Mining Unstructured Data
7. Dependency parsing

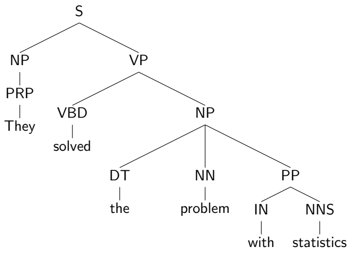
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

1. Dependency Trees
2. Dependency Parsing
3. Graph-based Dependency Parsing
   - Algorithm based on Maximum-Spanning Trees
4. Transition-Based Dependency parsers
   - Arc-Standard algorithm

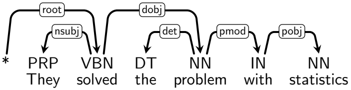
Theories of Syntactic Structure

Constituent Trees



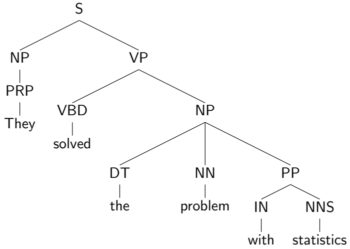
- Main element: constituents
- Constituent: linguistic unit subsuming a word sequence

Dependency Trees



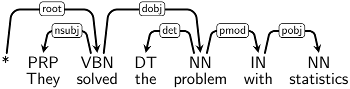
Theories of Syntactic Structure

Constituent Trees



- Main element: constituents
- Constituent: linguistic unit subsuming a word sequence

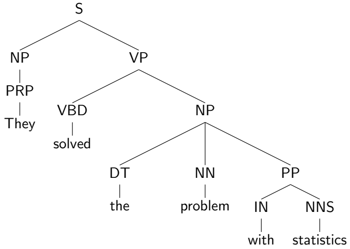
Dependency Trees



- Main element: dependency
- Dependency: a word has a grammatical function with respect to another word

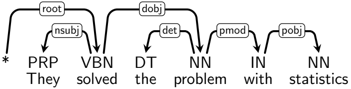
Theories of Syntactic Structure

Constituent Trees



- Main element: constituents
- Constituent: linguistic unit subsuming a word sequence
- Focus on combinations of constituents
- Builds nested trees

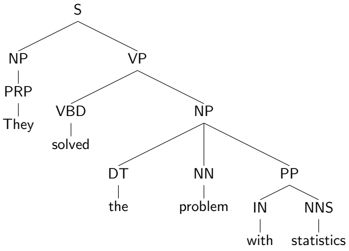
Dependency Trees



- Main element: dependency
- Dependency: a word has a grammatical function with respect to another word

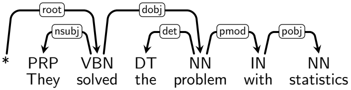
Theories of Syntactic Structure

Constituent Trees



- Main element: constituents
- Constituent: linguistic unit subsuming a word sequence
- Focus on combinations of constituents
- Builds nested trees

Dependency Trees



- Main element: dependency
- Dependency: a word has a grammatical function with respect to another word
- Focus on relations between words
- Builds dependency graphs

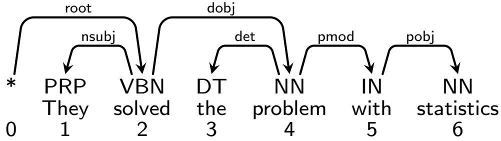
Notation: Dependency

Dependency Trees

Dependency Parsing

Graph-based Dependency Parsing

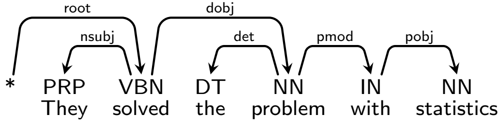
Transition-Based Dependency parsers



- $*$ is a special *root* symbol
- Each dependency is a tuple (h, m, k) where
  - h: index of the head word (root is 0)
  - m: index of the modifier word
  - k: dependency label

  e.g.: $(0, 2, \text{root})$, $(2, 1, \text{nsubj})$, $(2, 4, \text{dobj})$, $(4, 3, \text{det})$, $(4, 5, \text{pmod})$, $(5, 6, \text{pobj})$
- Sometimes we just consider unlabeled dependencies

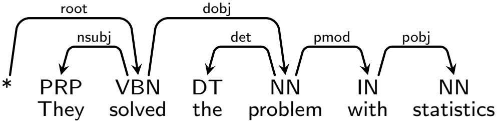
Notation: Dependency Tree



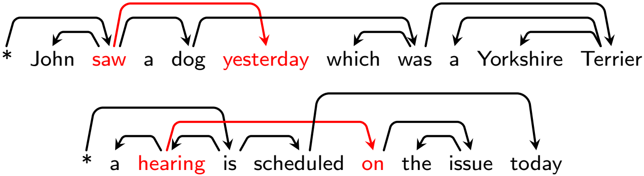
- $\mathbf{y}$ is a dependency tree if:
  - (a) $\mathbf{y}$ is a set of dependencies, $\{(h, m, k)_i\}$
  - (b) Each non-root token has exactly an incoming arc (i.e. one parent)
  - (c) The graph is connected
  - (d) There are no cycles
    - That is, dependency arcs form a directed tree rooted at $*$

Projectivity

- Projective dependency tree: no crossing dependencies

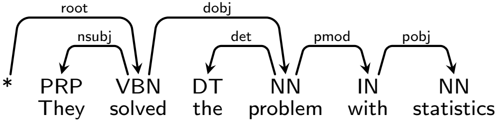


- Non-projective dependency tree: crossing dependencies

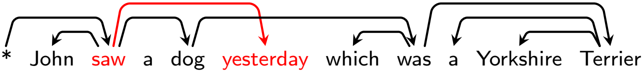

* John saw a dog yesterday which was a Yorkshire Terrier
* a hearing is scheduled on the issue today

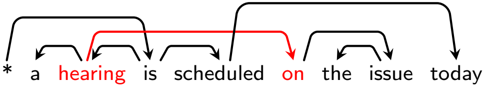
Projectivity

- Projective dependency tree: no crossing dependencies



- Non-projective dependency tree: crossing dependencies





On the contrary of constituent parsing, dependency parsing can manage different word orders, so it can provide both projective and non-projective trees

Outline

1 Dependency Trees

2 Dependency Parsing

3 Graph-based Dependency Parsing
- Algorithm based on Maximum-Spanning Trees

4 Transition-Based Dependency parsers
- Arc-Standard algorithm

Types of Dependency Parsing

- Regarding projectivity:
  - Projective parsing: produces projective dependency trees
  - Non-projective parsing: produces projective or non-projective dependency trees
    (how often occurs in a particular language -or treebank-?)

Types of Dependency Parsing

- Regarding projectivity:
    - Projective parsing: produces projective dependency trees
    - Non-projective parsing: produces projective or non-projective dependency trees
      (how often occurs in a particular language -or treebank-?)
- Regarding the techniques:
    - Graph-based dependency parsing:
        - Algorithms based on CKY
        - Algorithm based on Maximum-Spanning Trees
    - Transition-based dependency parsing:
        - Arc-standard algorithm
    - ...

Dependency Trees

Dependency Parsing

Graph-based Dependency Parsing

Transition-Based Dependency parsers

Outline

1 Dependency Trees

2 Dependency Parsing

3 Graph-based Dependency Parsing
- Algorithm based on Maximum-Spanning Trees

4 Transition-Based Dependency parsers
- Arc-Standard algorithm

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score
- A graph can be split into parts (arcs, sequences of 2 arcs, ...). Then, the score of a graph is the sum of the scores of its parts

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score
- A graph can be split into parts (arcs, sequences of 2 arcs, ...). Then, the score of a graph is the sum of the scores of its parts
- Arc-factored score: (arc-factored parsing)

$$Score(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} score(h,m,k)$$

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score
- A graph can be split into parts (arcs, sequences of 2 arcs, ...). Then, the score of a graph is the sum of the scores of its parts
- Arc-factored score: (arc-factored parsing)

$$Score(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} score(h,m,k)$$

1- How to compute $score(h, m, k)$?

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score
- A graph can be split into parts (arcs, sequences of 2 arcs, ...). Then, the score of a graph is the sum of the scores of its parts
- Arc-factored score: (arc-factored parsing)

$$Score(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} score(h,m,k)$$

1- How to compute $score(h, m, k)$?

2- How to find the highest scored tree?

Graph-based Dependency Parsing

- Goal: given an input sentence, provide the dependency tree with the highest score
- A graph can be split into parts (arcs, sequences of 2 arcs, ...). Then, the score of a graph is the sum of the scores of its parts
- Arc-factored score: (arc-factored parsing)

$$Score(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} score(h,m,k)$$

1- How to compute $score(h, m, k)$?

2- How to find the highest scored tree?

Ex: MST-based algorithm

Compute the Dependency Scores

$$score(h, m, k) = \mathtt{w}\ \mathtt{f}(h, m, k) = \sum_i w_i f_i(h, m, k)$$

where:

- $\{f_i\}$ is a binary feature set to represent any dependency
- w_i is the relevance of f_i given a treebank

Compute the Dependency Scores

$$score(h, m, k) = \mathtt{w}\ \mathtt{f}(h, m, k) = \sum_i w_i f_i(h, m, k)$$

where:

- $\{f_i\}$ is a binary feature set to represent any dependency
- w_i is the relevance of f_i given a treebank

Then,

$$Score(\mathbf{y}) = \mathtt{w}\ \mathtt{f}(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} \mathtt{w}\ \mathtt{f}(h, m, k)$$

where:

- $\mathtt{f}(\mathbf{y})$ is the feature vector of the dependency tree $\mathbf{y}$

Compute the Dependency Scores

$$score(h,m,k) = \mathtt{w}\ \mathtt{f}(h,m,k) = \sum_i w_i f_i(h,m,k)$$

where:

- $\{f_i\}$ is a binary feature set to represent any dependency
- w_i is the relevance of f_i given a treebank

Then,

$$Score(\mathbf{y}) = \mathtt{w}\ \mathtt{f}(\mathbf{y}) = \sum_{(h,m,k)\in\mathbf{y}} \mathtt{w}\ \mathtt{f}(h,m,k)$$

where:

- $\mathtt{f}(\mathbf{y})$ is the feature vector of the dependency tree $\mathbf{y}$

A treebank of sentences with their respective valid dependency parses is required to estimate w_i

Compute the Dependency Scores

Examples of features $f_i(h, m, k)$:

- Words, lemmas, PoS of h or m
- Words, lemmas, PoS of tokens in the context of h or m
- Distance in tokens between h and m
- Dependency k
- Direction of the dependency (right, left)
- Combinations of previous features

Outline

1 Dependency Trees

2 Dependency Parsing

3 Graph-based Dependency Parsing

- Algorithm based on Maximum-Spanning Trees

4 Transition-Based Dependency parsers

- Arc-Standard algorithm

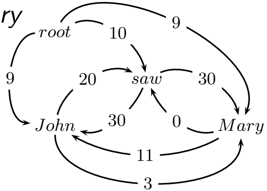
Algorithm based on Maximum-Spanning Trees

1- Build the graph:

- Nodes are tokens (and the root token)
- A weighted directed edge between any two nodes

$$w_{i,j} = \max_{1 \leq k \leq K} score(i, j, k)$$

Ex: *John saw Mary*



Algorithm based on Maximum-Spanning Trees

Dependency Trees

Dependency Parsing

Graph-based Dependency Parsing

Algorithm based on Maximum-Spanning Trees

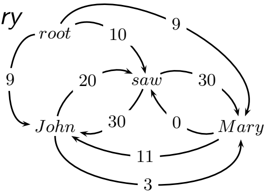
Transition-Based Dependency parsers

1- Build the graph:

- Nodes are tokens (and the root token)
- A weighted directed edge between any two nodes

$$w_{i,j} = \max_{1 \leq k \leq K} score(i, j, k)$$

Ex: *John saw Mary*

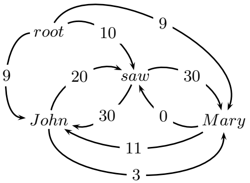


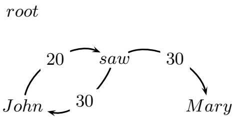
2- Perform non-projective parsing as maximum-spanning trees, using the Chu-Liu-Edmonds algorithm

Cost: $O(n^3)$, improved version $O(n^2)$

Chu-Liu-Edmonds, example

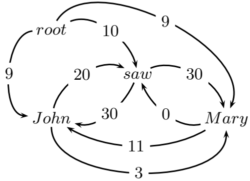
- Step 1: for each node, find highest-scoring incoming edge

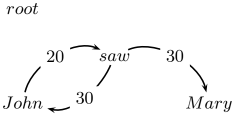




Chu-Liu-Edmonds, example

- Step 1: for each node, find highest-scoring incoming edge

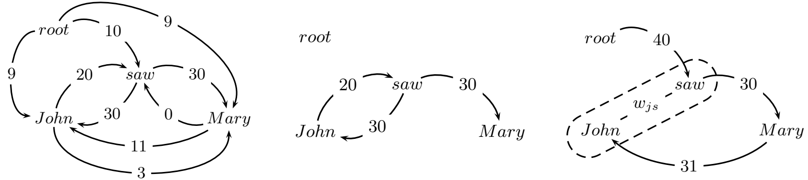




- If we get a tree, STOP. We have found the MST
- If not, there has to be a cycle

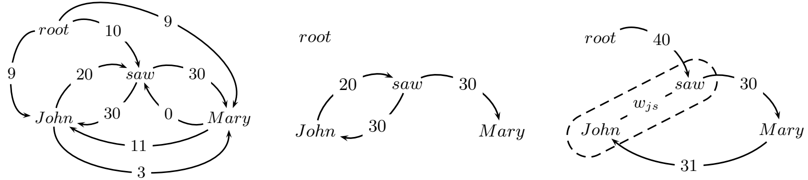
Chu-Liu-Edmonds, example

- Step 2: identify cycle and *contract* it into a new node c



Chu-Liu-Edmonds, example

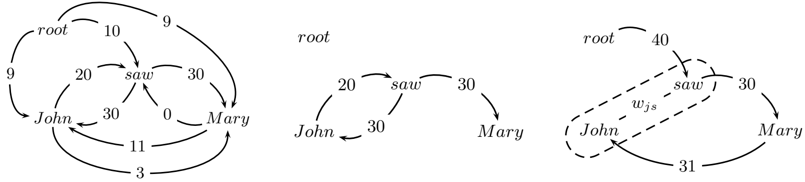
- Step 2: identify cycle and *contract* it into a new node c



- Weight of edges between c and other nodes i:

Chu-Liu-Edmonds, example

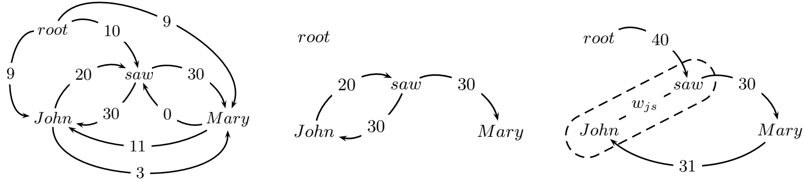
- Step 2: identify cycle and *contract* it into a new node c



- Weight of edges between c and other nodes i:
  - $c \rightarrow i$: max weight of any node in c to i

Chu-Liu-Edmonds, example

- Step 2: identify cycle and *contract* it into a new node c



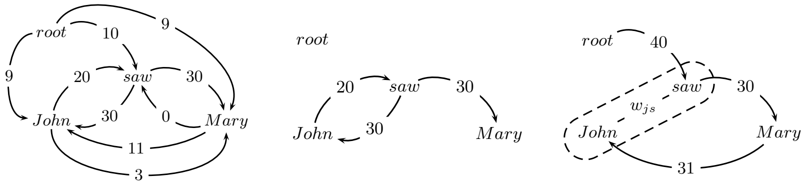
- Weight of edges between c and other nodes i:
  - $c \rightarrow i$: max weight of any node in c to i
  - $i \rightarrow c$: max weight of i that spans c

    root → *saw* → *John* : 40

    root → *John* → *saw* : 29

Chu-Liu-Edmonds, example

- Step 2: identify cycle and *contract* it into a new node c



- Weight of edges between c and other nodes i:
  - $c \rightarrow i$: max weight of any node in c to i
  - $i \rightarrow c$: max weight of i that spans c

    root → *saw* → *John* : 40

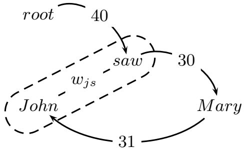
    root → *John* → *saw* : 29

    Mary → *John* → *saw* : 31

    Mary→ *saw* → *John* : 30

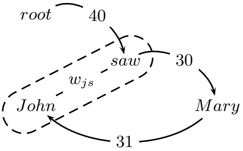
Chu-Liu-Edmonds, example

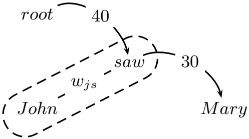
- Step 3: recursively call the algorithm on the new graph



Chu-Liu-Edmonds, example

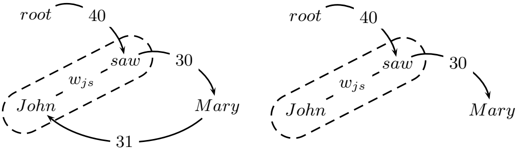
- Step 3: recursively call the algorithm on the new graph
    - Step 1: for each node, find highest-scoring incoming edge





Chu-Liu-Edmonds, example

- Step 3: recursively call the algorithm on the new graph
  - Step 1: for each node, find highest-scoring incoming edge



  - If we get a tree, STOP. We have found the MST (after one recursive call we get a tree)

Chu-Liu-Edmonds, example

Dependency Trees

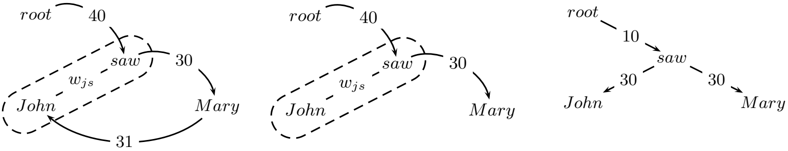
Dependency Parsing

Graph-based Dependency Parsing

Algorithm based on Maximum-Spanning Trees

Transition-Based Dependency parsers

- Step 3: recursively call the algorithm on the new graph
  - Step 1: for each node, find highest-scoring incoming edge



  - If we get a tree, STOP. We have found the MST (after one recursive call we get a tree)

- Step 4: reconstruct the original MST by undoing the contraction operations ($saw \xrightarrow{30} John$) (see (McDonald et al 2005) for details)

Outline

1. Dependency Trees
2. Dependency Parsing
3. Graph-based Dependency Parsing
    - Algorithm based on Maximum-Spanning Trees
4. Transition-Based Dependency parsers
    - Arc-Standard algorithm

Transition-Based parsers

- The parser has a current state or configuration consisting of a stack (of tokens processed and tree built so far) and a buffer (tokens remaining).
- At each step, a transition is chosen to alter the configuration and move (via a classifier).
- Parsing stops when a final configuration is reached
- No backtracking, cost is $\mathcal{O}(n)$

Transition-Based parsers

- The parser has a current state or configuration consisting of a stack (of tokens processed and tree built so far) and a buffer (tokens remaining).
- At each step, a transition is chosen to alter the configuration and move (via a classifier).
- Parsing stops when a final configuration is reached
- No backtracking, cost is $\mathcal{O}(n)$

- Different parsers are defined depending on the set of possible transitions: arc-standard model, arc-eager model, swap-based model, ...

Outline

1 Dependency Trees

2 Dependency Parsing

3 Graph-based Dependency Parsing
- Algorithm based on Maximum-Spanning Trees

4 Transition-Based Dependency parsers
- Arc-Standard algorithm

Arc-Standard algorithm

- A configuration (S, B, A) of the parser consists of:
    - A stack S containing seen words
    - A buffer B containing not-yet seen words
    - The dependency graph A built so far (not a tree yet)
- Initial configuration: $([\ ], [0 \dots n], [\ ])$
- Final configuration: $([0], [\ ], A)$
- Possible transitions:
    - shift: push next word in the buffer onto the stack
    - left-arc: add an arc from $S[0]$ to $S[1]$ and remove $S[1]$ from the stack
    - right-arc: add an arc from $S[1]$ to $S[0]$ and remove $S[0]$ from the stack

Arc-Standard Transition definitions

- shift (sh)
  $(\sigma, [i|\beta], A) \Rightarrow ([\sigma|i], \beta, A)$
- left-arc (la-L)
  $([\sigma|i|j], B, A) \Rightarrow ([\sigma|j], B, A \cup \{j, i, L\})$
- right-arc (ra-L):
  $([\sigma|i|j], B, A) \Rightarrow ([\sigma|i], B, A \cup \{i, j, L\})$

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	

* the woman saw the man with glasses

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh

* the woman saw the man with glasses

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	

* the woman saw the man with glasses

Dependency Trees

Dependency Parsing

Graph-based Dependency Parsing

Transition-Based Dependency parsers

Arc-Standard algorithm

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh

* the woman saw the man with glasses

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	

* the woman saw the man with glasses

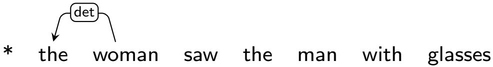
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det

* the woman saw the man with glasses

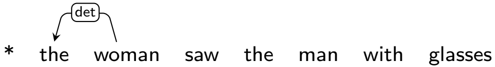
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	



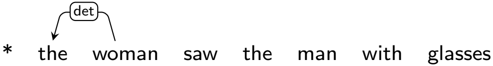
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh



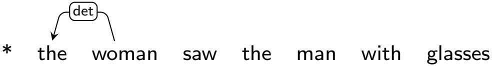
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	



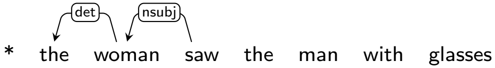
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj



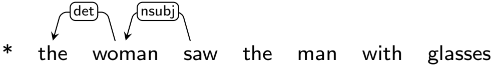
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	



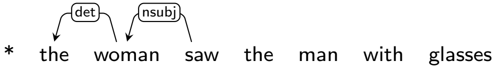
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh



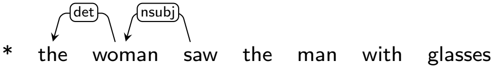
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	



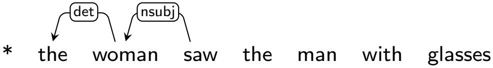
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh



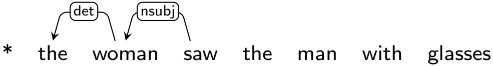
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	



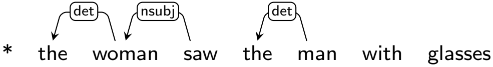
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det



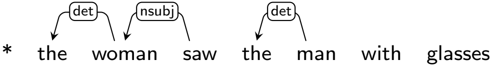
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	



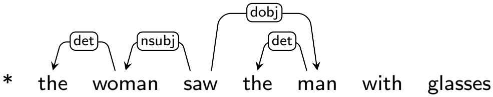
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj



Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	



Dependency Trees

Dependency Parsing

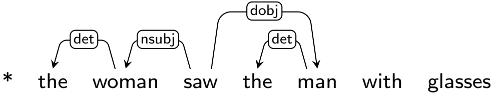
Graph-based Dependency Parsing

Transition-Based Dependency parsers

Arc-Standard algorithm

Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh



Dependency Trees

Dependency Parsing

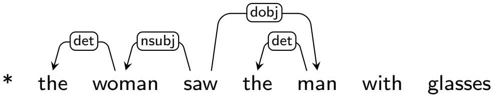
Graph-based Dependency Parsing

Transition-Based Dependency parsers

Arc-Standard algorithm

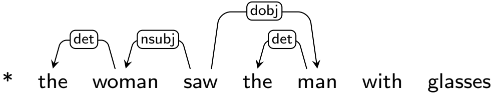
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	



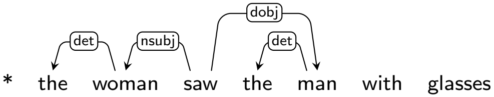
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh



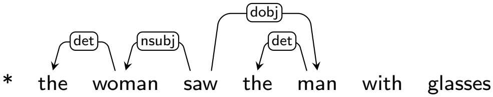
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		



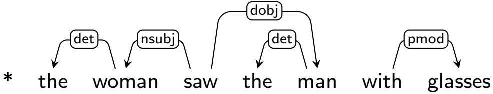
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod



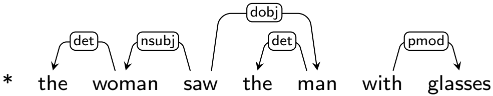
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		



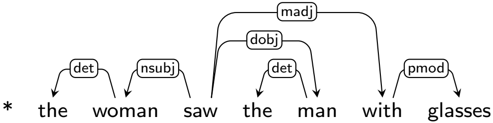
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		ra-madj



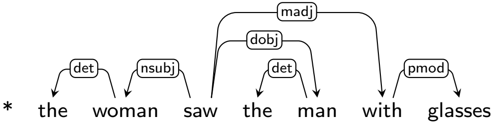
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		ra-madj
* saw		



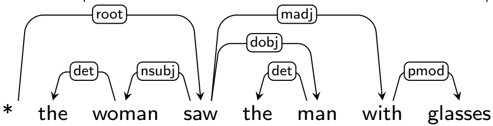
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		ra-madj
* saw		ra-root



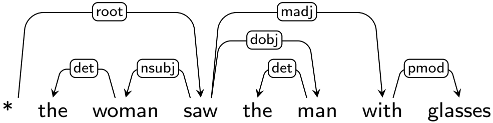
Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		ra-madj
* saw		ra-root
*		



Arc-Standard Example

Stack	Buffer	Transition
	* the woman saw the man with glasses	sh
* the	woman saw the man with glasses	sh
* the woman	saw the man with glasses	la-det
* woman	saw the man with glasses	sh
* woman saw	the man with glasses	la-subj
* saw	the man with glasses	sh
* saw the	man with glasses	sh
* saw the man	with glasses	la-det
* saw man	with glasses	ra-dobj
* saw	with glasses	sh
* saw with	glasses	sh
* saw with glasses		ra-pmod
* saw with		ra-madj
* saw		ra-root
*		stop



Transition Selection

- On the contrary to graph-based parsers, only one tree is produced. How to handle ambiguity?
    - Add probabilities to select which transition to apply at each step
        - Similar to CKY with PCFGs, but greedy search
        - May be made less greedy with e.g. beam-search
    - Use ML to learn a model for taking the decision
- Given that we apply local search, we can achieve a valid projective parse, but can be suboptimal.

Transition Selection

- Classifier: predicts the next transition (class) given the current configuration
- Learn the classification model from <configuration, transition> pairs annotated by hand in a treebank.
- Need to model the configurations as feature vectors and use ML.
- Typical features:
  - word/lemma/PoS for $S[0]$, $S[1]$, $B[0]$, $B[1]$
  - morphological features (gender, number, mode, tense, etc) in $S[0]$, $B[0]$
  - number of children of $S[0]$
  - dependency labels of $S[0]$ children
  - ..etc
- We can use SVM, perceptron, MBL, DT, ... any feature-based ML classifier, or deep learning as well

Variants of Transition-based Parsing

- Stack-stack arcs
  - Arc-standard (shift, left-arc, right-arc)
  - Non-projective (shift, swap, left-arc, right-arc)
- Stack-buffer arcs
  - Arc-eager (shift, reduce, left-arc, right-arc)
  - Arc-standard variant (shift, left-arc, right-arc)