Parsing Context-free grammars Part 2
ICS 482 Natural Language Processing

Lecture 13: Parsing Context-free grammars Part 2

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم

ICS 482 Natural Language Processing

Lecture 13: Parsing Context-free grammars Part 2

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

---

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit
Azadeh Maghsoodi

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

Previous Lectures

- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata & Regular Expressions & languages
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- Statistical NLP – Language Modeling
- N Grams
- Smoothing and NGram: Add-one & Witten-Bell
- Parts of Speech and Arabic Parts of Speech
- Syntax: Context Free Grammar (CFG)
- Parsing Context Free Grammars: Top-Down, Bottom-Up, Top-down parsing with bottom-up filtering

Today's Lecture

- Parsing Context Free Grammars
  - Dynamic
    - Earley’s Algorithm
- Reminder: Next Time
  - 25 Minutes Lecture
    - Tuesday 3rd April 2007
    - Chapters 6, 8, 9, 10 and covered material
    - Sample quiz will be on the site by Monday afternoon

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
- Since all parses implicitly stored, all available for later disambiguation
- Method:
  - Cocke-Younger-Kasami (CYK) (1960)
  - Graham-Harrison-Ruzzo (GHR) (1980)
  - Earley's (1970) algorithm

Earley's Algorithm

- Uses dynamic programming to do parallel top-down search
- First, L2R pass fills out a chart with N+1 states (N: the number of words in the input)
  - 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. E.g. chart[0] contains all partial parse trees generated at the beginning of the sentence

Earley's Algorithm

- Chart entries represent three type of constituents:
  - predicted constituents
  - in-progress constituents
  - completed constituents
- Progress in parse represented by Dotted Rules •
  - Position of • indicates type of constituent
  - $_0$ Book $_1$ that $_2$ flight $_3$

    S → • VP, [0,0] (predicting VP)

    NP → Det • Nominal, [1,2] (finding NP)

    VP → V NP •, [0,3] (found VP)
  - [x,y] tells us where the state begins (x) and where the dot lies (y) with respect to the input

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

S → • VP, [0,0]

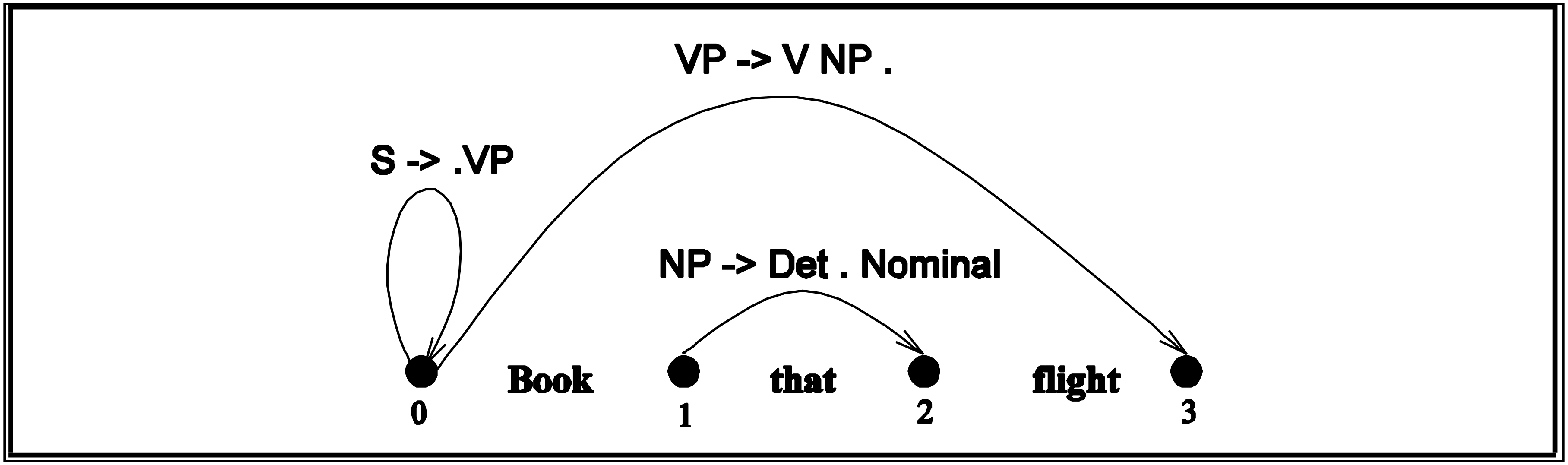
- First 0 means S constituent begins at the start of the input
- Second 0 means the dot here too
- So, this is a top-down prediction

NP → Det • Nominal, [1,2]

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

VP → V NP •, [0,3]

- Successful VP parse of entire input



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)

Parsing Procedure for the Earley's 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

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 → • VP [0,0]

- Adds new states to ***current*** chart
  - One new state for each expansion of the non-terminal in the grammar

    VP → • V [0,0]

    VP → • V NP [0,0]

S → NP VP	VP → Verb
S → Aux NP VP	Det → that | this | a
S → VP	Noun → book | flight | meal | money
NP → Det Nominal	Verb → book | include | prefer
NP → ProperNoun	Aux → does
Nominal → Noun Nominal	Prep → from | to | on
Nominal → Noun	ProperNoun → Houston | TWA
VP → Verb NP	PP → Prep NP

Scanner

- New states for predicted **part of speech**.
- Applicable when **part of speech** is to the right of a dot

  VP → • V NP [0,0] 'Book…'
- Looks at current word in input
- If match, adds state(s) to ***next*** chart

  VP → V • NP [0,1]

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 Nominal • [1,3]
- Find all states with 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]

CFG for Fragment of English

S → NP VP	VP → Verb
S → Aux NP VP	Det → that | this | a
S → VP	Noun → book | flight | meal | money
NP → Det Nominal	Verb → book | include | prefer
NP → ProperNoun	Aux → does
Nominal → Noun Nominal	Prep →from | to | on
Nominal → Noun	ProperNoun → Houston | TWA
VP → Verb NP	PP → Prep NP

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 → • ProperNoun	[0,0]	Predictor
VP → • Verb	[0,0]	Predictor
VP → • Verb NP	[0,0]	Predictor

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
Nominal → Noun
Nominal → Noun Nominal
NP → Proper-Noun
VP → Verb
VP → Verb NP
Nominal → Nominal PP
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Proper-Noun → Houston | TWA
Prep → from | to | on

- 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 → • Verb, [0,0] is reached, Scanner called, which consults first word of input, Book, and adds first state to

  Chart[1], VP → Book •, [0,1]
- Note: When VP → • Verb NP, [0,0] is reached in Chart[0], Scanner does not need to add VP → Book •, [0,1] again to Chart[1]

Chart[1]

Verb→ book •	[0,1]	Scanner
VP → Verb •	[0,1]	Completer
VP → Verb • NP	[0,1]	Completer
S → VP •	[0,1]	Completer
NP → • Det Nominal	[1,1]	Predictor
NP → • ProperNoun	[1,1]	Predictor

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

- When VP → Verb • is itself processed by the Completer, S → VP • is added to Chart[1]
- Last 2 rules in Chart[1] are added by Predictor when VP → Verb • NP is processed
- And so on….

Example

- Book that flight
- We should find… an S from 0 to 3 that is a completed state…

Example: Book that flight

Chart[0]

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 NOMINAL*	[0,0]	Predictor
NP	*Proper-Noun*	[0,0]	Predictor
VP	*Verb*	[0,0]	Predictor
VP	*Verb NP*	[0,0]	Predictor

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
Nominal → Noun
Nominal → Noun Nominal
NP → Proper-Noun
VP → Verb
VP → Verb NP
Nominal → Nominal PP
Det → that | this |a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Proper-Noun → Houston | TWA
Prep → from | to | on

Chart[0]

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 NOMINAL	[0,0]	Predictor
NP	Proper-Noun	[0,0]	Predictor
VP	Verb	[0,0]	Predictor
VP	Verb NP	[0,0]	Predictor

Chart[1]

Verb	book	[0,1]	Scanner
VP	Verb	[0,1]	Completer
S	VP	[0,1]	Completer
VP	Verb NP	[0,1]	Completer
NP	Det NOMINAL	[1,1]	Predictor
NP	Proper-Noun	[1,1]	Predictor

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
Nominal → Noun
Nominal → Noun Nominal
NP → Proper-Noun
VP → Verb
VP → Verb NP
Nominal → Nominal PP
Det → that | this |a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Proper-Noun → Houston | TWA
Prep → from | to | on

Example

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
Nominal → Noun
Nominal → Noun Nominal
NP → Proper-Noun
VP → Verb
VP → Verb NP
Nominal → Nominal PP
Det → that | this |a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Proper-Noun → Houston | TWA
Prep → from | to | on

Chart[1]

Verb	*book*	[0,1]	Scanner
VP	*Verb*	[0,1]	Completer
S	*VP*	[0,1]	Completer
VP	*Verb NP*	[0,1]	Completer
NP	*Det NOMINAL*	[1,1]	Predictor
NP	*Proper-Noun*	[1,1]	Predictor

Chart[2]

Det	*that*	[1,2]	Scanner
NP	*Det NOMINAL*	[1,2]	Completer
NOMINAL	*Noun*	[2,2]	Predictor
NOMINAL	*Noun NOMINAL*	[2,2]	Predictor

Example

Chart[2]

~~*Det*~~	~~*that*~~	~~[1,2]~~	~~Scanner~~
NP	*Det NOMINAL*	[1,2]	Completer
NOMINAL	*Noun*	[2,2]	Predictor
NOMINAL	*Noun NOMINAL*	[2,2]	Predictor

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
Nominal → Noun
Nominal → Noun Nominal
NP → Proper-Noun
VP → Verb

Chart[3]

Noun	*flight*	[2,3]	Scanner
NOMINAL	*Noun*	[2,3]	Complete
NOMINAL	*Noun NOMINAL*	[2,3]	Complete
NP	*Det NOMINAL*	[1,3]	Complete
VP	*Verb NP*	[0,3]	Complete
S	*VP*	[0,3]	Complete
NOMINAL	*Noun*	[3,3]	Predictor
NOMINAL	*Noun NOMINAL*	[3,3]	Predictor

VP → Verb NP
Nominal → Nominal PP
Det → that | this |a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Proper-Noun → Houston | TWA
Prep → from | to | on

Earley Algorithm

- The Earley algorithm has three main functions that do all the work.
  - **Predictor**: Adds predictions into the chart. It is activated when the dot (in a state) is in the front of a non-terminal which is not a part of speech.
  - **Completer**: Moves the dot to the right when new constituents are found. It is activated when the dot is at the end of a state.
  - **Scanner**: Reads the input words and enters states representing those words into the chart. It is activated when the dot (in a state) is in the front of a non-terminal which is a part of speech.
- The Early algorithm uses theses functions to maintain the chart.

Predictor

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

Completer

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

Scanner

procedure SCANNER((A $\rightarrow$ α • B β, [i,j]))

if (B $\in$ PARTS-OF-SPEECH(word[j]) **then**

ENQUEUE((B $\rightarrow$ word[j] • , [j,j+1]), chart[j+1])

end

Enqueue

procedure ENQUEUE(*state,chart-entry*)

if *state* is not already in *chart-entry* **then**

Add *state* at the end of *chart-entry*)

end

Early Code

```
function EARLY-PARSE(words,grammar) returns chart
    ENQUEUE((γ → • 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 PS then
           PREDICTOR(state)
        elseif INCOMPLETE?(state) and NEXT-CAT(state) is a PS then
           SCANNER(state)
        else
           COMPLETER(state)
      end
    end
    return(chart)
```

Retrieving Parse Trees from A Chart

- To retrieve parse trees from a chart, the representation of each state must be augmented with an additional field to store information about the completed states that generated its constituents.
- To collect parse trees, we have to update COMPLETER such that it should add a pointer to the older state onto the list of previous-states of the new state.
- Then, the parse tree can be created by retrieving these list of previous-states (starting from the completed state of S).

Chart[0] - with Parse Tree Info

S0 $\gamma \rightarrow \bullet$ S	[0,0]	[]	Dummy start state
S1 S $\rightarrow \bullet$ NP VP	[0,0]	[]	Predictor
S2 NP $\rightarrow \bullet$ Det NOM	[0,0]	[]	Predictor
S3 NP $\rightarrow \bullet$ ProperNoun	[0,0]	[]	Predictor
S4 S $\rightarrow \bullet$ Aux NP VP	[0,0]	[]	Predictor
S5 S $\rightarrow \bullet$ VP	[0,0]	[]	Predictor
S6 VP $\rightarrow \bullet$ Verb	[0,0]	[]	Predictor
S7 VP $\rightarrow \bullet$ Verb NP	[0,0]	[]	Predictor

Chart[1] - with Parse Tree Info

S8	Verb → book •	[0,1]	[]	Scanner
S9	VP → Verb •	[0,1]	[S8]	Completer
S10	S → VP •	[0,1]	[S9]	Completer
S11	VP → Verb • NP	[0,1]	[S8]	Completer
S12	NP → • Det NOM	[1,1]	[]	Predictor
S13	NP → • ProperNoun	[1,1]	[]	Predictor

Chart[2] - with Parse Tree Info

S14 Det → that •	[1,2]	[]	Scanner
S15 NP → Det • NOM	[1,2]	[S14]	Completer
S16 NOM → • Noun	[2,2]	[]	Predictor
S17 NOM → • Noun NOM	[2,2]	[]	Predictor

Chart[3] - with Parse Tree Info

S18 Noun → flight •	[2,3]	[]	Scanner
S19 NOM → Noun •	[2,3]	[S18]	Completer
S20 NOM → Noun • NOM	[2,3]	[S18]	Completer
S21 NP → Det NOM •	[1,3]	[S14,S19]	Completer
S22 VP → Verb NP •	[0,3]	[S8,S21]	Completer
S23 S → VP •	[0,3]	[S22]	Completer
S24 NOM → • Noun	[3,3]	[]	Predictor
S25 NOM → • Noun NOM	[3,3]	[]	Predictor

Global Ambiguity

$S \rightarrow Verb$ $S \rightarrow Noun$

Chart[0]

S0 $\gamma \rightarrow \bullet S$	[0,0]	[]	Dummy start state
S1 $S \rightarrow \bullet Verb$	[0,0]	[]	Predictor
S2 $S \rightarrow \bullet Noun$	[0,0]	[]	Predictor

Chart[1]

S3 $Verb \rightarrow book \bullet$	[0,1]	[]	Scanner
S4 $Noun \rightarrow book \bullet$	[0,1]	[]	Scanner
S5 $S \rightarrow Verb \bullet$	[0,1]	[S3]	Completer
S6 $S \rightarrow Noun \bullet$	[0,1]	[S4]	Completer

Error Handling

- What happens when we look at the contents of the last table column and don't find a $S \rightarrow \alpha \bullet$ 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

Popup Quiz #1 (7 Minutes)

- Given the following grammar and lexicon
- $S \rightarrow NP\ VP$, $NP \rightarrow N$, $VP \rightarrow V\ NP$,
- $N \rightarrow$ me, $N \rightarrow$ Ahmad, $V \rightarrow$ saw
- Assume the input is:
- Ahmad saw me.
- Show the charts content (states) along with the processors while applying Earley's algorithm to the above input sentence assuming the given grammar.

Thank you

- السلام عليكم ورحمة الله