Dependency Parsing

Introduction

Many slides are adapted from Chris Manning
Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies.

The arrow connects a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate).

Usually, dependencies form a tree (connected, acyclic, single-head).
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.

The closure of dependencies give constituency from a dependency tree.

But the dependents of a word must be at the same level (i.e., “flat”) – there can be no VP!
Methods of Dependency Parsing

1. Dynamic programming (like in the CKY algorithm)
   You can do it similarly to lexicalized PCFG parsing: an $O(n^5)$ algorithm
   Eisner (1996) gives a clever algorithm that reduces the complexity to $O(n^3)$, by producing parse items with heads at the ends rather than in the middle

2. Graph algorithms
   You create a Maximum Spanning Tree for a sentence
   McDonald et al.’s (2005) MSTParser scores dependencies independently using a ML classifier (he uses MIRA, for online learning, but it could be MaxEnt)

3. “Deterministic parsing”
   Greedy choice of attachments guided by machine learning classifiers
   MaltParser (Nivre et al. 2008) – transition based, shift-reduce
Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

1. Bilexical affinities  
   [issues → the] is plausible

2. Dependency distance  
   mostly with nearby words

3. Intervening material
   Dependencies rarely span intervening verbs or punctuation

4. Valency of heads
   How many dependents on which side are usual for a head?

ROOT Discussion of the outstanding issues was completed.
Projectivity

- Dependencies from a CFG tree using heads, must be *projective*
  - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- But dependency theory normally does allow non-projective structures to account for displaced constituents
  - You can’t easily get the semantics of certain constructions right without these nonprojective dependencies

Who did Bill buy the coffee from yesterday?
Quiz question!

- Consider this sentence:
  Retail sales drop in April cools afternoon market trading.
- Which word are these words a dependent of?
  1. sales
  2. April
  3. afternoon
  4. trading
Dependency Parsing

Introduction
Evaluation
Evaluation of Dependency Parsing: (labeled) dependency accuracy

Gold
1 2  She nsubj
2 0  saw root
3 5  the det
4 5  video nn
5 2  lecture dobj

Parsed
1 2  She nsubj
2 0  saw root
3 4  the det
4 5  video nsubj
ccomp
5 2  lecture
dobj

Acc = \# correctdeps / \# of deps

UAS = \frac{4}{5} = 80\%
LAS = \frac{2}{5} = 40\%
Representative performance numbers

- The CoNLL-X (2006) shared task provides evaluation numbers for various dependency parsing approaches over 13 languages
  - Performance varies depending greatly on language/treebank
- Here we give a few UAS numbers for English to allow some comparison to constituency parsing

<table>
<thead>
<tr>
<th>Parser</th>
<th>UAS%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sagae and Lavie (2006) ensemble of dependency parsers</td>
<td>92.7</td>
</tr>
<tr>
<td>Charniak (2000) generative, constituency</td>
<td>92.2</td>
</tr>
<tr>
<td>Collins (1999) generative, constituency</td>
<td>91.7</td>
</tr>
<tr>
<td>McDonald and Pereira (2005) – MST graph-based dependency</td>
<td>91.5</td>
</tr>
<tr>
<td>Yamada and Matsumoto (2003) – transition-based dependency</td>
<td>90.4</td>
</tr>
</tbody>
</table>
Evaluation
Dependencies encode relational structure

Relation Extraction with Dependencies
Dependency paths identify relations like protein interaction

[Erkan et al. EMNLP 07, Fundel et al. 2007]

```
KaiC nsubj interacts prep_with SasA
KaiC nsubj interacts prep_with SasA conj_and KaiA
KaiC nsubj interacts prep_with SasA conj_and KaiB
```

![Bar chart showing dependency distance and linear distance for different ranges.]

- Red bars represent dependency distance.
- Blue bars represent linear distance.

Axis labels:
- X-axis: Dependency distance categories (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, >10).
- Y-axis: Frequency counts.
Dependencies
encode relational structure

Relation Extraction with Dependencies