Parsing Algorithms: Charts

Solutions to parsing problems

- Modern parsing algorithms have three key ideas:
  - Solve the problem of performance with chart parsers
  - Solve the problems of pre-defining CFG or other grammars by using Treebanks and statistical parsing
  - Partially solve the problems of correctly choosing the best parse trees by using lexicalization (information about words from the Treebank)

Parsing Algorithms

- The simple parsers that we have seen are exponential in time (recursive descent with back-tracking) and (shift reduce with back-tracking) with respect to the length of the input
- Avoid back-tracking and re-doing subtrees
  - Recall that the backtracking recursive descent expanded some subtrees multiple times
  - Use charts to record subtrees to avoid redundant computation
- Use forms of dynamic programming to search for good parse trees
  - Attempt to perform exponential process in polynomial time

Binarization reduces exponential process

- Where binarization means only reducing rules with 2 right hand side (RHS) symbols
  - Allows 2 dimensional charts
- All CFG grammars have a Chomsky Normal Form where every rule has no more than 2 symbols on the RHS
  - Example grammar rule with 3 RHS symbols:

    VP -> Verb NP PP
  - Transformed to equivalent grammar with only 2 RHS symbols:

    VP -> Verb NPtemp

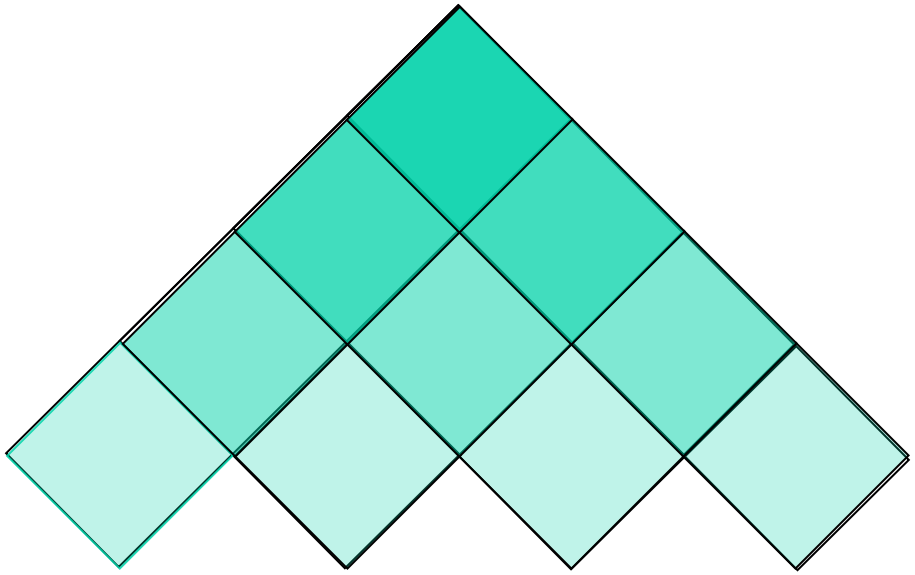
    Nptemp -> NP PP

Chart Parsers

- CKY (Cocke-Kasami-Younger) algorithm is an example
  - Bottom-up parser (but can also have top down chart parsers)
  - Requires grammar to be in Chomsky Normal Form, with only two symbols on the right-hand-side of each production
  - Fills in a data structure called a chart or a parse triangle
    - Other parsers, such as Earley's algorithm, use similar chart ideas to work on two subtrees at a time
- Key idea in parser development from 1970 - 1990

CKY Parsing

- For input of length n, fills a parse table triangle of size (n, n) , where each element has the non-terminal production representing the span of text from position i to j.
  - Cells in first (bottom) layer describe trees of single words
  - Cells in second layer describes how rewrite rules can be used to combine trees in first layer for trees with two words
  - Etc.



S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0
N → *people*	0.5
N → *fish*	0.2
N → *tanks*	0.2
N → *rods*	0.1
V → *people*	0.1
V → *fish*	0.6
V → *tanks*	0.3
P → *with*	1.0

	fish	people	fish	tanks
0	N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.0049 VP → V NP 0.105 S → VP 0.0105	NP → NP NP 0.0000686 VP → V NP 0.00147 S → NP VP 0.000882	NP → NP NP 0.0000009604 VP → V NP 0.00002058 S → NP VP 0.00018522
1		N → people 0.5 V → people 0.1 NP → N 0.35 VP → V 0.01 S → VP 0.001	NP → NP NP 0.0049 VP → V NP 0.007 S → NP VP 0.0189	NP → NP NP 0.0000686 VP → V NP 0.000098 S → NP VP 0.01323
2			N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.00196 VP → V NP 0.042 S → VP 0.0042
3				N → tanks 0.2 V → tanks 0.1 NP → N 0.14 VP → V 0.03 S → VP 0.003
4				

Call buildTree(score, back) to get the best parse

Example showing filled-in CKY chart for a PCFG
for sentence “fish people fish tanks” from Chris Manning