NLP
Introduction to NLP

Classic parsing methods
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
Parsing as search

• There are two types of constraints on the parses
  – From the input sentence
  – From the grammar

• Therefore, two general approaches to parsing
  – Top-down
  – Bottom-up
Top down parsing

S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
Top down parsing

S \rightarrow NP \ VP

NP \rightarrow DT \ N \mid NP \ PP
PP \rightarrow PRP \ NP
VP \rightarrow V \ NP \mid VP \ PP
DT \rightarrow 'a' \mid 'the'
N \rightarrow 'child' \mid 'cake' \mid 'fork'
PRP \rightarrow 'with' \mid 'to'
V \rightarrow 'saw' \mid 'ate'
Top down parsing

S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
Top down parsing

S ← NP VP
NP ← DT N | NP PP
PP ← PRP NP
VP ← V NP | VP PP
DT ← 'a' | 'the'
N ← 'child' | 'cake' | 'fork'
PRP ← 'with' | 'to'
V ← 'saw' | 'ate'
Top down parsing

S \rightarrow NP \ VP
NP \rightarrow DT \ N \mid NP \ PP
PP \rightarrow PRP \ NP
VP \rightarrow V \ NP \mid VP \ PP
DT \rightarrow 'a' \mid 'the'
N \rightarrow 'child' \mid 'cake' \mid 'fork'
PRP \rightarrow 'with' \mid 'to'
V \rightarrow 'saw' \mid 'ate'
Top down parsing

S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'

The child ate the cake with the fork.
Bottom up vs. top down methods

• Bottom up
  – explores options that won’t lead to a full parse
  – Example: shift–reduce (srparser in nltk)
  – Example: CKY (Cocke–Kasami–Younger)

• Top down
  – explores options that don’t match the full sentence
  – Example: recursive descent (rdparser in nltk)
  – Example: Earley parser

• Dynamic programming
  – caches of intermediate results (memoization)
Recursive Descent Parser

- In nltk

```python
>>> from nltk.app import rdparser;
>>> rdparser()
```
Introduction to NLP

Shift-Reduce Parsing
Shift-Reduce Parsing

• A bottom-up parser
  – Tries to match the RHS of a production until it can build an S
• Shift operation
  – Each word in the input sentence is pushed onto a stack
• Reduce–n operation
  – If the top $n$ words on the top of the stack match the RHS of a production, then they are popped and replaced by the LHS of the production
• Breadth-first search
• Stopping condition
  – The process stops when the input sentence has been processed and S has been popped from the stack
Shift-Reduce Parsing Example

[S [ * the child ate the cake]
R [ DT * child ate the cake]
S [ DT 'child' * ate the cake]
R [ DT N * ate the cake]
R [ NP * ate the cake]
S [ NP 'ate' * the cake]
R [ NP V * the cake]
S [ NP V 'the' * cake]
R [ NP V DT * cake]
S [ NP V DT 'cake' * ]
R [ NP V DT N * ]
R [ NP V NP * ]
R [ NP VP * ]
R [ S * ]
(S (NP (DT the) (N child)) (VP (V ate) (NP (DT the) (N cake)))))
Shift-Reduce Parsing

- In nltk

```python
>>> from nltk.app import srparser;
>>> srparser()
```