#### Master in Artificial Intelligence

Trees and Grammars

Constituency Parsing

Dependency Parsing

### Advanced Human Language Technologies



UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Facultat d'Informàtica de Barcelona



#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

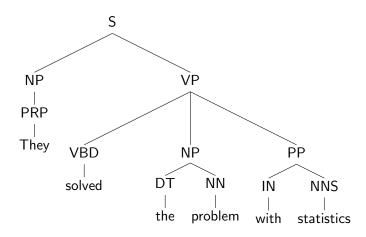
# Trees and Grammars

Constituency Parsing

### A Syntactic Tree

Trees and Grammars

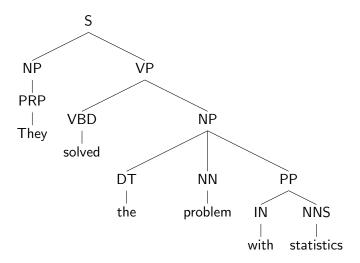
Constituency Parsing



### Another Syntactic Tree

Trees and Grammars

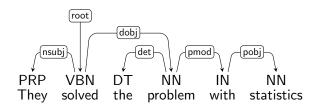
Constituency Parsing



### **Dependency Trees**

Trees and Grammars

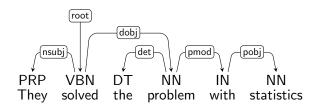
Constituency Parsing

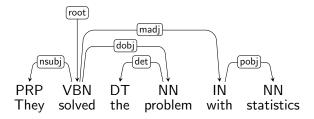


### **Dependency Trees**

Trees and Grammars

Constituency Parsing



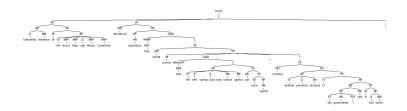


#### A "real" sentence

Trees and Grammars

Constituency Parsing

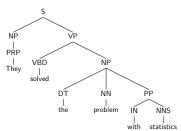
Dependency Parsing



Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts.

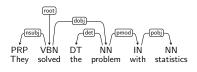
#### Theories of Syntactic Structure

#### Constituent Trees



- Main element: constituents (or phrases, or bracketings)
- Constituents = abstract linguistic units
- Results in nested trees

#### **Dependency Trees**



- Main element: dependency
- Focus on relations between words
- Handles free word order nicely.

## Trees and Grammars

Constituency Parsing

## Context Free Grammars (CFGs)

Trees and Grammars

Constituency Parsing

Dependency Parsing A context-free grammar is defined as a tuple  $G = \langle N, \Sigma, R, S \rangle$  where:

- lacksquare N is a set of non-terminal symbols
- ullet  $S \in N$  is a distinguished start symbol
- lacksquare  $\Sigma$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$

#### Context Free Grammars, Example

Trees and Grammars

Constituency Parsing

<sup>&</sup>lt;sup>1</sup>S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

#### Properties of CFGs

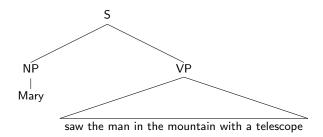
Trees and Grammars

Constituency Parsing

- A CFG defines a set of possible derivations (i.e. unique trees)
- A sequence of terminals  $s \in \Sigma^*$  is *generated* by the CFG (or *recognized* by it, or *belongs* to the language defined by it) if there is at least a derivation that produces s.
- Some sequences of terminals generated by the CFG may have more than one derivation (ambiguity).

Trees and Grammars

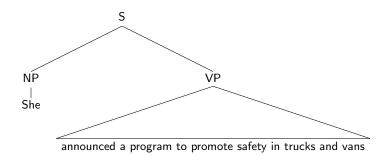
Constituency Parsing



- Mary used a telescope to see a man who was in the mountain
- Mary saw a man who was in the mountain and carried a telescope
- Mary was in the mountain and used a telescope to see a man
- Mary was in the mountain that has a telescope and saw a man
- Mary saw a man who was in the mountain that has a telescope
- Mary was in the mountain and saw a man carrying a telescope

Trees and Grammars

Constituency Parsing



- She announced a program aimed to make trucks and vans safer
- She used trucks and vans to announce a program aimed to promote safety
- She announced a program aimed to make trucks safer. She also announced vans
- She used trucks to announce a program aimed to promote safety.
   She also announced vans
- She announced a program. She did so in order to promote satefy in trucks and vans

Trees and Grammars

Constituency Parsing

Dependency Parsing Some trees are more likely than others...

Trees and Grammars

Constituency Parsing

Dependency Parsing Some trees are more likely than others...

Can we model that?

#### Context Free Grammar (CFGs)

A context-free grammar is defined as a tuple  $G = \langle N, \Sigma, R, S \rangle$  where:

- lacksquare N is a set of non-terminal symbols
- $lue{S} \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols

Trees and

Parsing

Dependency

Parsing

Constituency

■ R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$ 

## Context Free Grammar ( CFGs)

A context-free grammar is defined as a tuple  $G = \langle N, \Sigma, R, S \quad \rangle$  where:

- lacksquare N is a set of non-terminal symbols
- $lacksquare S \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$

Trees and Grammars

Constituency Parsing

A context-free grammar is defined as a tuple  $G = \langle N, \Sigma, R, S \quad \rangle$  where:

- lacksquare N is a set of non-terminal symbols
- ullet  $S \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$

Trees and Grammars

Constituency Parsing

A probabilistic context-free grammar is defined as a tuple  $G=\langle N, \Sigma, R, S \quad \rangle$  where:

- lacksquare N is a set of non-terminal symbols
- ullet  $S \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$

Trees and Grammars

Constituency Parsing

A probabilistic context-free grammar is defined as a tuple  $G=\langle N,\Sigma,R,S,q\rangle$  where:

- lacksquare N is a set of non-terminal symbols
- ullet  $S \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$

Trees and Grammars

Constituency Parsing

A probabilistic context-free grammar is defined as a tuple  $G=\langle N,\Sigma,R,S,q\rangle$  where:

- lacksquare N is a set of non-terminal symbols
- $lacksquare S \in N$  is a distinguished start symbol
- $lue{\Sigma}$  is a set of terminal symbols
- R is a set of rules of the form  $X \to Y_1 Y_2 \dots Y_n$  where  $n \ge 0, \ X \in N, \ Y_i \in N \cup \Sigma$
- lack q is a set of non-negative parameters, one for each rule  $X o \alpha \in R$  such that, for any  $X \in N$ ,

$$\sum_{(X \to \alpha) \in R} q(X \to \alpha) = 1$$

Trees and Grammars

Constituency Parsing

#### Context Free Grammars, Example

Trees and Grammars

Constituency Parsing

$$\begin{array}{lll} N & = & \{ \mathrm{S}, \mathrm{VP}, \mathrm{NP}, \mathrm{PP}, \mathrm{DT}, \mathrm{Vi}, \mathrm{Vt}, \mathrm{NN}, \mathrm{IN} \}^1 \\ S & = & \{ \mathrm{S} \} \\ \Sigma & = & \{ \mathrm{sleeps}, \mathrm{saw}, \mathrm{man}, \mathrm{woman}, \mathrm{telescope}, \mathrm{the}, \mathrm{with}, \mathrm{in} \} \\ & & \left\{ \begin{array}{lll} \mathrm{S} & \to \mathrm{NP} \; \mathrm{VP} & \mathrm{Vi} & \to \mathrm{sleeps} \\ \mathrm{NP} & \to \mathrm{DT} \; \mathrm{NN} & \mathrm{Vt} & \to \mathrm{saw} \\ \mathrm{NP} & \to \mathrm{DT} \; \mathrm{NN} & \mathrm{Vt} & \to \mathrm{saw} \\ \mathrm{NP} & \to \mathrm{NP} \; \mathrm{PP} & \mathrm{NN} & \to \mathrm{man} \\ \mathrm{PP} & \to \mathrm{IN} \; \mathrm{NP} & \mathrm{NN} & \to \mathrm{woman} \\ \mathrm{VP} & \to \mathrm{Vi} & \mathrm{NN} & \to \mathrm{telescope} \\ \mathrm{VP} & \to \mathrm{Vt} \; \mathrm{NP} & \mathrm{DT} & \to \mathrm{the} \\ \mathrm{VP} & \to \mathrm{VP} \; \mathrm{PP} & \mathrm{IN} & \to \mathrm{with} \\ \mathrm{IN} & \to \mathrm{in} \end{array} \right)$$

<sup>&</sup>lt;sup>1</sup>S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

#### Probabilistic Context Free Grammars, Example

Trees and Grammars

Constituency Parsing

```
N = \{S, VP, NP, PP, DT, Vi, Vt, NN, IN\}^{1}
   = {sleeps, saw, man, woman, telescope, the, with, in}
                                    1.0
                                              Vi \rightarrow sleeps
                                                                       1.0
              NP \rightarrow DT NN
                                    0.4 	ext{ Vt} \rightarrow \text{saw}
                                                                       1.0
             NP \rightarrow NP PP
                                    0.6
                                               NN \rightarrow man
                                                                       0.7
            PP \rightarrow IN NP
                                    1.0
                                               NN \rightarrow woman
                                                                       0.2
             VP \rightarrow Vi
                                    0.5
                                               NN \rightarrow telescope
                                                                       0.1
              VP \rightarrow Vt NP
                                    0.4
                                               DT \rightarrow the
                                                                       1.0
                                    0.1
                                               IN \rightarrow with
                                                                       0.5
                                               IN \rightarrow in
```

<sup>&</sup>lt;sup>1</sup>S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

### Properties of PCFGs

■ The probability of a parse tree  $t \in \mathcal{T}_G$  is computed as:

$$p(t) = \prod_{r \in t} q(r)$$

- If there is more than one tree for a sentence, we can rank them by probability.
- $\blacksquare$  The most likely tree for a sentence s is:

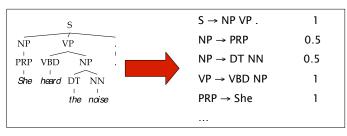
$$\arg\max_{t\in\mathcal{T}(s)}p(t)$$

Trees and Grammars

Constituency Parsing

## Learning Treebank Grammars

■ Read the grammar rules from a treebank



Constituency Parsing

Trees and

Dependency Parsing

■ Set rule weights by maximum likelihood

$$q(\alpha \to \beta) = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

- Smoothing issues apply
- Having the appropriate CFG is critical to success

#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

## Trees and Grammars

Constituency Parsing

#### Goal of a parser:

■ Find all possible trees

Grammars
Constituency
Parsing

Trees and

Dependency

#### Goal of a parser:

- Find all possible trees
- Find all possible trees, ranked by probability

Trees and Grammars Constituency

Parsing
Dependency
Parsing

## Goal of a parser:

- Find all possible trees
- Find all possible trees, ranked by probability
- Find most likely tree

Trees and Grammars

Constituency Parsing

#### Trees and Grammars

Constituency Parsing

Dependency Parsing

#### Goal of a parser:

- Find all possible trees
- Find all possible trees, ranked by probability
- Find most likely tree
- Many of the possible trees will share subtrees that we don't need to re-parse.

### Trees and

Constituency Parsing

Dependency Parsing

#### Goal of a parser:

- Find all possible trees
- Find all possible trees, ranked by probability
- Find most likely tree
- Many of the possible trees will share subtrees that we don't need to re-parse.
- Define a dynammic programming table (aka chart) to store intermediate results.

#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

# Trees and Grammars

Constituency Parsing CKY Algorithm

#### **CKY Algorithm**

Trees and Grammars

Constituency Parsing CKY Algorithm

CKY Algorithm

- Bottom-up
- Requires a grammar in Chomsky Normal Form (CNF).
- Dynammic programming: Store partial results that can be reused in different candidate solutions.
- Analogous to Viterbi in HMMs.
- Intermediate results stored in a chart structure.

## **CKY Algorithm**

#### Chart content:

Maximum probability of a subtree with root X spanning words i...j:

$$\pi(i,j,X)$$

Backpath to recover which rules produced the maximum probability tree:

$$\psi(i, j, X)$$

The goal is to compute:

- $\psi(1,n,S)$
- It is possible to use it without probabilities to get all parse trees (with higher complexity)

Trees and Grammars

Constituency Parsing CKY Algorithm

## **CKY Algorithm**

Base case: Tree leaves

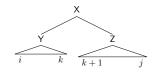
$$\forall i = 1 \dots n, \ \forall X \to w_i \in R, \ \pi(i, i, X) = q(X \to w_i)$$

Recursive case: Non-terminal nodes

$$\forall i = 1 \dots n, \ \forall j = (i+1) \dots n, \ \forall X \in N$$

$$\pi(i, j, X) = \max_{\substack{X \to YZ \in R \\ k: j < k < i}} q(X \to YZ) \times \pi(i, k, Y) \times \pi(k+1, j, Z)$$

$$\psi(i, j, X) = \arg \max_{\substack{X \to YZ \in R \\ k: i \leq k \leq j}} q(X \to YZ) \times \pi(i, k, Y) \times \pi(k+1, j, Z)$$



Output:

lacktriangle Return  $\pi(1,n,S)$  and recover backpath trough  $\psi(1,n,S)$ 

Trees and Grammars

Constituency Parsing CKY Algorithm

### CKY Algorithm - Example

Trees and Grammars

Constituency Parsing CKY Algorithm

```
N = \{S, VP, NP, PP, DT, Vi, Vt, NN, IN\}^{1}
    = {sleeps, saw, man, woman, telescope, the, with, in}
                                    1.0
                                              Vi \rightarrow sleeps
                                                                       1.0
              NP \rightarrow DT NN
                                    0.4 	ext{Vt} \rightarrow \text{saw}
                                                                       1.0
             NP \rightarrow NP PP
                                    0.6
                                               NN \rightarrow man
                                                                       0.7
            PP \rightarrow IN NP
                                    1.0
                                               NN \rightarrow woman
                                                                       0.2
              VP \rightarrow Vi
                                    0.5
                                               NN \rightarrow telescope
                                                                       0.1
              VP \rightarrow Vt NP
                                    0.4
                                               DT \rightarrow the
                                                                       1.0
                                    0.1
                                               IN \rightarrow with
                                                                       0.5
                                               IN \rightarrow in
```

 $<sup>^1</sup>$ S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

## CKY Algorithm - Example - CNF

Trees and Grammars

Constituency Parsing CKY Algorithm

```
N = \{S, VP, NP, PP, DT, Vi, Vt, NN, IN\}^{1}
    = {sleeps, saw, man, woman, telescope, the, with, in}
                                    1.0
                                             Vi \rightarrow sleeps
                                                                       1.0
              NP \rightarrow DT NN
                                    0.4 	ext{Vt} \rightarrow \text{saw}
                                                                       1.0
             NP \rightarrow NP PP
                                    0.6
                                              NN \rightarrow man
                                                                       0.7
            PP \rightarrow IN NP
                                    1.0
                                              NN \rightarrow woman
                                                                       0.2
             VP \rightarrow Vi
                                    0.5
                                              NN \rightarrow telescope
                                                                       0.1
              VP \rightarrow Vt NP
                                    0.4
                                              DT \rightarrow the
                                                                       1.0
                                    0.1
                                              IN \rightarrow with
                                                                       0.5
                                               IN \rightarrow in
```

<sup>&</sup>lt;sup>1</sup>S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

## CKY Algorithm - Example - CNF

Trees and Grammars

Constituency Parsing CKY Algorithm

```
N = \{S, VP, NP, PP, DT, Vi, Vt, NN, IN\}^{1}
          {sleeps, saw, man, woman, telescope, the, with, in}
               S \rightarrow NP VP
                                    0.5
                                              Vi \rightarrow sleeps
                                                                       1.0
              S \rightarrow NP Vi
                                    0.5 	 Vt \rightarrow saw
                                                                       1.0
              NP \rightarrow DT NN
                                    0.4
                                               NN \rightarrow man
                                                                       0.7
            NP \rightarrow NP PP
                                    0.6
                                               NN \rightarrow woman
                                                                       0.2
              PP \rightarrow IN NP
                                     1.0
                                               NN \rightarrow telescope
                                                                       0.1
              VP \rightarrow Vt NP
                                    0.4
                                               DT \rightarrow the
                                                                       1.0
                                    0.1
                                               IN \rightarrow with
                                                                       0.5
                                     0.5
                                               IN \rightarrow in
```

<sup>&</sup>lt;sup>1</sup>S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Trees and Grammars

Constituency Parsing CKY Algorithm

DT 1.0	NN 0.2	Vt 1 0	DT 1.0	NN 0.7	IN 0.5	DT 1.0	NN 0 1
D1 1.0	1414 0.2	VI 1.0	D1 1.0	1414 0.1	114 0.5	011.0	1414 0.1
	WOITIAIT	saw	the	man	with	the	reiescope
11	22	33	44	55	66	77	88

Grammars
Constituency
Parsing

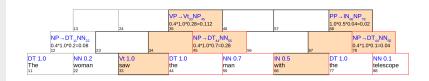
Trees and

Parsing CKY Algorithm

NP → DT 0.4*1.0*0 12		34		NP → DT <sub>44</sub> N 0.4*1.0*0.7=0		56		67		NP → D' 0.4*1.0*0	T <sub>44</sub> NN <sub>55</sub> 0.1=0.04
DT 1.0	NN 0.2	Vt 1.0	DT 1.0		NN 0.7		IN 0.5		DT 1.0		NN 0.1
The	woman	saw 33	the		man 55		with 66		the		telescope

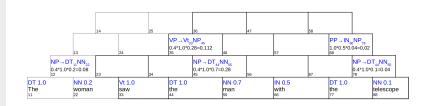
Trees and Grammars Constituency

Parsing CKY Algorithm



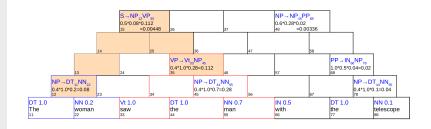
Trees and Grammars Constituency

Parsing CKY Algorithm



Trees and Grammars

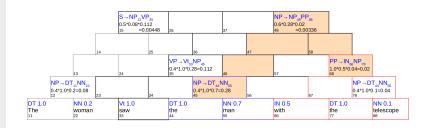
Constituency Parsing CKY Algorithm



Trees and Grammars

Constituency Parsing

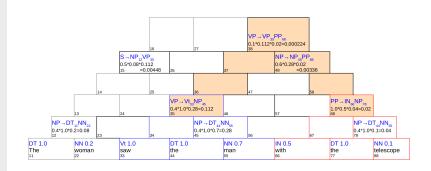
CKY Algorithm



Trees and Grammars

Constituency Parsing CKY Algorithm

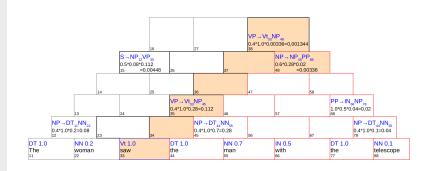
CKT Algorithm



Trees and Grammars

Constituency Parsing CKY Algorithm

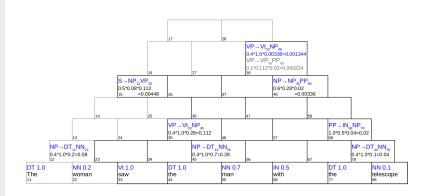
CKY Algorithm



Trees and Grammars

Constituency Parsing CKY Algorithm

CKY Algorithm

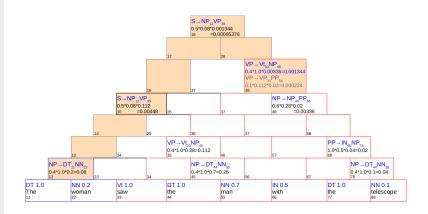


Trees and Grammars

Constituency Parsing CKY Algorithm

Dependency

Parsing



#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

# Trees and Grammars

Constituency Parsing Earley Algorithm

Trees and Grammars

Constituency Parsing Earley Algorithm

- Top-down
- Can deal with any CFG (even left-recursive)
- Dynammic programming: Store partial results that can be reused in different candidate solutions.
- Intermediate results stored in a chart structure.

#### Chart content:

■ Set of items (aka *states*), each describing the applicability status of each rule after each word:

$$[i,j,X\to\alpha\bullet\beta]$$

Backpath to recover which rules produced the complete tree:

$$\psi(i,j,X)$$

The goal is:

- Find if it is possible to reach  $[1, n, S \rightarrow \alpha \bullet]$
- lacktriangle Recover  $\psi(0,n,S)$  if it is
- Probabilistic versions exist, though not as straightforward as in CKY

Trees and Grammars

Constituency Parsing Earley Algorithm

#### Parsing state examples:

Trees and Grammars

Constituency Parsing Earley Algorithm

Dependency Parsing  $[0,0,S\to \bullet \ \mathrm{NP} \ \mathrm{VP}] \qquad \text{A $\mathrm{NP}$ is expected at the beginning} \\ \text{of the sentence}$ 

 $[1,2,NP \to DT \bullet NN] \hspace{0.5cm} \text{A NP has been partially matched} \\ \hspace{0.5cm} (DT \text{ was found between} \\ \hspace{0.5cm} \text{positions 1 and 2)}$ 

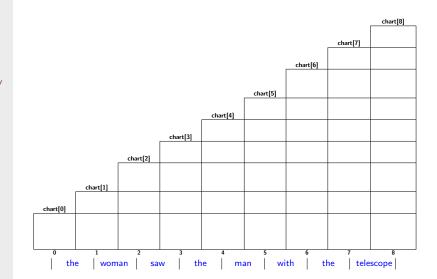
 $[0,3,\mathrm{VP} \to \mathrm{V} \ \mathrm{NP} \bullet]$  A VP has been completed between positions 0 and 3

```
def Earley(words,grammar):
                       chart = [ [ ] for i in range(len(words)+1) ]
                       chart[0].append([0,0,\gamma \rightarrow \bullet S])
                       for i in range(len(words)+1):
Trees and
                             for state in chart[i]:
Grammars
                                  if state.complete() : Complete(state)
Constituency
Parsing
                                  elif is_PoS(state.next()) : Scan(state)
Earley Algorithm
                                  else : Predict(state)
Dependency
Parsing
                       return chart
                  def Scan([i,j,A \rightarrow \alpha \bullet B\beta]):
                       if B in words[j].PoS(): chart[j+1].append([j,j+1,B\rightarrowword[j]\bullet])
                  def Predict([i,i,A \rightarrow \alpha \bullet B\beta]):
                       for B \to \gamma in grammar : chart[i].append([i,i,B \to \bullet \gamma])
                  def Complete([k,j,B \rightarrow \gamma \bullet]):
                       for [i,k,A \to \alpha \bullet B\beta] in chart[k]: chart[i].append([i,j,A \to \alpha B \bullet \beta])
```

Trees and Grammars Constituency

Parsing

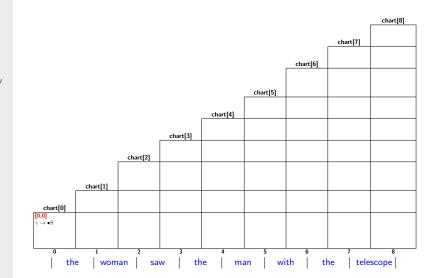
Earley Algorithm



Trees and Grammars

Constituency Parsing

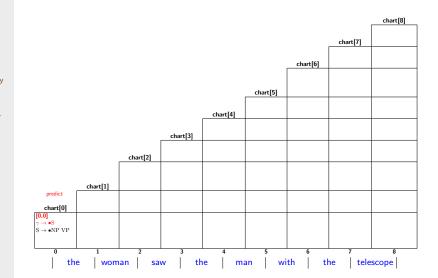
Earley Algorithm



Trees and Grammars

Constituency Parsing

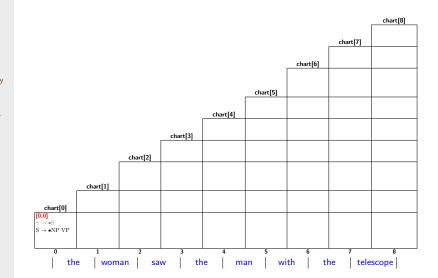
Earley Algorithm



Trees and Grammars

Constituency Parsing

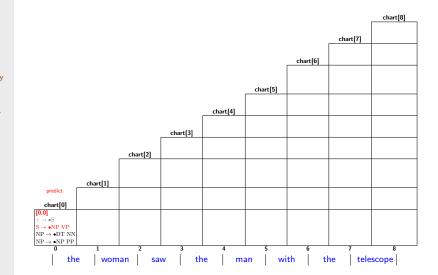
Earley Algorithm



Trees and Grammars

Constituency Parsing

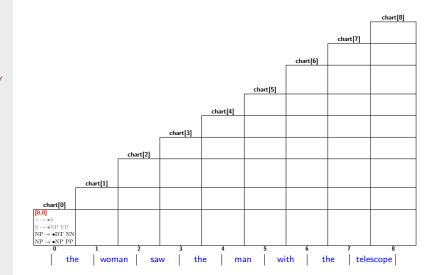
Earley Algorithm



Trees and Grammars

Constituency Parsing

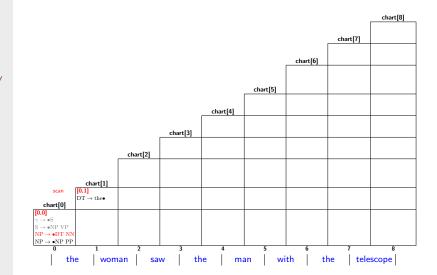
Earley Algorithm



Trees and

Constituency Parsing

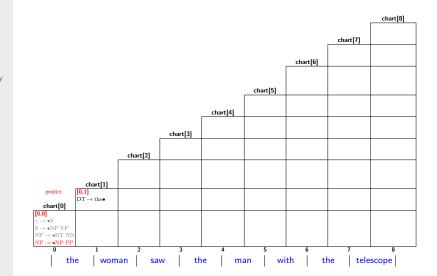
Earley Algorithm



Trees and

Constituency Parsing

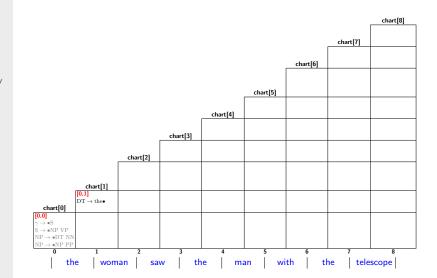
Earley Algorithm



Trees and Grammars

Constituency Parsing

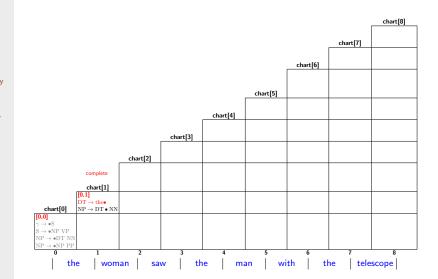
Earley Algorithm



Trees and

Constituency Parsing

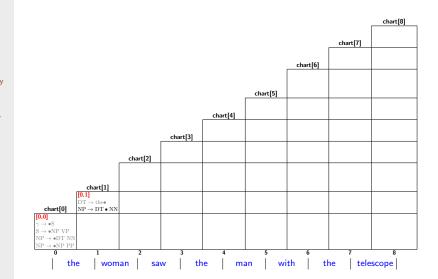
Earley Algorithm



Trees and Grammars

Constituency Parsing

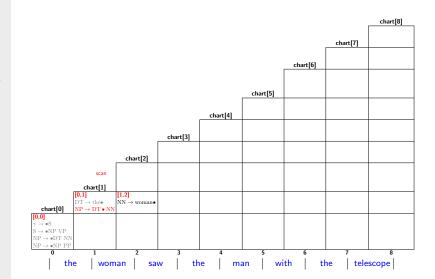
Earley Algorithm



Trees and

Constituency Parsing

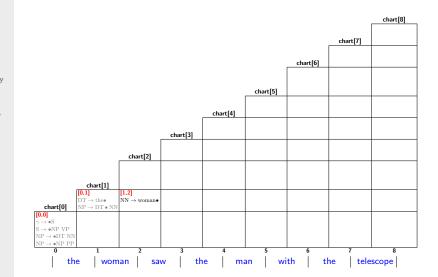
Earley Algorithm



Trees and

Constituency Parsing

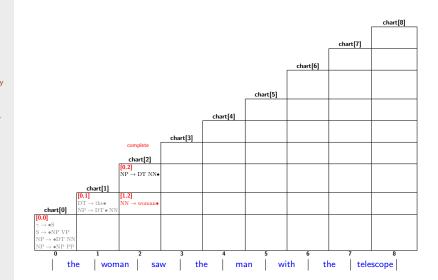
Earley Algorithm



Trees and

Constituency Parsing

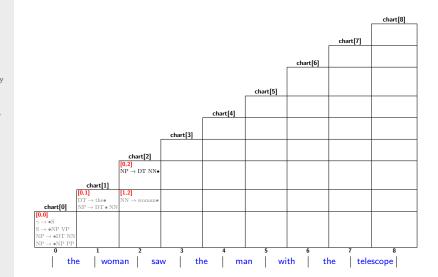
Earley Algorithm



Trees and Grammars

Constituency Parsing

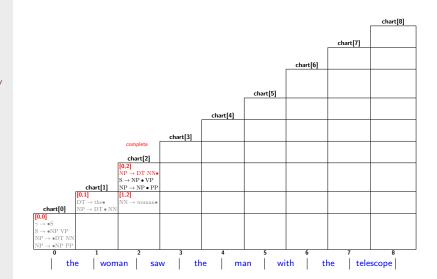
Earley Algorithm



Trees and

Constituency Parsing

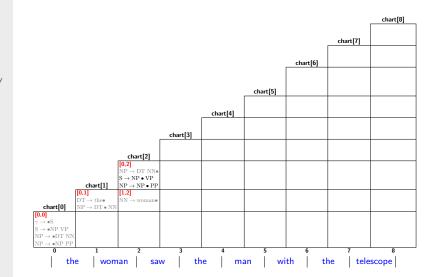
Earley Algorithm



Trees and

Constituency Parsing

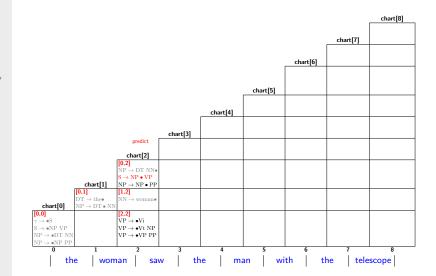
Earley Algorithm



Trees and

Constituency Parsing

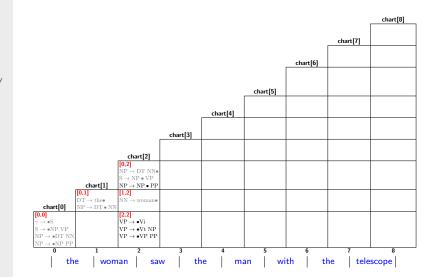
Earley Algorithm



Trees and

Constituency Parsing

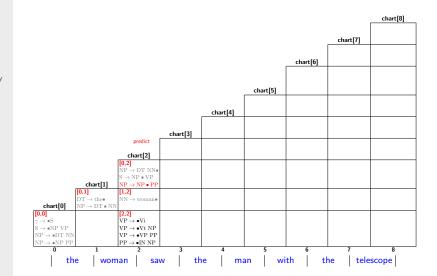
Earley Algorithm



Trees and Grammars

Constituency Parsing

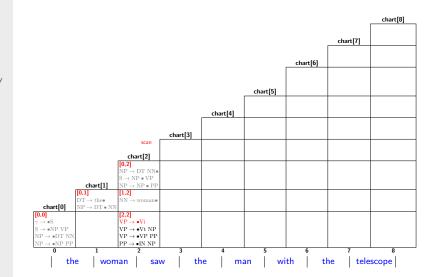
Earley Algorithm



Trees and

Constituency Parsing

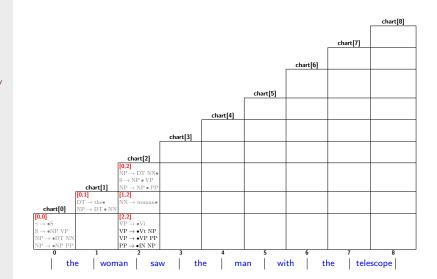
Earley Algorithm



Trees and

Constituency Parsing

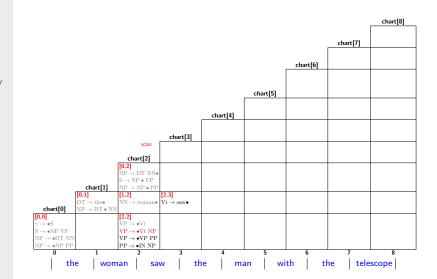
Earley Algorithm



Trees and Grammars

Constituency Parsing

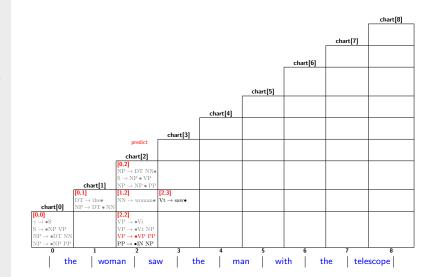
Earley Algorithm



Trees and Grammars

Constituency Parsing

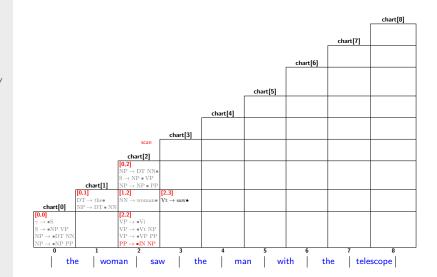
Earley Algorithm



Trees and Grammars

Constituency Parsing

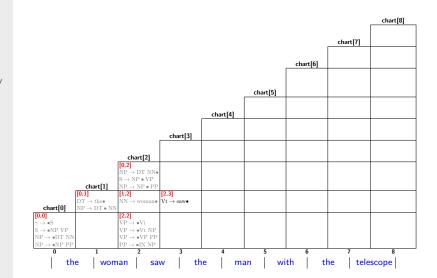
Earley Algorithm



Trees and Grammars

Constituency Parsing

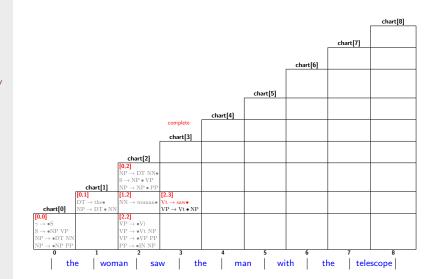
Earley Algorithm



Trees and Grammars

Constituency Parsing

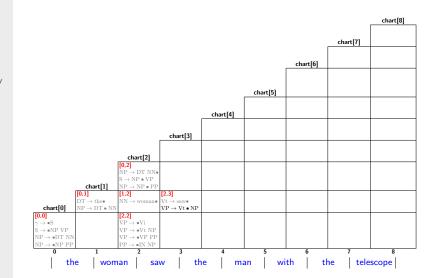
Earley Algorithm



Trees and Grammars

Constituency Parsing

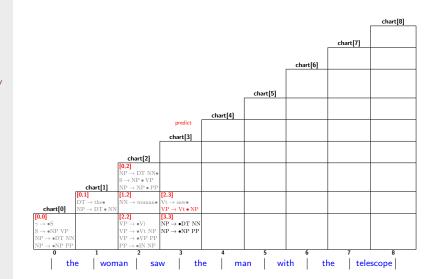
Earley Algorithm



Trees and

Constituency Parsing

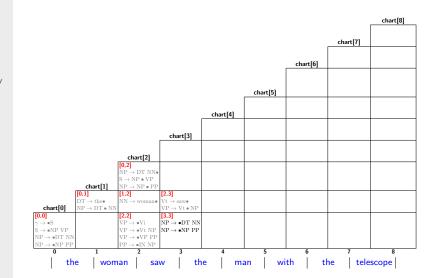
Earley Algorithm



Trees and Grammars

Constituency Parsing

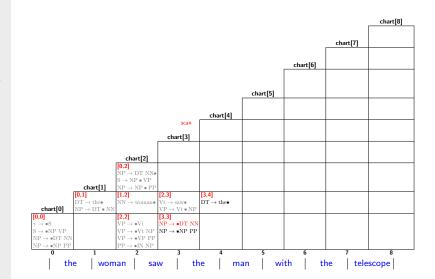
Earley Algorithm



Trees and Grammars

Constituency Parsing

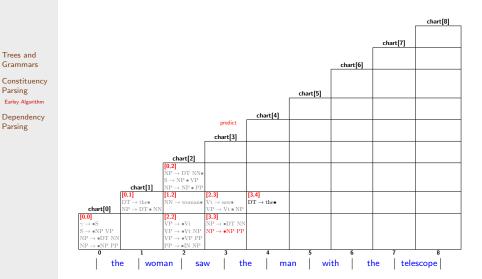
Earley Algorithm



Trees and Grammars

Parsing

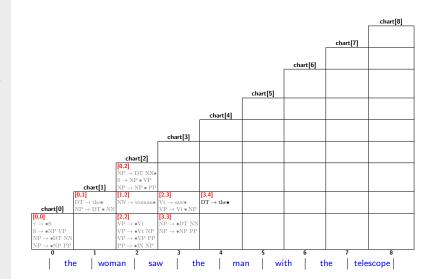
Earley Algorithm



Trees and Grammars

Constituency Parsing

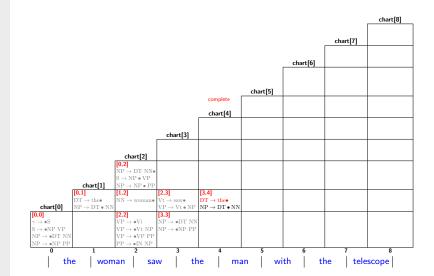
Earley Algorithm



Trees and

Constituency Parsing

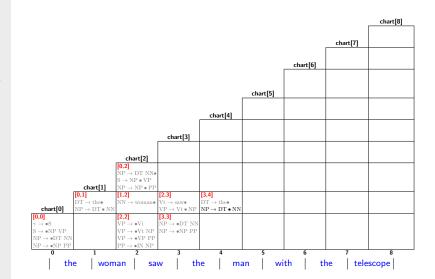
Earley Algorithm



Trees and Grammars

Constituency Parsing

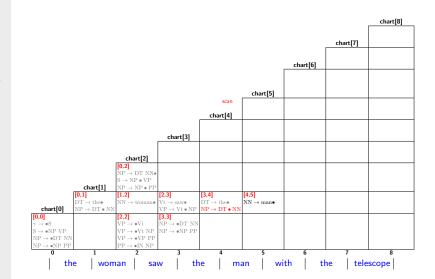
Earley Algorithm



Trees and Grammars

Constituency Parsing

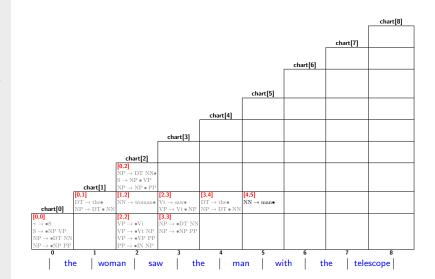
Earley Algorithm



Trees and Grammars

Constituency Parsing

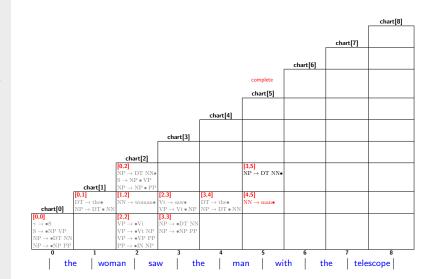
Earley Algorithm



Trees and

Constituency Parsing

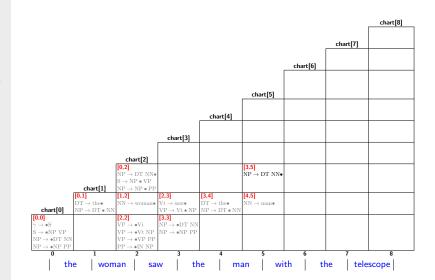
Earley Algorithm



Trees and

Constituency Parsing

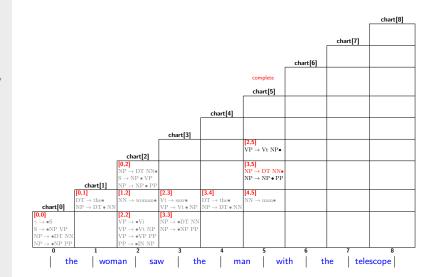
Earley Algorithm



Trees and

Constituency Parsing

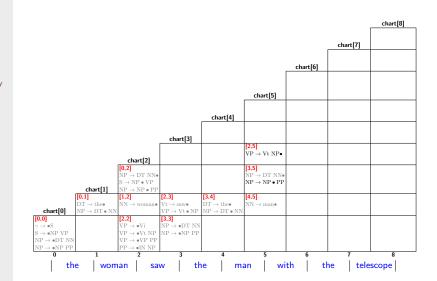
Earley Algorithm



Trees and Grammars

Constituency Parsing

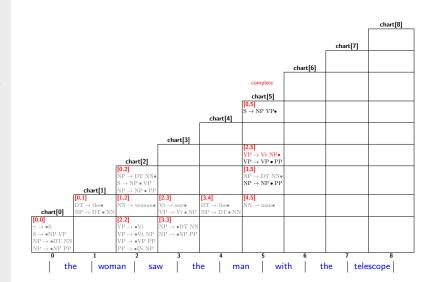
Earley Algorithm



Trees and Grammars

Constituency Parsing

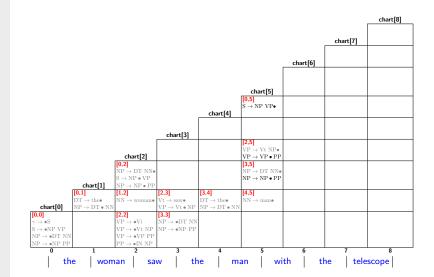
Earley Algorithm



Trees and

Constituency Parsing

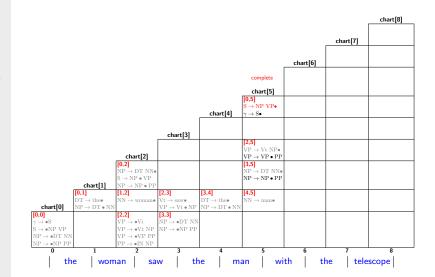
Earley Algorithm



Trees and

Constituency Parsing

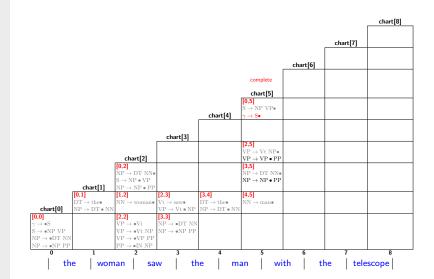
Earley Algorithm



Trees and Grammars

Constituency Parsing

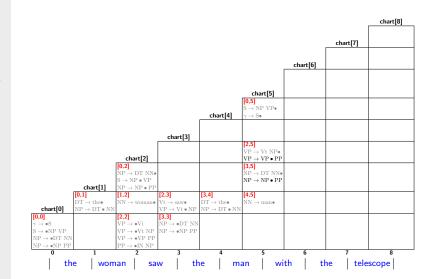
Earley Algorithm



Trees and Grammars

Constituency Parsing

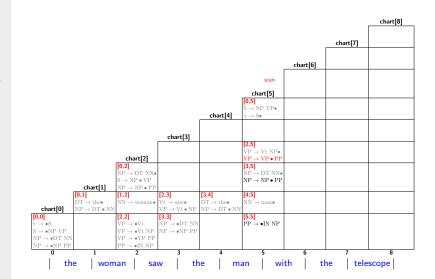
Earley Algorithm



Trees and

Constituency Parsing

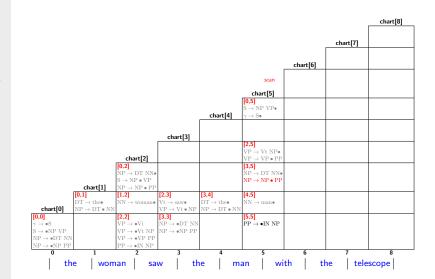
Earley Algorithm



Trees and

Constituency Parsing

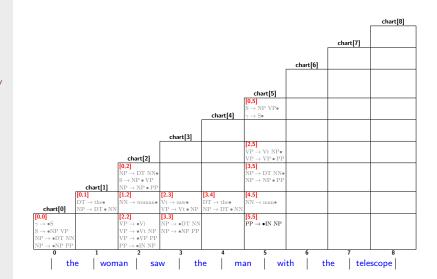
Earley Algorithm



Trees and

Constituency Parsing

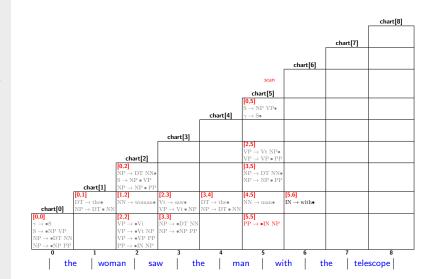
Earley Algorithm



Trees and

Constituency Parsing

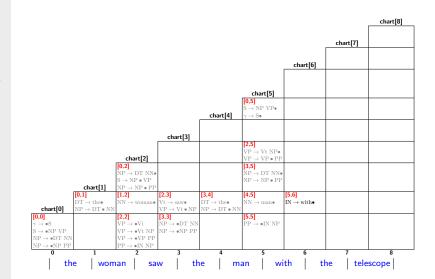
Earley Algorithm



Trees and

Constituency Parsing

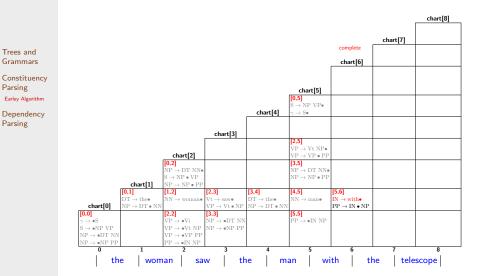
Earley Algorithm



Trees and Grammars

Parsing

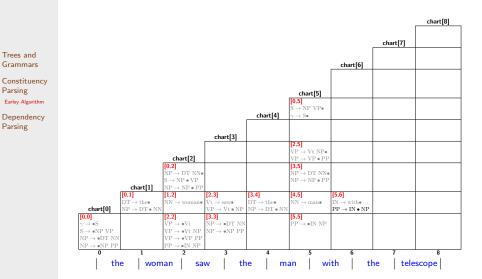
Earley Algorithm



Trees and Grammars

Parsing

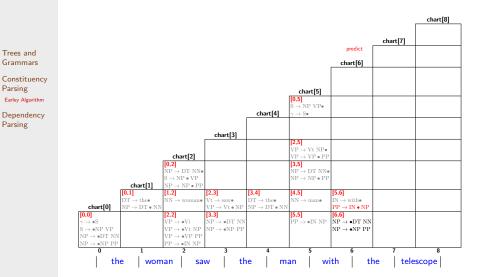
Earley Algorithm



Trees and Grammars

Parsing

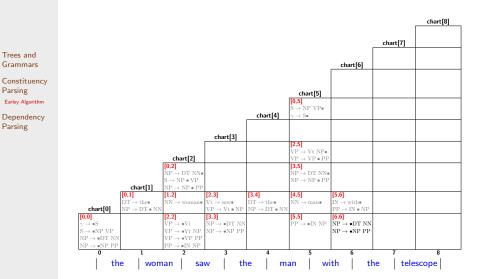
Earley Algorithm



Trees and Grammars

Parsing

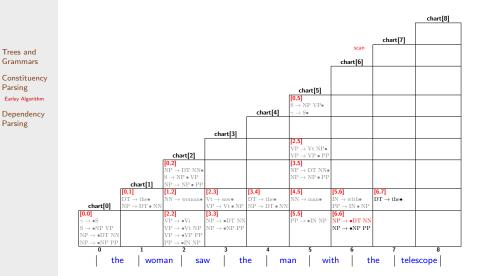
Earley Algorithm



Trees and Grammars

Parsing

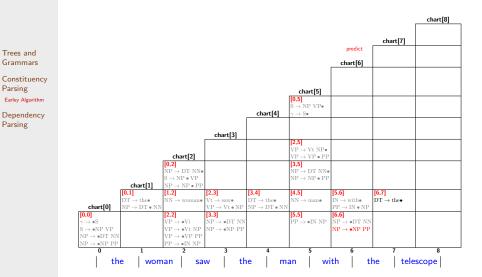
Earley Algorithm



Trees and Grammars

Parsing

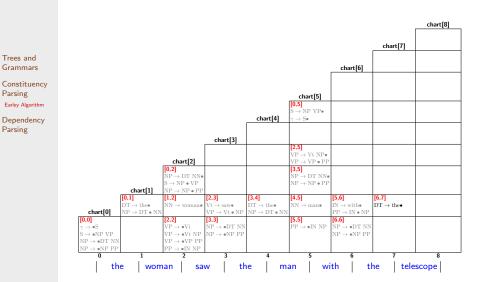
Earley Algorithm



Trees and Grammars

Parsing

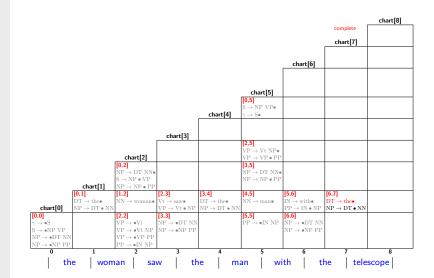
Earley Algorithm



Trees and Grammars

Constituency Parsing

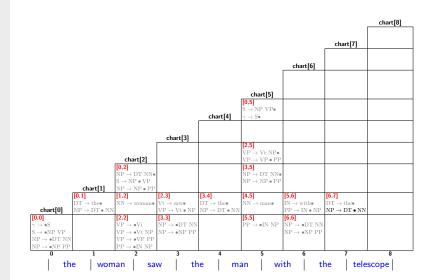
Earley Algorithm



Trees and Grammars

Constituency Parsing

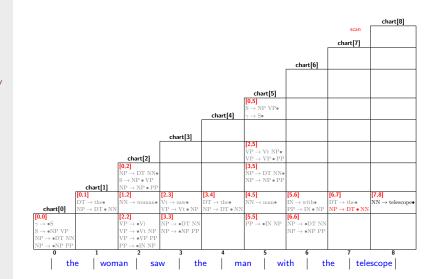
Earley Algorithm



Trees and

Constituency Parsing

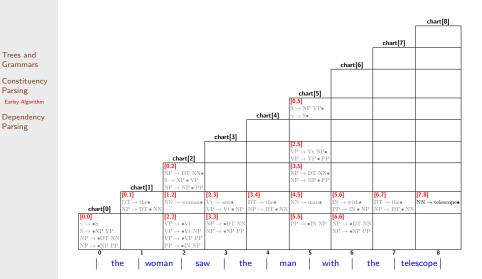
Earley Algorithm



Trees and Grammars

Parsing

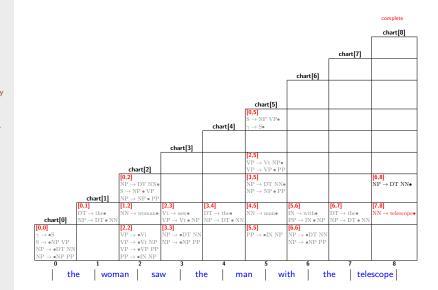
Earley Algorithm



Trees and

Constituency Parsing

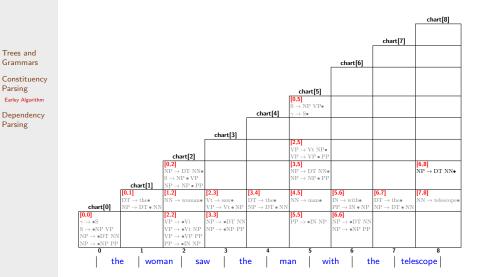
Earley Algorithm



Trees and Grammars

Parsing

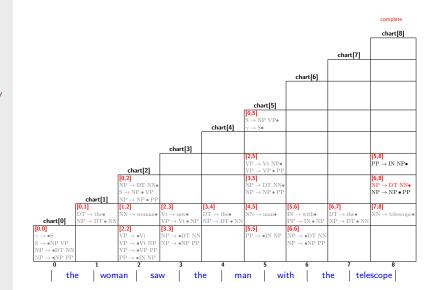
Earley Algorithm



Trees and

Constituency Parsing

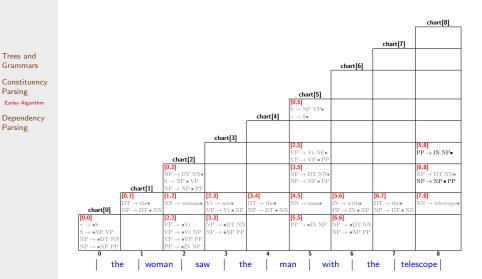
Earley Algorithm



Trees and Grammars

Parsing

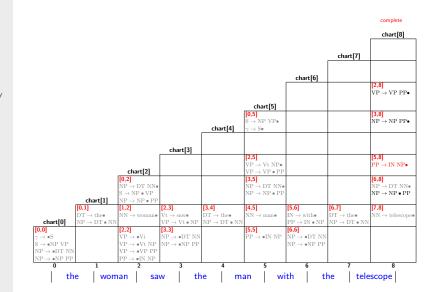
Earley Algorithm



Trees and

Constituency Parsing

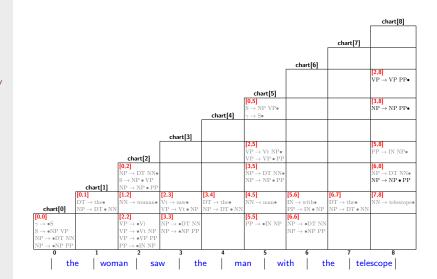
Earley Algorithm



Trees and Grammars

Constituency Parsing

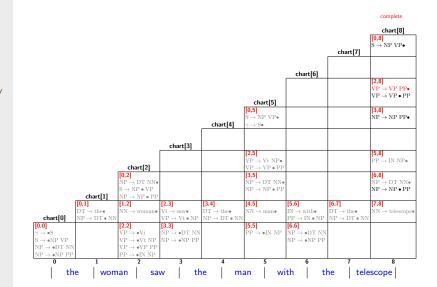
Earley Algorithm



Trees and

Constituency Parsing

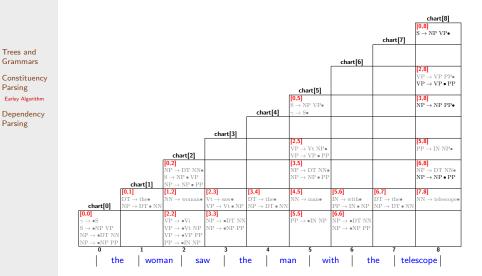
Earley Algorithm



Trees and Grammars

Parsing

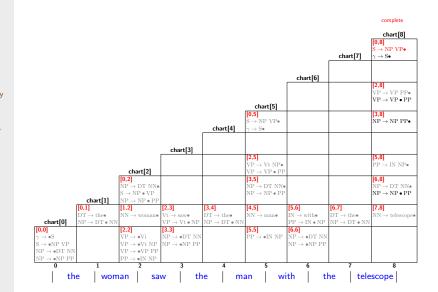
Earley Algorithm



Trees and

Constituency Parsing

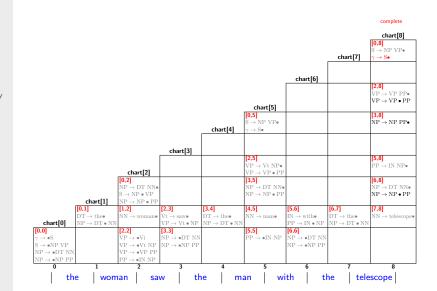
Earley Algorithm



Trees and

Constituency Parsing

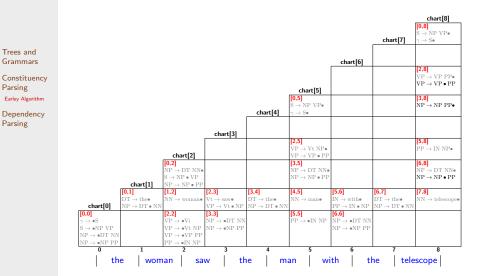
Earley Algorithm



Trees and Grammars

Parsing

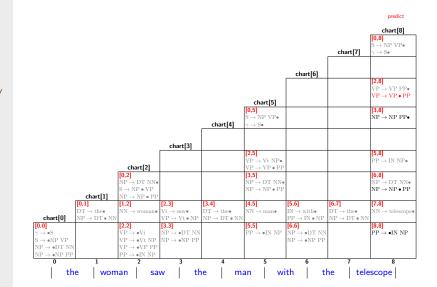
Earley Algorithm



Trees and

Constituency Parsing

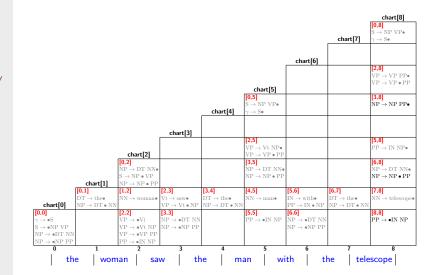
Earley Algorithm



Trees and Grammars

Constituency Parsing

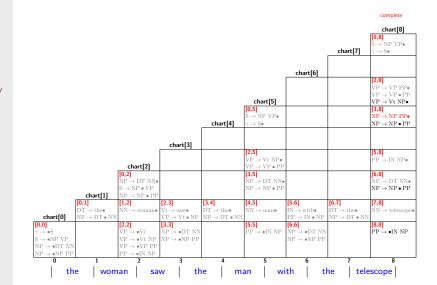
Earley Algorithm



Trees and

Constituency Parsing

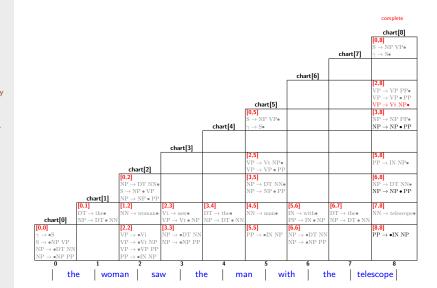
Earley Algorithm



Trees and

Constituency Parsing

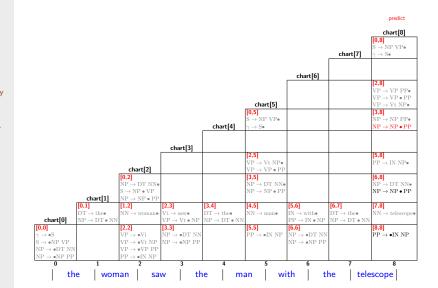
Earley Algorithm



Trees and

Constituency Parsing

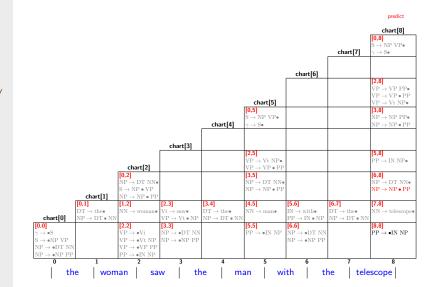
Earley Algorithm



Trees and

Constituency Parsing

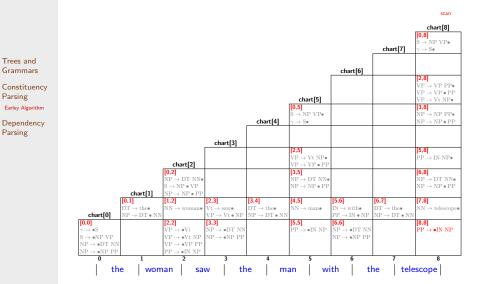
Earley Algorithm



Trees and Grammars

Parsing

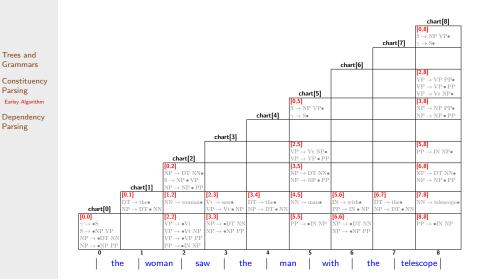
Earley Algorithm



Trees and Grammars

Parsing

Earley Algorithm



#### CKY vs Earley

Trees and Grammars

Constituency Parsing Earley Algorithm

Dependency Parsing

#### **CKY**

- Bottom-up
- Requires CNF
- Can compute all trees
- $\mathcal{O}(n^3)$
- Straightforward probabilistic version

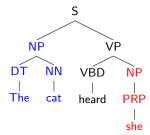
#### **Earley**

- Top-down
- Any CFG can be used, no need for CNF
- Can compute all trees
- $\mathcal{O}(n^3)$
- Not so straightforward probabilistic version

## Why *context-free* ?

 Context-free means independent of the context, i.e. assumes that any expansion of a non-terminal is applicable, regardless of the context in which it occurs.

Constituency NP VΡ Dependency **PRP** VŔD ΝP She heard the cat



Trees and Grammars

Parsing Earley Algorithm

Parsing

#### Natural Language is not Context-Free

Trees and Grammars

Constituency Parsing Earley Algorithm

Dependency Parsing NP expansion (for instance) is highly dependent on the parent of the NP



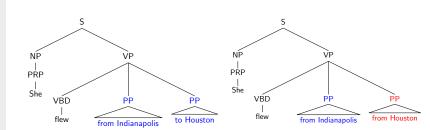
 Complete context independence is a too strong independence assumption for natural language.

## Natural Language is not Context-Free

 The application of a rule may affect the applicability of others

Constituency Parsing Earley Algorithm

Trees and Grammars



#### Natural Language is not Context-Free

Trees and Grammars

Constituency Parsing Earley Algorithm

Dependency Parsing May contain non-projective structures:

John saw the dog yesterday which was a Yorkshire Terrier

#### Outline

- 1 Trees and Grammars
  - 2 Constituency Parsing
    - CKY Algorithm
    - Earley Algorithm
  - 3 Dependency Parsing
    - Dependency Trees
    - Arc-factored Dependency Parsing
    - Parsing Projective Structures
    - Parsing non-Projective Structures
    - Transition-Based parsers

#### Trees and Grammars

Constituency Parsing

#### Outline

- 1 Trees and Grammar
  - 2 Constituency Parsing
    - CKY Algorithm
    - Earley Algorithm
    - 3 Dependency Parsing
      - Dependency Trees
      - Arc-factored Dependency Parsing
      - Parsing Projective Structures
      - Parsing non-Projective Structures
      - Transition-Based parsers

# Trees and Grammars

Constituency Parsing

Dependency Parsing Dependency Trees

## **Dependency Trees**

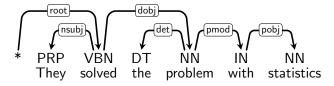
Trees and Grammars

Constituency Parsing

Dependency

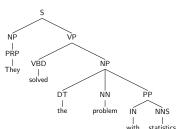
Parsing

Dependency Trees



## Theories of Syntactic Structure

#### Constituent Trees



Trees and Grammars

Constituency

Dependency

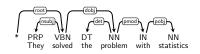
Dependency Trees

Parsing

Parsing

- Main element: constituents (or phrases, or bracketings)
- Constituents = abstract linguistic units
- Focus on word order
- Builds nested trees

#### Dependency Trees



- Main element: dependency
- Focus on relations between words
- Nicely handles free word order (fish the cat eats\*) and non-projectivity (John saw the dog yesterday which was a Yorkshire Terrier)
- Builds dependency graphs

#### Non-projective dependency trees

\* John saw a dog yesterday which was a Yorkshire Terrier

\* a hearing is scheduled on the issue today



©Starbuck's 2013

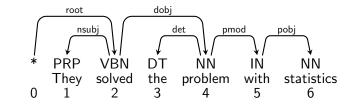
Trees and

Constituency Parsing

Dependency Parsing

Dependency Trees

### Dependency trees



Grammars Constituency Parsing

Trees and

Dependency Parsing

- \* is a special root symbol
- Each dependency is a tuple (h, m, l) where
  - h is the index of the head word (root is 0)
  - m is the index of the modifier word
  - l is a dependency label
  - $\bullet$  e.g.: (0, 2, root), (2, 1, nsubj), (2, 5, dobj), (4, 3, det), (4, 5, pmod), (5, 6, pobj)
- Sometimes we just consider unlabeled dependencies

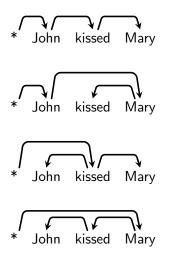
# Dependency trees for "John kissed Mary"

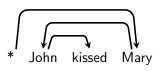
Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing



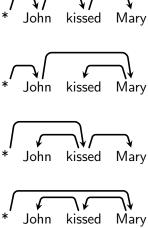


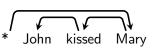
# Dependency trees for "John kissed Mary"

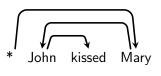
Trees and Grammars

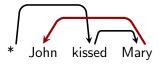
Constituency Parsing

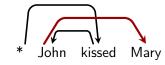
Dependency Parsing



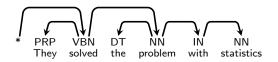








#### Conditions on Dependency Structures

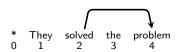


Trees and Grammars Constituency Parsing

Dependency Parsing

- y is a dependency tree if:
  - (a) Each non-root token has exactly an incoming arc (i.e. one parent)
  - The graph is connected
  - There are no cycles
  - That is, dependency arcs form a directed tree rooted at \*
- y is a projective dependency tree if:
  - Is a dependency tree
  - There are no crossing dependencies
- Note that a projective tree is also in the non-projective set -must be read as non-necessarily-projective

#### Some Notation



Trees and Grammars
Constituency

Dependency Parsing

Parsing

Parsing
Dependency Trees

Given a sentence with n words:

 ${f D}$  is the set of all possible dependencies that can be assigned to the sentence. Eg.

$$\mathcal{D} = \left\{ \begin{array}{cc} (0,1), (0,2), (0,3), (0,4), (1,2), (1,3), (1,4) \\ (2,1), (2,3), (2,4), (3,1), (3,2), (3,4) \\ (4,1), (4,2), (4,3) \end{array} \right\}$$

- $lue{y}$  is a valid parse for s if:
  - $\mathbf{y} \subseteq \mathcal{D}$
  - y is a dependency tree
- $\mathcal{Y} \subseteq 2^{\mathcal{D}}$  is the set of all valid dependency trees for the sentence

#### Outline

- 1 Trees and Grammars
  - 2 Constituency Parsing
    - CKY Algorithm
    - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

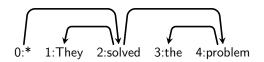
# Trees and Grammars

Constituency Parsing

Dependency Parsing

Arc-factored Dependency Parsing

### Probabilistic Arc-Factored Dependency Parsing



Trees and Grammars

Constituency Parsing

Dependency Parsing

Arc-factored

Dependency Parsing

- Assume we have  $p(\text{modifier word} \mid \text{head word})$
- In a probabilistic arc-factored model:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}, (*, 2), (2, 1), (2, 4), (4, 3))$$

$$= p(\mathbf{x}_{2}, (*, 2)) \times p(\mathbf{x}, (2, 1), (2, 4), (4, 3) | \mathbf{x}_{2}, (*, 2))$$

$$= p(*) \times p(\mathbf{x}_{2} | *) \times p(\mathbf{x}, (2, 1), (2, 4), (4, 3) | \mathbf{x}_{2}, (*, 2))$$

$$= \dots$$

$$= p(\mathbf{x}_{2} | *) \times p(\mathbf{x}_{1} | \mathbf{x}_{2}) \times p(\mathbf{x}_{4} | \mathbf{x}_{2}) \times p(\mathbf{x}_{3} | \mathbf{x}_{4})$$

$$= \prod_{(h,m) \in \mathbf{y}} p(\mathbf{x}_{m} | \mathbf{x}_{h})$$

Note that we assume independence between arcs

# Towards Linear Arc-Factored Dependency Parsing

Consider an arc-factored probabilistic model

where  $score(\mathbf{x}, h, m) = log p(\mathbf{x}_m \mid \mathbf{x}_h)$ 

$$p(\mathbf{x}, \mathbf{y}) = \prod_{(h,m) \in \mathbf{y}} p(\mathbf{x}_m \mid \mathbf{x}_h)$$

Prediction is:

$$\underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} p(\mathbf{x}, \mathbf{y}) = \underset{\mathbf{y}}{\operatorname{argmax}} \prod_{(h,m) \in \mathbf{y}} p(\mathbf{x}_m \mid \mathbf{x}_h) \\
= \underset{\mathbf{y}}{\operatorname{argmax}} \exp \left\{ \sum_{(h,m) \in \mathbf{y}} \log p(\mathbf{x}_m \mid \mathbf{x}_h) \right\} \\
= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{(h,m) \in \mathbf{y}} \log p(\mathbf{x}_m \mid \mathbf{x}_h) \\
= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{\mathbf{x} \in \mathcal{Y}} \operatorname{score}(\mathbf{x}, h, m)$$

Trees and Grammars

Constituency Parsing

Dependency Parsing

Dependency Parsing

# A CRF for Arc-Factored Dependency Parsing

lacksquare A log-linear distribution of trees f y given f x

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp(\sum_{(h, m, l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l))}{Z(\mathbf{x}; \mathbf{w})}$$

- $\mathbf{f}(\mathbf{x}, h, m)$  is a vector of d features of (h, m, l) assigned to x
- $\mathbf{w} \in \mathbb{R}^d$  are the parameters of the model
- $Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(\sum_{(h,m,l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l))$
- Prediction is linear:

$$\underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \frac{\exp(\sum_{(h,m,l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l))}{Z(\mathbf{x}; \mathbf{w})}$$
$$= \underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \sum_{(h,m,l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l)$$

Trees and Grammars

Constituency Parsing Dependency

Parsing
Arc-factored
Dependency Parsing

### Features in Arc-Factored Dependency Parsing

 $\mathbf{f}(\mathbf{x},l,h,m)$ : a vector of features of (h,m,l) assigned to x

- As in PoS tagging or NERC, we typically use indicator features
- Templates in (McDonald et al 2005):

word features
h-word, $h$ -pos
h-word
$h ext{-pos}$
m-word, $m$ -pos
<i>m</i> -word
<i>m</i> -pos

· · · · · · · · · · · · · · · · · · ·
dependency features
$h ext{-word}$ , $h ext{-pos}$ , $m ext{-word}$ , $m ext{-pos}$
$h ext{-pos},\ m ext{-word},\ m ext{-pos}$
$h ext{-}word,\ m ext{-}word,\ m ext{-}pos$
$h ext{-word},\ h ext{-pos},\ m ext{-pos}$
h-word, $h$ -pos, $m$ -word
h-word, $m$ -word
h-pos, $m$ -pos

Example: (feature template + dependency direction)

$$\mathbf{f}_j(\mathbf{x}, h, m, l) = \left\{ \begin{array}{ll} 1 & \text{if } \operatorname{word}(h) = \mathit{solve} \text{ and } \operatorname{word}(m) = \mathit{problem} \\ & \text{and } l = \mathit{dobj} \text{ and } h < m \\ 0 & \text{otherwise} \end{array} \right.$$

Trees and Grammars

Constituency Parsing

Dependency Parsing Arc-factored

Dependency Parsing

### A CRF for Arc-Factored Dependency Parsing

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp(\sum_{(h, m, l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l))}{Z(\mathbf{x}; \mathbf{w})}$$

■ Parameter estimation: Learn parameters w given training data

$$\left\{ (\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)}) \right\}$$

Decoding: predict the best dependency tree for x

$$\underset{\mathbf{v} \in \mathcal{V}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}; \mathbf{w})$$

when

- $\mathbf{y}$  is the set of projective trees for  $\mathbf{x}$
- $\mathbf{y}$  is the set of non-projective trees for  $\mathbf{x}$

Trees and Grammars

Constituency Parsing

Dependency Parsing

Arc-factored Dependency Parsing

# Parameter Estimation: CRFs for Parsing

...analogous to CRFs for Tagging

■ Goal: Estimate w given a training set

$$\left\{ (\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)}) \right\}$$

Define the conditional log-likelihood of the data:

$$L(\mathbf{w}) = \frac{1}{m} \sum_{k=1}^{m} \log P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$$

 $L(\mathbf{w})$  measures how well  $\mathbf{w}$  explains the data. A good value for  $\mathbf{w}$  will give a high value for  $P(\mathbf{y}^{(k)}|\mathbf{x}^{(k)};\mathbf{w})$  for all training examples  $k=1\ldots m$ .

lacktriangle We want lacktriangle that maximizes L(lacktriangle)

Trees and

Constituency Parsing Dependency

Parsing

Dependency Parsing

#### Learning the Parameters of a CRF

...analogous to CRFs for Tagging

Consider a regularized objective:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} L(\mathbf{w}) - \frac{\lambda}{2} ||\mathbf{w}||^2$$

where

- The first term is the log-likelihood of the data
- The second term is a regularization term, it penalizes solutions with large norm
- $flue{\lambda}$  is a parameter to control the trade-off between fitting the data and model complexity

Trees and Grammars Constituency

Parsing
Dependency

Parsing Arc-factored

Dependency Parsing

#### Learning the Parameters of a CRF

...analogous to CRFs for Tagging

Find

Trees and

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} L(\mathbf{w}) - \frac{\lambda}{2} ||\mathbf{w}||^2$$

Grammars
Constituency
Parsing

Dependency

Parsing
Arc-factored
Dependency Parsing

- In general there is no analytical solution to this optimization
- We use iterative techniques, i.e. gradient-based optimization
  - 1 Initialize  $\mathbf{w} = \mathbf{0}$
  - **2** Take derivatives of  $L(\mathbf{w}) \frac{\lambda}{2} ||\mathbf{w}||^2$ , compute gradient
  - f 3 Move f w in steps proportional to the gradient
  - 4 Repeat steps 2 and 3 until convergence

### Computing the gradient

...analogous to CRFs for Tagging

Trees and Grammars Constituency

Constituence Parsing

Dependency

Parsing

Arc-factored Dependency Parsing

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_{j}} = \frac{1}{m} \sum_{k=1}^{m} \mathbf{f}_{j}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$$
$$-\sum_{k=1}^{m} \sum_{\mathbf{y} \in \mathcal{Y}^{*}} P(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w}) \mathbf{f}_{j}(\mathbf{x}^{(k)}, \mathbf{y})$$

where

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

- First term: observed mean feature value
- Second term: expected feature value under current w

### Computing the gradient

...analogous to CRFs for Tagging

■ The first term is easy to compute, by counting explicitly

$$\frac{1}{m} \sum_{k=1}^{m} \sum_{(h,m,l) \in \mathbf{y}^{(k)}} \mathbf{f}_j(\mathbf{x}, h, m, l)$$

■ The second term is more involved,

$$\sum_{k=1}^{m} \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w}) \sum_{(h,m,l) \in \mathbf{y}} \mathbf{f}_{j}(\mathbf{x}^{(k)}, h, m, l)$$

because it sums over all sequences  $\mathbf{y} \in \mathcal{Y}$ 

lacktriangleright There exist efficient algorithms for summing over  $\mathcal{Y}$ , both for projective and non-projective sets of trees

Trees and Grammars

Constituency Parsing

Dependency Parsing

Dependency Parsing

#### Outline

- 1 Trees and Grammars
  - 2 Constituency Parsing
    - CKY Algorithm
    - Earley Algorithm
  - 3 Dependency Parsing
    - Dependency Trees
    - Arc-factored Dependency Parsing
    - Parsing Projective Structures
    - Parsing non-Projective Structures
    - Transition-Based parsers

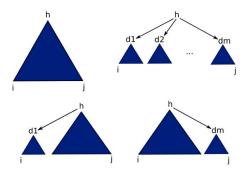
# Trees and Grammars

Constituency Parsing

Dependency Parsing

# Parsing Projective Structures (I)

- Any projective tree can be written as the combination of:
  - two smaller adjacent projective trees and
  - a dependency connecting their roots



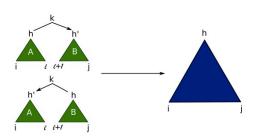
Trees and Grammars

Constituency Parsing

Dependency Parsing

# Parsing Projective Structures (II)

- The algorithm is a variation of CKY
- lacksquare  $\pi[i,j,h]$ : score of dependency tree from i to j with head h



$$\begin{split} \pi[i,j,h] &= \max_{\substack{i \leq l < j \\ 1 \leq k \leq K}} \quad \{ \quad \max_{l < h' \leq j} \pi[i,l,h] + \pi[l+1,j,h'] + \mathbf{w} \cdot \mathbf{f}(\mathbf{x},h,h') \quad , \\ \max_{i \leq h' \leq l} \pi[i,l,h'] + \pi[l+1,j,h] + \mathbf{w} \cdot \mathbf{f}(\mathbf{x},h,h') \quad \} \end{split}$$

• Cost:  $O(Kn^5)$ 

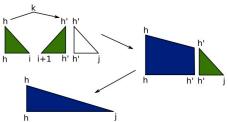
Trees and Grammars

Constituency Parsing

Dependency Parsing

### Parsing Projective Structures (III)

- Main idea: split constituents in half so that heads are at Trees and
- Parsing
- Dependency Parsing
- Parsing Projective Structures
- Grammars the boundary Constituency



 $\blacksquare$  (Eisner 1996), (Eisner 2000): an algorithm in  $O(Kn^3)$ 

#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

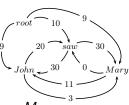
#### Trees and Grammars

Constituency Parsing

Dependency Parsing

### Parsing Non-Projective Structures

 (McDonald et al 2005): non-projective parsing as maximum-spanning trees, using the Chu-Liu-Edmonds algorithm



- Example for John saw Mary
- Build a graph:
  - Nodes are tokens (and the root token)
  - A weighted directed edge between any two vertices

$$w_{i,j} = \max_{1 \le k \le K} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, j, k)$$

Trees and Grammars

Constituency Parsing

Dependency Parsing

#### Chu-Liu-Edmonds, example

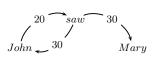
Trees and Grammars

Constituency Parsing

Dependency

Parsing

Parsing non-Projective Structures ■ Step 1: for each word, find highest-scoring incoming edge



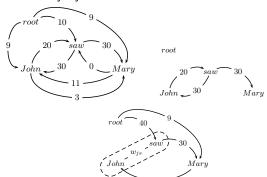
If we get a tree, we have found the MST

root

■ If not, there has to be a cycle

#### 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
  - lacksquare i 
    ightarrow c: max weight of tree with root i that spans c

$$root \rightarrow saw \rightarrow John: 40$$
  
 $root \rightarrow John \rightarrow saw: 29$ 

Trees and

Constituency Parsing

Dependency Parsing

#### Chu-Liu-Edmonds

■ Theorem (Leonidas 2003): the weight of the MST on the contracted graph is equal to the weight of the MST in the original graph

root 40 9 saw 1 30 Mary 31

Recursively call the algorithm on the new graph

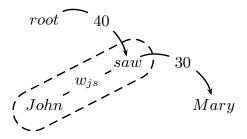
Trees and Grammars

Constituency Parsing

Dependency Parsing

#### Chu-Liu-Edmonds

■ After one recursive call we get



- It is a tree! (if not, contract and recurse)
- The original MST can be reconstructued by undoing the contraction operations (see (McDonald et al 2005) for details)
- Cost:  $O(n^3)$  (naive),  $O(n^2)$  (improved)

Trees and Grammars

Constituency Parsing

Dependency Parsing

#### Outline

- 1 Trees and Grammars
- 2 Constituency Parsing
  - CKY Algorithm
  - Earley Algorithm
- 3 Dependency Parsing
  - Dependency Trees
  - Arc-factored Dependency Parsing
  - Parsing Projective Structures
  - Parsing non-Projective Structures
  - Transition-Based parsers

# Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based parsers

#### Transition-Based parsers

Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based

- Inspired on shift-reduce 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.
- Parsing stops when a final configuration is reached
- No backtracking, cost is  $\mathcal{O}(n)$

	woman NN							telescope NN	
	Stack	Buffe	r		Tr	ansition			
		DT N	IN Vt [	I NN TC					

Trees and Grammars

Constituency Parsing

Dependency Parsing

	woman NN							telescope NN	
_	Stack	Buffe	r			Tr	ansition		
		DT N	IN Vt [	I NN TC	N	sh	ift		

Trees and Grammars

Constituency Parsing

Dependency Parsing

		woman NN							telescope NN
_	Stack Buffer								ansition
	DT NN Vt DT NN IN DT NN DT NN Vt DT NN IN DT NN								ift

Trees and Grammars

Constituency Parsing

Dependency Parsing

	woman NN							telescope NN
Stack Buffer								ansition
DT NN Vt DT NN IN DT NN DT NN Vt DT NN IN DT NN								ift ift

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based parsers

The	woman	saw	the	man	with	the	telescope
DT	NN	Vt	DT	NN	IN	DT	NN
	Stack	Buffe	r			٦	<b>Transition</b>
		DT N	IN Vt [	I NN TC	IN DT N	IN s	hift
	DT	NN V	t DT l	NN IN D	T NN	S	hift
	DT NN	Vt D	T NN I	N DT N	IN		

Trees and Grammars Constituency Parsing

Dependency Parsing

	woman NN						telescope NN
	Stack	Т	ransition				
	DT DT NN	sł	nift nift educe NP→DT NN				

Grammars
Constituency
Parsing

Trees and

Dependency Parsing

								telescope
	DT	NN	Vt	DΤ	NIN	IIN	וט	NN
		Stack	Buffe	r			Ti	ransition
_			DT N	IN Vt [	NN TC	N DT N	IN sh	ift
		DT	sh	ift				
		DT NN	Vt D	T NN I	N DT N	N	re	duce NP $\rightarrow$ DT NN
		NP	Vt D	T NN I	N DT N	N		

Grammars

Constituency
Parsing

Trees and

Dependency Parsing

The DT	woman NN			man NN			telescope NN
	Stack	Buffe	r			Т	ransition
		DT N	IN Vt [	I NN TC	N DT N	IN sh	nift
	DT	NN V	t DT ľ	NN IN D	T NN	sh	nift
	DT NN	Vt D	T NN I	N DT N	N	re	educe NP→DT NN
	NP	Vt D	T NN I	N DT N	N	sł	nift

Grammars
Constituency
Parsing

Trees and

Dependency Parsing

Parsing

Transition-Based parsers

The DT	woman NN				with IN		e telescope NN
	Stack	Buffe	r				Transition
		DT N	IN Vt I	I NN TC	IN DT N	N	shift
	DT	NN V	/t DT I	NN IN D	NN TO		shift
	DT NN	Vt D	T NN I	N DT N	IN		reduce NP $\rightarrow$ DT NN
	NP	Vt D	T NN I	N DT N		shift	
	NP Vt	DT N	IN IN I	NN TC			

Grammars Constituency Parsing

Trees and

Dependency Parsing

The DT	woman NN				with IN				ppe
	Stack	Buffe	r				Tra	nsition	
		DT N	IN Vt [	I NN TC	IN DT N	N	shi	ft	
	DT	NN V	t DT ľ	NN IN D	T NN		shi	ft	
	DT NN	Vt D	T NN I	N DT N	IN		red	uce NP-	ightarrowDT NN
	NP	Vt DT NN IN DT NN					shi	ft	
	NP Vt	DT NN IN DT NN					shi	ft	

Grammars Constituency Parsing

Trees and

Dependency Parsing

Transition-Based

parsers

	The DT	woman NN			man NN		th D		telescope NN
_		Stack	Buffe	r				Tr	ansition
			DT N	IN Vt [	NN TC	IN DT N	IN	sh	ift
		DT	NN Vt DT NN IN DT NN						ift
		DT NN	Vt DT NN IN DT NN						duce NP $\rightarrow$ DT NN
		NP	Vt D	T NN I	N DT N	IN		sh	ift
		NP Vt	DT NN IN DT NN						ift
	1	NP Vt DT	NN I	N DT I	ΝN				

Trees and Grammars Constituency Parsing

Dependency

The woman DT NN			man NN			e telescope T NN
Stack	Buffe	r				Transition
	DT N	IN Vt [	I NN TC	N DT N	N	shift
DT	NNV	t DT I	NN IN D	shift		
DT NN	Vt D	T NN I	N DT N	N		reduce NP $\rightarrow$ DT NN
NP	Vt D	T NN I	N DT N	N		shift
NP Vt	DT N	IN IN E	NN TC			shift
NP Vt DT	NN II	N DT N	NN			shift

Grammars
Constituency
Parsing

Trees and

Dependency Parsing

Parsing

	The DT	woman NN	saw Vt	the DT	man NN	with IN	th D		elescope NN
		Stack	Buffe	r		•		Tran	sition
_			DT N	IN Vt [	I NN TC	IN DT N	IN	shift	
		DT	NN V	t DT ľ	NN IN D	T NN		shift	
		DT NN	Vt D	T NN I	N DT N	IN		redu	ce NP→DT NN
		NP	Vt D	T NN I	N DT N	IN		shift	
		NP Vt	DT N	IN IN E	NN TC			shift	
	ļ	NP Vt DT	NN I	N DT I	NN			shift	
	NP V	√t DT NN	IN D	T NN					

Trees and Grammars

Constituency Parsing

Dependency Parsing

	The DT	woman NN	saw Vt	the DT		with IN	th D		telescope NN
_		Stack	Buffe	r				Tr	ansition
			DT N	IN Vt I	I NN TC	IN DT N	N	sh	ift
		DT	NN V	/t DT I	NN IN D	NN TO		sh	ift
		DT NN	Vt D	T NN I	N DT N	IN		rec	duce NP $\rightarrow$ DT NN
		NP	Vt D	T NN I	N DT N	IN		sh	ift
		NP Vt	DT N	IN IN I	NN TC			sh	ift
		NP Vt DT	NN I	N DT I	NN			sh	ift
	NP '	Vt DT NN	IN D	T NN				rec	duce NP $\rightarrow$ DT NN

Trees and Grammars

Constituency Parsing

Dependency Parsing

	The DT	woman NN	saw Vt	the DT	man NN	with IN	th D		telescope NN
_	וט				1414	1114			
		Stack	Buffe	r				Tra	nsition
			DT N	IN Vt [	I NN TC	IN DT N	N	shif	t
		DT	NN V	t DT ľ	NN IN D	T NN		shif	t
		DT NN	Vt D	T NN I	N DT N	IN		redi	$uce\;NP{\to}DT\;NN$
		NP	Vt D	T NN I	N DT N	IN		shif	t
		NP Vt	DT N	IN IN E	NN TC			shif	t
	ı	NP Vt DT	NN I	N DT I	NN			shif	t
	NP \	√t DT NN	IN D	T NN				redu	$uce\;NP{\to}DT\;NN$
	I	NP Vt NP	IN D	T NN					

Trees and Grammars

Constituency Parsing

Dependency Parsing

 he T	woman NN	saw Vt	the DT	man NN	with IN	th D		telescope NN
	Stack	Buffe	r				Tra	ansition
		DT N	IN Vt [	I NN TC	IN DT N	IN	shi	ft
	DT	NN V	t DT l	NN IN D	NN T		shi	ft
	DT NN	Vt D	T NN I	N DT N	IN		red	luce NP→DT NN
	NP	Vt D	T NN I	N DT N	IN		shi	ft
	NP Vt	DT N	IN IN E	OT NN			shi	ft
N	IP Vt DT	NN II	N DT I	NN			shi	ft
NP V	t DT NN	IN D	T NN				red	luce NP→DT NN
1	NP Vt NP	IN D	T NN				*re	educe $VP \rightarrow Vt NP$

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw	the	man	with	th	ie	telescope
DT NN	Vt	DT	NN	IN	D	Т	NN
Stack	Buffe	r				Tra	ansition
	DT N	N Vt I	NN TC	IN DT	NN	shi	ft
DT	NN V	t DT I	NI NI	DT NN		shi	ft
DT NN	Vt D	T NN I	N DT	NN		red	luce NP $\rightarrow$ DT NN
NP	Vt D	T NN I	N DT	NN		shi	ft
NP Vt	DT N	N IN [	NN TC			shi	ft
NP Vt DT	NN II	I TO I	NN			shi	ft
NP Vt DT NN	IN D	ΓNN				red	luce NP $\rightarrow$ DT NN
NP Vt NP	IN D	ΓNN				*re	educe VP→Vt NP
NP VP	IN D	ГΝΝ					

Trees and Grammars

Constituency Parsing

Dependency Parsing

Th	e	woman	saw	the	man	with	th	ie	telescope		
DT	-	NN	Vt	DT	NN	IN	D	Т	NN		
		Stack	Buffe	r				Tr	ransition		
			DT N	IN Vt I	DT NN	IN DT I	ΝN	sh	ift		
		DT	NN V	t DT I	NN IN E	NN TC		sh	ift		
		DT NN	Vt DT NN IN DT NN						${\sf reduce}\;{\sf NP} {\rightarrow} {\sf DT}\;{\sf NN}$		
		NP	Vt D	T NN I	IN DT N	١N		sh	ift		
		NP Vt	DT N	IN IN I	DT NN			sh	ift		
	ı	NP Vt DT	NN II	N DT I	NN			sh	ift		
N	Р١	Vt DT NN	IN D	T NN				re	duce NP $\rightarrow$ DT NN		
	-	NP Vt NP	IN D	T NN				*r	educe $VP \rightarrow Vt NP$		
		NP VP	IN D	T NN				sh	ift		
			i.								

Trees and Grammars

Constituency Parsing

Dependency Parsing

The	woman	saw	the	man	with	the	telescope	
DT	NN	Vt	DT	NN	IN	DT	NN	
	Stack	Buffe	r			Tı	ransition	
		DT N	IN Vt [	NN TC	IN DT N	N sh	ift	
	DT	NN \	/t DT ľ	NN IN C	NN TO	sh	ift	
	DT NN	Vt D	T NN I	N DT N	re	duce NP $\rightarrow$ DT NN		
	NP	Vt D	T NN I	N DT N	sh	shift		
	NP Vt	DT N	IN IN [	NN TC	sh	ift		
ļ	NP Vt DT	NN I	N DT I	NN		sh	ift	
NP V	/t DT NN	IN D	T NN			re	duce NP $\rightarrow$ DT NN	
	NP Vt NP	IN D	T NN			* r	educe VP→Vt NP	
	NP VP	IN D	T NN			sh	ift	
	NP VP IN	DT N	IN					

Trees and Grammars

Constituency Parsing

Dependency Parsing

The	woman	saw	the	man	with	the	telescope	
DT	NN	Vt	DT	NN	IN	DT	NN	
	Stack	Buffe	r			Tı	ransition	
		DT N	IN Vt [	OT NN	IN DT N	N sh	ift	
	DT	NN \	/t DT <mark>[</mark>	NN IN E	OT NN	sh	ift	
	DT NN	Vt D	T NN I	NDTN	IN	re	duce NP $\rightarrow$ DT NN	
	NP	Vt D	T NN I	NDTN	sh	shift		
	NP Vt	DT NN IN DT NN					ift	
	NP Vt DT	NN I	N DT I	NN		sh	ift	
NP '	Vt DT NN	IN D	T NN			re	duce NP→DT NN	
	NP Vt NP	IN D	T NN			* r	educe VP→Vt NP	
	NP VP	IN D	T NN			sh	ift	
	NP VP IN	DT N	IN			sh	ift	
		1						

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw	the	man	with	the	telescope	
DT NN	Vt	DT	NN	IN	DT	NN	
Stack	Buffe	r		Tr	Transition		
DT DT NN NP NP Vt NP Vt DT NP Vt DT NN	NN V Vt D <sup>-</sup> Vt D <sup>-</sup> DT N	t DT I F NN I F NN I N IN I N DT I	NN IN D N DT N N DT N DT NN	N	sh re sh sh	ift ift duce NP→DT NN ift ift ift duce NP→DT NN	
NP Vt <mark>NP</mark> NP VP NP VP IN NP VP IN DT	IN D <sup>-</sup> IN D <sup>-</sup> DT N NN	ГΝΝ			sh	educe VP→Vt NP ift ift	

Trees and Grammars

Constituency Parsing

Dependency Parsing

The	woman	saw	the	man	with	the	telescope	
DT	NN	Vt	DT	NN	IN	DT	NN	
	Stack	Buffe	r		Tı	ransition		
		DT N	IN Vt [	NN TC	IN sh	ift		
	DT	NN \	/t DT I	NN IN E	NN TC	sh	ift	
	DT NN Vt DT NN IN DT NN						duce NP $\rightarrow$ DT NN	
	NP	Vt D	T NN I	N DT N	sh	shift		
	NP Vt	DT N	IN IN [	NN TC	sh	shift		
	NP Vt DT	NN I	N DT I	NN	sh	shift		
NP \	Vt DT NN	IN D	T NN			re	duce NP $\rightarrow$ DT NN	
	NP Vt NP	IN D	T NN			*r	educe VP→Vt NP	
	NP VP	IN D	T NN			sh	ift	
	NP VP IN	DT N	IN		sh	shift		
NP	VP IN DT	NN				sh	ift	

Trees and Grammars

Constituency Parsing

Dependency Parsing

The	woman	saw	the	man	with	the	telescope		
DT	NN	Vt	DT	NN	IN	DT	NN		
	Stack	Buffe	r			Tı	Transition		
		DT N	IN Vt [	I NN TC	N DT N	N sh	ift		
	DT	NN V	/t DT I	NN IN D	T NN	sh	ift		
	DT NN	Vt D	T NN I	N DT N	N	re	reduce NP $\rightarrow$ DT NN		
	NP	Vt D	T NN I	N DT N	sh	shift			
	NP Vt	DT N	IN IN E	NN TC	sh	shift			
	NP Vt DT	NN IN DT NN					shift		
NP \	√t DT NN	IN D	T NN			re	reduce NP→DT NN		
	NP Vt NP	IN D	T NN			*r	*reduce VP→Vt NP		
	NP VP	IN D	T NN			sh	shift		
	NP VP IN	DT N	IN			sh	ift		
NP '	VP IN DT	NN			sh	shift			
NP VP I	N DT NN								

Trees and Grammars

Constituency Parsing

Dependency Parsing

The	woman	saw	the	man	with	the	telescope		
DT	NN	Vt	DT	NN	IN	DT	NN		
	Stack	Buffe	r			Tı	Transition		
		DT N	IN Vt I	OT NN	IN DT N	IN sh	ift		
	DT	NN \	/t DT I	NN IN E	NN TC	sh	ift		
	DT NN	Vt D	T NN I	N DT N	IN	re	duce NP $\rightarrow$ DT NN		
	NP	Vt D	T NN I	N DT N	sh	shift			
	NP Vt	DT N	IN IN [	NN TC	sh	shift			
	NP Vt DT	NN I	N DT I	NN	sh	shift			
NP '	Vt DT NN	IN D	T NN			re	${\sf reduce}\;{\sf NP}{\to}{\sf DT}\;{\sf NN}$		
	NP Vt NP	IN D	T NN		*r	$*$ reduce VP $\rightarrow$ Vt NP			
	NP VP	IN D	T NN			sh	shift		
	NP VP IN	DT N	IN			sh	ift		
NP	VP IN DT	NN				sh	ift		
NP VP	IN DT NN					re	duce NP $\rightarrow$ DT NN		

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw	the	man	with	the	telescope		
DT NN	Vt	DT	NN	IN	DT	NN		
Stack	Buffe	er			Ti	Transition		
	DT N	IN Vt I	DT NN	IN DT N	IN sh	ift		
DT	NN \	/t DT I	NN IN E	NN TO	sh	ift		
DT NN	DT NN Vt DT NN IN DT NN							
NP	NP Vt DT NN IN DT NN							
NP Vt	DT I	IN IN [	NN TC		sh	ift		
NP Vt DT	NN I	N DT I	NN		sh	shift		
NP Vt DT NN	IN D	T NN			re	duce NP $\rightarrow$ DT NN		
NP Vt NP	IN D	T NN			* r	educe VP→Vt NP		
NP VP	IN D	T NN			sh	ift		
NP VP IN	DT I	١N			sh	ift		
NP VP IN DT	NN				sh	ift		
NP VP IN DT NN				re	duce NP $\rightarrow$ DT NN			
NP VP IN NP								

Trees and Grammars

Constituency Parsing

Dependency Parsing

The w	oman	saw	the	man	with	the	telescope		
DT N	N	Vt	DT	NN	IN	DT	NN		
	Stack	Buffer				Tr	Transition		
		DT N	N Vt D	T NN I	N DT N	N sh	ift		
	DT	NN V	t DT N	IN IN D	T NN	shi	ift		
	DT NN Vt DT NN IN DT NN						duce NP $\rightarrow$ DT NN		
	NP Vt DT NN IN DT NN						ift		
	NP Vt	DT NN IN DT NN					shift		
NP	Vt DT	NN IN DT NN					shift		
NP Vt [	NN TC	IN DT	NN			red	duce NP $\rightarrow$ DT NN		
NP	Vt NP	IN DT	NN			*re	educe VP→Vt NP		
I	NP <mark>VP</mark>	IN DT	NN			sh	ift		
NP	VP IN	DT N	N			sh	ift		
NP VP	IN DT	NN				shi	shift		
NP VP IN [	VP IN DT NN					red	duce NP $\rightarrow$ DT NN		
NP VP	IN NP					red	duce PP→IN NP		

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw	the	man	with	the	telescope		
DT NN	Vt	DT	NN	IN	DT	NN		
Stack	Buffe	r			Tı	Transition		
	DT N	IN Vt [	I NN TC	IN DT N	IN sh	ift		
DT	NN V	t DT ľ	NN IN D	NN TO	sh	ift		
DT NN	Vt D	T NN I	N DT N	IN	re	duce NP $\rightarrow$ DT NN		
NP	NP Vt DT NN IN DT NN							
NP Vt	DT N	IN IN [	NN TC		sh	shift		
NP Vt DT	NN II	I TO I	NN		sh	shift		
NP Vt DT NN	IN D	T NN			re	duce NP $\rightarrow$ DT NN		
NP Vt NP	IN D	T NN			*r	*reduce VP→Vt NP		
NP VP	IN D	T NN			sh	ift		
NP VP IN	DT N	IN			sh	ift		
NP VP IN DT	NN				sh	ift		
NP VP IN DT NN					re	duce NP $\rightarrow$ DT NN		
NP VP IN <mark>NP</mark>					re	duce PP→IN NP		
NP VP PP								

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw the	man	with	the	telescope		
DT NN	Vt DT	NN	IN	DT	NN		
Stack	Buffer		Tr	Transition			
	DT NN Vt	DT NN	IN DT N	N sh	ift		
DT	NN Vt DT	NN IN D	T NN	sh	ift		
DT NN	Vt DT NN	IN DT N	IN	re	duce NP $\rightarrow$ DT NN		
NP	Vt DT NN	IN DT N	sh	shift			
NP Vt	DT NN IN	DT NN	sh	shift			
NP Vt DT	NN IN DT	NN	sh	shift			
NP Vt DT NN	IN DT NN		re	reduce NP $\rightarrow$ DT NN			
NP Vt NP	IN DT NN			*r	$educe VP \rightarrow Vt NP$		
NP VP	IN DT NN			shift			
NP VP IN	DT NN			sh	ift		
NP VP IN DT	NN			sh	ift		
NP VP IN DT NN	İ		reduce NP→DT N				
NP VP IN NP	ĺ		re	reduce PP→IN NP			
NP VP PP	ĺ			re	duce VP→VP PP		

Trees and Grammars

Constituency Parsing

Dependency Parsing

The woman	saw the man with	the telescope
DT NN	Vt DT NN IN	DT NN
Stack	Buffer	Transition
	DT NN Vt DT NN IN DT N	N shift
DT	NN Vt DT NN IN DT NN	shift
DT NN	Vt DT NN IN DT NN	reduce NP→DT NN
NP	Vt DT NN IN DT NN	shift
NP Vt	DT NN IN DT NN	shift
NP Vt DT	NN IN DT NN	shift
NP Vt DT NN	IN DT NN	reduce NP→DT NN
NP Vt NP	IN DT NN	*reduce VP→Vt NP
NP VP	IN DT NN	shift
NP VP IN	DT NN	shift
NP VP IN DT	NN	shift
NP VP IN DT NN		reduce NP→DT NN
NP VP IN NP		reduce PP→IN NP
NP VP PP		reduce VP→VP PP
NP VP		

Trees and Grammars

Constituency Parsing

Dependency Parsing

TI	.1	
The woman	saw the man wit	h the telescope
DT NN	Vt DT NN IN	DT NN
Stack	Buffer	Transition
	DT NN Vt DT NN IN D	NN shift
DT	NN Vt DT NN IN DT NI	l shift
DT NN	Vt DT NN IN DT NN	reduce NP→DT NN
NP	Vt DT NN IN DT NN	shift
NP Vt	DT NN IN DT NN	shift
NP Vt DT	NN IN DT NN	shift
NP Vt DT NN	IN DT NN	reduce NP→DT NN
NP Vt NP	IN DT NN	*reduce VP→Vt NP
NP <mark>VP</mark>	IN DT NN	shift
NP VP IN	DT NN	shift
NP VP IN DT	NN	shift
NP VP IN DT NN		reduce NP→DT NN
NP VP IN NP		reduce PP→IN NP
NP VP PP		reduce VP→VP PP
NP <mark>VP</mark>		reduce S→NP VP

Trees and Grammars

Constituency Parsing

Dependency Parsing

	The	woman	saw	the	man	with	the	telescope
	DT	NN	Vt	DT	NN	IN	DT	·
		Stack	Buffe	er				Transition
			DT N	IN Vt I	NN TC	IN DT N	IN :	shift
		DT	NN \	/t DT I	NN IN C	T NN		shift
су		DT NN	Vt D	T NN I	N DT N	IN		reduce NP $\rightarrow$ DT NN
Cy		NP	Vt D	T NN I	N DT N	IN		shift
y		NP Vt	DT N	IN IN [	NN TC			shift
		NP Vt DT	NN I	N DT I	NN			shift
ed	NP	Vt DT NN	IN D	T NN				reduce NP $\rightarrow$ DT NN
		NP Vt NP	IN D	T NN				*reduce VP $\rightarrow$ Vt NP
		NP VP	IN D	T NN				shift
		NP VP IN	DT N	١N				shift
	NP	VP IN DT	NN					shift
	NP VP	IN DT NN						reduce NP→DT NN
	NP	VP IN NP						reduce PP→IN NP
		NP VP PP						reduce $VP \rightarrow VP PP$
		NP VP						reduce $S \rightarrow NP VP$
		ς						

Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based

parsers

	The			مطه		ماعانيي	مطه	+01000000
	The	woman	saw	the	man	with	the	telescope
	DT	NN	Vt	DT	NN	IN	DT	NN
		Stack	Buffe	er			7	<b>Transition</b>
			DT I	NN Vt I	I NN TC	IN DT N	IN s	hift
		DT	NN \	/t DT I	NN IN D	T NN	S	hift
ncy		DT NN	Vt D	T NN I	N DT N	IN	r	educe NP $\rightarrow$ DT NN
iicy		NP	Vt D	T NN I	N DT N	IN	S	hift
су		NP Vt	DT I	I NI NI	NN TC		S	hift
		NP Vt DT	NN I	N DT I	NN		S	hift
sed	NP	Vt DT NN	IN D	T NN			r	educe NP→DT NN
		NP Vt NP	IN D	T NN			*	reduce VP→Vt NP
		NP VP	IN D	T NN			S	hift
		NP VP IN	DT I	NN			s	hift
	NP	VP IN DT	NN				S	hift
	NP VP	IN DT NN					r	educe NP→DT NN
	NP	VP IN NP					r	educe PP→IN NP
		NP VP PP					r	educe VP→VP PP
		NP VP					r	$educe S \rightarrow NP VP$
		S					s	top

Trees and Grammars

Constituence Parsing

Dependenc Parsing Transition-Base parsers

#### Transition-Based parsers

Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based

- Only one tree is produced: Not suitable for ambiguous grammars (common in NLP)
- We can add probabilities to select which transition is selected at each step: Similar to CKY with PCFGs, but greedy search (may be made less greedy with e.g. beam-search)
- Or better: we can add features and use ML to take the decision.

Let's see how it is applied to dependency parsing

### Arc-Standard algorithm

- Trees and Grammars
- Constituency Parsing
- Dependency Parsing

- $\blacksquare$  A configuration (S, B, A) of the parser consists of:
  - lacksquare 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
  - lacksquare left-arc: add an arc from S[0] to S[1] and remove S[1] from the stack
  - ullet right-arc: add an arc from S[1] to S[0] and remove S[0] from the stack

#### Arc-Standard Transition definitions

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based

parsers

• 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):  $([\sigma|i|j], B, A) \Rightarrow ([\sigma|i], B, A \cup \{i, j, L\})$

Stack	Buffer	Transition
	* the woman saw the man with glasses	

Trees and Grammars

Constituency

Parsing

Dependency Parsing

Transition-Based parsers

> woman saw the man with glasses

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

Trees and Grammars

Constituency Parsing

Parsing

Dependency Parsing

Transition-Based parsers

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

Trees and Grammars

Constituency Parsing

Dependency

Parsing Transition-Based parsers

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

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers

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	

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers

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

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers

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	

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers

det

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

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers

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	

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based

parsers

 $\checkmark$  \ \* the woman saw the man with glasses

_			
	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

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing

Transition-Based parsers

 $\checkmark$  \ \* the woman saw the man with glasses

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	

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based parsers



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

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based parsers

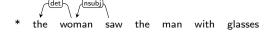
\* the woman saw the man with glasses

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

Trees and Grammars

Constituency Parsing

Dependency Parsing

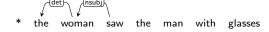


Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh

Trees and Grammars

Constituency Parsing

Dependency Parsing



Buffer Transition Stack \* 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-subi \* saw the man with glasses sh \* saw the man with glasses sh saw the man with glasses

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based



glasses

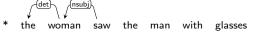
with

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

Trees and Grammars

Constituency Parsing

Dependency Parsing



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

Trees and Grammars

Constituency Parsing

Dependency Parsing



Buffer Transition Stack \* 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-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi

Trees and Grammars

Constituency Parsing

Dependency

Parsing

Transition-Based parsers

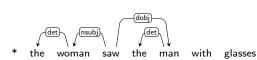


Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses

Trees and Grammars

Constituency Parsing

Dependency Parsing



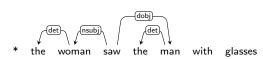
Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based



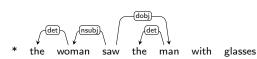
Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses

Trees and Grammars

Constituency Parsing

Dependency

Parsing Transition-Based

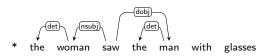


Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses sh

Trees and Grammars

Constituency Parsing

Dependency Parsing



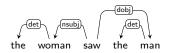
Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses sh \* saw with glasses

Trees and Grammars

Constituency Parsing

Dependency Parsing

Parsing Transition-Based



with glasses

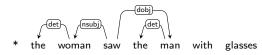
Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses sh \* saw with glasses ra-pmod

Trees and Grammars

Constituency Parsing

Dependency Parsing

Transition-Based parsers

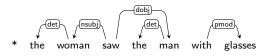


Buffer Transition Stack \* the woman saw the man with glasses sh \* the woman saw the man with glasses sh \* the woman saw the man with glasses la-det \* woman saw the man with glasses sh \* woman saw the man with glasses la-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses sh \* saw with glasses ra-pmod \* saw with

Trees and Grammars

Constituency Parsing

Dependency Parsing



Trees and

Grammars

Parsing

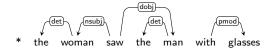
Parsing

Constituency

Dependency

Transition-Based

Buffer Transition Stack \* 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-subi \* saw the man with glasses sh \* saw the man with glasses sh \* saw the man with glasses la-det \* saw man with glasses ra-dobi \* saw with glasses sh \* saw with glasses sh \* saw with glasses ra-pmod \* saw with ra-madi

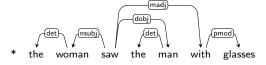


Stack

\* 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-subi \* saw the man with glasses sh Constituency \* saw the man with glasses sh \* saw the man with glasses la-det Dependency \* saw man with glasses ra-dobi \* saw with glasses sh Transition-Based \* saw with glasses sh \* saw with glasses ra-pmod

Buffer

\* saw with ra-madi \* saw



Transition

Trees and Grammars

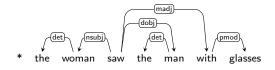
Parsing

Stack

\* saw

\* 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-subi \* saw the man with glasses sh Constituency \* saw the man with glasses sh \* saw the man with glasses la-det Dependency \* saw man with glasses ra-dobi \* saw with glasses sh Transition-Based \* saw with glasses sh \* saw with glasses ra-pmod \* saw with ra-madi

Buffer



Transition

ra-root

Trees and Grammars

Parsing

Stack

\* 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-subi \* saw the man with glasses sh Constituency \* saw the man with glasses sh \* saw the man with glasses la-det Dependency \* saw man with glasses ra-dobi \* saw with glasses sh Transition-Based \* saw with glasses sh \* saw with glasses ra-pmod \* saw with ra-madi \* saw ra-root root ĺmadi dobi pmod

saw

woman

the

man

Buffer

Transition

glasses

Trees and Grammars

Parsing

Stack

\* 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-subi \* saw the man with glasses sh Constituency \* saw the man with glasses sh \* saw the man with glasses la-det Dependency \* saw man with glasses ra-dobi \* saw with glasses sh Transition-Based \* saw with glasses sh \* saw with glasses ra-pmod \* saw with ra-madi \* saw ra-root stop root ĺmadi dobi pmod

saw

woman

the

man

Buffer

Transition

glasses

Trees and Grammars

Parsing

#### Alternative Transition Models

Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based parsers

#### 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)

#### Transition Selection

- Classifier that produces the best transition for the current configuration
- Too many possible configurations: Need to model them as feature vectors and use ML:
- Typical features:
  - word/lemma/PoS for S[0], S[1], B[0], B[1]
  - $\blacksquare$  morphological features (gender, number, mode, tense, etc) in  $S[0],\,B[0]$
  - $\blacksquare$  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

#### Trees and Grammars

Constituency Parsing

Dependency Parsing Transition-Based

#### Transition Selection

- Classifier that produces the best transition for the current configuration
- Too many possible configurations: Need to model them as feature vectors and use ML:
- Typical features:
  - lacktriangleq word/lemma/PoS for S[0], S[1], B[0], B[1]
  - $\blacksquare$  morphological features (gender, number, mode, tense, etc) in  $S[0],\,B[0]$
  - lacksquare number of children of S[0]
  - lacktriangle dependency labels of S[0] children
  - ..etc
- We can use SVM, perceptron, MBL, DT, ... any feature-based ML classifier

... or we can use Deep Learning

Trees and

Constituency Parsing

Dependency Parsing Transition-Based