Introduction to Natural Language Processing

Chapter 11

Today's Outline

HWs and Project

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

- *The cat sleeps.*
- *The cats sleep.*

Subcategorization

- *Cats dream.*
- *Cats like mice.*

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:

$\star$ The cat sleep.

$\star$ 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:

$\star$ Cats dream mice.

$\star$ He took.

$\star$ He slept the cot.

$\star$ She disappeared the elephant.

Elegant Solution

We need to constrain the grammar rules to enforce number agreement, subcategorization differences, etc.

We'll do this with feature structures and the constraint-based unification formalism.

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

Why is this inelegant?

Features

An alternative is to rethink the terminal and non-terminals as complex objects with associated properties (called features) that can be manipulated.

- features take on different values.
- the application of grammar rules is constrained by testing on these features.

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 (that is, the NP and VP agree in number).

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

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

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

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

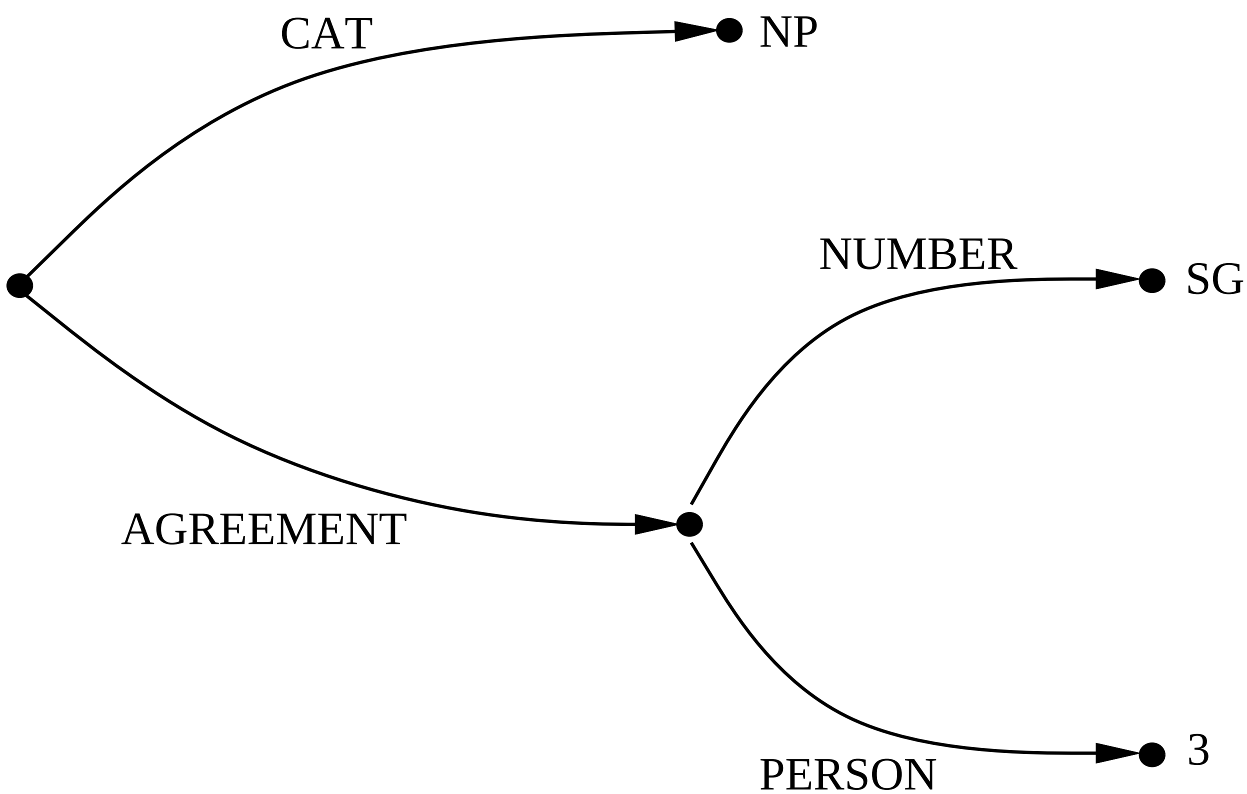
Bundles of Features

Feature values can be feature structures themselves.

Useful when certain features commonly co-occur, as number and person.

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

Feature Structures as DAGs



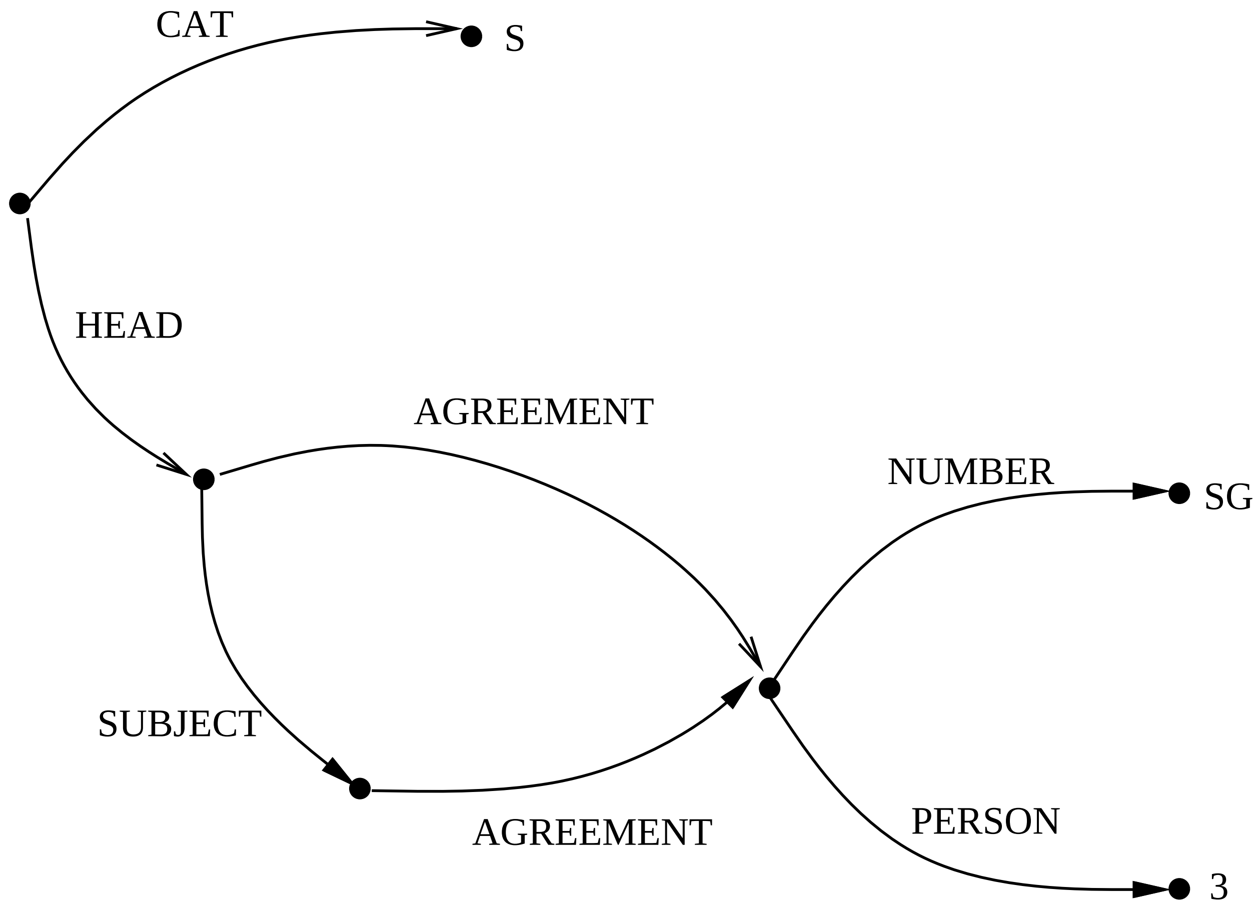
Reentrant Structures

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.

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

Numerical indices indicate the shared value.

Reentrant DAGs



Feature Paths

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

$\langle$HEAD AGREEMENT NUMBER$\rangle$

$\langle$HEAD SUBJECT AGREEMENT NUMBER$\rangle$

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

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.

Merging two feature structures produces a new feature structure that is more specific (has more information) than, or is identical to, each of the input feature structures.

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!

Structures are compatible if they contain no features that are incompatible.

If so, unification returns the union of all feature/value pairs.

The Unification Operation

$$[\textsf{number sg}] \sqcup [\textsf{number } [\,]\,] = [\textsf{number sg}]$$

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

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!

Properties of Unification

Monotonic: if some description is true of a feature structure, it will still be true after unifying it with another feature structure.

Order independent: given a set of feature structures to unify, we can unify them in any order and we'll get the same result.

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

Features of most grammatical categories are copied from head child to parent (e.g., from V to VP, Nom to NP, N to Nom)

- VP $\rightarrow$ V NP
- $\langle$ VP HEAD $\rangle$ = $\langle$ V HEAD $\rangle$

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 EARLY-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
```

Modifying Earley

Completer

- Recall: Completer adds new states to chart by finding states whose dot can be advanced (i.e., category of next constituent matches that of completed constituent)
- Now: Completer will only advance those states if their feature structures unify

Also, new test for whether to enter a state in the chart

- Now DAGs may differ, so check must be more complex
- Don't add states that have DAGs that are more specific than states in chart; is new state subsumed by existing states?

Example

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

where Dag1 equals

$$\begin{bmatrix} np & [head1] & & \\ det & [head & [agreement2 & [numbersg]]] \\ nominal & [head & 1[agreement2]] & \end{bmatrix}$$

Example

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

where Dag2 equals

$$\begin{bmatrix} nominal & [head1] \\ noun & [head \quad 1[agreement \quad [numbersg]]] \end{bmatrix}$$

and Dag1 as before

$$\begin{bmatrix} np & [head1] & & \\ det & [head & [agreement2 & [numbersg]]] \\ nominal & [head & 1[agreement2]] & \end{bmatrix}$$

Copying

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

Subcategorization Example

Recall: Different verbs take different types of arguments

Some techniques

- atomic subcat symbols
- encoding subcat lists as feature structures

Verbs also subcategorize for subjects, and other POS subcategorize

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.

Summary

Feature structures encode rich information about components of grammar rules

Unification provides a mechanism for merging structures and for comparing them

Feature structures can be quite complex as with subcategorization

Unification parsing can merge or fail, and be incorporated into Earley

For Next Time

Read parsing paper, project training data, study for exam!