Intra-Chunk Dependency Annotation: Expanding Hindi Inter-Chunk Annotated Treebank

Prudhvi Kosaraju, Bharat Ram Ambati, Samar Husain Dipti Misra Sharma, Rajeev Sangal
Language Technologies Research Centre
IIIT Hyderabad
Treebank

• Linguistic resources in which each sentence has
  – Parse tree
  – morphological, syntactic and lexical information marked explicitly

• Some treebanks
  – Penn Treebank (Marcus et al., 1993) for English
  – Prague Dependency Treebank (Hajicova, 1998) for Czech.

• For Indian Languages
  – Lack of such treebank been a major bottleneck for advance research and development of NLP tools and applications
Treebank creation

• Annotated manually or semi-automatically
• Manual creation
  – Annotators has to follow prescribed guidelines
  – Costly process in terms of both money & time
• Semi-automatic creation
  – Running of tools or parsers
  – Manual correction of Errors

Note: An accurate annotating parser/tool saves cost and time for both the annotation as well as the validation task
Hindi Treebank

• Multi-layered and multi-representational treebank having
  – Dependency relations
  – Verb arguments (PropBank, Palmer et al., 2005)
  – Phase structure

• Dependency treebank has information at
  – morpho-syntactic (morphological, part-of-speech (POS) and chunk) level
  – syntactico-semnatic (dependency) level
Hindi Dependency Treebank

• Manual annotation has been done at
  – Part_of_speech level
  – Chunk level
  – Morph level
  – Inter-chunk dependency level
Inter-chunk annotated sentence

Sentence 1: नीली किताब गिर गई
niilii kitaab gir gaii
‘blue’ ‘book’ ‘fall’ ‘go-perf’
The blue book fell down

Figure 1: SSF Representation

Figure 2: Inter-chunk dependency tree of sentence 1
Intra-chunk dependencies

• Intra-chunk dependencies left unannotated since
  – Identification of intra-chunk dependencies are quite deterministic
  – Can be automatically annotated with high degree of accuracy

• Marking intra-chunk dependencies on inter-chunk dependency annotated trees results expansion of the later

• Automatic conversion to phase structure depends upon the expanded version of the treebank

• Hence, a High quality intra-chunk dependency annotator/parser is required
Fig 3: SSF representation of complete dependency tree

Fig 4: complete dependency tree of Sentence 1
Intra-chunk dependency annotation Guidelines

• Tags can be classified into
  – Normal dependencies
    • nmod__adj, jjmod__intf etc
  – Local word group dependencies (lwg)
    • lwg__psp, lwg__vaux, lwg__neg etc
  – Linking local word group dependencies
    • lwg__cont etc

• Total of 12 tags were used for experiments
Various types of adjectival modifications are shown using this label.

An adjective modifying a head noun is one such instance.

The label also incorporates various other modifications such as a demonstrative or a quantifier modifying a noun.

**Chunk:** नीली किताब

NP ((niilii_JJ kitaab_NN))

‘blue‘ ‘book’

\[
\text{niilii} \downarrow \text{nmod__adj} \downarrow \text{kitaab}
\]
lwg__psp

- Used to attach post-positions/ auxiliaries associated with the noun or a verb.
- ‘lwg’ in the label name stands for local word grouping and associates all the postpositions with the head noun

Chunk: अभिषेक ने
NP((abhishek_NNP ne_PSP))
‘abhishek’ ‘ERG’

abhishek
\[ lwg__psp \]
ne
To show that a group of lexical items inside a chunk together perform certain function.

In such cases, we do not commit on the dependencies between these elements.

We see this with complex post-positions associated with a noun/verb or with the auxiliaries of a verb.

‘cont’ stands for continue.

Chunk: जा सकता है

VGF((jaa_VM sakataa_VAUX hai_VAUX))

‘go’               ‘can’               ‘be-pres’

jaa

lwg__vaux

sakataa

lwg__cont

hai
Intra-chunk dependency annotator/parser

• Built a robust intra-chunk dependency parser for Hindi
  – Rule based Approach
  – Statistical Approach
  – Hybrid Approach (using heuristic based post-processing component on top of statistical approach)

• The rule based tool can easily adaptable to other languages as well
Rule based intra-chunk dependency annotator

- Identifies modifier-modified (parent-child) relationship inside a chunk
- Rules provided in a fixed rule template
- Heads in each chunk determined by head computation module
- All information present in the SSF can be captured through the rule template
Rule template

- We capture the rules in form of constraints applicable at
  - Chunk Label
  - Parent Constraints
  - Child Constraints
  - Contextual Constraints

<table>
<thead>
<tr>
<th>Chunk Name</th>
<th>Parent Constraints</th>
<th>Child Constraints</th>
<th>Contextual Constraints</th>
<th>Dep. Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>POS == NN</td>
<td>POS == JJ</td>
<td>posn(parent) &gt; posn(child);</td>
<td>nmod__adj</td>
</tr>
</tbody>
</table>

Table 1 : Rule template
Statistical approach : Sub-tree parsing using Malt parser

• Malt parser (*Nivre et al., 2007*) , transition based dependency parser is best suited for identifying short range dependencies (*Nivre, 2003*)
• Each chunk is separated and called sub-tree
• Data is divided into training (192 sentences), development(64) and testing(64)
• We followed the strategies used in kosaraju et.al,2010
  – Feature pool
  – Pruning features using forward selector
## Results (on gold data)

### Table 2: Data Statistics

<table>
<thead>
<tr>
<th></th>
<th>No: of Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>192</td>
</tr>
<tr>
<td>Development</td>
<td>64</td>
</tr>
<tr>
<td>Testing</td>
<td>64</td>
</tr>
</tbody>
</table>

### Table 3: Rule based accuracies

<table>
<thead>
<tr>
<th></th>
<th>LAS</th>
<th>UAS</th>
<th>LS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>97.89</td>
<td>98.50</td>
<td>98.38</td>
</tr>
</tbody>
</table>

### Table 4: Statistical approach showing baseline, POS and best templates

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>POS -template</th>
<th>Best template</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>95.70</td>
<td>96.80</td>
<td>97.35</td>
</tr>
<tr>
<td>UAS</td>
<td>97.07</td>
<td>97.62</td>
<td>98.26</td>
</tr>
<tr>
<td>LS</td>
<td>96.80</td>
<td>97.80</td>
<td>97.90</td>
</tr>
</tbody>
</table>
Data Statistics

<table>
<thead>
<tr>
<th></th>
<th>No:of Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>192</td>
</tr>
<tr>
<td>Development</td>
<td>64</td>
</tr>
<tr>
<td>Testing</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 2 : Data Statistics
Results (on gold data)

Table 3: Rule based accuracies

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>97.89</td>
<td></td>
</tr>
<tr>
<td>UAS</td>
<td>98.50</td>
<td></td>
</tr>
<tr>
<td>LS</td>
<td>98.38</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Statistical approach showing baseline, POS and best templates

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>POS -template</th>
<th>Best template</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>95.70</td>
<td>96.80</td>
<td>97.35</td>
</tr>
<tr>
<td>UAS</td>
<td>97.07</td>
<td>97.62</td>
<td>98.26</td>
</tr>
<tr>
<td>LS</td>
<td>96.80</td>
<td>97.80</td>
<td>97.90</td>
</tr>
</tbody>
</table>
Hybrid approach

- Post processed the statistical approach output using the rules as heuristics
- Only those tag associated rules are considered for which recall in rule-based is greater than statistical approach
  - Pof__cn, nmod__adj,rsym

<table>
<thead>
<tr>
<th>Approach</th>
<th>LAS</th>
<th>UAS</th>
<th>LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule-based</td>
<td>97.89</td>
<td>98.50</td>
<td>98.38</td>
</tr>
<tr>
<td>Statistical</td>
<td>97.35</td>
<td>98.26</td>
<td>97.90</td>
</tr>
<tr>
<td>Hybrid</td>
<td>98.17</td>
<td>98.81</td>
<td>98.63</td>
</tr>
</tbody>
</table>
Special Cases

‘Chunks are self contained units. Intra-chunk dependencies are chunk internal and do not span outside a chunk.’

• The above is the basis for neat division of inter-chunk and intra-chunk parsing

• However, there are two cases this constraint does not hold.
  – In these two cases a chunk internal element that is not the head of the chunk has a relation with a lexical item outside its chunk

• Hence, these relations are to be handled separately
Special cases

• rsym__EOS (End of Sentence marker):
  – Occurs in the last chunk, Attached to head of the sentence

• lwg__psp :
  – According to guidelines, psp attaches to head of the chunk with lwg__psp
  – However, if the right most child of a CCP (conjunction chunk) is a nominal (NP or VGNN), one needs to attach the PSP of this nominal child to the head of the CCP during expansion
  – If there are multiple PSP, then first PSP gets a lwg__psp and second lwg__cont
Special case (lwg___psp)

NP(raama_NNP) CCP(aur_CC) NP(siitaa_NNP ne_PSP)
‘ram’      ‘and’      ‘sita’      ‘ERG’

Fig 5: Expanded sub-tree with PSP connected with CC
Conclusion

• Described annotation guidelines for marking intra-chunk dependency relations

• Approaches:
  1. Rule based
  2. Statistical
  3. Hybrid (using 1&2)

• By error analysis the outputs, only certain tags are not being marked correctly.

• This is good news because then one can make very targeted manual corrections after the automatic tool is run
THANK YOU