Parsing

Earley Parsing

Laura Kallmeyer, Magnus Roos
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2014

Overview

1. Idea
2. Algorithm
3. Tabulation
4. Parse trees
5. Lookaheads
6. Conclusion

Idea (1)

Goal: overcome problems with pure TD/BU approaches.

Earley's algorithm can be seen as a

- bottom-up parser with top-down control, i.e., a bottom-up parsing that does only reductions that can be top-down predicted from S, or a
- top-down parser with bottom-up recognition

Idea (2)

At each time of parsing, one production $A \to X_1 \dots X_k$ is considered such that

- some part $X_1 \dots X_i$ has already been bottom-up recognized (completed)
- while some part $X_{i+1} \dots X_k$ has been top-down predicted.

As in the left-corner chart parser, this situation can be characterized by a dotted production (sometimes called Earley item) $A \to X_1 \dots X_i \bullet X_{i+1} \dots X_k$.

Dotted productions are called active items. Productions of the form $A \to \alpha\bullet$ are called completed items.

Idea (3)

The Earley parser simulates a top-down left-to-right depth-first traversal of the parse tree while moving the dot such that for each node

- first, the dot is to its left (the node is predicted),
- then the dot traverses the tree below,
- then the dot is to its right (the subtree below the node is completed)

Each state of the parser can be characterized by a set of dotted productions $A \to X_1 \dots X_i \bullet X_{i+1} \dots X_k$. For each of these, one needs to keep track of the input part spanned by the completed part of the rhs, i.e., by $X_1 \dots X_i$.

Algorithm (1)

The items describing partial results of the parser contain a dotted production and the start and end index of the completed part of the rhs:

Item form: $[A \to \alpha \bullet \beta, i, j]$ with $A \to \alpha\beta \in P, 0 \leq i \leq j \leq n$.

Parsing starts with predicting all S-productions:

Axioms: $$\frac{}{[S \to \bullet\alpha, 0, 0]} \quad S \to \alpha \in P$$

Algorithm (2)

If the dot of an item is followed by a non-terminal symbol B, a new B-production can be predicted. The completed part of the new item (still empty) starts at the index where the completed part of the first item ends.

Predict: $$\frac{[A \to \alpha \bullet B\beta, i, j]}{[B \to \bullet\gamma, j, j]} \quad B \to \gamma \in P$$

If the dot of an item is followed by a terminal symbol a that is the next input symbol, then the dot can be moved over this terminal (the terminal is scanned). The end position of the completed part is incremented.

Scan: $$\frac{[A \to \alpha \bullet a\beta, i, j]}{[A \to \alpha a \bullet \beta, i, j+1]} \quad w_{j+1} = a$$

Algorithm (3)

If the dot of an item is followed by a non-terminal symbol B and if there is a second item with a dotted B-production and a fully completed rhs and if, furthermore, the completed part of the second item starts at the position where the completed part of the first ends, then the dot in the first can be moved over the B while changing the end index to the end index of the completed B-production.

Complete: $$\frac{[A \rightarrow \alpha \bullet B\beta, i, j], [B \rightarrow \gamma \bullet, j, k]}{[A \rightarrow \alpha B \bullet \beta, i, k]}$$

The parser is successfull if a completed S-production spanning the entire input can be deduced:

Goal items: $[S \rightarrow \alpha \bullet, 0, n]$ for some $S \rightarrow \alpha \in P$.

Algorithm (4)

Note that

- this algorithm can deal with ϵ-productions;
- loops and left-recursions are no problem since an active item is generated only once;
- the algorithm works for any type of CFG.

Algorithm (5)

Example: $S \to aB \mid bA$, $A \to aS \mid bAA \mid a$, $B \to bS \mid aBB \mid b$.

$w = abab$. Set of deduced items:

1. $[S \to \bullet aB, 0, 0]$ axiom
2. $[S \to \bullet bA, 0, 0]$ axiom
3. $[S \to a \bullet B, 0, 1]$ scan with 1.
4. $[B \to \bullet bS, 1, 1]$ predict with 3.
5. $[B \to \bullet b, 1, 1]$ predict with 3.
6. $[B \to \bullet aBB, 1, 1]$ predict with 3.
7. $[B \to b \bullet S, 1, 2]$ scan with 4.
8. $[B \to b\bullet, 1, 2]$ scan with 5.
9. $[S \to aB\bullet, 0, 2]$ complete with 3. and 8.

Algorithm (6)

10. $[S \to \bullet aB, 2, 2]$ predict with 7.
11. $[S \to \bullet bA, 2, 2]$ predict with 7.
12. $[S \to a \bullet B, 2, 3]$ scan with 10.
13. $[B \to \bullet bS, 3, 3]$ predict with 12.
14. $[B \to \bullet b, 3, 3]$ predict with 12.
15. $[B \to \bullet aBB, 3, 3]$ predict with 12.
16. $[B \to b \bullet S, 3, 4]$ scan with 13.
17. $[B \to b\bullet, 3, 4]$ scan with 14.
18. $[S \to \bullet aB, 4, 4]$ predict with 16.
19. $[S \to \bullet bA, 4, 4]$ predict with 16.
20. $[S \to aB\bullet, 2, 4]$ complete with 12. and 17.
21. $[B \to bS\bullet, 1, 4]$ complete with 7. and 20.
22. $[S \to aB\bullet, 0, 4]$ complete with 3. and 21.

Algorithm (7)

Soundness and completeness:

The following holds:

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

iff

$$S \stackrel{*}{\Rightarrow} w_1 \ldots w_i A\gamma \Rightarrow w_1 \ldots w_i \alpha\beta\gamma \stackrel{*}{\Rightarrow} w_1 \ldots w_i w_{i+1} \ldots w_j \beta\gamma$$

for some $\gamma \in (N \cup T^*)$.

The algorithm is in particular prefix-valid: if there is an item with end position j, then there is a word in the language with prefix $w_1 \ldots w_j$.

Algorithm (8)

In addition, one can use passive items $[A, i, j]$ with $A \in N$, $0 \leq i \leq j \leq n$.

Then, we need an additional *convert* rule, that converts a completed active item into a passive one:

Convert: $$\frac{[B \rightarrow \gamma \bullet, j, k]}{[B, j, k]}$$

The goal item is then $[S, 0, n]$.

Algorithm (9)

The *Complete* rule can use passive items now:

$$\text{Complete:} \quad \frac{[A \rightarrow \alpha \bullet B\beta, i, j], [B, j, k]}{[A \rightarrow \alpha B \bullet \beta, i, k]}$$

The advantage is that we obtain a higher degree of factorization: A B-subtree might have different analyses. *Complete* can use this B category now independent from the concrete analyses, i.e., there is only one single application of *Complete* for all of them.

Tabulation (1)

We can tabulate the dotted productions depending on the indices of the covered input.

I.e., we adopt a $(n+1) \times (n+1)$-chart C with

$A \to \alpha \bullet \beta \in C_{i,j}$ iff $[A \to \alpha \bullet \beta, i, j]$.

The chart is initialized with

$C_{0,0} := \{S \to \bullet\alpha \,|\, S \to \alpha \in P\}$ and

$C_{i,j} = \emptyset$ for all $i, j \in [0..n]$ with $i \neq 0$ or $j \neq 0$.

It can then be filled in the following way:

Tabulation (2)

Let us consider the version without passive items.

The chart is filled row by row:

for every end-of-span index k:

- we first compute all applications of *predict* and *complete* that yield new items with end-of-span index k;
- then, we compute all applications of *scan* which gives items with end-of-span index $k+1$.

Tabulation (3)

```
for all k ∈ [0..n]:
    do until chart does not change any more:
        for all j ∈ [0..k] and all p ∈ C_{j,k}:
            if p = A → α • Bβ
            then add B → •γ to C_{k,k} for all B → γ ∈ P        predict
            else if p = B → γ•
            then for all i ∈ [0..j]:
                if there is a A → α • Bβ ∈ C_{i,j}
                then add A → αB • β to C_{i,k}        complete
    for all j ∈ [0..k] and for all p ∈ C_{j,k}:
        if p = A → α • w_{k+1}β
        then add A → αw_{k+1} • β to C_{j,k+1}        scan
```

Tabulation (4)

Note that *predict* and *complete* do not increment the end of the spanned input, i.e., they add only elements to the fields $C_{\dots,k}$ (the k-th row of the chart).

Scan however adds elements to the $C_{\dots,k+1}$ (the $k+1$-th row).

This is why first, all possible *predict* and *complete* operations are performed to generate new chart entries in the k-th row. Then, *scan* is applied and one can move on to the next row $k+1$.

Since *predict* and *complete* are applied as often as possible, ϵ-productions and left recursion are no problem for this algorithm.

Tabulation (5)

Implementation:

Besides the chart, for every k, we keep an agenda A_k of those items from the chart that still need to be processed.

Initially, for $k = 0$, this agenda contains all $S \to \bullet\alpha$, the other agendas are empty.

We process the items in the agendas from $k = 0$ to $k = n$. For each k, we stop once the k-agenda is empty.

Tabulation (6)

- Items x of the form $A \to \alpha \bullet B\beta$ trigger a *predict* operation.

  The newly created items, if they are not yet in the chart, are added to chart and k-agenda.

  In addition, if ϵ-productions are allowed, x also triggers a *complete* where the chart is searched for a completed B-item ranging from k to k. The new items (if not in the chart yet) are added to the k-agenda and the chart.

  x is removed from the k-agenda.

Tabulation (8)

- Items x of the form $B \to \gamma\bullet$ trigger a *complete* operation where the chart is searched for corresponding items $A \to \alpha \bullet B\beta$.

  The newly created items are added to the chart and the k-agenda (if they are not yet in the chart), x is removed from the k-agenda.

- Items x of the form $A \to \alpha \bullet a\beta$ trigger a *scan* operation.

  The newly created items (if not yet in the chart) are added to the chart and the $k+1$-agenda, x is removed from the k-agenda.

Tabulation (9)

Example 1 (no ϵ-productions): $S \rightarrow ASB \mid c, A \rightarrow a, B \rightarrow b$.

Input $w = acb$

$A_0 = \{[S \rightarrow \bullet ASB, 0, 0], [S \rightarrow \bullet c, 0, 0]\}$

2				
1				
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$			
	0	1	2	3

$S \rightarrow \bullet ASB$ triggers a *predict*:

Tabulation (10)

$A_0 = \{[S \rightarrow \bullet c, 0, 0], [A \rightarrow \bullet a, 0, 0]\}$

2				
1				
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$S \rightarrow \bullet c$ triggers a scan that fails, $A \rightarrow \bullet a$ triggers a successful scan:

Tabulation (11)

$A_0 = \{\}, A_1 = \{[A \rightarrow a\bullet, 0, 1]\}$

2				
1	$A \rightarrow a\bullet$			
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$A \rightarrow a\bullet$ triggers a complete:

Tabulation (12)

$A_0 = \{\}, A_1 = \{[S \rightarrow A \bullet SB, 0, 1]\}$

2				
1	$S \rightarrow A \bullet SB$ $A \rightarrow a\bullet$			
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$S \rightarrow A \bullet SB$ triggers a predict:

Tabulation (13)

$A_0 = \{\}, A_1 = \{[S \to \bullet ASB, 1, 1], [S \to \bullet c, 1, 1]\}$

2				
1	$S \to A \bullet SB$ $A \to a \bullet$	$S \to \bullet ASB$ $S \to \bullet c$		
0	$S \to \bullet ASB$ $S \to \bullet c$ $A \to \bullet a$			
	0	1	2	3

$S \to \bullet ASB$ triggers a predict:

Tabulation (14)

$A_0 = \{\}, A_1 = \{[S \to \bullet c, 1, 1], [A \to \bullet a, 1, 1]\}$

2				
1	$S \to A \bullet SB$ $A \to a\bullet$	$S \to \bullet ASB$ $S \to \bullet c$ $A \to \bullet a$		
0	$S \to \bullet ASB$ $S \to \bullet c$ $A \to \bullet a$			
	0	1	2	3

$S \to \bullet c$ triggers a scan:

Tabulation (15)

$A_0 = \{\}, A_1 = \{[A \rightarrow \bullet a, 1, 1]\}, A_2 = \{[S \rightarrow c\bullet, 1, 2]\}$

2		$S \rightarrow c\bullet$		
1	$S \rightarrow A \bullet SB$ $A \rightarrow a\bullet$	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$		
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$A \rightarrow \bullet a$ triggers a scan that fails, then $S \rightarrow c\bullet$ triggers a complete:

Tabulation (16)

$A_0 = \{\}, A_1 = \{\}, A_2 = \{[S \rightarrow AS \bullet B, 0, 2]\}$

2	$S \rightarrow AS \bullet B$	$S \rightarrow c\bullet$		
1	$S \rightarrow A \bullet SB$ $A \rightarrow a\bullet$	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$		
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$S \rightarrow AS \bullet B$ triggers the prediction of $[B \rightarrow \bullet b, 2, 2]$, which triggers a successful scan:

Tabulation (17)

$A_0 = \{\}, A_1 = \{\}, A_2 = \{\}, A_3 = \{[B \rightarrow b\bullet, 2, 3]\}$

3			$B \rightarrow b\bullet$	
2	$S \rightarrow AS \bullet B$	$S \rightarrow c\bullet$	$B \rightarrow \bullet b$	
1	$S \rightarrow A \bullet SB$ $A \rightarrow a\bullet$	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$		
0	$S \rightarrow \bullet ASB$ $S \rightarrow \bullet c$ $A \rightarrow \bullet a$			
	0	1	2	3

$B \rightarrow b\bullet$ triggers a complete which leads to a goal item:

Tabulation (18)

$A_0 = \{\}, A_1 = \{\}, A_2 = \{\}, A_3 = \{\}$

3	$S \to ASB\bullet$		$B \to b\bullet$	
2	$S \to AS \bullet B$	$S \to c\bullet$	$B \to \bullet b$	
1	$S \to A \bullet SB$ $A \to a\bullet$	$S \to \bullet ASB$ $S \to \bullet c$ $A \to \bullet a$		
0	$S \to \bullet ASB$ $S \to \bullet c$ $A \to \bullet a$			
	0	1	2	3

Tabulation (19)

Example 2 (with ϵ-productions):

$S \rightarrow ABB, A \rightarrow a, B \rightarrow \epsilon.$

Input $w = a$

1	$A \rightarrow a\bullet$	
	$S \rightarrow A \bullet BB$	$B \rightarrow \bullet$
	$S \rightarrow AB \bullet B$	
	$S \rightarrow ABB\bullet$	
0	$S \rightarrow \bullet ABB$	
	$A \rightarrow \bullet a$	
	0	1

Tabulation (20)

Overview: TD/BU and mixed approaches:

	TD/BU	item form	chart parsing
Top-down	TD	$[\Gamma, i]$	no
Bottom-up	BU	$[\Gamma, i]$	no
CYK	BU	$[A, i, l]$	yes
Left corner	mixed	$[\Gamma_{compl}, \Gamma_{td}, \Gamma_{lhs}]$	no
		$[A \to \alpha \bullet \beta, i, l]$	yes
Earley	mixed	$[A \to \alpha \bullet \beta, i, j]$	yes

Parsing (1)

So far, we have a recognizer.

- One way to extend it to a parser it to read off the parse tree in a top-down way from the chart.
- Alternatively, in every completion step, we can record in the chart the way the new item can be obtained by adding pointers to its pair of antecedents. Then, for constructing the parse tree, we only need to follow the pointers.

Parsing (2)

First possibility (initial call `parse-tree(`S`,`0`,`n`)`):

`parse-tree(`X`,`i`,`j`)`

 `trees :=` $\emptyset$;

 `if` $X = w_j$ `and` $j = i + 1$ `then trees :=` $\{w_j\}$

 `else for all` $X \rightarrow X_1 \dots X_r \bullet \in C_{i,j}$

  `for all` $i_1, \dots, i_r$, $i \leq i_1 \leq \dots \leq i_{r-1} \leq i_r = j$

   `and all` $t_1, \dots t_r$ `with`

   $t_1 \in$ `parse-tree(`X_1`,` i`,` i_1`) and`

   $t_l \in$ `parse-tree (`X_l`,` i_{l-1}`,` i_l`) for` $1 < l \leq r$`:`

  `trees := trees` $\cup \{X(t_1, \dots, t_r)\}$;

 `output trees`

Parsing (3)

Second possibility:

We equip items with an additional set of pairs of pointers to other items in the item set.

Whenever an item $[A \rightarrow \alpha A \bullet \beta, i, k]$ is obtained in a complete operation from $[A \rightarrow \alpha \bullet A\beta, i, j]$ and $[A \rightarrow \gamma \bullet, j, k]$, we add a pair of pointers to the two antecedent items to the pointer set of the consequent item.

Obviously, items might have more than one element in their set if the grammar is ambiguous.

Parsing (4)

Example: $S \to AB, A \to Ac \,|\, a, B \to cB \,|\, b$. Input $w = acb$.

Item set (with list of pointer pairs):

1 $[S \to \bullet AB, 0, 0]$ 2 $[A \to \bullet Ac, 0, 0]$ 3 $[A \to \bullet a, 0, 0]$

4 $[A \to a\bullet, 0, 1]$

5 $[A \to A \bullet c, 0, 1], \{\langle 2, 4\rangle\}$ 6 $[S \to A \bullet B, 0, 1], \{\langle 1, 4\rangle\}$

7 $[B \to \bullet cB, 1, 1]$ 8 $[B \to \bullet b, 1, 1]$

9 $[A \to Ac\bullet, 0, 2]$ 10 $[B \to c \bullet B, 1, 2]$

11 $[A \to A \bullet c, 0, 2], \{\langle 2, 9\rangle\}$ 12 $[S \to A \bullet B, 0, 2], \{\langle 1, 9\rangle\}$

13 $[B \to \bullet cB, 2, 2]$ 14 $[B \to \bullet b, 2, 2]$ 15 $[B \to b\bullet, 2, 3]$

16 $[B \to cB\bullet, 1, 3], \{\langle 10, 15\rangle\}$

17 $[S \to AB\bullet, 0, 3], \{\langle 6, 16\rangle, \langle 12, 15\rangle\}$

Lookaheads

Two kinds of lookaheads: a prediction lookahead and a reduction lookahead:

Predict with lookahead:

$$\frac{[A \to \alpha \bullet B\beta, i, j]}{[B \to \bullet\gamma, j, j]} \quad B \to \gamma \in P, w_{j+1} \in First(\gamma) \text{ or } \epsilon \in First(\gamma)$$

Complete with lookahead:

$$\frac{[A \to \alpha \bullet B\beta, i, j], [B \to \gamma\bullet, j, k]}{[A \to \alpha B \bullet \beta, i, k]} \quad \begin{array}{l} w_{k+1} \in First(\beta) \\ \text{or } \epsilon \in First(\beta) \text{ and } w_{k+1} \in Follow(A) \end{array}$$

Instead of precomputing the *Follow* sets, one can compute the actual follows while predicting.

Conclusion

- Earley is a top-down restricted bottom-up parser.
- The three operations to compute new partial parsing results are predict, scan and complete.
- Earley is a chart parser.
- Earley can parse all CFGs.