Constraint based Hindi dependency parsing

by

Meher Vijay Yeleti, Kalyan Deepak

in

ICON-2009: International Conference on Natural Language Processing, NLP Tools Contest
(INCON-2009)


Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2009
Constraint based Hindi dependency parsing

Meher Vijay Yeleti, Kalyan Deepak
Language Technologies Research Centre
IIIT-Hyderabad, India.
meher.vijay@students.iiit.ac.in, deepak_s@students.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. The experiments tried out for soft constraints are elucidated in detail. Finally, we evaluate the implemented parser on ICON tools contest Hindi data. Best Labeled and unlabeled attachment accuracies for Hindi are 62.20% and 85.55% respectively.

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 Hindi1, (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.

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 describe the experiments performed and report the results in Section 4.

---

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 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-tasks. 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; Shiuan and Ann, 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.

---

\(^2\) 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/ls/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf
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 inde-
pendently. 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 program-
ning has been successfully tried for Indian lan-
guages (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 trans-
formation frames (Bharati et al., 1995) to con-
struct 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 Bha-
ratii et al. (1995, 2008a) for formulating the con-
straints 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-clausal3 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 de-
pendencies, 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 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
parses(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 2a, 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 unpro-
cessed nodes as its children. The dependency tree
thus obtained in the 1st stage is partial, but lin-
guistically sound. Later in the 2nd stage the rela-
tionship between various clauses are identified.
The 2nd stage parse for the above sentences is
also shown in figure 2b. At the end of 2nd stage,
the subordinate conjunct kyomki gets attached to
the matrix clause and takes the root of the subor-
dinate 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).

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

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

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.

Some of the graphical S-constraints that we have used are, 1) \{child POS, parent POS\}, 2) \{child POS, GGP POS\}, 3) \{child POS, child Width\}, 4) \{child POS, parent Depth\}, 5) \{parent POS, GP POS\}, where POS is the Part-of-Speech tag and GP is the grandparent and GGP is the great grandparent of the child and parent Depth and child Width are the depths and widths of the parent and child sub-tree respectively.

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 score \( \zeta(p) \) of a parse \( p \) is calculated as follows:

\[
\zeta(p) = \zeta(R_p) \tag{1}
\]

where, \( \zeta \) is a recursive scoring function, \( R_p \) is the root node of the parse \( p \)

\[
\zeta(n) = \sum_e [\zeta(e) + k \cdot \zeta(C_e)] \quad \tag{2}
\]

\[4\] For details on the corpus type, annotation scheme, tagset, etc. see Begum et al. (2008a).
where, $C_{ne}$ is the child of node $n$ along edge $e$ and $k$ is a parameter

$$
\zeta(e) = \sum [k_i \cdot (P(r / \gamma_i) + P(\gamma_i))] \quad (3)
$$

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

$$
P(r / \gamma_i) = C(\gamma_i, r) / C(r) \quad (4)
$$

where, $C(\gamma_i, r)$ is the count of occurrence of relation $r$ and $\gamma_i$ together and $C(r)$ is the count of occurrence of the relation $r$ in training data. These counts will be calculated from the training data for each of the S-constraint $\gamma_i$ and stored.

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 $k_i$ 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 Experiments and Results

Initially the two stage constraint based parser is used to parse the sentences. It outputs multiple parses for each of the sentence. These parses are ranked using the scoring technique discussed in section 3.2. Three variations of the scoring technique are explained below. The probabilities required are calculated from the training data.

4.1 Method1

In this method the parses are scored using a single S-constraint using the scoring function discussed in section 3.2 with $i=1$. The parse with the best score is the required output. In this case there may be multiple parses with the same highest score using a single S-constraint. Output is the $1^{st}$ parse out of all parses having the highest score. To solve the problem of multiple parses with the highest score, second method is used. For this method we tried with different values of $k$ in equation 2 like 0.1, 0.5, 1, 2, 5 etc. The best results are for $k=2$. So for all the other methods experiments are done by fixing the $k$ value as 2.

4.2 Method2

In the second method, initially only one S-constraint is used to score the parses using the scoring function similar to the method1. If there are multiple parses with highest score then second S-constraint is used to resolve them and so on till there is unique parse or till all the S-constraints are used. If there are still multiple parses the first-one is the output. In this method, the order in which we use the S-constraints is important. This affects the accuracy of the parser. The order which we have used is the descending order of the best accurate (found using the hindi development data with method1) S-constraint.

4.3 Method3

In the third method all the soft-constraints are used in parallel with weights $k_i$ associated with each of them. Then the parses are scored using the scoring technique discussed in section 3.2. The algorithm for boosting loss function as discussed in Collins (2000) is used to learn the weights for each of the S-constraints.

This algorithm runs on the training data and learns the weights for each S-constraint. This has several parameters: 1) the margins for each parse of each sentence, 2) Number of iterations ($N$) to find the best possible weights.

Margins are initialized with the best single S-constraint score obtained from the method1. The algorithm is run with different values of $N$. After $N=2000$ weights of the features did not change considerably. So the value of $N$ is fixed as 2000.

4.4 Parameters

The parameters that are finalized after experimenting with all the possible values and used in all of the above methods are shown below in table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $k$</td>
<td>2</td>
</tr>
<tr>
<td>Best S-constraint (Method1)</td>
<td>[C-POS,P-POS]</td>
</tr>
<tr>
<td>Value of $N$</td>
<td>2000</td>
</tr>
<tr>
<td>Margins initialization</td>
<td>[C-POS,P-POS]</td>
</tr>
</tbody>
</table>

Table 1. Parameters, where C-POS and P-POS are the child and parent POS respectively

4.5 Results

We have tried all the three methods on the development data and method 2 gave the best results on the development data. So we gave the output of method 2 on the testing data for the
evaluation for the tools contest. The second method is used starting with the best S-constraint \{C-POS, P-POS\}. Our results on the Hindi test data are shown below in Table 2.

<table>
<thead>
<tr>
<th>Method2</th>
<th>UA</th>
<th>LA</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>85.55</td>
<td>62.20</td>
<td>65.88</td>
</tr>
</tbody>
</table>

Table 2 Results on Hindi Test data

Acknowledgements

We would like to thank Mr. Samar Husain for his valuable guidance and suggestions. We would also like to thank Dr. Dipti Misra Sharma and Dr. Rajeev Sangal for their valuable suggestions and support.

References


