Earley Parsing

Informatics 2A: Lecture 18

Mirella Lapata

School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

27 October 2011

CYK Chart entries

The CYK algorithm avoids redundant work by storing in a chart all the constituents it finds.

- Populates the table with phantom constituents.
- These are constituents that cannot occur in the context they are being suggested.
- No justification for a chart entry – why it was built.

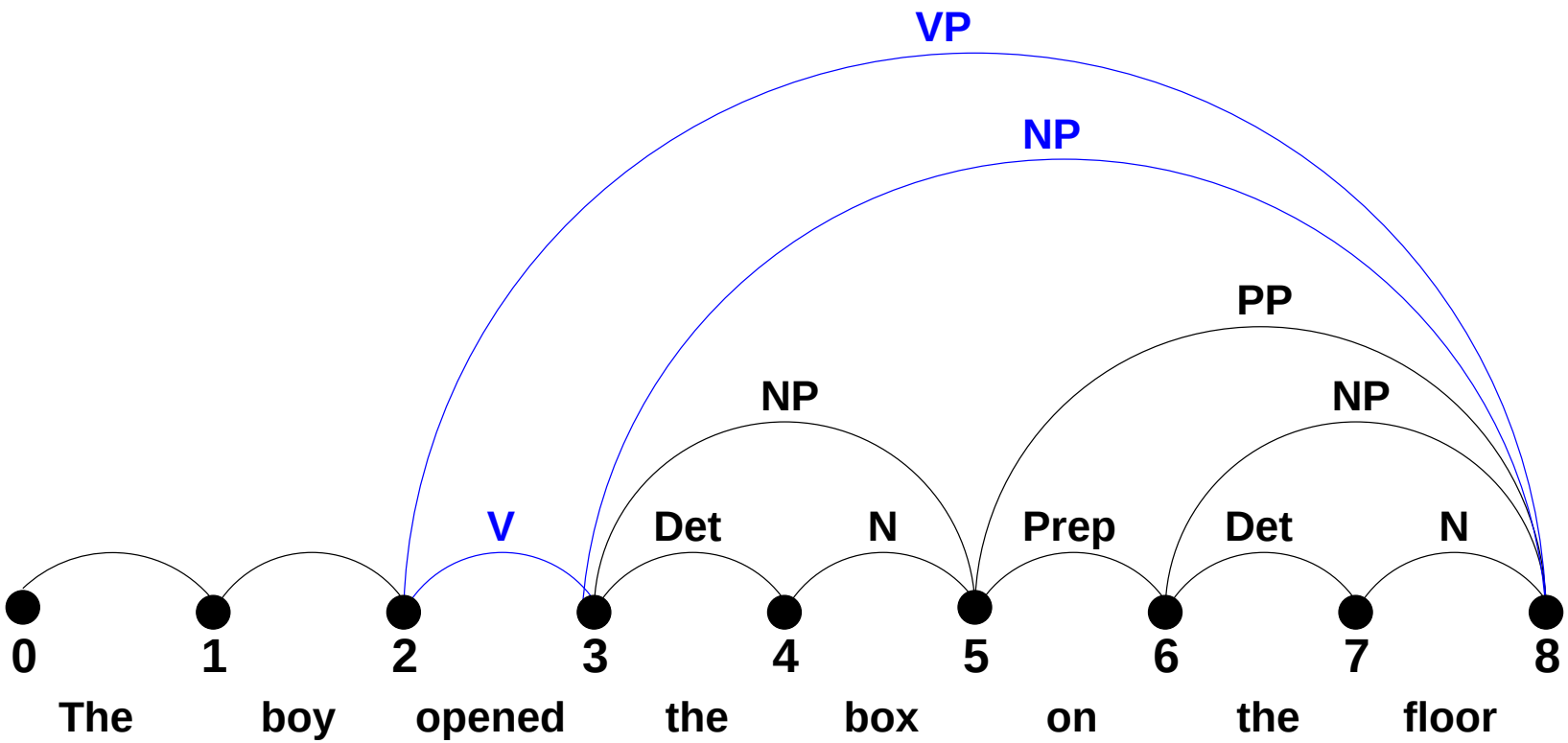
So if we have two VP rules:

$\text{VP} \rightarrow \text{V NP}$
$\text{VP} \rightarrow \text{VP PP}$

and the input string *The boy opened the box on the floor*

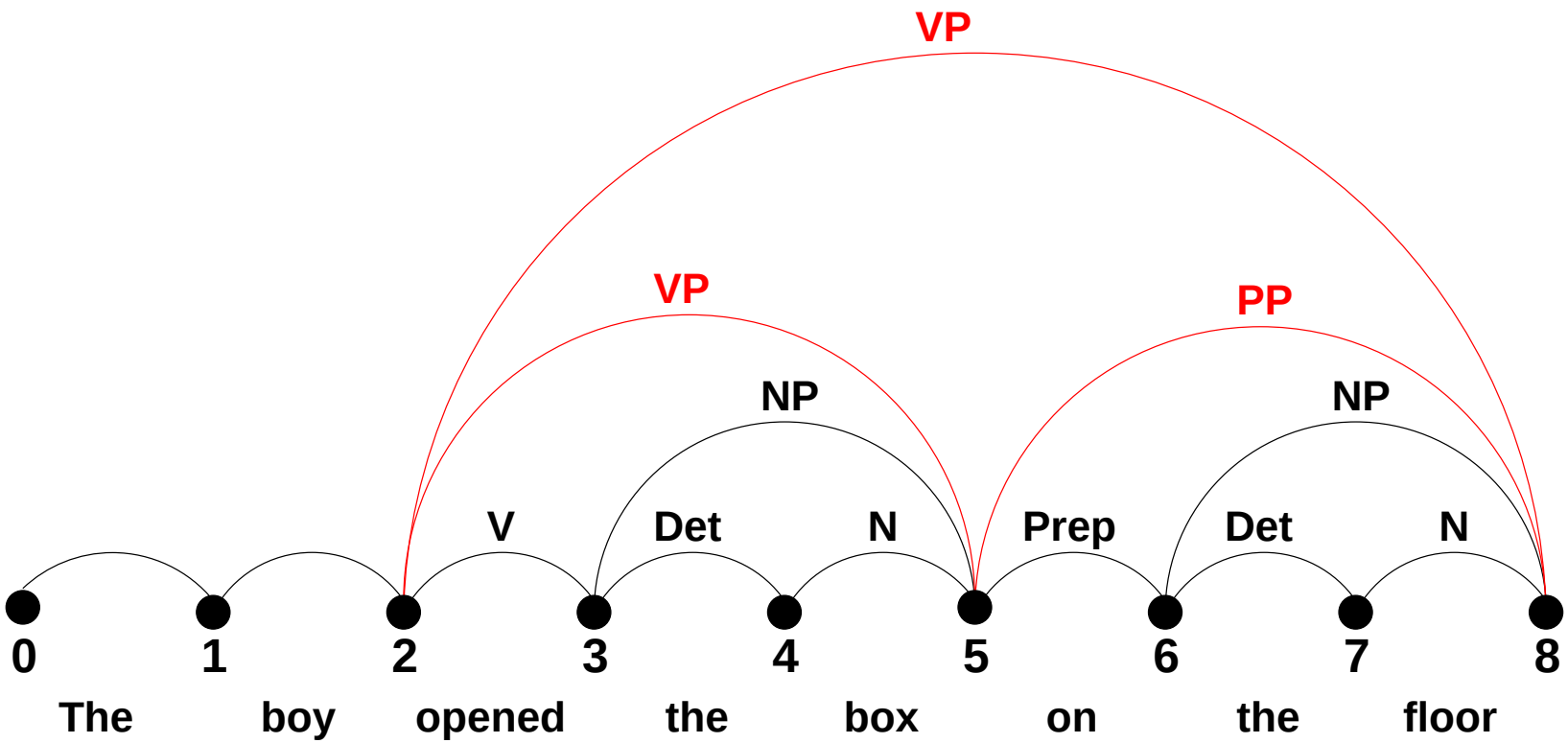
CYK Chart entries

We don't know which production the VP arc [2, 8] represents: $VP \rightarrow V\ NP$ or $VP \rightarrow VP\ PP$ (graph-based representation).



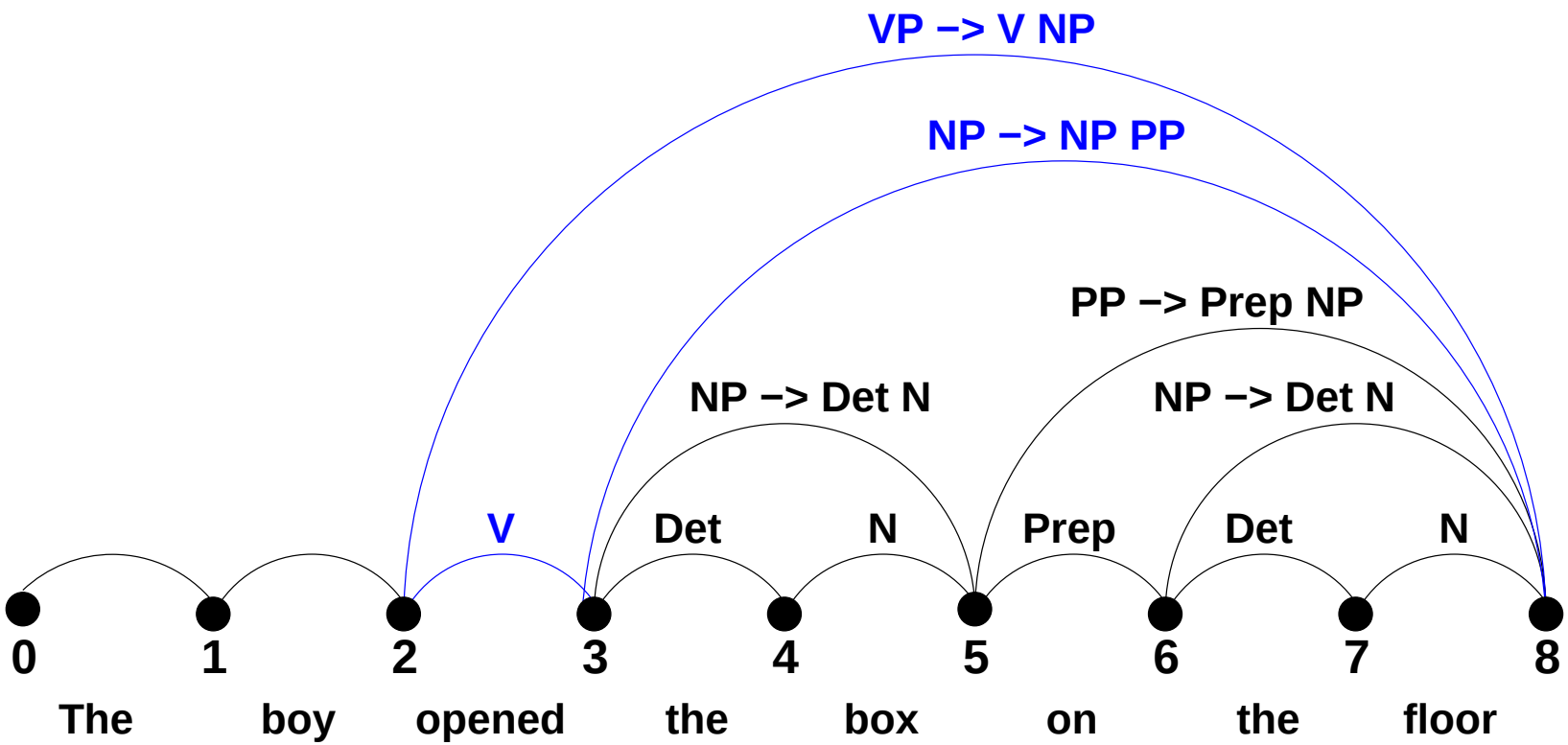
CYK Chart entries

We don't know which production the VP arc [2, 8] represents: $VP \rightarrow V\ NP$ or $VP \rightarrow VP\ PP$ (graph-based representation).



CYK Chart entries

If the entire production were recorded, rather than just its LHS (ie, the constituent that it analyses), then we'd know.



CYK Chart entries

If the entire production were recorded, rather than just its LHS (ie, the constituent that it analyses), then we'd know.

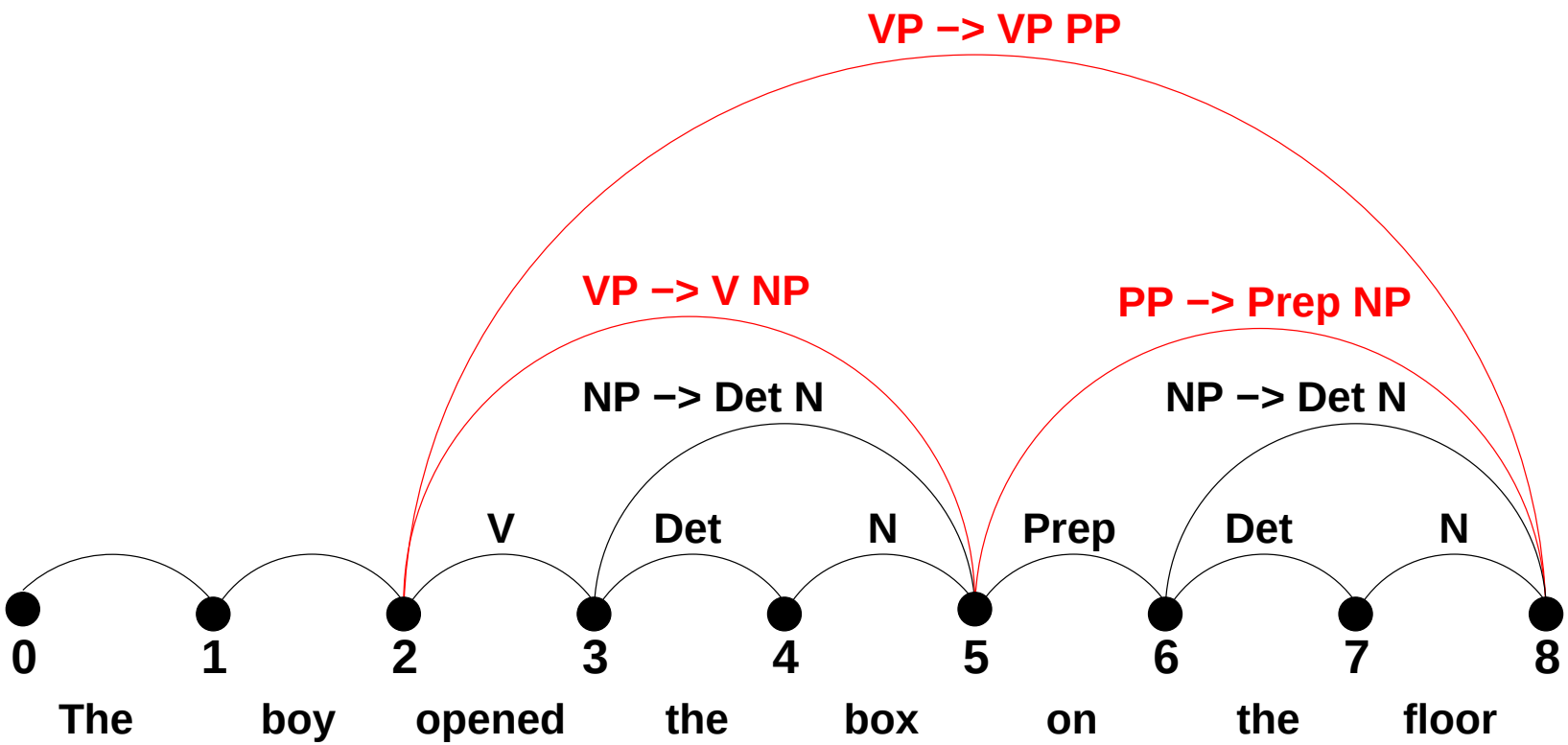
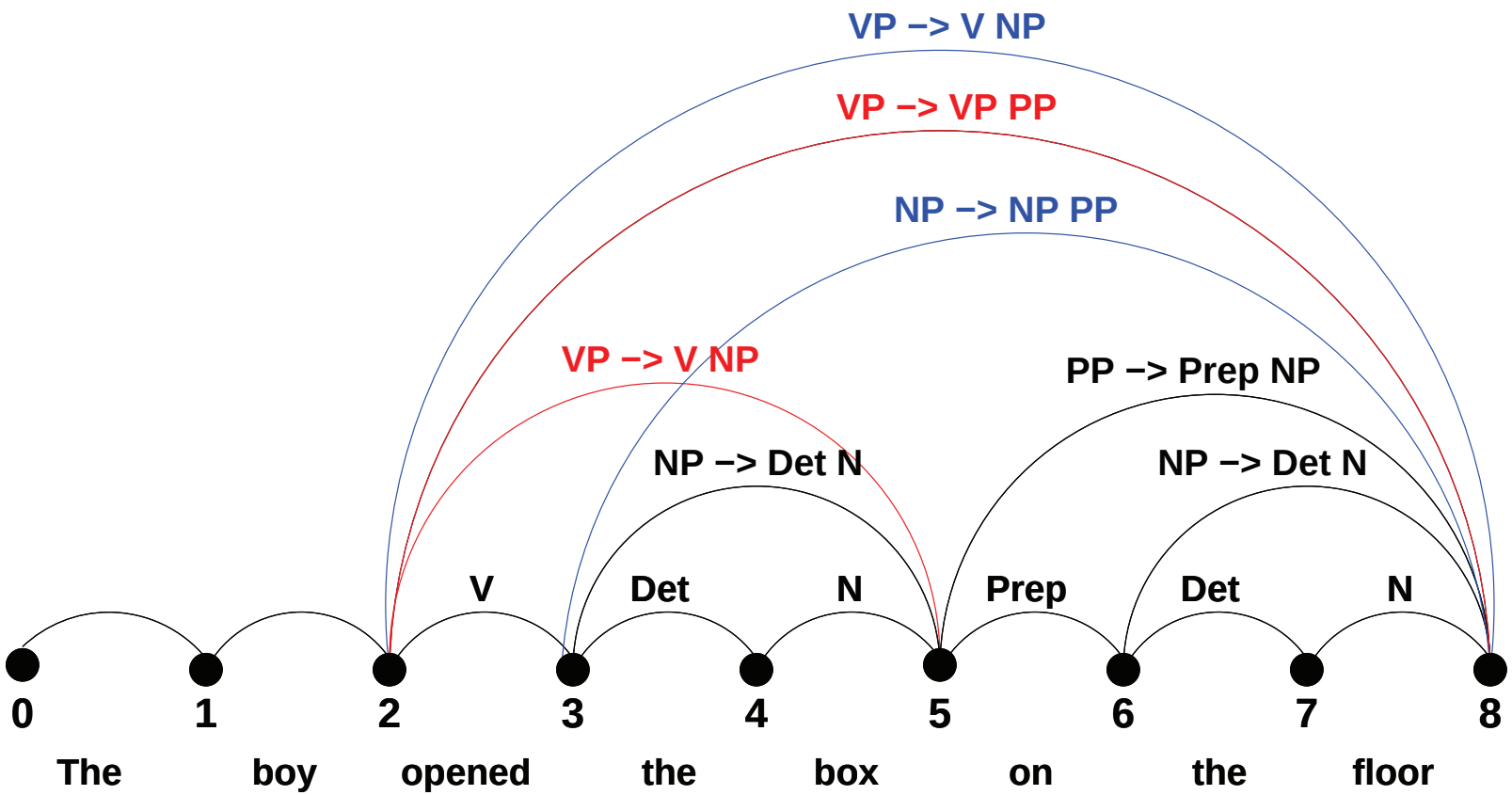


Chart entries: Both analyses



Earley Parsing

Key idea: if we record completed productions (ie, ones whose entire RHS have been recognized), we could also consider recording incomplete productions (ie, ones for which there may so far be only partial evidence).

- Incomplete productions (aka incomplete constituents) are effectively predictions about what might come next and what will be learned from finding it.
- Incomplete constituents can be represented using an extended form of production rule called a dotted rule.
- The dot indicates how much of the RHS has already been found. The rest is a prediction of what is to come.

Earley Parsing

- Allows arbitrary CFGs
- Top-down control
- Fills a table in a single sweep over the input
- Table is length $N + 1$; N is number of words
- Table entries represent:
  - Completed constituents and their locations
  - In-progress constituents
  - Predicted constituents

States

The table entries are called states and are represented with dotted-rules.

$S \rightarrow \bullet\ VP\ [0,0]$	A *VP* is predicted at the start of the sentence
$NP \rightarrow Det \bullet Nominal\ [1,2]$	An NP is in progress; seen *Det*, *Nominal* is expected
$VP \rightarrow V\ NP \bullet [0,3]$	A VP has been found starting at 0 and ending at 3

Once chart is populated there should be an S the final column that spans from 0 to N and is complete: $S \rightarrow \alpha \bullet [0, N]$. If that's the case you're done.

Sketch of Earley Algorithm

Sweep through the table from 0 to N:

1. Predict all the states you can upfront, start top-down from S
2. Read a word
   1. Extend states based on matches
   2. Generate new predictions
   3. Go to step 2
3. When you are out of words, look at the chart to see if you have a winner

The algorithm uses three basic operations to process states in the chart: Predictor and Completer add states to the chart entry being processed; Scanner adds a state to the next chart entry.

Predictor

- Creates new states representing top-down expectations
- Applied to any state that has a non-terminal immediately to its right other than a part-of-speech category
- Application results in creation of one new state for each alternative expansion of that non-terminal
- New states placed into same chart entry as generating state

$S \rightarrow \bullet VP$, [0,0]
$VP \rightarrow \bullet$ *Verb*, [0,0]
$VP \rightarrow \bullet$ *Verb NP*, [0,0]
$VP \rightarrow \bullet$ *Verb NP PP*, [0,0]
$VP \rightarrow \bullet$ *Verb PP*, [0,0]
$VP \rightarrow \bullet$ *VP PP*, [0,0]

Scanner

- Applies to states with a part-of-speech category to right of dot
- Incorportates into chart a state corresponding to prediction of a word with particular part-of-speech
- Creates new state from input state with dot advanced over predicted input category
- Unlike CYK, only parts-of-speech of a word that are predicted by some existing state will enter the chart (top-down input)

VP $\rightarrow \bullet$ *Verb NP*, [0,0]
VP $\rightarrow$ book $\bullet$, [0,1]

Completer

- Applied to state when its dot has reached right end of the rule
- This means that parser has successfully discovered a particular grammatical category over some span of the input
- Completer finds and advances all previously created states that were looking for this category at this position in input
- Creates states copying the older state, advancing dot over expected category, and installing new state in chart

$NP \rightarrow Det\ Nominal \bullet$, [1,3]			
finds state	VP	$\rightarrow$	$Verb \bullet NP$, [0,1]
finds state	VP	$\rightarrow$	$Verb \bullet NP\ PP$, [0,1]
adds complete state	VP	$\rightarrow$	$Verb\ NP \bullet$, [0,3]
adds incomplete state	VP	$\rightarrow$	$Verb\ NP \bullet PP$, [0,3]

The Earley Algorithm

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

ENQUEUE($(\gamma \rightarrow \bullet S, [0,0])$, *chart[0]*)
for $i \leftarrow$ **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*)

The Earley Algorithm

procedure PREDICTOR($(A \rightarrow \alpha \bullet B\ \beta,\ [i,j])$)
  for each $(B \rightarrow \gamma)$ **in** GRAMMAR-RULES-FOR($B, grammar$) **do**
    ENQUEUE($(B \rightarrow \bullet\ \gamma,\ [j,j]), chart[j]$)
  end

procedure SCANNER($(A \rightarrow \alpha \bullet B\ \beta,\ [i,j])$)
  if B $\subset$ PARTS-OF-SPEECH($word[j]$) **then**
    ENQUEUE($(B \rightarrow word[j],\ [j, j+1]), chart[j+1]$)

procedure COMPLETER($(B \rightarrow \gamma \bullet, [j,k])$)
  for each $(A \rightarrow \alpha \bullet B\ \beta,\ [i,j])$ **in** $chart[j]$ **do**
    ENQUEUE($(A \rightarrow \alpha\ B \bullet \beta,\ [i,k]), chart[k]$)
  end

Visualizing the Chart

We will use the grammar to parse the sentence "*Book that flight*".

Grammar Rules

$S \rightarrow NP\ VP$	$VP \rightarrow Verb$
$S \rightarrow Aux\ NP\ VP$	$VP \rightarrow Verb\ NP$
$S \rightarrow VP$	$VP \rightarrow Verb\ NP\ PP$
$NP \rightarrow Pronoun$	$VP \rightarrow Verb\ PP$
$NP \rightarrow Proper\text{-}Noun$	$VP \rightarrow VP\ PP$
$NP \rightarrow Det\ Nominal$	$PP \rightarrow Preposition\ NP$
$Nominal \rightarrow Noun$	$Verb \rightarrow book \mid include \mid prefer$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow book \mid flight \mid meal$
$Nominal \rightarrow Nominal\ PP$	$Det \rightarrow that \mid this \mid these$

Visualizing the Chart[0]

state	rule	start/end	reason
S1	$S \rightarrow \bullet\ NP\ VP$	[0,0]	Predictor
S2	$S \rightarrow \bullet\ Aux\ NP\ VP$	[0,0]	Predictor
S3	$S \rightarrow \bullet\ VP$	[0,0]	Predictor
S4	$NP \rightarrow \bullet\ Pronoun$	[0,0]	Predictor
S5	$NP \rightarrow \bullet\ Proper\text{-}Noun$	[0,0]	Predictor
S6	$NP \rightarrow \bullet\ Det\ Nominal$	[0,0]	Predictor
S7	$VP \rightarrow \bullet\ Verb$	[0,0]	Predictor
S8	$VP \rightarrow \bullet\ Verb\ NP$	[0,0]	Predictor
S9	$VP \rightarrow \bullet\ Verb\ NP\ PP$	[0,0]	Predictor
S10	$VP \rightarrow \bullet\ Verb\ PP$	[0,0]	Predictor
S11	$VP \rightarrow \bullet\ VP\ PP$	[0,0]	Predictor

Visualizing the Chart[1]

state	rule	start/end	reason
S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
S17	$S \rightarrow VP \bullet$	[0,1]	Completer
S18	$VP \rightarrow VP \bullet PP$	[1,1]	Completer
S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

Visualizing the Chart[2]

state	rule	start/end	reason
S23	$Det \rightarrow that \bullet$	[1,2]	Scanner
S24	$NP \rightarrow Det \bullet Nominal$	[1,2]	Completer
S25	$Nominal \rightarrow \bullet Noun$	[2,2]	Predictor
S26	$Nominal \rightarrow \bullet Nominal\ Noun$	[2,2]	Predictor
S27	$Nominal \rightarrow \bullet Nominal\ PP$	[2,2]	Predictor

Visualizing the Chart[3]

state	rule	start/end	reason
S28	$Noun \rightarrow \bullet\ flight$	[2,3]	Scanner
S29	$Nominal \rightarrow Noun\ \bullet$	[2,3]	Completer
S30	$NP \rightarrow Det\ Nominal\ \bullet$	[1,3]	Completer
S31	$Nominal \rightarrow Nominal\ \bullet\ Noun$	[2,3]	Completer
S32	$Nominal \rightarrow Nominal\ \bullet\ PP$	[2,3]	Completer
S33	$VP \rightarrow Verb\ NP\ \bullet$	[0,3]	Completer
S34	$VP \rightarrow Verb\ NP\ \bullet\ PP$	[0,3]	Completer
S35	$PP \rightarrow Prep\ \bullet\ NP$	[3,3]	Predictor
S36	$S \rightarrow VP\ \bullet$	[0,3]	Completer
S37	$Nominal \rightarrow VP\ \bullet\ PP$	[0,3]	Completer

Parsing the Input

As with CKY we have formulated a recognizer. We can change it to a parser by adding backpointers so that each state knows where it came from.

Chart[1]	S12	*Verb* → *book* •	[0,1]	Scanner
Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	(S28)
	S30	*NP* → *Det Nominal* •	[1,3]	(S23, S29)
	S33	*VP* → *Verb NP* •	[0,3]	(S12, S30)
	S36	*S* → *VP* •	[0,3]	(S33)

Comparing Earley and CYK

- For such a simple example, there seems to be a lot of useless stuff in the chart.
- We are predicting things inconsistent with the input!
- That's the flipside to the CKY problem.

Did we solve ambiguity? Both CKY and Earley will result in multiple S structures for the $[0, N]$ table entry. They efficiently store the sub-parts shared between multiple parses but neither can tell us which one is right.

Clicker Questions

1. The CYK parser processes the input string:
   (a) top-down (b) bottom-up (c) a bit of both
2. The Earley parser processes the input string:
   (a) top-down (b) bottom-up (c) neither, just left-to-right
3. A shift reduce parser processes the input string:
   (a) top-down (b) bottom-up (c) with a stack
4. A recursive decent parser process the input string:
   (a) top-down (b) bottom-up (c) using backtracking

Summary

- The Earley algorithm uses dynamic programming to implement a top-down search strategy.
- Single left to right pass that fills chart with $N + 1$ entries.
- Dotted rule represents progress in recognizing RHS of rule.
- Algorithm always moves forward, never backtracks to previous chart entry, once it has moved on.
- States are processed using Predictor, Completer, Scanner operations.

Reading: Same as for Lecture 17
Next lecture: statistical parsing or how to solve ambiguity.