Chart Parsing: the CYK Algorithm

Informatics 2A: Lecture 17

Mirella Lapata

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

26 October 2011

Grammar Restructuring

Deterministic parsing (e.g., LL(1)) aims to address a limited amount of local ambiguity – the problem of not being able to decide uniquely which grammar rule to use next in a left-to-right analysis of the input string.

By re-structuring the grammar, the parser can make a unique decision, based on a limited amount of look-ahead.

Recursive Descent parsing demands grammar restructuring, in order to eliminate left-recursive rules that can get it into a hopeless loop.

Left Recursion

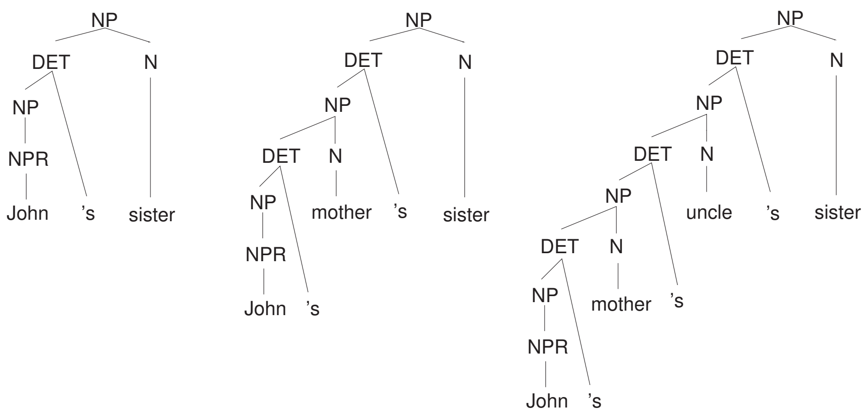
But grammars for natural human languages should be revealing, re-structuring the grammar may destroy this. (Indirectly) left-recursive rules are needed in English.

NP $\longrightarrow$ DET N
NP $\longrightarrow$ NPR
DET $\longrightarrow$ NP 's

These rules generate NPs with possesive modifiers such as:

John's sister
John's mother's sister
John's mother's uncle's sister
John's mother's uncle's sister's niece

Left Recursion



We don't want to re-structure our grammar rules just to be able to use a particular approach to parsing. Need an alternative.

Problems with Parsing as Search

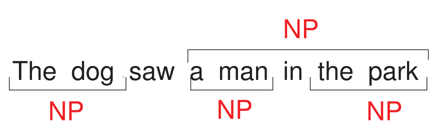
1. A **top-down parser** will do badly if there are many different rules for the same LHS; hopeless for rewriting parts of speech (preterminals) with words (terminals).
2. **A bottom-up parser** does a lot of useless work: locally possible, but globally impossible; inefficient when there is great lexical ambiguity.
3. Both strategies do repeated work by re-analyzing the same sub-string many times!

The next lectures will look at other ways of handling ambiguity:

- Chart parsing: using the parser alone;
- Probabilistic Grammars: using both grammar and parser.

Dynamic Programming

With a CFG, a parser should be able to avoid re-analyzing sub-strings because the analysis of any sub-string is independent of the rest of the parse.



The parser's exploration of its search space can exploit this independence if the parser uses dynamic programming.

Dynamic programming is the basis for all chart parsing algorithms.

Parsing as Dynamic Programming

- Given a problem, systematically fill a table of solutions to sub-problems: this is called memoization.
- Once solutions to all sub-problems have been accumulated, solve the overall problem by composing them.
- For parsing, the sub-problems are analyses of sub-strings and correspond to constituents that have been found.
- Sub-trees are stored in a chart (aka well-formed substring table), which is a record of all the substructures that have ever been built during the parse.

Solves **re-parsing problem**: sub-trees are looked up, not re-parsed!
Solves **ambiguity problem**: chart implicilty stores all parses!

Depicting a Chart

A chart can be depicted as a matrix:

- Rows and columns of the matrix correspond to the start and end positions of a span (ie, starting right before the first word, ending right after the final one);
- A cell in the matrix corresponds to the sub-string that starts at the row index and ends at the column index.
- It can contain information about the type of constituent (or constituents) that span(s) the substring, pointers to its sub-constituents, and/or predictions about what constituents might follow the substring.

Depicting a chart as a Matrix

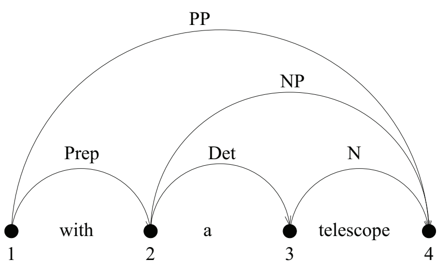
	1	2	3	4	5	6
0	V					
1		Prep		PP		
2			Det	NP		
3				N		
4						
5						

$_0$ *See* $_1$ *with* $_2$ *a* $_3$ *telescope* $_4$ *in* $_5$ *hand* $_6$

Depicting a chart as a Graph

A chart can be also depicted as a graph:

- nodes/vertices represent positions in the text string, starting before the first word, ending after the final word.
- arcs/edges connect vertices at the start and the end of a span to represent a particular substring. Edges can be labelled with the same information as in a cell in the matrix representation.



CYK Algorithm

CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.

- Assumes that the grammar is in CNF (i.e., binarized)
- Rules are restricted to the form $A \longrightarrow BC$ or $A \longrightarrow w$
- Rules have at most two symbols on their RHS

1. $INF\text{-}VP \longrightarrow to\ VP$ $INF\text{-}VP \longrightarrow TOVP$
$TO \longrightarrow to$
2. $NP \longrightarrow Pronoun$ $NP \longrightarrow I|she|me$
3. $S \longrightarrow Aux\ NP\ VP$ $S \longrightarrow X1\ VP$
$X1 \longrightarrow Aux\ NP$

Chart Parsing with the CYK Algorithm

function CKY-Parse(*words*, *grammar*) **returns** *table* **for**

$j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k]$
          $C \in table[k,j]\}$

Chart Parsing with the CYK Algorithm

function CKY-Parse(*words*, *grammar*) **returns** *table* **for**

$j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do** — loop over the columns
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$ — fill bottom cell
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do** — fill row i in column j
    for $k \leftarrow i+1$ **to** $j-1$ **do** — loop over split locations between i and j
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k]$
          $C \in table[k,j]\}$

Check the grammar for rules that link the constituents in $[i,k]$ with those in $[k,j]$. For each rule found store LHS in cell $[i,j]$.

Visualizing the Chart

Grammar Rules in CNF

$S \rightarrow NP\ VP$	$Nominal \rightarrow book \mid flight \mid money$
$S \rightarrow X1\ VP$	$Nominal \rightarrow Nominal\ noun$
$X1 \rightarrow Aux\ VP$	$Nominal \rightarrow Nominal\ PP$
$S \rightarrow book \mid include \mid prefer$	$VP \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$	$VPVerb \rightarrow NP$
$S \rightarrow X2$	$VP \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$	$X2 \rightarrow Verb\ NP$
$S \rightarrow VP\ PP$	$VP \rightarrow Verb\ NP$
$NP \rightarrow TWA \mid Houston$	$VP \rightarrow VP\ PP$
$NP \rightarrow Det\ Nominal$	$PP \rightarrow Preposition\ NP$
$Verb \rightarrow book \mid include \mid prefer$	$Noun \rightarrow book \mid flight \mid money$

Let's parse *Book the flight through Houston*!

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S_1, VP, X2, S_2, VP, S_3 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Clicker Questions about CYK

1. Does the CYK algorithm ever stop before it reaches the end of the string?
   (a) no, this is impossible (b) yes, if string is ungrammatical
2. How does the CYK algorithm show that a string belongs to the language?
   (a) it derives the start symbol (b) it cannot show this
3. Does CYK have to proceed left-to-right? Could one guarantee that no constituent would be missed if CYK proceeded right-to-left?
   (a) no, process is symmetric (b) right-to-left works only for languages read from right to left.
4. Can we tell from the CYK chart what the syntactic analysis (tree structure) is for our sentence?
   (a) sure, this is the whole point! (b) no, chart only recognizes

From CYK Recognizer to CYK Parser

- We just have a chart recognizer, a way of determining whether a string belongs to the language generated by the grammar.
- Changing this to a parser requires recording which existing constituents were combined to make each new constituent.
- This requires another field to record the one or more ways in which a constituent spanning (i,j) can be made from constituents spanning (i,k) and (k,j).

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	[0, 2]	S, VP, X2 [0, 3]	[0, 4]	S_1, VP, X2, S_2, VP, S_3 [0, 5]
	Det [1, 2]	NP [1, 3]	[1, 4]	NP [1, 5]
		Nominal, Noun [2, 3]	[2, 4]	Nominal [2, 5]
			Prep [3, 4]	PP [3, 5]
				NP, Proper-Noun [4, 5]

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	[0, 2]	S, VP, X2 [0, 3]	[0, 4]	S_1, VP, X2, S_2, VP, S_3 [0, 5]
	Det [1, 2]	NP [1, 3]	[1, 4]	NP [1, 5]
		Nominal, Noun [2, 3]	[2, 4]	Nominal [2, 5]
			Prep [3, 4]	PP [3, 5]
				NP, Proper-Noun [4, 5]

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	[0, 2]	S, VP, X2 [0, 3]	[0, 4]	S_1, VP, X2, S_2, VP, S_3 [0, 5]
	Det [1, 2]	NP [1, 3]	[1, 4]	NP [1, 5]
		Nominal, Noun [2, 3]	[2, 4]	Nominal [2, 5]
			Prep [3, 4]	PP [3, 5]
				NP, Proper-Noun [4, 5]

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	[0, 2]	S, VP, X2 [0, 3]	[0, 4]	S_1, VP, X2, S_2, VP, S_3 [0, 5]
	Det [1, 2]	NP [1, 3]	[1, 4]	NP [1, 5]
		Nominal, Noun [2, 3]	[2, 4]	Nominal [2, 5]
			Prep [3, 4]	PP [3, 5]
				NP, Proper-Noun [4, 5]

Summary

- Parsing as search is inefficient and cannot handle ambiguity.
- Alternative: use dynamic programming and memoize sub-analysis in a chart to avoid duplicate work.
- The chart can be visualized as as a matrix.
- The CYK algorithm builds a chart in $O(n^3)$ time. It is specified as a recognizer, but can be used as a parser, if more information is recorded in the chart.

Reading: J&M (2nd ed), Chapter. 13, Sections 13.3–13.4
NLTK Book, Chapter. 8 (*Analyzing Sentence Structure*), Section 8.4

Next lecture: the Early parser or dynamic programming for top-down parsing