard POS tagged input

Language	POS Tagging Accuracy %	Chunking Accuracy %
Hindi	76.34	73.80
Bengali	72.17	65.28
Telugu	53.17	50.38

Table 3. Combined Results of the Text Chunker with POS tagged input by the system

In POS tagging, when a hybrid system is used, the disadvantages of one component is either nullified or reduced by the other component. Hence the two components rule-based tagger and

HMM based tagger of the hybrid system complement each other. In this system, as discussed in section (3) the transition probability becoming zero in the statistical tagger leaves the sentence untagged. The rule-based tagger tags such sentences. For the rule-based system, it is difficult to tag an un-inflected word. The statistical tagger already tags the un-inflected word. Hence after analyzing the results of hybrid POS tagger it can be concluded that the confidence of the correctness of the tagged output is more reliable.

The Text chunker when provided with gold standard POS tagged text, we obtain encouraging results. But when the chunker is provided with POS tagged output of our POS tagger, the results are less, this is because of the errors in the POS tagging. Here we have not done any post processing of the text after the POS tagging.

6 Conclusion

In this paper we have discussed a hybrid part-of-speech tagger and a text chunker for Indian languages. Here we have taken the tags into consideration and the script encoding used is not considered. Hence the system developed can work on any Indian language with different encoding schemes. The work shows that, the hybrid POS tagger for an agglutinative, relatively free word order language works well and gives high precision and recall. This was also shown in the literature for Tamil language (Arulmozhi and Sobha, 2006).

In developing text chunker we have used the machine learning technique, transformation based learning method. The results are promising.

References

Steven Abney (1996). Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop, August 1996.*

Arulmozhi. P, Sobha. L, Kumara Shanmugam. B. (2004). Parts of Speech Tagger for Tamil. *In the Proceedings of the Symposium on Indian Morphology, Phonology & Language Engineering, Indian Institute of Technology, Kharagpur*. pp 55-57.

Arulmozhi. P, Sobha L (2006) HMM based POS Tagger for a Relatively Free Word Order Language. In *journal of Research in Computing Science , Volume 8, Feb 2006.*

P. Arulmozhi , T. Pattabhi R K Rao, Sobha L. (2006), A Hybrid POS Tagger for a Relative Free Word Order Language. *In Proceedings of the MSPIL-06, Indian Institute of Technology, Bombay*. pp 79-85.

Eric Brill (1994). Some Advances in transformation Based Part of Speech Tagging. *In proceedings of the Twelfth International Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA. pp 722-727.

Eric, Brill. (1993b). *A Corpus-Based Approach to Language Learning***.** Ph.D. thesis, University of Pennsylvania, USA.

Eric, Brill and Philip Resnik. (1994). A rule-based approach to prepositional attachment disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics, Japan.* (cmp-lg/9410026). pp 1198-1204.

Eugene Charniak (1993): Statistical Language Learning,. MIT Press, Cambridge, London

Kenneth W. Church. (1988). A Stochastic Parts Program and Noun Phrase Parser for unrestricted Text. In *Proceedings of the Second Conference on Applied Natural language Processing, Austin, Texas.* pp 136-143.

Jun'ichi Kazama, Yusuke Miyao and Jun'ichi Tsujii (2001) A Maximum Entropy Tagger with Unsupervised Hidden Markov Models. *Proceedings of the Sixth Natural Language Processing, Pacfic Rim Symposium.* pp 333-340.

Shereen KHOJA (2001): *APT:* Arabic Part-of-speech Tagger, *Proceedings of the Student Workshop at the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Carnegie Mellon University, Pittsburgh, Pennsylvania,USA.

L. Ramshaw and M. Marcus. (1995). Text Chunking using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*. pp 82-94.

Sobha L, Vijay Sundar Ram R (2006). Noun Phrase Chunking in Tamil. *In Proceedings of the MSPIL-06, Indian Institute of Technology, Bombay.* pp 194-198.

An HMM based Part-Of-Speech tagger and statistical chunker for 3 Indian languages

G.M. Ravi Sastry
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur,
West Bengal-721302.
gmravi2003@gmail.com

Sourish Chaudhuri
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur,
West Bengal-721302.
sourish@iitkgp.ac.in

P. Nagender Reddy
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur,
West Bengal-721302.
npaduru@gmail.com

Abstract

In this paper, we describe our experiences in building an HMM based Part-Of-Speech (POS) tagger and statistical chunker for 3 Indian languages-Bengali, Hindi and Telugu. We employ the TnT tagger model for POS tagging of the corpus. The POS tagging accuracies for Bengali, Hindi and Telugu are 74.58, 78.35 and 75.37 respectively. For chunking, we use the training data to extract chunk pattern templates defined as a sequence of POS tags. These templates, in conjunction with the POS tag of the word following the chunk, are used to decide chunk boundaries in the unannotated text. A dynamic programming algorithm is used to find the best possible chunk sequence. The chunk accuracies obtained are 67.52 for Bengali, 69.98 for Hindi and 68.32 for Telugu. The techniques used are generic and are expected to produce comparable accuracies for different languages.

1 Introduction

Shallow parsing (Manning and Schutze, 1999) consists of POS tagging and chunking. They are usually carried out by rule based approaches (Brill, 1992) or by learning approaches (Clark and Thollard, 2002) or a combination of both. In this work, we use machine learning approaches to tag and chunk the texts in order to make the method more generic for different languages.

Given a text in Hindi, Bengali or Telugu, this work aims to generate POS tags for the words, and then chunk the text into word groups. A corpus of about 25000 words in each language is used for training the system and it is evaluated on test data of about 5000 words. As a result, the chunker accuracy is limited by the accuracy of the POS tagger. The rest of the paper is organized as follows: Section 2 describes the POS tagging approach and section 3 describes the chunking approach.

2 Part-Of-Speech (POS) Tagging

In this section, we outline the POS tagging technique used by us. In subsection 2.1, we discuss some of the existing techniques commonly adopted in POS tagging, 2.2 describes our approach, and 2.3 discusses the results obtained on the test set.

2.1 Background

The earliest attempts at POS tagging were based on a 2 stage architecture (Harris,1962;Klein and Simmons,1963;Greene and Rubin ,1971) .The ENGTOWL tagger(Voutilainene 1995) is based on the same 2 stage architecture with more sophisticated lexicon and disambiguation rules. Probabilities were first used by Stolz et al (1965),) and various statistical taggers were built in the 1980's (Marshall, 1983). Trigrams'n'Tags (TnT) is an efficient statistical POS tagger based on HMM model which works by training on tagged data. Kupiec(1992), Cutting et al (1992)showed that it is also possible to train a HMM tagger on unlabelled data using EM algorithm.

2.2 Approach.

For POS tagging, we use the Trigrams'n'Tags or the TnT tagger (Brants, 2000). The tagger implements the Viterbi algorithm for second order

Markov models. TnT is an efficient statistical part-of-speech tagger that can be trained on different languages and virtually any tag set. The component for parameter generation trains on tagged corpora. The system incorporates several methods of smoothing and of handling unknown words. TnT is not optimized for a particular language. Instead, it is optimized for training on a large variety of corpora. For our purposes, we were required to adapt the TnT tagger for each of the 3 Indian languages given.

In this model, smoothing is carried out using linear interpolation, and the respective weights are determined by deleted interpolation. Unknown words are handled by a suffix trie and successive abstraction. TnT is not tuned to perform better for any particular language. In our work, since we test the tagger over 3 different languages, no language specific optimizations have been attempted.

2.3 Results

Table 1 lists the POS tagging accuracies for the final test data supplied:

	No of Test Tokens	Correctly Tagged	Accuracy
Bengali	5225	3897	74.58%
Hindi	4924	3858	78.35%
Telugu	5193	3909	75.27%

Table 1: POS Tagging Accuracies

The following tables summarize the precision and recall values for different POS tags over the test data. Only the most frequent POS tags have been chosen.

POS Tag	Bengali	Hindi	Telugu
CC	85.5	90.4	80.1
JJ	61.8	45.4	62.2
NN	70.5	76.7	75.8
PREP	88.8	96.8	64.0
PRP	88.1	88.2	77.9
SYM	100.0	99.6	99.7
VAUX	71.6	86.6	38.5
VFM	74.5	80.9	75.0
VRB	58.8	55.3	66.4

Table 2: Precision values in percentage

POS Tag	Bengali	Hindi	Telugu
CC	90.4	91.3	81.2
JJ	69.5	62.3	70.1
NN	86.4	77.3	82.0
PREP	78.2	96.8	62.1
PRP	88.9	95.0	81.5
SYM	100.0	97.6	98.6
VAUX	73.4	93.0	10.9
VFM	74.8	83.7	82.9
VRB	67.2	75.0	73.4

Table 3: Recall Values in percentage

3 Chunking

We have attempted to implement a novel method to obtain the chunks using the Cocke-Kasami-Young (CKY) algorithm. The following subsections outline the existing techniques in chunking, our approach and the results obtained on testing our method on the test data.

3.1 Background

Several methods have been used to carry out shallow parsing of text in European languages. Transformation based learning (Marcus and Ramshaw, 1995) is an accepted technique for chunking. Linear observed time statistical parsers using maximum entropy models (Ratnaparkhi, 1997), usage of decision trees in probabilistic POS tagging (Schmid, 1994), parsing by chunks (Abney, 1991), graphical parsing models (Silvescu and Honavar). There have been efforts made to develop parsing techniques that would be independent of the language used (Kinyon, 2002) and we have attempted to build our system on similar lines so that the results are comparable for different languages.

3.2 Approach

The training files are used to extract chunk templates. These templates are used to statistically chunk the testing files. The POS tags attached to each word of the test files are obtained using the POS tagging technique described earlier. The chunking technique uses a dynamic programming algorithm which utilizes the probabilities of various parameters extracted during chunking to obtain the best possible chunk ordering within a sentence.

The system is trained on the POS-tagged and chunked training data files supplied and extracts the different chunk templates. Essentially, it estimates the following parameters:

1) *Chunk-POS combinations*: When a chunk template C is discovered with the POS tag of the word just outside the chunk being p, we increment the count of "chunk-POS" combination C-p. This way we estimate the number of times a chunk C occurs with the POS tag of the word just outside the chunk being p.

2) *Chunk structure*: The label of a chunk indicates the kind of chunk it is. E.g. If we have the chunk template ((NP UH SYM)), then the label of the chunk is NP, and it consists of words with POS tags UH and SYM.
3) *Conditioning on the POS tag of the word following a chunk*: Every time a chunk is discovered, the count of chunks occurring with that chunk label and the POS tag of word following the chunk as p is increased.
4) The number of times *a particular POS tag follows a chunk*.

For each sentence in the test data, we seek to find the best possible chunk ordering (Jurafsky and Martin, 2005):

$$c\hat{} = \underset{C \epsilon \zeta(S)}{argmax}\ P(C|S) \quad \ldots(1)$$

where $\zeta(S)$ denotes all possible chunk combinations for the sentence S.

From (1), we can derive,

$$c\hat{} = \underset{C \epsilon \zeta(S)}{argmax}\ P(C,S) \quad \ldots(2)$$

Define P(C,S) as $\Pi\ P(C_i)$
where C_i are labelled chunks forming C.

Hence, $c\hat{} = \underset{C \epsilon \zeta(S)}{argmax}\ \Pi P(Ci) \quad \ldots(3)$

The dynamic programming algorithm tries to find a chunk ordering for each sentence such that the probability of occurrence of the chunk sequence is maximised.

Each sentence can be divided into a number of chunks. We start by evaluating the probability that each POS tag may be a chunk by itself. Then, we evaluate the probability of 2 consecutive POS tags being part of the same chunk. In this way, we continue evaluating probabilities of up to *n* consecutive POS tags forming a chunk (n is the length of the sentence).

Each chunk is assigned a label using the chunk templates learnt while training. In case a POS combination is found that was not encountered while training the system, a dummy chunk label is assigned to it.

The probability of occurrence of any of the chunks, given the label of the chunk, is conditioned on the POS tag of the word just outside the chunk and on the right in that particular sentence. *apos* represents the POS tag of the word following the chunk in the sentence. Hence, Eq 3 modifies to:

$$c\hat{} = \underset{C \epsilon \zeta(S)}{argmax}\ \Pi\ P(chunk|label,apos)*P(apos) \quad \ldots(4)$$

Using M.L.E estimate on (4), we get

$$P(chunk|label,apos) = C(chunk,label,apos)/C(label,apos) \quad (5)$$

where $C(x)$ is count of x

In case $C(chunk,label,apos)=0$

then, $P(chunk|label)=c*[P(chunk,label)/P(label)] \quad \ldots(6)$

where c is a constant = 10^-7 in our experiments.

We now use these probabilities to find the optimal chunk ordering for the sentence. Consider the set of POS tags from position *i* in the sentence to position *j*. We can find the optimal chunking sequence by maximizing:

$$P([i,j]) = \underset{i \le k \le j}{max}\ (P([i,k])*P([k,j])) \quad \ldots(7)$$

The values of *k* for which the probability becomes maximum for different *i* , *j* are stored. In order to find the chunk ordering for the complete sentence, set *i=1*, and *j=n*. The values of *k* stored earlier are used to obtain the possible chunk ordering for the complete sentence.

3.3 Results

The accuracies of chunking are as given in Table 4.

	Precision	Recall	FB1
Bengali	67.52%	67.52%	67.52%
Hindi	78.35%	78.35%	78.35%
Telugu	68.32%	68.32%	68.32%

Table 4: Chunking Accuracies

The precision and recall values for the more frequently occurring chunks are listed in Tables 5 and 6.

Chunk	Bengali	Hindi	Telugu
B-BLK	95.0	4.11	0.0
B-CCP	18.4	13.8	7.7
B-NP	80.4	75.3	75.6
B-VG	77.3	60.3	66.8
I-NP	55.5	80.1	65.7
I-VG	42.2	87.3	76.1

Table 5: Precision values in percentage

Chunk	Bengali	Hindi	Telugu
B-BLK	45.7	9.1	0.0
B-CCP	46.3	24.3	15.0
B-NP	76.3	73.4	73.6
B-VG	80.3	79.6	68.6
I-NP	46.9	65.8	67.6
I-VG	74.0	72.7	69.6

Table 6: Recall values in percentage

4 Conclusion

The TnT tagger uses the Viterbi algorithm. Its processing time may be reduced by introducing a beam search which can greatly increase the speed of execution. However, it no longer guarantees the optimal solution.

Telugu, owing to its agglutinative nature, tends to give slightly lower accuracies for POS tagging in most cases.

OOV words are often mistakenly tagged. A confusion matrix reveals that about 50% NNPs (proper nouns) have been tagged as NNs (common nouns).

The chunking algorithm uses a constant c= 10^-7. Typically, the probability values extracted from the training data are of the order 10^-5. This empirically chosen constant reduces the probability of the unseen chunk as compared to others. It might be possible to improve the accuracy of the chunking algorithm by finding an optimal value for this constant. Further, it might be possible to guess the actual chunk label of a hitherto unseen chunk by looking at part of the POS tags that constitute the chunk.

5 Acknowledgements

We are extremely grateful to Prof. Sudeshna Sarkar, Prof. Anupam Basu, and Monojit Choudhury for their invaluable suggestions and encouragement.

6 References

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, May 1999.

Eric Brill. 1992. *A Simple Rule-based Part Of Speech Tagger.* Proceedings of the DARPA Speech And Natural Language Workshop, pp112-116.

Alexander Clark and Franck Thollard. 2002. *Shallow Parsing Using Probabilistic Grammatical Inference.* ICGI 2002, LNAI 2484, pp. 269–282, 2002

B.B. Greene and G.M. Rubin.1971. *Automatic grammatical tagging of English.* Technical Report,Department of Linguistics, Brown University.

Atro Voutilainen 1995. *A syntax-based part of speech analyser*, Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics.

W. Stolz.1965. *A probabilistic procedure for grouping words into phrases*. Language and Speech

I. Marshall.1983. *Choice of grammatical word-class without global syntactic analysis*: Tagging words in the lob corpus. Computers in Humanities, 17:139--150,1983

Cutting, Kupiec, Pedersen, Sibun.1992.*A Practical Part-of-Speech Tagger*.Proceedings of the Third Conference on Applied Natural Language Processing

L.A. Ramshaw & M.P. Marcus. 1995. *Text Chunking using Transformation−Based Learning*, ACL Third Workshop on Very Large Corpora, pp.82−94, 1995.

A. Ratnaparkhi 1997.*Linear observed time statistical parser based on maximum entropy models.* Technical Report cmplg/9706014.

H. Schmid 1994 *Probabilistic Part−Of−Speech Tagging Using Decision Trees.* Proc. NEMLAP'94.

S. Abney 1991. *Parsing by chunks. In Principle−based Parsing.* (R. Berwick, S. Abney and C. Tenny eds), Kluwer academic publishers.

Adrian Silvescu and Vasant Honavar. *A Graphical model for Shallow Parsing Sequences.*

Alexandra Kinyon. 2002 *A Language-independent Shallow-parser Compiler.* University of Pennysylvania.

Thosten Brants, 2000. *TnT A Statistical part of speech tagger.* Proceedings of the 6th Applied Natural Language Processing Conference.

Daniel Jurafsky and James Martin, 2005. *Speech and Language processing.* Pearson Education, 2005.

Part of Speech Tagging and Shallow Parsing of Indian Languages

Delip Rao
Department of Computer Science,
Johns Hopkins University, USA
delip@cs.jhu.edu

David Yarowsky
Department of Computer Science,
Johns Hopkins University, USA
yarowsky@cs.jhu.edu

Abstract

This paper describes and evaluates shallow parsing of several Indian languages utilizing Conditional Random Field models. We show how performance can be substantially improved by several feature enhancements and improved modeling techniques, including expanding the chunk tag inventory, and separating punctuation from linguistic phrases. We also report results from part of speech tagging of Hindi, Bengali and Telugu using generative methods.

1 Introduction

Part of speech tagging and shallow parsing are basic tasks used in several NLP applications. Part of speech tagging refers to assigning words to a closed set of linguistic classes. Shallow parsing involves identification of constituents in a sentence. These constituents are helpful in various NLP tasks like Named Entity Recognition, Information Extraction, Question Answering and also to speed up deep parsing. We describe a general language independent approach to shallow parsing that we apply to Indian languages.

2 Part of Speech Tagging

We tried several approaches for part of speech tagging Hindi, Bengali, and Telugu. The first model (INV) uses a Naive Bayes approach and handles out of vocabulary words (OOV) as NN. The second model (SUFFIX) uses the Naive Bayes assumption but for OOV words it assigns a tag that maximizes the likelihood of the tag given a fixed length suffix of the word. That is, if the word is OOV then $predictedtag = \arg\max_{tag} P(tag \mid \textit{suffix})$. These probabilities are easily estimated from the training data.

The use of suffixes for predicting the tags for OOV words is based on the intuition that in the absence of morphological taggers, a fixed length suffix can serve as an approximation to the morphological model. This is demonstrated in Table 1 derived from the development data.

Table 1: An example of SUFFIX model tag prediction on OOV words

Language	OOV word	suffix	PTag	P(PTag| suffix)
Hindi	darshanasi.nha	.nha	NNP	0.91
Hindi	visheShataao.n	ao.n	NN	0.96
Telugu	praWamamEna	mEna	JJ	0.79
Telugu	nixarSanamulu	mulu	NN	1.00

The choice of suffix length is important as illustrated in Table 2. Based on this information, we selected suffix lengths for the SUFFIX model as 2, 3, and 3 for Hindi, Bengali, and Telugu respectively.

We then compare two Hidden Markov Model tagger implementations QTag (Tufis and Mason, 1998) and TnT (Brants, 2000). The results for part of speech tagging for all words is given the Table 3. Tables 4 and 5 show the part of speech tagging results for out of vocabulary and in vocabulary words separately. It is interesting to note that TnT consis-

Table 2: An example of SUFFIX model tagging performance for different suffix lengths on development data. Bold is the best performing suffix length for the given language.

Suffix length	Hindi	Bengali	Telugu
1	78.0	61.3	73.6
2	**78.3**	64.3	74.5
3	77.8	**66.3**	**74.6**
4	77.4	61.9	72.4

tently performs well on out of vocabulary words.

Table 3: Part-of-speech tagging results on the development data (all words)

Method	Hindi	Bengali	Telugu
INV	78.9	73.4	61.7
SUFFIX	79.0	74.6	**69.8**
QTAG	78.3	73.2	65.7
TnT	**84.3**	**78.0**	68.3

Table 4: Part-of-speech tagging results on the development data (OOV words only)

Method	Hindi	Bengali	Telugu
INV	44.5	48.1	41.2
SUFFIX	44.9	51.5	**55.7**
QTAG	45.1	47.7	49.0
TnT	**56.9**	**56.6**	55.5

There are two important observations to be made: Firstly, in the absence of a morphological tagger for a new language, merely using the word suffixes improves performance. Secondly, TnT on an average performs better than Qtag. Table 6 shows the result of TnT tagger (best approach) on the test data.

The following section explains our approach to shallow parsing.

3 Shallow Parsing

We treat the task of shallow parsing as any knowledge discovery (KDD) process and try to perform feature enhancement. We show how the feature enhancement process improves the performance of the machine learning system. We used Conditional Random Fields for this purpose. For Conditional

Table 5: Part-of-speech tagging results on the development data (non-OOV words only)

Method	Hindi	Bengali	Telugu
INV	88.2	86.3	**87.7**
SUFFIX	88.2	86.3	**87.7**
QTAG	87.1	86.1	86.5
TnT	**91.7**	**88.9**	84.7

Table 6: Part-of-speech tagging results on test data

Language	Accuracy
Hindi	76.68
Bengali	74.20
Telugu	76.01

Random Fields, we used the SimpleTagger command line interface of the Mallet package (McCallum, 2002). Our feature sets for CRFs were same as that used in (Sha and Pereira, 2003).

3.1 Feature Enhancement

3.1.1 Relabeling

Although the input presented to us in the SSF format (Bharathi et al., 2003) is very informative about the boundary structure, it is not easy work with. The BIO format (Ramshaw and Marcus, 1995),where each word is tagged as being in the Beginning (B), Inside (I) or Outside (O) of a chunk, on the other hand is simpler to work with, and makes the boundary structure implicit. We augment the BIO notation to form a new representation BIES – Begin (B), Inside (I), End (E), and Singleton (S). The BIES notation is functionally equivalent to the BIO notation, and lossless bidirectional conversion from one to the other is possible, but a representation in BIES form allows better discrimination of the chunk boundaries as our results indicate.

3.1.2 Handling punctuation

Our treatment of the punctuation symbols is motivated by the following observation. Consider the following three noun phrases from the Hindi training data supplied to us:

```
[ bhUmi/NN ]/NP
[ kalI/NN '/SYM ]/NP
[ saakShaatkaara/NN -/SYM -/SYM ]/NP
```

The above NPs are syntactically similar except for the trailing SYM at the end of the second and the third chunk. The presence of these punctuation symbols at the end act as noise and deters the learning of clear syntactic patterns (e.g. which parts of speech begin and end chunks). We handle trailing punctuation symbols by purging them out of the chunk and create separate chunks for them as shown in the example below.

```
[ saakShaatkaara/NN -/SYM -/SYM ]/NP
[ saakShaatkaara/NN ]/NP
[-/SYM ]/BLK [-/SYM ]/BLK
```

3.2 Effect of composite classifiers

We tried to treat the chunking task as a composite classification task where two separate classifiers were learnt for the boundary prediction and the chunk labeling tasks respectively. As shown[1] in table 7, we get lower performance by composite classifiers on the CRF model.

Table 7: Effect of two CRF classifiers

Method	F1
CRF	88.51
CRF (2 classifiers)	86.77

3.3 Experiments

In this section we perform pairwise evaluation of the observations made in section 3.1. These experiments were performed on the Hindi development dataset.

3.3.1 Utility of the BIES representation

In this section we evaluate the impact of the BIES representation over the popular BIO representation. We converted the data from the BIO format to the BIES representation. Note that these two representations are isomorphic but the BIES format makes the chunk boundaries explicit. A chunk [BIII] would be converted to [BIIE]. S is used for singleton chunks. Table 8 shows the improvement obtained by mapping BIO to BIES alone.

[1]The comparative performance for the variant CRF chunking models given in Table 6-8 on development data are based on the use of goldstandard part-of-speech tags for the development data, rather than model derived tags.

Table 8: Gain from BIO to BIES: CRF

Method	F1
CRF + BIO	79.01
CRF + BIES	82.53

From these results it is clear that conversion to BIES performs better than BIO.

3.3.2 Effect of handling punctuation

We wanted to find out how much difference would be made by handling punctuation the way it was described in Section 3.1.2. The effect of excluding punctuation from chunks in the data is shown in Table 9.

Table 9: Effect of punctuation on CRF+BIES

Method	F1
CRF + BIES + No Punct	82.53
CRF + BIES + Punct	88.51

These results make a compelling case to factor punctuation symbols outside the chunks and later add them back in the result.

4 Submitted System

4.1 Description

Our submitted system for the Shallow Parsing task made use of a combination of Naive Bayes Classifier for chunk labeling and boundary identification. Some additional constraint rules were used to add the correct the boundaries produced by the Naive Bayes Classifier. The constraint rules are listed below:

$b_{-1} = B$ and $b_0 = B => b_0 = I$
$b_{-1} = E$ and $b_0 = I => b_0 = B$
$b_{-1} = E$ and $b_0 = E => b_0 = B$
$b_{-1} = S$ and $b_0 = E => b_0 = B$
$b_{-1} = S$ and $b_0 = I => b_0 = B$
$b_{-1} = B$ and $b_0 = I => b_0 = B$
$b_{-1} = I$ and $b_0 = I => b_0 = B$

Here b_0 and b_{-1} are the boundary labels of the current and previous words respectively.

4.2 Results

The performance of the submitted system on the test data is shown in Table 10.

Table 10: Chunking results on test data

Language	F1
Hindi	73.58
Bengali	64.41
Telugu	68.75

5 Lessons learnt

Some key lessons were learnt during this exercise.

1. The BIO notation when used as features has lesser discriminatory power than the BIES notation. Better results were obtained by simple mapping of labels from BIO to BIES format alone.

2. Punctuation hinders learning of patterns (rules) in the data, and performance can be improved by separating punctuation into separate standalone chunks.

3. In the absence of morphological information, the mere use of suffix-based PoS tagging models helps in improving PoS tagging accuracy on out-of-vocabulary words (not seen in training data).

6 Conclusion

In this paper, we presented and contrasted several techniques for the part of speech tagging and text chunking task in several Indian languages. The methods presented here are not specific to any language and apply to Hindi, Bengali and Telugu. We showed among the generative methods considered in the paper, TnT performed best for part of speech tagging. For the chunking task we showed how performance of Conditional Random Fields can be substantially improved by several feature enhancements and improved modeling techniques, including expanding the chunk tag inventory, and separating punctuation from linguistic phrases.

References

Akshar Bharathi, Rajeev Sanghal, and Dipti M. Sharma. 2003. Shakti analyser: Ssf representation.

Thorsten Brants. 2000. TnT–A statistical part-of-speech tagger. In *Proceedings of Applied Natural Language Processing*, pages 224–231.

Andrew McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94. Association for Computational Linguistics.

Fei Sha and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220.

Dan Tufis and Oliver Mason. 1998. Tagging romanian texts: a case study for qtag, a language independent probabilistic tagger. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, pages 589–596.

Part-Of-Speech Tagging and Chunking using Conditional Random Fields and Transformation Based Learning

Avinesh.PVS, Karthik G
Dept. of Computer Science
IIIT - Hyderabad
{avinesh, karthikg}@students.iiit.ac.in

Abstract

In this paper we describe Part Of Speech (POS) tagging and Chunking using Conditional Random Fields (CRFs) and Transformation Based Learning (TBL) for Telugu, Hindi and Bengali. We show here how to train CRFs to achieve good performance over any other ML techniques. Improved training methods based on the morphological information, contextual and the lexical rules (developed using TBL) were critical in achieving good results. The CRF and TBL based POS tagger has an accuracy of about 77.37%, 78.66%, and 76.08% for Telugu, Hindi and Bengali, and the chunker performs at 79.15%, 80.97% and 82.74% for Telugu, Hindi and Bengali respectively.

1. Introduction:

POS-tagging is the process of assigning the part of speech tags to the natural language text based on both its definition and its context. Identifying the POS-tags in a given text is an important aspect of any Natural Language Application.

POS tagging has been developed using the statistical implementations, linguistic rules and sometimes both. Some of the statistical models are the Hidden Markov Models (HMMs) (Cutting 1992), Maximum Entropy Models (MEMMs) (Adwait Raatnaparakhi [2] 1999), CRFs (Fei Sha and Fernando Pereira [1] 2002) and TBL (Eric Brill [7] 1992). These taggers don't work well when small amount of tagged data is used to estimate the parameters of the tagger. So we need to add some extra information like morphological roots and the possible tags for the words in the corpus to improve the performance of the tagger.

Chunking or shallow parsing is the task of identifying and segmenting the text into syntactically correlated word groups. It is considered as an intermediate step towards full parsing.

This paper presents the use of CRFs with the help of morphological information and the transformation rules in POS tagging and Chunking of Indian languages. In Indian languages, the availability of the tagged corpus is very less and so most of the techniques suffer due to data sparseness problem. For the current task the training and the test data is provided by SPSAL workshop at IJCAI 2007.

2. Similar Works

Most of the previous work used two main machine-learning approaches for sequence labeling. The first approach lies on k-order generative probabilistic models of paired input sequences, for instance HMM (Frieda and McCallum [1] 2000) or multilevel Markov Models (Bikel et al. 1999).The second approach views the sequence labeling problem as a sequence of a classification problem, one for each of the labels in the sequence.

CRFs bring together the best of generative and classification models. Like classification models, they can accommodate many statistically correlated features of the inputs, and they are trained discriminatively. But like generative models they can trade off decisions at different sequence positions to obtain a globally optimal labeling. Lafferty [5] (2001) showed that CRFs beat related classification models as well as HMMs on synthetic data and on POS-tagging task.

Among the text chunking techniques Fei Sha and Fernando Pereira [1] (2000) proposed a Conditional Random Field based approach; Lance A. Ramshaw (1995) proposed a Transformation-Based Learning approach. There are also other approaches based on Maximum entropy (Rob Koeling), memory- based etc.

3.1 Conditional Random Fields

Conditional random field is a probabilistic framework for labeling and segmenting data. It is

a form of undirected graphical model that defines a single log-linear distribution over label sequences given a particular observation sequence. CRFs define conditional probability distributions $P(\mathbf{Y}|\mathbf{X})$ of label sequences given input sequences. Lafferty et al. defines the probability of a particular label sequence Y given observation sequence X to be a normalized product of potential functions each of the form

$$\exp(\sum \lambda_j t_j (Y_{i-1}, Y_i, X, i) + \sum \mu_k s_k (Y_i, X, i))$$

where $t_j(Y_{i-1}, Y_i, X, i)$ is a transition feature function of the entire observation sequence and the labels at positions *i* and *i-1* in the label sequence; $s_k(Y_i, X, i)$ is a state feature function of the label at position I and the observation sequence; and λ_j and μ_k are parameters to be estimated from training data.

$$F_j(Y,X) = \sum f_j (Y_{i-1}, Y_i, X, i)$$

where each $f_j(Y_{i-1}, Y_i, X, i)$ is either a state function $s(Y_{i-1}, Y_i, X, i)$ or a transition function $t(Y_{i-1}, Y_i, X, i)$. This allows the probability of a label sequence **Y** given an observation sequence **X** to be written as

$$P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\lambda}) = (1/Z(\mathbf{X})) \exp(\sum \lambda_j F_j(\boldsymbol{Y}, \boldsymbol{X}))$$

$Z(\mathbf{X})$ is a normalization factor.

3.2 Transformation Based Learning

Transformation-based learning starts with a supervised training corpus that specifies the correct values for some linguistic feature of interest, a baseline heuristics for predicting the values for that feature, and a set of rule templates that determine a space of possible features in the neighborhood surrounding a word, and their action is to change the system's current guess as to the feature for the word. To learn a model, one first applies the baseline heuristic to produce initial hypotheses of the training corpus. Where this baseline prediction is not correct, the templates are then used to form the instantiated candidate rules. This process eventually identifies all the rules candidates generated by that template set that would have a positive effect on the current tag assignments anywhere in the corpus. Those candidate rules are then tested against the rest of corpus, to identify the negative changes. This entire process is then repeated on the transformed corpus deriving candidate rules, scoring them, and selecting one with the maximal positive effect.

This way the lexical and the contextual rules are generated from the training corpus.

4. Approach:

4.1 POS Tagging

The approach we used for POS tagging is as follows: CRF model (CRF++) is used to perform the initial tagging and then a set of transformation rules is applied to correct the errors produced by CRFs.

Initially we used basic features in CRFs, later added the morphological information like the root word, all possible pos tags for the words in the corpus, the suffix and prefix information. This information is added to the training corpus and then it is trained using these features.

To measure the performance of CRFs against other ML approaches we carried out various experiments using Brant's TnT [3] for HMMs, Maxent for MEMMs. Interestingly HMMs performed as high as CRFs with basic features. We preferred CRFs over HMMs as addition of features like the root words were much easier in CRFs.

4.2.Chunking

For chunking first we tried out HMM's to mark the chunk labels. Later the system is trained on the feature templates for predicting the chunk boundary names using CRFs. Finally the chunk labels and the chunk boundary names are merged to obtain the chunk tag. It is basically HMM+ CRF model for chunking.

5. Experiments

5.1 POS Tagging

Initially we trained the CRF on baseline template i.e. over local words of the current word with a window size of 2, 4, and 6; and tried all possible combinations of features. It was observed that CRFs gave better results with a window size of 4 and the combinations of previous words and the current word. Using the basic template the accuracy was 73.47%.

As Telugu is an agglutinative language i.e. the words are joined together to form new words and the postfixes are often attached to the word, so we used the suffix information for each word. The last two letters, last three, and last four letters of the word are added as suffixes in the training corpus.

Later we added the root information and the probable tags of the word from the morphological analyzer. The combination of the word and its root marked an increment of 1% in the performance of the system.

The size of the word also played a major role in assigning the POS tags. The threshold considered was 3 i.e. words whose length is less than three belonged to one class and the rest to the other. This marked an improvement of 1% in the performance. This is because the average length of non-functional words in Indian languages is around 3.

Transformation rules produced by TBL are then used to change the incorrect tags produced by the CRFs. Interestingly it gave an increase of 0.6% for Hindi where as for Telugu initially the accuracy decreased. This is due to the agglutinative nature of Telugu. Due to this the rules had a negative effect some times.

These errors are further reduced by the deleting few of the transformation rules which induced negative effect. This gave an improvement of 1% for all the three languages.

The same model is used for Telugu, Hindi and Bengali except for few differences in the window size i.e. for Hindi, Bengali and Telugu we used a window size of 6, 6 and 4 respectively.

Language	Training Data	Test Data	Results (%)
Telugu	**21425**	**5193**	**77.37**
Bengali	**20397**	**5225**	**76.08**
Hindi	**21470**	**4924**	**78.66**

Fig.1 POS Tagging Results and Data size

5.2 Chunking

Initially we tried out HMM's to mark the chunk boundary and then some rules to identify the boundary names which gave a precision of 76.59% (Telugu test data).

Then we used CRF model with basic features such as words, POS tags and the combination of the both; which improved the performance of the system to 79.15 % (Telugu test data).The Fig.2 shows the result of chunking (Telugu test data) using the POS tags provided with the test set.

Basically the same HMM+CRF model is used for Telugu, Hindi and Bengali chunking. And the features and the combination of features used in the CRF model are also the same.

Chunk Labels	Precision (%)	Recall (%)	$F_{\beta=1}$
B-CCP	**79.15**	**67.21**	**72.97**
B-JJP	**50.00**	**10.00**	**16.67**
B-NP	**78.17**	**90.27**	**83.79**
B-RBP	**44.83**	**27.08**	**33.77**
B-VG	**76.50**	**79.76**	**78.09**
I-CCP	**42.86**	**37.50**	**40.00**
I-JJP	**100.00**	**16.67**	**28.57**
I-NP	**82.45**	**71.19**	**76.41**
I-RBP	**38.46**	**27.78**	**32.26**
I-VG	**83.93**	**80.13**	**81.98**
Overall	**79.15**	**79.15**	**79.15**

Fig2. Chunking (Telugu) with reference POS tags

Language	Results (%)
Telugu	**79.15**
Bengali	**82.74**
Hindi	**80.97**

Fig 3. Chunking Results

6. Error Analysis

Actual tag	Assigned tag	Counts
NN	NNP	218
NN	JJ	208
NN	RB	85
PREP	NLOC	82
NN	PREP	61
VRB	VFM	58
JJ	NN	50
NN	QFNUM	46
VFM	NEG	24
PRP	NN	10

Fig 4.Error Analysis for POS-tagging

7. Conclusion

The overall results obtained for POS tagging is 77.37% and for chunking it is 79.17% (Telugu). The accuracy of the Telugu POS Tagging seemed to be low compared to other Indian Languages due to agglutinative nature of the language.

One more interesting thing to observe is that in some of the cases the sandhi is splited and in some other cases it is not splited.

Eg:

1: *pAxaprohAlace* (NN) = *pAxaprahArAliiu* (NN) + *ce* (PREP)

2: *vAllumtAru*(V) = *vAlylyu*(NN) + *uMtAru*(V)

We demonstrated the use of CRFs and TBL for POS tagging for Indian Languages which gave us good results. This could be future looked into to improve the performance of the tagger.

8. Acknowledgments

We would like to express our gratitude to Dr. Dipti Mishra and Prof. Sangal for their guidance and support. We would also like to thank Prashanth and Himanshu for their valuable suggestions.

References

[1] Fei Sha and Fernando Pereira 2003, Shallow Parsing with Conditional Random Fields.*In the Proceedings of HLT-NAACL*.

[2] Adwait Ratnaparkhi , 1998, Maximum Entropy Model For Natural Language Ambiguity Resolution, Dissertation in Computer and Information Science, University Of Pennslyvania, 1998.

[**3] Thorsten Brants, 2000**. TnT – a Statistical Part-of-Speech Tagger. *Proceeding of the sixth conference on Applied Natural Language Processing (2000) pg 224-231*.

[4] Charles Sutton, An Introduction to Conditional Random Fields for Relational Learning

[5] John Lafferty, Andrew McCallum and Fernando Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.

[6] Himanshu Agarwal and Anirudh Mani, Part of Speech Tagging and Chunking with Conditional Random Fields**.** *In the Proceedings of NWAI workshop 2006.*

[7] Eric Brill. 1995 Transformation-based error driven learning and natural language processing: A case study in part-of-speech tagging**.** *Computational Linguistics.*

[8] CRF++: Yet Another Toolkit

http://chasen.org/~taku/software/CRF++

POS Tagging Using HMM and Rule-based Chunking

Asif Ekbal
Comp. Sc. & Engg. Deptt.
Jadavpur University
India
ekbal_asif12@yahoo.co.in

Samiran Mandal
Comp. Sc. & Engg. Deptt.
Jadavpur University
India
samiranjucs@gmail.com

Sivaji Bandyopadhyay
Comp. Sc. & Engg. Deptt.
Jadavpur University
India
sivaji_cse_ju@yahoo.com

Abstract

This work reports about a Part of Speech (POS) tagger based on the Hidden Markov Model and a rule-based chunker. The POS tagger has been initially trained on a Bengali training set consisting of 20396 tokens. The tagger has been tested on the Bengali development test set consisting of 5022 tokens and demonstrated 90.9% accuracy. A Named entity recognition (NER) system and a lexicon in Bengali with basic POS information have been developed to take care of the unknown words in Bengali. The POS tagger is then trained on the Hindi and Telegu training sets, consisting of 21470 and 21415 tokens respectively. The POS tagger has been tested with Hindi and Telegu development sets, consisting of 5681 and 6098 tokens and it has demonstrated 82.05% and 63.93% accuracies respectively. At present, the Hindi and Telegu POS tagger cannot efficiently handle the unknown words. A rule-based chunker has been developed and tested on the Bengali development test set and demonstrated 85.88% accuracy. This chunker has then been tested on the Hindi and Telegu development sets and demonstrated 73.88% and 55.96% accuracies. Finally, the POS tagger has demonstrated 77.73%, 76.87% and 67.49% accuracies with the unannotated test sets of Bengali, Hindi and Telegu. The chunking system has demonstrated 80.63%, 71.65% and 53.15% accuracies with the unannotated Bengali, Hindi and Telegu test sets.

1 Hidden Markov Model based POS tagging

A POS tagger based on Hidden Markov Model (HMM) (Jurafsky and Martin, 2000) 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 (Viterbi, 1967). The task of Part of Speech (POS) tagging is to find the sequence of POS tags $T = \{t_1, t_2, t_3, \ldots t_n\}$ that is optimal for a word sequence $W = \{w_1, w_2, w_3 \ldots w_n\}$. The tagging problem becomes equivalent to searching for $\text{argmax}_T P(T) * P(W \mid T)$, by the application of Bayes' law.

The probability of the tag i.e., P (T) can be calculated by Markov assumption which states that the probability of a tag is dependent only on a small, fixed number of previous tags. Here, in this work, a tri-gram model has been used. So, the probability of a tag depends on two previous tags, and then we have,

$P(T) = P(t_1) * P(t_2 \mid t_1) * P(t_3 \mid t_1, t_2) * P(t_4 \mid t_2, t_3) * \ldots * P(t_n \mid t_{n-2}, t_{n-1})$

An additional tag '$' (dummy tag) has been introduced in this work to represent the beginning of a sentence. So, the previous probability equation can be slightly modified as:

$P(T) = P(t_1 \mid \$) * P(t_2 \mid \$, t_1) * P(t_3 \mid t_1, t_2) * P(t_4 \mid t_2, t_3) * \ldots * P(t_n \mid t_{n-2}, t_{n-1})$

Due to sparse data problem, the linear interpolation method has been used to smooth the tri-gram probabilities as follows:

$P'(t_n \mid t_{n-2}, t_{n-1}) = \lambda_1 P(t_n) + \lambda_2 P(t_n \mid t_{n-1}) + \lambda_3 P(t_n \mid t_{n-2}, t_{n-1})$ such that the λs sum to 1.

The values of λs have been calculated by the following method (Brants, 2000):

set $\lambda_1 = \lambda_2 = \lambda_3 = 0$

for each tri-gram t_1, t_2, t_3 with $\text{freq}(t_1, t_2, t_3) > 0$

depending on the maximum of the following three values:

case : $\frac{freq(t_1,t_2,t_3)-1}{freq(t_1,t_2)-1}$: increment λ_3 by $freq(t_1, t_2, t_3)$

case : $\frac{freq(t_2,t_3)-1}{freq(t_2)-1}$: increment λ_2 by $freq(t_1, t_2, t_3)$

case : $\frac{freq(t_3)-1}{N-1}$: increment λ_1 by $freq(t_1, t_2, t_3)$

end

end

normalize $\lambda_1, \lambda_2, \lambda_3$

Here, N is the corpus size, i.e., the number of tokens present in the training corpus. If the denominator in one of the expression is 0, we define the result of that expression to be 0. The -1 in both the numerator and denominator has been considered for taking unseen data into account.

By making the simplifying assumption that the relation between a word and its tag is independent of context, we can simplify P (W | T) as $P(W \mid T) \approx P(w_1 \mid t_1) * P(w_2 \mid t_2) * \ldots * P(w_n \mid t_n)$

The emission probabilities in the above equation can be calculated from the training set as:

Emission Probability: $P(w_i \mid t_i) = \frac{freq(w_i, t_i)}{freq(t_i)}$

There have been enormous attempts for English POS tagging employing machine-learning techniques like transformation-based error-driven learning (Brill, 1995), decision trees (Black et al., 1992), maximum entropy methods (Ratnaparakhi, 1996), conditional random fields (Lafferty, 2001) etc. In Indian languages, particularly in Hindi, the works on POS tagging and chunking can be found in (Singh et. al., 2006) and (Singh et. al., 2005) respectively.

1.1 Context Dependency

To make the Markov model more powerful, additional context dependent features have been introduced to the emission probability in this work that specifies the tag probability of the current word depends on the tag of the previous word and the tag to be assigned to the current word. Now, P (W | T) is calculated by the equation: $P(W \mid T) \approx P(w_1 \mid \$, t_1) * P(w_2 \mid t_1, t_2) * \ldots * P(w_n \mid t_{n-1}, t_n)$

So, the emission probability can be calculated as $P(w_i \mid t_{i-1}, t_i) = \frac{freq(t_{i-1}, t_i, w_i)}{freq(t_{i-1}, t_i)}$

Here also the smoothing technique is applied rather than using the emission probability directly. The emission probability is calculated as: $P'(w_i \mid t_{i-1}, t_i) = \theta_1 P(w_i \mid t_i) + \theta_2 P(w_i \mid t_{i-1}, t_i)$, where θ_1, θ_2 are two constants such that all θs sum to 1. The values of θs should be different for different words. But the calculation of θs for every word takes a considerable time and hence θs are calculated for the entire training corpus. In general, the values of θs can be calculated by the same method that was adopted in calculating λs.

Now, the emission probability and transition probability have been joined together to set up the modified hidden Markov model.

1.2 Handling the Unknown Words

Handling of unknown words is an important issue in POS tagging. Viterbi algorithm attempts to assign a POS tag to unknown words. Specifically, three different approaches have been adopted to take care of the unknown words in Bengali.

1.2.1. Unknown Word Features

For words, which have not been seen in the training set, $P(w_i \mid t_i)$ is estimated based on features of the unknown words, such as whether the word contains a particular suffix. The list of suffixes has been prepared for Bengali. At present there are 435 suffixes; many of them usually appear at the end of verb, noun and adjective words. A null suffix is also kept for those words that have none of the suffixes in the list. The probability distribution of a particular suffix with respect to specific POS tags is generated from all words in the training set that share the same suffix. Apart from suffix analysis, two other features have been included that tackle tokens of digits and symbols.

1.2.2. Named Entity Recognition

For handling unknown words, a pattern based bootstrapping method has been used to develop a Named Entity Recognition (NER) system in Bengali. The Bengali training set is searched to find the named entities. This can be done by finding out the tags: NNPC and NNP. The NER algorithm generates a lexical pattern *p* using a context window of width 4 around each occurrence of the named entities (NEs) in the training set like: $p = [l_{-2}\, l_{-1} <T> l_{+1}\, l_{+2}]$, where T is NNPC* NNP and $l_{\pm i}$ are context POS tags. The width of the lexical patterns may vary depending upon the presence of the SYM POS tag. The new patterns are generated from these patterns using boot-

strapping. All these patterns form the set of potential patterns, denoted by P. Every pattern p in the set 'P' is matched against the test set. In a position, where the context of *p* matches, the system predicts where one boundary of a name in the text would occur. During pattern matching against the test set, the maximum length of a named entity is considered to be of six words. If the identified NEs are found to be unknown words, then the NNPC and NNP tags are assigned accordingly to these words in the output.

1.2.3. Lexicon Features

To handle the unknown words further, a lexicon consisting of approximately 20000 entries has been used. The lexicon has been developed in a semi-supervised way from a tagged Bengali news corpus, developed from the web. Lexicon contains the root words and their basic part of speech information such as: noun, verb, adjective, adverb, pronoun and indeclinable. If an unknown word is found in the lexicon, then the corresponding part of speech information is extracted and the basic POS tag is assigned to the words in the output.

2 Chunker

For the chunker, the rule-based approach has been used. The chunking algorithm is divided into two phases, namely: Chunk boundary identification and Chunk labeling.

2.1 Chunk Boundary Identification

To identify the chunks, it is necessary to find and mark the positions where a chunk can end and a new chunk can begin. The POS tag assigned to every token by the POS tagger is used to discover these positions. The chunk boundaries are identified by some handcrafted linguistic rules that check whether two neighboring POS tags belong to the same chunk or not. If they do not, then a chunk boundary is assigned in between the words.

2.2 Chunk Labeling

After chunk boundary identification the chunks are labeled. The components (i.e., POSs of tokens) within a chunk help to assign the label on the chunk.

3 Results and Discussions

The POS tagger is trained separately on the Bengali, Hindi and Telegu training data. The POS tagger has been tested on the development test sets of Bengali, Hindi and Telegu. After testing, the development test sets are included as part of the training sets to evaluate the system with the unannotated tests sets of Bengali, Hindi and Telegu. The POS tagger is able to assign tags to all the words in test sets and hence the *precision* and *recall* figures of the POS tagger are same and have been considered as the *accuracy* of the tagger. The training and test sets statistics for the three different languages have been presented in Table 1 and Table 2. The Following information are provided in the table: No. of tokens in the training set (NTTR), No. of tokens in the development and unannotated test sets (NTTST), No. of unknown tokens in the test sets (UTST) and the percentage of unknown tokens in the test set.

Language Type	NTTR	NTTST	UTST	UTST (in %)
Bengali	20396	5022	1002	19.96
Hindi	21470	5681	1132	19.93
Telegu	21415	6098	3411	55.94

Table 1: Training and development test sets statistics

Language Type	NTTR	NTTST	UTST	UTST (in %)
Bengali	25418	5225	1726	33.04
Hindi	27151	4924	1416	28.76
Telegu	27513	5193	2375	45.74

Table 2: Training and unannotated test sets statistics

Test Set	CTT	Accuracy (in %)
Bengali	4564	90.9
Hindi	4661	82.05
Telegu	3898	63.93

Table 3: POS tagger Results (Development sets) (CTT: No. of tokens correctly tagged by the POS tagger)

Test Set	CTT	Accuracy (in %)
Bengali	4061	77.73
Hindi	4924	76.87
Telegu	3505	67.49

Table 4: POS tagger Results (Unannotated sets)

The POS tagger has been tested with all three different development test sets and their results have been reported in Table 3. The POS tagger is then tested with three different unannotated test sets and their results have been presented in Table 4. Here, different development test sets have

been included as part of the training sets. It is observed from Table 3 and Table 4 that the POS tagger performs best for Bengali development and unannotated test sets. The key to this higher accuracy, compared to Hindi and Telegu, is the mechanism of handling of unknown words. Unknown word features, named entity recognizer and lexicon features are used to cope with the unknown words in the Bengali development test data. On the other hand, the system cannot handle the unknown words problem in Hindi and Telegu at present. POS tagger performs better with the Hindi compared to Telegu. The presence of large number of unknown words in the Telegu test sets (development and unannotated test sets) and the agglutinative nature of the language are the main reasons behind the fall in accuracy. The POS tagger produces 73.17% accuracy for the Bengali unannotated test set with the unknown word features only. The accuracy increases to 77.73% with the inclusion of named entity recognition system and lexicon features. The accuracy of the POS tagger falls from 90.9% to 77.73% for the unannotated test set as it contains more unknown words than the development test set.

Error analysis of the POS tagger for the Bengali development test set has been done with the help of a confusion matrix. A close scrutiny of the confusion matrix suggests that some of the probable tagging errors facing the current POS tagger are NNC vs NN, JJ vs NN, JJ vs JVB, VFM vs VAUX and VRB vs NN. A multiword extraction unit for Bengali would have taken care of the NNC vs NN problem. The other ambiguities can be taken care of with the use of linguistic rules.

The rule-based chunking system is designed for Bengali and the same chunker is then applied for Hindi and Telegu test data. After removing the chunk labels and chunk boundary markers (keeping the POS tags intact) of the manually chunked corpus (development test sets), the chunker is evaluated and the obtained results are presented in Table 5 and Table 6 for the development test sets and unannotated test sets respectively. Following abbreviations are used in this table: No. of chunks in the test set (NOCT), No. of correctly identified chunks (CIC) and Chunk accuracy (CA).

Test Set	NOCT	CIC	CA (in %)
Bengali	3915	3362	85.88
Hindi	2757	2037	73.88
Telegu	3556	1990	55.96

Table 5: Chunking Results (Development sets)

Test Set	NOCT	CIC	CA (in %)
Bengali	5225	4213	80.63
Hindi	4924	3528	71.65
Telegu	5193	2760	53.15

Table 6: Chunking Results (Unannotated sets)

The accuracy figures of the chunking results for Hindi and Telegu test sets are not satisfactory as the rules of the chunker were mainly designed for Bengali. These accuracy figures can be increased by deriving the rules of chunking for Hindi and Telegu language.

References

Adwait Ratnarparakhi. 1996. *A Maximum Entropy Part-Of –Speech Tagger,* EMNLP, 1996.

Brants T. 2000. *TnT a statistical parts-of-speech tagger*. In Proceedings of the sixth conference on applied natural language processing ANLP-2000, 2000, Seattle, WA, 224-231.

Eric Brill. 1995. *Transformation-Based Error Driven Learning and Natural Language Processing: A Case Study in Part-Of-Speech Tagging*. Computational Linguistics 21(94): 543-566. 1995.

E. Black, F. Jelinek, J. Lafferty, R. Mercer and S. Roukos. 1992. *Decision tree models applied to labeling of text with part-of speech.* In DARPA Workshop on Speech and Natural Language. Harriman, NY.

John Laffertey, Andrew McCallum, Fernando Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* In Proceedings of the 18th International Conference on Machine Learning, 2001.

Jurafsky D. and Martin J. H. 2000. *Speech and Language Processing*. Prentice-Hall.

Viterbi A. J. 1967. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*. IEEE Transaction on Information Theory, IT-13(2), 260-269.

Singh, A., S. Bendre and R. Sangal. *HMM Based Chunker for Hindi*. In Proceedings of the IJCNLP-05:126-131.

Singh, Smriti, K. Gupta, M. Shrivastava and P. Bhattacharyya. *Morphological Richness offsets Resources Demand-Experiences in Constructing a POS Tagger for Hindi*. In Proceedings of COLING -ACL-2006, Sydney, Australia.

Part-of-Speech Tagging and Chunking with Maximum Entropy Model

Sandipan Dandapat
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
India 721302
sandipan@cse.iitkgp.ernet.in

Abstract

This paper describes our work on Part-of-speech tagging (POS) and chunking for Indian Languages, for the SPSAL shared task contest. We use a Maximum Entropy (ME) based statistical model. The tagger makes use of morphological and contextual information of words. Since only a small labeled training set is provided (approximately 21,000 words for all three languages), a ME based approach does not yield very good results. In this work, we have used a morphological analyzer to improve the performance of the tagger. The tagger uses variety of features, which works effectively not only for Bengali, but also for other Indian Languages such as Hindi and Telugu. The chunker makes use of only language independent contextual information.

The work primarily focuses on the development of a Bengali POS tagger and chunker. Further, we have transported the model for Hindi and Telugu to understand the relative performance of the statistical model. The tagger has the overall accuracy on development data of about 88% for Bengali, and about 83% and 68% for Hindi and Telugu respectively. The best accuracy achieved for chunking by our method on the development data are 84.45%, 79.88% and 65.92% for Bengali, Hindi and Telugu respectively on per word basis.

1 Introduction

Part-of-Speech (POS) tagging for natural language texts are developed using linguistic rules, stochastic models and a combination of both (hybrid taggers). Stochastic models (Cutting, 1992; Merialdo, 1994; Brants, 2000) have been widely used in POS tagging task for simplicity and language independence of the models. Among stochastic models, Hidden Markov Models (HMM) are quite popular. Development of a stochastic tagger requires large amount of annotated text. Stochastic taggers with more than 95% word-level accuracy have been developed for English, German and other European Languages, for which large labeled data is available. The problem is quite difficult for Indian Languages due to lack of such large annotated corpus.

Simple HMM models do not work well when small amount of labeled data are used to estimate the model parameters. Incorporating a diverse set of overlapping features in a HMM-based tagger is difficult and complicates the smoothing typically used for such taggers. In contrast, a ME based methods (Ratnaparkhi, 1996) can deal with diverse, overlapping features.

After POS tagging, the next step is chunking, which divides sentences into non recursive inseparable phrases. It can serve as the first step for full parsing. The task of identifying chunk boundaries and chunk labels is modeled in the same way as of identifying POS tags.

In this paper, we present a ME based statistical POS tagger and chunker for Indian Languages. We have developed separate models for POS tagging and chunking task based on ME principle, which can be used to tag unlabeled text.

2 Our Approach

In our work, we use a Maximum Entropy (ME) based statistical model. The ME tagger and chunker is based on Ratnaparkhi(1996)'s POS tagger. We are using a Morphological Analyzer (R. Maitra, 2004) for Bengali in our POS tagging technique. The Morphological Analyzer takes a word as input and gives all possible POS tags for the word. This in turn restricts the set of possible tags for a given word.

The ME model estimates the probabilities based on the imposed constraints. Such constraints are derived from the training data, maintaining some relationship between features and outcomes. The tagger uses model of the form:

$$P(t \mid h) = \frac{1}{Z(h)} \exp\left(\sum_{j=1}^{n} \lambda_j f_j(h,t) \right) \qquad (1)$$

Where, t is tag, h is the context and the $f_j(h,t)$ are the *features* with associated weight λj.

The problem of POS tagging can be formally stated as follows. Given a sequence of words $w_1 \ldots w_n$, we want to find the corresponding sequence of tags $t_1 \ldots t_n$, drawn from a set of tags T, which satisfies:

$$P(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1,n} P(ti \mid h_i) \qquad (2)$$

Where, h_i is the context for word w_i. The Beam Search algorithm is used find the most probable sequence given the sentence.

The features are binary valued functions which associate a tag with various elements of the context; for example:

$$f_j(h,t) = \begin{cases} 1 & \text{if word(h)=Ami and t=PRP} \\ 0 & \text{othetrwise} \end{cases}$$

General Iterative Scaling is used to estimate the value of the weights. We have used the Java-based OpenNLP maximum entropy package[1].

3 Features

Feature selection plays a crucial role in ME framework. Experiments were carried out to find out the most suitable features for the POS tagging and chunking task.

3.1 POS Tagging Features

The main features for the POS tagging task have been identified based on the different possible combination of available word and tag context. The features also include prefix and suffix for all words. The term prefix/suffix is a sequence of first/last few characters of a word, which does not mean a linguistically meaningful prefix/suffix. The use of prefix and suffix information works well for highly inflected languages. We considered different combination from the following set for inspecting the best feature set for POS tagging task:

$$F = \left\{ w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}, t_{i-1}, t_{i-2}, \left| pre \right| \le 4, \left| suf \right| \le 4 \right\}$$

[1] http://maxent.sourceforge.net

Forty different experiments were conducted taking several combinations from set 'F' to identify the best suited feature set for POS tagging task. From our empirical analysis we found that the combination of a context window of size three, prefixes and suffixes gives the best performance for the tagging task. It is interesting to note that the inclusion of prefix and suffix for all words gives better result instead of using only for rare words. This can be explained by the fact that due to small amount of annotated data, a significant number of instances are not found for most of the word of the language vocabulary. Table 1 lists the features used in our maximum entropy model.

Condition	Features
Static features for all words	Current word(w_i) Previous word (w_{i-1}) Next word (w_{i+1}) |prefix| ≤ 4 |suffix| ≤ 4
Dynamic Features for all words	POS tag of previous word (t_{i-1})

Table 1: Features used for POS tagging

3.2 Chunking Features

The main features used for the chunking task are listed in Table 2. The most suitable features for chunking are found in the window size of five (POS tags of previous and next two words, previous two chunk categories and current word with its POS tag). Contextual words do not yield good result in the chunking task; as only a small amount of annotated corpora is available for training.

Condition	Features
Static features for all words	Current word(w_i) POS tag of the current word(t_i) POS tags of previous two words (t_{i-1} and t_{i-2}) POS tags of next two words (t_{i+1} and t_{i+2})
Dynamic Features for all words	Chunk tags of previous two word (C_{i-1} and C_{i-2})

Table 2: Features used for chunking

4 Experiments

4.1 Experiments for POS Tagging

We define the baseline model as the one where the tag probabilities depend only on the current word:

$$P(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1,n} P(t_i \mid w_i) \qquad (3)$$

In this model each word in the test data will be assigned the tag which occurred most frequently for that word in the training data.

The POS tagger has been implemented based on ME model (described in section 2) to find the most probable tag sequence for a given sequence of words. The models use a set of 27 tags (T) for every word in a sentence and the most probable tag sequence is determined using the Beam Search algorithm using equation 2.

In order to further improve the tagging accuracy, we integrate morphological information with the ME model. We assume that the POS-tag of a word w can take values from the set $T_{MA}(w)$, where $T_{MA}(w)$ is computed by the Morphological Analyzer. Note that the size of $T_{MA}(w)$ is much smaller than T. Thus, we have a restricted choice of tags as well as tag sequences for a given sentence. Since the correct tag t for w is always in $T_{MA}(w)$ (assuming that the morphological analyzer is complete), it is always possible to find out the correct tag sequence for a sentence even after applying the morphological restriction. Due to a much reduced set of possibilities, this model performs better even when only small labeled training text is available. We shall call these new models **ME + MA**.

All parts of the training text are used for the experiments. The most probable tag sequence is estimated based on equation (2). The models ME+MA uses a morphological analyzer to restrict the possible tags of a word, but in reality it is possible that the morphological analyzer is neither complete nor sound. Two different experiments are carried out to understand the relative performance of the tagger under two situations: (1) when the morphological restriction is neither sound nor complete (we shall call this ME + IMA) and (2) when the morphological restriction is sound and complete (we shall call this ME + CMA). If the word is unknown to the morphological analyzer (in case of ME+IMA), we assume the POS-tags of that word belongs to the open class grammatical categories (all classes of Noun, Verb, Adjective, Adverb and Interjection). The experiment with morphological restriction on tags has been conducted only for Bengali. The experiments with ME+MA model are not conducted for Hindi and Telugu due to unavailability of the desired morphological analyzer for the languages.

4.2 Experiments for Chunking

Our experiment for chunking has been conducted in the same way as of implementing POS tagger. The most probable chunk labels are identified based on the equation (2) using the features described in section 3.2. Our chunker is heavily loaded with the POS tags of the context words hence, experiments were conducted to understand the performance of the chunker under the situation: (1) when the POS labels are correct (we shall call this ME+C_POS) and (2) when the POS labels are according to the output of our system (we shall call this ME+I_POS).

4.3 Data Used for the Experiments

The ME model is trained from the annotated text corpus (provided during the competition[2]). The training data has been manually annotated using different number of POS tags and chunk labels for different languages.

In order to measure the performance of the system we use annotated test data (development data set and test data set in two phases) for each language. Table 3 summarizes the size of the data, the number of POS and chunk categories used for the language.

Language	Bengali	Hindi	Telugu
Training data (in words)	20,396	21,470	21,416
Development data (in words)	5,023	5,681	6,098
Test data (in words)	5,226	4,924	5,193
No. of POS tags	27	25	25
No. of Chunk labels	6	7	6

Table 3: Data size and tags used for the languages

5 System Performance

5.1 Tagging Accuracy for the Development Data Set

We define the tagging accuracy as the ratio of the correctly tagged words to the total number of words. Table 4 summarizes the final accuracies achieved for three different languages in the development set. Note that the baseline model has an accuracy of 58.88% for Bengali and 68.93% for Hindi.

[2] The training data includes only the data provided for training for the three different languages.

Method	Bengali	Hindi	Telugu
Baseline	58.88	68.93	-
ME	79.74 (89.3, 60.5)	83.10 (90.9,53.7	67.82 (82.570.0)
ME + IMA	83.51 (84.2, 82.1)	-	-
ME + CMA	88.25 (89.3, 86.2)	-	-

Table 4: Tagging accuracies (in %) of different models for different languages in the form of *Overall Accuracy (Known Word Accuracy, Unknown Word Accuracy)*.

It is interesting to note that the use of morphological restriction (ME + CMA) gives an improvement of around 9% over the ME model for Bengali. Even the incomplete morphological analyzer (ME + IMA) boosts up the performance of the model by around 4%. We have used simple ME model for Hindi and Telugu due to unavailability of desired morphological analyzer for the languages. The tagger has an accuracy of 83.10% for Hindi and 67.82% for Telugu.

Another significant observation is that the accuracy for Telugu is quite less compare to Bengali and Hindi. It has been noticed that the accuracy for known and unknown words for Telugu are 82.48% and 65.77% respectively. But the percentage of unkown words (87%) is much higher in the Telugu development data set compare to Bengali and Hindi.

5.2 Chunking Accuracy for the Development Data Set

The accuracy of the chunker has been evaluated under two situations: (1) per words basis and (2) number of correctly identified chunk groups along with the number of correctly identified chunk labels for the correctly identified chunk groups (per chunk basis). Table 5 reports the accuracy obtained by the system under two different assumptions as described in section 4.2. In the above mentioned second case, the accuracies are represented in the form of *Correctly Identified Chunks, Correctly Labeled Chunks.*

Evaluation Criteria	Method	Bengali	Hindi	Telugu
Per word basis	ME + I_POS	84.45	79.88	65.92
Per chunk basis	ME + I_POS	87.3,80.6	74.1,67.4	69.6,56.7
	ME + C_POS	93.3,87.7	78.5,74.4	-

Table 5: Chunking accuracies (in %) of different models for different languages in the development set.

6 Tagging and Chunking Accuracy for the Test Data Set

The test data set has been tagged using ME + IMA for Bengali and simple ME model for Hindi and Telugu as described in 4.1. The experiment for chunking for the test data was conducted on the POS tagged output of the test data set using the model described in section 3.2. Table 6 summarizes the final accuracies of the POS tagger and chunker for the test data set.

Language	POS Tagging Accuracy	Chunking Accuracy
Bengali	77.61	80.59
Hindi	75.69	74.92
Telugu	74.47	68.59

Table 6: POS Tagging and Chunking accuracies (in %) for different languages for the test data set

7 Conclusion

In this report we have described an approach for automatic stochastic tagging of natural language text. The models described here are very simple and efficient for automatic tagging even when the amount of available labeled text is small. The models have a much higher accuracy than the naïve baseline model. However, the performance of the current system is not as good as that of the best POS-taggers available for English and other European languages. The performance for Hindi and Telugu can be improved once the morphological analyzers for the languages are available.

References

T. Brants. 2000. TnT – A statistical part-of-sppech tagger. In *Proc. of the 6th Applied NLP Conference,* pp. 224-231.

D. Cutting, J. Kupiec, J. Pederson and P. Sibun. 1992. A practical part-of-speech tagger. In *Proc. of the 3rd Conference on Applied NLP*, pp. 133-140.

R. Maitra. 2000. Inflectional Morphological Analyzers for Hindi and Bengali Languages. M. Tech. Thesis. Dept. of CSE. Indian Institute of Technology Kharagpur.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155-171.

A. Ratnaparkhi. 1996. A Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the EMNLP conference*, pp. 133-142.

POS Tagging and Chunking using Decision Forests

Sathish Chandra Pammi
Language Technologies Research Center
IIIT, Hyderabad, India
sathishp@students.iiit.net

Kishore Prahallad
Language Technologies Institute
Carnegie Mellon University, USA
skishore@cs.cmu.edu

Abstract

This paper presents the building of POS Tagger and Chunk Tagger using Decision Forests and also focuses on the investigation towards exploring different methods for parts-of-speech tagging of Indian languages using sub-words as units. The two models POS Tagger and Chunk Tagger were tested with 3 different Indian languages Hindi, Bengali, Telugu and achieved the accuracies as 69.92%, 70.99%, 74.74% and 69.35%, 60.08%, 77.20% respectively.

1 Introduction

Annotated corpora serve as an important tool for investigators of natural language processing, speech recognition and other related areas. It proves to be a basic building block for constructing statistical models for automatic processing of natural languages. For example in Prosody added Speech Synthesis(Alan W Black et al., 1997) Parts of speech sequenses are very important.

Today most of the taggers can be characterized as Rule-based or statistical based. To distinguish the tag ambiguity, Rule-based taggers(Eric Brill, 1992) use hand-written rules. Stochastic based taggers use the probabilities of occurrences of words for a particular tag. Since Indian languages are morphologically rich in nature, developing rule based taggers which is a cumbersome process. But, stochastic taggers require large amount of annotated data to train upon.

Many POS Taggers use morphological analyzer as a module in their tagger. But building morphological analyzer to a particular Indian language is very difficult. So, we tried to capture similar information in indirect way by splitting the word to be tagged in to sub-words.

Part-of-speech (POS) tagging involves many difficult problems, such as insufficient amounts of training data, inherent POS ambiguities, and most seriously, many types of unknown words which are ubiquitous in any application and cause major tagging failures in many cases. The investigation towards exploring various methods to resolve these types of problems by entering to sub-word units like syllable, phoneme and onset vowel coda are presented in this paper.

The next section describes the properties of the Indian Language Scripts. Section 3 will concentrate on the data base used for the POS tagging. Section 4 deals with Feature Selection of POS Tagging and Chunking. Section 5 explains about the Classification and Regression Tree (CART)(Breiman et al., 1984), and its parameters. Section 6 details about the Decision forest. Section 7 explains different experiments conducted and their results of POS Tagging and Chunking leading to conclusion at section 8 by providing overall results.

2 Indian Language Scripts

Indian languages have essentially a common alphabet, though they use different forms to express. A character in Indian language scripts is close to a syllable and can be typically of the form: C*VC*, where C and V are consonant and vowel respectively. Indic languages are partially syllabic in nature, and the inherent vowel is an important concept.

The scripts of Indian language have originated from the ancient Brahmi script, where basic units are syllabic in nature. Syllable is typically in the form C*VC*. We can again subdivide syllable

into onset, vowel and coda. In a syllable, 'V' in between the two C*'s is vowel, C*'s that are left and right of the vowel are onset and coda respectively.

For example, the following word written in IT3 notation describes Sub-Word units of Indian languages.

Word: /san:gharshha/

Syllables: /san:/, /gha/, /rshha/

Onset-Vowel-Coda of syllable /san:/ is /s/-/a/-/n:/, /gha/ is /gh/-/a/-/#/ and /rshha/ is /rshh/-/a/-/#/

Phonemes: /s/, /a/, /n:/, /gh/, /a/, /r/, /shh/, /a/

All the Indian languages have a common phonetic base. It has been observed that in some Indian languages it is quite difficult to get good performance with rule-based system. This is because of the large number of exceptions for a particular rule. To handle this problem, statistical approaches such as Decision Trees are employed. In this paper we tried to show how the performance of an advanced machine learning concept, Decision Forest is applied to POS tagging.

3 Data Used for the Experiments

Manual annotated data for Hindi, Bengali and Telugu is available at (spsal, 2007). A limited number of training set around 20000 words for each language is available.

Table 1: POS Tagging Database

	Hindi	Bengali	Telugu
NW for training	21446	20352	21393
NW for testing	4925	5226	5194
Tags	25	27	24

NW - Total number of words
Tags - Number of unique tags

4 Feature Selection

Many experiments were conducted on word level as well as sub-word level with different possible features. Basic experiments are conducted on 3-levels of sub-words. They are 1) Syllable level features, 2) phoneme level features and 3) onset, vowel and coda level (OVC) features.

But we found that below given set of features of sub-words significantly participated for POS Tagging:

- First syllable, last two syllables and remaining phonemes of present word (F1).
- First syllable, last two syllables of present word, previous word and it's last 2 syllables and next word and it's first syllable (F2).
- First 5 and last 5 Onset, vowel and coda (OVC) of present word, last 2 OVC's of previous word and first 2 OVC's of next word (F3).
- First syllable, last two syllables and their onsets of present word, Last 2 syllables and their onsets of previous word and first syllable and its onset of next word (F4).
- Previous word and their last 2 syllables, next word and its first syllable (F5).

For chunking we have used 2-tag scheme (Akshay Singh et al., 2005). Features used for chunking are 2-level context of POS Tags. i.e. present, previous, previous-previous, next and next-next word POS Tags were used as features for chunking.

5 Classification and Regression Trees

Classification And Regression Tree is a decision-tree procedure introduced in (Breiman et al., 1984) . CART uses an exhaustive, recursive partitioning routine to generate binary splits that divide each parent node into two child nodes by posing a series of yes-no questions. CART searches for questions that split nodes into relatively homogenous child nodes. As the tree evolves, the nodes become increasingly more homogenous, identifying segments. The basic CART building algorithm is a greedy algorithm which chooses the locally best discriminatory feature at each stage in the process.

5.1 Stop Parameter

The stop parameter specifies the minimum number of samples necessary in the training set before a question is hypothesized to distinguish the group. Normally with smaller stop value the model may become over-trained. The optional stop value may differ for different datasets of different languages.

5.2 Predictee

In a given feature set, the feature that is to be predicted as the output, is termed as the predictee. By default the first feature in the feature-set is taken as the predictee, but we can always specify the predictee while giving the description of the data.

Some times CART is over-fit with training data. Thus performance may reduce. While further exploring different techniques to improve the performance of this module, we came across the decision forest algorithm, which can avoid over-fitting with bagging.

6 Decision Forests

A decision forest is a set of several decision trees which is similar to Random forest(Breiman L, 2001). These trees can be formed by various methods (or by one method, but with various parameters of work), by different sub-samples of observations over one and the same phenomenon, by using different characteristics. Such many-sided consideration of a problem, as a rule, gives the improvement of quality of forecasting and a better understanding of laws of the researched phenomenon. Let us consider a set of trees and an observation x. Each tree gives a forecast for x. Using a voting method, a class attributed to observation x is a class which is preffered by majority of trees. In the regression analysis problem, the predicted value is a mean of forecasts of all trees.

Decision tree forest models often have a degree of accuracy that cannot be obtained using a large single-tree model. It uses the 'out of bag' data rows for validation of the model. This provides an independent test without requiring a separate data set or holding back rows from the tree construction. The stochastic (randomization) element in the decision tree forest algorithm makes it highly resistant to over fitting.

The general decision forest while building the tree employ randomness both in preparing the datasets as well as in selection of the features. But in the decision forest algorithm used here, we have employed the concept of randomness only while building the datasets.

While using the decision forest, the focus will be on the following two things : a) Datasets, b) Algorithm used for building the trees.

6.1 Preparing Datasets

The datasets are manipulated for employing the decision tree concept. The underlying concept of building datasets from a given training dataset is as follows: Take a random sample of N vectors from the data set by replacement. Some observations will be selected more than once, and others will not be selected. About 2/3 of the rows will be selected by the sampling. The remaining 1/3 of the rows are called the out of bag (OOB) rows. A new random selection of rows is performed for each tree constructed.

6.2 Bagging

Bagging (Bootsrap AGGregatING)(Breiman L, 1996) produces replications of the training set by sampling with replacement. Each replication of the training set has the same size as the original set, but some examples can appear more than once while others dont appear at all. Bagging should only be used if the learning machine is unstable. Bagging proves quite useful here Since decision trees are one of such type. Bagging improves the estimate if the learning algorithm is unstable and reduces the variance of predictions without changing the bias

6.3 Building Forest

With each of the datasets thus created, a decision tree is built. Here we have chosen the CART (explained in the previous section) algorithm for building the decision tree. After building the trees for each dataset, the trees are then used with the testdata set for predicting the output of each of the vectors of the testset. Thus with the obtained sets of results, the decision forest predicts the exact output by a voting strategy.

6.4 Voting Strategy

In decision forest, the final output is chosen based on voting strategy, that will weigh the most popular output from the set of outputs. Given the several outputs, a weightage-by-count method is employed to assign weightage to each output. Then the output with the highest weightage is recorded as the final output.

7 Experiments

7.1 POS Tagging

Using bagging, several trees were constructed with stop value 1. Each feature set described in

Section 4 builds their corresponding forest. By using voting strategy forest annotates given data. Table-2 shows results of Tagger using decision forest.

Table 2: Performance of POS Tagging using Decision Forest (in %)

Features	Hindi	Bengali	Telugu
F1	67.80	59.40	73.76
F2	64.91	59.80	73.72
F3	63.47	54.31	75.81
F4	63.21	53.12	74.22
F5	61.33	51.70	71.63

Because CART is less immune to overfitting, we used Decision Forest for POS Tagging. Decision Forest using sub-word features contributed significantly to annotate data. At last Voting strategy is applied to above model and input is a 5 featured set simultaneously. This strategy gets the output of each feature set and chooses the result which occurs with the highest frequency. Finally we achieved performance 69.35%, 60.08% and 77.20% for Hindi, Bengali and Telugu respectively.

7.2 Chunking

In the feature set discussed in section 4 for chuncking , the CART and the Decision Forest are giving percentages as shown in the table-3.

Table 3: Performance of Chunking using CART and Decision Forest (in %)

Method	Hindi	Bengali	Telugu
CART	69.11	69.46	73.04
Decision Forest	69.92	70.99	74.74

8 Conclusion

Since only a small labeled training set is available (around 20,000 words), sub-word unit based approach gives significantly good results. The two models POS Tagger and Chunk Tagger were tested with 3 different Indian languages Hindi, Bengali, Telugu and achieved the accuracies as 69.92%, 70.99%, 74.74% and 69.35%, 60.08%, 77.20% respectively.

References

Breiman L., J. Friedman, R. Olshen, and C. Stone 1984. *Classification and Regression Trees*, Wadsworth and Brooks Pacific Grove CA.

Leo Breiman 1996. *Bagging Predictors*, Machine Learning, 24:123-140

Breiman L 2001. *Random Forests*, Machine Learning, 45:5-15

Eric Brill 1992. *A simple rule-based part of speech tagger*, Proceedings of the Third Annual Conference on Applied Natural Language Processing, ACL

Alan W Black and Paul Taylor 1997. *Assigning Phrase Breaks From Part-of-Speech Sequences*, Proceedings of the Eurospeech.

Akshay Singh, S M Bendre, Rajeev Sangal 2005. *HMM Based Chunker for Hindi*, Proceedings of International Joint Conference on Natural Language Processing.

Daniel Jurafsky and James H. Martin 2000. *Speech and Language Processing*, Prentice Hall PTR

http://shiva.iiit.ac.in/SPSAL2007/index.php 2007 *Workshop on Shallow Parsing in South Asian Languages*

POS tagging and Chunking for Indian Languages

Himanshu Agrawal
Department of Computer Science
IIIT - Hyderabad
agrawal@students.iiit.ac.in

Abstract

In this paper we present our approach for a part of speech tagger and chunker for South Asian Languages. We have used a Conditional Random Fields based approach to train the system on the corpus made available by the SPSAL workshop at ICJAI 2007. We have worked on improving the machine's learning without using any language specific tools like dictionaries, morphological analyzers etc. Apart from the annotated training data we also use a large raw unannotated text . The average performance figures over all the three languages are 79.13% for POS tagging and 92.36% for chunking over the 3 languages. The highest being 84.90 % for Hindi.

1. Introduction

POS Tagging is the task of assigning grammatical classes to words in a natural language sentence. Similarly chunking consists of dividing a text in syntactically correlated parts of words. For example, the sentence, "He reckons the current account deficit will narrow to only # 1.8 billion in September." can be divided as follows:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in [NP September] .

Identifying the POS tags and chunk tags for the words in a given text is an important aspect in any language processing task. Both are important intermediate steps for full parsing.

As a part of the IJCAI workshop on Shallow Parsing for South Asian Languages, the task was to build a POS tagger and chunker for three South Asian Languages Hindi,Telugu and Bengali. The training consisted of 21470,21425, 20397, words and the development data consisted of 5682, 6098, 5357 words respectively.

Conditional Random Fields were first used for the task of shallow parsing by Lafferty et. al.(2001), where CRFs were applied for NP chunking for English on WSJ corpus and reported a performance of 94.38%. For Hindi CRF's were first applied to shallow parsing by Ravindran et. al.(2006) and Himanshu et. al.(2006) for POS tagging and chunking, where they reported a performance of 89.69% and 90.89% respectively for chunking.

The use of unannotated data for POS tagging was reported by Ratnaparkhi et. al.(1996) where rare word information was used during training with maximum entropy.

Several POS taggers using supervised learning both over word instances and tagging rules report precision greater than 96% for English. For Hindi and other South Asian languages, the tagged copora is limited and together with higher complexity of these languages it poses a difficulty in achieving results as good as those achieved for English in the past.

One way to improve the performance of POS tagging systems is to identify the problems these systems suffer from and use language specific details to work for each of these problems separately for each language. This would mean rigorous and in depth language analysis for each language. Another way could be to work with the machine's learning and devise approaches and guidelines for an overall improvement in the training. Because such an improvement is not language specific, it should mean an improvement for all languages. In this paper we concentrate on improving the learning and also explore the use of unannotated raw data to help the learning and tagging process.

2. Conditional Random Fields

The model

Charles Sutton *et. al (2005)." Let G be a factor graph over Y . Then p(y|x) is a conditional random field if for any fixed x, the distribution p(y|x) factorizes according to G. Thus, every conditional distribution p(y|x) is a CRF for some, perhaps trivial, factor graph. If F = {A } is the set of factors in G, and each factor takes the exponential family form (1.3), then the conditional distribution can be written as*

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_A \in G} \exp\left\{ \sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak}(\mathbf{y}_A, \mathbf{x}_A) \right\}.$$

". In addition, practical models rely extensively on parameter tying. For example, in the linear-chain case, often the same weights are used for the factors t (yt , yt−1 , xt) at each time step. To denote this, we partition the factors of G into C = {C1 , C2 , . . . CP }, where each Cp is a clique template whose parameters are tied. This notion of clique template generalizes that in Taskar et al. [2002], Sutton et al. [2004], and Richardson and Domingos [2005]. Each clique template Cp is a set of factors which has a corresponding set of sufficient statistics {fpk (xp , yp)} and parameters p ∈ K (p) . Then the CRF can be written as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p),$$

where each factor is parameterized as

$$\Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) = \exp\left\{ \sum_{k=1}^{K(p)} \lambda_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) \right\},$$

and the normalization function is

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p).$$

As here, in a linear-chain conditional random field, typically one clique template
C = {t (yt , yt−1 , xt)}T=1 is used for the entire network.

$$C = \{\Psi_t(y_t, y_{t-1}, \mathbf{x}_t)\}_{t=1}^{\mathrm{T}}$$

3. Approach

A. POS Tagging

The training for pos tagging involved training the corpora on the Gold Standard Tagset consisting of 26 tags. We use the following feature template for training.

<Word -2>,
<Word -1>,
<Word 0>,
<Word 1>,
<Word 2>
<Last 2 characters>,
<Last 3 characters>,
<Last 4 characters>.
<Presence of special characters>
<Word Length>

This model performed at 82.95 % accuracy and served as a baseline model for subsequent approaches.

Upon our error analysis we found that, if we classify the gold standard tags into 5 broad families viz Nouns, Verbs, Adjectives ,Adverbs and Noise: All the errors could be classified into two categories

1. Errors, where the machine's predicted tag belonged to the same family as that of the annotated tag. Ex. NNP going to NN.

2. Errors, where the machine's predicted tag did not belong to the same family as that of the annotated tag. Ex NNP going to JJ.

We devised two separate approaches for reducing the number of errors of each type.

Approach for reducing errors of type 1.

Building a *knowledge database* to re-tag over the machine's output.

The knowledge database is a database of words and their respective gold standard POS tags seen in the training data. To add more more words to the database we trained the system on the 5 broad families. Using this training, we then tagged the entire 1,50,000 words of the unannotated data and extracted the words for which the tagging confidence was very high. These words were then added to the database with the gold standard POS tags of their family. For example if a word X was tagged as Adjective, X was added to the database with JJ and JVB as its prospective tags.

We use this database to limit the final POS tagging of a word to within its set of prospective gold Standard tags.

This approach improved the performance by 1.95%, taking it to 84.9%.

Approach for reducing errors of type 2.

A "*fineness followed coarseness*" approach.

This approach uses a two stage mechanism. In the first stage we train the system over the 5 broad families, ie Noun, Verb, Adverb, Adjective and Noise. In the second stage the system is trained over each of these families to tag the word with its gold standard POS tag.

This approach proved to be a failure. It decreased the performance to 73.33%

B. Chunking

We followed the approach used by Himanshu et. al.(2006), where the corpus is trained separately for chunk labels and chunk boundaries with each using the other's information. The corpus is first trained over B-L (Boundary-Label) classes. Chunk Labels are extracted from here and subsequently added to the corpus as training information for training over B (chunk boundary). This approach helps the learning incorporate useful information like, "the changing of chunk labels must induce a change of boundary". And although it is not conversely true, training for chunk labels using boundary information has shown to produce improvement over not.

The training features we used are as follows

<Word -2>
<Word -1>
<Word 0>
<Word 1>
<Word 2>
<POS tag -2>
<POS tag -1>
<POS tag 0>
<POS tag 1>
<POS tag 2>
<POS tag -1>/<POS tag 0>
<POS tag -0>/<POS tag 1>

4. Results

Without following any methodology to reduce the number of errors the system performed at 82.95%. Using the approach for reducing errors of type 1, we registered an improvement of 1.95%, taking the performance to 84.90%.

	Hindi POS Tagging
Baseline	82.95 %
Approach 1	84.9 %
Approach 2	73.33 %

However approach 2, ie. the "*fineness followed coarseness*" approach for reducing errors of type 2 failed to improve. Where we registered a performance of 94.77% in the first stage of identifying family tags, in the second stage it performed at only 79.33 %, taking the overall performance of the system to as low as 73.33%.

We figure that the reason for this failure is that: Although the first stage looks promising, which is only because the number of tags was reduced to just 5. The second stage fails to perform very well because the algorithm is unable to learn the language patterns in absence of the rest of the tags and jittered progression/flow of the text.

Thus it is very important to train on a tagset which is able to mark the language patterns and tag sequences. They may later be transformed to the standard pos tagset in a post processing step.

This is also corroborated in Akshay et. al.(2005), where for the task of identifying chunk boundary markers, he reports an improvement with a coarsening followed by fineness approach. He uses a 4 tag scheme (start_chunk, continue_chunk, stop_chunk, start_stop_chunk) to train and generalizes it to a standard 2 tag scheme (start_chunk, continue_chunk) in a post processing step.

Overall the results for all the three languages are as follows.

	Hindi	***Telugu***	***Bengali***
POS Tagging	84.90 %	71.22 %	81.09 %
Chunking	92.69 %	91.77 %	94.90 %

5. Conclusion

Training must be done with a tagset which is best able to mark the sequence-instance-class relationship and differentiate the properties of one tag from another, while keeping a balance on the number of tags. By and large it has been seen that using a more specific and larger tagset for training and transforming it to a standard, smaller tagset post processing helps. The vice versa however generally multiplies the errors.

6. References

1] Charles Sutton, An Introduction to Conditional Random Fields for Relational Learning

[2] Adwait Ratnaparkhi ,1998, Maximum Entropy Models For Natural Language Ambiguity Resolution, Dissertation in Computer and Information Science,University Of Pennslyvania,1998.

[3] Akshay Singh, Sushma Bendre, Rajeev Sangal, 2005 ,HMM Based Chunker for Hindi, IIIT hyderabad.

[4] Pranjal Awasthi, Delip Rao, Balaraman Ravindran,2006. Part Of Speech Tagging and Chunking with HMM and CRF. Proceedings of the NLPAI MLcontest workshop, National Workshop on Artificial Intelligence.

[5] Thorsten Brants. 2000. TnT - A Statistical Part-of-Speech Tagger Proceedings of the sixth conference on Applied Natural Language Processing (2000) 224–231.

[6] Himanshu Agrawal, Anirudh Mani 2006, Part Of Speech Tagging and Chunking Using Conditional Random Fields: Proceedings of the NLPAI MLcontest workshop, National Workshop on Artificial Intelligence.

[7] CRF++: Yet Another Toolkit
Copyright © 2005 Taku Kudo
http://chasen.org/~taku/software/CRF++/