Computational Psycholinguistics

Lecture 2: Parsing

Matthew W Crocker

Computerlinguistik
Universität des Saarlandes

Lecture Overview

- Incremental Parsing
  - Top-down
  - Bottom-up
  - Mixed strategy
- Ambiguity and parsing
  - Serial with Backtracking (non-deterministic)
  - Serial deterministic
  - Parallel

A Simple Theory of Grammar

The Grammar

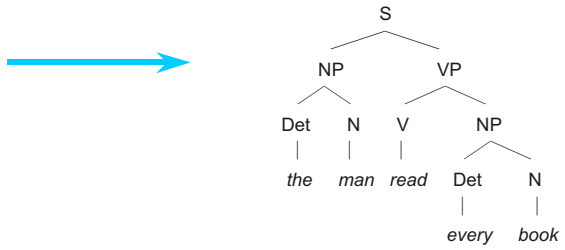
- S → NP VP
- NP → PN
- NP → Det N
- NP → NP PP
- PP → P NP
- VP → V
- VP → V NP
- VP → V NP PP

The Lexicon

- Det = {*the, a, every*}
- N = {*man, woman, book, hill, telescope*}
- PN = {*John, Mary*}
- P = {*on, with*}
- V = {*saw, put, open, read, reads*}

Parsing Algorithms

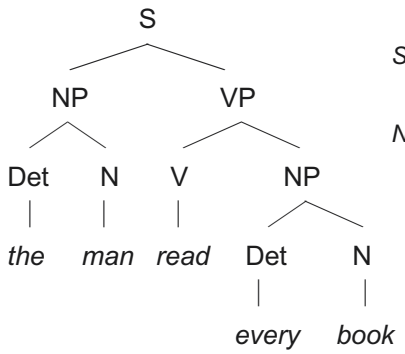
- How do we build a syntactic analysis for an input utterance?
  - *"The man read every book"*



  - What do we know about how people parse and interpret utterances?

A Generated Sentence

- the man read every book



S → *NP VP*

NP → *Det N* *VP* → *V NP*

NP → *Det N*

Phrase Structure Grammars

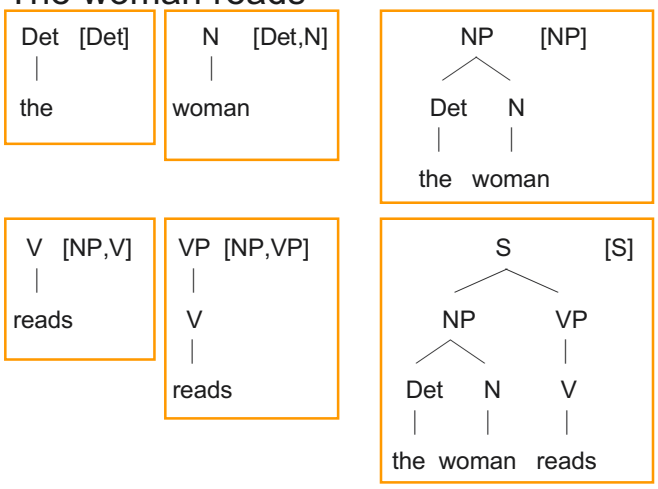
- What language is generated by a PSG:
  - Node admissibility criterion
- Issues:
  - Finite grammar and lexicon can generates an infinite language (and infinite number of strings)
  - Recursion: e.g. NP → NP PP
- Equivalent to a push-down automata (require 'memory', unlike a FSA)
- Not quite powerful enough for NLs:
  - Minimally, we probably require "indexed" context free languages
- But, it is also possible to approximate a complex grammar with a simpler formalism (FSA, regular languages)
  - I.e. if we limit the depth of recursion, exploit "complex" states

Parsing Algorithms for PSGs

- An algorithm to recover the parse tree for an utterance, given that it is in the language
- Dimensions of variation:
  - left-to-right, head-driven, right to left
  - top-down, bottom-up, mixed
  - deterministic, serial, parallel
- Complexity:
  - Time: what time is required (worst or average case) to parse a sentence as a function of sentence length, grammar size?
  - Space: how much memory does the parser require?

Bottom-up Parsing

- "The woman reads"

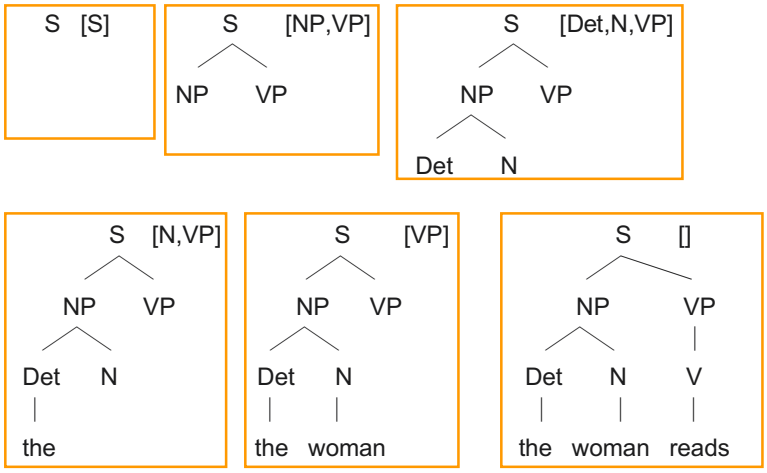


Shift-reduce Algorithm

1. Initialise *Stack* = []
2. loop: Either *shift*:
   - Determine category, *C*, for next word in sentence;
   - Push *C* onto the stack;
3. Or *reduce*:
   - If categories on the *Stack* match the RHS of a rule:
     - Remove those categories from the *Stack*;
     - Push the LHS category onto the *Stack*;
4. No more words to process?
   - If *Stack* = [S], then done;
5. Goto ②

Top-down Parsing

- "The woman reads"



Top-down Algorithm

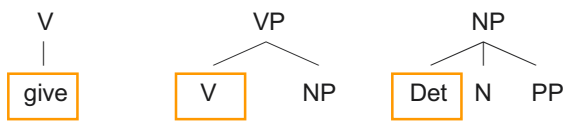
1. Initialise *Stack* = [S]
2. If top(*Stack*) is a non-terminal, *N:*
   - Select rule $N \longrightarrow RHS$;
   - pop(*N)* off the stack and push(*RHS*) on the stack;
3. If top(*Stack*) is a pre-terminal, *P:*
   - Get next word, W, from the input;
   - If $P \longrightarrow W$, then pop(*P*) from the stack;
   - Else fail;
4. No more words to process?
   - If *Stack* = [], then done;
5. Goto ②

Evaluating top-down & bottom-up

- Are these parsers psychologically plausible?
- Incrementality:
  - Bottom-up: no
  - Top-down: yes
- Input-driven:
  - Bottom-up: yes
  - Top-down: no
  - ➡ Problems with left-recursion

A Psychologically Plausible Parser

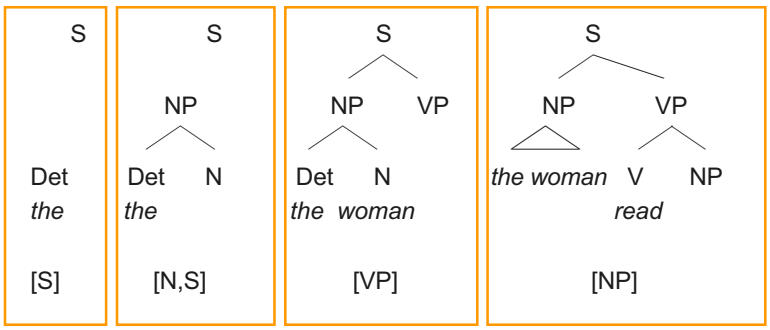
- Left-Corner Parsing
- Rules are ‘activated’ by their ‘left-corner’



- Combines input-driven with top-down
- There is a ‘class’ of LC parsers

An example LC parse

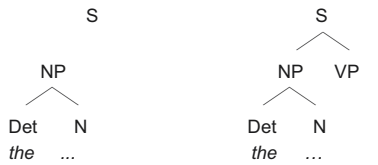
- “The woman read the book”



- Is this incremental?

Evaluating the LC Parser

- Almost incremental
- Variations:
  - Using a 'top-down' oracle of LC relation
  - Arc-standard versus arc-eager



- Left-recursion: NP ⟶ NP PP

Incrementality and Memory

- It wasn't incrementality that led to the LC algorithm, but memory load
  - *"The mouse died"*
  - *"The mouse the cat chased died"*
  - *"The mouse the cat the dog bit chased died"*

  (Cf: "The mouse that the cat that the dog bit chased died")
- Grammatical, not ambiguous, what's the problem?
- Memory load: too high for centre embedding
  - *"[The mouse [the cat [the dog bit] chased] died]"*

Memory Load in Parsing

- Left-embedding (LE) is easy:
  - [[[John's brother]'s car door]'s handle] broke off.
- So is right-embedding (RE):
  - John believes [Bill knows [Mary said [she likes cats]]]
- But centre-embedding (CE) is hard:
  - [The mouse [the cat [the dog bit] chased] died]

- Top-down: LE: hard CE: hard RE: easy
- Bottom-up: LE: easy CE: hard RE: hard
- Left-corner: LE: easy CE: hard RE: easy

Ambiguity in Parsing

- Parsing involves rule selection: *what if more than one rule can be selected?*
- Local ambiguity: a parse derivation may fail later
- Global ambiguity: multiple parses can succeed

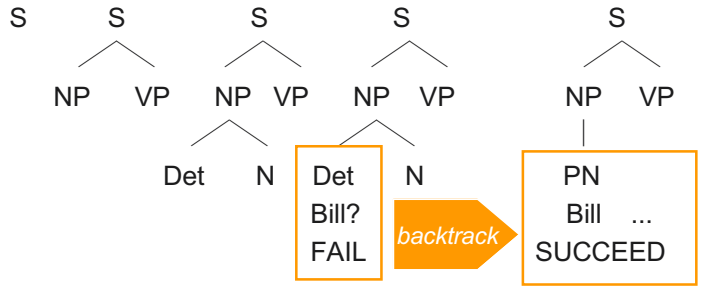
- How can we handle local and global ambiguities during parsing:
  - Backtracking
  - Parallelism
  - Determinism
  - Underspecification

Backtracking Parsers

- Parsing is a sequence of rule selections
- If at one point, more than one rule can be applied, this is called a choice point
- Make a decision, based on some selection rule
- If subsequently parsing 'blocks', return to a choice point and re-parse from there
- Which choice point to return to?
  - usually the last, why?
  - what other choice point selection rules could be used

Backtracking: an example

- "Bill reads"



Deterministic parsing

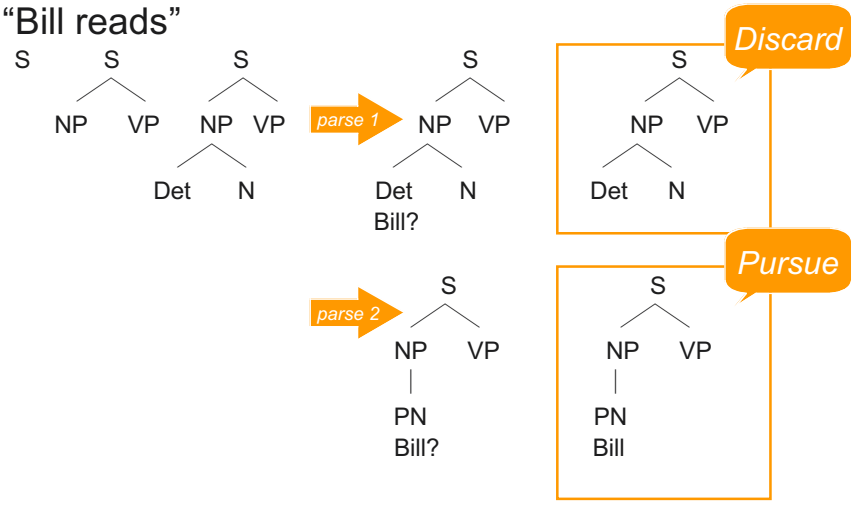
- A deterministic parser consists of unambiguous parsing actions.
  - At every "state" during parsing (current parse + current input) the parser can take precisely one action
- Marcus (1980) "Parsifal": A theory of human syntactic recognition
  - A deterministic LR parser for English
  - Easy sentences are those which can be parsed deterministically
  - Sentences which cannot be parsed, are predicted to be garden paths
- Criticism:
  - Not incremental:
    - Constituents are often buffered, requires 3 look ahead constituents
    - Relies on information from selecting heads (I.e. verbs):
      - Fine for English, but poor for head-final languages
  - Poor explanation of "graded" human behaviour and the influence of various information sources (semantics, frequency, etc.)

Parallel Parsers

- Build parse trees through successive rule selections
- If more than one rule may be applied, create a new parse derivation for each possibility
- Pursue all parses in parallel
- If any of the parses 'blocks', discard it
- Note: because of multiple local ambiguities, the number of parallel derivation grows exponentially
- Bounded parallelism: pursue a fixed number:
  - How do we choose which ones to keep?

Parallel: an example

- "Bill reads"



Issues

- What is an appropriate mechanism for constructing interpretations:
  - Incremental parsing & memory characteristics
  - Left-corner seems like a good first approximation
  - Problem: true incrementality leads to recursion problems
- Rule selection
  - On what basis should we decide between alternatives?
    - In selecting a single structure or ranking parallel alternatives?
  - What information sources: Syntax, semantics, recency, memory
- Backtracking/reanalysis
  - What triggers reanalysis?
  - Mechanical versus intelligent reanalysis?
  - Standard backtracking involves destruction of previously parsed material, can we implement more intelligent reanalysis mechanisms?
  - In parallel models, how are parsers and when are alternative "re-ranked"?