Introduction to NLP

Dependency Parsing
Techniques (1)

• Constraint-based methods
  – Maruyama 1990, Karlsson 1990
  – Example
    • \( \text{word(pos(x))} = \text{DET} \Rightarrow (\text{label(X)} = \text{NMOD}, \text{word(mod(x))} = \text{NN}, \text{pos(x)} < \text{mod(x)}) \)
    • A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
  – NP-complete problem; heuristics needed

• Constraint graph
  – For initial constraint graph using a core grammar: nodes, domains, constraints
  – Find an assignment that doesn’t contradict any constraints. If more than one assignment exists, add more constraints.
Techniques (2)

• Dynamic programming
  – CKY – similar to lexicalized PCFG, cubic complexity (Eisner 96)
Techniques (3)

• Deterministic parsing
  – Covington 2001

• Graph-based methods
  – Maximum spanning trees (MST)
    • MST Parser by McDonald et al.
  – (next)

• Transition-based
  – MaltParser by Nivre et al.
  – and its variants
  – (later lecture)
The Eisner (1996) Method

Figure 4. Comparison of the chart items in the CKY algorithm (left) and Eisner’s algorithm (right).

[Figure from Nivre 2013]
• Split-head representation
  – Represent half-trees in the CKY table
  – Keep track whether the head is on the left or the right

• Two operations
  – Combine two half-trees by adding a dependency arc between their heads. This creates an incomplete half-tree.
  – Then combine an incomplete half-tree with a complete half-tree


Figure 5. Eisner’s cubic-time algorithm for arc-factored dependency parsing. Items of the form $C[i][j][d][c]$ represent subgraphs spanning from word $i$ to $j$; $d = \leftarrow$ if the head is at the right periphery and $d = \rightarrow$ if the head is at the left periphery (the arrow pointing towards the dependents); $c = 1$ if the item is complete (that is, contains a head and its complete half-tree on the left/right) and $c = 0$ if the item is incomplete (that is, contains a head linked to a dependent with both inside half-trees).
The Eisner (1996) Method

Figure 4. Comparison of the chart items in the CKY algorithm (left) and Eisner’s algorithm (right).

[Figure from Nivre 2013]
The Eisner (1996) Method

First-order Projective Parsing

Eisner algorithm
[Eisner 1996]

Chart items either:
1) Create a new dependency
2) Absorb left/right subtree

Each chart item stores two indexes:
1) left boundary
2) right boundary

All operations require 3 indexes: $O(n^3)$

[Slide from McDonald and Nivre]
The Eisner (1996) Method

Projective dependency parsing: The Eisner Algorithm

ROOT She gave him the ball
ROOT (She ← gave) (gave → him) (the ← ball)
[ ROOT ] [She ← gave] [gave → him] [the ← ball]

ROOT she gave him the ball

( [ ROOT ] → [She ← gave]) ([gave → him] → [the ← ball])

ROOT she gave him the ball

( [ ROOT ] → [She ← gave]) ([gave → him] → [the ← ball])

( [ ROOT ] → [She ← gave]) → ([gave → him] → [the ← ball])

ROOT she gave him the ball

[Slide from McDonald and Nivre]
Introduction to NLP

Graph-based Dependency Parsing
Graph-based Dependency Parsing

• Background
  – McDonald et al. 2005

• Idea
  – Dependency parsing is equivalent to search for a maximum spanning tree in a directed graph.
  – Chu and Liu (1965) and Edmonds (1967) give an efficient algorithm for finding MST for directed graphs.
MST Parser example

- Consider the sentence “John saw Mary”
- Recursively remove cycle
- The Chu-Liu-Edmonds algorithm gives the MST on the right hand side (right). This is in general a non-projective tree.
Notes

- Complexity
  - Interestingly, it’s $O(n^2)$, compared with $O(n^3)$ for Eisner, even though MST is non-projective.
NLP