

CSE842: Natural Language Processing

Lecture 10: Parsing and Probabilistic Parsing

2/16/2009

CSE842, Spring 2009, MSU

1

Announcement

- The “paper survey and presentation” assignment is posted on the Angel system

2/16/2009

CSE842, Spring 2009, MSU

2

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)

2/16/2009

CSE842, Spring 2009, MSU

3

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

2/16/2009

CSE842, Spring 2009, MSU

4

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

2/16/2009

CSE842, Spring 2009, MSU

5

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 |

2/16/2009

CSE842, Spring 2009, MSU

6

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

S-- \rightarrow 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 Det • Nom, [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

2/16/2009

CSE842, Spring 2009, MSU

7

Successful Parse

- Final answer found by looking at last entry in chart
- If entry resembles S $\rightarrow \alpha \bullet$ [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)

2/16/2009

CSE842, Spring 2009, MSU

8

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

2/16/2009

CSE842, Spring 2009, MSU

9

Core Earley Code

```
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)
```

2/16/2009

CSE842, Spring 2009, MSU

10

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]

2/16/2009

CSE842, Spring 2009, MSU

11

Scanner

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

2/16/2009

CSE842, Spring 2009, MSU

12

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 → Det Nom • [1,3]
- Find all states w/dot at 1 and expecting an NP
VP → V • NP [0,1]
- Adds new (completed) state(s) to **current** chart
VP → V NP • [0,3]

2/16/2009

CSE842, Spring 2009, MSU

13

Earley Code

```

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

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

procedure COMPLETER((B → γ •, [j, k]))
  for each (A → α • B β, [i, j]) in chart[j] do
    ENQUEUE((A → α B • β, [i, k], chart[k])
  end
    
```

2/16/2009

CSE842, Spring 2009, MSU

14

CFG for Fragment of English

S → NP VP	
S → Aux NP VP	Det → that this a
S → VP	N → book flight meal money
NP → Det Nom	V → book include prefer
NP → PropN	Aux → does
Nom → N	Prep → from to on
Nom → Nom N	PropN → Houston TWA
VP → V	
VP → V NP	

Note: the example here uses less rules than the example in the book.

2/16/2009

CSE842, Spring 2009, MSU

15

Book that flight (Chart [0])

Seed chart with top down predictions for S from grammar

γ → • S	[0,0]	Dummy start state
S → • NP VP	[0,0]	Predictor
S → • Aux NP VP	[0,0]	Predictor
S → • VP	[0,0]	Predictor
NP → • Det Nom	[0,0]	Predictor
NP → • PropN	[0,0]	Predictor
VP → • V	[0,0]	Predictor
VP → • V NP	[0,0]	Predictor

2/16/2009

CSE842, Spring 2009, MSU

16

- When dummy start state is processed, it's passed to **Predictor**, which produces states representing every possible expansion of S, and adds these and every expansion of the left corners of these trees to bottom of Chart[0]
- When VP → V • [0,0] is reached, **Scanner** called, which consults first word of input, **Book**, and adds first state to Chart[1], V → • Book •, [0,1]
- Note: When VP → V NP • [0,0] is reached in Chart[0], Scanner does not need to add V → • Book •, [0,1] again to Chart[1]

2/16/2009

CSE842, Spring 2009, MSU

17

Chart[1]

V → book •	[0,1]	Scanner
VP → V •	[0,1]	Completer
VP → V • NP	[0,1]	Completer
S → VP •	[0,1]	Completer
NP → • Det Nom	[1,1]	Predictor
NP → • PropN	[1,1]	Predictor

V → • book • passed to **Completer**, which finds 2 states in Chart[0] whose left corner is V and adds them to Chart[1], moving dots to right

2/16/2009

CSE842, Spring 2009, MSU

18

- When $VP \rightarrow V \bullet$ is itself processed by the Completer, $S \rightarrow VP \bullet$ is added to Chart[1] since VP is a left corner of S
- Last 2 rules in Chart[1] are added by **Predictor** when $VP \rightarrow V \bullet NP$ is processed
- And so on....

2/16/2009

CSE842, Spring 2009, MSU

19

Chart[2]

Det \rightarrow that \bullet	[1,2]	Scanner
NP \rightarrow Det \bullet Nom	[1,2]	Completer
Nom $\rightarrow \bullet$ N	[2,2]	Predictor
Nom $\rightarrow \bullet$ Nom N	[2,2]	Predictor

2/16/2009

CSE842, Spring 2009, MSU

20

Chart[3]

N \rightarrow flight \bullet	[2,3]	Scanner
Nom \rightarrow N \bullet	[2,3]	Completer
NP \rightarrow Det Nom \bullet	[1,3]	Completer
Nom \rightarrow Nom \bullet N	[2,3]	Completer
VP \rightarrow V NP \bullet	[0,3]	Completer
S \rightarrow VP \bullet	[0,3]	Completer

2/16/2009

CSE842, Spring 2009, MSU

21

How do we retrieve the parses at the end?

- Augment the Completer to add pointers to prior states it advances as a field in the current state
 - i.e. what state did we advance here?
 - Read the pointers back from the final state
- Need to add these pointers in your homework 2 assignment.

2/16/2009

CSE842, Spring 2009, MSU

22

Efficiency

- For such a simple example, there seems to be a lot of useless stuff in there.
- Why?
 - It's predicting things that aren't consistent with the input
 - That's the flipside to the CKY problem.

2/16/2009

CSE842, Spring 2009, MSU

23

Ambiguity

- Did we solve the ambiguity problem?
- No...
 - Both CKY and Earley will likely result in multiple S structures for the $[0,N]$ table entry.
 - They both efficiently store the sub- parts that are shared between multiple parses.
 - And they obviously avoid re-deriving those sub parts.
 - But neither can tell us which one is right.

2/16/2009

CSE842, Spring 2009, MSU

24

Error Handling

- What happens when we look at the contents of the last table column and don't find a $S \rightarrow \alpha$ rule?
 - Is it a total loss?

2/16/2009

CSE842, Spring 2009, MSU

25

Error Handling

- What happens when we look at the contents of the last table column and don't find a $S \rightarrow \alpha$ rule?
 - Is it a total loss? No...
 - Chart contains every constituent and combination of constituents possible for the input given the grammar
- Also useful for partial parsing or shallow parsing used in information extraction

2/16/2009

CSE842, Spring 2009, MSU

26

Probabilistic Context Free Grammars

- The simplest augmentation of the Context Free Grammar (CFG) is the Probabilistic Context Free Grammar (PCFG)
- CFG is defined by (N, Σ, P, S)
 - N : a set of non-terminals
 - Σ : set of terminals
 - P : set of productions of the form: $A \rightarrow \beta$, $A \in N$, β a string of symbols $s \in (\Sigma \cup N)$
 - S : a designated start symbol

2/16/2009

CSE842, Spring 2009, MSU

27

PCFG Augmentation

- Each rule in P is assigned a conditional probability $A \rightarrow \beta | p$
- A PCFG is a 5 tuple $G = (N, \Sigma, P, S, D)$ where D is a function assigning probabilities to each rule in P $P(A \rightarrow \beta)$ or $P(A \rightarrow \beta | A)$
- Given a non-terminal A , and r_1, r_2, \dots, r_i , all expanding A

$$\sum_{j=1}^i P(r_j) = 1$$

2/16/2009

CSE842, Spring 2009, MSU

28

Example

Grammar		Lexicon	
$S \rightarrow NP VP$	[.80]	<i>Det</i> \rightarrow <i>that</i> [.10] <i>a</i> [.30] <i>the</i> [.60]	
$S \rightarrow Aux NP VP$	[.15]	<i>Noun</i> \rightarrow <i>book</i> [.10] <i>flight</i> [.30]	
$S \rightarrow VP$	[.05]	<i>meal</i> [.15] <i>money</i> [.05]	
$NP \rightarrow Pronoun$	[.35]	<i>flights</i> [.40] <i>dinner</i> [.10]	
$NP \rightarrow Proper-Noun$	[.30]	<i>Verb</i> \rightarrow <i>book</i> [.30] <i>include</i> [.30]	
$NP \rightarrow Det Nominal$	[.20]	<i>prefer</i> [.40]	
$NP \rightarrow Nominal$	[.15]	<i>Pronoun</i> \rightarrow <i>I</i> [.40] <i>she</i> [.05]	
$Nominal \rightarrow Noun$	[.75]	<i>me</i> [.15] <i>you</i> [.40]	
$Nominal \rightarrow Nominal Noun$	[.20]	<i>Proper-Noun</i> \rightarrow <i>Houston</i> [.60]	
$Nominal \rightarrow Nominal PP$	[.05]	<i>NWA</i> [.40]	
$VP \rightarrow Verb$	[.35]	<i>Aux</i> \rightarrow <i>does</i> [.60] <i>can</i> [.40]	
$VP \rightarrow Verb NP$	[.20]	<i>Preposition</i> \rightarrow <i>from</i> [.30] <i>to</i> [.30]	
$VP \rightarrow Verb NP PP$	[.10]	<i>on</i> [.20] <i>near</i> [.15]	
$VP \rightarrow Verb PP$	[.15]	<i>through</i> [.05]	
$VP \rightarrow Verb NP NP$	[.05]		
$VP \rightarrow VP PP$	[.15]		
$PP \rightarrow Preposition NP$	[.10]		

2/16/2009

CSE842, Spring 2009, MSU

29

Using Probabilities

- To estimate probabilities concerning a sentence and its parse trees
- Useful in disambiguation
- How can we define the probability of a tree?

$$P(T, S) = \prod_{u \in T} P(r(u))$$

$$P(T, S) = P(T) P(S | T) = P(T)$$

2/16/2009

CSE842, Spring 2009, MSU

30

Use in Disambiguation

- The most likely tree for a sentence S

$$\hat{T}(S) = \arg \max_{T \in \tau(S)} P(T | S)$$

$$\hat{T}(S) = \arg \max_{T \in \tau(S)} \frac{P(T, S)}{P(S)}$$

$$\hat{T}(S) = \arg \max_{T \in \tau(S)} P(T, S)$$

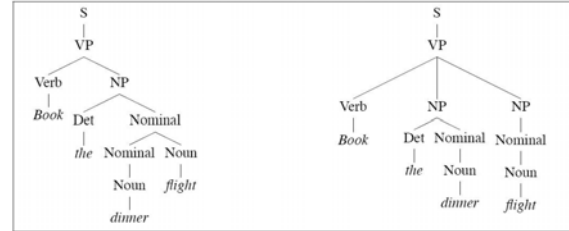
But $P(T, S) = P(T)$ \rightarrow $\hat{T}(S) = \arg \max_{T \in \tau(S)} P(T)$

2/16/2009

CSE842, Spring 2009, MSU

31

Disambiguation Example



2/16/2009

CSE842, Spring 2009, MSU

32

Disambiguation Example

Rules	P	Rules	P
S → VP	.05	S → VP	.05
VP → Verb NP	.20	VP → Verb NP NP	.10
NP → Det Nominal	.20	NP → Det Nominal	.20
Nominal → Nominal Noun	.20	NP → Nominal	.15
Nominal → Noun	.75	Nominal → Noun	.75
Verb → book	.30	Nominal → Noun	.75
Det → the	.60	Verb → book	.30
Noun → dinner	.10	Det → the	.60
Noun → flights	.40	Noun → dinner	.10
		Noun → flights	.40

$$P(T_{\text{left}}) = 0.05 * 0.2 * 0.2 * 0.2 * 0.75 * 0.3 * 0.6 * 0.1 * 0.4 = 2.2 \times 10^{-6}$$

$$P(T_{\text{right}}) = 0.05 * 0.1 * 0.2 * 0.15 * 0.75 * 0.75 * 0.3 * 0.6 * 0.1 * 0.4 = 6.1 \times 10^{-7}$$

2/16/2009

CSE842, Spring 2009, MSU

33

Learning PCFG Probabilities

Given a set of examples trees, use MLE

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

2/16/2009

CSE842, Spring 2009, MSU

34

Algorithms for PCFG

Given a PCFG and a sentence S, define $\tau(S)$ to be the set of trees with S as the yield.

- How do we find the best parse tree for the sentence S?

$$\arg \max_{T \in \tau(S)} P(T, S)$$

- How do we find the probability of S?

$$P(S) = \sum_{T \in \tau(S)} P(T, S)$$

2/16/2009

CSE842, Spring 2009, MSU

35

Review: Chomsky Normal Form

A context free grammar $G = (N, \Sigma, P, S)$ in Chomsky Normal Form is as follows

- N is a set of non terminal symbols
- Σ is a set of terminal symbols
- P is a set of rules which take one of two forms
 - $X \rightarrow Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
 - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

2/16/2009

CSE842, Spring 2009, MSU

36

Probabilistic Parsing

- CYK Algorithm: bottom up parser
- Input:
 - A Chomsky normal form PCFG, $G = (N, \Sigma, P, S, D)$ Assume that the K non-terminals have indices $1, 2, \dots, K$, and the start symbol S has index 1
 - n words w_1, \dots, w_n
- Data Structure:
 - A dynamic programming array $\pi[i, j, k]$ holds the **maximum probability** for a constituent with non-terminal index N_k spanning words $i..j$.
- Output: The maximum probability parse $\pi[0, n, 1]$

2/16/2009

CSE842, Spring 2009, MSU

37

Base Case

- CYK fills out $\pi[i, j, k]$ by induction
- Base case
 - Input strings with length = 1 (individual words w_i)
 - In CNF, the probability of a given non-terminal N_k expanding to a single word w_i must come only from the rule $N_k \rightarrow w_i$

For all $j = 1..n$, for $k = 1..K$

$$\pi[j-1, j, k] = P(N_k \rightarrow w_j | N_k)$$

2/16/2009

CSE842, Spring 2009, MSU

38

Recursive Case

- For strings of words $> 1, N_k \Rightarrow^* w_{ij}$ if and only if there is one rule $N_k \rightarrow N_l N_m$ and $s, i \leq s \leq j$ such that N_l derives the first $s-i$ symbols and N_m derives the last $j-s$ symbols.
- Since w_{is} and w_{sj} are shorter than w_{ij} , $\pi[i, s, l]$ and $\pi[s+1, j, m]$ are already stored.

$$\pi[i, j, k] = \pi[i, s, l] * \pi[s+1, j, m]$$

for all $j = 1..n, i = (j-2)$ downto $0, k = 1..K,$

$$\pi[i, j, k] = \max_{\substack{l \leq s < j \\ l \leq l \leq K \\ l \leq m \leq K}} \{ P(N_k \rightarrow N_l N_m | N_k) \times \pi[i, s, l] \times \pi[s+1, j, m] \}$$

2/16/2009

CSE842, Spring 2009, MSU

39

CYK Algorithm for Probabilistic Parsing

Initialization

for $j = 1..n, k = 1..K$

$$\pi[j-1, j, k] = P(N_k \rightarrow w_j | N_k)$$

Main Loop:

for $j = 1..length$

for $i = j-2$ downto 0

for $k = 1..K$

max = 0;

for $s = (i+1)..(j-1), l = 1..K, m = 1..K,$

prob = $P(N_k \rightarrow N_l N_m | N_k) \times \pi[i, s, l] \times \pi[s+1, j, m]$

if prob > max

max = prob;

split $(i, j, k) = (s, l, m);$ //store backpointers to indicate the best split

$\pi[i, j, k] = \max$

2/16/2009

CSE842, Spring 2009, MSU

40