

CSE842: Natural Language Processing

Lecture 9: Parsing with Context Free Grammar

2/11/2009

CSE842, Spring 2009, MSU

1

Syntactic Parsing

- **Declarative** formalisms like CFGs define the legal strings of a language but don't specify how to recognize or assign structure to them
- **Parsing algorithms** specify how to recognize the strings of a language and assign each string one or more syntactic structures
- **Parse trees** useful for grammar checking, semantic analysis, MT, QA, information extraction, speech recognition...and almost every task in NLP

2/11/2009

CSE842, Spring 2009, MSU

2

Parsing

- Parsing with CFGs refers to the task of assigning proper trees to input strings
- Proper here means a tree that covers **all and only the elements of the input** and **has an S at the top**
- It doesn't actually mean that the system can select the correct tree from among all the possible trees

2/11/2009

CSE842, Spring 2009, MSU

3

Comparing Top-Down and Bottom-Up

- **Top Down parsers** never explore illegal parses (e.g. can't form an S) -- but waste time on trees that can never match the input
- **Bottom Up parsers** never explore trees inconsistent with input -- but waste time exploring illegal parses (no S root)
- Of course, in both cases we left out how to keep track of the search space and how to make choices
 - Which node to try to expand next
 - Which grammar rule to use to expand a node

2/11/2009

CSE842, Spring 2009, MSU

4

Dynamic Programming

- Create table of solutions to sub problems (e.g. subtrees) as parse proceeds
- Look up subtrees for each constituent rather than re parsing, avoiding repeated work.
- Since all parses implicitly stored, all available for later disambiguation
- We will look at two approaches corresponding to top down and bottom up
 - CYK: Cocke-Younger-Kasami (CYK) (1960),
 - Earley: Earley (1970)

2/11/2009

CSE842, Spring 2009, MSU

5

CKY Parsing

- Limit our grammar to Chomsky Normal Form
 - $A \rightarrow BC$ (two non-terminals)
 - $A \rightarrow a$ (one terminal)
- Consider the rule $A \rightarrow BC$
 - If there is an A somewhere in the input then there must be a B followed by a C in the input.
 - If the A spans from i to j in the input then there must be some k st. $i < k < j$
 - I.e. The B splits from the C someplace.

2/11/2009

CSE842, Spring 2009, MSU

6

Problem

- What if your grammar isn't binary?
 - As in the case of the TreeBank grammar?
- Convert it to binary... any arbitrary CFG can be rewritten into Chomsky-Normal Form automatically.
- What does this mean?
 - The resulting grammar accepts (and rejects) the same set of strings as the original grammar.
 - **But** the resulting derivations (trees) are different.
 - Weak equivalence

2/11/2009

CSE842, Spring 2009, MSU

7

Problem

- More specifically, we want our rules to be of the form

$A \rightarrow BC$

Or

$A \rightarrow a$

That is, rules can expand to either 2 non terminals or to a single terminal.

2/11/2009

CSE842, Spring 2009, MSU

8

Conversion to CNF

- Replace terminals with non terminals for rules that mix terminals and non terminals on RHS.
 - Eliminate chains of unit productions.
 - Introduce new intermediate non terminals into the grammar that distribute rules with **length > 2** over several rules.
 - So... $S \rightarrow ABC$ turns into
 - $S \rightarrow XC$ and
 - $X \rightarrow AB$
- Where X is a symbol that doesn't occur anywhere else in the the grammar.

2/11/2009

CSE842, Spring 2009, MSU

9

Sample L1 Grammar

Grammar	Lexicon
$S \rightarrow NP VP$	$Det \rightarrow that this a$
$S \rightarrow Aux NP VP$	$Noun \rightarrow book flight meal money$
$S \rightarrow VP$	$Verb \rightarrow book include prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I she me$
$NP \rightarrow Proper-Noun$	$Proper-Noun \rightarrow Houston NWA$
$NP \rightarrow Det Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from to on near through$
$Nominal \rightarrow Nominal Noun$	
$Nominal \rightarrow Nominal PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb NP$	
$VP \rightarrow Verb NP PP$	
$VP \rightarrow Verb PP$	
$VP \rightarrow VP PP$	
$PP \rightarrow Preposition NP$	

2/11/2009

CSE842, Spring 2009, MSU

10

CNF Conversion

\mathcal{L}_1 Grammar	\mathcal{L}_1 in CNF
$S \rightarrow NP VP$	$S \rightarrow NP VP$
$S \rightarrow Aux NP VP$	$S \rightarrow X1 VP$
	$X1 \rightarrow Aux NP$
$S \rightarrow VP$	$S \rightarrow book include prefer$
	$S \rightarrow Verb NP$
	$S \rightarrow X2 PP$
	$S \rightarrow Verb PP$
	$S \rightarrow VP PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I she me$
$NP \rightarrow Proper-Noun$	$NP \rightarrow TWA Houston$
$NP \rightarrow Det Nominal$	$NP \rightarrow Det Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book flight meal money$
$Nominal \rightarrow Nominal Noun$	$Nominal \rightarrow Nominal Noun$
$Nominal \rightarrow Nominal PP$	$Nominal \rightarrow Nominal PP$
$VP \rightarrow Verb$	$VP \rightarrow book include prefer$
$VP \rightarrow Verb NP$	$VP \rightarrow Verb NP$
$VP \rightarrow Verb NP PP$	$VP \rightarrow X2 PP$
	$X2 \rightarrow Verb NP$
$VP \rightarrow Verb PP$	$VP \rightarrow Verb PP$
$VP \rightarrow VP PP$	$VP \rightarrow VP PP$
$PP \rightarrow Preposition NP$	$PP \rightarrow Preposition NP$

2

CKY Intuition

- So let's build a table so that each cell $[i,j]$ in the table stores the constituent (e.g., A) spanning from i to j in the input.
- So a non-terminal spanning an entire string will sit in cell $[0, n]$
 - Hopefully an S
- If we build the table bottom-up, we'll know that the parts of the A must go from i to k and from k to j , for some k .
- In other words, if we think there might be an A spanning i,j in the input then we need to check if there exists a rule $A \rightarrow BC$ where B is in $[i,k]$ and C is in $[k,j]$ for some $i < k < j$

2/11/2009

CSE842, Spring 2009, MSU

12

CKY

- So to fill the table loop over the cell $[i,j]$ values in some systematic way
 - What constraint should we put on that systematic search?
 - For each cell, loop over the appropriate k values to search for things to add.

2/11/2009

CSE842, Spring 2009, MSU

13

CKY

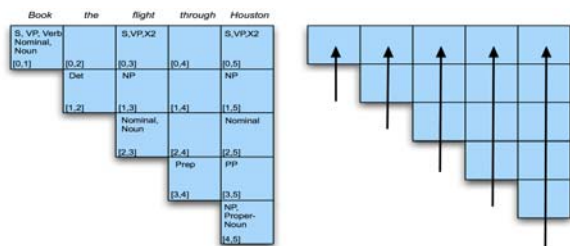
- We arranged the loops to fill the table a column at a time, from left to right, bottom to top.
 - This assures us that whenever we're filling a cell, the parts needed to fill it are already in the table (to the left and below)
 - It's somewhat natural in that it processes the input a left to right a word at a time
 - Known as online

2/11/2009

CSE842, Spring 2009, MSU

14

Example



2/11/2009

CSE842, Spring 2009, MSU

15

CKY Algorithm

```

function CKY-PARSE(words, grammar) returns table
  for j ← from 1 to LENGTH(words) do
    table[j-1, j] ← {A | A → words[j] ∈ grammar}
    for i ← from j-2 downto 0 do
      for k ← i+1 to j-1 do
        table[i, j] ← table[i, j] ∪
          {A | A → BC ∈ grammar,
            B ∈ table[i, k],
            C ∈ table[k, j]}
    
```

2/11/2009

CSE842, Spring 2009, MSU

16

CKY Parsing

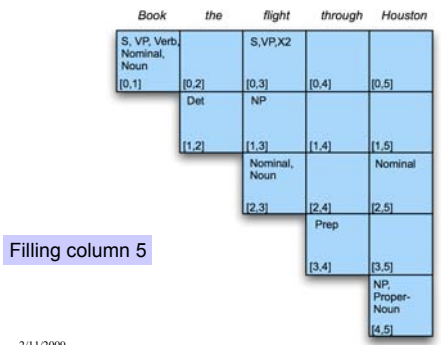
- Is that really a parser?
- What needs to be changed to turn this algorithm into a parser?

2/11/2009

CSE842, Spring 2009, MSU

17

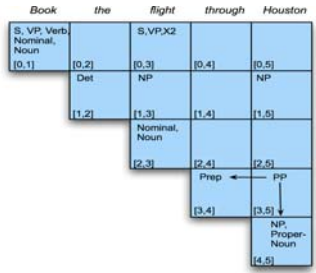
Example



2/11/2009

18

Example

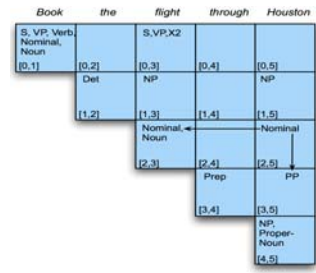


2/11/2009

CSE842, Spring 2009, MSU

19

Example

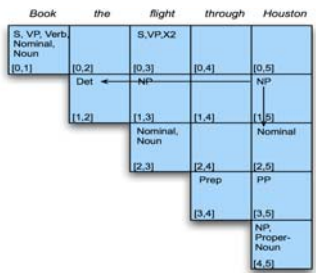


2/11/2009

CSE842, Spring 2009, MSU

20

Example

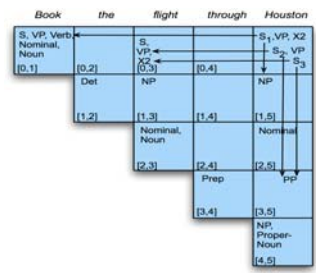


2/11/2009

, MSU

21

Example



2/11/2009

CSE842, Spring 2009, MSU

22

CKY Notes

- The problems of constructing parsing trees using CKY in practice.
 - Post process resulting trees for restoration.
- Since it's bottom up, CKY populates the table with a lot of phantom constituents.
 - Segments that by themselves are constituents but cannot really occur in the context in which they are being suggested.
 - To avoid this we can switch to a top down control strategy

2/11/2009

CSE842, Spring 2009, MSU

23

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
 - Think of chart entries as sitting between words in the input string keeping track of **states** of the parse at these positions
 - For each word position, chart contains set of states representing all partial parse trees generated to date.
 - Completed constituents and their locations
 - In-progress constituents
 - Predicted constituents

2/11/2009

CSE842, Spring 2009, MSU

24

States

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

$S \rightarrow \cdot VP$ A VP is predicted
 $NP \rightarrow Det \cdot Nominal$ An NP is in progress
 $VP \rightarrow V NP \cdot$ A VP has been found

2/11/2009

CSE842, Spring 2009, MSU

25

States/Locations

$-[x,y]$ tells us where the state begins (x) and where the dot lies (y) with respect to the input

- $S \rightarrow \cdot VP$ [0,0] • A VP is predicted at the start of the sentence
- $NP \rightarrow Det \cdot Nominal$ [1,2] • An NP is in progress; the Det goes from 1 to 2
- $VP \rightarrow V NP \cdot$ [0,3] • A VP has been found starting at 0 and ending at 3

2/11/2009

CSE842, Spring 2009, MSU

26

$_0$ Book $_1$ that $_2$ flight $_3$

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

- First 0 means S constituent begins at the start of the input
- Second 0 means the dot is here too
- So, this is a top-down prediction

$NP \rightarrow \cdot Det$, [1,2]

- the NP begins at position 1
- the dot is currently at position 2
- so, Det has been successfully parsed
- Nom is predicted next

2/11/2009

CSE842, Spring 2009, MSU

27

Successful Parse

- Final answer found by looking at last entry in chart
- If entry resembles $S \rightarrow \cdot \alpha$ [0,N] then input parsed successfully
- But note that chart will also contain a record of all possible parses of input string, given the grammar -- not just the successful one(s)

2/11/2009

CSE842, Spring 2009, MSU

28

Parsing Procedure for the Earley Algorithm

- Move through each set of states in order, applying one of three operators to each state:
 - predictor**: add predictions to the chart
 - scanner**: read input and add corresponding state to chart
 - completer**: move dot to right when new constituent found
- Results (new states) added to current or next set of states in chart
- No backtracking and no states removed: keep complete history of parse

2/11/2009

CSE842, Spring 2009, MSU

29

Core Earley Code

```
function EARLEY-PARSE(words, grammar) returns chart
  ENQUEUE( $\gamma \rightarrow \cdot S$ , [0,0], 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)
```

2/11/2009

CSE842, Spring 2009, MSU

30

Predictor

- Intuition: create new states represent top down expectations
- Applied when non part of speech non terminals are to the right of a dot
 $S \rightarrow \bullet VP [0,0]$
- Adds new states to *current* chart
 - One new state for each expansion of the non-terminal in the grammar
 $VP \rightarrow \bullet V [0,0]$
 $VP \rightarrow \bullet V NP [0,0]$

2/11/2009

CSE842, Spring 2009, MSU

31

Scanner

- New states for predicted part of speech.
- Applicable when part of speech is to the right of a dot
 $VP \rightarrow V \bullet NP [0,0]$ 'Book...'
- Looks at current word in input
- If match, adds state(s) to *next* chart
 $V \rightarrow \bullet book [0,1]$

2/11/2009

CSE842, Spring 2009, MSU

32

Completer

- Intuition: parser has discovered a constituent, so must find and advance states all that were waiting for this
- Applied when dot has reached right end of rule
 $NP \rightarrow Det \bullet Nom [1,3]$
- Find all states w/dot at 1 and expecting an NP
 $VP \rightarrow V \bullet NP [0,1]$
- Adds new (completed) state(s) to *current* chart
 $VP \rightarrow V NP \bullet [0,3]$

2/11/2009

CSE842, Spring 2009, MSU

33

Earley Code

```
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  $B \in$  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
```

2/11/2009

CSE842, Spring 2009, MSU

34