Roadmap

- Motivation:
  - Recognition and Analysis

- Parsing as Search
  - Search algorithms
  - Top-down parsing
  - Bottom-up parsing
  - Issues: Ambiguity, recursion, garden paths
Parsing

- CFG parsing is the task of assigning proper trees to input strings
- For any input $A$ and a grammar $G$, assign (zero or more) parse-trees $T$ that represent its syntactic structure, and
  - Cover all and only the elements of $A$
  - Have, as root, the start symbol $S$ of $G$
Parsing

- CFG parsing is the task of assigning proper trees to input strings
- For any input $A$ and a grammar $G$, assign (zero or more) parse-trees $T$ that represent its syntactic structure, and
  - Cover all and only the elements of $A$
  - Have, as root, the start symbol $S$ of $G$
    - Do not necessarily pick one (or correct) analysis
Parsing

- CFG parsing is the task of assigning proper trees to input strings
  - For any input A and a grammar G, assign (zero or more) parse-trees T that represent its syntactic structure, and
    - Cover all and only the elements of A
    - Have, as root, the start symbol S of G
      - Do not necessarily pick one (or correct) analysis

- Recognition:
  - Subtask of parsing
  - Given input A and grammar G, is A in the language defined by G or not
Motivation

• Parsing goals:
  • Is this sentence in the language – is it grammatical?  
    *I prefer United has the earliest flight.*
  • FSAs accept the regular languages defined by automaton
  • Parsers accept language defined by CFG
Motivation

- Parsing goals:
  - Is this sentence in the language – is it grammatical? *I prefer United has the earliest flight.*
  - FSAs accept the regular languages defined by automaton
  - Parsers accept language defined by CFG

- What is the syntactic structure of this sentence?
  - *What airline has the cheapest flight?*
  - *What airport does Southwest fly from near Boston?*
  - Syntactic parse provides framework for semantic analysis
    - What is the subject?
Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:
Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input.

- Formally, search problems are defined by:
  - A start state $S$, 
Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input.

- Formally, search problems are defined by:
  - A start state $S$,
  - A goal state $G$,
Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input.
- Formally, search problems are defined by:
  - A start state $S$,
  - A goal state $G$,
  - A set of actions, that transition from one state to another
    - Successor function
Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input.

- Formally, search problems are defined by:
  - A start state $S$,
  - A goal state $G$,
  - A set of actions, that transition from one state to another
    - Successor function
  - A path cost function
Parsing as Search

- The parsing search problem (one model):
  - Start State S:
Parsing as Search

• The parsing search problem (one model):
  • Start State S: Start Symbol

• Goal test:
Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol

- Goal test:
  - Does parse tree cover all and only input?

- Successor function:
Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol

- Goal test:
  - Does parse tree cover all and only input?

- Successor function:
  - Expand a non-terminal using production in grammar where non-terminal is LHS of grammar
Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol

- Goal test:
  - Does parse tree cover all and only input?

- Successor function:
  - Expand a non-terminal using production in grammar where non-terminal is LHS of grammar

- Path cost:
  - We’ll ignore here
Parsing as Search

- Node:
Parsing as Search

- Node:
  - Partial solution to search problem:
    - Partial parse

- Search start node:
  - Initial state:
Parsing as Search

- **Node:**
  - Partial solution to search problem:
    - Partial parse

- **Search start node:**
  - Initial state:
    - Input string
    - Start symbol of CFG

- **Goal node:**
Parsing as Search

- **Node:**
  - Partial solution to search problem:
    - Partial parse

- **Search start node:**
  - Initial state:
    - Input string
    - Start symbol of CFG

- **Goal node:**
  - Full parse tree: covering all and only input, rooted at S
Search Algorithms

- Many search algorithms
  - Depth first
Search Algorithms

• Many search algorithms
  • Depth first
    • Keep expanding non-terminal until reach words
      • If no more expansions, back up
  
• Breadth first
Search Algorithms

- Many search algorithms
  - Depth first
    - Keep expanding non-terminal until reach words
    - If no more expansions, back up

- Breadth first
  - Consider all parses with a single non-terminal expanded
    - Then all with two expanded and so

- Other alternatives if have associated path costs
Parse Search Strategies

- Two constraints on parsing:
  - Must start with the start symbol
  - Must cover exactly the input string

- Correspond to main parsing search strategies
  - Top-down search (Goal-directed search)
  - Bottom-up search (Data-driven search)
A Grammar

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>$Det \rightarrow that</td>
</tr>
<tr>
<td>$S \rightarrow Aux\ NP\ VP$</td>
<td>$Noun \rightarrow book</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$Verb \rightarrow book</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$Pronoun \rightarrow I</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$Proper-Noun \rightarrow Houston</td>
</tr>
<tr>
<td>$NP \rightarrow Det\ Nominal$</td>
<td>$Aux \rightarrow does$</td>
</tr>
<tr>
<td>$Nominal \rightarrow Noun$</td>
<td>$Preposition \rightarrow from</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal\ Noun$</td>
<td></td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow VP\ PP$</td>
<td></td>
</tr>
<tr>
<td>$PP \rightarrow Preposition\ NP$</td>
<td></td>
</tr>
</tbody>
</table>
Top-down Search

- All valid parse trees must start with start symbol
- Begin search with productions with S on LHS
  - E.g., S -> NP VP
Top-down Search

- All valid parse trees must start with start symbol
  - Begin search with productions with S on LHS
    - E.g., S -> NP VP

- Successively expand non-terminals
  - E.g., NP – Det Nominal; VP -> V NP
Top-down Search

- All valid parse trees must start with start symbol
  - Begin search with productions with S on LHS
    - E.g., S -> NP VP

- Successively expand non-terminals
  - E.g., NP – Det Nominal; VP -> V NP

- Terminate when all leaves are terminals
  - *Book that flight*
Top-down Search
Depth-first Search
Depth-first Search
Depth-first Search
Breadth-first Search
Breadth-first Search
Breadth-first Search
Breadth-first Search
Pros and Cons of Top-down Parsing

- Pros:
Pros and Cons of Top-down Parsing

- Pros:
  - Doesn’t explore trees not rooted at S
Pros and Cons of Top-down Parsing

Pros:
- Doesn’t explore trees not rooted at S
- Doesn’t explore subtrees that don’t fit valid trees
Pros and Cons of Top-down Parsing

- **Pros:**
  - Doesn’t explore trees not rooted at S
  - Doesn’t explore subtrees that don’t fit valid trees

- **Cons:**
  - Produces trees that may not match input
Pros and Cons of Top-down Parsing

- **Pros:**
  - Doesn’t explore trees not rooted at S
  - Doesn’t explore subtrees that don’t fit valid trees

- **Cons:**
  - Produces trees that may not match input
  - May not terminate in presence of recursive rules
Pros and Cons of Top-down Parsing

**Pros:**
- Doesn’t explore trees not rooted at S
- Doesn’t explore subtrees that don’t fit valid trees

**Cons:**
- Produces trees that may not match input
- May not terminate in presence of recursive rules
- May rederive subtrees as part of search
Bottom-Up Parsing

- Try to find all trees that span the input
- Start with input string
  - Book that flight.
Bottom-Up Parsing

- Try to find all trees that span the input
  - Start with input string
    - Book that flight.

- Use all productions with current subtree(s) on RHS
  - E.g., N -> Book; V -> Book
Bottom-Up Parsing

- Try to find all trees that span the input
  - Start with input string
    - Book that flight.

- Use all productions with current subtree(s) on RHS
  - E.g., N -> Book; V -> Book

- Stop when spanned by S (or no more rules apply)
Bottom-Up Search

Book that flight
Bottom-Up Search

Verb | Det | Noun
---|---|---
Book | that | flight
Bottom-Up Search
Bottom-Up Search

NP
  ____________
  |       |       |
  Verb   Det   Noun
  |       |       |
  Book   that flight
Bottom-Up Search
Bottom-Up Search
Bottom-Up Search
Bottom-Up Search
Pros and Cons of Bottom-Up Search

- Pros:
Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don’t match input
Pros and Cons of Bottom-Up Search

- **Pros:**
  - Will not explore trees that don’t match input
  - Recursive rules less problematic
Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don’t match input
  - Recursive rules less problematic
  - Useful for incremental/fragment parsing
Pros and Cons of Bottom-Up Search

- **Pros:**
  - Will not explore trees that don’t match input
  - Recursive rules less problematic
  - Useful for incremental/fragment parsing

- **Cons:**
  - Explore subtrees that will not fit full sentences
Parsing Challenges

- Ambiguity
- Repeated substructure
- Recursion
Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V

- Structural ambiguity: Main types:
  - Attachment ambiguity
    - Constituent can attach in multiple places
      - *I shot an elephant in my pyjamas.*
Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V

- Structural ambiguity: Main types:
  - Attachment ambiguity
    - Constituent can attach in multiple places
      - *I shot an elephant in my pyjamas.*

- Coordination ambiguity
  - Different constituents can be conjoined
    - *Old men and women*
Ambiguity

Speech and Language Processing - Jurafsky and Martin
Disambiguation

• Global ambiguity:
  • Multiple complete alternative parses
  • Need strategy to select correct one
    • Approaches exploit other information
Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic
      - Pragmatic
        - E.g., elephants and pyjamas
  - Alternatively, keep all

- Local ambiguity:
  - Ambiguity in subtree, resolved globally
Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch

- No static backtracking strategy can avoid

- Efficient parsing techniques require storage of shared substructure
  - Typically with dynamic programming
Shared Sub-Problems

NP

Det Nominal

a Noun

flight...
Shared Sub-Problems

```
NP
  /\  \\
Det /   \
  /    \
\a/    Nominal
     /  \
    /  PP
   /     \
Noun /     from Indianapolis...
   /       \
flight/     \
```
Shared Sub-Problems

```
NP
  \--- Det
       a
  \--- Nominal
     \--- Nominal
        \--- Noun
            \--- flight
        \--- PP
            \--- to Houston...
     \--- PP
        \--- from Indianapolis
```
Shared Sub-Problems
Recursion

- Many grammars have recursive rules
  - E.g., $S \rightarrow S \text{ Conj } S$

- In search approaches, recursion is problematic
  - Can yield infinite searches
    - Esp., top-down
Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis

- The horse raced past the barn fell =>
Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis

- The horse raced past the barn fell =>
- The horse, raced past the barn, fell =>
Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis

- The horse raced past the barn fell =>
- The horse, raced past the barn, fell =>
- The horse which was raced past the barn fell.
Dynamic Programming

- Challenge: Repeated substructure -> Repeated work
- Insight:
  - Global parse composed of parse substructures
  - Can record parses of substructures
- Dynamic programming avoids repeated work by tabulating solutions to subproblems
  - Here, stores subtrees
Parsing w/Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
  - Polynomial time in input length
    - Typically cubic \( n^3 \) or less
- Several different implementations
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley algorithm
  - Chart parsing
Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - $A \rightarrow B \ C$, or
    - $A \rightarrow a$
  - However, most of our grammars are not of this form
    - E.g., $S \rightarrow \text{Wh-NP} \ \text{Aux} \ \text{NP} \ \text{VP}$
- Need a general conversion procedure
  - Any arbitrary grammar can be converted to CNF
CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP -> to VP
  - Unit productions:
    - A -> B
  - Long productions:
    - A -> B C D
CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP -> to VP
    - INF-VP -> TO VP; TO -> to

- Unit productions:
  - Rewrite RHS with RHS of all derivable non-unit productions
    - If $A \Rightarrow^* B$ and $B \rightarrow w$, then add $A \rightarrow w$
CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - $S \rightarrow \text{Aux NP VP}$
    - $S \rightarrow X_1 \text{ VP}; X_1 \rightarrow \text{Aux NP}$

- For all non-conforming rules,
  - Convert terminals to dummy non-terminals
  - Convert unit productions
  - Binarize all resulting rules
<table>
<thead>
<tr>
<th>$L_1$ Grammar</th>
<th>$L_1$ in CNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>$S \rightarrow NP \ VP$</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>$S \rightarrow X1 \ VP$</td>
</tr>
<tr>
<td>$X1 \rightarrow Aux \ NP$</td>
<td>$S \rightarrow book \mid include \mid prefer$</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$S \rightarrow Verb \ NP$</td>
</tr>
<tr>
<td>$S \rightarrow X2 \ PP$</td>
<td>$S \rightarrow Verb \ PP$</td>
</tr>
<tr>
<td>$S \rightarrow VP \ PP$</td>
<td>$S \rightarrow VP \ PP$</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$NP \rightarrow I \mid she \mid me$</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$NP \rightarrow TWA \mid Houston$</td>
</tr>
<tr>
<td>$NP \rightarrow Det \ Nominal$</td>
<td>$NP \rightarrow Det \ Nominal$</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Noun</td>
<td>Nominal $\rightarrow$ book $\mid$ flight $\mid$ meal $\mid$ money</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal Noun</td>
<td>Nominal $\rightarrow$ Nominal Noun</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal PP</td>
<td>Nominal $\rightarrow$ Nominal PP</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>$VP \rightarrow book \mid include \mid prefer$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP$</td>
<td>$VP \rightarrow Verb \ NP$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP \ PP$</td>
<td>$VP \rightarrow X2 \ PP$</td>
</tr>
<tr>
<td>$X2 \rightarrow Verb \ NP$</td>
<td>$VP \rightarrow VP \ PP$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ PP$</td>
<td>$VP \rightarrow VP \ PP$</td>
</tr>
<tr>
<td>$VP \rightarrow VP \ PP$</td>
<td>$PP \rightarrow Preposition \ NP$</td>
</tr>
</tbody>
</table>