Constraint Based Hybrid Approach to Parsing Indian Languages

Akshar Bharati, Samar Husain, Meher Vijay, Kalyan Deepak,
Dipti Misra Sharma and Rajeev Sangal

Language Technologies Research Centre
IIIT-Hyderabad, India.
samar@research.iiit.ac.in, {meher_vijay, deepak_s}@students.iiit.ac.in, {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.

Keywords: Dependency parsing, Hybrid approach, Modularity, Indian language parsing, Constraint based parsing.

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 of the constraint based parsers known in the literature are Karlsson et al. (1995), Maruyama (1990), Bharati et al. (1993, 2002), Tapanainen and Järvinen (1998), Schröder (2002), and more recently, Debusmann et al. (2004). Some attempts at parsing Hindi using data driven approach have been (Bharati et al., 2008b; Husain et al., 2009). Later in Section 4, we'll compare the results of data-driven Hindi parsing with that of our approach.

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

[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.

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 used as weights that 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.

The paper is arranged as follows: Section 2 describes in detail the proposed approach for parsing free word order languages. Section 3 discusses the types of constraints used. We evaluate the parser and compare the results with that of two data-driven parsers in Section 4. Section 5 gives certain observations on the approach. We conclude the paper in Section 7.

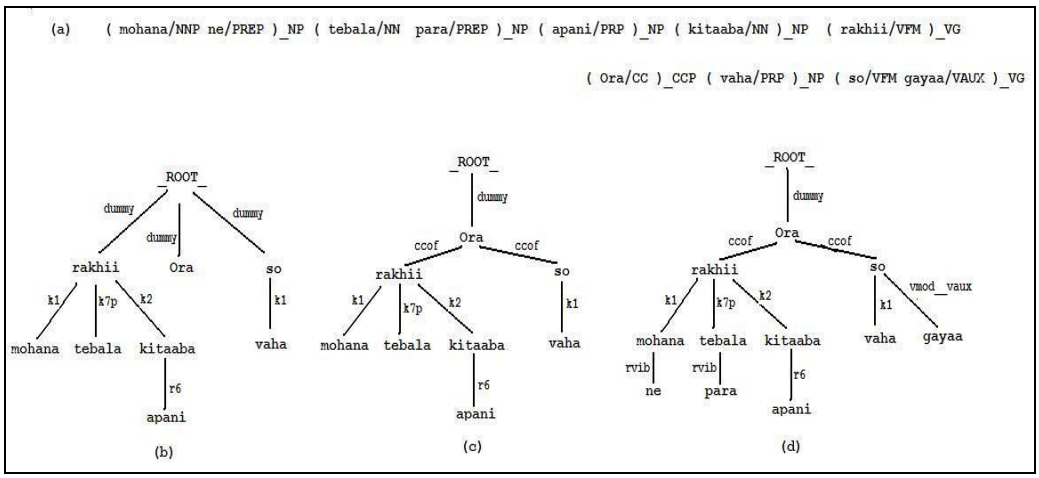

Figure 1: (a) POS tagged and chunked sentence, (b) and (c) are parse tree obtained using CBHP, (d) complete parse after IRCH

2 Approach

We try to solve the task of dependency parsing using a hybrid approach. A grammar driven approach is complemented by a controlled statistical strategy to achieve high performance and robustness. The overall task of dependency parsing is attacked using modularity, wherein specific tasks are broken down into smaller linguistically motivated sub-tasks. Figure 1 above shows the output of each of these sub-tasks.

2.1 Background

Data driven parsing is usually a single stage process wherein a sentence is parsed at one go. Many attempts have, however, tried to divide the overall task into sub-task. One trend has been to first identify dependencies and then add edge labels over them (McDonald et al., 2005, Chen et al., 2007). The other trend has been towards performing smaller linguistically relevant tasks as a precursor to complete parsing (Abney, 1997; Bharati et al., 1995; Attardi and Dell'Orletta, 2008; Peh and Ting, 1996).

In our approach we divide the task of parsing into the following sub-tasks (layers):

1. POS tagging, chunking (POSCH),
2. Constraint based hybrid parsing (CBHP),
3. Intra-chunk dependencies (IRCH) identification.

(a) POSCH is treated as pre-processing to the task of parsing. A bag represents a set of adjacent words which are in dependency relations with each other, and are connected to the rest of the words by a single incoming dependency arc. Thus a bag is an unexpanded dependency tree connected to the rest only by means of its root. A noun phrase or a noun group chunk is a bag in which there are no verbs, and vice versa for verb chunks. The relations among the words in a

chunk are not marked and hence allow us to ignore local details while building the sentence level dependency tree. In general, all the nominal inflections, nominal modifications (adjective modifying a noun, etc.) are treated as part of a noun chunk, similarly, verbal inflections, auxiliaries are treated as part of the verb chunk (Bharati et al., 2006).
(b) CBHP takes the POS tagged and chunked sentence as input and parses it in two stages. The parser makes use of knowledge base of the language along with syntactico-semantic preferences to arrive at the final parse. Broadly, modularity in CBHP works at two layers (cf. Figure 3): (1) The sentence analysis layer, and (2) The parse selection layer. We discuss this approach to parsing in the following sections.
(c) IRCH dependencies are finally identified as a post-processing step to (b) and (c). Once this is done, the chunks can be removed and we can get the complete dependency tree. We will not discuss IRCH in this paper.

In the dependency trees (b) and (c) shown in Figure 1, each node is a chunk and the edge represents the relations between the connected nodes labeled with suitable relations[2]. After removing the chunks in (d) each node is a lexical item of the sentence.

Eg. 1: mohana ne tebala para apani kitaaba
'Mohan' 'ERG' 'table' 'on' 'his' 'book'
rakhii Ora vaha so gayaa
'kept' 'and' 'he' 'sleep' 'PRFT'
'Mohan placed his book on the table and slept'

From (a) to (d) in Figure 1, outputs of each of the previously discussed layers have been shown. Note that one can use any of these outputs independently. More importantly, (b) is a partial parse obtained after the 1st stage of CBHP, and (c) is the output after the 2nd stage of CBHP. We'll elaborate on this in the following sections. To test the performance of the proposed parser we use gold POS tagged and chunked data, instead of using the outputs of POS tagger and chunker.

2.2 Constraint Parsing

Constraint based parsing using integer programming has been successfully tried for Indian languages (Bharati et al., 1993; 2002). Under this scheme the parser exploits the syntactic cues present in a sentence and forms constraint graphs (CG) based on the generalizations present. It uses such notions as basic demand frames and transformation frames (Bharati et al., 1995) to construct the CG. It then translates the CG into an integer programming (IP) problem. The solutions to the problem provide the possible parses for the sentence. We follow the approach used by Bharati et al. (1995, 2008a) for formulating the constraints as IP problem and solving them to get the parses.

2.3 Two Stage Parsing

The proposed parser tries to analyze the given input sentence, which has already been POS tagged and chunked, in 2 stages; it first tries to extract intra-clausal[3] dependency relations. These 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 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

[2] All the relations marked by the parser are syntactico-semantic labels. For a detailed analysis see Bharati et al. (1995). Many relations shown in the diagrams of this paper are described in Begum et al. (2008a). For the complete tagset description, see http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf
[3] A clause is a group of word such that the group contains a single finite verb chunk.

example verbs in the 1st stage and conjuncts and inter-verb relations in the 2nd stage. The key ideas 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/are the desired full parse(s). These form the *sentence analysis* layer in the overall design. Figure 3 shows this clearly.

The 1st stage output for example 2 is shown in Figure 2(a).

Eg. 2: mai ghar gayaa kyomki mai
'I' 'home' 'went' 'because' 'I'
bimaar thaa
'sick' 'was'
'I went home because I was sick'

In Figure 2(a), the parsed matrix clause subtree '*mai ghar gayaa*' and the subordinate clause are attached to _ROOT_. The subordinating conjunct *'kyomki'* (because) 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 2(b). At the end of 2nd stage, the subordinate conjunct *kyomki* gets attached to the matrix clause and takes the root of the subordinate clause as its child. Similar to example 2, the analysis of example 1 is shown in Figure 1. Note that under normal conditions the 2nd stage does not modify the parses obtained from the 1st stage, it only establishes the relations between the clauses. However, sometimes under very strict conditions, *repair* is possible (Bharati et al., 2008a).

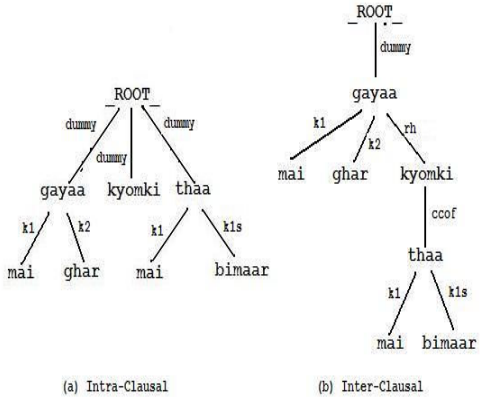

Figure 2: (a): 1st stage output for Eg. 2, (b): 2nd stage final parse for Eg. 2

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., 2002, 2008a). 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[4]. They reflect various preferences that a language has towards various linguistic phenomena. They are used to prioritize the parses and

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

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 3 schematically shows the overall design of the proposed parser and places these constraints in that context.

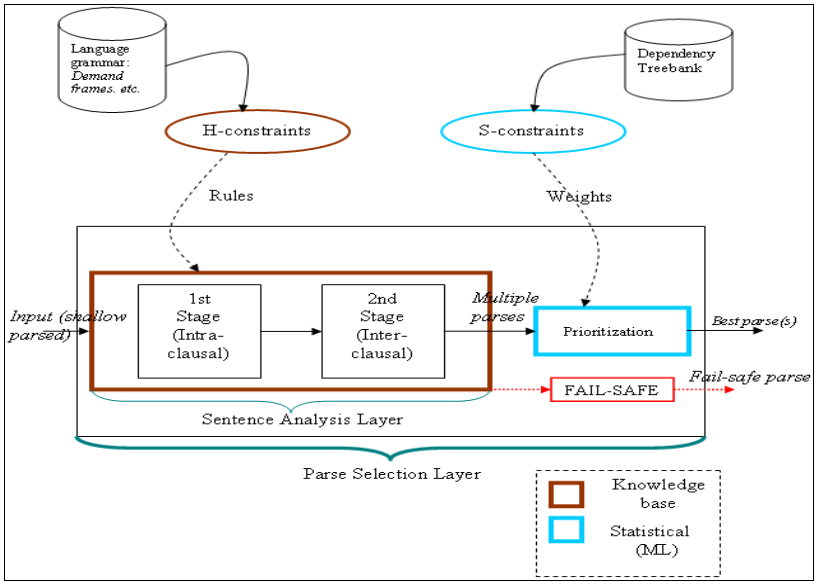

Figure 3: Schematic representation of CBHP

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 curtail 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 3). 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) (Bharati et al., 2008a). Some of the H-constraints are: (1) *Structural constraints* (ensuring the solution graph to be a tree, 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 that 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 weights are used to score the parse trees. 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*, (4) *Structural preferences/General graph properties* (mild non-projectivity, valency, dominance, etc.), etc.

Parses obtained after the 2nd stage, satisfies all the relevant H-constraints. We score these parses based on the S-constraints and the parse with the max score is selected. The scoring of a parse *p* is done as follows:

$$\zeta(p) = \zeta(R_p) \quad (1)$$

where, ζ is a recursive scoring function, R_p is the root of the parse p

$$\zeta(n) = \sum_e [\zeta(e) + k * \zeta(C_{ne})] \quad (2)$$

where, C_{ne} is the child of node n along edge e and k is a parameter

$$\zeta(e) = \sum_i [k_i^1 * \{P(r/\gamma_i) + P(\gamma_i)\}] \quad (3)$$

where, $P(r/\gamma_i)$ is the probability of the relation r given that γ_i is the i^{th} S-constraint and $P(\gamma_i)$ is the probability of occurrence of γ_i and k_i^1 is a weight associated with γ_i .

The ranking function tries to select a parse p for a sentence such that the overall accuracy of the parser is maximized. The parameters k and k_i^1 in (2) and (3) above are set using maximum likelihood estimation. Note that the scoring function considers structure of the parse along with the linguistic constraints under which this structure can occur.

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. (2008b). Parsers were trained on a subset of a Hindi Treebank (Begum et al., 2008a). The training set had 1185 sentences, development and test set had 268 and 220 sentences respectively. 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 (CBHP) on the same dataset[5]. Table 1 shows the performance in terms of unlabeled attachments (UA), labeled (L) and labeled attachment (LA) accuracy. In Table 1, CBHP shows the performance of the system when a prioritizer (cf. Section 3.2) is used, while CBHP'' shows it for the best parse that is available in the first 25 parses. We show CBHP'' to show that a good parse is available in as few as the first 25 parses. Once the prioritizer is further improved the overall performance (of CBHP) will potentially cross CBHP''.

Table 1: Parser Evaluation

	UA	LA	L
CBHP	88.24	71.13	74.12
CBHP"	91.24	79.07	80.82
MST	87.62	78.35	81.85
Malt[6]	88.04	79.86	82.68

5 Observations

The initial results show that the proposed parser gives comparable results with the state-of-the-art data driven Hindi parsers.

The performance of our parser is affected due to the following reasons:
(1) *Small lexicon (linguistic demands of various heads)*: One of the main reasons for the low LA is that the total number of linguistic demand frames that the parser currently uses is very low. Unlike the data driven parsers CBHP doesn't employ any ML techniques to induce this information. For Hindi, 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

[5] For details on the corpus type, annotation scheme, tagset, etc. see Begum et al. (2008a).
[6] Results obtained for Malt using version 1.2

automatically increase. Note that in spite of this major drawback, CBHP still manages to achieve considerable high accuracy.
(2) *Unhandled constructions*: The parser still doesn't handle some constructions, and
(3) *Prioritization mistakes*: As stated earlier the prioritizer is still being improved. Various scoring and ranking functions are being tried out. The overall performance will inevitably increase with the improvement of the prioritizer.

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. (2008b) 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) IP formulation allows for handling non-projective parsing (Riedel and Clarke, 2006),
(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) Two-stage parsing lends itself seamlessly to parsing complex sentences by modularizing the task of overall parsing,
(5) The problem of label bias faced by the data driven Hindi parsers (Bharati et al., 2008b) 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 proposed approach although tested for Hindi will work equally well for other closely related Indian languages such as Urdu, Punjabi, etc. It is important to note here that the approach is language independent, and that the machinery and the design are not language specific. It is easy to see that given a treebank for a language, weights for various S-constraints in that language can automatically be learnt. A set of S-constraints can be common for various Indian languages but can vary in their weights. In fact, one can potentially induce the H-constraints (in the form of linguistic demands for various heads, etc.) from a treebank as well. The H-constraints and S-constraints for closely related languages will tend to be very similar.

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. Our 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 is being 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

Abney, S. 1997. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344.

Attardi, G. and F. Dell'Orletta. 2008. Chunking and Dependency Parsing. *LREC Workshop on Partial Parsing: Between Chunking and Deep Parsing*. Marrakech, Morocco.

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

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

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

Bharati, A., D. M. Sharma, L. Bai and R. Sangal. 2006. AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *LTRC-TR31*.

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

Bharati, A., S. Husain, D. M. Sharma, and R. Sangal. 2008a. A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages. *In Proceedings of the COLIPS IALP*.

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

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

Chen, W., Y. Zhang and H. Isahara. 2007. A Two-Stage Parser for Multilingual Dependency Parsing. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL*.

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

Hall, J., 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*.

Husain, S., P. Gadde, B. Ambati, D. Sharma and R. Sangal. 2009. A modular cascaded approach to complete parsing. *In Proc. of COLIPS IALP-09. Singapore*.

Karlsson, F., A. Voutilainen, J. Heikkilä and A. Anttila, (eds). 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Mouton de Gruyter.

Maruyama, H. 1990. Structural disambiguation with constraint propagation. In *Proc. of ACL:90*.

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

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

Nivre, J., 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*.

Peh, L-S. and C.H.A. Ting. 1996. A Divide-and-Conquer Strategy for Parsing. *In Proc. of IWPT*.

Riedel, S. and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. EMNLP*.

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

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

Tapanainen, P. and T. Järvinen. 1997. A non-projective dependency parser. *Proc. of the 5th Conference on Applied Natural Language Processing*, pp. 64–71.