1 ICL/Intro to Parsing CFGs/2005-11-03

Contents

2 What is Parsing?

Review CFGs

- Sets of terminals (either lexical items or parts of speech).
- Sets of non-terminals (the constituents of the language).
- Sets of rules (or 'productions') of the form $A \rightarrow \alpha$, where α is a string of zero or more terminals and non-terminals.

DERIVES:

- If grammar G contains the rule $A \rightarrow \gamma$ and $\alpha A \beta$ is a string in $(N \cup \Sigma)^*$, then $\alpha A \beta$ DIRECTLY DERIVES $\alpha\gamma\beta$ in G: $\alpha A \beta \Rightarrow \alpha\gamma\beta$.
- $\overset{*}{\Rightarrow}$ (DERIVES) is the reflexive, transitive closure of $\Rightarrow$; e.g., S $\overset{*}{\Rightarrow} \alpha$ if S $\overset{*}{\Rightarrow} \alpha_0$, $\alpha_0 \overset{*}{\Rightarrow} \alpha_1$, ..., $\alpha_n \overset{*}{\Rightarrow} \alpha$.

Parsing
Assign a correct tree to a string, given a grammar G, i.e.,

- The leaves of the tree cover all and only the input.
- The tree corresponds to a valid derivation according to G.
- 'correct':
  - means the tree is consistent with the input and the grammar;
  - **doesn't** mean that it's the proper way to represent English in any general sense.

Declarative vs./ Procedural Knowledge

- CFGs are declarative: they tell us **what** the well-formed structures and strings are.
- Parsers are procedural: they tell us **how** to compute the structure(s) for a given string.

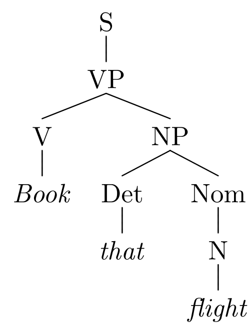
Parsing as Search
Syntactic parsing can be viewed as a search (cf. Jurafsky & Martin):

- search space: all possible trees generated by the grammar;
- search space defined by the grammar;
- search guided by the structure of the space and the input.

Mini Grammar & Lexicon

S → NP VP | Aux NP VP | VP	Det → *that* | *this* | *a*
NP → Det Nom | PropN	N → *book* | *flight* | *meal*
Nom → Nom PP | N Nom	V → *book* | *include* | *prefer*
PP → P NP	Aux → *does*
VP → V | V NP	P → *from* | *to* | *on*
Nom → N PP | N Nom	
PropN → *Houston* | *TWA*	

Example Parse Tree
The parse of the sentence *Book that flight* using the mini grammar and lexicon



3 Parsing Strategies

Parsing
What kind of constraints can be used to connect the grammar and our example sentence when searching for the parse tree?

- top-down (goal-directed) strategy:
  - e.g., tree should have one root (grammar constraint)
- bottom-up (data-driven) strategy:
  - e.g., tree should have 3 leaves (input sentence constraint)

A Note on the Input
We assume the following:

- The input is not tagged.
- The input consists of unanalyzed word tokens.
- e words in the input are 'known' (i.e., are leaves of lexical rules in grammar).
- All the words in the input are available simultaneously (i.e., they're buffered).

3.1 Top-Down

Top-Down Parsing

- When the search is primarily goal- or expectation-driven (by the structure of the grammar), we're doing a top-down search.
- Primary goal is to find a tree rooted at S that derives the input string.
- Trees are built from the root node S to the leaves.
- NLTK-Lite demo of Recursive Descent parser

```
>>> from nltk_lite.draw.rdparser import demo
>>> demo()
```

3.2 Bottom-Up

Bottom-Up Parsing

- When the search is primarily data-driven (by the input string), we're doing a BOTTOM-UP search.
- The primary consideration here is that all of the sub-trees of the final tree must hook up with the start symbol.
- NLTK-Lite demo of Shift-Reduce parser

```
>>> from nltk_lite.draw.srparser import demo
>>> demo()
```

Search Control Issues
Some parameters still need to be made explicit:

- non-parallel strategies (e.g., depth-first vs. breadth-first);
- which node in the tree to expand next (e.g., leftmost);
- which of the applicable grammar rule to try (e.g., order in the grammar)

Top-Down vs. Bottom-Up
There are advantages and disadvantages to both.
TOP-DOWN:

- only searches in the space of 'reasonable' answers;
- suggests hypotheses that are not consistent with the input string;
- has problems with left-recursion (covered later).

BOTTOM-UP

- only forms hypotheses consistent with the input strings;
- suggests hypotheses that make no sense 'globally'.

4 Left Corner Parsing

A Hybrid Approach

- Neither top-down nor bottom-up adequately exploit all the constraints.
- There are many way to combine top-down expectations with bottom-up data to get a more efficient search.
- The most popular methods use one method as the basic search control strategy to generate trees, and
- then use constraints from the other method to dynamically filter out "bad" structures.
- Example: top-down parsing with bottom-up filtering.

Left Corner Parsing

- Consider a top-down parser parsing the following input:

  Will this flight arrive on time?

  Assume that the grammar contains the following S rules:

  S→ NP VP
  S→ Aux NP VP
  S→ VP

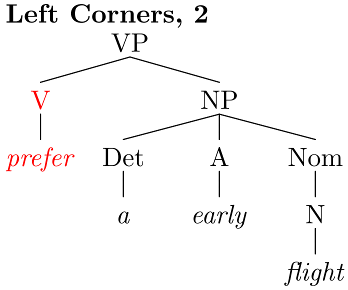
- Left-Corner Observation: in a successful parse, the current input word is first in the derivation of the unexpanded node.

Left Corners

- A category B (terminal or non-terminal) is a LEFT CORNER of a tree rooted in A if A derives $B\alpha$.
- Left corners for each non-terminal in our mini-grammar:

Category	*Left Corners*
S	Det, Proper-Noun, Aux, V
NP	Det, Proper-Noun
Nom	N
VP	V

Left Corners, 2



- V and *prefer* are both left-corners of the tree rooted in VP.
- Filtering with left corners:
  - *Only consider an expansion if current input can serve as the left-corner of that expansion.*

5 Problems

5.1 Left Recursion

Left Recursion
In top-down, depth-first, left-to-right parsers, a left recursive grammar can cause the search to never terminate.

- $A \rightarrow A\beta$

  Nom → Nom PP
  VP → VP PP
  S → S and S

- A derives $A\beta$ (i.e., the grammar contains a non-terminal that contains itself anywhere along its leftmost branch)

  NP→ NP_{poss} Nom
  NP_{poss} → NP *'s*

Left Recursion, cont.

- Demo example: Nom→ Nom PP

Some (poor) solutions:

- Rewrite the grammar to a weakly equivalent one (how?)
  - might not get a correct or useful parse tree.
- Limit the depth during search
  - limit is arbitrary.

5.2 Ambiguity

Ambiguity
Given a grammar, Global Ambiguity potentially leads to multiple parses for the same input (if we force it to).

I saw a woman with a telescope.

Local Ambiguity, in contrast, leads to hypotheses that are locally reasonable but eventually lead nowhere and result in inefficient backtracking. Filtering helps a little.

Book that flight.

Common Structural Ambiguities

- See this week's Lab Exercises.

Why is Ambiguity Problematic?

- There are potentially an exponential number of parses for a sentence.
  - Returning all structurally valid parses isn't always a good idea.
- Some solutions:
  - exploit regularities in the search space to derive common subparts only once;
  - heuristic search strategies;
  - incorporate semantics into the disambiguation process.

6 Summary

Summary

- Important parsing concepts:
  - Top-down vs. Bottom-up strategies
  - Examples of each:
    - Recursive Descent
    - Shift-Reduce
  - Backtracking
  - Global vs. Local Ambiguity

Reading

- Jurafsky and Martin Chapter 10
- NLTK Parsing Tutorial