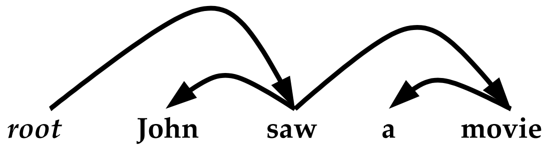
Global Linear Models Part 3: The Perceptron Algorithm for Dependency Parsing

Michael Collins, Columbia University

Overview

- Dependency parsing
- GLMs for dependency parsing
- Results from McDonald (2005)

Unlabeled Dependency Parses

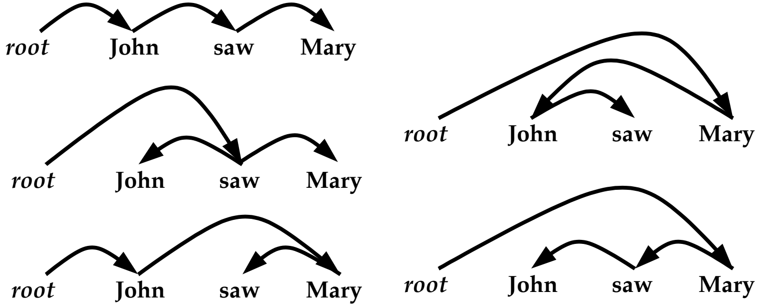


- `root` is a special *root* symbol

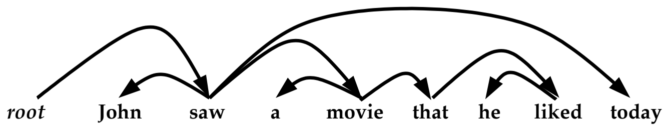
- Each dependency is a pair (h, m) where h is the index of a head word, m is the index of a modifier word. In the figures, we represent a dependency (h, m) by a directed edge from h to m.

- Dependencies in the above example are $(0, 2)$, $(2, 1)$, $(2, 4)$, and $(4, 3)$. (We take 0 to be the root symbol.)

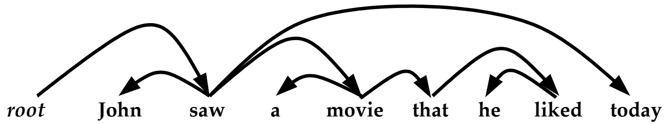
All Dependency Parses for *John saw Mary*



A More Complex Example



Conditions on Dependency Structures

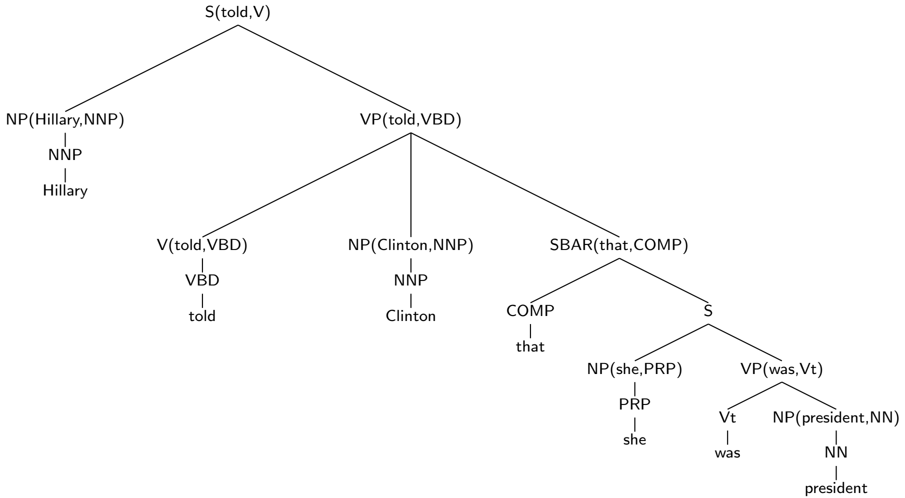


- The dependency arcs form a *directed tree*, with the `root` symbol at the root of the tree.
  (Definition: A directed tree rooted at *root* is a tree, where for every word w other than the root, there is a directed path from *root* to w.)

- There are no "crossing dependencies".
  Dependency structures with no crossing dependencies are sometimes referred to as **projective** structures.

Dependency Parsing Resources

- CoNLL 2006 conference had a "shared task" with dependency parsing of 12 languages (Arabic, Chinese, Czech, Danish, Dutch, German, Japanese, Portuguese, Slovene, Spanish, Swedish, Turkish). 19 different groups developed dependency parsing systems. (See also CoNLL 2007).
- PhD thesis on the topic: Ryan McDonald, *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*, University of Pennsylvania.
- For some languages, e.g., Czech, there are "dependency banks" available which contain training data in the form of sentences paired with dependency structures
- For other languages, we can extract dependency structures from treebanks



Unlabeled Dependencies:

(0,2)	(for `root` $\rightarrow$ told)
(2,1)	(for told $\rightarrow$ Hillary)
(2,3)	(for told $\rightarrow$ Clinton)
(2,4)	(for told $\rightarrow$ that)
(4,6)	(for that $\rightarrow$ was)
(6,5)	(for was $\rightarrow$ she)
(6,7)	(for was $\rightarrow$ president)

Efficiency of Dependency Parsing

- PCFG parsing is $O(n^3G^3)$ where n is the length of the sentence, G is the number of non-terminals in the grammar
- Lexicalized PCFG parsing is $O(n^5G^3)$ where n is the length of the sentence, G is the number of non-terminals in the grammar.
- Unlabeled dependency parsing is $O(n^3)$.

Overview

- Dependency parsing
- Global Linear Models (GLMs) for dependency parsing
- Results from McDonald (2005)

GLMs for Dependency parsing

- x is a sentence
- $\mathbf{GEN}(x)$ is set of all dependency structures for x
- $\mathbf{f}(x, y)$ is a feature vector for a sentence x paired with a dependency parse y

GLMs for Dependency parsing

- To run the perceptron algorithm, we must be able to efficiently calculate

$$\arg\max_{y \in \mathbf{GEN}(x)} \mathbf{w} \cdot \mathbf{f}(x, y)$$

- Local feature vectors: define

$$\mathbf{f}(x, y) = \sum_{(h,m) \in y} \mathbf{g}(x, h, m)$$

  where $\mathbf{g}(x, h, m)$ maps a sentence x and a dependency (h, m) to a local feature vector

- Can then use dynamic programming to calculate

$$\arg\max_{y \in \mathbf{GEN}(x)} \mathbf{w} \cdot \mathbf{f}(x, y) = \arg\max_{y \in \mathbf{GEN}(x)} \sum_{(h,m) \in y} \mathbf{w} \cdot \mathbf{g}(x, h, m)$$

Definition of Local Feature Vectors

- Features from McDonald et al. (2005):
  - Note: define w_i to be the i'th word in the sentence, t_i to be the part-of-speech (POS) tag for the i'th word.
  - *Unigram* features: Identity of w_h. Identity of w_m. Identity of t_h. Identity of t_m.
  - *Bigram* features: Identity of the 4-tuple $\langle w_h, w_m, t_h, t_m \rangle$. Identity of sub-sets of this 4-tuple, e.g., identity of the pair $\langle w_h, w_m \rangle$.
  - *Contextual features:* Identity of the 4-tuple $\langle t_h, t_{h+1}, t_{m-1}, t_m \rangle$. Similar features which consider t_{h-1} and t_{m+1}, giving 4 possible feature types.
  - *In-between features:* Identity of triples $\langle t_h, t, t_m \rangle$ for any tag t seen between words h and m.

Overview

- Dependency parsing
- Global Linear Models (GLMs) for dependency parsing
- Results from McDonald (2005)

Results from McDonald (2005)

Method	Accuracy
Collins (1997)	91.4%
1st order dependency	90.7%
2nd order dependency	91.5%

- Accuracy is percentage of correct unlabeled dependencies
- Collins (1997) is result from a lexicalized context-free parser, with dependencies extracted from the parser's output
- 1st order dependency is the method just described.
  2nd order dependency is a model that uses richer representations.
- Advantages of the dependency parsing approaches: simplicity, efficiency ($O(n^3)$ parsing time).