COMS W4705: Natural Language Processing (Fall 2018)
Problem Set #2

Wenbo Gao - `wg2313@columbia.edu`

October 10, 2018

Problem 1 - PCFGs and HHMs

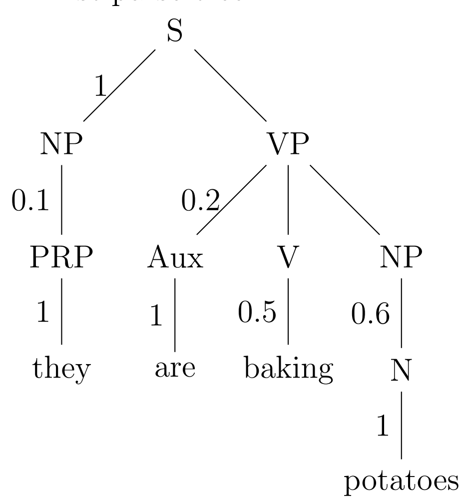
Both PCFGs and HMMs can be seen as generative models that produce a sequence of POS tags and words with some probability (of course the PCFG will generate even more structure, but it will also generate POS tags and words).

(a)

Problem. Revisit the example sentence "*they are baking potatoes*" and grammar from Problem 2. For each sequence of POS tags that is possible for this sentence according to the grammar, what is the joint probability $P(\text{tags}, \text{words})$ according to the PCFG? Hint: consider all parses for the sentence.

Solution.

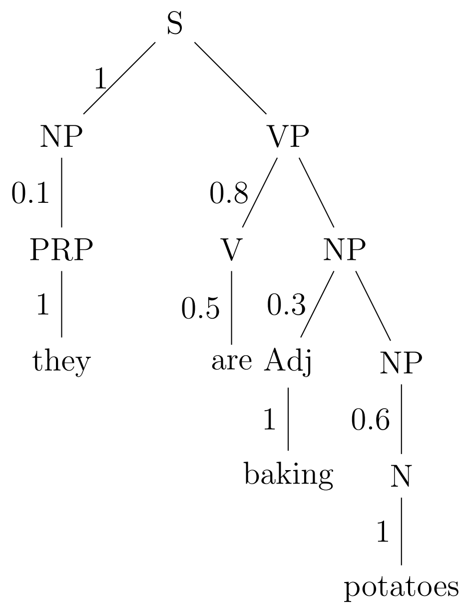
first parse tree:



There is only one possible parse tree (the first parse tree) given tags = {PRP, Aux, V, N}, and words = {they, are, baking, potatoes}, thus

$$
\begin{aligned}
&P[tags = \{PRP, Aux, V, N\}, words = \{they, are, baking, potatoes\}] \\
=&\sum_{tree \in \{\text{parses tree with}\{they, are, baking, potatoes\}\text{as leaf nodes and}\{PRP, Aux, V, N\}\text{one layer above}\}} P[tree] \\
=&P[\text{first parse tree}] \\
=&0.006
\end{aligned}
$$

second parse tree:

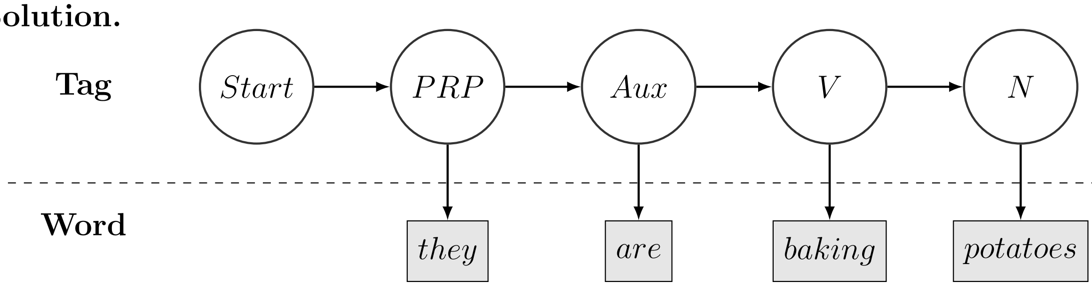


There is only one possible parse tree given tags = { PRP, V, Adj, N}, and words = {they, are, baking, potatoes}, thus

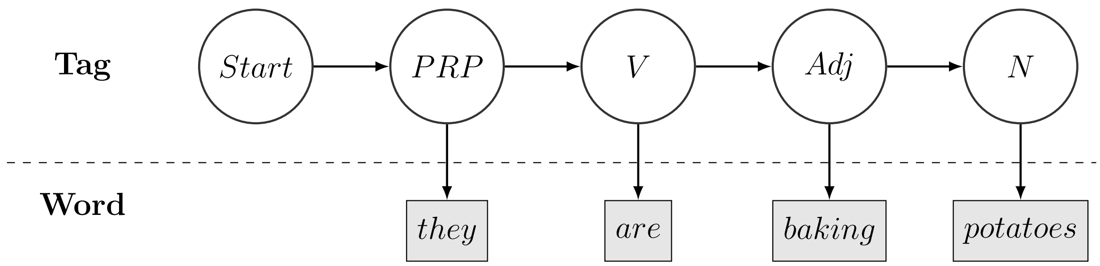
$$
\begin{aligned}
&P[tags = \{PRP, V, Adj, N\}, words = \{they, are, baking, potatoes\}] \\
=&\sum_{tree \in \{\text{parse trees with}\{they, are, baking, potatoes\}\text{as leaf nodes and}\{PRP, V, Adj, N\}\text{one layer above}\}} P[tree] \\
=&P[\text{second parse tree}] \\
=&0.0072
\end{aligned}
$$

(b)

Solution.



$$
\begin{aligned}
&P[tags = \{PRP, Aux, V, N\}, words = \{they, are, baking, potatoes\}]\\
=&P[tag = PRP|tag = Start] \cdot P[word = they|tag = PRP]\\
&\cdot P[tag = Aux|tag = PRP] \cdot P[word = are|tag = Aux]\\
&\cdot P[tag = V|tag = Aux] \cdot P[word = baking|tag = V]\\
&\cdot P[tag = N|tag = V] \cdot P[word = potatoes|tag = N]\\
=&0.006
\end{aligned}
$$



$$
\begin{aligned}
&P[tags = \{PRP, V, Adj, N\}, words = \{they, are, baking, potatoes\}]\\
=&P[tag = PRP|tag = Start] \cdot P[word = they|tag = PRP]\\
&\cdot P[tag = V|tag = PRP] \cdot P[word = are|tag = V]\\
&\cdot P[tag = Adj|tag = V] \cdot P[word = baking|tag = Adj]\\
&\cdot P[tag = N|tag = Adj] \cdot P[word = potatoes|tag = N]\\
=&0.0072
\end{aligned}
$$

Here is my HMM:

1. tag-word transition distributions (the same as in PCFG):
   - $P[word = they|tag = PRP] = 1$
   - $P[word = are|tag = Aux] = 1$
   - $P[word = baking|tag = V] = 0.5$, $P[word = are|tag = V] = 0.5$
   - $P[word = potatoes|tag = N] = 1$
2. tag-tag transition distributions:
   - $P[tag = PRP|tag = Start] = 1$
   - $P[tag = Aux|tag = PRP] = 0.5$, $P[tag = V|tag = PRP] = 0.5$
   - $P[tag = N|tag = V] = 0.5$, $P[tag = Adj|tag = V] = 0.5$
   - $P[tag = V|tag = Aux] = 0.048$, $P[tag = PRP|tag = Aux] = 0.952$
   - $P[tag = N|tag = Adj] = 0.0576$
   - $P[tag = PRP|tag = N] = 1$

(c)

Problem. In general, is it possible to translate any PCFG into an HMM that produces the identical *joint* probability P(tags, words) as PCFG (i.e. not just for a single sentence)? Explain how or why not. No formal proof is necessary. Hint: This has nothing to do with probabilities, but with language classes.

Solution. The hidden/latent state space in HMM can be formally described by a finite state machine which is strictly less expressive than pushdown automaton / context-free grammar in PCFG.

Problem 2

Consider the following probabilistic context free grammar.

$$\begin{array}{rll} S \rightarrow NP\ VP & & [1.0] \\ NP \rightarrow Adj\ NP & & [0.3] \\ NP \rightarrow PRP & & [0.1] \\ NP \rightarrow N & & [0.6] \\ VP \rightarrow V\ NP & & [0.8] \\ VP \rightarrow Aux\ V\ NP & & [0.2] \\ PRP \rightarrow they & & [1.0] \\ N \rightarrow potatoes & & [1.0] \\ Adj \rightarrow baking & & [1.0] \\ V \rightarrow baking & & [0.5] \\ V \rightarrow are & & [0.5] \\ Aux \rightarrow are & & [1.0] \end{array}$$

(a)

Problem. Using this grammar, show how the **Earley algorithm** would parse the following sentence.

they are baking potatoes

Write down the complete parse chart. The chart should contain $n + 1$ entries where n is the length of the sentence. Each entry i should contain *all* parser items generated by the parser that end in position i. You can ignore the probability for part (a).

Solution.

Chart[0]

> init

$s0$: S $\rightarrow \cdot$ NP VP [0,0]

> predict $s0$, ${}_0|\text{NP}|_1$

$s1$: NP $\rightarrow \cdot$ Adj NP [0,0]

$s2$: NP $\rightarrow \cdot$ PRP [0,0]

$s3$: NP $\rightarrow \cdot$ N [0,0]

> predict $s1$, ${}_0|\text{Adj}|_1$

$s4$: Adj $\rightarrow \cdot$ baking [0,0]

> predict $s2$, ${}_0|\text{PRP}|_1$

$s5$: PRP $\rightarrow \cdot$ they [0,0]

> predict $s3$, ${}_0|\text{N}|_1$

$s6$: N $\rightarrow \cdot$ potatoes [0,0]

> scan $s4$, ${}_0|\text{baking}|_1$, failure

> scan $s5$, ${}_0|\text{they}|_1$, success

Chart[1]

$s7$: PRP $\rightarrow$ they$\cdot$ [0,1]
> scan $s6$, $_0$|potatoes|$_1$, failure
> complete with $s7$, $_0$|PRP|$_1$, traverse Chart[0]
. hit $s2$
$s8$: NP $\rightarrow$ PRP $\cdot$ [0,1]
> complete with $s8$, $_0$|NP|$_1$, traverse Chart[0]
. hit $s0$
$s9$: S $\rightarrow$ NP $\cdot$ VP [0,1]
> predict $s9$, $_1$|VP|$_2$
$s10$: VP $\rightarrow$ $\cdot$ V NP [1,1]
$s11$: VP $\rightarrow$ $\cdot$ Aux V NP [1,1]
> predict $s10$, $_1$|V|$_2$
$s12$: V $\rightarrow$ $\cdot$ baking [1,1]
$s13$: V $\rightarrow$ $\cdot$ are [1,1]
> predict $s11$, $_1$|Aux|$_2$
$s14$: Aux $\rightarrow$ $\cdot$ are [1,1]
> scan $s12$, $_1$|baking|$_2$, failure
> scan $s13$, $_1$|are|$_2$, success

Chart[2]

$s15$: V $\rightarrow$ are $\cdot$ [1,2]
> scan $s14$, $_1$|are|$_2$, success
$s16$: Aux $\rightarrow$ are $\cdot$ [1,2]
> complete with $s15$, $_1$|V|$_2$, traverse Chart[1]
. hit $s10$
$s17$: VP $\rightarrow$ V $\cdot$ NP [1,2]
> complete with $s16$, $_1$|Aux|$_2$, traverse Chart[1]
. hit $s11$
$s18$: VP $\rightarrow$ Aux $\cdot$ V NP [1,2]
> predict $s17$, $_2$|NP|$_3$
$s19$: NP $\rightarrow$ $\cdot$Adj NP [2,2]
$s20$: NP $\rightarrow$ $\cdot$PRP [2,2]
$s21$: NP $\rightarrow$ $\cdot$N [2,2]
> predict $s18$, $_2$|V|$_3$
$s22$: V $\rightarrow$ $\cdot$ baking [2,2]
$s23$: V $\rightarrow$ $\cdot$ are [2,2]
> predict $s19$, $_2$|Adj|$_3$
$s24$: Adj $\rightarrow$ $\cdot$ baking [2,2]
> predict $s20$, $_2$|PRP|$_3$
$s25$: PRP $\rightarrow$ $\cdot$ they [2,2]
> predict $s21$, $_2$|N|$_3$
$s26$: N $\rightarrow$ $\cdot$ potatoes [2,2]
> scan $s22$, $_2$|baking|$_3$, success

Chart[3]

$s27$: V $\rightarrow$ baking $\cdot$ [2,3]
> scan $s23$, ${}_2|\text{are}|_3$, failure
> scan $s24$, ${}_2|\text{baking}|_3$, success
$s28$: Adj $\rightarrow$ baking $\cdot$ [2,3]
> scan $s25$, ${}_2|\text{they}|_3$, failure
> scan $s26$, ${}_2|\text{potatoes}|_3$, failure
> complete with $s27$, ${}_2|\text{V}|_3$, traverse Chart[2]
. hit $s18$
$s29$: VP $\rightarrow$ Aux V $\cdot$ NP [1,3]
> complete with $s28$, ${}_2|\text{Adj}|_3$, traverse Chart[3]
. hit $s19$
$s30$: NP $\rightarrow$ Adj $\cdot$ NP [2,3]
> predict $s29$, ${}_3|\text{NP}|_4$
$s31$: NP $\rightarrow$ $\cdot$ Adj NP [3,3]
$s32$: NP $\rightarrow$ $\cdot$ PRP [3,3]
$s33$: NP $\rightarrow$ $\cdot$ N [3,3]
> predict $s30$, ${}_3|\text{NP}|_4$, duplicate
> predict $s31$, ${}_3|\text{Adj}|_4$
$s34$: Adj $\rightarrow$ $\cdot$ baking [3,3]
> predict $s32$, ${}_3|\text{PRP}|_4$
$s35$: PRP $\rightarrow$ $\cdot$ they [3,3]
> predict $s33$, ${}_3|\text{N}|_4$
$s36$: N $\rightarrow$ $\cdot$ potatoes [3,3]
> scan $s34$, ${}_3|\text{baking}|_4$, failure
> scan $s35$, ${}_3|\text{they}|_4$, failure
> scan $s36$, ${}_3|\text{potatoes}|_4$, success

Chart[4]

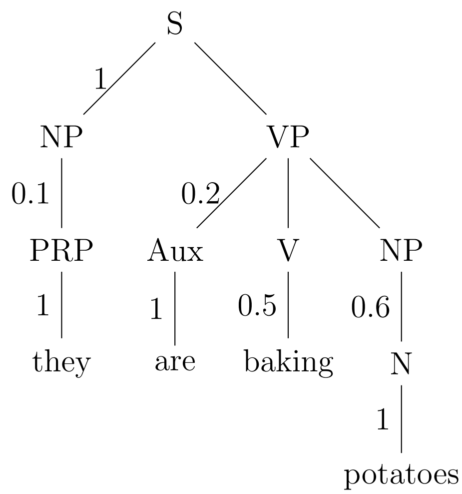
$s37$: N $\rightarrow$ potatoes $\cdot$ [3,4]
> complete with $s37$, ${}_3|\text{N}|_4$, traverse Chart[3]
. hit $s33$
$s38$: NP $\rightarrow$ N $\cdot$ [3,4]
> complete with $s38$, ${}_3|\text{NP}|_4$, traverse Chart[3]
. hit $s29$
$s39$: VP $\rightarrow$ Aux V NP $\cdot$ [1,4]
. hit $s30$
$s40$: NP $\rightarrow$ Adj NP $\cdot$ [2,4]
> complete with $s39$, ${}_1|\text{VP}|_4$, traverse Chart[1]
. hit $s9$
$s41$: S $\rightarrow$ NP VP $\cdot$ [0,4], **solution 1**($s9$, $s39$)
> complete with $s40$, ${}_2|\text{NP}|_4$, traverse Chart[2]
. hit $s17$
$s42$: VP $\rightarrow$ V NP $\cdot$ [1,4]
> complete with $s42$, ${}_1|\text{VP}|_4$, traverse Chart[1]

. hit $s9$
S → NP VP · [0,4], duplicate of $s41$, **solution 2**($s9$, $s42$)

(b)

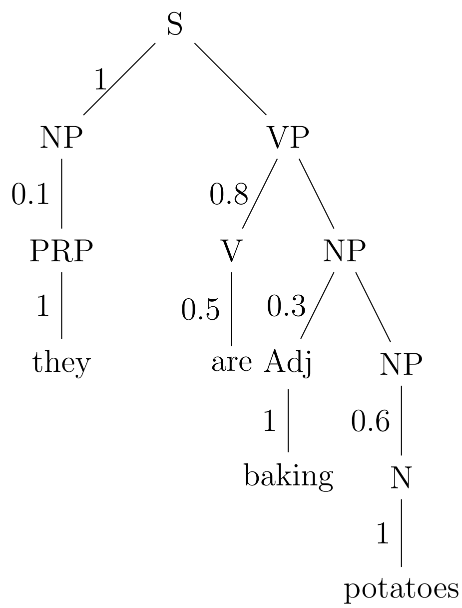
Problem. Write down *all* parse trees for the sentence and grammar from problem 2 and compute their probabilities according to the PCFG.

Solution. first parse tree:



$$
\begin{aligned}
&P[\text{first parse tree}]\\
=&P[S \to NP\ VP] \cdot P[NP \to PRP] \cdot P[VP \to Aux\ V\ NP] \cdot P[PRP \to \mathit{they}]\\
&\cdot P[Aux \to \mathit{are}] \cdot P[V \to \mathit{baking}] \cdot P[NP \to N] \cdot P[N \to \mathit{potatoes}]\\
=&1 \cdot 0.1 \cdot 0.2 \cdot 1 \cdot 1 \cdot 0.5 \cdot 0.6 \cdot 1\\
=&0.006
\end{aligned}
$$

second parse tree:



$$
\begin{aligned}
&P[\text{second parse tree}]\\
=&P[S \to NP\ VP] \cdot P[NP \to PRP] \cdot P[VP \to V\ \ NP] \cdot P[PRP \to they]\\
&\cdot P[V \to are] \cdot P[NP \to Adj\ \ NP] \cdot P[Adj \to baking] \cdot P[NP \to N] \cdot P[N \to potatoes]\\
=&1 \cdot 0.1 \cdot 0.8 \cdot 1 \cdot 0.5 \cdot 0.3 \cdot 1 \cdot 0.6 \cdot 1\\
=&0.0072
\end{aligned}
$$

Problem 3 - CKY parsing

(a)

Problem. Convert the grammar from problem 2 into an equivalent grammra in Chomsky Normal Form (CNF). Write down the new grammar. Also explain what the general rule is for dealing with

1. Rules of the form A → B (i.e. a single non-terminal no the right hand side).
2. Rules with three or more non-terminals on the right hand side (e.g. A → B C D E).

You don not have to deal with the case in which terminals and non-terminals are mixed in a rule right-hand side. You also do not have to convert the probabilites. Hint: Think about adding new non-terminal symbols.

Solution.

New grammar:

S → NP VP	[0.3]
S → PRP VP	[0.1]
S → N VP	[0.6]
NP → Adj NP	[0.3]
NP → Adj PRP	[0.1]
NP → Adj N	[0.6]
VP → V NP	[0.24]
VP → V PRP	[0.08]
VP → V N	[0.48]
VP → AV NP	[0.06]
VP → AV PRP	[0.02]
VP → AV N	[0.12]
AV → Aux V	[1.0]

General rules:

1. for A → B
   - find all rules with A on the right-hand side, e.g. C → A D
   - for each of them, substitute A on the right-hand side by B, i.e. C → B D
   - add the modified rules to the rule set, and remove the original rule
2. for A → B C D E
   - add non-terminals { F, G }
   - add rules:
     (a) A → B F

(b) F → C G

(c) G → D E

- remove the original rule

(b)

Problem. Using your grammar, fill the CKY parse chart as shown in class and show all parse trees.

Solution. See Figure 1.

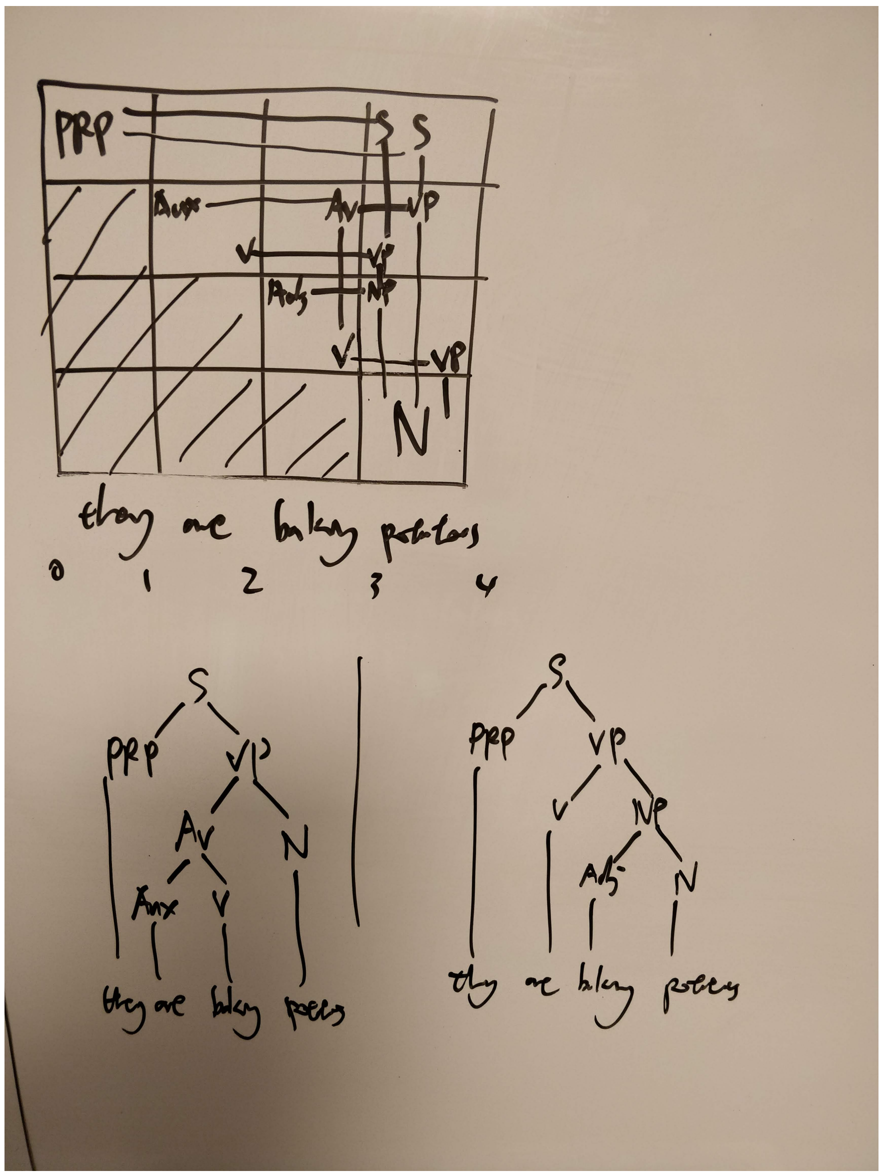
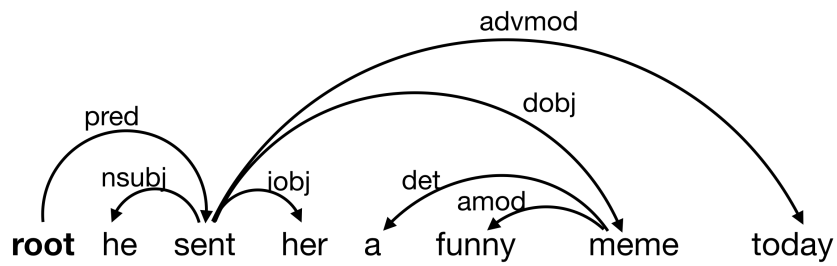


Figure 1: problem 3 (b)

Problem 4 - Transition Based Dependency Parsing

Problem. Consider the following dependency graph. Write down the sequence of transitions



that an arc-standard dependency parser would have to take to generate this dependency tree from the initial state.

([**root**]σ, [he, sent, her, a, funny, meme, today]β, {}A)

Also write down the state resulting from each transition.

Solution.
s0: [root], [he, send, her, a, funny, meme, today], []
> shift
s1: [root, he], [send, her, a, funny, meme, today], []
> Left-Arc$_{\text{nsubj}}$
s2: [root], [send, her, a, funny, meme, today], [(send, nsubj, he)]
> shift
s3: [root, send], [her, a, funny, meme, today], [...]
> Right-Arc$_{\text{iobj}}$
s4: [root], [send, a, funny, meme, today], [..., (send, iobj, her)]
> shift
s5: [root, send], [a, funny, meme, today], [...]
> shift
s6: [root, send, a], [funny, meme, today], [...]
> shift
s7: [root, send, a, funny], [meme, today], [...]
> Left-Arc$_{\text{amod}}$
s8: [root, send, a], [meme, today], [..., (meme, amod, funny)]
> Left-Arc$_{\text{det}}$
s9: [root, send], [meme, today], [..., (meme, det, a)]
> Right-Arc$_{\text{dobj}}$
s10: [root], [send, today], [..., (send, dobj, meme)]
> shift
s11: [root, send], [today], [...]
> Right-Arc$_{\text{advmod}}$
s12: [root], [send], [..., (send, advmod, today)]
> Right-Arc$_{\text{pred}}$

s13: [], [root], [..., (root, pred, send)]
$>$ shift
s14: [root], [], [...]