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

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge
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.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

they  can  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.

<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></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

$S = S$

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

$VP \rightarrow VM \ VV \ 0.9$

$VP \rightarrow VV \ NP \ 0.1$

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

$VM \rightarrow can \ 1.0$

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

they       can       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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(NP_{(they)}^{0.5})</td>
<td>(VV_{(can)}^{0.2})</td>
<td>(VM_{(can)}^{1.0})</td>
<td></td>
</tr>
</tbody>
</table>

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

```
  1   2   3

0  \text{\textit{NP}}^{0.5}_{(\text{they})}

1  \text{\textit{VV}}^{0.2}_{(\text{can})}
   \text{\textit{VM}}^{1.0}_{(\text{can})}

2
```

\[ \mathcal{N} = \{ S, NP, VP, VV, VM \} \]
\[ \Sigma = \{ \text{can, fish, they} \} \]
\[ S = S \]
\[ \mathcal{P} = \{ S \rightarrow NP \ VP \ 1.0 \}
   VP \rightarrow VM \ VV \ 0.9 
   VP \rightarrow VV \ NP \ 0.1 
   VV \rightarrow \text{can} \ 0.2 \ | \ \text{fish} \ 0.8 
   VM \rightarrow \text{can} \ 1.0 
   NP \rightarrow \text{they} \ 0.5 \ | \ \text{fish} \ 0.5 \} \]
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_{(they)}^{0.5})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(VV_{(can)}^{0.2})</td>
<td>(VM_{(can)}^{1.0})</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>(VV_{(fish)}^{0.8})</td>
<td>(NP_{(fish)}^{0.5})</td>
</tr>
</tbody>
</table>

\[N = \{S, NP, VP, VV, VM\}\]
\[\Sigma = \{can, fish, they\}\]
\[S = S\]
\[P = \{S \rightarrow NP \: VP \: 1.0\}
VP \rightarrow VM \: VV \: 0.9
VP \rightarrow VV \: NP \: 0.1
VV \rightarrow can \: 0.2 \mid fish \: 0.8
VM \rightarrow can \: 1.0
NP \rightarrow they \: 0.5 \mid fish \: 0.5\}\]
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
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
 0  NP_{(they)}
 1  VV_{(can)}^0.2  VP_{([1,2]VM,[2,3]VV)}^{1.0*0.8*0.9=0.72}
 2  VP_{([1,2]VM,[2,3]VV)}^{1.0*0.8*0.9=0.72}
      NP_{(fish)}^0.5
```

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

`they can 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

0  NP\textsuperscript{0.5}_{(\text{they})} . S\textsuperscript{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]NP,[1,3]VP)}

1  VV\textsuperscript{0.2}_{(can)}  VP\textsuperscript{1.0*0.8*0.9=0.72}_{([1,2]VM,[2,3]VV)}

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

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

- Practically, we apply beam dynamically at each cell.
- Typical value for $\alpha$ is **0.0001**
- To find n-best, select $n$ most probable $S$ parses from top right cell.

- Strategy can cause some loss of accuracy.
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.

- Practically, we apply beam dynamically at each cell.
- Typical value for $\alpha$ is 0.0001
- To find n-best, select $n$ most probable $S$ parses from top right cell.

- Strategy can cause some loss of accuracy.
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.

Practically, we apply beam dynamically at each cell.

- Typical value for $\alpha$ is **0.0001**
- To find n-best, select $n$ most probable $S$ parses from top right cell.

Strategy can cause some loss of accuracy.
For n-best in CKY **discard** based on **beam**
For n-best in CKY discard based on n-best lists

- Alternatively, exploit fact that 2nd best parse will differ from best parse by just 1 of its parsing decisions
- 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
For n-best in CKY **discard** based on **n-best lists**

- Alternatively, exploit fact that **2nd best parse will differ from best parse by just 1 of its parsing decisions**
- 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
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 can recover a 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
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></td>
<td></td>
</tr>
<tr>
<td>b a c d f e</td>
<td></td>
<td></td>
<td></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.
Reminder: the shift-reduce dependency parser

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

- 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>bacdf</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>acdfe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>dfe</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>

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>b</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></td>
<td>ade</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$bacdfe$
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></td>
<td></td>
<td></td>
</tr>
<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>
</tbody>
</table>

Left-arc $\text{e} \rightarrow \text{f}$

Right-arc $\text{d} \rightarrow \text{e}$

Right-arc $\text{a} \rightarrow \text{d}$

Terminate $\text{root} \rightarrow \text{a}$
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>bacdfe</td>
<td>acdfe</td>
<td>SHIFT</td>
<td>a → b</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>
</tbody>
</table>

\[ \text{b} \quad \text{a} \quad \text{c} \quad \text{d} \quad \text{f} \quad \text{e} \]
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></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></td>
</tr>
<tr>
<td></td>
<td>adf</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ade</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ade</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$b \rightarrow a$
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></td>
<td>cdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
```
[ b , a , c , d , f ]
```

---

Paula Buttery (Computer Lab)
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>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ac</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>terminate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

```
  b a c d f e
     ^
```

`a → b`
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>bacdfe</td>
<td></td>
<td></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>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td></td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
</tbody>
</table>

\[ \begin{array}{c}
  b \quad a \quad c \quad d \quad f \quad e \\
  \end{array} \]
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>$bacdfe$</td>
<td>SHIFT</td>
<td>$a \rightarrow b$</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>$a \rightarrow c$</td>
</tr>
<tr>
<td>ac</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$bacdfe$

Shift

$bacdfe$

Shift

$ba$

Left-arc $a \rightarrow b$

$ad$

Right-arc $d \rightarrow e$

$ad$

Right-arc $a \rightarrow d$

$a$

Terminate

Root $a \rightarrow a$
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>bacdfe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td>SHIFT</td>
<td>a → b</td>
</tr>
<tr>
<td>ba</td>
<td>cdefe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>cdefe</td>
<td>LEFT-ARC</td>
<td>a → c</td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>SHIFT</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>bacdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>acdfe</td>
<td>SHIFT</td>
<td>a → b</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>a → c</td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>fe</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>bacdfe</td>
<td>acdfe</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 → b</td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>SHIFT</td>
<td></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>
<tr>
<td>ad</td>
<td>fe</td>
<td>SHIFT</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

\begin{align*}
\text{STACK} & \quad \text{BUFFER} & \quad \text{ACTION} & \quad \text{RECORD} \\
& \text{bacdfe} & \text{bacdfe} & \text{SHIFT} & a \rightarrow b \\
b & \text{acidfe} & \text{acdfe} & \text{SHIFT} & \text{} \\
ba & \text{cdfe} & \text{cdefe} & \text{LEFT-ARC} & \text{} \\
a & \text{cdefe} & \text{dfe} & \text{SHIFT} & a \rightarrow c \\
ac & \text{dfe} & \text{efe} & \text{RIGHT-ARC} & \text{} \\
a & \text{dfe} & \text{efe} & \text{SHIFT} & \text{} \\
ad & \text{efe} & \text{e} & \text{SHIFT} & \text{root} \\
adf & \text{efe} & \text{e} & \text{SHIFT} & \text{root} \\
\end{align*}
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 |
 ad | fe    | SHIFT |
 adf | e     | SHIFT |
```

b  a  c  d  f  e

Diagram:

- Shift actions:
  - `b -> b` (no shift, stack remains `b`)
  - `ba -> b` (shift `a`)
  - `a -> a` (shift `c`)
  - `ac -> ac` (shift `d`)
  - `ad -> ad` (shift `f`)
  - `adf -> adf` (shift `e`)

- Arc actions:
  - `a -> b` (left arc)
  - `a -> c` (right arc)
  - `a -> d` (right arc)

- Terminating with root symbol `a`
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{align*}
\text{STACK} & \quad \text{BUFFER} & \quad \text{ACTION} & \quad \text{RECORD} \\
& \quad \text{bacdfe} & \quad \text{shift} & \quad \text{a} \rightarrow \text{b} \\
\text{b} & \quad \text{acdfe} & \quad \text{shift} & \quad \text{a} \rightarrow \text{c} \\
\text{ba} & \quad \text{cdfe} & \quad \text{left-arc} & \quad \text{a} \rightarrow \text{d} \\
\text{a} & \quad \text{dfe} & \quad \text{right-arc} & \quad \text{a} \rightarrow \text{e} \\
\text{ac} & \quad \text{fe} & \quad \text{terminate} & \quad \text{root} \\
\text{a} & \quad \text{fe} & & \\
ad & \quad \text{e} & & \\
adf & & & \\
adfe & & & \\
\end{align*}
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>cdfef</td>
<td>LEFT-ARC</td>
<td>e → f</td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>dfef</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>adf</td>
<td>dfef</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>adfe</td>
<td>dfef</td>
<td>LEFT-ARC</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- Stack: b a c d f e
- Buffer: bacdfe, acdfe, cdfef, dfe, dfef, dfef
- Action: SHIFT, LEFT-ARC, RIGHT-ARC, SHIFT
- Record: a → b, a → c, e → f
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
             bacdfe          SHIFT
b            acdfe          SHIFT
ba            cdf e         LEFT-ARC
a            dfe            Shift
ac            dfe            RIGHT-ARC
a            fe             Shift
ad            e             Shift
adf           e             LEFT-ARC
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 $bacdfe$

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

```
STACK | BUFFER | ACTION | RECORD
------|--------|--------|--------
| b     | b      | SHIFT  | a → b  |
| ba    | a      | SHIFT  | a → c  |
| a     | ac     | LEFT-ARC | a → f  |
| ac    | d     | RIGHT-ARC | d → e  |
| a    | ad    |      |        |
| ad   | adfe  |      |        |
| 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 *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
    ba       cdfe
    a        dfe
    ac       e
    d        a
    a        c
    ad       b
    adf      d
    ade      e
    adfe
    ade
    ad
    e        f
    d        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 \textbf{STACK} \hspace{2cm} \textbf{BUFFER} \hspace{2cm} \textbf{ACTION} \hspace{2cm} \textbf{RECORD}
  \begin{align*}
  &b \quad \text{bacdfe} \quad \text{SHIFT} \quad a \rightarrow b \\
  &ba \quad \text{acdfe} \quad \text{SHIFT} \\
  &a \quad \text{cdfe} \quad \text{LEFT-ARC} \quad a \rightarrow c \\
  &ac \quad \text{dfe} \quad \text{SHIFT} \\
  &a \quad \text{dfe} \quad \text{SHIFT} \\
  &ad \quad \text{fe} \quad \text{SHIFT} \\
  &adf \quad \text{e} \quad \text{LEFT-ARC} \quad e \rightarrow f \\
  &adfe \quad \text{ade} \quad \text{RIGHT-ARC} \\
  &ad \quad \text{ad} \quad \text{RIGHT-ARC} \\
  & \quad \text{a} \quad \text{d} \quad \text{a} \rightarrow d
  \end{align*}
\end{itemize}
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{tabular}{l|l|l|l}
STACK & BUFFER & ACTION & RECORD \\
\hline
b & bacdfe & \textsc{shift} & a \rightarrow b \\
ba & acdfe & \textsc{shift} & \\
am & cdf & \textsc{left-arc} & a \rightarrow c \\
ac & dfe & \textsc{shift} & \\
d & fe & \textsc{shift} & \\
adf & e & \textsc{shift} & \\
adfe & & \textsc{left-arc} & e \rightarrow f \\
adfe & & \textsc{right-arc} & d \rightarrow e \\
ad & & \textsc{right-arc} & a \rightarrow d \\
a & & & \\
\end{tabular}
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>ba</td>
<td>acdfe</td>
<td>LEFT-ARC</td>
<td>a \rightarrow b</td>
</tr>
<tr>
<td>a</td>
<td>cdfe</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>dfe</td>
<td>RIGHT-ARC</td>
<td>a \rightarrow c</td>
</tr>
<tr>
<td>a</td>
<td>dfe</td>
<td>SHIFT</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>LEFT-ARC</td>
<td>e \rightarrow f</td>
</tr>
<tr>
<td>adfe</td>
<td></td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>ade</td>
<td></td>
<td>RIGHT-ARC</td>
<td>d \rightarrow e</td>
</tr>
<tr>
<td>ad</td>
<td></td>
<td>RIGHT-ARC</td>
<td>a \rightarrow d</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>TERMINATE</td>
<td>\textit{root} \rightarrow a</td>
</tr>
</tbody>
</table>
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 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
Graph-Based Dependency Parsing

Graph-based approaches to dependency parsing search through the space of possible trees for a given sentence for a tree (or trees) that maximize some score. These methods encode the search space as directed graphs and employ methods drawn from graph theory to search the space for optimal solutions. More formally, given a sentence $S$ we're looking for the best dependency tree in $G_S$, the space of all possible trees for that sentence, that maximizes some score.

$$\hat{T}(S) = \arg\max_{t \in G_S} \text{score}(t, S)$$

As with the probabilistic approaches to context-free parsing discussed in Chapter 12, the overall score for a tree can be viewed as a function of the scores of the parts of the tree. The focus of this section is on edge-factored approaches where the edge-factored

```python
function DEPENDENCYBEAMPARSE(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 ∈ VALIDOPERATORS(state) } do
                child ← APPLY(t, state)
                newagenda ← ADDTOBEAM(child, newagenda, width)
            agenda ← newagenda
        return BESTOF(agenda)

function ADDTOBEAM(state, agenda, width) returns updated agenda
    if LENGTH(agenda) < width then
        agenda ← INSERT(state, agenda)
    else if SCORE(state) > SCORE(WORSTOF(agenda))
        agenda ← REMOVE(WORSTOF(agenda))
        agenda ← INSERT(state, agenda)
    return agenda
```

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

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