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 parsing 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

	1	2	3
0			
1			
2			
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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	$NP^{0.5}_{(they)}$		
1			
2			
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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	$NP^{0.5}_{(they)}$		
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	
2			
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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	$NP^{0.5}_{(they)}$	.	
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	
2			
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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	$NP^{0.5}_{(they)}$	.	
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

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

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	$NP^{0.5}_{(they)}$	.	
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	$VP^{0.2*0.5*0.1=0.01}_{1\rightarrow([1,2]_{VV},[2,3]_{NP})}$ $VP^{1.0*0.8*0.9=0.72}_{2\rightarrow([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

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

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	$NP^{0.5}_{(they)}$	.	
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	$VP^{1.0*0.8*0.9=0.72}_{([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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	$NP^{0.5}_{(they)}$	.	$S^{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	$VP^{1.0*0.8*0.9=0.72}_{([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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 α times the maximum score for that cell
- Typical value for α 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

	1	2	3
0	$NP^{0.5}_{(they)}$	.	$S^{0.5*0.2*0.5*0.1*1.0=0.005}_{([0,1]_{NP},[1,3]_{VP})}$ $S^{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	$VP^{0.2*0.5*0.1=0.01}_{1\rightarrow([1,2]_{VV},[2,3]_{NP})}$ $VP^{1.0*0.8*0.9=0.72}_{2\rightarrow([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

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

	1	2	3
0	$NP^{0.5}_{(they)}$	.	$S^{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$
1		$VV^{0.2}_{(can)}$ $VM^{1.0}_{(can)}$	$VP^{0.2*0.5*0.1=0.01}_{1\rightarrow([1,2]_{VV},[2,3]_{NP})}$ $VP^{1.0*0.8*0.9=0.72}_{2\rightarrow([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

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

$\alpha = 0.05$

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	$NP^{0.5}_{(they)}$	.	$S^{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$
1		$VM^{1.0}_{(can)}$	$VP^{1.0*0.8*0.9=0.72}_{([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

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

$\alpha = 0.2$

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	$NP^{0.5}_{(they)}$	.	$S^{0.5*1.0*0.8*0.9*1.0=0.36}_{([0,1]_{NP},[1,3]_{VP})}$
1		$VM^{1.0}_{(can)}$	$VP^{1.0*0.8*0.9=0.72}_{([1,2]_{VM},[2,3]_{VV})}$
2			$VV^{0.8}_{(fish)}$ $NP^{0.5}_{(fish)}$
	they	*can*	*fish*

$$
\begin{array}{rcl}
\mathcal{N} & = & \{S, NP, VP, VV, VM\} \\
\Sigma & = & \{can, fish, they\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP\ 1.0 \\
 & & VP \rightarrow VM\ VV\ 0.9 \\
 & & VP \rightarrow VV\ NP\ 0.1 \\
 & & VV \rightarrow can\ 0.2 \mid fish\ 0.8 \\
 & & VM \rightarrow can\ 1.0 \\
 & & NP \rightarrow they\ 0.5 \mid fish\ 0.5\ \}
\end{array}
$$

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*[*buys*/*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

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		

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

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	

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

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe		

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

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	

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

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe		

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

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$

Reminder: the shift-reduce dependency parser

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

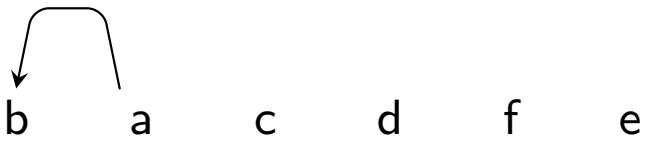
- Actions selected from a classifier based on the features of the configuration of items on the buffer and stack

STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe		

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

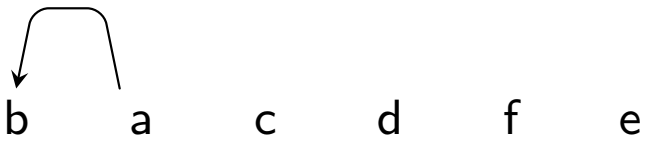


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	

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

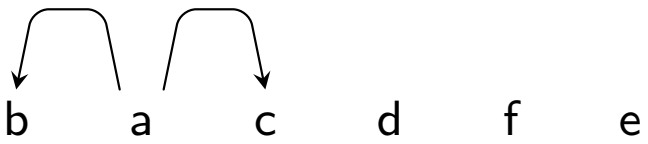


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe		

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

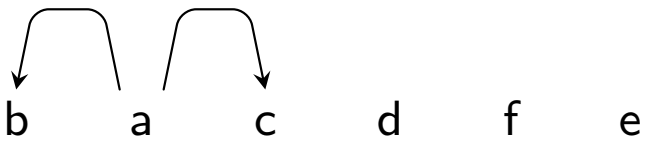


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$

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

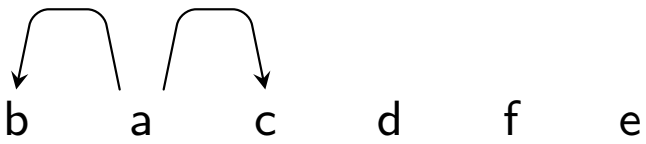


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe		

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

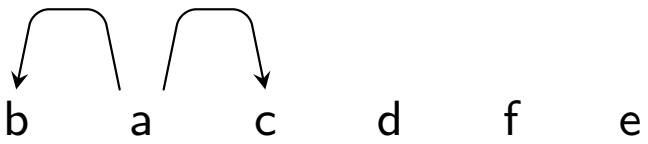


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	

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

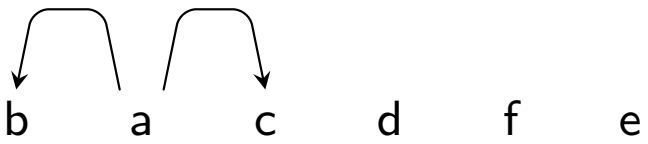


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe		

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

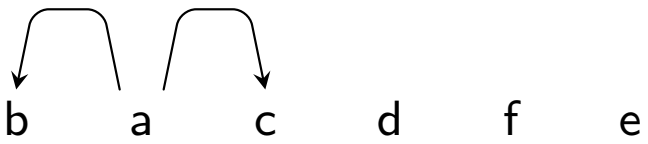


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	

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

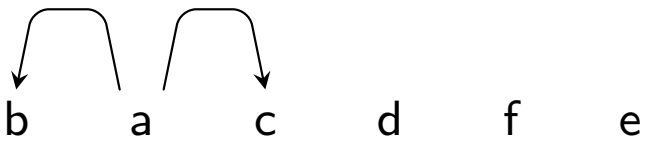


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e		

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

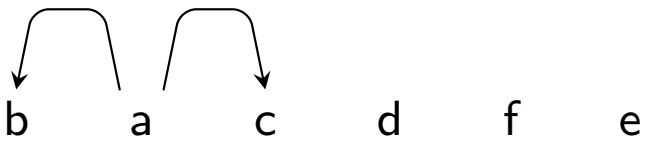


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	

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

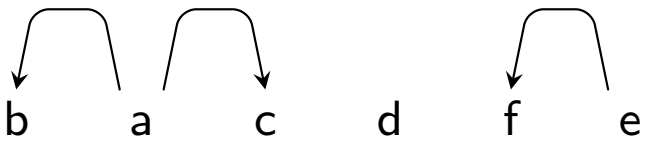


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe			

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

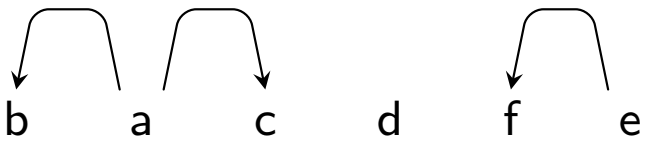


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$

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

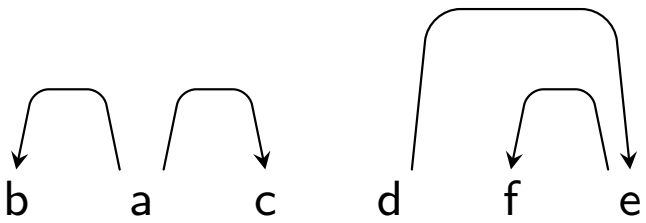


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade			

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

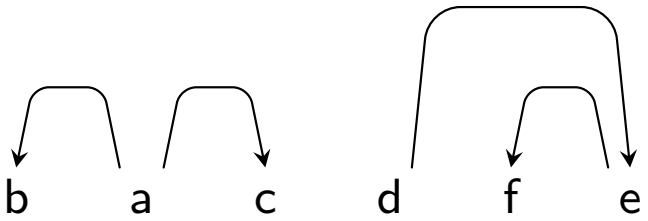


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade		RIGHT-ARC	$d \rightarrow e$

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

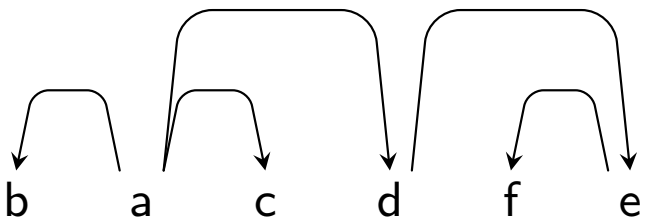


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade		RIGHT-ARC	$d \rightarrow e$
ad			

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

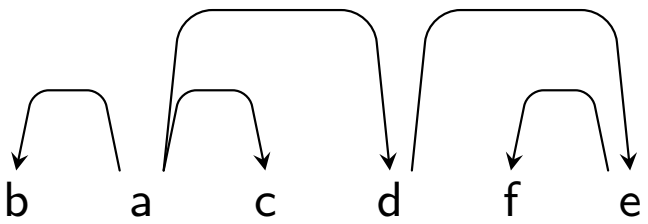


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade		RIGHT-ARC	$d \rightarrow e$
ad		RIGHT-ARC	$a \rightarrow d$

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

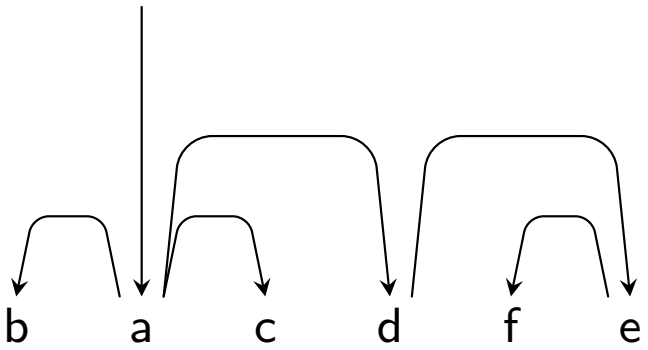


STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade		RIGHT-ARC	$d \rightarrow e$
ad		RIGHT-ARC	$a \rightarrow d$
a			

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



STACK	BUFFER	ACTION	RECORD
	bacdfe	SHIFT	
b	acdfe	SHIFT	
ba	cdfe	LEFT-ARC	$a \rightarrow b$
a	cdfe	SHIFT	
ac	dfe	RIGHT-ARC	$a \rightarrow c$
a	dfe	SHIFT	
ad	fe	SHIFT	
adf	e	SHIFT	
adfe		LEFT-ARC	$e \rightarrow f$
ade		RIGHT-ARC	$d \rightarrow e$
ad		RIGHT-ARC	$a \rightarrow d$
a		TERMINATE	$root \rightarrow a$

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 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 let 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 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 let 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

N-best dependency parse algorithm

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 in class

Next time

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