A Statistical Parser for Hindi

Pranjali Kanade

Anoop Sarkar
Vivek Mehta
Mona Parakh
T. Papi Reddy

AU-KBC Center, Madras Institute of Technology
December 17-31, 2001
Corpus-Based Natural Language Processing Workshop

A Statistical Parser for Hindi
Initial Goals

• Build a statistical parser for Hindi
• Train on the Hindi Treebank (built at LTRC, Hyderabad)
• Disambiguate existing rule-based parser (Papi’s Parser) using the Treebank
• Active learning experiments: Informative sampling of data to be annotated

(provides single-best parse for a given input)
Annotated corpus for Hindi, "AnnCorra" prepared at LTRC, IIT, Hyderabad.

Corpus description: extracts from Premchand's novels.

Corpus size: 338 sentences.

Manually annotated corpus; marked for verb-argument relations.

Initial Linguistic Resources
Goals: Reconsidered

- Papir’s parser and sentence units

- Corpus Cleanup and Correction

- Default rules and explicit dependency trees

- Various models of parsing based on the Treebank

- Trigram tagger/chunker

- Probabilistic CFG parser (stemming, no smoothing)

- Fully lexically-statistical parser (with smoothing)

- Papi’s parser and sentence units

- Various models of parsing based on the Treebank

Corpus Cleanup and Correction
Problems in the Corpus:

- Inconsistency in tags
- Discrepancy in the use of tags
- Improper local word grouping
- Inter-annotator consistency on labels

Cause of these problems: Inter-annotator consistency on labels.
Corpus Cleanup and Correction

Solution: Annotators who were part of the team manually corrected

- Inconsistency of tags resolved.
- Resolved the discrepancies in the tags.
- Problems of local word grouping resolved.
- Explicitly marked the clause boundaries to disambiguate long complex sentences without punctuation in the corpus.
Explicit dependencies are not marked

Evaluated the default rules and built a program to convert original cor-

Default rules are listed in the guidelines

Raw corpus:

\{ \text{nasha-hta-ho-gayA:VA} \wedge \text{v}\text{ de sa } \text{mi ti ta-mem}/K7.1 [\text{ha ra-pra ra bage}]/K1 \}
DefaultrulesandExplicitDependencyTrees

{ nashitra-ho-gaya:
    ▼

    ▼
    dasa

    ▼
    hara-bhara

    ▼
    miniya-meli

    ▼
    dasa

nashtar-ho-gaya:
{ dasa miniya-meli [K7.1] hara-bhara baga [K1] }
DefaultrulesandExplicitDependencyTrees

Defaultrulescouldnothandle24outof334sentences

(added as parent of all clauses)
ad-hoc defaultrules for multiple sentence units within a single sentence

Default rules could not handle 24 out of 334 sentences

Default rules and Explicit Dependency Trees
Input:

{\text{tahasIlamadarasAbarA.Nva_ke} /6
\text{prathamAdhyApakamuMshIbhavAnIsahAya_ko} /k1
\text{bAgavAnI_kA} /6
\text{kuchha} /adv
\text{vyasana_thA} /v}

Converted to representation for tagger:

{\text{vyasana_thA} /v
\text{kuchha} /adv
\text{bAgavAnI_kA} /6
\text{prathamAdhyApakamuMshIbhavAnIsahAya_ko} /k1
\text{tahasIlamadarasAbarA.Nva_ke} /6}

Tigram_Tagger/Chunker

Input:

Tigram_Tagger/Chunker

Converted to representation for tagger:

{\text{vyasana_thA} /v
\text{kuchha} /adv
\text{bAgavAnI_kA} /6
\text{prathamAdhyApakamuMshIbhavAnIsahAya_ko} /k1
\text{tahasIlamadarasAbarA.Nva_ke} /6}
TrigramTagger/Chunker

Bootstrapped using existing supertagger code

http://www.cis.upenn.edu/~xtag/

Testing on training data performance:

- tag accuracy: 71.8%
- chunk accuracy: 71.8%

70-30 training-test split

Unseen Test data

- tag accuracy: 55.17%
- chunk accuracy: 95.69%

Trigram Tagger/Chunker
ProbabilisticCFGParser

Extracted context-free rules from the Treebank

Estimated probabilities for each rule using counts from the Treebank

Extracted context-free rules from the Treebank

Used some existing code written earlier for Prob CKY parsing

Used PCFG parser to compute the best derivation for a given sentence

http://www.cis.upenn.edu/~anoop/distrb/t_trees/ckycfg/
<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>1 min 27 secs</td>
</tr>
<tr>
<td>Number of Valid Sentence</td>
<td>297</td>
</tr>
<tr>
<td>Number of Skip Sentence</td>
<td>0</td>
</tr>
<tr>
<td>Number of Error Sentence</td>
<td>13</td>
</tr>
<tr>
<td>Number of Sentence</td>
<td>310</td>
</tr>
<tr>
<td>Bracketing Precision</td>
<td>86.29</td>
</tr>
<tr>
<td>Bracketing Recall</td>
<td>76.94</td>
</tr>
<tr>
<td>Complete Match</td>
<td>48.82</td>
</tr>
<tr>
<td>Average Crossing</td>
<td>0.12</td>
</tr>
<tr>
<td>No Crossing</td>
<td>91.25</td>
</tr>
<tr>
<td>2 or less crossing</td>
<td>99.33</td>
</tr>
</tbody>
</table>
Probabilistic CFG Parser: Results with Stemming on Training Data

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Error</td>
<td>13</td>
</tr>
<tr>
<td>Number of Skip</td>
<td>0</td>
</tr>
<tr>
<td>Number of Valid</td>
<td>297</td>
</tr>
<tr>
<td>Number of Sentence</td>
<td>310</td>
</tr>
<tr>
<td>Bracketing Recall</td>
<td>59.74</td>
</tr>
<tr>
<td>Bracketing Precision</td>
<td>60.05</td>
</tr>
<tr>
<td>Complete match</td>
<td>25.59</td>
</tr>
<tr>
<td>Average crossing</td>
<td>0.58</td>
</tr>
<tr>
<td>No crossing</td>
<td>66.33</td>
</tr>
<tr>
<td>2 or less crossing</td>
<td>94.95</td>
</tr>
</tbody>
</table>
Probabilistic CFG Parser: Unseen Data: Test Data: 20%

2 or less crossing = 91.23%
No crossing = 73.68%
Average crossing = 0.53%
Complete match = 5.26%
Bracketing Precision = 53.45%
Bracketing Recall = 37.96%
Number of Valid sentence = 57
Number of Skip sentence = 0
Number of Error sentence = 5
Number of Sentence = 62
Lexicalized StatParser: Building up the parse tree
Lexicalized StatParser: Building up the Parse Tree.
\[
(\text{p}_3(\text{top}, \text{w} \text{a}), \text{top}) \times (\text{p}_2(\text{top}, \text{w} \text{a}), \text{top}) \\
\times (\text{p}_1(\text{top}, \text{w} \text{a}), \text{top}) = (\text{p}_3(\text{top}, \text{w} \text{a}), \text{top})
\]

<table>
<thead>
<tr>
<th>(\text{p}_3(\text{top}))</th>
<th>(\text{p}_2(\text{top}))</th>
<th>(\text{p}_1(\text{top}))</th>
<th>(\text{p}(\text{top}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Lexicalized Sparser: Start Probabilities
\[
\left( \rightarrow \land \neg \text{tag}, \land n \neg \text{tag} \right)_{l w} D
\times \left( \rightarrow \land \neg \text{tag}, \land n \neg \text{tag} \right)_{l w} D
\times \left( \rightarrow \land \neg \text{tag}, \land n \neg \text{tag} \right)_{l w} D
\]
\[
= \left( \rightarrow \land \neg \text{tag}, \land n \neg \text{tag} \right)_{l w} D
\]

| (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | 3 |
| (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | 2 |
| (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | 1 |
| (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | (\text{tag}, \text{tag})_{l w} | 0 |

Lexicalized SPARQL: Modification Probabilities
\[
\left( \frac{1}{\forall_a b q} \right) \epsilon_{ud} P
\times \left( \frac{1}{\forall_a b} \right) \epsilon_{ud} P
\times \left( \frac{1}{\forall_a b} \right) \epsilon_{ud} P
\times \left( \frac{1}{\forall_a b} \right) \epsilon_{ud} P
= \left( \frac{1}{\forall_a b q} \right) \epsilon_{ud} P
\]

<table>
<thead>
<tr>
<th>(u)</th>
<th>(v)</th>
<th>(w)</th>
<th>(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>(\epsilon_{ud} P)</td>
</tr>
</tbody>
</table>
Contributions of the project

Cleaned and clause-bracketed Hindi Treebank

New NLP tools developed for Hindi:
- Trigram tagger/chunker (with evaluation)
- Probabilistic CFG parser (with evaluation)
- Lexicalized statistical parsing model (still in progress)

Conversion of Anncora into dependency trees

Implementation of default rules listed in the Anncora guidelines

Implementation of default rules listed in the Anncora guidelines
Future Work:

- Corpus development and bugfixes

- Corpus: fix remaining errors in annotated clause boundaries

- Eliminate stemming from PCFG parser

- Combine part-of-speech information into the corpus

  - Part-of-speech into PCFG and lexicalized

  - Current assumption: LWG gets 100% of the groups correct

  - Evaluate the local word grouper performance

Future Work: Corpus development and bugfixes
Future Work: Lexicalized Statistical Parser

• Write a paper describing the project
• Clean up the clause-bracketing annotation in the corpus
• Continue implementation and evaluation of lexicalized statistical parser
• Active learning experiments: Informative sampling of data to be annotated based on the parser
Future Work: Active Learning

An annotation combined with learning

Annotation $\rightarrow$ Machine Learner $\rightarrow$ Annotation

: Learning

Model we can explore (similar to ideas in online learning and active learning)

Learning has no impact on the original annotated data

Current learning model: fixed size of training and test data
Future Work: Improving Existing Rule-based Parser for Hindi

- Transformation rules that modify Karaka charts based on tense-aspect-modality.
- Verb-argument dependency: Demanding (Karaka) charts.
- Dependency parser for Indian languages.
Future Work: Improving Existing Rule-based Parser for Hindi

Current Limitations of the Parser:

- Insufficient lexical resources (≈ 19 Demand charts)
- Local word grouper performs only on verb chunks
- Noun chunks that are larger than basal noun-phrases have to be handled
- Creates number of spurious analyses when handling multiple-clause sentences
Future Work: Improving Existing Rule-based Parser for Hindi

Current directions for improvement:

- Heuristics for specifying clausal boundaries.
- Dealing with ellipsis, negation, etc.
- Using default Karaka charts for unknown verbs.
- Learning the Karaka charts and the transformation rules from the annotated corpus.
- Associating adjectives with the corresponding nouns.