Two stage constraint based hybrid approach to free word order language dependency parsing

Akshar Bharati, Samar Husain, Dipti Misra and Rajeev Sangal
Language Technologies Research Centre, IIIT-Hyderabad, India
{samar, dipti, sangal}@mail.iiit.ac.in

Abstract

The paper describes the overall design of a new two stage constraint based hybrid approach to dependency parsing. We define the two stages and show how different grammatical construct are parsed at appropriate stages. This division leads to selective identification and resolution of specific dependency relations at the two stages. Furthermore, we show how the use of hard constraints and soft constraints helps us build an efficient and robust hybrid parser. Finally, we evaluate the implemented parser on Hindi and compare the results with that of two data driven dependency parsers.

1 Introduction

Due to the availability of annotated corpora for various languages since the past decade, data driven parsing has proved to be immensely successful. Unlike English, however, most of the parsers for morphologically rich free word order (MoR-FWO) languages (such as Czech, Turkish, Hindi, etc.) have adopted the dependency grammatical framework. It is well known that for MoR-FWO languages, dependency framework provides ease of linguistic analysis and is much better suited to account for their various structures (Shieber, 1975; Mel'Cuk, 1988; Bharati et al., 1995). The state of the art parsing accuracy for many MoR-FWO languages is still low compared to that of English. Parsing experiments (Nivre et al., 2007; Hall et al., 2007) for these languages have pointed towards various reasons for this low performance. For Hindi[1], (a) *difficulty in extracting relevant linguistic cues*, (b) *non-projectivity*, (c) *lack of explicit cues*, (d) *long distance dependencies*, (e) *complex linguistic phenomena*, and (f) *less corpus size*, have been suggested (Bharati et al., 2008) for low performance. The approach proposed in this paper shows how one can minimize these adverse effects and argues that a hybrid approach can prove to be a better option to parsing such languages. There have been, in the past, many attempts to parsing using constraint based approaches. Some recent works include (Debusmann et al., 2004; Schröder, 2002; Bharati et al., 1993).

The paper describes the overall design of a new two stage constraint based hybrid approach to dependency parsing. We define the two stages and show how different grammatical construct are parsed at appropriate stages. This division leads to selective identification and resolution of specific dependency relations at two different stages. Furthermore, we show how the use of hard constraints (H-constraints) and soft constraints (S-constraints) helps us build an efficient and robust hybrid parser. Specifically, H-constraints incorporate the knowledge base of the language and S-constraints are weights corresponding to various constraints. These weights are automatically learnt from an annotated treebank. Finally, we evaluate the implemented parser on Hindi and compare the results with that of two data driven dependency parsers.

2 Two Stage Parsing

The parser tries to analyze the given input sentence, which has already been POS tagged and chunked[2], in 2 stages; it first tries to extract intra-clausal[3] dependency relations. These relations generally correspond to the argument structure of the verb, noun-noun genitive relation, infinitive-verb relation, infinitive-noun relation, adjective-noun, adverb-verb relations, etc. In the 2nd stage it then tries to handle more complex relations such as conjuncts, relative clause, etc. What this

[1] Hindi is a verb final language with free word order and a rich case marking system. It is one of the official languages of India, and is spoken by ~800 million people.

[2] A chunk is a set of adjacent words which are in dependency relation with each other, and are connected to the rest of the words by a single incoming arc. The parser marks relations between the head of the chunks (inter-chunk relations); this is done to avoid local details and can be thought as a device for modularity.

[3] A clause is a group of word such that the group contains a single finite verb chunk.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 77–80,
Paris, October 2009.

essentially means is a 2-stage resolution of dependencies, where the parser selectively resolves the dependencies of various lexical heads at their appropriate stage, for example verbs in the 1st stage and conjuncts and inter-verb relations in the 2nd stage. The key ideas of the proposed layered architecture are: (1) There are two layers stages, (2) the 1st stage handles intra-clausal relations, and the 2nd stage handles inter-clausal relations, (3) the output of each layer is a linguistically valid partial parse that becomes, if necessary, the input to the next layer, and (4) the output of the final layer is the desired full parse.

By following the above approach we are able to get 4-fold advantage, (1) Each layer in effect does linguistically valid partial parsing, (2) by dividing the labels into different functional sets (intra-clausal and inter-clausal) we localize the dependencies that need to be identified, hence the problem of long distance dependencies is minimizes, (3) by attacking the problem in a modular way, i.e. handling only individual clauses at 1st stage, we reduce non-projective structures significantly, and (4) the two stage constraint based approach can easily capture complex linguistic cues that are difficult to learn via the data-driven parsers. We'll revisit these points in Section 5. The 1st stage output for example 1 is shown in figure 1 (a).

Eg. 1: *mai ghar gayaa kyomki mai*
'I' 'home' 'went' 'because' 'I'
bimaar thaa
'sick' 'was'
'I went home because I was sick'

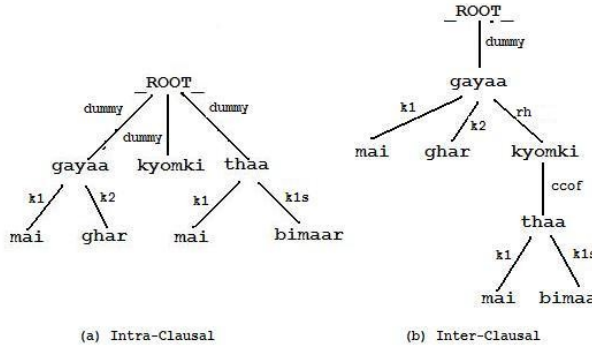

Figure 1. Eg 1 (a): 1st stage output, (b): 2nd stage final parse

In figure 1a, the parsed matrix clause subtree '*mai ghar gayaa*' and the subordinate clause are attached to _ROOT_. The subordinating conjunct '*kyomki*' is also seen attached to the _ROOT_. _ROOT_ ensures that the parse we get after each stage is connected and takes all the analyzed 1st stage sub-trees along with unprocessed nodes as its children. The dependency tree thus obtained in the 1st stage is partial, but linguistically sound. Later in the 2nd stage the relationship between various clauses are identified. The 2nd stage parse for the above sentences is also shown in figure 1b. Note that under normal conditions the 2nd stage does not modify the parse sub-trees obtained from the 1st stage, it only establishes the relations between the clauses.

3 Hard and Soft Constraints

Both 1st and 2nd stage described in the previous section use linguistically motivated constraints. These *hard* constraints (H-constraints) reflect that aspect of the grammar that in general cannot be broken. H-constraints comprise of lexical and structural knowledge of the language. The H-constraints are converted into integer programming problem and solved (Bharati et al., 1995). The solution(s) is/are valid parse(s). The *soft* constraints (S-constraints) on the other hand are learnt as weights from an annotated treebank. They reflect various preferences that a language has towards various linguistic phenomena. They are used to prioritize the parses and select the best parse. Both H & S constraints reflect the linguistic realities of the language and together can be thought as the grammar of a language. Figure 2 shows the overall design of the proposed parser schematically.

3.1 Hard Constraints

The core language knowledge being currently considered that cannot be broken without the sentence being called ungrammatical is named H-constraints. There can be multiple parses which can satisfy these H-constraints. This indicates the ambiguity in the sentence if only the limited knowledge base is considered. Stated another way, H-constraints are insufficient to restrict multiple analysis of a given sentence and that more knowledge (semantics, other preferences, etc.) is required to curtain the ambiguities. Moreover, we know that many sentences are syntactically ambiguous unless one uses some pragmatic knowledge, etc. For all such constructions there are multiple parses. As described earlier, H-constraints are used during intra-clausal (1st stage) and inter-clausal (2nd stage) analysis (cf. Figure 2). They are used to form a constraint graph which is converted into integer programming equalities (or inequalities). These are then solved to get the final solution graph(s). Some of the H-constraints are: (1) *Structural constraints* (ensuring the solution graph to be a tree,

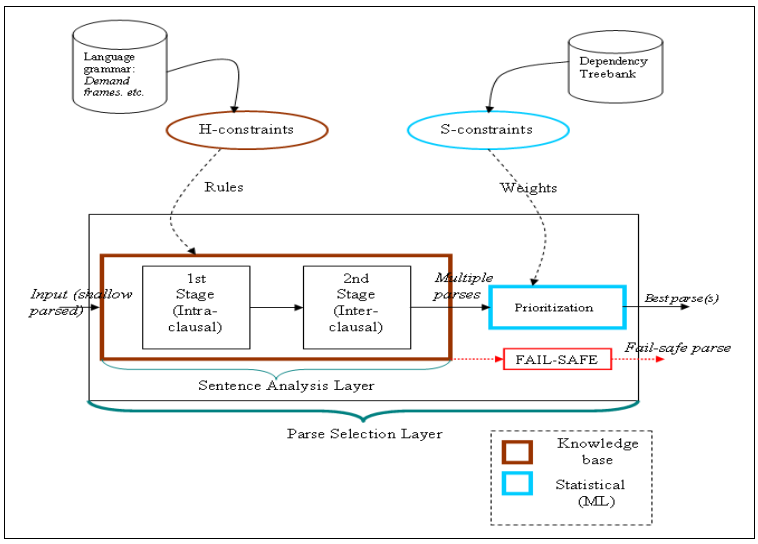

Figure 2. Overall parser design

removing implausible language specific ungrammatical structures, etc.), (2) *Lexicon* (linguistic demands of various heads), and (3) *Other lexical constraints* (some language specific characteristics), etc.

3.2 Soft Constraints

The S-constraints on the other hand are the constraints which can be broken, and are used in the language as preferences. These are used during the prioritization stage. Unlike the H-constraints that are derived from a knowledge base and are used to form a constraint graph, S-constraints have weights assigned to them. These weights are automatically learnt using a manually annotated dependency treebank. The tree with the maximum overall score is the best parse. Some such S-constraints are, (1) *Order of the arguments*, (2) *Relative position of arguments w.r.t. the verb*, (3) *Agreement principle*, (4) *Alignment of prominence scale*, and (5) *Structural preferences/General graph properties* (mild non-projectivity, valency, dominance, etc.), etc.

4 Evaluation

Malt Parser (version 0.4) (Nivre et al., 2007), and MST Parser (version 0.4b) (McDonald et al., 2005) have been tuned for Hindi by Bharati et al. (2008). Parsers were trained on a subset of a Hindi Treebank (Begum et al., 2008a). We use the same experimental setup (parameters, features, etc.) used by them and compare the results of the two data driven parsers with that of the proposed constraint based hybrid parser (CBP) on the same dataset[4] in terms of unlabeled attachments (UA), label (L) and labeled attachment (LA) accuracy. In Table 1, CBP' shows the performance of the system when a basic prioritizer is used, while CBP'' shows it for the best parse that is available in the first 25 parses. CBP gives the accuracy when the 1st parse is selected. We show CBP'' to show that a good parse is available in as few as the first 25 parses and that once the prioritizer is further improved the overall performance will easily cross CBP''.

	UA	LA	L
CBP	86.1	63	65
CBP'	87.69	69.67	72.39
CBP"	**90.1**	***75***	***76.9***
MST	*87.8*	70.4	72.3
Malt	86.6	68.0	70.6

Table 1. Parser Evaluation

5 Observations

The initial results show that the proposed parser performs better than the state-of-the-art data driven Hindi parsers. There are various reasons why we think that the proposed approach is better suited to parsing MoR-FWO. (1) Complex linguistic cues can easily be encoded as part of various constraints. For example, it has been shown by Bharati et al. (2008) that, for Hindi, complex agreement patterns, though present in the data, are not being learnt by data driven parsers. Such patterns along with other idiosyncratic language properties can be easily incorporated as constraints, (2) Making clauses as basic parsing unit drastically reduces non-projective

[4] For details on the corpus type, annotation scheme, tagset, etc. see Begum et al. (2008a).

sentences. Experiments in parsing MoR-FOW have shown that such non-projective sentences impede parser performances (Bharati et al., 2008; Hall et al., 2007). Note that there will still remain some intra-clausal non-projective structures in the 1st stage, but they will be short distance dependencies, (3) Use of H-constraints and S-constraints together reflect the grammar of a language. The rules in the form of H-constraints are complemented by the weights of S-constraints learnt from the annotated corpus, (4) 2 stage parsing lends itself seamlessly to parsing complex sentences by modularizing the task of overall parsing, (5) the problem of label bias (Bharati et al., 2008) faced by the data driven Hindi parsers for some cases does not arise here as contextually similar entities are disambiguated by tapping in hard to learn features, (6) Use of clauses as basic parsing units reduces the search space at both the stages, (7) Parsing closely related languages will become easy.

The performance of our parser is affected due to the following reasons, (a) *Small lexicon (linguistic demands of various heads)*: The total number of such demand frames which the parser currently uses is very low. There are a total of around 300 frames, which have been divided into 20 verb classes (Begum et al., 2008b). As the coverage of this lexicon increases, the efficiency will automatically increase. (b) *Unhandled constructions*: The parser still doesn't handle some constructions, such as the case when a conjunct takes another conjunct as its dependent, and (c) *Prioritization mistakes*: As stated earlier the prioritizer being used is basic and is still being improved. The overall performance will increase with the improvement of the prioritizer.

6 Conclusion

In this paper we proposed a new two stage constraint based hybrid approach to dependency parsing. We showed how by modularizing the task of overall parsing into 2 stages we can overcome many problems faced by data driven parsing. We showed how in the 1st stage only intra-clausal dependencies are handled and later in the 2nd stage the inter-clausal dependencies are identified. We also briefly described the use of H-constraints and S-constraints. We argued that such constraints complement each other in getting the best parse and that together they represent the grammar of the language. We evaluated our system for Hindi with two data driven parsers. Initial results show that the proposed parser performs better than those parsers. Finally, we argued why the proposed hybrid approach is better suited to handle the challenges posed by MoR-FWO and gave few pointers as how we can further improve our performance.

The proposed parser is still being improved at various fronts. To begin with a prioritization mechanism has to be improved. We need to enrich the verb frame lexicon along with handling some unhandled constructions. This will be taken up as immediate future work.

References

R. Begum, S. Husain, A. Dhwaj, D. Sharma, L. Bai, and R. Sangal. 2008a. Dependency annotation scheme for Indian languages. *Proc. of IJCNLP08*.

R. Begum, S. Husain, D. Sharma and L. Bai. 2008b. Developing Verb Frames in Hindi. *Proc. of LREC08*.

A. Bharati, S. Husain, B. Ambati, S. Jain, D. Sharma and R. Sangal. 2008. Two Semantic features make all the difference in Parsing accuracy. *Proc. of ICON-08*.

A. Bharati and R. Sangal. 1993. Parsing Free Word Order Languages in the Paninian Framework. *Proc. of ACL: 93*.

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

R. Debusmann, D. Duchier and G. Kruijff. 2004. Extensible dependency grammar: A new methodology. *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pp. 78–85.

J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi, M. Nilsson and M. Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. *Proc. of EMNLP-CoNLL shared task 2007*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. *Proc. of HLT/EMNLP*.

I. A. Mel'Cuk. 1988. Dependency *Syntax: Theory and Practice*, State University Press of New York.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *NLE*.

S. M. Shieber. 1985. Evidence against the context-freeness of natural language. In *Linguistics and Philosophy*, p. 8, 334–343.

I. Schröder. 2002. *Natural Language Parsing with Graded Constraints*. PhD thesis, Hamburg Univ.