Penn Chinese Treebank: Linguistic Characteristics

[Xue, Xia, Chou, & Palmer 2005]
- Source: Xinhua news service articles
- Segmented text
  - It's harder when you compose in errors from word segmentation as well...
- Nearly identical sentence length as WSJ Treebank
- Annotated in a much more GB-like style
  - CP and IP
  - (Fairly) Consistent differentiation of modifiers from complements

Syntactic sources of ambiguity

- English: PP attachment (well understood); coordination scoping (less well understood)
- Chinese: modifier attachment less of a problem, as verbal modifiers & direct objects aren't adjacent, and NP modifiers are overtly marked.

Error tabulation

[Levy and Manning 2003]

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat as multilevel</td>
<td>6</td>
</tr>
<tr>
<td>IP</td>
<td>1</td>
</tr>
<tr>
<td>NP-NP Modification</td>
<td>False Positive 13</td>
</tr>
<tr>
<td>Prepositional Modification</td>
<td>False Negative 26</td>
</tr>
<tr>
<td>Prepositional Modification</td>
<td>False Positive 5</td>
</tr>
<tr>
<td>Coordination Attachments</td>
<td>False Negative 7</td>
</tr>
<tr>
<td>VP as N</td>
<td>17</td>
</tr>
<tr>
<td>N as V</td>
<td>5</td>
</tr>
<tr>
<td>Other tagging errors</td>
<td>14</td>
</tr>
</tbody>
</table>
Tagging errors

- N/V tagging a major source of parse error
  - V as N errors outnumber N as V by 3.2:1
  - Corpus-wide N:V ratio about 2.5:1
  - N/V errors can cascade as N and V project different phrase structures (NP is head-final, VP is not)
- Possible disambiguating factors:
  - derivational or inflectional morphology
  - function words in close proximity (c.f. English the, to)
  - knowledge of prior distribution for tag frequency
  - non-local context

- Chinese has little to no morphological inflection
  - As a result, the part-of-speech ambiguity problem tends to be greater than in English.
  - Increase
  - Increase
  - Increasing

- Function words are also much less frequent in Chinese

- Tagging error experiment [Levy and Manning 2003]
  - N/V error experiment: merge all N and V tags in training data
  - Results in 5.1% F1 drop for vanilla PCFG; 1.7% drop for enhanced model
  - In English, with equivalent-sized training set, tag merge results in 0.21% drop in recall and 0.06% increase in precision for vanilla PCFG
  - Indicates considerable burden on POS priors in Chinese

- Chinese lexicalized parser learning curve [Levy and Manning 2003]
  - Chinese Treebank 3.0 release
  - (100% ~300,000 words)

- A hotly debated case: German
  - Linguistic characteristics, relative to English
  - Ample derivational and inflectional morphology
  - Freer word order
  - Verb position differs in matrix/embedded clauses
  - Main ambiguities similar to English
  - Most used corpus: Negra
    - ~400,000 words newswire text
    - Flatter phrase structure annotations (few PPs!)
    - Explicitly marked phrasal discontinuities
  - Newer Treebank: TueBaDz
    - ~470,000 words newswire text (27,000 sentences)
    - [Not replacement; different group; different style]

- German results
  - Dubey and Keller [ACL 2003] present an unlexicalized PCFG outperforming Collins on NEGRA – and then get small wins from a somewhat unusual sister-head model, but...
  - \[L_{Prec} \quad L_{Rec} \quad F1\]
    - D&K PCFG Baseline 66.69 70.56 68.57
    - D&K Collins 66.07 67.91 66.98
    - D&K Sister-head all 70.93 71.32 71.12
  - Stanford PCFG Baseline 72.72 73.64 73.59
  - Stanford Lexicalized 74.61 76.23 75.41
  - See also [Arun & Keller ACL 2005, Kübler & al. EMNLP 2006]
Prominent ambiguities

- PP attachment

- Sentential complement vs. relative clause

Dependency parsing

- A sentence is parsed by relating each word to other words in the sentence which depend on it.
- The idea of dependency structure goes back a long way
  - To Pāṇini’s grammar (c. 5th century BCE)
- Constituency is a new-fangled invention
  - 20th century invention
- Modern work often linked to work of L. Tesniere (1959)
  - Dominant approach in “East” (Eastern bloc/East Asia)
- Among the earliest kinds of parsers in NLP, even in US:
  - David Hays, one of the founders of computational linguistics, built early (first?) dependency parser (Hays 1962)

Dependency structure

- Words are linked from head (regent) to dependent
- Warning! Some people do the arrows one way; some the other way (Tesniere has them point from head to dependent...).
- Usually add a fake ROOT so every word is a dependent

Relation between CFG to dependency parse

- A dependency grammar has a notion of a head
- Officially, CFGs don’t
- But modern linguistic theory and all modern statistical parsers (Charniak, Collins, Stanford, ...) do, via hand-written phrasal “head rules”:
  - The head of a Noun Phrase is a noun/number/adj/…
  - The head of a Verb Phrase is a verb/modal/…
- The head rules can be used to extract a dependency parse from a CFG parse (follow the heads)
- A phrase structure tree can be got from a dependency tree, but dependents are flat (no VPI)
Propagating head words

- Small set of rules propagate heads

Extracted structure

NB. Not all dependencies shown here

- Dependencies are inherently untyped, though some work like Collins (1996) types them using the phrasal categories

Sources of information:

- Bilexical dependencies
- Distance of dependencies
- Valency of heads (number of dependents)

A word’s dependents (adjuncts, arguments) tend to fall near it in the string.

Dependency Grammar Cubic Recognition/Parsing (Eisner & Satta, 1999)

- Triangles: span over words, where tall side of triangle is the head, other side is dependent, and no non-head words expecting more dependents

- Trapezoids: span over words, where larger side is head, smaller side is dependent, and smaller side is still looking for dependents on its side of the trapezoid
Dependency Grammar Cubic Recognition/Parsing (Eisner & Satta, 1999)

It takes two to tango goal

One trapezoid per dependency

A triangle is formed with some left (or right) subtrees.

Cubic Recognition/Parsing (Eisner & Satta, 1999)

\[ O(n^3) \] combinations

\[ O(n^3) \] combinations

\[ O(n^3) \] combinations

Gives \[ O(n^3) \] dependency grammar parsing

Evaluation of Dependency Parsing:
Simply use (labeled) dependency accuracy

1 2 3 4 5
1 2 3 4 5
We SUBJ
2 0 eat ROOT
3 5 the DET
4 2 cheese MOD
5 2 sandwich SUBJ

Accuracy = \[ \frac{\text{number of correct dependencies}}{\text{total number of dependencies}} \]

2 / 5 = 0.40
40%

McDonald et al. (2005 ACL):
Online Large-Margin Training of Dependency Parsers

- Builds a discriminative dependency parser
- Can condition on rich features in that context
  - Best-known recent dependency parser
  - Lots of recent dependency parsing activity connected with CoNLL 2006/2007 shared task
- Doesn’t/can’t report constituent LP/LR, but evaluating dependencies correct:
  - Accuracy is similar to but a fraction below dependencies extracted from Collins:
    - 90.9% vs. 91.4% ... combining them gives 92.2% [all lengths]
    - Stanford parser on length up to 40:
      - Pure generative dependency model: 85.0%
      - Lexicalized factored parser: 91.0%

McDonald et al. (2005 ACL):
Online Large-Margin Training of Dependency Parsers

- Score of a parse is the sum of the scores of its dependencies
- Each dependency is a linear function of features times weights
- Feature weights are learned by MIRA, an online large-margin algorithm
  - But you could think of it as using a perceptron or maxent classifier
- Features cover:
  - Head and dependent word and POS separately
  - Head and dependent word and POS bigram features
  - Words between head and dependent
  - Length and direction of dependency

Extracting grammatical relations from statistical constituency parsers

[de Marneffe et al. LREC 2006]

- Exploit the high-quality syntactic analysis done by statistical constituency parsers to get the grammatical relations [typed dependencies]
- Dependencies are generated by pattern-matching rules