HMM Based Chunker for Hindi

Akshay Singh
IIIT
Hyderabad, India
akshay@students.iiit.ac.in

Sushma Bendre
Dept. of Mathematics and Statistics
University of Hyderabad
Hyderabad, India
smbsm@uohyd.ernet.in

Rajeev Sangal
IIIT
Hyderabad, India
sangal@iiit.ac.in

Abstract

This paper presents an HMM-based chunk tagger for Hindi. Various tagging schemes for marking chunk boundaries are discussed along with their results. Contextual information is incorporated into the chunk tags in the form of part-of-speech (POS) information. This information is also added to the tokens themselves to achieve better precision. Error analysis is carried out to reduce the number of common errors. It is found that for certain classes of words, using the POS information is more effective than using a combination of word and POS tag as the token. Finally, chunk labels are also marked on the chunks.

1 Introduction

1.1 Motivation and Problem Statement

A robust chunker or shallow parser has emerged as an important component in a variety of NLP applications. It is employed in information extraction, named entity identification, search, and even in machine translation. While chunkers may be built using handcrafted linguistic rules, these tend to be fragile, need a relatively long time to develop because of many special cases, and saturate quickly. The task of chunking is ideally suited for machine learning because of robustness and relatively easy training.

A chunker or shallow parser identifies simple or non-recursive noun phrases, verb groups and simple adjectival and adverbial phrases in running text. In this work, the shallow parsing task has been broken up into two subtasks: first, identifying the chunk boundaries and second, labelling the chunks with their syntactic categories.

The first sub-problem is to build a chunker that takes a text in which words are tagged with part of speech (POS) tags as its input, and marks the chunk boundaries in its output. Moreover, the chunker is to be built by using machine learning techniques requiring only modest amount of training data. The second sub-problem is to label the chunks with their syntactic categories.

The presented work aims at building a chunker for Hindi. Hindi is spoken by approximately half a billion people in India. It is a relatively free word order language with simple morphology (albeit a little more complex than that of English). At present, no POS taggers or chunkers are available for Hindi.

1.2 Survey of Related Work

Chunking has been studied for English and other languages, though not very extensively. The earliest work on chunking based on machine learning goes to (Church K, 1988) for English. (Ramshaw and Marcus, 1995) used transformation based learning using a large annotated corpus for English. (Skut and Brants, 1998) modified Church's approach, and used standard HMM based tagging methods to model the chunking process. (Zhou,et al., 2000) continued using the same methods, and achieved an accuracy of 91.99% precision and 92.25% recall using a contextual lexicon.

(Kudo and Matsumoto, 2001) use support vec-

tor machines for chunking with 93.48% accuracy for English. (Veenstra and Bosch, 2000) use memory based phrase chunking with accuracy of 91.05% precision and 92.03% recall for English. (Osborne, 2000) experimented with various sets of features for the purpose of shallow parsing.

In this work, we have used HMM based chunking. We report on a number of experiments showing the effect of different encoding methods on accuracy. Different encodings of the input show the effect of including either words only, POS tags only, or a combination thereof, in training. Their effect on transition probabilities is also studied. We do not use any externally supplied lexicon. Analogous to (Zhou,et al., 2000), we found that for certain POS categories, a combination of word and the POS category must be used in order to obtain good results. We report on detailed experiments which show the effect of each of these combinations on the accuracy. This experience can also be used to build chunkers for other languages. The overall accuracy reached for Hindi is 92.63% precision with 100% recall for chunk boundaries.

The rest of the paper is structured as follows. Section 2 discusses the problem formulation and reports the results of some initial experiments. In Section 3, we present a different representation of chunks which significantly increased the accuracy of chunking. In Section 4, we present a detailed error analysis, based on which changes in chunk tags are carried out. These changes increased the accuracy. Section 5 describes experiments on labelling of chunks using rule-based and statistical methods.

2 Initial Experiments

Given a sequence of words $W^n = (w_1, w_2, \cdots, w_n)$, $w_i \in \mathcal{W}$, where $\mathcal{W}$ is the word set and the sequence of corresponding part of speech (POS) tags $T^n = (t_1, t_2, \cdots, t_n)$, $t_i \in \mathcal{T}$ where $\mathcal{T}$ is the POS tag set, the aim is to create most probable chunks of the sequence W^n. The chunks are marked with chunk tag sequence $C^n = (c_1, c_2, \cdots, c_n)$ where c_i stands for the chunk tag corresponding to each word w_i, $c_i \in \mathcal{C}$. $\mathcal{C}$ here is the chunk tag set which may consist of symbols such as STRT and CNT for each word marking it as the start or continuation of a chunk. In our experiment, we combine the corresponding words and POS tags to get a sequence of new tokens $V^n = (v_1, v_2, \cdots, v_n)$ where $v_i = (w_i, t_i) \in \mathcal{V}$. Thus the problem is to find the sequence C^n given the sequence of tokens V^nwhich maximizes the probability

$$P(C^n|V^n) = P(c_1, c_2, \cdots, c_n|v_1, v_2, \cdots, v_n), \quad (1)$$

which is equivalent to maximizing $P(V^n|C^n)P(C^n)$.

We assume that given the chunk tags, the tokens are statistically independent of each other and that each chunk tag is probabilistically dependent on the previous k chunk tags ($(k+1)$-gram model). Using chain-rule, the problem reduces to that of Hidden Markov Model (HMM) given by

$$\max_{c_i \in \mathcal{C}} \prod_{i=1}^{n} P(v_i|c_i)P(c_{i+k}|c_i, \cdots, c_{i+k-1}) \quad (2)$$

where the probabilities in the first term are emission probabilities and in the second term are transition probabilities. The optimal sequence of chunk tags can be found using the Viterbi algorithm. For training and testing of HMM we have used the TnT system (Brants, 2000). Since TnT is implemented up to a tri-gram model, we use a second order HMM ($k = 2$) in our study.

Before discussing the possible chunk sets and the token sets, we consider an example below.

```
(( sher )) (( hiraN  ke    pIche ))
   lion       deer   of    behind
   NN         NN     PREP  PREP
   STRT       STRT   CNT   CNT

(( jangal  meM )) (( bhAgA  . ))
   forest  in        ran    .
   NN      PREP      VB     SYM
   STRT    CNT       STRT   CNT
```

In this example, the chunk tags considered are STRT and CNT where STRT indicates that the new chunk starts at the token which is assigned this tag and CNT indicated that the token which is assigned this tag is inside the chunk. We refer to this as 2-tag scheme. Under second-order HMM, the prediction of chunk tag at i^{th} token is conditional on the only two previous chunk tags. Thus in the example, the fact that the chunk terminates at the word `pIche` (behind) with the POS tag PREP is not captured in tagging the token `jangal` (forest). Thus, the assumptions that the

tokens given the chunk tags are independent restricts the prediction of subsequent chunk tags. To overcome this limitation in using TnT, we experimented with additional chunk tags.

We first considered a 3-tag scheme by including an additional chunk tag STP which indicates end of chunk. It was further extended to a 4-tag scheme by including one more chunk tag STRT_STP to mark the chunks which consist of a single word. A summary of the different tag schemes and the tag description is given below.

1. 2-tag Scheme: {STRT, CNT}
2. 3-tag Scheme: {STRT, CNT, STP}
3. 4-tag Scheme: {STRT, CNT, STP, STRT_STP}

where tags stand for:

- STRT: A chunk starts at this token
- CNT: This token lies in the middle of a chunk
- STP: This token lies at the end of a chunk
- STRT_STP: This token lies in a chunk of its own

We illustrate the three tag schemes using part of the earlier example sentence.

```
     (( sher )) (( hiraN  ke    pIche))...
        lion       deer   of    behind ...
        NN         NN   PREP    PREP   ...
2-tag:  STRT       STRT  CNT    CNT    ...
3-tag:  STRT       STRT  CNT    STP    ...
4-tag:  STRT_ST    STRT  CNT    STP    ...
```

We further discuss the different types of input tokens used in the experiment. Since the tokens are obtained by combining the words and POS tags we considered 4 types of tokens given by

1. Word only
2. POS tag only: Only the part of speech tag of the word was used
3. Word_POStag: A combination of the word followed by POS tag
4. POStag_Word: A combination of POS tag followed by word.

Note that the order of Word and POS tag in the token might be important as the TnT module uses suffix information while carrying out smoothing of transition and emission probabilities for sparse data. An example of the Word_POStag type of tokens is given below.

```
    ((sher ))((hiraN    ke      pIche))...
      lion     deer     of      behind...
      NN       NN       PREP    PREP...
Token:sher_NN  hiran_NN ke_PREP pIche_PREP
2-tag:STRT     STRT     CNT     CNT ...
```

The annotated data set contains Hindi texts of 200,000 words. These are annotated with POS tags, and chunks are marked and labelled (NP, VG, JJP, RBP, etc). This annotated corpus was prepared at IIIT Hyderabad from funds provided by HP Labs. The POS tags used in the corpus are based on the Penn tag set. Hewever, there are a few additional tags for compound nouns and verbs etc.

Out of the total annotated data, 50,000 tokens were kept aside as unseen data. A set of 150,000 tokens was used for training the different HMM representations. This set converted into the appropriate format based on the representation being used. 20,000 tokens of the unseen data were used for development testing.

Table 1: Initial Results of Chunking (% Precision)

	Word	POS	POS Word	Word POS
2 Tags	79.21	80.32	81.42	81.85
3 Tags	75.30	71.99	77.05	77.80
4 Tags	70.41	68.25	72.59	73.64
3 Tags	75.30	71.99	77.05	77.80
4→3 Tags	76.95	74.65	78.78	79.56
2 Tags	79.21	80.32	81.42	81.85
3→2 Tags	81.14	79.66	82.58	83.30
4→2 Tags	83.37	82.41	84.89	85.60

The initial results using various tag sets and token sets are presented in Table 1. The first three rows show the raw scores of different tagging schemes. To compare across the different schemes, the output were converted to the reduced chunk tag sets which are denoted by 4→3, 4→2 and 3→2 in the table. This ensures that the measurement metric is the same no matter which tagging scheme is used, thus allowing us to compare across the tagging schemes. The last three rows show the result of using

It should be noted that converting from the 4 tag set to 3 or 2 tags results in no loss in information. This is because it is trivial to convert

fromt the 2-tag set to the corresponding 4-tag set and vice-versa. Even though the information content in the 3 different chunk tag representations is the same, using higher tag scheme for training and then later converting back to 2-tags results in a significant improvement in the precision of the tagger. For example, in the case where we took 'Word_POSTag' as the token, using 4-tag set the original precision was 73.64%. When precision was measured by reducing the tag set to 3 tags, we obtained a precision of 79.56%. Four tags reduced to two gave the highest precision of 85.6%. However, these differences may be interpreted as the result of changing the measurement metric. This figure of 85.6% may be compared with a precision of 81.85% obtained when the 2-tag set was used. Recall in all the cases was 100%.

3 Incorporating POS Context in Output Tags

We attempted modification of chunk tags using contextual information. The new output tags considered were a combination of POS tags and chunk tags using any one of the chunk tag schemes discussed in the earlier section. The new format of chunk tags considered was POS:ChunkTag, which is illustrated for 2-tag scheme in the example below.

```
    (( sher )) (( hiraN    ke      ...
       lion       deer     of      ...
       NN         NN       PREP    ...
Token: sher_NN    hiran_NN ke_PREP...
2-tag: NN:STRT    NN:STRT  PREP:CNT...
```

The tokens (V) were left unchanged. Our intention in doing this was to bring in a finer degree of learning. By having part of speech information in the chunk tag, the information about the POS-tag of the previous word gets incorporated in the transition probabilities. In the earlier chunk schemes, this information was lost due to the assumption of independence of tokens given chunk tags. In other words, part of speech information would now influence both the transition and emission probabilities of the model instead of just the emission probabilities.

We carried out the experiment with these modified tags. Based on the results in Table 1 for various tokens, we restricted our choice of tokens to Word_POStags only. Also, while combining POS tags with chunk tags, the 4-tag scheme was used. The accuracy with 4-tag scheme was 78.80% and for $4 \rightarrow 2$ scheme, it turned out to be 88.63%. This was a significant improvement.

4 Error Analysis and Further Enhancements

We next carried out the error analysis on the results of the last experiment. We looked at which type of words were resulting in the maximum errors, that is, we looked at the frequencies of errors corresponding to the various part of speech. These figures are given in Table 2. On doing this analysis we found that a large number of errors were associated with NN (nouns), VFM (finite verbs) and JJ (adjectives). Most of these errors were coming in possibly because of sparsity of the data. Hence we removed the word information from these types of input tokens and left only the POS tag. This gave us an improved precision of 91.04%. Further experiments were carried out on the other POS tags. Experiments were done to see what performed better - a combination of word and POS tag or the POS tag alone. It was found that seven groups of words - PRP, QF (quantifiers), QW, RB (adverbs), VRB, VAUX (auxillary verbs) and RP (particles) performed better with a combination of word and POS tag as the token. All the other words were replaced with their POS tags.

Table 2: Error Analysis I - With Word_POSTag as the Token and POSTag:ChunkTag as Output Tag

POS Tag	Total Errors	Total Tokens	% Error
NN	1207	4063	29.71 %
VFM	459	2108	21.77 %
SYM	420	2483	16.92 %
PRP	402	1528	26.31 %
JJ	260	911	28.54 %
PREP	237	2526	9.38 %
NNP	142	389	36.50 %
RP	129	589	21.90 %

An analysis of the errors associated with punctuations was also done. It was found that the set of punctuations { ! : ? , ' } was better at marking chunks than other symbols. Therefore, these punctuations were kept in the tokens while the

other symbols were reduced to a common marker (SYM).

After performing these steps, the chunker was tested on the same testing corpus of 20,000 tokens. The precision achieved was 92.03% with a recall of 100% for the development testing data. Table 3 gives the stepwise summary of results of this experiment. The first coloumn of the table gives different token sets described above. Error

Table 3: Stepwise Summary of Results for Identifying Chunk Boundaries

Method	Precision (4 Tags)	Precision ($4 \rightarrow 2$)
Adding POS Context info	78.80	88.63
Reducing NN, JJ, RP to POS only	83.14	91.04
Limiting word info. to 7 POS groups	84.02	91.79
Limiting punctuation marks to {! , : ? '}	84.03	92.03

analysis of this experiment is given in Table 4. On comparing with Table 2, it may be seen that the number of errors associated with almost all the POS types has reduced significantly, thereby resulting in the improved precision.

Table 4: Error Analysis II

POS Tag	Total Errors	Total Tokens	% Error
NN	557	4063	13.71%
VFM	200	2108	9.49%
JJ	99	911	10.87%
PRP	84	1528	5.50%
SYM	79	2483	3.18%
RP	64	589	10.87%
CC	61	748	8.16%
QFN	59	310	19.03%

5 Chunk Labels

Once the chunk boundaries are marked, the next task is to classify the chunk. In our scheme there are 5 types of chunks - NP (noun phrase), VG (verb group), JJP (adjectival phrase) RBP (adverbial phrase) and BLK (others). We tried two methods for deciding chunk labels. One was based on machine learning while the other was based on rules.

5.1 HMM Based Chunk Labelling

In this method, the chunk boundary tags are augmented with the chunk labels while learning. For example, the tags for the last token in a chunk could have additional information in the form of the chunk label.

```
    (( sher )) (( hiraN    ke
       lion       deer     of
       NN         NN       PREP
Token: sher_NN    hiran_NN ke_PREP
2-tag: NN:STRT#NP NN:STRT  PREP:CNT

       pIche )) (( jangal    meM )) ...
       behind      forest    in
       PREP        NN        PREP
Token: pIche_PREP  jangal_NN meM_PREP
2-tag: PREP:CNT#NP NN:STRT   PREP:CNT#NP
```

Three schemes for putting chunk labels in the tags were tried.

- Scheme 1: The token at the start of the chunk was marked with the chunk label.
- Scheme 2: All the tokens were marked with the chunk labels.
- Scheme 3: The token at the end of the chunk was marked with the chunk label. (See example above.)

The best results were obtained with scheme 3, which when reduced to the common metric of 2-tags only gave a precision of 92.15% (for chunk boundaries only) which exceeded the result for chunk boundaries alone (92.03%). The accuracy for scheme 3 with the chunk boundaries and chunk labels together was 90.16%. The corresponding figures for scheme 1 were 91.70% and 90.00%, while for scheme 2 they were 92.02% and 88.05%.

5.2 Rules Based Chunk Labels

Since there are only five types of chunks, it turns out that the application of rules to find out the chunk-type is very effective and gives good results. An outline of the algorithm used for the purpose is given below.

- For each chunk, find the last token t_i whose POS does not belong to the set {SYM, RP, CC, PREP, QF}. (Such tags do not help in classifying the chunks.)

- If t_i is a noun/pronoun, verb, adjective or adverb, then label the chunk as NP, VG, JJP or RBP respectively.
- Otherwise, label the chunk as BLK.

In our experiments, we found that over 99% of the chunks identified were given the correct chunk labels. Thus, the best method for doing chunk boundary identification is to train the HMM with both boundary and syntactic label information together (as given in Section 6.1). Now given a test sample, the trained HMM can identify both the chunk boundaries and labels. The chunk labels are then dropped to obain data marked with chunk boundaries only. Now rule based labelling is applied (with an accuracy of over 99%) yielding a precision of 91.70% (test set) for the composite task.

Table 5: Summary of Chunk Labelling

Method	Prec-1	Prec-2
HMM with label at the start of the chunk	91.70	90.00
HMM with chunk labels for all the tokens	92.02	88.05
HMM with label at the end of the chunk	92.15	90.16
HMM with label at the end of the chunk (test set)	92.63	91.70

Prec-1 - Precision for Chunk Boundaries
Prec-2 - Precision for Chunk Boundaries and Chunk Labels

6 Conclusions

In this paper, we have studied HMM based chunking for Hindi. We tried out several schemes for chunk labels and input tokens. We found that for a certain type of words (function words), word information along with POS information gave better precision. A similar differentiation was done for punctuations. We tried several methods to classify the chunks and found that a simple rule-based approach gave the best results. The final precision we got was 92.63% for chunk boundary identification task and 91.70% for the composite task of chunk labelling with a recall of 100%.

This paper raises the issue that if there are two tag sets T_1 and a more finely differentiated set T_2, then T_2 might give better accuracy than T_1, provided the errors are measured using the same metric (say, using the T_1 set). This, we believe, is likely to happen, when T_2 is more finely and appropriately differentiated. The most striking example was where T_1 consisted of chunk boundaries and T_2 consisted of boundaries and labels. Training with T_2 outperformed T_1 for the boundary task, even though it did not perform very well in the labelling task.

References

Steven Abney. 1996. Tagging and Partial Parsing. *Corpus-Based Methods in Language and Speech. Kluwer Academic Publishers, Dordrecht* (1996)

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

K. Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of Second Conference on Applied Natural Language Processing* (1988) 136–143

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. *Proceedings of NAACL 2001* (2001) 1013–1015

Miles Osborne 2000. Shallow Parsing as Part-of-Speech Tagging. *Proceedings of CoNLL-2000*. (2000)

Lance A. Ramshaw, and Mitchell P. Marcus. 1995. Text Chunking Using Transformation-Based Learning. *Proceedings of the 3rd Workshop on Very Large Corpora* (1995) 88–94

W. Skut and T. Brants 1998. Chunk Tagger, Statistical Recognition of Noun Phrases. *ESSLLI-1998* (1998)

Zhou, GuoDong, Jian Su and TongGuan Tey 2000. Hybrid Text Chunking. *Proceedings of CoNLL-2000 and LLL-2000* (2000) 163–165.

Jorn Veenstra and Antal van den Bosch 2000. Single-Classifier Memory-Based Phrase Chunking. *Proceedings of CoNLL-2000 and LLL-2000* (2000) 157–159.