Parsing beyond context-free grammar:

Simple RCG: Incremental Earley Parsing

Laura Kallmeyer, Wolfgang Maier

Sommersemester 2009

Overview

1. Incremental Earley Parsing
   (a) Deduction Rules
   (b) Filters
2. Treebank Grammar Extraction
3. Demo

Incremental Earley Parsing

- Modification of Incremental CYK (MCFG) for ordered simple RCG
- Implemented in the TuLiPA system (`http://sourcesup.cru.fr/tulipa`)
- Strategy:
  - Process LHS arguments incrementally, starting from an S-rule
  - Whenever we reach a variable, move into rule of correponding rhs predicate (**predict** or **resume**).
  - Whenever we reach the end of an argument, **suspend** the rule and move into calling parent rule.
  - Whenever we reach the end of the last argument **convert** item into a passive one and **complete** parent item.

Incremental Earley Parsing: Deduction Rules (1)

- **Passive items**: $[A, \vec{\rho}]$ where A is a predicate of arity k and $\vec{\rho}$ is a range vector of arity k
- **Active items**:

$$[A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \ldots A_m(\vec{\phi_m}), pos, \langle i, j \rangle, \vec{\rho}]$$

where

- $A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \ldots A_m(\vec{\phi_m}) \in P$;
- $pos \in \{0, \ldots, n\}$: We have reached input position pos;
- $\langle i, j \rangle \in \mathbb{N}^2$: We have reached the jth element of ith argument (dot position);
- $\vec{\rho}$ is a range vector containing variable and terminal bindings. All elements are initialized to "?", an initialized vector is called $\vec{\rho}_{init}$.

Incremental Earley Parsing: Deduction Rules (2)

- Notation:
  - $\vec{\rho}(X)$: range bound to variable X.
  - $\vec{\rho}(\langle i, j\rangle)$: range bound to jth element of ith argument on LHS.
- Applying a range vector $\vec{\rho}$ containing variable bindings for given rule c to the argument vector of the lefthand side of c means mapping the ith element in the arguments to $\vec{\rho}(i)$ and concatenating adjacent ranges. The result is defined iff every argument is thereby mapped to a range.

Incremental Earley Parsing: Initialize, Goal item

Initialize: $\dfrac{}{[S(\vec{\phi}) \rightarrow \vec{\Phi}, 0, \langle 1, 0\rangle, \vec{\rho}_{init}]}\ S(\vec{\phi}) \rightarrow \vec{\Phi} \in P$

Goal Item: $[S(\vec{\phi}) \rightarrow \vec{\Phi}, n, \langle 1, j\rangle, \psi]$ with $|\vec{\phi}(1)| = j$ (i.e., dot at the end of lhs argument).

Incremental Earley Parsing: Scan

If next symbol after dot is next terminal in input, scan it.

Scan: $\dfrac{[A(\vec{\phi}) \rightarrow \vec{\Phi}, pos, \langle i, j\rangle, \vec{\rho}]}{[A(\vec{\phi}) \rightarrow \vec{\Phi}, pos+1, \langle i, j+1\rangle, \vec{\rho}']}\ \vec{\phi}(i, j+1) = w_{pos+1}$

where $\vec{\rho}'$ is $\vec{\rho}$ updated with $\vec{\rho}(\langle i, j+1\rangle) = \langle pos, pos+1\rangle$.

Incremental Earley Parsing: Predict

Whenever our dot is left of a variable that is the first argument of some rhs predicate B, we predict new B-rules:

Predict: $\dfrac{[A(\vec{\phi}) \rightarrow \ldots B(X, \ldots) \ldots, pos, \langle i, j\rangle, \vec{\rho}_A]}{[B(\vec{\psi}) \rightarrow \vec{\Psi}, pos, \langle 1, 0\rangle, \vec{\rho}_{init}]}$

where $\vec{\phi}(i, j+1) = X, B(\vec{\psi}) \rightarrow \vec{\Psi} \in P$

Incremental Earley Parsing: Suspend

Suspend:

$$\frac{[B(\vec{\psi}) \to \vec{\Psi}, pos', \langle i,j\rangle, \vec{\rho}_B], [A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle k,l\rangle, \vec{\rho}_A]}{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos', \langle k,l+1\rangle, \vec{\rho}]}$$

where

- the dot in the antecedent A-item precedes the variable $\vec{\xi}(i)$,
- $|\vec{\psi}(i)| = j$ (ith argument has length j, i.e., is completely processed),
- $|\vec{\psi}| < i$ (ith argument is not the last argument of B),
- $\vec{\rho}_B(\vec{\psi}(i)) = \langle pos, pos'\rangle$
- and for all $1 \leq m < i$: $\vec{\rho}_B(\vec{\psi}(m)) = \vec{\rho}_A(\vec{\xi}(m))$.

$\vec{\rho}$ is $\vec{\rho}_A$ updated with $\vec{\rho}_A(\vec{\xi}(i)) = \langle pos, pos'\rangle$.

Incremental Earley Parsing: Convert

Whenever we arrive at the end of the last argument, we convert the item into a passive one:

Convert:

$$\frac{[B(\vec{\psi}) \to \vec{\Psi}, pos, \langle i,j\rangle, \vec{\rho}_B]}{[B, \rho]} \quad |\vec{\psi}(i)| = j, |\vec{\psi}| = i, \vec{\rho}_B(\vec{\psi}) = \rho$$

Incremental Earley Parsing: Complete

Whenever we have a passive B item we can use it to move the dot over the variable of the last argument of B in a parent A-rule:

Complete: $$\frac{[B, \vec{\rho}_B], [A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle k,l\rangle, \vec{\rho}_A]}{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos', \langle k,l+1\rangle, \vec{\rho}]}$$ where

- the dot in the antecedent A-item precedes the variable $\vec{\xi}(|\vec{\rho}_B|)$,
- the last range in $\vec{\rho}_B$ is $\langle pos, pos'\rangle$,
- and for all $1 \leq m < |\vec{\rho}_B|$: $\vec{\rho}_B(m) = \vec{\rho}_A(\vec{\xi}(m))$.

$\vec{\rho}$ is $\vec{\rho}_A$ updated with $\vec{\rho}_A(\vec{\xi}(|\vec{\rho}_B|)) = \langle pos, pos'\rangle$.

Incremental Earley Parsing: Resume

Whenever we are left of a variable that is not the first argument of one of the rhs predicates, we resume the rule of the rhs predicate.

Resume: $$\frac{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle i,j\rangle, \vec{\rho}_A], \quad [B(\vec{\psi}) \to \vec{\Psi}, pos', \langle k-1,l\rangle, \vec{\rho}_B]}{[B(\vec{\psi}) \to \vec{\Psi}, pos, \langle k,0\rangle, \vec{\rho}_B]}$$

where

- $\vec{\phi}(i, j+1) = \vec{\xi}(k), k > 1$ (the next element is a variable that is the kth element in $\vec{\xi}$, i.e., the kth argument of B),
- $|\vec{\psi}(k-1)| = l$, and
- $\vec{\rho}_A(\vec{\xi}(m)) = \vec{\rho}_B(\vec{\psi}(m))$ for all $1 \leq m \leq k-1$.

Incremental Earley Parsing: Filters

- Filters can be applied to decrease the number of items in the chart
- A filter is an additional condition on the form of items.
- E.g., in a ε-free grammar, the number of variables in the part of the lefthand side arguments of a rule that has not been processed yet must be lower or equal to the length of the remaining input.

Incremental Earley Parsing: Remaining Input Length Filter

- In ε-free grammars each variable must cover at least one input symbol.
- i input symbols left implies no prediction of a clause with more than i variables or terminals on LHS since no instantiation is possible
- Condition on active items, can be applied with predict, resume, suspend and complete

An item $[A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \dots A_m(\vec{\phi_m}), pos, \langle i, j\rangle, \vec{\rho}]$ satisfies the **length filter** iff

$$(n - pos) \geq (|\vec{\phi}(i)| - j) + \Sigma_{k=i+1}^{dim(A)} |\vec{\phi}(k)|$$

Incremental Earley Parsing: Preterminal Filter (1)

- Check for the presence of (pre)terminals in the predicted part of a clause in the remaining input, and
- check that terminals appear in the predicted order and that distance between two of them is at least the number of variables/terminals in between.

continued...

Incremental Earley Parsing: Preterminal Filter (2)

In other words, an active item $[A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \dots A_m(\vec{\phi_m}), pos, \langle i, j\rangle, \vec{\rho}]$ satisfies the **preterminal filter** iff we can find an injective mapping $f_T : \text{Term} = \{\langle k, l\rangle \mid \vec{\phi}(k, l) \in T$ and either $k > i$ or $(k = i$ and $l > j)\} \rightarrow \{pos + 1, \dots, n\}$ such that

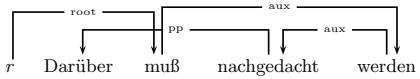
1. $w_{f_T(\langle k,l\rangle)} = \vec{\phi}(k, l)$ for all $\langle k, l\rangle \in Term$;
2. for all $\langle k_1, l_1\rangle, \langle k_2, l_2\rangle \in Term$ with $k_1 = k_2$ and $l_1 < l_2$: $f_T(\langle k_2, l_2\rangle) \geq f_T(\langle k_1, l_1\rangle) + (l_2 - l_1)$;
3. for all $\langle k_1, l_1\rangle, \langle k_2, l_2\rangle \in Term$ with $k_1 < k_2$: $f_T(\langle k_2, l_2\rangle) \geq f_T(\langle k_1, l_1\rangle) + (|\vec{\phi}(k_1)| - l_1) + \Sigma_{k=k_1+1}^{k_2-1} |\vec{\phi}(k)| + l_2$.

Treebank Grammar Extraction

- (P)CFGs can only be extracted from projective dependency trees (constituent trees without crossing branches)
- Dependency treebanks generally contain non-projective dependencies (e.g., PDT); Constituent treebanks generally do not contain crossing branches annotation, however there are counterexamples: TIGER, NeGra, BulTreebank.
- LCFRS/simple RCG can handle non-projective dependencies (crossing branches) as well, extraction algorithms can deliver ordered ε-free simple RCGs without useless rules.

Treebank Grammar Extraction: Example

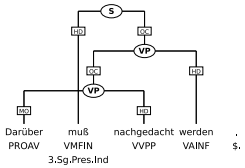
Example: A dependency tree and the set of obtained clauses.



$$
\begin{aligned}
\text{pp(Darüber)} &\rightarrow \varepsilon \\
\text{root}(X_1\text{muß}X_3) &\rightarrow \text{aux}(X_1,X_3) \\
\text{aux}(X_1,\text{nachgedacht}) &\rightarrow \text{pp}(X_1) \\
\text{aux}(X_1, X_2\text{werden}) &\rightarrow \text{aux}(X_1,X_2)
\end{aligned}
$$

Treebank Grammar Extraction: Example

Example: A constituent tree and the set of obtained clauses.



$$
\begin{aligned}
\text{PROAV1(Darüber)} &\rightarrow \varepsilon \\
\text{VMFIN1(muß)} &\rightarrow \varepsilon \\
\text{VVPP1(nachgedacht)} &\rightarrow \varepsilon \\
\text{VAINF1(werden)} &\rightarrow \varepsilon \\
\text{S1}(X_1X_2X_3) &\rightarrow \text{VP2}(X_1,X_3)\ \text{VMFIN1}(X_2) \\
\text{VP2}(X_1,X_2X_3) &\rightarrow \text{VP2}(X_1,X_2)\ \text{VAINF1}(X_3) \\
\text{VP2}(X_1,X_2) &\rightarrow \text{PROAV1}(X_1)\ \text{VVPP1}(X_2)
\end{aligned}
$$

Demo

Conclusion

- Incremental Earley Algorithm for simple RCG takes advantage of variable ordering
- Filters can drastically reduce the number of items in the chart
- Grammar extraction algorithms extract simple RCG for probabilistic parsing