Chart parsing with non-atomic categories

L545
Spring 2016
(With thanks to Detmar Meurers)

Changes to the chart representation

Each state will be extended to include the LHS feature structure (FS), which can get augmented as it goes along
- i.e., Add a feature structure (in DAG form) to each state
  - So, $S \rightarrow \bullet \text{NP VP, [0,0]}$
  - Becomes $S \rightarrow \bullet \text{NP VP, [0,0], FS}$

The predictor, scanner, and completer have to pass the FS, so all three operations have to be altered.

Earley parser with atomic categories

Prediction:
for each $[A \rightarrow \alpha, B \beta]$ in chart
for each $B' \rightarrow \gamma$ in rules
add $[[r(B \rightarrow \gamma \bullet)]$ with $\sigma = \text{mgu}(B, B')$ to chart

Scanning:
let $w_1 \ldots w_n$ be the input string
for each $[A \rightarrow \alpha, B \beta]$ in chart
add $[A \rightarrow \alpha w_j \beta]$ to chart

Completion (fundamental rule of chart parsing):
for each $[A \rightarrow \alpha, B \beta]$ and $[B \rightarrow \gamma \bullet]$ in chart
add $[A \rightarrow \alpha B \beta]$ to chart

Earley parser with unification

Prediction:
for each $[A \rightarrow \alpha, B \beta]$ in chart
for each $B' \rightarrow \gamma$ in rules
add $[[r(B \rightarrow \gamma \bullet)]$ with $\sigma = \text{mgu}(B, B')$ to chart

The predictor takes the specification of $B$ (i.e., FS) and finds the most general unifier (mgu) of $B$ with $B'$
- If $B$ & $B'$ do not unify, the rule for $B'$ is not added to the chart
- Initially (i.e., at position 0), all that happens is that a dotted rule with a FS is added to the chart

By utilizing unification as we parse, we can eliminate parses that don’t work in the end
- e.g., eliminate NPs that don’t match in agreement features with their VPs as we parse, instead of as a filter

Changes to the chart representation

Prediction:
for each $[A \rightarrow \alpha, B \beta]$ in chart
for each $B' \rightarrow \gamma$ in rules
add $[[r(B \rightarrow \gamma \bullet)]$ with $\sigma = \text{mgu}(B, B')$ to chart
Completion

Completion (fundamental rule of chart parsing):

for each $[A \rightarrow \alpha \bullet B \beta]$ and $[B' \rightarrow \gamma \bullet]$, in chart

add $[c(A \rightarrow \alpha \bullet B \beta)]$ with $c = \text{mgu}(B, B')$ to chart

Again, a step of unification is added.

- $B$ and $B'$ must unify in order for the dot to move
- The resulting FS is added to the chart

The subsumption problem (based on Covington 1994)

- $S \rightarrow \text{NP VP}$
- $\text{NP} \rightarrow \text{Det N}$
- $\text{VP} \rightarrow V'(0)$
- $V'(X) \rightarrow V(X)$
- $V'(X) \rightarrow \text{Adv} V(X)$
- $\text{Comps}(1) \rightarrow \text{NP}$
- $\text{Comps}(2) \rightarrow \text{NP NP}$
- $\text{Det} \rightarrow \text{the}$
- $\text{N} \rightarrow \text{dog}$
- $\text{N} \rightarrow \text{cat}$
- $\text{Adv} \rightarrow \text{often}$
- $V(0) \rightarrow \text{sings}$
- $V(1) \rightarrow \text{chases}$
- $V(2) \rightarrow \text{gives}$

Using subsumption to check the chart

Subsumption check: Do not add a state to the chart if an equivalent or more general state is already there.

- In trying to add a singular determiner state at $[x, y]$, if the chart already has a determiner state at $[x, y]$ unspecified for number, do not add it
- Without a subsumption restriction, we could add two states at $[x, y]$, one expecting to see a singular determiner, the other just a determiner.
  - On seeing a singular determiner, the parser advances the dot on both rules, creating two edges (since singular unifies with singular and with unspecified).
  - As a result, we would get duplicate edges.
- With subsumption, if either a singular or plural determiner is encountered, we advance the dot, creating only one edge (singular or plural) at $[x, y]$

Checking for subsumption

Case 1

Let's define a function `subsumes_chk` which takes 2 arguments: more general item & more specific item

No variables:

- `subsumes_chk(V'(1), V'(1))`, → yes
- `subsumes_chk(V'(1), V'(2))`, → no

Compound terms without variables are either identical or different, i.e., here: subsumption = unification
Checking for subsumption

Case 2

Variables only in more general term:
- subsumes_chk(V'(X),V'(1)) → yes
- subsumes_chk(foo(X,X),foo(1,1)) → yes
- subsumes_chk(foo(X),foo(1,2)) → no

Succeeds if a consistent variable assignment exists, i.e., here: subsumption = unification

The restriction problem

Shieber et al 1995: Grammar accepting ab^n with N being instantiated to the successor representation of n.

\[
\text{start} \rightarrow r(0,N) \\
r(X,N) \rightarrow r(s(X),N) b \\
r(N,N) \rightarrow a
\]

Prediction step with unification will loop:

1. \( \text{o[start} \rightarrow o_r(0,N)] \)
2. \( \text{o[r(0,N) in 1]} \)
3. \( \text{o[r(s(0),N) in 2]} \)
4. \( \text{o[r(s(s(0)),N) in 3]} \)
5. \( \text{o[r(s(s(s(0))),N) in 3]} \)

Using restriction to prevent prediction loops

- Prediction terminates for grammars with atomic categories, since a new item is only added to the chart if not already there and there is a finite number of atomic categories.
- Moving beyond atomic categories, there can be an infinite number of non-atomic categories.
- Prediction loop on left-recursive rules can be problem again.
- Solution: restrict number of predicted categories to finitely many cases

Prediction with restriction

for each \([A \rightarrow a \bullet B \beta]\) in chart
for each \(B' \rightarrow \gamma\) in rules

add \(\text{o[σ[B' \rightarrow \gamma]]}\) with \(σ = \text{restriction}(\text{mgu}(B,B'))\) to chart

restriction(\(\text{mgu}(B,B'))\) can be any operation reducing the number of possible substitutions:
- elimination of terms that are known to grow indefinitely
- use of only selected terms known not to grow indefinitely

This is sound since prediction only creates a hypothesis to be completed!

Example

Grammar:

\[
\text{start} \rightarrow r(0,N) \\
r(X,N) \rightarrow r(s(X),N) b \\
r(N,N) \rightarrow a
\]

Parsing using a restrictor that replaces every term deeper than 2 with a variable:

1. \( \text{o[start} \rightarrow o_r(0,N)] \)
2. \( \text{o[r(0,N) in 1]} \)
3. \( \text{o[r(s(0),N) in 2]} \)
4. \( \text{o[r(s(s(A)),N) in 3]} \)
5. \( \text{o[r(s(s(A)),N) in 4]} \) = edge 4

Variables in both terms:
- subsumes_chk(vbar(foo(X),foo(Y))), → yes
- subsumes_chk(vbar(foo(X),vbar(foo(1,Y))), → yes
- subsumes_chk(vbar(foo(1,2)),vbar(foo(1,Y))). → no
- Succeeds if terms can be unified without further instantiating more specific term; in other words:
  - Unification should not require a particular instantiation of a variable in the more specific term.
  - Idea: Identify each variable in more specific term with a unique, variable-free term; then subsumption = unification.