## Parsing Tricks

### Left-Corner Parsing
- Technique for 1 word of lookahead in algorithms like Earley's
- (can also do multi-word lookahead but it's harder)

### Basic Earley's Algorithm

<table>
<thead>
<tr>
<th>0</th>
<th>Papa</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ROOT . S</td>
<td>ONP Papa .</td>
</tr>
<tr>
<td>0</td>
<td>NP . Det N</td>
<td>ONP NP . PP</td>
</tr>
<tr>
<td>0</td>
<td>NP . Papa</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Det . the</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Det . a</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>VP . V NP</td>
<td>1 VP . VNP</td>
</tr>
<tr>
<td>0</td>
<td>NP . Papa</td>
<td>1 VP . VP PP</td>
</tr>
<tr>
<td>0</td>
<td>Det . the</td>
<td>1 PP . PNP</td>
</tr>
<tr>
<td>0</td>
<td>Det . a</td>
<td>1 PP . PP</td>
</tr>
</tbody>
</table>

- attach

<table>
<thead>
<tr>
<th>0</th>
<th>Papa</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ROOT . S</td>
<td>ONP Papa .</td>
</tr>
<tr>
<td>0</td>
<td>NP . Det N</td>
<td>ONP NP . PP</td>
</tr>
<tr>
<td>0</td>
<td>NP . Papa</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Det . the</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Det . a</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>VP . V NP</td>
<td>1 VP . VNP</td>
</tr>
<tr>
<td>0</td>
<td>NP . Papa</td>
<td>1 VP . VP PP</td>
</tr>
<tr>
<td>0</td>
<td>Det . the</td>
<td>1 PP . PNP</td>
</tr>
<tr>
<td>0</td>
<td>Det . a</td>
<td>1 PP . PP</td>
</tr>
</tbody>
</table>

- predict

- $\cdot . V$ makes us add all the verbs in the vocabulary!
- Slow – we’d like a shortcut.
- Every VP adds all VP → ... rules again.
- Before adding a rule, check it’s not a duplicate.
- Slow if there are > 700 VP → ... rules, so what will you do in Homework 4?

1-word lookahead would help

- No point in adding words other than ate

With Left-Corner Filter

- PP can’t start with ate
- Birth control – now we won’t predict 1 PP . P NP 1 P . with either!
- Need to know that ate can’t start PP
- Take closure of all categories that it does start ...

1-word lookahead would help

- No point in adding words other than ate

In fact, no point in adding any constituent that can’t start with ate
- Don’t bother adding PP, P, etc.
Merging Right-Hand Sides

- Grammar might have rules
  $$X \rightarrow AGHP$$
  $$X \rightarrow BGHP$$
- Could end up with both of these in chart:
  $$(2, X \rightarrow A \cdot GHP)$$ in column 5
  $$(2, X \rightarrow B \cdot GHP)$$ in column 5
- But these are now interchangeable: if one produces X then so will the other
- To avoid this redundancy, can always use dotted rules of this form: $$X \rightarrow \ldots \cdot GHP$$

- Combining the two previous cases:
  $$X \rightarrow AGHP$$
  $$X \rightarrow AGHQ$$
  $$X \rightarrow BGHP$$
  $$X \rightarrow BGHQ$$
becomes
  $$X \rightarrow (A \mid B) \cdot GHP$$
- And often nice to write stuff like
  $$NP \rightarrow (Det \mid \epsilon) \cdot Adj* \cdot N$$

Merging Right-Hand Sides

- Similarly, grammar might have rules
  $$X \rightarrow AGHP$$
  $$X \rightarrow AGHQ$$
- Could end up with both of these in chart:
  $$(2, X \rightarrow A \cdot GHP)$$ in column 5
  $$(2, X \rightarrow A \cdot GHQ)$$ in column 5
- Not interchangeable, but we’ll be processing them in parallel for a while ...
- Solution: write grammar as $$X \rightarrow AGH (P \mid Q)$$

- Combining the two previous cases:
  $$X \rightarrow (A \mid B) \cdot G(H(P \mid Q))$$
  $$NP \rightarrow (Det \mid \epsilon) \cdot Adj* \cdot N$$
- These are regular expressions!
- Build their minimal DFAs:
- Automaton states replace dotted rules ($$X \rightarrow AGHP$$)
**Merging Right-Hand Sides**

Indeed, all \( NP \rightarrow \) rules can be unioned into a single DFA!

\[
NP \rightarrow ADJP ADJP JJ JJ NN NNS
\]
\[
NP \rightarrow ADJP DT NN
\]
\[
NP \rightarrow ADJP JJ NN
\]
\[
NP \rightarrow ADJP JJ NN NNS
\]
\[
NP \rightarrow ADJP JJ NNS
\]
\[
NP \rightarrow ADJP NN
\]
\[
NP \rightarrow ADJP NN NN
\]
\[
NP \rightarrow ADJP NN NNS
\]
\[
NP \rightarrow ADJP NNS
\]
\[
NP \rightarrow ADJP NPR
\]
\[
NP \rightarrow ADJP NPRS
\]
\[
NP \rightarrow DT
\]
\[
NP \rightarrow DT ADJP
\]
\[
NP \rightarrow DT ADJP, JJ NN
\]
\[
NP \rightarrow DT ADJP ADJP NN
\]
\[
NP \rightarrow DT ADJP JJ JJ NN
\]
\[
NP \rightarrow DT ADJP JJ NN
\]
\[
NP \rightarrow DT ADJP JJ NN NN
\]

etc.

**Earley’s Algorithm on DFAs**

- What does Earley’s algorithm now look like?

```
VP → ... PP...
NP → ...
```

Column 4

```
... (2, ●) predict
```

- 

\[
ADJP
\]
\[
ADJ
\]
\[
P
\]
\[
Det
\]
\[
Adj
\]
\[
N
\]

\[
PP
\]

\[
NP
\]

```
PP → ●< ...
```

- 

Column 4 | Column 5 | ... | Column 7
---|---|---|---
(2, ●) | ... | (4, ○) predict or attach? (4, ○) predict or attach?

(4, ○) | (4, ○) predict or attach? (4, ○) predict or attach? (2, ●) predict or attach? Both!
Pruning and Prioritization

- Heuristically throw away constituents that probably won't make it into best complete parse.
- Use probabilities to decide which ones.
- So probs are useful for speed as well as accuracy!
- Both safe and unsafe methods exist
  - Iterative deepening: Throw x away if \( p(x) < 10^{-200} \)
    (and lower this threshold if we don't get a parse)
  - Heuristic pruning: Throw x away if \( p(x) < 0.01 \cdot p(y) \)
    for some y that spans the same set of words (for example)
- Prioritization: If \( p(x) \) is low, don't throw x away; just postpone using it until you need it (hopefully you won't).

Prioritization continued:
Agenda-Based Parsing

1. until we pop a parse from agenda or fail with empty agenda
   - pop top element \( Y \) from agenda into chart
   - for each right neighbor \( Z \) of \( Y \) in grammar
     - put \( Y \) onto the agenda
   - for each rule \( X \) in grammar
     - put \( X \) onto the agenda

Prioritization continued:
Agenda-Based Parsing

1. until we pop a parse from agenda or fail with empty agenda
   - pop top element \( Y \) from agenda into chart
   - for each right neighbor \( Z \) of \( Y \) in grammar
     - put \( Y \) onto the agenda
   - for each rule \( X \) in grammar
     - put \( X \) onto the agenda

Prioritization continued:
Agenda-Based Parsing

1. until we pop a parse from agenda or fail with empty agenda
   - pop top element \( Y \) from agenda into chart
   - for each right neighbor \( Z \) of \( Y \) in grammar
     - put \( Y \) onto the agenda
   - for each rule \( X \) in grammar
     - put \( X \) onto the agenda

Prioritization continued:
Agenda-Based Parsing

1. until we pop a parse from agenda or fail with empty agenda
   - pop top element \( Y \) from agenda into chart
   - for each right neighbor \( Z \) of \( Y \) in grammar
     - put \( Y \) onto the agenda
   - for each rule \( X \) in grammar
     - put \( X \) onto the agenda

Prioritization continued:
Agenda-Based Parsing

1. until we pop a parse from agenda or fail with empty agenda
   - pop top element \( Y \) from agenda into chart
   - for each right neighbor \( Z \) of \( Y \) in grammar
     - put \( Y \) onto the agenda
   - for each rule \( X \) in grammar
     - put \( X \) onto the agenda

Outside Estimates for better Pruning and Prioritization

- Iterative deepening: Throw x away if \( p(x) \cdot q(x) < 10^{-200} \)
  (lower this threshold if we don't get a parse)
- Heuristic pruning: Throw x away if \( p(x) \cdot q(x) < 0.01 \cdot p(y) \cdot q(y) \)
  for some y that spans the same set of words
- Prioritized agenda: Priority of x on agenda is \( p(x) \cdot q(x) \); stop at first parse

In general, the “inside prob” \( p(x) \) will be higher for smaller constituents
- Not many rule probabilities inside them
- Many rules have the same probabilities

The “outside prob” \( q(x) \) is intended to correct for this
- Estimates the prob of all the rest of the rules needed to build x into full parse
- So \( p(x) \cdot q(x) \) estimates prob of the best parse that contains x

But if we take \( q(x)=1 \), that’s just the methods from previous slides
- And iterative deepening and prioritization were safe there
- If we take \( q(x) \) to be an “optimistic estimate” (always \( \leq p(x) \))
  - Not many rule probabilities inside them
  - Not safe! Prioritized agenda is then an example of an “A*” algorithm
**Outside Estimates for better Pruning and Prioritization**

- **Iterative deepening:** Throw away if \(p(x)q(x) < 10^{-200}\) (lower this threshold if we don't get a parse)
- **Heuristic pruning:** Throw away if \(p(x)q(x) < 0.01p(y)q(y)\) for some \(y\) that spans the same set of words
- **Prioritized agenda:** Priority of \(x\) on agenda is \(p(x)q(x)\); stop at first parse

In general, the "inside prob" \(p(x)\) will be higher for smaller constituents
- Not many rule probabilities inside them

The "outside prob" \(q(x)\) is intended to correct for this
- Estimates the prob of all the rest of the rules needed to build \(x\) into full parse
- So \(p(x)q(x)\) estimates prob of the best parse that contains \(x\)

If we take \(q(x)\) to be the best estimate we can get
- Methods may no longer be safe (but may be fast!)
- Prioritized agenda is then called a "best-first algorithm"

But if we take \(q(x) = 1\), that's just the methods from previous slides
- And iterative deepening and prioritization were safe there

If we take \(q(x)\) to be an "optimistic estimate" (always \(\geq\) true prob)
- Still safe! Prioritized agenda is then an example of an "A* algorithm"

Terminology warning: Here "inside" and "outside" mean probability of the best partial parse inside or outside \(x\)
- But traditionally, they mean total prob of all such partial parses (as in the "inside algorithm" that we saw in the previous lecture)

**Preprocessing**

- First "tag" the input with parts of speech:
  - Guess the correct preterminal for each word, using faster methods we'll learn later
  - Now only allow one part of speech per word
  - This eliminates a lot of crazy constituents!
  - But if you tagged wrong you could be hosed

- Raise the stakes:
  - What if tag says not just "verb" but "transitive verb"?
  - Or "verb with a direct object and 2 PPs attached"? ("supertagging")

- Safer to allow a few possible tags per word, not just one ...

**Center-Embedding**

```
if x
  if y
    STATEMENT -> if EXPR then STATEMENT endif
  endif
else b
  STATEMENT -> if EXPR then STATEMENT else STATEMENT endif
endif
```

- This is the rat that ate the malt.
- This is the malt that the rat ate.
- This is the cat that bit the rat that ate the malt.
- This is the malt that the rat that the cat bit ate.
- This is the dog that chased the cat that bit the rat that ate the malt.
- This is the malt that [the rat that [the cat that the dog chased] bit] ate.

**More Center-Embedding**

- [What did you disguise [those handshakes that you greeted [the people we bought [the bench [Billy was read to]]] with]] for]?
- [Which mantelpiece did you put [the idol I sacrificed [the fellow we sold [the bridge you threw [the bench [Billy was read to]]]]] on]?

**Center Recursion vs. Tail Recursion**

- "pied piping" – NP moves leftward, preposition follows along
## Disallow Center-Embedding?

- Center-embedding seems to be in the grammar, but people have trouble processing more than 1 level of it.
- You can limit # levels of center-embedding via features: e.g., $S[S\_\text{DEPTH}=n+1] \rightarrow A S[S\_\text{DEPTH}=n]$ $B$
- If a CFG limits # levels of embedding, then it can be compiled into a finite-state machine — we don’t need a stack at all!
  - Finite-state recognizers run in linear time.
  - However, it’s tricky to turn them into parsers for the original CFG from which the recognizer was compiled.
  - And compiling a small grammar into a much larger FSA may be a net loss — structure sharing in the parse chart is expanded out to duplicate structure all over the FSA.

## Parsing Algs for non-CFG

- If you’re going to make up a new kind of grammar, you should also describe how to parse it.
- Such algorithms exist, e.g.,
  - for TAG (where the grammar specifies not just rules but larger tree fragments, which can be combined by “substitution” and “adjunction” operations)
  - for CCG (where the grammar only specifies preterminal rules, and there are generic operations to combine slashed nonterminals like $X/Y$ or $(X/Z)/(Y/W)$)