Dependency Parsing
Lecture 19: November 13, 2013

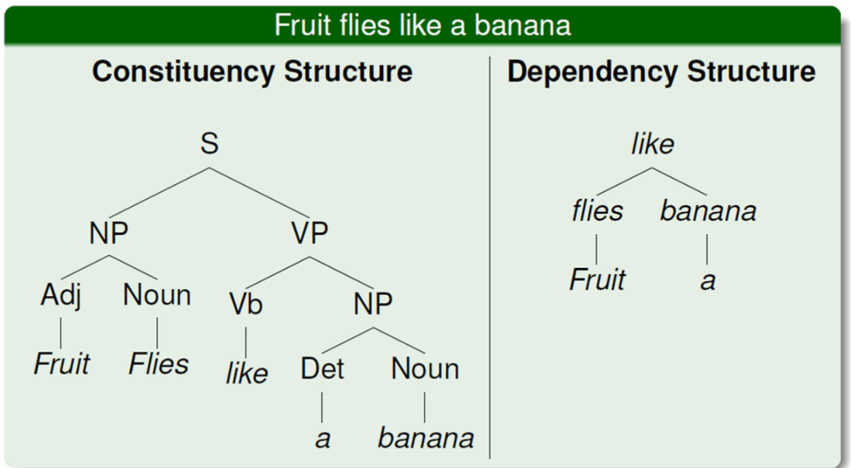
CS886-2 Natural Language Understanding

University of Waterloo

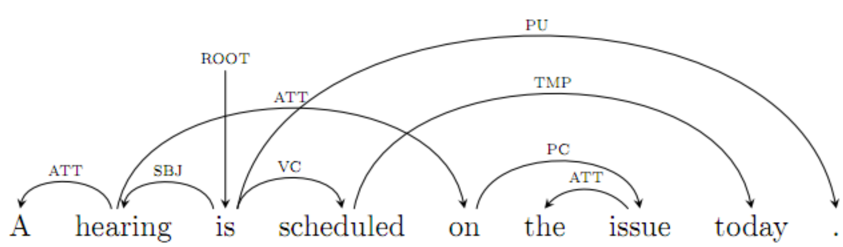
Parsing

- Constituency Parsing
  - Hierarchical phrase based parsing
- Shallow parsing/syntactic chunking
  - Flat phrase based parsing
- Dependency Parsing
  - Identify dependencies/relations between words

Constituent vs Dependency Tree



Labeled Dependency Parsing



Dependency Graph

- Edge: head → dependent
  - Head: governing word
  - Dependent: attribute or modifier of the head
- Labels:
  - Relation type
  - E.g., ROOT, ATT (attribute), SUBJ (subject), OBJ (object), VC (verb clause), NP (noun phrase), PU (punctuation), TMP (temporal marker), PP (preposition phrase), NMOD (noun modifier)

Dependency Graph

- Three principles [Robinson, 1970]:
  1. One and only one word is the root
  2. All other words have one and only one parent
  3. If w_i is the parent of w_j and w_k is in between w_i and w_j then w_k must be a descendent of w_i.
- Notes
  - 1. and 2. imply that the graph is a directed tree
  - 3. implies that the directed tree is projective

Projective Dependency Graph

- Projective property:
  - Edges do not cross each other
- Most sentences are projective, but not all
  - 10-25% of sentences are non-projective
  - ~1% of arcs do not respect principle 3.
  - Czech, Turkish and German have more freedom in the order of the words, which often leads to non-projective sentences

Projective vs Non-Projective

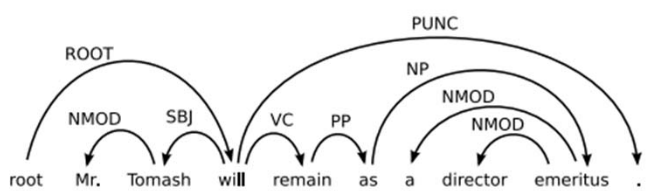


Figure 1: A projective dependency graph.

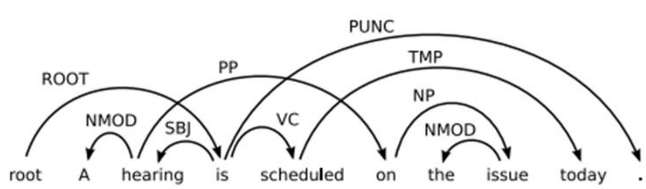


Figure 2: Non-projective dependency graph.

Algorithms

- Transition-based Parsers
  - Process sentence in a forward fashion
  - Focus on projective sentences (but non-projective extensions possible)
- Graph-based Parsers
  - Process entire sentence at once
  - Focus on non-projective sentences (but projective restrictions possible)

Graph-based Parsers

- Scoring function
  - Estimate weight for each edge based on labeled data
- Maximum spanning Tree
  - Search for directed spanning tree that maximizes the sum of he weights of the edges
- Label edges
  - Estimate the labels by sequence tagging

Maximum Spanning Tree

- Suppose we have a weight for each possible edge between any pair of nodes
- Finding the best dependency tree can be cast as a maximum spanning tree problem
  - Non-projective tree:
    - Plain maximum spanning tree (MST)
    - Complexity: $O(n^2)$
  - Projective tree:
    - Constrained spanning tree
    - Complexity: $O(n^3)$

Chu-Liu-Edmonds MST

- Maximum Spanning Tree Algorithm
  - Greedily assign highest scoring incoming edge to each vertex
  - If graph is a not a tree (there must be a cycle)
    - Contract each cycle into a single node
    - Recursively call MST on compacted graph
    - Expand back the compacted graph
  - Return graph

Chu-Liu-Edmonds MST

- Contraction
  - Replace each cycle by a new node c
  - For each $y \notin$ cycle: if $\exists x \in cycle$
    - Add $c \rightarrow y$ with weight
$$w(c \rightarrow y) = \max_{x \in cycle} w(x \rightarrow y)$$
  - For each $x \notin$ cycle: if $\exists y \in cycle$
    - Add $x \rightarrow c$ with weight
$$w(x \rightarrow c) = \max_{y \in cycle} w(x \rightarrow y) - w(pa(y) \rightarrow y) + \sum_{y' \in cycle} w(pa(y') \rightarrow y')$$

Example

- John saw Mary