Lecture 10: The CKY parsing algorithm

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center
Office Hours: Tuesdays, 2:00-3:00pm

http://cs.illinois.edu/class/fa10/cs498jh
Parsing algorithms for CFGs
Naive top-down parsing

The number of trees is exponential!
Many subtrees are the same.
The right-hand side of a standard CFG can have an arbitrary number of symbols (terminals and nonterminals):

\[ VP \rightarrow \text{ADV eat NP} \]

A CFG in **Chomsky Normal Form** (CNF) allows only two kinds of right-hand sides:
- **Two nonterminals**: \( VP \rightarrow \text{ADV VP} \)
- **One terminal**: \( VP \rightarrow \text{eat} \)

Any CFG can be transformed into an equivalent CNF:

\[ VP \rightarrow \text{ADVP } VP_1 \]
\[ VP_1 \rightarrow VP_2 \text{ NP} \]
\[ VP_2 \rightarrow \text{eat} \]
A note about $\epsilon$-productions

Formally, context-free grammars are allowed to have $c$ ($\epsilon =$ the empty string):

\[ \text{VP} \to V \text{ NP} \quad \text{NP} \to \text{DT Noun} \quad \text{NP} \to \epsilon \]

These can always be eliminated without changing the language generated by the grammar:

\[ \text{VP} \to V \text{ NP} \quad \text{NP} \to \text{DT Noun} \quad \text{NP} \to \epsilon \]

becomes

\[ \text{VP} \to V \epsilon \quad \text{NP} \to \text{DT Noun} \]

which in turn becomes

\[ \text{VP} \to V \text{ NP} \quad \text{VP} \to V \epsilon \quad \text{NP} \to \text{DT Noun} \]

We will assume that our grammars don’t have $\epsilon$-productions
CKY chart parsing algorithm

Bottom-up parsing:
start with the words

Dynamic programming:
save the results in a table/chart
re-use these results in finding larger constituents

Complexity: $O(n^3|G|)$
$n$: length of string, $|G|$: size of grammar

Presumes a CFG in Chomsky Normal Form:
Rules are all either $A \rightarrow BC$ or $A \rightarrow a$
(with $A,B,C$ nonterminals and $a$ a terminal)
The CKY parsing algorithm

S → NP VP
VP → V NP
V → eat
NP → we
NP → sushi

We eat sushi
CKY: filling the chart

CS 498 JH: Introduction to NLP
CKY: filling one cell

chart[2][6]:

$w_1 \ W_2 \ W_3 \ W_4 \ W_5 \ W_6 \ W_7$

chart[2][6]:

$w_1 \ W_2 W_3 W_4 W_5 W_6 W_7$

chart[2][6]:

$w_1 \ W_2 W_3 W_4 W_5 W_6 W_7$

chart[2][6]:

$w_1 \ W_2 W_3 W_4 W_5 W_6 W_7$

CS 498 JH: Introduction to NLP
Cocke Kasami Younger

ckyParse(Array words):
  n = length(words)
  initChart(n)
  fillChart(n)

initChart(int n):
  chart = n x n array of cells
  for i = 1...n:
    initCell(i)

initCell(int i):
  for c in POSTags(words[i]):
    addToCell(chart[i][i], c, null, null)

addToCell(cell, Parent, Left, Right)
  if (cell.hasEntry(Parent)):
    P = cell.getEntry(Parent)
    P.addBackpointers(Left, Right)
  else
    cell.addEntry(Parent, Left, Right)

fillChart(n):
  for span = 1...n-1:
    for i = 1...n-span:
      fillCell(i, i+span)

fillCell(i,j):
  for k = i..j-1:
    combineCells(i, k, j)

combineCells(i,k,j):
  for Y in cell[i][k]:
    for Z in cell[k+1][j]:
      for X in Nonterminals:
        if X → Y Z in Rules:
          addToCell(cell[i][j], X, Y, Z)
Today’s key concepts

Parsing with CFGs:
  Chomsky Normal Form
  CKY parsing