WSTA Lectures 18 & 19
Grammars and Parsing

- **Some Applications**
  - Grammar checking
  - Machine translation
  - Dialogue systems
  - Summarization

- **Sources of complexity**
  - Size of search space
  - No independent source of knowledge about the underlying structures
  - Lexical and structural ambiguity

---

Slide credits: Steven Bird
Overview

1. Chomsky hierarchy
   - RG, CFG, CSG; rule formalisms & automata
   - where is natural language on this hierarchy?

2. Syntax
   - constituency, parse trees
   - Penn Treebank syntactic tags

3. Phrase Structure Grammar
   - productions, syntactic ambiguity, attachment

4. Parsing algorithms
   - top-down, bottom-up, chart parsing
1. The Chomsky Hierarchy

- **Type 3: REGULAR (FSA)**
  - $A \rightarrow xB$, $A \rightarrow xyB$, $A \rightarrow x$

- **Type 2: CONTEXT FREE (PDA)**
  - $A \rightarrow BCD$, $A \rightarrow BxDy$, $A \rightarrow x$

- **Type 1: CONTEXT SENSITIVE (LBA)**
  - $xAy \rightarrow xay$, $xAy \rightarrow xBCy$, $A \rightarrow x$

- **Type 0: RECURSIVELY ENUMERABLE (TM)**
  - ?? $\rightarrow$ ??
Is English a Regular Language?

- **Arguments against:**
  - $XX^R$: "The sentence $S$, said in reverse, is $S^R$"
  - "This is the [something] that $S$" e.g.,
    "This is the rat that ate the cheese
    That lay in the house that Jack built."
  - $NP^n (\text{transitive-verb})^{n-1} \text{ VP}$

- **Arguments for:**
  - depth restriction (<3)
2. Syntax

the part of a grammar that represents a speaker's knowledge of the structure of phrases and sentences

Why word order is significant:

- may have no effect on meaning
  - Jack Horner stuck in his thumb
    Jack Horner stuck his thumb in

- may change meaning
  - Salome danced for Herod
    Herod danced for Salome

- may render a sentence ungrammatical
  - *for danced Herod Salome
Syntax (cont)

1. The farmer loaded sand into the cart
   The farmer loaded the cart with sand

2. The farmer dumped sand into the cart
   *The farmer dumped the cart with sand

3. *The farmer filled sand into the cart
   The farmer filled the cart with sand
Syntax (cont)

Sentence Types

- **Simple sentences (single verb group / clause):**
  - Her uncle *had put* the gifts in the car

- **Coordinate sentences (two clauses conjoined):**
  - Denise bought a new coat, *but* she didn't wear it often

- **Complex sentences (embedded clauses):**
  - Sue said *Danny fell*

- **Other constructions:**
  - Conditionals, passives, interrogatives, clefts, topicalization

- **written vs spoken language:**
  - harder: disfluencies, filled pauses
  - easier: intonation may remove syntactic ambiguities
Syntactic Constituency

1. Ability to stand alone: *exclamations and answers*
   - *What do many executives do?*
     - Eat at really fancy restaurants
   - *Do fancy restaurants do much business?*
     - *Well, executives eat at*

2. Substitution by a pro-form: *pronouns, pro-verbs (do, be, have), pro-adverbs (there, then), pro-adjective (such)*
   - *Many executives do*

3. Movement: *fronting or extrapositing a fragment*
   - *At really fancy restaurants, many executives eat*
     - *Fancy restaurants many executives eat at really*
Constituency: Tree diagrams

S
  NP
    JJ JJ NN CC NN NNS MD
    Ordinary daily multivitamin and mineral supplements could
  VP
    VB NP PP VP
    help NNS IN NP VB RP NP
    adults with diabetes fight off some minor infections
Major Syntactic Constituents

- **Noun Phrase (NP)**
  - referring expressions

- **Verb Phrase (VP)**
  - predicating expressions

- **Prepositional Phrase (PP)**
  - direction, location, etc

- **Adjectival Phrase (AdjP)**
  - modified adjectives (e.g. "really fancy")

- **Adverbial Phrase (AdvP)**

- **Complementizers (COMP)**
Constituency: Further Issues

- Constituency is relative to the sentence in question
  - Pat and Leslie raised llamas
  - Robin raised Pat and Leslie adopted Chris

- Constituency is hierarchical
  - i.e. no overlapping of constituents

- Representing constituency: Tree diagrams
  - nltk.tree module, defines Tree and TreeToken
    >>> tree = Tree('NP', ['John'])
    ('NP': 'John')
    >>> tree.node
    'NP'
    >>> tree[0]
    'John'
  - Other methods: len(tree), height(), leaves(), draw()
(S (S-TPC-1
   (NP-SBJ (NP (NP A form) (PP of (NP asbestos)))
   (RRC (ADVP-TMP once)
      (VP used (NP *) (S-CLR (NP-SBJ *)
      (VP to (VP make
      (NP Kent cigarette filters)))))))))
(VP has (VP caused
   (NP (NP a high percentage)
   (PP of (NP cancer deaths))
   (PP-LOC among
   (NP (NP a group)
   (PP of (NP
   (NP workers)
   (RRC (VP exposed (NP *)
   (PP-CLR to (NP it))
   (ADVP-TMP (NP (QP more than 30) years) ago)))))))))
(NP-SBJ researchers) (VP reported (SBAR 0 (S *T*-1))).)
>>> import nltk

>>> t = nltk.corpus.treebank.parsed_sents() [0]

Tree('S', [Tree('NP-SBJ', [Tree('NP', [Tree('NNP', ['Pierre']), Tree('NNP', ['Vinken'])]), …

>>> t.draw()
Treebank Phrase Tags

- **ADJP - Adjective phrase**
  - Phrasal category headed by an adjective (including comparative and superlative adjectives).
  - *outrageously expensive*

- **ADVP - Adverb phrase.**
  - Phrasal category headed by an adverb (including comparative and superlative adverbs).
  - *rather timidity, very well indeed.*

- **NP - Noun phrase.**
  - Phrasal category that includes all constituents that depend on a head noun.
Treebank Phrase Tags (cont)

- **PP** - Prepositional phrase.
  - Phrasal category headed by a preposition.
- **S** - Simple declarative clause
- **SBAR**: Clause introduced by a (possibly empty) subordinating conjunction.
- **VP** - Verb phrase: Phrasal category headed a verb.
- **WHNP** - Wh-noun phrase. Noun phrase containing a wh-determiner, as in *which book* or *whose daughter*, or consisting of a wh-pronoun like *who*. 
The Structure of Noun Phrases

- **Determiners and adjectives, agreement**
  - All kind women (DT JJ NN)
  - The kind but impatient woman (DT JJ CC JJ NN)
  - The other cheap direct first-class flight
  - One flight, Three flights

- **Prepositional phrases**
  - The flight from London on Thursday (DT NN PP PP)

- **Relative clauses**
  - The woman who John saw 0 yesterday (DT NN S'[gap])
  - The woman who 0 saw John (DT NN S'[gap])
The Structure of Verb Phrases

Overview:

- **Major verb classes**
  - intransitive, transitive, ditransitive, sentential complement
- **Complementation patterns**
  - e.g. NP PP[for] PP[with]: open, repair, fix
- **Arguments vs adjuncts**
  - Arguments: subject, direct object, indirect object
  - Adjuncts: optional modifiers
Major Verb Classes

- **Intransitive (NP V)**
  - run, walk, sleep, sigh, sneeze

- **Transitive (NP V NP)**
  - buy, meet, kill, throw, see

- **Ditransitive (NP V NP NP)**
  - give, sell, tell

- **Verbs with sentential complements (NP V S’)**
  - believe, ask, say

- **Key term: valency**
Complementation Patterns

Complement = arguments of the verb other than the subject

- NP PP[for]: buy, reserve
- NP PP[loc]: put, place, stand
- PP[to] PP[about]: talk, speak

Complement structure

- The “signature” of the verb
- Determines appropriate syntactic trees
- Serves as a bridge to underlying semantic structure
Complements vs Adjuncts

Complements (arguments)
- central to the activity of the verb
- subject, direct object, indirect object

Adjuncts
- optional, can usually be moved without changing meaning
- She saw the Woody Allen movie... yesterday / in Paris / with two friends

Distinguishing them depends on the verb
- He put the chair on the stage
  She gave her presentation on the stage
3. Phrase Structure Grammar

Grammaticality:

- doesn't depend on:
  - having heard the sentence before
  - the sentence being true (*I saw a unicorn yesterday*)
  - the sentence being meaningful
    
    
    (colorless green ideas sleep furiously vs
    *furiously sleep ideas green colorless*)
  - learned rules of grammar

- a formal property that we can investigate and model
Recursive Grammars

- set of well formed English sentences is infinite
  - no *a priori* length limit
  - Sentence from A.A. Milne (next slide)

- a grammar is a finite-statement about well-formedness
  - it has to involve iteration or recursion

- examples of recursive rules
  - NP → NP PP (in a single rule)
  - NP → S, S → NP VP (recursive pair)

- therefore search is over a possibly infinite set
Recursive Grammars (cont)

You can imagine Piglet's joy when at last the ship came in sight of him. In after-years he liked to think that he had been in Very Great Danger during the Terrible Flood, but the only danger he had really been in was the last half-hour of his imprisonment, when Owl, who had just flown up, sat on a branch of his tree to comfort him, and told him a very long story about an aunt who had once laid a seagull's egg by mistake, and the story went on and on, rather like this sentence, until Piglet who was listening out of his window without much hope, went to sleep quietly and naturally, slipping slowly out of the window towards the water until he was only hanging on by his toes, at which moment, luckily, a sudden loud squawk from Owl, which was really part of the story, being what his aunt said, woke the Piglet up and just gave him time to jerk himself back into safety and say, "How interesting, and did she?" when -- well, you can imagine his joy when at last he saw the good ship, Brain of Pooh (Captain, C. Robin; 1st Mate, P. Bear) coming over the sea to rescue him...

A.A. Milne: *In which Piglet is Entirely Surrounded by Water*
Productions and Local Trees

- The form of CFG productions:
  - $A \rightarrow BCD$, $A \rightarrow BxDy$, $C \rightarrow x$

- A production *licenses a local tree*
Verb Phrases

- **Overgeneralization:**
  - \( VP \rightarrow V \ NP^* \ PP^* \)

- **Productions sensitive to gross verb classes:**
  - \( VP \rightarrow IV \)
  - \( VP \rightarrow TV \ NP \)
  - \( VP \rightarrow DV \ NP \ NP \)

- **Productions sensitive to verb subcategorization:**
  - \( VP \rightarrow V_{TALK} \ PP_{TO} \ PP_{ABOUT} \)
  - \( V_{TALK} \rightarrow 'talk', 'speak', 'chat' \)
  - \( PP_{TO} \rightarrow 'to' NP \)
Trees from Local Trees

- A tree is just a set of connected local trees
- Each local tree is licensed by a production
- Each production is included in the grammar
- The fringe of the tree is a given sentence
- Parsing = discovering the tree(s) for a given sentence
- A SEARCH PROBLEM
Syntactic Ambiguity 1

- I saw the man in the park with a telescope
  - several "readings"
  - attachment ambiguity
Syntactic Ambiguity 2

- **How do we avoid this?**
  - production RHS: VP ... PP, NP ... PP are the problem
  - they are independently motivated
  - ambiguity is genuine (so why avoid it?)
    - the tuna can hit the boat
  - yet how do we explain preferences?
    - the cop saw the burglar with the binoculars
    - the cop saw the burglar with the gun
  - how do we explain garden path effects?
    - the horse raced past the barn fell
    - we painted all the walls with cracks
    - after the man left the shop closed
Syntactic Ambiguity 3

- **Lexical preferences:**
  - I wanted the dog in the house (NP adjunct)
  - I kept the dog in the house (VP adjunct)
  - I put the dog in the house (VP argument)

- **Empirical study:**
  - load treebank data
  - search for PPs; where are they attached?
  - report some aspect of the context
    - verb, object, preposition
  - can you find any good predictors of attachment?
  - Another data source: ppattach corpus (28k examples)
Grammars

S -> NP, VP

VP -> V, NP

NP -> Det, N, PP

NP -> 'I'

Det -> 'the'

V -> 'saw'

P -> 'with'

N -> 'dog'

NP -> Det, N

VP -> V, NP, PP

VP -> V, NP

PP -> P, NP

PP -> P, NP

N -> 'man'

Det -> 'a'

P -> 'in'

N -> 'park'

N -> 'telescope'
Kinds of Parsing

- Top down, Bottom up
- Chart parsing
- Chunk parsing (recall IOB chunk tagging last week)
Top-Down Parsing (Recursive Descent Parsing)

\[\text{parse}(\text{goal, sent}):\]

- if goal and string are empty we're done, else
- is the first element of the goal the same as the first element in the string?
  - if so, strip off these first elements and continue processing
  - otherwise, check if any of the rule LHSs match the first element of the goal
    - if so, replace this element with the RHS of the rule
    - do this for all rules
    - new continue with the new goal

- Demonstration
Bottom-Up Parsing

parse(sent):
- if sent is [S] then finish
- otherwise, for every rule, check if the RHS of the rule matches any substring of the sentence
- if it does, replace the substring the the LHS of the rule
- continue with this sentence

Demonstration
Issues and Solutions

- **top-down parsing:**
  - wasted processing hypothesizing words and phrases (relevant lexical items are absent), repeated parsing of subtrees
  - infinite recursion on left-recursive rules (transforming the grammar)

- **bottom-up parsing:**
  - builds sequences of constituents that top-down parsing will never consider

- **solutions:**
  - BU to find categories of lexical items, then TD
  - left-corner parsing (bottom-up filtering)
Chart Parsing

1. Problems with naive parsers
2. Tokens and charts
3. Productions, trees and charts
4. Chart Parsers
5. Adding edges to the chart
6. Rules and strategies
7. Demonstration
Issues and Solutions

- **top-down parsing:**
  - wasted processing hypothesizing words and phrases (relevant lexical items are absent), repeated parsing of subtrees
  - infinite recursion on left-recursive rules (transforming the grammar)

- **bottom-up parsing:**
  - builds sequences of constituents that top-down parsing will never consider

- **solutions:**
  - BU to find categories of lexical items, then TD
  - left-corner parsing (bottom-up filtering)

- **More general, flexible solution:** dynamic programming
Tokens and Charts

- An input sentence can be stored in a chart
  - Sentence = list of tokens
  - Token = (type, location) -> Edge

- E.g. [l_{0:1}, saw_{1:2}, the_{2:3}, dog_{3:4}]
  - NLTK: ['I'@[0:1], 'saw'@[1:2], 'the'@[2:3], 'dog'@[3:4]]
  - Abbrev: ['l'@[0], 'saw'@[1], 'the'@[2], 'dog'@[3]]

- Chart representation:
Productions, Trees & Charts

- **Productions:**
  - \( A \rightarrow BCD, \ C \rightarrow x \)

- **Trees:**

  \[
  \begin{array}{c}
  A \\
  B \\
  C \\
  D \\
  \end{array}
  \begin{array}{c}
  C \\
  \end{array}
  \begin{array}{c}
  x \\
  \end{array}
  \]

- **Charts:**

  - **Nonterminals:** \( A \)
  - **Pre-terminal:** \( C \)
  - **Terminal:** \( x \)
Edges and “Dotted” Productions

- Edges decorated with dotted production and tree

1. $A \rightarrow \cdot BCD$

2. $A \rightarrow B\cdot CD$

3. $A \rightarrow BC\cdot D$

4. $A \rightarrow BCD\cdot$

- Partial vs complete edges; zero-width edges
Charts and Chart Parsers

- **Chart:**
  - collection of edges

- **Chart parser:**
  - Consults three sources of information:
    - Grammar
    - Input sentence
    - Existing chart
  - **Action:**
    - Add more edges to the chart
    - Report any completed parse trees

- Three ways of adding edges to the chart...
Adding Edges to the Chart

1. Adding LeafEdges

2. Adding self loops
Adding Edges to the Chart (cont)

3. Adding “fundamental rule” edges

A → BC•D  D → EF•

A
B C

D
E F

B C E F

A → BCD•

B C D

E F

B C E F
Chart Rules: Bottom-Up Rule

- **Bottom-Up Rule:**
  - For each complete edge C, set X = LHS of production
  - For each grammar rule with X as first element on RHS
  - Insert zero-width edge to left of C

Bottom Up Init Rule | [--] . . . . . . | 'I'.
Bottom Up Init Rule | .  [--] . . . . . | 'saw'.
Bottom Up Init Rule | .  .  [--] . . . . | 'the'.
Bottom Up Init Rule | .  .  .  [--] . . . | 'dog'.
Bottom Up Init Rule | .  .  .  .  [--] . . | 'with'.
Bottom Up Init Rule | .  .  .  .  .  [--] | 'my'.
Bottom Up Init Rule | .  .  .  .  .  .  [-] | 'cookie'.
Bottom Up Rule | . . . . . > . | N -> * 'cookie'
Bottom Up Rule | . . . . . > . | Det -> * 'my'
Bottom Up Rule | . . . . . > . | P -> * 'with'
Bottom Up Rule | . . . > . . . . | N -> * 'dog'
Bottom Up Rule | . . > . . . . . | Det -> * 'the'
Bottom Up Rule | . > . . . . . . | V -> * 'saw'
Bottom Up Rule | > . . . . . . . | NP -> * 'I'
Chart Rules: Top-Down Rules

**Top down initialization:**
- For every production whose LHS is the base category:
  create the corresponding dotted rule
  put dot position at the start of RHS

Top Down Init Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{S} \rightarrow * \text{NP VP}$

**Top down expand rule:**
- For each production and for each incomplete edge:
  if the expected constituent matches the production:
  insert zero-width edge with this production on right

Top Down Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{NP} \rightarrow * 'I'$
Top Down Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{NP} \rightarrow * \text{Det N}$
Top Down Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{NP} \rightarrow * \text{NP PP}$
Top Down Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{Det} \rightarrow * 'the'$
Top Down Rule $\Rightarrow \ldots \ldots \ldots \ldots \ldots \ldots \ldots | \text{Det} \rightarrow * 'my'$
Rules, Strategies, Demo

- **Fundamental rule:**
  - For each pair of edges $e_1$ and $e_2$:
    - If $e_1$ is incomplete and its expected constituent is $X$:
      - If $e_2$ is complete and its LHS is $X$:
        - Add $e_3$ spanning both $e_1$ and $e_2$, with dot moved right

- **Parsing Strategies:**
  - [TopDownInitRule, TopDownExpandRule, FundamentalRule]
  - [BottomUpRule, FundamentalRule]

- **Demonstration:**
  - `python nltk/draw/chart.py`
Reading

- JM chapters 10 (parsing) and 9 (grammars)
- NLTK book chapter 8