L95: Natural Language Syntax and Parsing
6) N-best Parsing

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge
Reminder...

We have looked at the following algorithms:

- CKY
- Shift-Reduce
- A*

But so far we have discussed finding the best parse... what if we want to find the n-best parses?
Recall that full CKY is **optimal** and **exhaustive**

For the best parse we keep the most probable partial derivation for every non-terminal at each cell

![Diagram]

\[
\begin{array}{ccc}
1 & 2 & 3 \\
\end{array}
\]

\[
\begin{align*}
N & = \{ S, NP, VP, VV, VM \} \\
\Sigma & = \{ can, fish, they \} \\
S & = S \\
\mathcal{P} & = \{ S \rightarrow NP \ VP \ 1.0 \\
& \quad VP \rightarrow VM \ VV \ 0.9 \\
& \quad VP \rightarrow VV \ NP \ 0.1 \\
& \quad VV \rightarrow can \ 0.2 \mid fish \ 0.8 \\
& \quad VM \rightarrow can \ 1.0 \\
& \quad NP \rightarrow they \ 0.5 \mid fish \ 0.5 \} \\
\end{align*}
\]

they can fish
Recall that full CKY is \textbf{optimal} and \textbf{exhaustive}

For the best parse we keep the most probable partial derivation for every non-terminal at each cell.

\[
\begin{array}{ccc}
1 & 2 & 3 \\
0 \quad NP^{0.5}_{(they)} & & \\
1 & & \\
2 & & \\
\end{array}
\]

\[
\begin{align*}
\mathcal{N} &= \{S, NP, VP, VV, VM\} \\
\Sigma &= \{can, fish, they\} \\
S &= S \\
\mathcal{P} &= \{S \rightarrow NP VP 1.0 \\
& \quad VP \rightarrow VM VV 0.9 \\
& \quad VP \rightarrow VV NP 0.1 \\
& \quad VV \rightarrow can 0.2 \mid fish 0.8 \\
& \quad VM \rightarrow can 1.0 \\
& \quad NP \rightarrow they 0.5 \mid fish 0.5 \}\end{align*}
\]

\text{they} \quad \text{can} \quad \text{fish}
Recall that full CKY is **optimal** and **exhaustive**

For the best parse we keep the most probable partial derivation for every non-terminal at each cell

```
1   2   3

\[ NP^{0.5}_{(they)} \]
\[ VV^{0.2}_{(can)} \]
\[ VM^{1.0}_{(can)} \]
```

\[ \mathcal{N} = \{ S, NP, VP, VV, VM \} \]
\[ \Sigma = \{ can, fish, they \} \]
\[ S = S \]
\[ \mathcal{P} = \{ S \rightarrow NP \ VP \ 1.0 \\
     VP \rightarrow VM \ VV \ 0.9 \\
     VP \rightarrow VV \ NP \ 0.1 \\
     VV \rightarrow can \ 0.2 \ | \ fish \ 0.8 \\
     VM \rightarrow can \ 1.0 \\
     NP \rightarrow they \ 0.5 \ | \ fish \ 0.5 \} \]
Recall that full CKY is optimal and exhaustive.

For the best parse we keep the most probable partial derivation for every non-terminal at each cell.

```
0  NP_{(they)}^{0.5}
1  VV_{(can)}^{0.2}  VM_{(can)}^{1.0}
2

Ν = { S, NP, VP, VV, VM }
Σ = { can, fish, they }
S = S
P = { S → NP VP 1.0
     VP → VM VV 0.9
     VP → VV NP 0.1
     VV → can 0.2 | fish 0.8
     VM → can 1.0
     NP → they 0.5 | fish 0.5 }
```

Paula Buttery (Computer Lab)  L95: Natural Language Syntax and Parsing
Recall that full CKY is **optimal** and **exhaustive**

- For the best parse we keep the most probable partial derivation for every non-terminal at each cell

\[
\begin{array}{ccc}
1 & 2 & 3 \\
NP_{(they)}^{0.5} & & \\
VV_{(can)}^{0.2} & VM_{(can)}^{1.0} & \\
NP_{(fish)}^{0.5} & VV_{(fish)}^{0.8} & \\
\end{array}
\]

- \( \mathcal{N} = \{ S, NP, VP, VV, VM \} \)
- \( \Sigma = \{ can, fish, they \} \)
- \( S = S \)
- \( \mathcal{P} = \{ S \rightarrow NP \ VP \ 1.0 \\
    VP \rightarrow VM \ VV \ 0.9 \\
    VP \rightarrow VV \ NP \ 0.1 \\
    VV \rightarrow can \ 0.2 \mid fish \ 0.8 \\
    VM \rightarrow can \ 1.0 \\
    NP \rightarrow they \ 0.5 \mid fish \ 0.5 \} \)

Paula Buttery (Computer Lab)
Recall that full CKY is **optimal** and **exhaustive**.

For the best parse we keep the most probable partial derivation for every non-terminal at each cell.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 0 | **NP**
  |   |   |
| 1 | **VV**
  |   |   |
| 2 | **NP**
  |   |   |

\[
\begin{align*}
V_{\text{can}} & : 0.2 \\
V_{\text{fish}} & : 0.8 \\
N_{\text{they}} & : 0.5 \\
N_{\text{can}} & : 1.0 \\
V_{\text{VP}} & : 0.2 \times 0.5 \times 0.1 = 0.01 \\
V_{\text{NP}} & : 1.0 \times 0.8 \times 0.9 = 0.72 \\
N_{\text{NP}} & : 0.5 \times 0.5 \\
S & : 0.5 \\
\end{align*}
\]

\[
\begin{align*}
\mathcal{N} & = \{ S, NP, VP, VV, VM \} \\
\Sigma & = \{ \text{can, fish, they} \} \\
\mathcal{P} & = \{ S \rightarrow NP \ VP \ 1.0 \\
 & VP \rightarrow VM \ VV \ 0.9 \\
 & VP \rightarrow VV \ NP \ 0.1 \\
 & VV \rightarrow can \ 0.2 \ | \ fish \ 0.8 \\
 & VM \rightarrow can \ 1.0 \\
 & NP \rightarrow they \ 0.5 \ | \ fish \ 0.5 \} \\
\end{align*}
\]
Recall that full CKY is **optimal** and **exhaustive**

- For the best parse we keep the most probable partial derivation for every non-terminal at each cell

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$NP_{(they)}^{0.5}$</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$VV_{(can)}^{0.2}$</td>
<td>$VP^{1.0 \times 0.8 \times 0.9 = 0.72}$</td>
<td>([1,2]$<em>{VM}$,[2,3]$</em>{VV}$)</td>
</tr>
<tr>
<td></td>
<td>$VM_{(can)}^{1.0}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$NP_{(fish)}^{0.5}$</td>
<td>$VV_{(fish)}^{0.8}$</td>
<td></td>
</tr>
</tbody>
</table>

$\mathcal{N} = \{S, NP, VP, VV, VM\}$

$\Sigma = \{can, fish, they\}$

$S = S$

$\mathcal{P} = \{S \rightarrow NP \ VP \ 1.0$

$VP \rightarrow VM \ VV \ 0.9$

$VP \rightarrow VV \ NP \ 0.1$

$VV \rightarrow can \ 0.2 \ | \ fish \ 0.8$

$VM \rightarrow can \ 1.0$

$NP \rightarrow they \ 0.5 \ | \ fish \ 0.5 \}$
Recall that full CKY is **optimal** and **exhaustive**

- For the best parse we keep the most probable partial derivation for every non-terminal at each cell

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NP\textsuperscript{0.5}_{(they)}</td>
<td>.</td>
<td>S\textsuperscript{0.5\times1.0\times0.8\times0.9\times1.0=0.36} ([0,1]<em>{NP},[1,3]</em>{VP})</td>
</tr>
<tr>
<td>1</td>
<td>VP\textsuperscript{1.0\times0.8\times0.9=0.72} \textsuperscript{(}[1,2]<em>{VM},[2,3]</em>{VV})</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NP\textsuperscript{0.5}_{(fish)}</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
N & = \{S, NP, VP, VV, VM\} \\
\Sigma & = \{can, fish, they\} \\
S & = S \\
\mathcal{P} & = \{S \rightarrow NP \ VP \ 1.0 \\
& VP \rightarrow VM \ VV \ 0.9 \\
& VP \rightarrow VV \ NP \ 0.1 \\
& VV \rightarrow can \ 0.2 \mid fish \ 0.8 \\
& VM \rightarrow can \ 1.0 \\
& NP \rightarrow they \ 0.5 \mid fish \ 0.5 \}
\end{align*}
\]
For n-best in CKY discard based on beam

An example beam strategy:

- Discard partial derivations based on a score rather than their non-terminal type
- Discard all partial derivations whose score is less than $\alpha$ times the maximum score for that cell
- Typical value for $\alpha$ is 0.0001
- Strategy can cause some loss of accuracy
For n-best in CKY **discard** based on **beam**

- Can discard partial derivations when probability is $\leq \alpha$ times the maximum score for that cell

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$NP^{0.5}_{(they)}$</td>
<td>$S^{0.5<em>0.2</em>0.5<em>0.1</em>1.0=0.005}$($[0,1]<em>{NP},[1,3]</em>{VP}$)</td>
<td>$S^{0.5<em>1.0</em>0.8<em>0.9</em>1.0=0.36}$($[0,1]<em>{NP},[1,3]</em>{VP}$)</td>
</tr>
<tr>
<td>1</td>
<td>$VV^{0.2}_{(can)}$</td>
<td>$VP^{0.2<em>0.5</em>0.1=0.01}$($1\rightarrow([1,2]<em>{VV},[2,3]</em>{NP}$)</td>
<td>$VP^{1.0<em>0.8</em>0.9=0.72}$($2\rightarrow([1,2]<em>{VM},[2,3]</em>{VV}$)</td>
</tr>
<tr>
<td>2</td>
<td>$NP^{0.5}_{(fish)}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Vocabulary**

- $\mathcal{N} = \{ S, NP, VP, VV, VM \}$
- $\Sigma = \{ can, fish, they \}$
- $S = S$
- $\mathcal{P} = \{ S \rightarrow NP \ VP \ 1.0$ $VP \rightarrow VM \ VV \ 0.9$ $VP \rightarrow VV \ NP \ 0.1$ $VV \rightarrow can \ 0.2 \ | \ fish \ 0.8$ $VM \rightarrow can \ 1.0$ $NP \rightarrow they \ 0.5 \ | \ fish \ 0.5 \}$

$\alpha = 0.0001$
For n-best in CKY **discard** based on **beam**

- Can discard partial derivations when probability is $\leq \alpha$ times the maximum score for that cell

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$NP_{(they)}^{0.5}$</td>
<td>$S^{0.5<em>1.0</em>0.8<em>0.9</em>1.0=0.36}$</td>
<td>$([0,1]<em>{NP},[1,3]</em>{VP})$</td>
</tr>
<tr>
<td></td>
<td>$VV_{(can)}^{0.2}$</td>
<td>$VP^{0.2<em>0.5</em>0.1=0.01}$</td>
<td>$([1,2]<em>{VV},[2,3]</em>{NP})$</td>
</tr>
<tr>
<td></td>
<td>$VM_{(can)}^{1.0}$</td>
<td>$VP^{1.0<em>0.8</em>0.9=0.72}$</td>
<td>$([1,2]<em>{VM},[2,3]</em>{VV})$</td>
</tr>
<tr>
<td>2</td>
<td>$VV_{(fish)}^{0.8}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$NP_{(fish)}^{0.5}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$N = \{S, NP, VP, VV, VM\}$

$\Sigma = \{can, fish, they\}$

$S = S$

$P = \{S \rightarrow NP \text{ VP } 1.0 \}$

$VP \rightarrow VM \text{ VV } 0.9$

$VP \rightarrow VV \text{ NP } 0.1$

$VV \rightarrow can \text{ 0.2 } | fish \text{ 0.8}$

$VM \rightarrow can \text{ 1.0}$

$NP \rightarrow they \text{ 0.5 } | fish \text{ 0.5}$

$\alpha = 0.05$
Can discard partial derivations when probability is $\leq \alpha$ times the maximum score for that cell

For n-best in CKY **discard based on beam**

- $NP^{0.5}_{(\text{they})}$
- $S^{0.5\cdot1.0\cdot0.8\cdot0.9\cdot1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$

\[\mathcal{N} = \{S, NP, VP, VV, VM\}\]
\[\Sigma = \{\text{can, fish, they}\}\]
\[S = S\]
\[\mathcal{P} = \{S \rightarrow NP\ VP \ 1.0\]
\[VP \rightarrow VM\ VV \ 0.9\]
\[VP \rightarrow VV\ NP \ 0.1\]
\[VV \rightarrow can \ 0.2\ | \ fish \ 0.8\]
\[VM \rightarrow can \ 1.0\]
\[NP \rightarrow they \ 0.5\ | \ fish \ 0.5 \}\]

$\alpha = 0.2$
For n-best in CKY discard based on beam

- Can discard partial derivations when probability is \( \leq \alpha \) times the maximum score for that cell

```
1     2     3

0  \( NP^{0.5}_{(they)} \) .  \( S^{0.5 \times 1.0 \times 0.8 \times 0.9 \times 1.0 = 0.36} \)

1  \( VM^{1.0}_{(can)} \)  \( VP^{1.0 \times 0.8 \times 0.9 = 0.72} \)

2  \( VV^{0.8}_{(fish)} \)  \( NP^{0.5}_{(fish)} \)
```

\( \mathcal{N} = \{ S, NP, VP, VV, VM \} \)
\( \Sigma = \{ can, fish, they \} \)
\( S = S \)
\( \mathcal{P} = \{ \begin{array}{l} S \rightarrow NP \ VP \ 1.0 \\ VP \rightarrow VM \ VV \ 0.9 \\ VP \rightarrow VV \ NP \ 0.1 \\ VV \rightarrow can \ 0.2 \ | \ fish \ 0.8 \\ VM \rightarrow can \ 1.0 \\ NP \rightarrow they \ 0.5 \ | \ fish \ 0.5 \end{array} \} \)

Question: in what scenario is the best-parse lost when using a beam?
For n-best in CKY **discard** based on **beam**

- Can apply beam dynamically at each cell
- To find n-best, select *n* most probable *S* parses from top right cell

- Alternatively, exploit fact that **2nd best parse will differ from best parse by just 1 of its parsing decisions**
- for nth-best parse all but one of its decisions will be involved in one of the 2nd through the (n -1)th-best parses.
- So first find the best parse, then find the second-best parse, then the third-best, and so on...
- Practically, at each cell keep an **ordered list of n-best partial derivations**, combine with n-best lists for adjacent partial derivations until you have exactly *n* to store in the new cell
Coarse-to-fine n-best strategies, Charniak

Charniak parser adopts a **coarse-to-fine** parsing strategy:

1. produce a parse forest using simple version of the grammar
   i.e. find possible parses using coarse-grained non-terminals, e.g. $VP$

2. refine most promising of coarse-grained parses using complex grammar
   i.e. with feature-based, lexicalised non-terminals, e.g. $VP[\text{buys} / \text{VBZ}]$
Coarse-to-fine n-best strategies, Charniak

- **Coarse-grained step** can be **efficiently parsed** using e.g. CKY
- But the simple grammar **ignores contextual features** so best parse might not be accurate
- **Output a pruned packed parse** forest for the parses generated by the simple grammar (using a beam threshold)
- **Evaluate remaining parses with complex grammar** (i.e. each coarse-grained state is split into several fine-grained states)
- To create **n-best parses** fine-grained step keeps the n-best possibilities at each cell
Discriminative reranking is used to recover best parse

- Use parser to produce n-best list of parses
- Define an **initial ranking** of these parses based on original parse score
- Use **second model** (e.g. max-ent) to **improve the initial ranking** (using additional features)

- Collins re-ranking:
  http://www.aclweb.org/anthology/J05-1003
- Charniak re-ranking:
  https://dl.acm.org/citation.cfm?id=1219862
- Provides small improvements **PARSEVAL** metrics on Penn Treebank
Example of shift-reduce parse for the string **bacdfe**

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
b a c d f e
```
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b a c</td>
<td>d f e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a c d</td>
<td>f e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d f e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Paula Buttery (Computer Lab)  L95: Natural Language Syntax and Parsing
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textbf{Actions:}

- \textit{shift}
- \textit{left-arc} \textit{a} \rightarrow \textit{b}
- \textit{right-arc} \textit{a} \rightarrow \textit{c}
- \textit{shift}
- \textit{shift}
- \textit{shift}
- \textit{left-arc} \textit{e} \rightarrow \textit{f}
- \textit{right-arc} \textit{d} \rightarrow \textit{e}
- \textit{right-arc} \textit{a} \rightarrow \textit{d}
- \textit{terminate}
Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cdfe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

String sequence: $b a c d f e$
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
STACK & BUFFER & ACTION & RECORD \\
\hline
bacdfe & & & \\
b & acdfe & SHIFT & \\
ba & cdfe & SHIFT & \\
& & LEFT-ARC & \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tikzpicture}
\node (b) at (0,0) {b};
\node (a) at (1,0) {a};
\node (c) at (2,0) {c};
\node (d) at (3,0) {d};
\node (f) at (4,0) {f};
\node (e) at (5,0) {e};
\draw[->] (b) -- (a);
\draw[->] (a) -- (c);
\draw[->] (c) -- (d);
\draw[->] (d) -- (f);
\end{tikzpicture}
\end{center}
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ba</td>
<td>LEFT-ARC</td>
<td>a → b</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b a c d f e
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string $bacdfe$

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bcdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>cdfe</td>
<td>LEFT-ARC</td>
<td>$a \rightarrow b$</td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
</tbody>
</table>

```
  b   a   c   d   f   e
  \downarrow
  b   a   c   d   f   e
```
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>cdfe</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
\text{b} \\
\text{a} \\
\text{c} \\
\text{d} \\
\text{f} \\
\text{e}
\end{array}
\]

\[
\begin{array}{c}
\text{a} \\
\text{→} \\
\text{b}
\end{array}
\]
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>c</td>
<td>LEFT-ARC</td>
<td>a → b</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td>a → c</td>
</tr>
</tbody>
</table>

- b → a
- a → c
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bacdfe</td>
<td>shift</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td>shift</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>cdfe</td>
<td>left-arc</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>right-arc</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>shift</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Paula Buttery (Computer Lab)  
L95: Natural Language Syntax and Parsing
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string `bacdfe`

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>LEFT-ARC</td>
<td>a → b</td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td>a → c</td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
</tbody>
</table>
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{bacdfe}</td>
<td>\textit{bacdfe}</td>
<td>\textit{bacdfe}</td>
<td>\textit{bacdfe}</td>
</tr>
<tr>
<td>b</td>
<td>\textit{acdfe}</td>
<td>\textit{acdfe}</td>
<td>\textit{acdfe}</td>
</tr>
<tr>
<td>ba</td>
<td>\textit{cdfe}</td>
<td>\textit{cdfe}</td>
<td>\textit{cdfe}</td>
</tr>
<tr>
<td>a</td>
<td>\textit{dfe}</td>
<td>\textit{dfe}</td>
<td>\textit{dfe}</td>
</tr>
<tr>
<td>ac</td>
<td>\textit{dfe}</td>
<td>\textit{dfe}</td>
<td>\textit{dfe}</td>
</tr>
<tr>
<td>a</td>
<td>\textit{fe}</td>
<td>\textit{fe}</td>
<td>\textit{fe}</td>
</tr>
<tr>
<td>ad</td>
<td></td>
<td></td>
<td>\textit{a \rightarrow b}</td>
</tr>
</tbody>
</table>

Diagram:

\[ \text{b} \quad \text{a} \quad \text{c} \quad \text{d} \quad \text{f} \quad \text{e} \]
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td>a → b</td>
</tr>
<tr>
<td>ba</td>
<td>acdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>LEFT-ARC</td>
<td>a → c</td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>fe</td>
<td>SHIFT</td>
<td></td>
</tr>
</tbody>
</table>

bacdfe

bacdfe

ba

ba

a

ac

ac

a

ad

b

a

c

d

e

f

e
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td>a → b</td>
</tr>
<tr>
<td>ba</td>
<td>acdfe</td>
<td>SHIFT</td>
<td>a → c</td>
</tr>
<tr>
<td>a</td>
<td>cdf</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>adf</td>
<td>e</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
 STACK  |  BUFFER  |  ACTION  |  RECORD
      |          |          |        
 b    | bacdfe   | SHIFT   | a → b  
 b    | acdfe    | SHIFT   |        
 ba   | cdfef    | LEFT-ARC | a → c  
 a    | cdfef    | SHIFT   |        
 ac   | dfe      | RIGHT-ARC |        
 a    | dfe      | SHIFT   |        
 a    | dfe      | SHIFT   |        
 ad   | fef      |         |        
 adf  | e        |         |        
```

Paula Buttery (Computer Lab)  L95: Natural Language Syntax and Parsing
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\textit{bacdfe}</td>
<td>SHIFT</td>
<td>$a \rightarrow b$</td>
</tr>
<tr>
<td>b</td>
<td>\textit{acdfe}</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>\textit{cde}</td>
<td>LEFT-ARC</td>
<td>$a \rightarrow c$</td>
</tr>
<tr>
<td>a</td>
<td>\textit{dfe}</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>\textit{fe}</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>\textit{e}</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adfe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

<table>
<thead>
<tr>
<th>STACK</th>
<th>BUFFER</th>
<th>ACTION</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>acdfe</td>
<td>SHIFT</td>
<td>a → b</td>
</tr>
<tr>
<td>a</td>
<td>cdfef</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>cdfef</td>
<td>SHIFT</td>
<td>a → c</td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>adf</td>
<td>fe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>adfe</td>
<td>e</td>
<td>LEFT-ARC</td>
<td>e → f</td>
</tr>
</tbody>
</table>

Paula Buttery (Computer Lab)  L95: Natural Language Syntax and Parsing  10 / 16
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
STACK       BUFFER       ACTION       RECORD
            bacdfe       SHIFTH       a \rightarrow b
            b           acdfe       SHIFTH       a \rightarrow c
            ba          cdfc        LEFT-ARC
            a           dfe         SHIFT       e \rightarrow f
            ac          fe          SHIFT       
            a           dfe          
            ad          e           
            ade         
```

Paula Buttery (Computer Lab) L95: Natural Language Syntax and Parsing 10 / 16
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
STACK
  b
  ba
  a
  ac
  a
  ad
  adf
  ade

BUFFER
  bacdfe
  acdfe
  cdfe
  dfe
  fe
  e

ACTION
  SHIFT
  SHIFT
  LEFT-ARC
  SHIFT
  RIGHT-ARC
  SHIFT
  SHIFT
  LEFT-ARC
  RIGHT-ARC

RECORD
  a \rightarrow b
  a \rightarrow c
  e \rightarrow f
  d \rightarrow e
```
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

\begin{itemize}
\item b\quad a\quad c\quad d\quad f\quad e
\end{itemize}

\textbf{STACK} | \textbf{BUFFER} | \textbf{ACTION} | \textbf{RECORD}
\hline
b | bacdfe | SHIFT | \hline
ba | acdfe | SHIFT | \hline
a | cdf | LEFT-ARC | \hline
ac | dfe | SHIFT | \hline
a | de | RIGHT-ARC | \hline
ad | e | SHIFT | \hline
adf | | | \hline
adfe | | | \hline
ade | | | \hline
ad | | | \hline
\begin{itemize}
\item a \rightarrow b
\item a \rightarrow c
\item e \rightarrow f
\item d \rightarrow e
\end{itemize}
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string bacdfe

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack
Reminder: the shift-reduce dependency parser

Example of shift-reduce parse for the string *bacdfe*

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

```
STACK | BUFFER | ACTION | RECORD
      |        |        |        
  b    | bacdfe | SHIFT  | a → b  
  b    | acdfe  | SHIFT  | a → c  
  ba   | cdf    | LEFT-ARC | e → f 
  a    | cdf    | SHIFT  | d → e  
  ac   | dfe    | RIGHT-ARC | a → d
  a    |       | SHIFT  |        
  ad   | fe     | SHIFT  |        
  a    | e      | LEFT-ARC |        
  adf  |       | RIGHT-ARC |        
  adfe |       |        |        
  ade  |       |        |        
  ad   |       |        |        
  a    |       |        |        
```
Example of shift-reduce parse for the string \textit{bacdfe}

- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

$$
\begin{array}{c|c|c|c}
\text{STACK} & \text{BUFFER} & \text{ACTION} & \text{RECORD} \\
\hline
b & \text{bacdfe} & \text{SHIFT} & a \rightarrow b \\
ba & \text{acdfe} & \text{SHIFT} & \\
ba & \text{cdfe} & \text{LEFT-ARC} & a \rightarrow c \\
ac & \text{dfe} & \text{SHIFT} & \\
ad & \text{afe} & \text{SHIFT} & \\
adf & \text{adfe} & \text{SHIFT} & \\
adfe & \text{ade} & \text{LEFT-ARC} & e \rightarrow f \\
adf & \text{ade} & \text{RIGHT-ARC} & d \rightarrow e \\
ad & \text{ad} & \text{RIGHT-ARC} & a \rightarrow d \\
a & \text{a} & \text{TERMINATE} & \text{root} \rightarrow a
\end{array}
$$
The shift-reduce parser is **greedy**

- Shift-reduce parser makes a single pass through the sentence making greedy decisions
- Makes the algorithm very efficient, \( O(n) \) for sentence length \( n \)
- Stuck with early decisions no matter how much later evidence contradicts them
Retrieve n-best shift-reduce parses using agenda

- To get the n-best parses we need to systematically explore and score alternative action sequences.
- This gives rise to an exponential number of potential sequences.
- Solution is to score and filter possible sequences to within a fixed beam size.

- Use an agenda to store possible buffer/stack configurations along with a score of the actions that led to that configuration.
- Apply all actions to the top item on the agenda and then score the resulting configurations.
- Add new configurations to the agenda until the beam is full and then replace lowest scoring items with higher scoring ones.
- Continue as long as non-terminating configurations exist on the agenda (guarantees best parse will be found).
Retrieve n-best shift-reduce parses using agenda

- To get the n-best parses we need to systematically explore and score alternative action sequences
- This gives rise to an exponential number of potential sequences
- Solution is to score and filter possible sequences to within a fixed beam size

- Use an agenda to store possible buffer/stack configurations along with a score of the actions that led to that configuration
- **Apply all actions** to top item on the agenda and then score the resulting configurations
- Add new configurations to the agenda until the beam is full and then replace lowest scoring items with higher scoring ones
- Continue as long as non-terminating configurations exist on the agenda (guarantees best parse will be found)
Score reflects **action-sequences** rather than actions

- In the **greedy algorithm** the classifier acted as an **oracle** — **actions are scored**
- With the **beam search** we want to score action sequences — **action sequences are scored**

- Notice that **beam** here is constrained by the size of the agenda
With a beam search we are now searching through the space of decision sequences, so it makes sense to base the score for a configuration on its entire history. More specifically, we can define the score for a new configuration as the score of its predecessor plus the score of the operator used to produce it.

\[
\text{ConfigScore}(c_0) = 0.0
\]

\[
\text{ConfigScore}(c_i) = \text{ConfigScore}(c_{i-1}) + \text{Score}(t_i, c_{i-1})
\]

This score is used both in filtering the agenda and in selecting the final answer.

The new beam search version of transition-based parsing is given in Fig. 13.11.

```pseudo
def DEPENDENCY_BEAMPARSE(words, width) returns dependency tree:
    state ← \{[root], [words], [], 0.0\} ; initial configuration
    agenda ← 〈state〉; initial agenda

    while agenda contains non-final states:
        newagenda ← 〈〉
        for each state ∈ agenda do
            for all \{t | t ∈ VALID_OPERATORS(state)\} do
                child ← APPLY(t, state)
                newagenda ← ADD_TO_BEAM(child, newagenda, width)
                agenda ← newagenda
        return BEST_OF(agenda)

def ADD_TO_BEAM(state, agenda, width) returns updated agenda:
    if LENGTH(agenda) < width then
        agenda ← INSERT(state, agenda)
    else if SCORE(state) > SCORE(WORST_OF(agenda))
        agenda ← REMOVE(WORST_OF(agenda))
        agenda ← INSERT(state, agenda)
    return agenda
```

Psuedo code from Jurafsky and Martin version 3
n-best shift-reduce parser example in class
Next time

- Lexicalised PCFGs
- More on features and training...