Introduction to Natural Language Processing

PARSING:

Earley, Bottom-Up Chart Parsing

Jean-Cédric Chappelier

Jean-Cedric.Chappelier@epfl.ch

Artificial Intelligence Laboratory

Objectives of this lecture

➥ After CYK algorithm, present two other algorithms used for syntactic parsing

Earley Parsing

Top-down algorithm(predictive)

Bottom-up = inference

Top-down = search

3 advantages:

- best known worst-case complexity (as CYK)
- adaptive complexity for least complex languages (e.g. regular languages)
- No need for a special form of the CF grammar

2 drawbacks :

- No way to correct/reconstruct non-parsable sentences (”early error detection”)
- not very intuitive

Earley Parsing (2)

Idea: on-line (i.e during parsing) binarization of the grammar

☞ doted rules and "Earley items"

doted rules: $X \to X_1 \ldots X_k \bullet X_{k+1} \ldots X_m$

with $X \to X_1 \ldots X_m$ a rule of the grammar

Earley item: one doted rule with one integer i

($0 \leq i \leq n$, n: size of the input string)

☞ the part before the dot (●) represents the subpart of the rule that derives a substring of the input string starting at position $i+1$

Example: $(VP \to V \bullet NP, 2)$ is an Earley item for input string

the	cat	ate	a	mouse
1	2	3	4	5

Earley Parsing (3)

Principle: Starting from all possible $(S \rightarrow \bullet X_1 \dots X_m, 0)$, parallel construction of all the dotted rules deriving (larger and larger) substrings of the input string, up to a point where the whole input sentence is derived

☞ construction of sets of items (E_j) such that:

$$(X \rightarrow \alpha \bullet \beta, i) \in E_j \iff$$

$$\exists \gamma, \delta : S \Rightarrow^* \gamma \, X \, \delta \quad \text{and} \quad \gamma \Rightarrow^* w_1 \dots w_i \quad \text{and} \quad \alpha \Rightarrow^* w_{i+1} \dots w_j$$

Example: in the former example $(VP \rightarrow V \bullet NP, 2) \in E_3$

The input string (length n) is syntactically correct (accepted) iff at least one $(S \rightarrow X_1 \dots X_m \bullet, 0)$ is in E_n

Earley Parsing (4)

❶ Initialization: construction of E_0

1. For each rule $S \rightarrow X_1 \dots X_m$ in the grammar: add $(S \rightarrow \bullet\, X_1 \dots X_m, 0)$ to E_0
2. For each $(X \rightarrow \bullet Y\ \beta, 0)$ in E_0 and every rule $Y \rightarrow \gamma$, add $(Y \rightarrow \bullet\, \gamma, 0)$ to E_0
3. Iterate (2) until convergence of E_0

Earley Parsing: Interpretation

❷ Iterations: building of derivations of $w_1 ... w_j$ (E_j)

1. **Linking with words**: Introduce word w_j whenever a derivation of $w_1 ... w_{j-1}$ can "eat" w_j (i.e. "*there is a ● before* w_j")

2. **Stepping in the derivation**: Whenever non-terminal X can derive a subsequence starting at w_{i+1} and if there exists one subderivation ending in w_i which can "eat" X, do it!

3. **Prediction** (of useful items): If at some place Y could be "eaten" by some rule, then introduce all the rules that might (later on) produce Y

Earley Parsing (next)

❷ Iterations: construction of the E_j sets ($1 \leq j \leq n$)

1. for all $(X \to \alpha \bullet w_j \, \beta, i)$ in E_{j-1}, add $(X \to \alpha \, w_j \bullet \beta, i)$ to E_j
2. For all $(X \to \gamma \bullet, i)$ of E_j, for all $(Y \to \alpha \bullet X \beta, k)$ of E_i,
   add $(Y \to \alpha \, X \bullet \beta, k)$ to E_j
3. For all $(Y \to \alpha \bullet X \beta, i)$ in E_j and for each rule $X \to \gamma$,
   add $(X \to \bullet \gamma, j)$ to E_j
4. Repeat to (2) while E_j keeps changing

Earley Parsing: Full Example

Example for "*I think*", and the grammar:

$S \rightarrow NP\ VP$

$NP \rightarrow Pron$

$NP \rightarrow Det\ N$

$VP \rightarrow V$

$VP \rightarrow V\ S$

$VP \rightarrow V\ NP$

$Pron \rightarrow$ I

$V \rightarrow$ think

E_0: $(S \rightarrow \bullet NP\ VP, 0)$ $(NP \rightarrow \bullet\ Pron, 0)$
$(NP \rightarrow \bullet\ Det\ N, 0)$ $(Pron \rightarrow \bullet\,\text{I}, 0)$

E_1: $(Pron \rightarrow \text{I}\,\bullet, 0)$ $(NP \rightarrow Pron\,\bullet, 0)$
$(S \rightarrow NP \bullet\ VP, 0)$ $(VP \rightarrow \bullet\,V, 1)$
$(VP \rightarrow \bullet\,V\ P, 1)$ $(VP \rightarrow \bullet\,V\ NP, 1)$
$(V \rightarrow \bullet\,\text{think}, 1)$

E_2: $(V \rightarrow \text{think}\,\bullet, 1)$ $(VP \rightarrow V\,\bullet, 1)$
$(VP \rightarrow V\ \bullet\ S, 1)$ $(VP \rightarrow V\ \bullet\ NP, 1)$
$(S \rightarrow NP\ VP\,\bullet, 0)$ $(S \rightarrow \bullet\,NP\ VP, 2)$
$(NP \rightarrow \bullet\,Pron, 2)$ $(NP \rightarrow \bullet\,Det\ N, 2)$
$(Pron \rightarrow \bullet\,\text{I}, 2)$

Link between CYK and Earley

$$(X \to \alpha \bullet \beta, i) \in E_j \quad \Longleftrightarrow \quad (X \to \alpha \bullet \beta) \in \text{cell}_{j-i,\, i+1}$$

	.... $(S \to NP\ VP \bullet, 0)$	
	$(S \to NP \bullet VP, 0)$ .... $(NP \to Pron \bullet, 0)$ $(Pron \to \text{I} \bullet, 0)$	$(VP \to V \bullet S, 1)$ $(VP \to V \bullet NP, 1)$ .... $(VP \to V \bullet, 1)$ $(V \to \text{think} \bullet, 1)$
$(S \to \bullet NP\ VP, 0)$ $(NP \to \bullet Pron, 0)$ $(NP \to \bullet Det\ N, 0)$ $(Pron \to \bullet \text{I}, 0)$	$(VP \to \bullet V, 1)$ $(VP \to \bullet V\ S, 1)$ $(VP \to \bullet V\ NP, 1)$ $(V \to \bullet \text{think}, 1)$	$(S \to \bullet NP\ VP, 2)$ $(NP \to \bullet N, 2)$ $(NP \to \bullet Det\ N, 2)$ $(Pron \to \bullet \text{I}, 2)$
	I	think

Bottom-up Chart Parsing

Idea: keep the best of both CYK and Earley

☞ on-line binarization ”à la” Earley (and even better) within a bottom-up CYK-like algorithm

Mainly:

- no need for indices in items: cell position is enough
- factorize (with respect to α) all the $X \to \alpha \bullet \beta$ ☞ $\alpha \bullet \ldots$

  This is possible when processing **bottom-up**
- replace all the $X \to \alpha \bullet$ simply by X
- supression of $X \to \bullet \alpha$

  This is possible when processing **bottom-up** (and without lookahead)

Bottom-up Chart Parsing: Example

S				
S		VP		
NP • ... NP			NP • ... NP	
Det • ... Det	N	V • ... V VP	Det • ... Det	N
The	crocodile	ate	the	cat

Bottom-up Chart Parsing (3)

More formally, a CYK algorithm in which:

If cell contents are denoted by $[\alpha \bullet ..., i, j]$ and $[X, i, j]$ respectively

Then initialization is $w_{ij} \Rightarrow [X, i, j]$ for $X \rightarrow w_{ij} \in \mathcal{R}$

and the completion phase becomes:

(association of two cells)

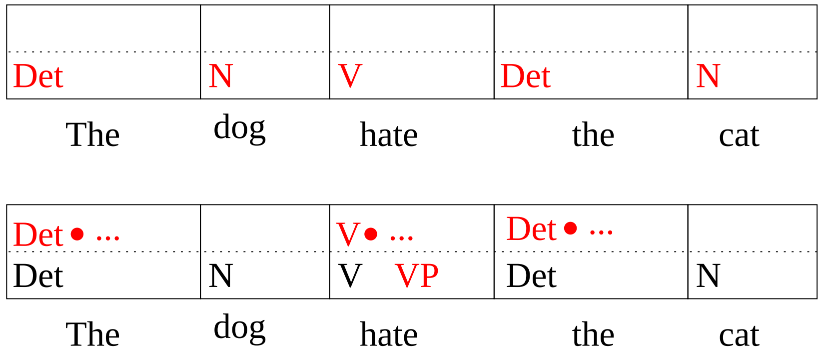
$$[\alpha \bullet ..., i, j] \oplus [X, k, j+i] \Rightarrow \begin{cases} [\alpha\, X \bullet ..., i+k, j] & \text{if } Y \rightarrow \alpha\, X\beta \in \mathcal{R} \\ [Y, i+k, j] & \text{if } Y \rightarrow \alpha\, X \in \mathcal{R} \end{cases}$$

("self-filling")

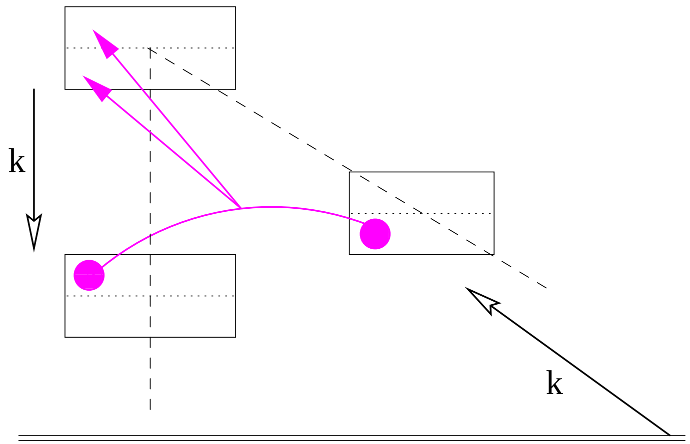
$$[X, i, j] \Rightarrow \begin{cases} [X \bullet ..., i, j] & \text{if } Y \rightarrow X\beta \in \mathcal{R} \\ [Y, i, j] & \text{if } Y \rightarrow X \in \mathcal{R} \end{cases}$$

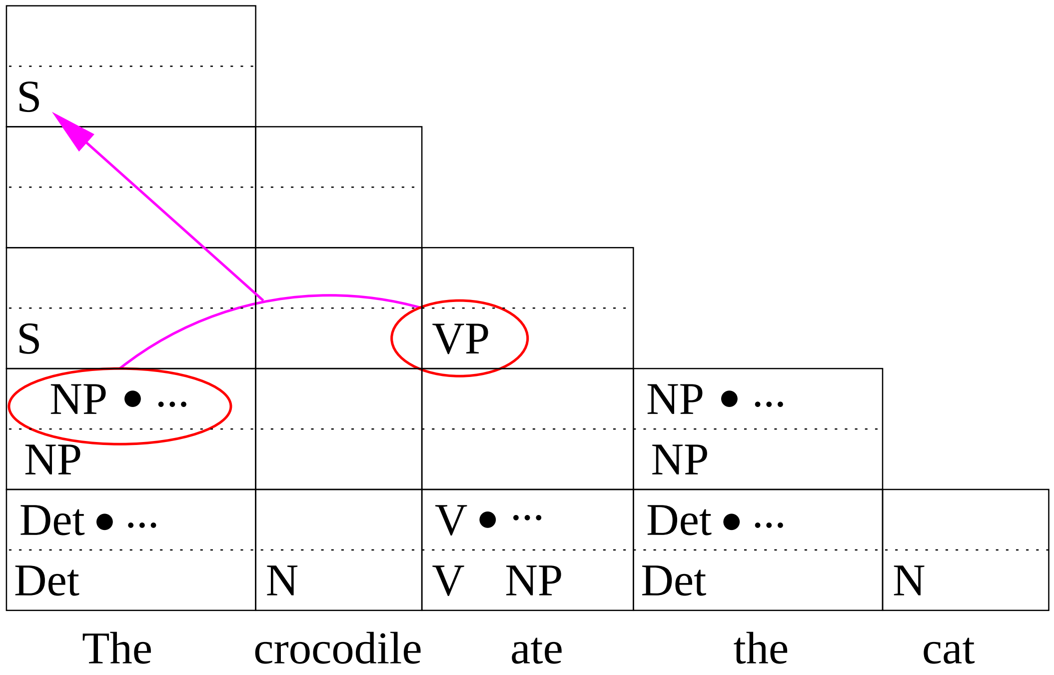
Bottom-up Chart Parsing: Example

Initialization:



Completion:




S
S
VP
NP • ...
NP
NP • ...
NP
Det • ...
Det
N
V • ...
V NP
Det • ...
Det
N
The
crocodile
ate
the
cat

Dealing with compounds

Example on how to deal with compouds during initialization phase:

N	
N V	N
credit	card

Complexity

still in $\mathcal{O}(n^3)$

What coefficient for n^3? (with respect to grammar parameters)

$$m(\mathcal{R}') \cdot |\mathcal{NT}| \cdot n^3$$

where $m(\mathcal{R}')$ the number of **internal** nodes of the trie of the right-hand sides of the **non-lexical** grammar rules

$\mathcal{NT}$: the set of non-terminals

$\mathcal{R}'$: the set of non-lexical grammar rules

Keypoints

- The way algorithms work (Earley items, linking, stepping, prediction, link CYK-Earley)
- worst-case complexity $\mathcal{O}(n^3)$
- Advantages and drawbacks of algorithms

References

[1] D. Jurafsky & J. H. Martin, *Speech and Language Processing*, pp. 377-385, Prentice Hall, 2000.

[2] R. Dale, H. Moisi, H. Somers, *Handbook of Natural Language Processing*, pp. 69-73, Dekker, 2000.