A Three Stage Hybrid Parser for Hindi

Sanjay Chatterji, Arnab Dhar, Sudeshna Sarkar, Anupam Basu
Department of Computer Sc. & Engineering, Indian Institute of Technology, Kharagpur, India
Email: {schatt,arnabdhar,sudeshna,anupam}@cse.iitkgp.ernet.in

ABSTRACT

The present paper describes a three stage technique to parse Hindi sentences. In the first stage we create a model with the features of head words of each chunk and their dependency relations. Here, the dependency relations are inter-chunk dependency relations. We have experimentally fixed a feature set for learning this model. In the second stage, we extract the intra-chunk dependency relations using a set of rules. The first stage is combined with the second stage to build a two-stage word level Hindi dependency parser. In the third stage, we formulate some rules based on features and used them to post-process the output given by the two-stage parser.

1 Introduction

Parsing a sentence can be considered as finding the dependency relations of some pair of words in a sentence. The words must be related in such a way that they form a leveled tree structure where nodes are words, edges are assigned between the pairs of words which are related and levels are name of relations. Leveled tree structures for a set of sentences are referred to as dependency Treebank.

In Hindi dependency tree, the groups of words which are related by intra-chunk dependency relations occur as adjacent nodes. Each such group is referred to as chunk. The Hindi Treebank released in MTPIL – 2012 shared task contains the chunk information (boundary, tag, head) and dependency relations between the constituents of the chunk (intra-chunk dependency relations) and between chunks (inter-chunk dependency relations).

Instead of considering each word as a node, we reduce Hindi dependency tree structure by considering each chunk as a node in a statistical parser. However, this tree structure does not contain the intra-chunk relations. Intra-chunk relations are identified using a rule based approach.

We have compared this chunk level statistical parsing followed by the rule based intra-chunk relation identification stage (two-stage technique) with the word level statistical parsing where each word is considered as a node. Finally, we have formulated some rules based on the list of Hindi specific constraints and used them to improve the relations assigned by the two-stage technique. Each stages of this three-stage parsing are evaluated using the official CoNLL-07 shared task evaluation script eval07.pl.

2 Related Work

Some work has been done on using Paninian framework for parsing Hindi sentences. Bharati et al. (2002) have developed a Hindi parser by translating Hindi grammatical constraints to integer programming constraints. Bharati et al (2009) described a two-stage constraint based hybrid approach to dependency parsing. In the first stage they have handled the intra-clausal dependencies and in the second stage they have handled the inter-clausal dependencies. They have also shown the affect of hard constraints (H-constraints) and soft constraints (S-constraints) to build an efficient and robust hybrid Hindi parser. Their two-stage parser outperforms the data driven parsers.

Some work has been done on the rule based postprocessing on the baseline data driven dependency parser. Ambati et al (2010) described transition-based approach to Hindi dependency parsing and analyzed the output of data driven Hindi parsers to observe the role of different morphosyntactic features in Hindi dependency parsing. They have shown 10 experiments for selecting the best feature for Hindi chunk level data driven dependency parsing using MaltParser. Gadde et al (2010) described a data driven dependency parsing which uses clausal information of a sentence for improving parser performance.

Husain et al (2009) proposed a modular cascaded approach to data driven dependency parsing. Each module or layer leading to the complete parse produces a linguistically valid partial parse. They did this by introducing an artificial root node in the dependency structure of a sentence and by catering to distinct dependency label sets that reflect the function of the set internal labels at different layers. Output (partial parse) from each layer is accessed independently.

3 Our Work

Word level dependency Treebank contains a large number of relations and words (tokens). This big tagset can be divided into two parts namely inter-chunk tags and intra-chunk tags. We hypothesize that the statistical parser trained and tested on lesser tagset performs better than the parser trained and tested on larger tagset. We wish to establish this hypothesis with the help of word level and chunk level parsing experiments in the same experimental setups and system configurations.

Finding the chunks of a sentence is a non-trivial task. Rather when chunks are given, the intra-chunk relations between the head and children nodes can be identified using a small set of rules. Therefore, we wish to identify the inter-chunk relations statistically and intra-chunk relations using rules.

3.1 Flow chart

The flow chart of the proposed three stage system is shown in Figure 1. Inputs to these three stages of this system are Feature set, Rule-set1, and Rule-set2, respectively. These three stages are executed serially in the system.

3.2 First Stage

In the first stage chunks are extracted from the training, development and test data to build the chunk level training, development and test data. This data contains head of the chunk, its features and dependency relation between chunk heads. The names of the children of the chunk and their features are stored in a database named ChildDB.

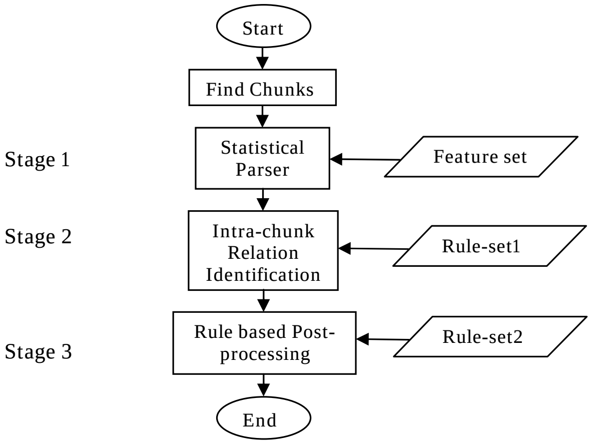

Figure1: Flow chart of the proposed system

A model is learned using the chunk level training data and the feature set calculated experimentally. Each chunk in this chunk level Treebank contains following attributes.

1. HWORD: Head word of chunk.
2. HROOT: Root of head word of chunk.
3. HPOS: Part-of-speech tag of head word.
4. CHUNK: Name of Chunk.
5. HMORPH: Morphological features of head word: gender, number, person, case, suffix, type, and voice.
6. DEPREL: Dependency relation of that chunk with another chunk.

We have applied the Covington's algorithm using the MaltParser of Nivre (2006, 2007, 2009) for building the model and testing. In this algorithm, partially processed chunks are stored in stacks and unprocessed chunks are stored in buffer. The chunks in the buffer are annotated based on the features of processed and unprocessed tokens. We have experimentally selected following set of features for training the model.

1. A set of HWORD features over stack and buffer of length 2.
2. A set of HROOT features over stack and buffer of length 2.
3. A set of HPOS features over stack and buffer of length 4.
4. A set of CHUNK features over stack and buffer of length 1.
5. A set of HMORPH features over stack and buffer of length 2.
6. A set of HWORD features over dependents and head of length 1.
7. A set of HPOS features over dependents and head of length 1.

8. A set of CHUNK features over dependents and head of length 1.
9. A set of DEPREL features over dependents of length 1.
10. A set of combinations of the HWORD and HPOS features of length 1.

The data driven chunk level parser built using these features achieves accuracy of 81.44% (Label Attachment Score).

3.3 Second Stage

The second stage takes as input the tree structures with the relations between chunk heads as given by the first stage parser and the features of the chunks as stored in the ChunkDB. The children and the head words are combined and their intra-chunk relations are identified using Rule-set1. There are two types of rules in Rule-set1 namely rules for identifying the relations between heads and children (head-child relation) and rules for identifying the relations between two children (child-child relation). The rules are in the following format.

1. Head-child Rule: FH FC Rel
2. Child-child Rule: FC1 FC2 Rel

According to first type of rule, if feature of head is FH and feature of child is FC then child is related to head with Rel relation. Similarly, according to second type of rule, if feature of first child is FC1 and feature of second child is FC2 then first child is related to second child with Rel relation. We have identified 84 head-child rules and 61 child-child rules. The data driven chunk level parser followed by the rule based intra-chunk relation identification stage is referred to as two-stage parser. This parser achieves accuracy of 89.36% (Label Attachment Score).

3.4 Third Stage

In the third stage, we analyze the mistakes in dependency relations assigned by the two-stage parser. The development data is used for this analysis. Based on this analysis a set of Hindi specific constraints are identified for some verb roots. These constraints are used to correct the mistakes made by the above two-stages. The constraints are stored using demand frames. The demand frames we use in our task contain 4 fields namely relation name, necessity of the relation, features of the dependent node of that relation and features of the head node of that relation.

A root of a head node can have a single demand frame with each row representing a dependency relation from a child. We build 12 demand frames for this rule based post-processing task. A relation in the demand frame for Hindi verb घुसना is shown using Table 1. According to this demand frame, the noun chunk having postposition में is related to verb chunk घुस आया था with k7 dependency relation.

Dependency relation	Necessity	Feature of Dependents	Feature of Head
k7	D	cat-n|vib-0_में	cat-v|vib-0_आ+या_था

Table - 1: A relation in the demand frame for the Hindi verb घुसना.

K7: Adhikaran, D: Desirable, cat: lexical category, vib: vibhakti/suffix.

Applying these constraints on the two-stage Hindi parser we build a three stage parser for Hindi and achieve accuracy of 89.45% (Label Attachment Score).

4 Evaluation

To find the optimal feature set we have experimented on the features in data driven parser. The same feature set is used for parsing both the chunk level and word level parsing. The chunk level parsing is then combined with the second and third stages to build the three-stage parser (Parser1). Again, the word level parsing is combined with the third stage to build a hybrid parser (Parser2). Parser1 and Parser2 are compared to find the effect of our approach.

4.1 Experimental Setup

The Hindi Treebank used in this task is provided by the organizers of the MTPIL -2012 Shared Task. Three parts of the Treebank and their sizes are shown below.

- Training corpus: 12041 sentences, 268,093 words
- Development Corpus: 1233 sentences, 26416 words
- Test Corpus: 1828 sentences, 39775 words

We have used MaltParser of Nivre (2006, 2007, 2009) for training the models and using them for statistically annotating dependency relations between words or chunks in Hindi sentences.

4.2 Parsing Performance

The chunk boundary and the head words of chunk are annotated in the Treebank. We have used this information to build the chunk level Treebank. There are three evaluation tasks namely the performance of the chunk level parsing, the performance of the two-stage parsing and the performance of the constraint based post-processing on the two-stage parsing (three-stage parsing). We have shown these three results in first three columns of Table 2.

	Chunk Level Parsing	**Two-stage Parsing**	**Three Stage Parsing**	**Word Level Parsing**	**Hybrid Parsing**
LAS	81.44 %	89.36 %	89.45 %	89.23 %	89.34 %
UAS	91.77 %	95.08 %	95.17 %	94.83 %	94.98 %
LA	84.26 %	91.19 %	91.25 %	91.17 %	91.23 %

Table – 2: Three Stage Parser Evaluation Results.
LAS: Label Attachment Score, UAS: Unlabeled Attachment Score, LA: Label Accuracy

We have also used word level dependency relation tagged training corpus for training a model. The performance of this model is tested on word level dependency relations in the test data. The performance of this model and that of the constraint based post-processing on the word level parsing (hybrid parsing) are also shown in Table 2.

Comparing the performances of three-stage parser and hybrid parser, we observed that the two-stage word level parsing performs better than the word level (single stage) statistical parsing. Finally, the constraint based post-processing stage applied on the two-stage parser achieves higher accuracy compared to the constraint based post-processing stage applied on the statistical parser.

4.3 Efficiency

There are total 92 dependency relations among which 74 are inter-chunk relations and 18 are intra-chunk relations. The word level data driven dependency parser uses 92 tags and 268,093 words of the training corpus to train the model. This parser uses 39775 words to test the model. Here each word can be considered as nodes.

On the other hand, the chunk level data driven dependency parser uses 74 tags and 141431 chunks of the training corpus to train the model. This parser uses 11165 chunks to test the model. Here each chunk can be considered as nodes.

The time taken by the word level and the chunk level data driven dependency parsers to train and test the models are shown in Table 3. The time taken to execute the second and third stages of our approach can be negligible.

	Training Time	Test Time
Word Level	211:41:38	6:3:46
Chunk Level	27:11:54	3:11:13

Table – 3: Times taken by the word level and chunk level data driven dependency parsers in the form of (hour: minute: second).

5 Conclusion

A three-stage hybrid framework for parsing Hindi sentences is presented in this paper. In the first stage a data driven parser is used to identify the inter-chunk dependency relations. The head of each chunk is used as node in this tree structure. Children of each chunk along with the corresponding dependency relation are added into the tree in the second stage. This two-stage parsing system is efficient and performs better than the word level data driven dependency parser. In the third stage we correct the mistakes made by the two-stage parser using a set of Hindi constraints. Some more constraints can be included experimentally to achieve more accurate Hindi parser.

Acknowledgement

This work is partially supported by the ILMT project sponsored by TDIL program of MCIT, Govt. of India. We would like to thank all the members in Communication Empowerment Lab, IIT Kharagpur.

Reference

Ambati, B. R., Husain, S., Nivre, J. and Sangal, R. (2010) *On the Role of Morphosyntactic Features in Hindi Dependency Parsing.* In Proceedings of NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA.

Bandyopadhyay, S. and Ekbal, A. (2006). *HMM Based POS Tagger and Rule-Based Chunker for Bengali.* In Proceedings of the Sixth International Conference on Advances In Pattern Recognition: pp. 384-390, Kolkata.

Bharati, A., Sangal, R., and Reddy, T. P. (2002). *A Constraint Based Parser Using Integer Programming.* In Proceedings of ICON-2002.

Bharati, A., Husain, S., Vijay, M., Deepak, K., Sharma, D. M., and Sangal, R. (2009). *Constraint Based Hybrid Approach to Parsing Indian Languages.* In Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC 23), Hong Kong.

Bhat, R. A., and Sharma, D. M. (2011). *A Hybrid Approach to Kashmiri Shallow Parsing.* In LTC-2011: The 5th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC-2011).

Gadde, P., Jindal, K., Husain, S., Sharma, D.M., Sangal, R. (2010). *Improving Data Driven Dependency Parsing using Clausal Information.* Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 657–660, Los Angeles, California.

Husain, S., Gadde, P., Ambati, B. R., Sharma, D., Sangal, R. (2009). *A Modular Cascaded Approach to Complete Parsing.* IALP 2009, pages 141-146.

Nivre, J., Hall, J., and Nilsson J. (2006). *MaltParser: A Data-Driven Parser-Generator for Dependency Parsing.* In Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006), Genoa, Italy, pages 2216-2219.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov S., and Marsi, E. (2007). *MaltParser: A language-independent system for data-driven dependency parsing.* Natural Language Engineering, 13(2), pages 95-135.

Nivre, J. (2009). *Parsing Indian Languages with MaltParser.* In Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India.