LTAG-spinal treebank and parser for Hindi

Prashanth Mannem, Aswarth Abhilash and Akshar Bharati

LTRC, IIIT
Hyderabad, India 500032
prashanth@research.iiit.ac.in, abhilash.d@students.iiit.ac.in

Dec 16, 2009
Outline

1. Introduction
   - Lexicalized Tree Adjoining Grammar
   - Hyderabad Dependency Treebank

2. LTAG-spinal formalism

3. Hindi LTAG-spinal treebank
   - Extraction from HyDT
   - Assigning spines to words

4. LTAG-spinal Parser

5. Results

6. Summary
Introduction

- Build an LTAG treebank for Hindi
- To annotate one is an expensive process
- Treebanks can be converted from one formalism to another with either no or minimal manual annotation
- Dependency treebank available for Hindi (1800 sentences)
  - Hyderabad Dependency Treebank (HyDT)
- Extract an LTAG-treebank from HyDT
Lexicalized Tree Adjoining Grammar

- Each word is associated with **elementary trees**
  - Elementary trees represent extended projections of lexical items
  - They encapsulate all syntactic/semantic arguments of the lexical anchor
- Larger trees are derived by **substitution** and **adjunction**
  - *Substitution*: replacing a leaf with a new tree, a so-called initial tree
  - *Adjunction*: replacing an internal node with a new tree, a so-called auxiliary tree
- Tree resulting from combination of e-trees is a **derived tree**
- **Derivation tree** records the history of how a derived tree is built
From Derived tree to Derivation tree
**HyDT - Hyderabad Dependency Treebank**

- Sentences annotated with
  - POS tags
  - Minimal constituents (chunks) and their heads
  - Relations between chunks (inter-chunk)
  - Intra-chunk dependencies left unspecified
  - *Trees can be expanded if needed*

- Annotation scheme based on Paninian framework
  - [Begum et al. 2008](#)
  - Syntactic cues help in determining the type of relation

```
(( meraa_PRP baDzaa_JJ bhaaii_NN ))_NP  (( bahuta_QF phala_NN ))_NP  (( khaataa_VM hai_VAUX ))_VG
my big brother lots−of fruits eat PRES.
```
LTAG treebank from HyDT!

- HyDT doesn’t have subcategorization information
- Argument/Adjunct distinction hard to make
- No phrase structure trees to get the elementary trees
- How about a framework where the representations for arguments and adjuncts are similar?
  - Leave the disambiguation for post-processing
  - Elementary trees are in spinal form and don’t have the subcategorization information
Introduction Hyderabad Dependency Treebank

**LTAG treebank from HyDT**

- HyDT doesn’t have subcategorization information
- Argument/Adjunct distinction hard to make
- No phrase structure trees to get the elementary trees
- How about a framework where the representations for arguments and adjuncts are similar?
  - Leave the disambiguation for post-processing
  - Elementary trees are in spinal form and don’t have the subcategorization information
Outline

1 Introduction
   Lexicalized Tree Adjoining Grammar
   Hyderabad Dependency Treebank

2 LTAG-spinal formalism

3 Hindi LTAG-spinal treebank
   Extraction from HyDT
   Assigning spines to words

4 LTAG-spinal Parser

5 Results

6 Summary
**LTAG-spinal formalism**  [Shen et al. 2008]

- Underspecifies predicate argument-adjunction distinction and subcategorization frames
- Allows efficient statistical processing in LTAG framework
- Elementary trees are in the spinal form
- Elementary trees *initial* and *auxiliary*
- A *spinal initial tree* is composed of a lexical spine from the root to the anchor and nothing else
  - the substitution nodes that are present in LTAG are missing
- A *spinal auxiliary tree* is composed of a lexical spine and a recursive spine from the root to the foot node.
LTAG-spinal formalism

- Has three operations `attach`, `adjoin`, `conjoin`
- **Attachment** is used to represent both substitution and sister adjunction in traditional LTAG
  - Argument and adjunct ambiguity is encoded in a single structure
  - Disambiguation left for further deep processing
- **Adjunction** is same as adjunction in traditional LTAG
- **Conjunction** is used to encode ambiguity of argument sharing with underspecification in coordination structures
LTAG-spinal Example

a parser which seems new to me
Outline

1. Introduction
   Lexicalized Tree Adjoining Grammar
   Hyderabad Dependency Treebank

2. LTAG-spinal formalism

3. Hindi LTAG-spinal treebank
   Extraction from HyDT
   Assigning spines to words

4. LTAG-spinal Parser

5. Results

6. Summary
Define the three spinal operations for Hindi

Extract the LTAG-spinal tree for each sentence from labeled dependency trees in HyDT

- Identify the substructures in the dependency tree which correspond to the 3 actions
- Transform the dependency tree to reflect LTAG-spinal kind of tree
- Assign spinal elementary trees to each node in the tree
Operations in Hindi

- **Adjoin** is used to represent non-projective structures
  - The head in a non-projective relation is combined using *adjoin* to its parent
  - Adjunction extends the *visibility* of the e-tree
- **Conjoin** is used to address NP as well as VP coordination
- In HyDT, connective is the head in a coordination construction and the conjuncts are its children. Whereas in LTAG-spinal,
  - the conjunct nearest to the connective’s parent is made the head of the coordination structure
  - the rest of the conjuncts are *conjoined* to the previous conjunct
  - the connective is *attached* to its immediate conjunct
- **Attach** is used to represent the rest of the relations
I lost the coins that I brought from Prais and Prague.
I lost the coins that I brought from Prais and Prague.
Assigning spines

- No phrase structure trees to extract spines
- Use heuristics
- Two issues in assigning spines heuristically
  - *Which* phrase categories should we include in the spines?
  - *Where* in the spine should each of the child spinal trees be connected?
- Minimal spines by retaining the core properties of the dependency treebank
Types of elementary spines

a) POS

b) CHK

POS

POS

c) XP

CHK

POS

d) XP

CHK

XP*

POS

e) XP

CHK

POS

XP*
Rules for assigning spines (a) & (b)

a)

The shortest spine contains just the POS tag of the word

Assigned to non-head leaf nodes of a chunk in the tree

b)

Contains the chunk label (CHK) and the POS tag

Assigned to words which are either

- head words of chunks with no modifiers outside the chunk
- or non-heads of chunks with intra chunk modifiers
I lost the coins I brought from Paris and Prague.
I lost the coins I brought from Paris and Prague

that coins lose PAST. which I-ERG Paris and Prague from bring PAST.
Rules for assigning spines (c)

- XP node at the top apart from the chunk label and the POS tag
- For words which have a modifier outside its own chunk
I lost the coins I brought from Paris and Prague
Rules for assigning spines (d) & (e)

- Auxiliary spine for words which are *adjoined* on the left/right
I lost the coins I brought from Paris and Prague.
I lost the coins I brought from Paris and Prague
Hindi LTAG-spinal treebank

- Convert the dependency trees in HyDT into an LTAG-spinal trees
- 1781 sentences with 17651 chunks and 33638 words
- 319 words with auxiliary trees (adjunctions)
Outline

1 Introduction
   - Lexicalized Tree Adjoining Grammar
   - Hyderabad Dependency Treebank

2 LTAG-spinal formalism

3 Hindi LTAG-spinal treebank
   - Extraction from HyDT
   - Assigning spines to words

4 LTAG-spinal Parser

5 Results

6 Summary
Bidirectional LTAG dependency parsing (BLDP) [Shen and Joshi 2008]

- Doesn’t follow a direction while searching (left-to-right or right-to-left)
- Picks the island with maximum confidence
- Takes advantage of the nearby hypotheses
- Uses LTAG Spinal treebank for statistical processing
- Defines features over neighboring words in the sentence as well as neighboring partial trees during parsing
Outline

1 Introduction
   Lexicalized Tree Adjoining Grammar
   Hyderabad Dependency Treebank

2 LTAG-spinal formalism

3 Hindi LTAG-spinal treebank
   Extraction from HyDT
   Assigning spines to words

4 LTAG-spinal Parser

5 Results

6 Summary
Results

<table>
<thead>
<tr>
<th>Sentence Local</th>
<th>SC(0)</th>
<th>SC(1)</th>
<th>SC(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC(0)</td>
<td>80.30%</td>
<td>80.75%</td>
<td>80.24%</td>
</tr>
<tr>
<td>LC(1)</td>
<td>80.44%</td>
<td>80.49%</td>
<td>80.00%</td>
</tr>
<tr>
<td>LC(2)</td>
<td>80.49%</td>
<td>80.49%</td>
<td>80.86%</td>
</tr>
</tbody>
</table>

Table: LTAG dependency accuracies for various context.

- LTAG dependency accuracy is the percentage of words with correct operations and parents.
- BLDP doesn’t produce spines and only produces LTAG dependencies.
Outline

1 Introduction
   Lexicalized Tree Adjoining Grammar
   Hyderabad Dependency Treebank

2 LTAG-spinal formalism

3 Hindi LTAG-spinal treebank
   Extraction from HyDT
   Assigning spines to words

4 LTAG-spinal Parser

5 Results

6 Summary
Summary

- An initial effort into creating LTAG-spinal treebank for Hindi
  - *adjoin* used only for wrapping adjunction which generates non-projective dependencies
  - *conjoin* used to handle all coordination structures (VP as well as NP coordination)
  - *attach* is used to connect rest of the nodes
  - Minimal spines are assigned using rules
- Trained a parser on the spinal treebank and achieved 80.86% LTAG dependency accuracy
Future work

- Richer LTAG-spinal treebank for Hindi
  - [Mannem et al. 2009] classified non-projectivity in HyDT into various classes and provided cues to identify them
  - Extend adjoin operation to all the linguistic phenomena which may result in non-projectivity
    - Phrase structure trees can be used to assign richer spines
- Parser being extended to generate full fledged LTAG-spinal tree by predicting spines
- Try out different ways of assigning spines and see how it effects LTAG parsing
- Use LTAG-spinal trees with dependency labels for Semantic Role Labeling
Questions?
Thank You

Prashanth Mannem, Himani Chaudhry, and Akshar Bharati.
2009.
Insights into non-projectivity in hindi.

Libin Shen and Aravind Joshi.
2008.
LTAG dependency parsing with bidirectional incremental construction.
Bibliography

LTAG-spinal Example

Diagram of a LTAG-spinal tree structure with nodes labeled with parts of speech and words. The tree includes nodes for S, VP, NP, V, N, DT, IN, TO, RB, and other parts of speech, with words like "the", "market", "could", "continue", "1", "to", "soften", "in", "months", and "ahead".
a parser which seems new to me
LTAG-spinal Example

a parser which seems new and interesting to me