Computational Psycholinguistics

Lecture 3: Parsing and Memory

Matthew W Crocker

Computerlinguistik
Universität des Saarlandes

Overview

- Computational psycholinguistics
  - Models of how people use linguistic knowledge to process language
    - Grammaticality: possibly including degrees of acceptability
    - Complexity: when sentences are too complex
    - Behaviour: preferences in resolving ambiguity and reanalysis
- Computational models provide the framework for examining the characteristics and predictions of particular theories.
- Implementing Parsers in Prolog
  - Top-down
  - Shift-reduce (bottom-up)
- Implementing Left-corner parsers in Prolog:
  - Arc-standard
  - Arc Eager
- Profiling memory requirements for embeddings

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

Grammars as Programs

- So, for a grammar like:

```
s   --> np, vp.
np  --> det, n.
vp  --> v, np, np.
vp  --> v.
det --> [ the] .          det --> [ a] .
n   --> [ man] .          n --> [ woman] .
v   --> [ gave] .  v --> [ swims] .
```

- In Prolog:

```
s(A,B) :- np(A,C), vp(C,B).
np(A,B) :- det(A,C), n(C,B).
vp(A,B) :- v(A,C), np(C,D),np(D,B).
det(A,B) :- 'C'(A,the,B).
'C'([ Head|Tail] ,Head,Tail).
```

Grammars as Data

- Writing an parser
  - write a parser which takes a grammar and string as it's input
  - Since '-->' converts a rule into a Prolog clause, use a new operator: '--->'
- So, for a grammar like:

```
:- op(1100, xfx, '--->').
s   ---> np, vp.
np  ---> det, n.
vp  ---> v, np.
vp  ---> v.
det ---> [the] .  det ---> [a] .
n   ---> [man] .  n ---> [woman] .
v   ---> [gave] . v ---> [swims] .
```


A Top-Down Parser in Prolog

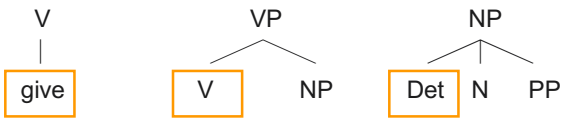
```
% Expand the current NT using a rule
td_parse(NT, P0, P) :-
      (NT ---> Body),
      td_parse(Body, P0, P).
% Parse each of the symbols on the RHS
td_parse((Body1, Body2), P0, P) :-
      td_parse(Body1, P0, P1),
      td_parse(Body2, P1, P).
% Consume a word in the input
td_parse([Word], P0, P) :-
      connects(P0, Word, P).

% Difference list handling
connects([Word|List],Word,List).
```

The Shift-Reduce Parser in Prolog

```
% Done
parse([s],[]).
% Reduce
parse([Y,X|Rest],String) :-
    (LHS ---> X,Y),
    parse([LHS|Rest],String).
% Shift
parse(Stack,[Word|Rest]) :-
    (Cat ---> [Word]),
    parse([Cat|Stack],Rest).
```


A Psychologically Plausible Parser

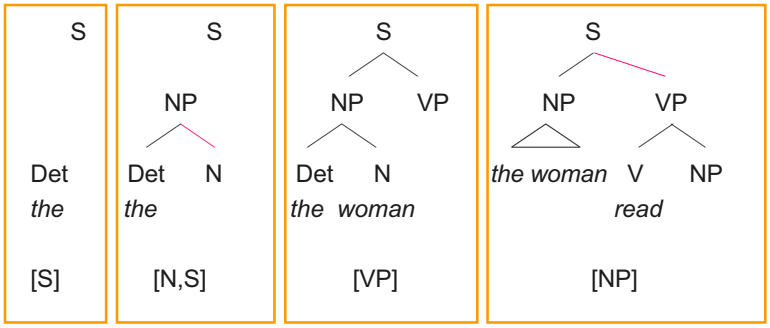
- Left-Corner Parsing
- Rules are 'activated' by their 'left-corner'



- 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?

A Grammar

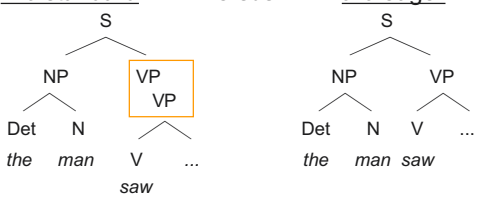
```
:- op(1100,xfx, '--->').

s ---> np, vp.
np ---> det, n.
np ---> np, rc.
rc ---> rpro, sgap.
sgap ---> np, vt.
vp ---> vt, np.
vp ---> vs, s.
...
det ---> [the].
N ---> [cat].
vt ---> [chased].
vs ---> [knows].
...
```

Left-Corner Parsing in Prolog

```
parse(Phrase,S1,S0) :-
    connects(S1,Word,S2),
    (Cat ---> [Word]),
    lc(Cat,Phrase,S2,S1).

% Reflexive closure
lc(Phrase,Phrase,S0,S0).

% Transitive closure
lc(SubPhrase,SuperP,S1,S0) :-
   (Phrase ---> SubPhrase, Right),
    parse(Right,S1,S2),
    lc(Phrase,SuperP,S2,S0).

% Difference list handling
connects([Word|List],Word,List).
```


Evaluating the LC Parser

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



- Affect on ambiguity resolution for arc-eager:
  - Commitment to attachments is early (before constituents are complete)
  - Top-down use of syntactic context
- Possible left-recursion problems

Using an Oracle

- The left-corner relation holds for only a finite pair of categories.

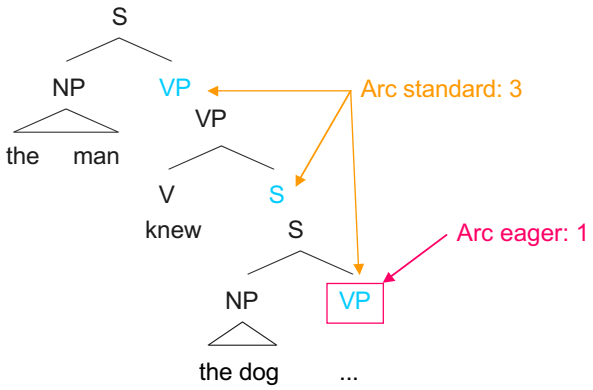
```
% Grammar
s ---> np, vp.
np ---> det, n.
det ---> np, poss.
vp ---> v, np.

% Oracle: reflexive, transitive closure.
link(s,s).    link(np,s).   link(np,np).
link(n,n).    link(det,s). link(v,vp).
link(det,np). link(vp,vp).
link(det,det).
```


Left-Corner Parsing with Oracle

```
parse(Phrase,S1,S0) :-
    connects(S1,Word,S2),
    (Cat ---> [Word]),
    link(Cat,Phrase),
    lc(Cat,Phrase,S2,S1).

% Reflexive closure
lc(Phrase,Phrase,S0,S0).

% Transitive closure
lc(SubPhrase,SuperP,S1,S0) :-
    (Phrase ---> SubPhrase, Right),
    link(SubPhrase,SuperP),
    parse(Right,S1,S2),
    lc(Phrase,SuperP,S2,S0).
```

Evaluating the LC Parser

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



Eager Left-Corner Parsing

```
% Reflexive closure
lc(Phrase,Phrase,S0,S0).

% Eager composition
lc(Phrase,SuperP,S1,S0) :-
    (SuperP ---> Phrase, Right),
    parse(Right,S1,S0).

% Transitive closure
lc(SubPhrase,SuperP,S1,S0) :-
    (Phrase ---> SubPhrase, Right),
    parse(Right,S1,S2),
    lc(Phrase,SuperP,S2,S0).
```

Summary of Behaviour

Node	Arcs	Left	Centre	Right
Top-down	Either	O(n)	O(n)	O(1)
Shift-reduce	Either	O(1)	O(n)	O(n)
Left-corner	Standard	O(1)	O(n)	O(n)
Left-corner	Eager	O(1)	O(n)	O(1)
People		O(1)	O(n)	O(1)

Comments on Left-Corner

- Mixed data-driven and hypothesis driven approaches
- An oracle can increase the top-down component, reduce ambiguity

- Composition:
  - Eager parsing corresponds to *composition of partial structures*
  - Combinatory categorial grammar (CCG) directly incorporates such composition into the grammar formalism.

- Trade-off:
  - Arc Standard: less ambiguity
    - attachments are made when constituents are complete: safer
    - delayed attachment means more must be kept on the stack
  - Arc Eager: less memory
    - early composition reduces memory for the push-down automata
    - eager attachments are made with less bottom-up evidence

Parsing and ambiguity resolution

- We can motivate the LC-eager parser in term of memory requirements
- What predictions does it make for ambiguity resolution?

- Consider the following high-low attachment ambiguity:
  - "Two sisters reunited after eighteen years in a checkout counter"
  - "John said that he will go to Edinburgh last week"
- These sentence are perceived as odd, because people prefer to attach the modifier "low" (to the immediately preceding phrase), but it must be attached "high" (to the main verb)

- Extending the computational model:
  - What attachment is preferred by the parsers discussed here?
  - Does the choice of grammar make a difference?
  - Is there some what to implement this "low attachment" preference in the parsers discussed?