COMS W4705: Natural Language Processing (sec:002) - Homework #2

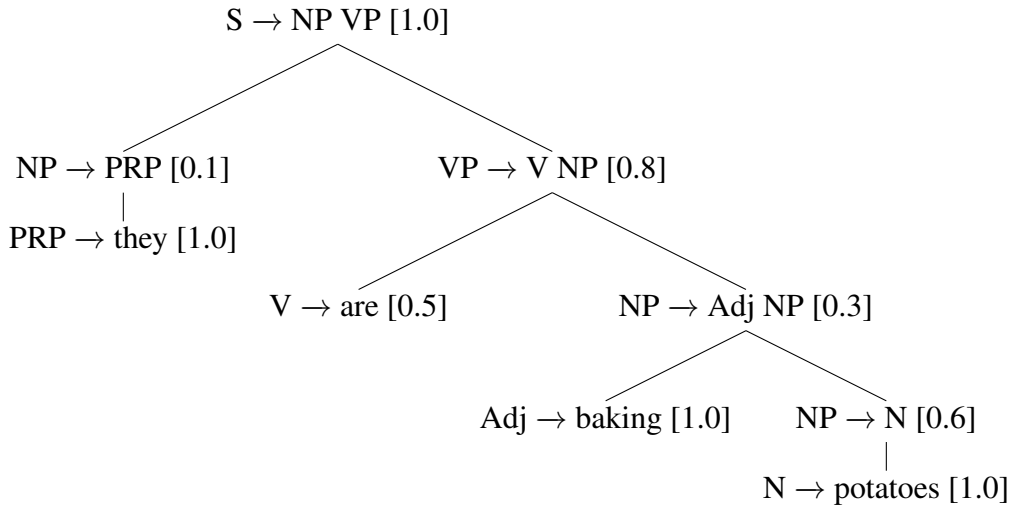
Name: Geraldi Dzakwan (gd2551)

October 13, 2019

Problem 1

a. Taking the solution to Problem 2, there are two possible parses (sequence of tags) for the sentence, which are:

1.



In other words, the sequence of tags are:

tags1 = [PRP, V, Adj, N]

There are two ways of computing the joint probability P(tags1, words):

1. The first way is just to use the parse tree probability from Problem 2 for the first parse tree, which equals to $\mathbf{0.0072}$. They are the same value basically because when we create a parse tree, we also give tags to the words at the same time.
   Reference: https://piazza.com/class/k09vvrvx2l846o?cid=124
2. Another way to compute the joint probability is by using the HMM created in Problem 1b. Note that NP and VP nonterminals aren't used (the detail on

why and how is in the answer for Problem 1b). Then, the joint probability for tags1 would be:

$$P(tags1, words) = \prod_{i=1}^{4} P(t_i|t_{i-1})P(w_i|t_i)$$

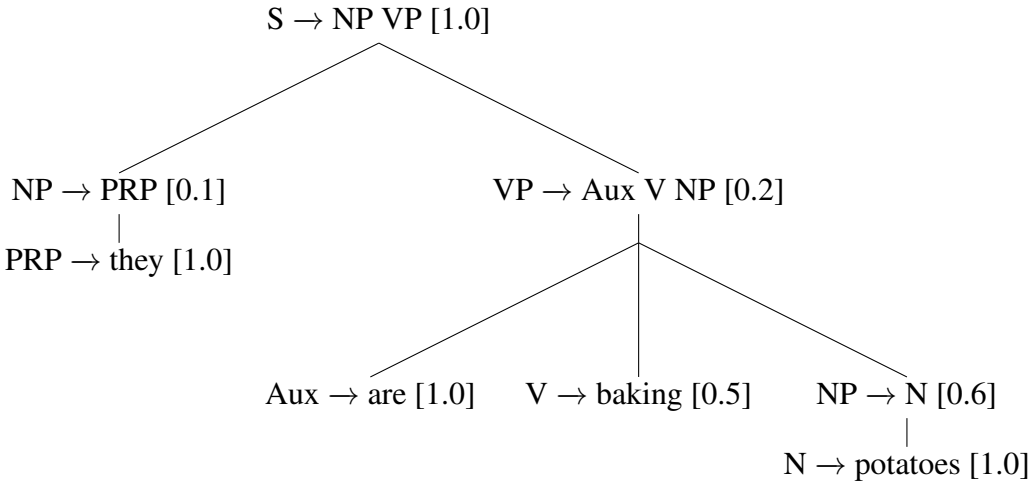
$$P(tags1, words) = P(PRP|S) * P(they|PRP) * P(V|PRP) * P(are|V) * P(Adj|V) * P(baking|Adj) * P(N|Adj) * P(potatoes|N)$$

$$P(tags1, words) = 0.1 * 1 * 0.8 * 0.5 * 0.3 * 1.0 * 0.6 * 1.0 = \mathbf{0.0072}$$

Both yield the same result, thus:

$$\boldsymbol{P(tags1, word) = 0.0072}$$

2.



In other words, the sequence of tags are:

tags2 = [PRP, Aux, V, N]

There are two ways of computing the joint probability P(tags2, words):

1. Just like before, the first way is just to use the parse tree probability from Problem 2 for the second parse tree, which equals to $\mathbf{0.006}$.
2. Another way to compute the joint probability is by using the HMM created in Problem 1b. Then, the joint probability for tags2 would be:

$$P(tags2, words) = \prod_{i=1}^{4} P(t_i|t_{i-1})P(w_i|t_i)$$

$$P(tags2, words) = P(PRP|S) * P(they|PRP) * P(Aux|PRP) * P(are|Aux) * P(V|Aux) * P(baking|V) * P(N|V) * P(potatoes|N)$$

$$P(tags2, words) = 0.1 * 1 * 0.2 * 1.0 * 1.0 * 0.5 * 0.6 * 1.0 = \mathbf{0.006}$$

Both yield the same result, thus:

$$\boldsymbol{P(tags2, word) = 0.006}$$

b. First, define our HMM states (POS tags) and observations (words):

1. Set of states: {S, Adj, PRP, N, V, Aux}
   Note that I omit VP and NP in this scenario because they don't directly produce a terminal (word). In later part, I will show how I handle this when calculating transition probability.
2. Set of observations: {they, potatoes, baking, are}

Then, we can calculate our transition probabilities (only for those which aren't zero). Let's consider the rule S→NP VP. There are two transitions in this case, which are S→NP and NP→VP. Let's examine each transition and let TP be transition probability.

1. S → NP
   As I've said before, NP is not part of the set of states. To handle that, we can replace NP with its production rules instead: NP → Adj NP, NP → PRP and NP→N. Note that these will result in initial transition probabilities because S is the start. Thus, we will have initial transition probabilities:

(TP1) S → Adj [0.3] thus P(Adj|S) = 0.3

(TP2) S → PRP [0.1] thus P(PRP|S) = 0.1

(TP3) S → N [0.6] thus P(N|S) = 0.6

Next, we should notice that since NP can recurse (NP → Adj NP), we will have to also assign transition probabilities from Adj to itself, to PRP and to N:

(TP4) Adj → Adj [0.3] thus P(Adj|Adj) = 0.3

(TP5) Adj → PRP [0.1] thus P(PRP|Adj) = 0.1

(TP6) Adj → N [0.6] thus P(N|Adj) = 0.6

2. NP → VP
   Again, as I've said before, VP is not part of the set of states. To handle that, we can replace VP with its production rules instead: VP → V NP, VP → Aux V NP. Notice that we will go to VP only if the previous state is PRP or N (if it's Adj we have to recurse for NP). Thus, we will have these transmission probabilities:

(TP7) PRP → V [0.8] thus P(V|PRP) = 0.8

(TP8) PRP → Aux [0.2] thus P(Aux|PRP) = 0.2

(TP9) N → V [0.8] thus P(V|N) = 0.8

(TP10) N $\rightarrow$ Aux [0.2] thus P(Aux|N) = 0.2

Next, we know that Aux will always be followed by V, so:

(TP11) Aux $\rightarrow$ V [1.0] thus P(V|Aux) = 1.0

Finally, V is always followed by NP, and we can then again divide the transitions into three cases:

(TP12) V $\rightarrow$ Adj [0.3] thus P(Adj|V) = 0.3

(TP13) V $\rightarrow$ PRP [0.1] thus P(PRP|V) = 0.1

(TP14) V $\rightarrow$ N [0.6] thus P(N|V) = 0.6

Those above are valid probabilities because the probability of every transition that starts with the same state will sum to one, e.g. TP4 + TP5 + TP6 = 1. We can now calculate the emission probabilities by just copying them from the PCFG:

(EP1) PRP $\rightarrow$ they [1.0] thus P(they|PRP) = 1.0

(EP2) N $\rightarrow$ potatoes [1.0] thus P(potatoes|N) = 1.0

(EP3) Adj $\rightarrow$ baking [1.0] thus P(baking|Adj) = 1.0

(EP4) V $\rightarrow$ baking [0.5] thus P(baking|V) = 0.5

(EP5) V $\rightarrow$ are [0.5] thus P(are|V) = 0.5

(EP6) Aux $\rightarrow$ are [1.0] thus P(are|Aux) = 1.0

To summarize, the HMM is defined as below:

1. Set of states: {S, Adj, PRP, N, V, Aux}
2. Set of observations: {they, potatoes, baking, are}
3. Set of transition probabilities. There are 14 transition probabilities that are non zero, given by TP1 - TP14 which are already computed above. Three of them are the initial transition probabilities (TP1 - TP3).
4. Set of emission probabilities. There are 6 emission probabilities that are non zero, given by EP1 - EP6 which are already computed above.

c. In general, the answer is $\boldsymbol{no}$. Some PCFG can be translated into its corresponding HMM, just like PCFG in Problem 2. But, there is also some PCFG that can't be translated into an HMM that produces the same joint probability. Consider a PCFG that produces sentences in which there are some number of nouns followed by the same number of verbs $(N^nV^n, n > 1)$. It will have production rules as below.

$$\text{S} \rightarrow \text{N S V [0.25]}$$

$$\text{S} \rightarrow \text{N V [0.75]}$$

The 0.25 are 0.75 probabilities are picked arbitrarily just for an example. For any sentence in the form of N^nV^n, the probability of that sentence would be $(0.25)^{n-1}(0.75)$.

Any other sentences will have zero probability. In this case, there will be no HMM that can have the same probability distribution as that PCFG. The theoretical reason is because HMM doesn't track how many times a terminal has already been emitted. Its state and transition is modeled as a finite state automata, thus no memory is involved. We can of course have some HMM that accepts the same pattern as that PCFG (N^nV^n), but it will also accept some other patterns, i.e. it doesn't give zero probability to sentences other than N^nV^n. For example, consider an HMM:

$$\text{P(N|S)} = 1.0$$

$$\text{P(N|N)} = 0.25$$

$$\text{P(V|N)} = 0.75$$

$$\text{P(V|V)} = 1.0$$

No terminals are involved for simplicity. We can assume that there is only one terminal that can be emitted from N and V. Then, for any sentence N^nV^n we will have a joint probability of:

$$\text{P(N}^n\text{V}^n\text{)} = \text{P(N|S)} * \text{P(N|N)}^{n-1} * \text{P(V|N)} * \text{P(V|V)}^{n-1}$$

$$\text{P(N}^n\text{V}^n\text{)} = (1.0) * (0.25)^{n-1} * (0.75) * (1.0)^{n-1}$$

$$\text{P(N}^n\text{V}^n\text{)} = (0.25)^{n-1}(0.75)$$

This is the same joint probability as our PCFG. But, notice that this HMM will also accept some other sentences like:

1. N^nV^{n+1}

$$\text{P(N}^n\text{V}^{n+1}\text{)} = (1.0) * (0.25)^{n-1} * (0.75) * (1.0)^{n} = (0.25)^{n-1}(0.75)$$

2. $N^{n+1}V^n$

$$\text{P(N}^{n+1}\text{V}^n\text{)} = (1.0) * (0.25)^{n} * (0.75) * (1.0)^{n-1} = (0.25)^{n}(0.75)$$

These above make the HMM produces different joint probability from that of the PCFG. Some sentences other than N^nV^n are also accepted, i.e. they have nonzero probabilities. To summarize, in general, not all PCFG can be translated into an HMM that produces the same joint probability. We can think of HMM as a model with finite state that can only handle regular grammar, meanwhile PCFG is a more general model that can handle context free grammar. One difference between regular and context free grammar is that context free grammar can represent center embeddings, i.e. the N^nV^n example that I explain above.

Problem 2

Symbols used for this problem: Sn $\rightarrow$ n-th state; Chart[i] $\rightarrow$ Chart at position i, 0 to 4 (sentence length + 1); Op: {Perform/Scan/Complete} Sn $\rightarrow$ operation performed on Sn; w[i] $\rightarrow$ word at index i, 0 to 3. Below is the complete Earley parse chart:

- Chart[0]

  S0 S $\rightarrow$. NP VP [0,0]

  S1 NP $\rightarrow$. Adj NP [0,0] Op: Predict S1

  S2 NP $\rightarrow$. PRP [0,0] Op: Predict S1

  S3 NP $\rightarrow$. N [0,0] Op: Predict S1

  S4 Adj $\rightarrow$. baking [0,0] Op: Predict S2

  S5 PRP $\rightarrow$. they [0,0] Op: Predict S3

  S6 N $\rightarrow$. potatoes [0,0] Op: Predict S4

  Chart[0] is done. Since w[0] = ”they”, scan operation on S4 and S6 will fail. Scan operation on S5 will succeed, thus we add shifted S5 to Chart[1].

- Chart[1]

  S7 PRP $\rightarrow$ they . [0,1] Op: Scan S5

  S8 NP $\rightarrow$ PRP . [0,1] Op: Complete S7 using S2

  S9 S $\rightarrow$ NP . VP [0,1] Op: Complete S8 using S0

  S10 VP $\rightarrow$. V NP [1,1] Op: Predict S9

  S11 VP $\rightarrow$. Aux V NP [1,1] Op: Predict S9

  S12 V $\rightarrow$. baking [1,1] Op: Predict S10

  S13 V $\rightarrow$. are [1,1] Op: Predict S10

  S14 Aux $\rightarrow$. are [1,1] Op: Predict S11

  Chart[1] is done. Since w[1] = ”are”, scan operation on S12 will fail. Scan operation on S13 and S14 will succeed, thus we add shifted S13 and S14 to Chart[2].

- Chart[2]

  S15 V $\rightarrow$ are . [1,2] Op: Scan S13

  S16 Aux $\rightarrow$ are . [1,2] Op: Scan S14

  S17 VP $\rightarrow$ V . NP [1,2] Op: Complete S15 using S9

  S18 VP $\rightarrow$ Aux . V NP [1,2] Op: Complete S16 using S11

  S19 NP $\rightarrow$. Adj NP [2,2] Op: Predict S17

S20 NP → . PRP [2,2] Op: Predict S17

S21 NP → . N [2,2] Op: Predict S17

S22 V → . baking [2,2] Op: Predict S18

S23 Adj → . baking [2,2] Op: Predict S19

S24 PRP → . they [2,2] Op: Predict S20

S25 N → . potatoes [2,2] Op: Predict S21

Chart[2] is done. Since w[2] = ”baking”, scan operation on S24 and S25 will fail. Scan operation on S22 and S23 will succeed, and we add shifted S22 and S23 to Chart[3].

- Chart[3]

S26 V → baking . [2,3] Op: Scan S22

S27 Adj → baking . [2,3] Op: Scan S23

S28 VP → Aux V . NP [1,3] Op: Complete S26 using S18

S29 NP → Adj . NP [2,3] Op: Complete S27 using S19

S30 NP → . Adj NP [3,3] Op: Predict S28 and S29 (both are NP)

S31 NP → . PRP [3,3] Op: Predict S28 and S29 (both are NP)

S32 NP → . N [3,3] Op: Predict S28 and S29 (both are NP)

S33 Adj → . baking [3,3] Op: Predict S30

S34 PRP → . they [3,3] Op: Predict S31

S35 N → . potatoes [3,3] Op: Predict S32

Chart[3] is done. Since w[3] = ”potatoes”, scan operation on S33 and S34 will fail. Scan operation on S35 will succeed and we add shifted S35 to Chart[4].

- Chart[4]

S36 N → potatoes . [3,4] Op: Scan S35

S37 NP → N . [3,4] Op: Complete S36 using S32

S38 VP → Aux V NP . [1,4] Op: Complete S37 using S28

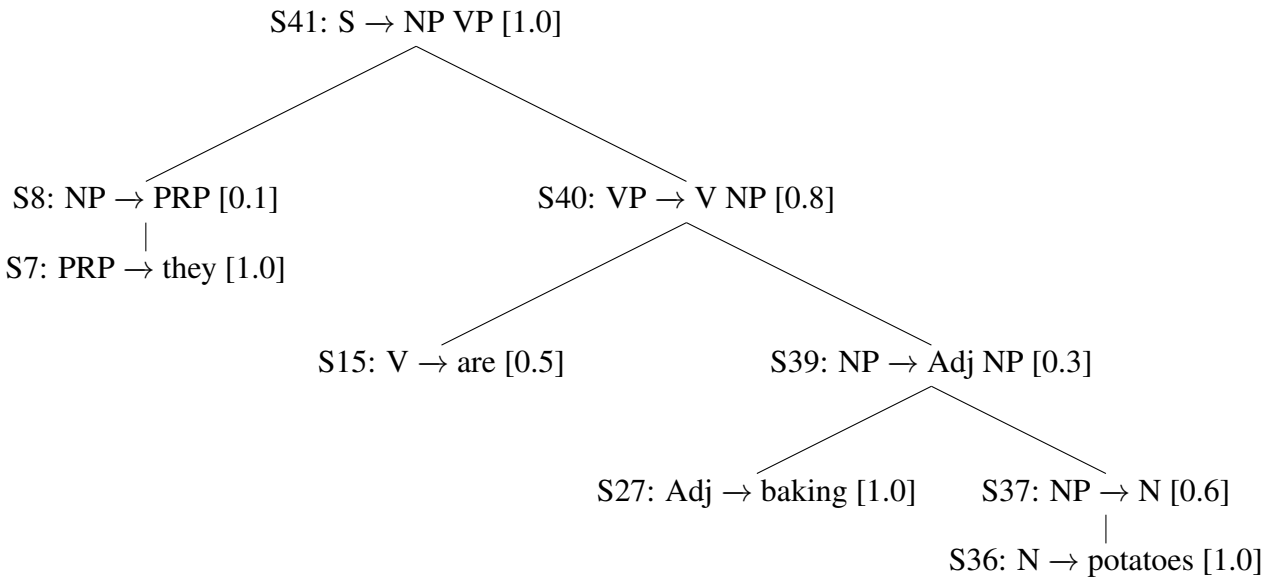
S39 NP → Adj NP . [2,4] Op: Complete S37 using S29

S40 VP → V NP . [1,4] Op: Complete S39 using S17

S41 S → NP VP . [0,4] Op: Complete S38 and S40 using S9 (two backpointers)

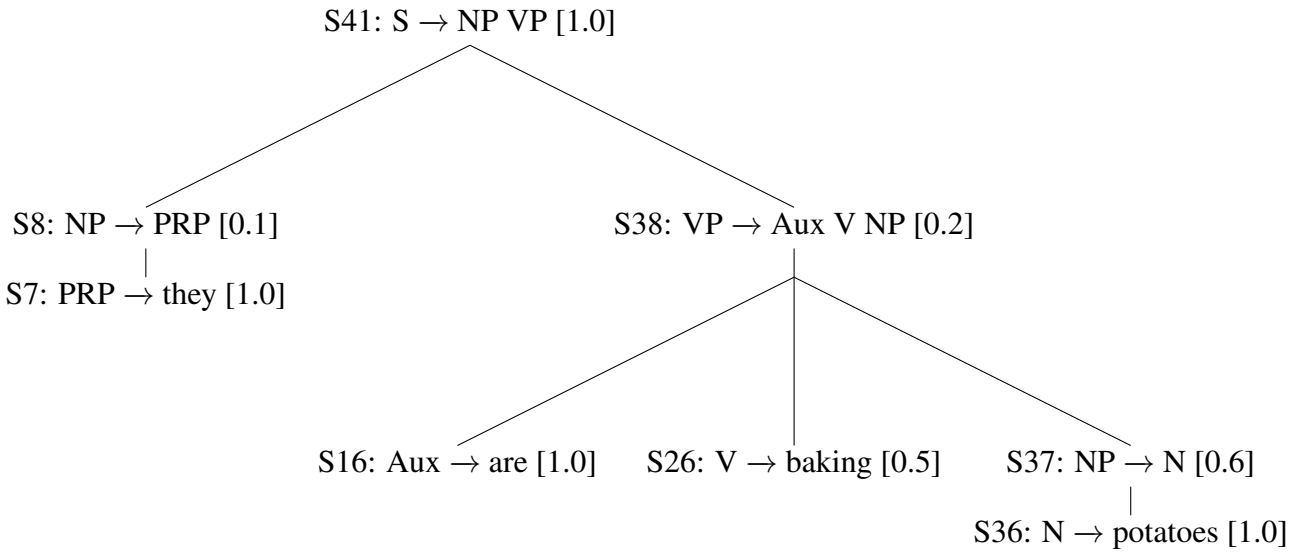
Chart[4] is done, we have visited all the states and ended up with S and full span [0,4], meaning that the sentence belongs to the PCFG and we can get its parse trees.

b. There are two parse trees that can be traced. The trace, the production rule and the probability are shown in each node of the tree. For example, S41: S $\rightarrow$ NP VP [1.0] means that S41 is the state/trace, S $\rightarrow$ NP VP is the rule and [1.0] is the rule probability.



The first parse tree probability according to the PCFG is:

$$P(t_1) = \prod_{i=1}^{9} P(A_i \rightarrow B_i) = (1.0)(0.1)(1.0)(0.8)(0.5)(0.3)(1.0)(0.6)(1.0) = \mathbf{0.0072}$$



The second parse tree probability according to the PCFG is:

$$P(t_2) = \prod_{i=1}^{8} P(A_i \rightarrow B_i) = (1.0)(0.1)(1.0)(0.2)(1.0)(0.5)(0.6)(1.0) = \mathbf{0.006}$$

Problem 3

a. The general rules to convert an arbitrary CFG to CNF are as below:

1. For a CFG rule A→B, we can simply replace B with its production rules. For example, if these are the production rules for B:

$$B \rightarrow CD, B \rightarrow b \text{ (b is a terminal)}$$

Then, we will have:

$$A \rightarrow CD, A \rightarrow b$$

We can then remove B from our grammar. Note that we have to replace B with A if B appears on some other rules right hand side. For example, if there is a rule C→BD, then change it to C→AD.

2. For a CFG rule A→BCDE, we can introduce one or more new nonterminals (as needed), so that A will produce exactly two nonterminals. For example, let's create two new nonterminals F and G where they have these production rules:

$$F \rightarrow BC, G \rightarrow DE$$

Then, we will have:

$$A \rightarrow FG$$

PCFG in Problem 2 has 3 rules that are needed to be modified in order to have a CNF form of the grammar. Below are the needed modifications:

1. NP→PRP. We can use rule 1. There is only one production rule for PRP: PRP→they. Thus, the modification will be:

$$NP \rightarrow \text{they}$$

We can then remove PRP from our language. Since PRP doesn't appear on any rule right hand side now, no further modification is needed.

2. NP→N. Using the same logic as above, knowing the production rule of N→potatoes, the modification will be:

$$NP \rightarrow \text{potatoes}$$

We can then remove N from our language. Since N doesn't appear on any rule right hand side now, no further modification is needed.

3. VP→Aux V NP. We can use rule 2. Let's introduce a new nonterminal AV that has this production rule: AV→Aux V. Thus, we will have:

$$VP \rightarrow AV\ NP$$

Finally, by applying above modifications, we will end up with this CNF:
S → NP VP [1.0]
NP → Adj NP [0.3]
NP → they [0.1]
NP → potatoes [0.6]
VP → V NP [0.8]
VP → AV NP [0.2]
AV → Aux V [1.0]
Adj → baking [1.0]
V → baking [0.5]
V → are [0.5]
Aux → are [1.0]

b. The CKY chart is depicted below in Table 1. The symbol "X" means that there are no production rules that apply for that cell range. For example, in cell [0,2], NP V and NP Aux don't appear on any rules right hand side, so we put "X" there. Any nonterminal that is paired with X will also result in X. For example, X Adj and X V will all be X so we put X in cell [0,3]. This makes sense since there are no rules with one terminal on the right hand side for CNF.

	0 they 1	1 are 2	2 baking 3	3 potatoes 4
0	[0,1] NP	[0,2] X	[0,3] X	[0,4] S
1		[1,2] V Aux	[1,3] AV	[1,4] VP
2			[2,3] Adj, V	[2,4] NP, VP
3				[3,4] NP

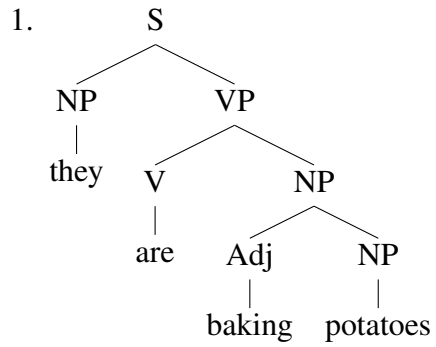
Table 1: CKY Chart for "They are baking potatoes"

I find it's a little bit messy to put the backpointers in the table, so I list them below.

1. [1,3][AV] = [1,2][Aux] + [2,3][V]
2. [2,4][NP] = [2,3][Adj] + [3,4][NP]
3. [2,4][VP] = [2,3][V] + [3,4][NP]
4. [1,4][VP] = [1,2][V] + [2,4][NP]
5. [1,4][VP] = [1,3][AV] + [3,4][NP]
6. [0,4][S] = [0,1][NP] + [1,4][VP]

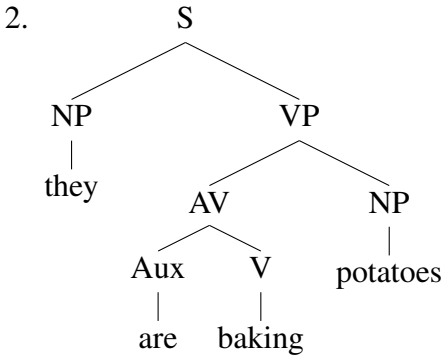
There are two parse trees yielded:

1.



The first parse tree is as above. This somehow makes less sense since ”baking potatoes” is treated as a noun phrase that explains the subject ”they”, something very unlikely in the real world. But, this one has higher probability somehow according to the answer for Problem 2b.

2.



The second parse tree is as above. This is the better parse tree in my opinion since ”baking” should be more probable to act as a verb in this sentence even though this tree got lower probability.

Problem 4

Let S0 be our initial state as described in the problem.

S0 Next Operation: SHIFT

$$([\boldsymbol{root}]_\sigma, [\text{he, sent, her, a, funny, meme, today}]_\beta, []_\alpha)$$

S1 Next Operation: LEFT ARC

$$([\boldsymbol{root}, \text{he}]_\sigma, [\text{sent, her, a, funny, meme, today}]_\beta, []_\alpha)$$

S2 Next Operation: SHIFT

$$([\boldsymbol{root}]_\sigma, [\text{sent, her, a, funny, meme, today}]_\beta, [\{\text{sent, r, he}\}]_\alpha)$$

S3 Next Operation: RIGHT ARC

$$([\boldsymbol{root}, \text{sent}]_\sigma, [\text{her, a, funny, meme, today}]_\beta, [\{\text{sent, r, he}\}]_\alpha)$$

S4 Next Operation: SHIFT

$$([\boldsymbol{root}]_\sigma, [\text{sent, a, funny, meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}]_\alpha)$$

S5 Next Operation: SHIFT

$$([\boldsymbol{root}, \text{sent}]_\sigma, [\text{a, funny, meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}]_\alpha)$$

S6 Next Operation: SHIFT

$$([\boldsymbol{root}, \text{sent, a}]_\sigma, [\text{funny, meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}]_\alpha)$$

S7 Next Operation: LEFT ARC

$$([\boldsymbol{root}, \text{sent, a, funny}]_\sigma, [\text{meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}]_\alpha)$$

S8 Next Operation: LEFT ARC

$$([\boldsymbol{root}, \text{sent, a}]_\sigma, [\text{meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}, \{\text{meme, r, funny}\}]_\alpha)$$

S9 Next Operation: RIGHT ARC

$$([\boldsymbol{root}, \text{sent}]_\sigma, [\text{meme, today}]_\beta, [\{\text{sent, r, he}\}, \{\text{sent, r, her}\}, \{\text{meme, r, funny}\}, \{\text{meme, r, a}\}]_\alpha)$$

S10 Next Operation: SHIFT

($[\boldsymbol{root}]_\sigma$, [sent, today]$_\beta$, [{sent, r, he}, {sent, r, her}, {meme, r, funny}, {meme, r, a}, {sent, r, meme}]$_\alpha$)

S11 Next Operation: RIGHT ARC

($[\boldsymbol{root}$, sent]$_\sigma$, [today]$_\beta$, [{sent, r, he}, {sent, r, her}, {meme, r, funny}, {meme, r, a}, {sent, r, meme}]$_\alpha$)

S12 Next Operation: RIGHT ARC

($[\boldsymbol{root}]_\sigma$, [sent]$_\beta$, [{sent, r, he}, {sent, r, her}, {meme, r, funny}, {meme, r, a}, {sent, r, meme}, {sent, r, today}]$_\alpha$)

S13 Next Operation: SHIFT

([]$_\sigma$, $[\boldsymbol{root}]_\beta$, [{sent, r, he}, {sent, r, her}, {meme, r, funny}, {meme, r, a}, {sent, r, meme}, {sent, r, today}, {$\boldsymbol{root}$, r, sent}]$_\alpha$)

S14 Next Operation: -.

($[\boldsymbol{root}]_\sigma$, []$_\beta$, [{sent, r, he}, {sent, r, her}, {meme, r, funny}, {meme, r, a}, {sent, r, meme}, {sent, r, today}, {$\boldsymbol{root}$, r, sent}]$_\alpha$)

Buffer is empty and stack contains single word (root). Terminal condition is reached. There are 14 transitions done:

1. SHIFT
2. LEFT ARC
3. SHIFT
4. RIGHT ARC
5. SHIFT
6. SHIFT
7. SHIFT
8. LEFT ARC
9. LEFT ARC
10. RIGHT ARC
11. SHIFT
12. RIGHT ARC
13. RIGHT ARC
14. SHIFT