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

Sentence1: नीली किताब गिर गई

niilii kitaab gir gaii

‘blue’ ‘book’ ‘fall’ ’go-perf’

The blue book fell down

```
1       ((      NP      <name=’NP’ drel=’k1:VGF’>
1.1     niilii  JJ      <name='niilii'>
1.2     kitaab  NN      <name='kitaab'>
        ))
2       ((      VGF     <name=’VGF’>
2.1     gir     VM      <name='gir'>
2.2     gaii    VAUX    <name='gaii'>
        ))
```

Figure 1: SSF Representation

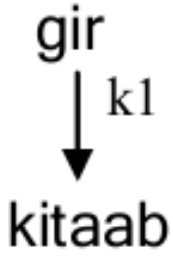


Figure 2: Inter-chunk dependency tree of sentence 1

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

```
1    niilii   JJ    <fs drel='nmod__adj:kitaab' chunkType='child:NP' name='niilii '>
2    kitaab   NN    <fs drel='k1:gir' name='kitaab' chunkId='NP' chunkType='head:NP'>
3    gir      VM    <fs name='gir' chunkId='VGF' chunkType='head:VGF'>
4    gaii     VAUX  <fs drel='lwg__aux:gir' name='gaii' chunkType='child:VGF'>
```

Fig 3: SSF representation of complete dependency tree

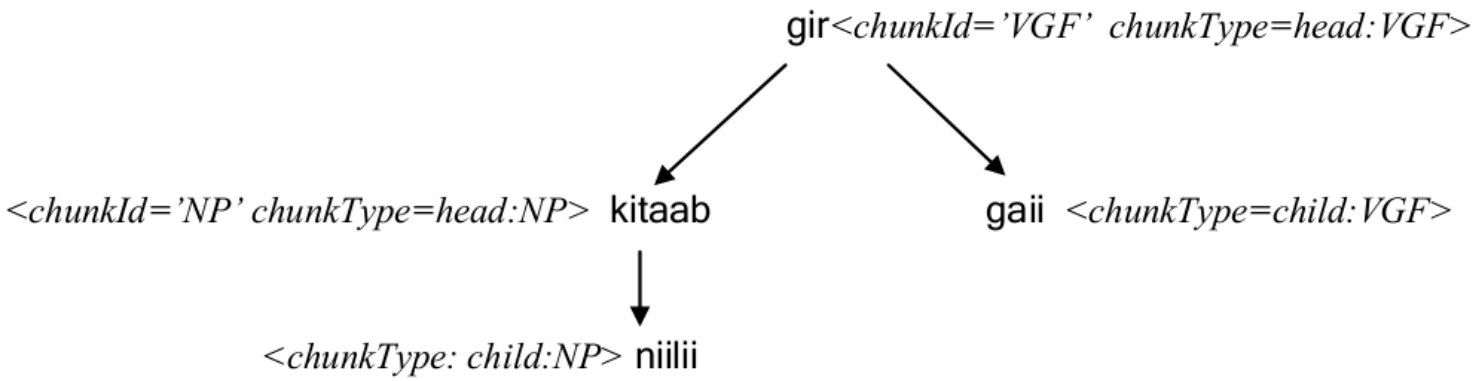


Fig 4: complete dependency tree of Sentence 1

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

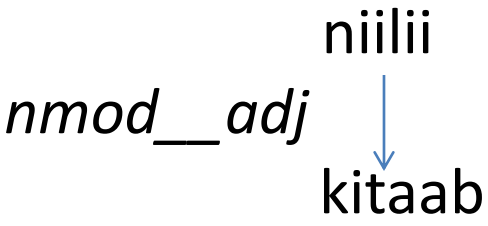
nmod__adj

- 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'



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

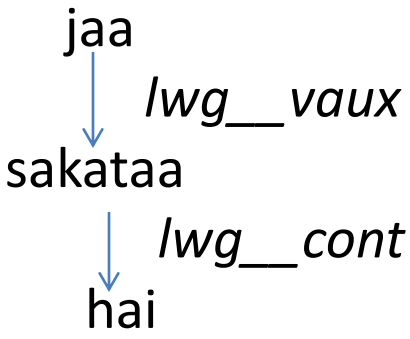
lwg__cont

- 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’



Intra-chunk dependnecy 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

Chunk Name	Parent Constraints	Child Constraints	Contextual Constraints	Dep. Relation
NP	POS == NN	POS == JJ	posn(parent) > posn(child);	nmod__adj

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

	No:of Sentences
Training	192
Development	64
Testing	64

Table 3: Rule based accuracies

LAS	**97.89**
UAS	98.50
LS	98.38

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

	Baseline	**POS -template**	**Best template**
LAS	95.70	96.80	97.35
UAS	97.07	97.62	98.26
LS	96.80	97.80	97.90

Data Statistics

	No:of Sentences
Training	192
Development	64
Testing	64

Table 2 : Data Statistics

Results (on gold data)

Table 3: Rule based accuracies

LAS	**97.89**
UAS	98.50
LS	98.38

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

	Baseline	**POS -template**	**Best template**
LAS	95.70	96.80	97.35
UAS	97.07	97.62	98.26
LS	96.80	97.80	97.90

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 5: All methods parsing accuracies

Approach	LAS	UAS	LS
Rule-based	97.89	98.50	98.38
Statistical	97.35	98.26	97.90
Hybrid	98.17	98.81	98.63

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 seperately

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’

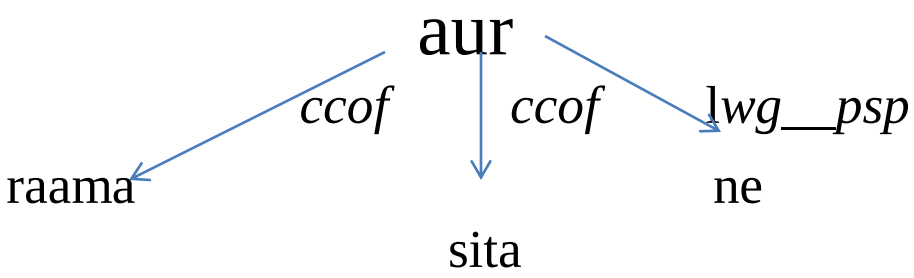


Fig 5: Expanded sub-tree with PSP connected with CC

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