Natural Language Processing

Lecture 13

Today's Outline

HW1

HW2

Exam

Review: Earley

Features and Unification

Unification and Parsing

Subcategorization

Review

Reuse: despite ambiguity and backtracking there are invariants to be taken advantage of to reduce inefficiency.

The Earley algorithm fills a table (the chart) with solutions to sub-problems that:

- does not do repeated work
- does top-down search with bottom-up filtering (sort of)
- solves the left-recursion problem
- solves an exponential problem in less time

Features and Unification

Recall that there were things that we knew that CFGs could not handle very well.

In particular, agreement and subcategorization could only be handled in a very awkward way.

Agreement

What was the problem with the following rules?

S $\rightarrow$ NP VP

NP $\rightarrow$ Det Nominal

Agreement

What was the problem with the following rules?

S $\rightarrow$ NP VP

NP $\rightarrow$ Det Nominal

Answer: Since they don't enforce number and person agreement constraints, they overgenerate and allow the following:

$\star$ They sleeps.

$\star$ This dogs

Subcategorization

What was the problem with the following rules?

VP $\rightarrow$ V

VP $\rightarrow$ V NP

Subcategorization

What was the problem with the following rules?

VP $\rightarrow$ V

VP $\rightarrow$ V NP

Answer: Since they don't enforce subcategorization constraints, they allow the following:

$\star$ He took.

$\star$ He slept the cot.

$\star$ She disappeared the elephant.

3sgNP

One way to handle these phenomena in a strictly context-free approach is to encode the constraints into the non-terminal categories and then into the rules. As in the following:

- S $\rightarrow$ SgS
- S $\rightarrow$ PlS
- SgS $\rightarrow$ SgNP SgVP
- SgNP $\rightarrow$ SgDet SgNom

and

- IntransVP $\rightarrow$ IntransVerb
- TransVP $\rightarrow$ TransVerb NP

Features

An alternative is to rethink the terminal and non-terminals as complex objects with associated properties (called features) that can be manipulated. This would allow us to code rules as the following:

$S \rightarrow NP\ VP$

Only if the number of the NP is equal to the number of the VP.

This allows us to have the best of both worlds.

Features and Feature Structures

We can encode these properties by associating what are called Feature Structures with grammatical constituents.

Features structures are sets of feature-value pairs where:

- the features are atomic symbols and
- the values are either atomic symbols or feature structures.

$$\begin{bmatrix} feature_1 & value_1 \\ feature_2 & value_2 \\ \ldots & \\ feature_n & value_n \end{bmatrix}$$

Example Feature Structures

$$\begin{bmatrix} number & sg \end{bmatrix}$$

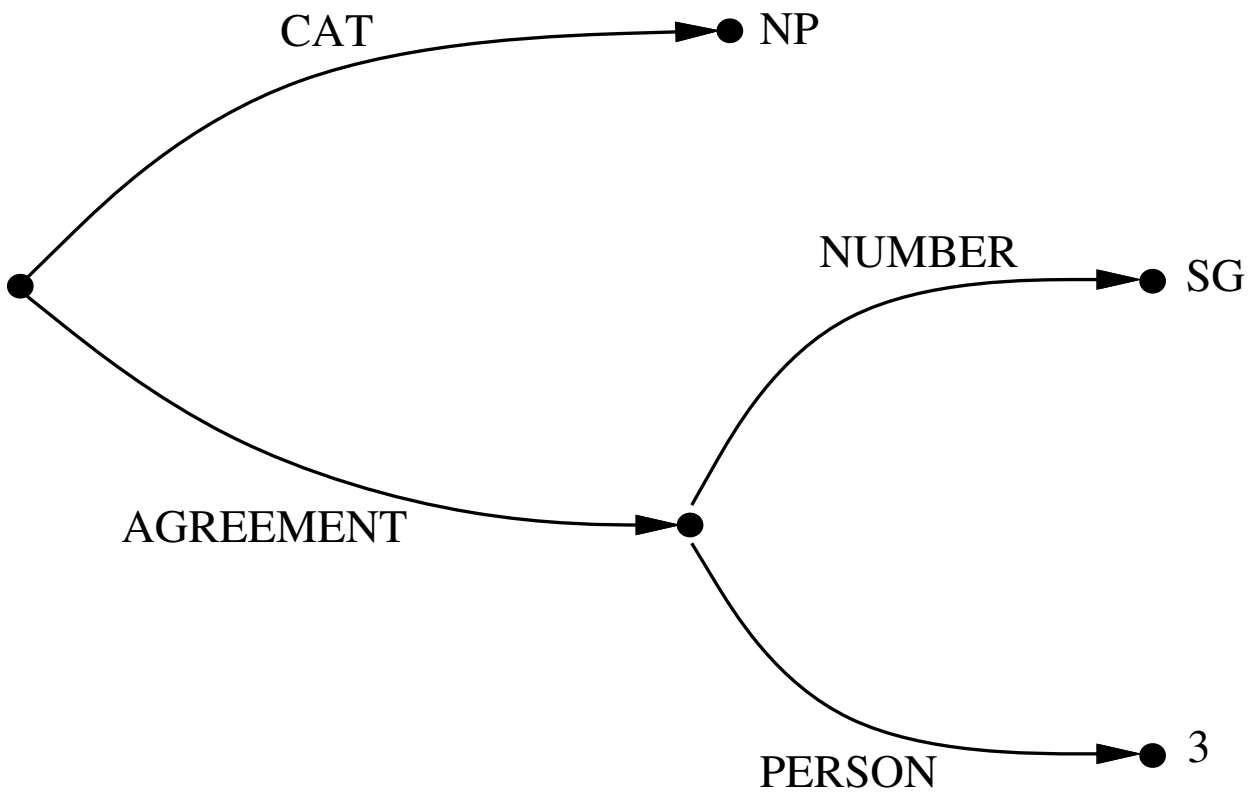
$$\begin{bmatrix} number & sg \\ person & 3 \end{bmatrix}$$

$$\begin{bmatrix} cat & np \\ number & sg \\ person & 3 \end{bmatrix}$$

Bundles of Features

$$\begin{bmatrix} cat & np \\ agreement & \begin{bmatrix} number & sg \\ person & 3 \end{bmatrix} \end{bmatrix}$$

Feature Structures as DAGs

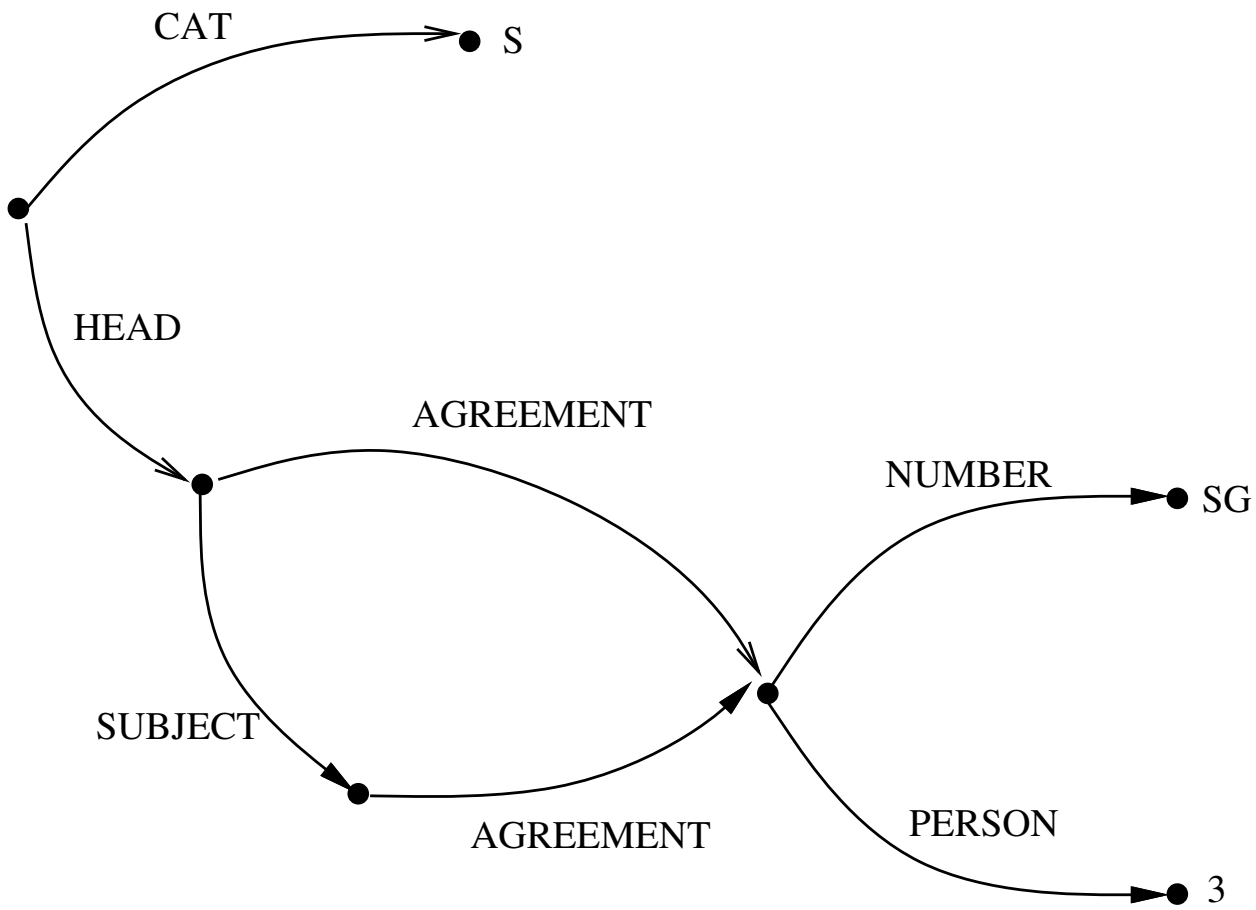


Reentrant Structures

We'll allow multiple features in a feature structure to share the same values. By this, we mean that they share the same structures not just that they have the same value.

$$\begin{bmatrix} cat & s \\ head & \begin{bmatrix} agreement & 1 \begin{bmatrix} number & sg \\ person & 3 \end{bmatrix} \\ subject & \begin{bmatrix} agreement & 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Reentrant DAGs



Feature Paths

It will also be useful to talk about paths through feature structures. As in the paths

⟨HEAD AGREEMENT NUMBER⟩

⟨HEAD SUBJECT AGREEMENT NUMBER⟩

$$\begin{bmatrix} cat & s \\ head & \begin{bmatrix} agreement & 1 \begin{bmatrix} number & sg \\ person & 3 \end{bmatrix} \\ subject & \begin{bmatrix} agreement & 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The Unification Operation

So what do we want to do with these things...

- check the compatibility of two structures
- merge the information in two structures

We can do both with an operation called Unification.

The Unification Operation

We say that two feature structures can be unified if the component features that make them up are compatible.

[number sg] $\sqcup$ [number sg] = [number sg]

[number sg] $\sqcup$ [number pl] fails!

The Unification Operation

[number sg] $\sqcup$ [number []] = [number sg]

[number sg] $\sqcup$ [person 3] = $\begin{bmatrix} number & sg \\ person & 3 \end{bmatrix}$

The Unification Operation

$$\begin{bmatrix} agreement & 1\begin{bmatrix} number & sg \\ person & 3 \end{bmatrix} \\ subject & [agreement 1] \end{bmatrix}$$

$$\sqcup$$

$$\begin{bmatrix} agreement & \begin{bmatrix} number & sg \\ person & 3 \end{bmatrix} \\ subject & \begin{bmatrix} agreement & \begin{bmatrix} number & pl \\ person & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fails!

Features, Unification and Grammars

We'll incorporate all this into our grammars in two ways.

- we'll assume that constituents are objects which have feature-structures associated with them
- we'll associate sets of unification constraints with grammar rules that must be satisfied for the rule to be satisfied.

Unification Constraints

$\beta_0 \rightarrow \beta_1 \ldots \beta_n$

{ set of constraints }

$\langle \beta_i$ feature path $\rangle$ = atomic value

$\langle \beta_i$ feature path $\rangle = \langle \beta_j$ feature path $\rangle$

Agreement

NP $\rightarrow$ Det Nominal

$\langle$ Det AGREEMENT $\rangle = \langle$ Nominal AGREEMENT $\rangle$

$\langle$ NP AGREEMENT $\rangle = \langle$ Nominal AGREEMENT $\rangle$

Noun $\rightarrow$ flight

$\langle$ Noun AGREEMENT NUMBER $\rangle$ = SG

Noun $\rightarrow$ flights

$\langle$ Noun AGREEMENT NUMBER $\rangle$ = PL

Nominal $\rightarrow$ Noun

$\langle$ Nominal AGREEMENT $\rangle = \langle$ Noun AGREEMENT $\rangle$

Det $\rightarrow$ this

$\langle$ Det AGREEMENT NUMBER $\rangle$ = SG

Unification and Parsing

OK, let's assume we've augmented our grammar with sets of path-like unification constraints.

What changes do we need to make to a parser to make use of them?

- building feature structures and associate them with subtree
- unifying feature structures as subtrees are created
- blocking ill-formed constituents

Unification and Earley Parsing

With respect to an Earley-style parser...

- building feature structures (represented as DAGs) and associate them with states in the chart
- unifying feature structures as states are advanced in the chart
- block ill-formed states from entering the chart

Building Feature Structures

S $\rightarrow$ NP VP

$\langle$ NP HEAD AGREEMENT $\rangle = \langle$ VP HEAD AGREEMENT $\rangle$

$\langle$ S HEAD $\rangle = \langle$ VP HEAD $\rangle$

corresponds to

$$\begin{bmatrix} s & [head 1] & \\ np & [head & [agreement 2]] \\ vp & [head & 1[agreement 2]] \end{bmatrix}$$

Augmenting States with DAGs

We just add a new field to the representation of the states.

S $\rightarrow$ $\cdot$ NP VP, [0,0], [], Dag

Unifying States and Blocking

We want to unify the DAGs of existing states as they are combined as specified by the grammatical constraints - in the COMPLETER.

```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE((γ → • S, [0,0], dag_γ), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
            NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
            NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)

procedure PREDICTOR((A → α • B β, [i, j], dag_A))
    for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE((B → • γ, [j, j], dag_B), chart[j])
    end

procedure SCANNER((A → α • B β, [i, j], dag_A))
    if B ⊂ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE((B → word[j], [j, j + 1], dag_B), chart[j+1])

procedure COMPLETER((B → γ •, [j,k], dag_B))
    for each (A → α • B β, [i, j], dag_A) in chart[j] do
      if new-dag ← UNIFY-STATES(dag_B, dag_A, B) ≠ Fails!
        ENQUEUE((A → α B • β, [i,k], new − dag), chart[k])
    end

procedure UNIFY-STATES(dag1, dag2, cat)
  dag1-cp ← COPYDAG(dag1)
  dag2-cp ← COPYDAG(dag2)
  UNIFY(FOLLOW-PATH(cat, dag1-cp), FOLLOW-PATH(cat, dag2-cp))

procedure ENQUEUE(state, chart-entry)
    if state is not subsumed by a state in chart-entry then
        PUSH(state, chart-entry)
    end
```

Example

NP $\rightarrow$ Det $\cdot$ Nominal[0,1], [S$_{Det}$], Dag1

where Dag1 equals

$$\begin{bmatrix} np & [head 1] & & \\ det & [head & [agreement 2 & [number sg]]] \\ nominal & [head & 1[agreement 2]] & \end{bmatrix}$$

Example

Nominal $\rightarrow$ Noun $\cdot$, [1,2], [S$_{Noun}$], Dag2

where Dag2 equals

$$\begin{bmatrix} nominal & [head 1] & \\ noun & [head & 1[agreement & [number sg]]] \end{bmatrix}$$

and Dag1 as before

$$\begin{bmatrix} np & [head 1] & & \\ det & [head & [agreement 2 & [number sg]]] \\ nominal & [head & 1[agreement 2]] & \end{bmatrix}$$

Copying

Why the need for all the copying in the procedure UNIFY-STATES?

Subcategorization Example

Some techniques

- atomic subcat symbols
- encoding subcat lists as feature structures

Subcat Symbols

VP $\rightarrow$ Verb

$\langle$ VP HEAD $\rangle = \langle$ Verb HEAD $\rangle$

$\langle$ VP HEAD SUBCAT $\rangle$ = INTRANS

VP $\rightarrow$ Verb NP

$\langle$ VP HEAD $\rangle = \langle$ Verb HEAD $\rangle$

$\langle$ VP HEAD SUBCAT $\rangle$ = TRANS

VP $\rightarrow$ Verb NP NP

$\langle$ VP HEAD $\rangle = \langle$ Verb HEAD $\rangle$

$\langle$ VP HEAD SUBCAT $\rangle$ = DITRANS

Subcat Symbols

Verb $\rightarrow$ slept

$\langle$ Verb HEAD SUBCAT $\rangle$ = INTRANS

Verb $\rightarrow$ served

$\langle$ Verb HEAD SUBCAT $\rangle$ = TRANS

Verb $\rightarrow$ gave

$\langle$ Verb HEAD SUBCAT $\rangle$ = DITRANS

Subcat Lists

Verb $\rightarrow$ gave

$\langle$ Verb HEAD SUBCAT FIRST CAT $\rangle$ = NP

$\langle$ Verb HEAD SUBCAT SECOND CAT $\rangle$ = NP

$\langle$ Verb HEAD SUBCAT THIRD $\rangle$ = END

Subcat Lists

VP $\rightarrow$ Verb NP NP

$\langle$ VP HEAD $\rangle = \langle$ Verb HEAD $\rangle$

$\langle$ VP HEAD SUBCAT FIRST CAT $\rangle = \langle$ NP CAT $\rangle$

$\langle$ VP HEAD SUBCAT SECOND CAT $\rangle = \langle$ NP CAT $\rangle$

$\langle$ VP HEAD SUBCAT THIRD $\rangle$ = END

All we're really doing is encoding lists using positional features.

For Next Time

Attend talks, do hw, study for exam!

Acknowledgements

Some of this material was taken from the course web pages of James Martin.