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*