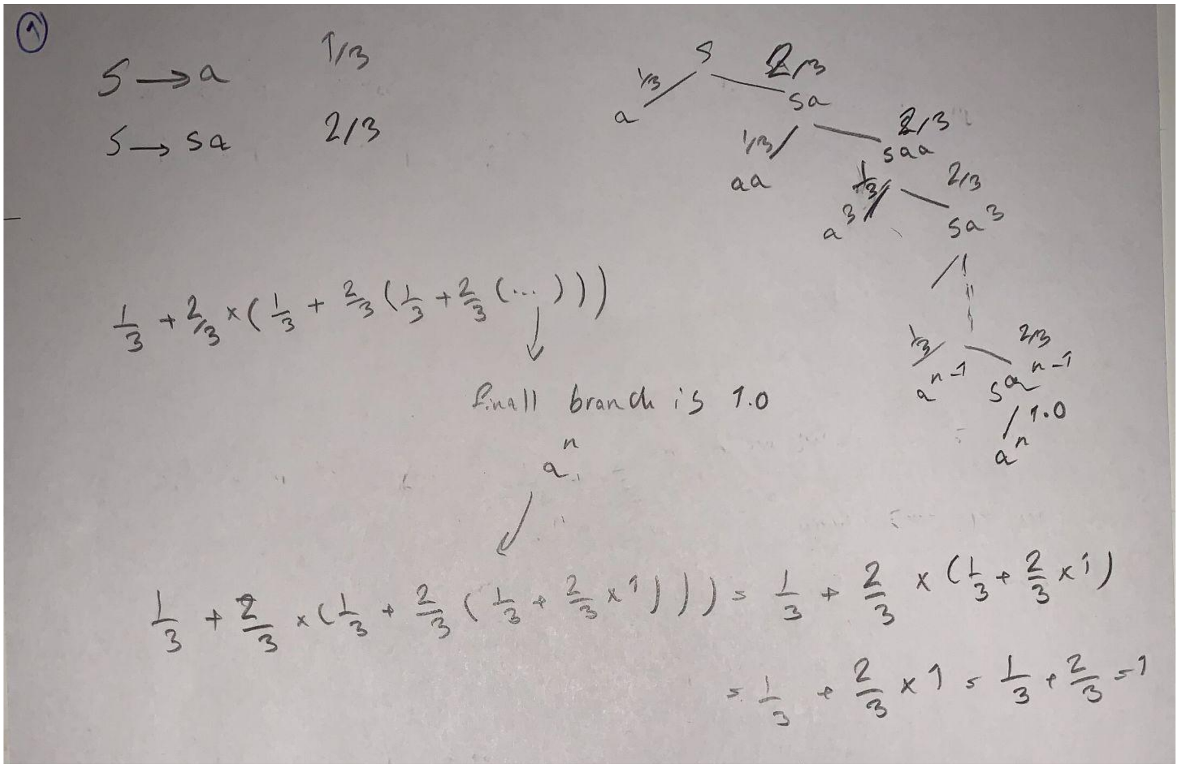
Homework 5
NLP / CS 425

Amin Hosseiny Marani

1-



2-

To show that both parsing trees have similar probabilities, we just need to compute probability of each tree.

a)

P(NP - > NP Conj NP) * P(NP->NP PP) * P(NP->Noun) * P(Noun->dogs) * P(PP->Prep NP) * P(Prep -> in) * P(NP -> noun) * P (Noun -> houses) * P (Conj -> and) * P (NP -> Noun) * P (Noun -> cats)

b)

P(NP -> NP PP) * P(NP->Noun) * P(Noun->dogs) * P(PP->Prep NP) * P(Prep -> in) * P(NP - > NP Conj NP) * P(NP -> noun) * P (Noun -> houses) * P (Conj -> and) * P (NP -> Noun) * P (Noun -> cats)

If you look at both probabilities, we see the exact same probability computations. I used similar colors to show that more easily.

3-

We know that length of the sentence or number of words is m and the CFG has N non-terminals.

The CYK algorithm is as follows (I removed unnecessary details):

1. For j <- from 1 to m (length of words = m) do
    a. For all {A | A -> words[j]}
        i. Table [j-1,j] <- table[j-1,j] U A
    b. For i <- from j-2 down to 0 do
        i. For k <- i + 1 to j-1 do
            1. For all {A | A -> BC}
                a. Table[i,j] <- table[i,j] U A

To compute time complexity for CYK algorithm we have to compute time complexity for each for.

1. For j ~ O(m) [because it loops over m words)
    a. For all {A | A -> words[j] } ~ O(1) [because it has 1 or multiple mapping. Since it depends on the CFG we could consider O(1) or O(K), however this won't affect the final result]
    b. For i ~ O(m) [because it has m-2 operations as it starts from j-2)
        i. For k ~ O(m) [it has at most m-1 operations when i = 0 and j = m]
            1. For all {A|A -> BC} ~ O(N) [since we have at most N non-terminals}

Finally time complexity would be the summation of for at loop 'a' and all loops at loop 'b' multiplied by 'm' as the main loop. CYK = O(m*1 + m*m*m*N) = O($m + m^3N) = O(m^3N)$

*Even if we consider loop 'a' as O(K) the final result would be the same.

Amin Hosseiny Marani

4-

A)

$$\alpha_j(p,q) = \sum_{f,g \neq j} \sum_{e=q+1}^{m} \alpha_f(p,e)\, \beta_g(q+1,e)\, P(N^f \to N^j N^g)$$

$$+ \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q)\, \beta_g(e,p-1)\, P(N^f \to N^g N^j)$$

Page 3, lecture 18: $\alpha_1(1,m) = 1 \Longrightarrow Pr(\phi \mid S \to w_{1m}) = 1$

$\alpha_j(1,m) = 0$

(T)

this would give a top-right start

$\alpha_S = 1$

to make this easier to follow, let's consider m=5. we can later generalize this to any number m.

$\alpha_1(1,5) = 1$

$$\alpha_j(2,5) = \sum_f \sum_{e=1}^{1} \boxed{\alpha_f(e,5)}\, \beta_g(1,e)\, P(N^f \to N^g N^j)$$

$$+ \sum_f \sum_{e=6}^{5} \alpha_f(2,e)\, \beta(6,e)\, P(N^f \to N^j N^g) \to 0$$

$$= \boxed{\alpha_f(1,5)}\, \beta_g(1,1)\, P(N^f \to N^g N^j)$$

$$\alpha_j(1,4) = \sum_{f,j} \sum_{e=5}^{5} \alpha(1,e)\, \beta(5,e)\, P(N^f \to N^j N^g)$$

$$+ \sum_{f,j} \sum_{e=1}^{0} \to 0 = \alpha(1,5)\, \beta(5,1)\, P(N^f \to N^j N^g)$$

Amin Hosseiny Marani

4) cont'd

Now we can do the same for $\alpha_j(3,5)$, $\alpha_j(2,4)$, and $\alpha_j(1,3)$, since they are dependant on $\alpha_1(1,5)$, $\alpha(2,5)$, and $\alpha(1,4)$:

$$\alpha_j(3,5) = \sum_{f,g}\sum_{6} \to 0 + \sum_{f,g}\sum_{e=1}^{2} \alpha(e,5)\,\beta(e,2)\,P(N^f \to N^g N^j)$$

$$= \alpha(1,5)\,\beta(1,2)\,P + \alpha(2,5)\,\beta(2,2)\,P$$

$$\alpha_j(2,4) = \sum_{f,g}\sum_{e=5}^{5} \alpha(2,e)\,\beta(5,e)\,P \quad [\alpha(2,5),\ \beta(5,5)]$$

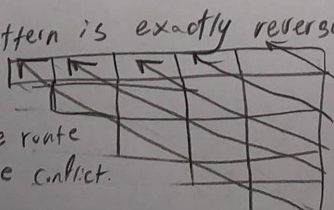
$$+ \sum_{f,g}\sum_{e=1}^{1} \alpha(e,4)\,\beta(e,1)\,P \quad [\alpha(1,4),\ \beta(1,1)]$$

$$\alpha_j(1,3) = \sum_{f,g}\sum_{e=4}^{5} \alpha(1,e)\,\beta(4,e)\,P$$

$$+ \sum_{f,g}\sum_{e=1}^{0} \to 0 \quad = \alpha(1,4)\,\beta(4,4)\,P + \alpha(1,5)\,\beta(4,5)\,P$$

Similarly using these new αs, we can compute: $\alpha_j(4,5)$, $\alpha_j(3,4)$, $\alpha(2,3)$, and $\alpha(1,2)$

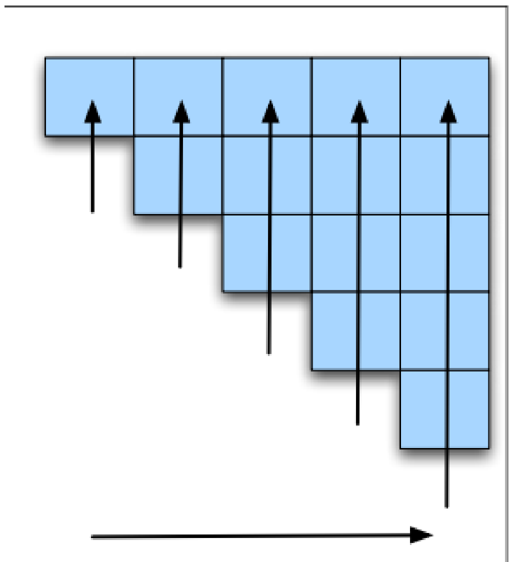
the pattern is exactly reversed of inside probability.



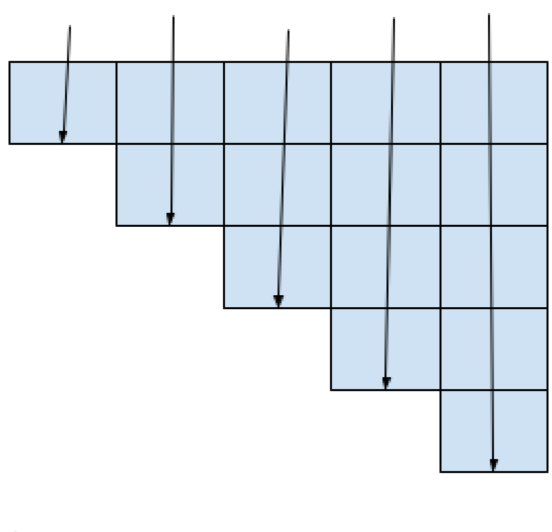
without loss of generality, it can be applied to any sentence with length m.

* any other route would cause conflict.

I just realized there is another way of solving this. In some resources (online resources and books) CYK filling using both inside and outside probability is running diagonally as I just described. But in some like the lecture 16, it happens from bottom to top and left to right, similar to the figure below:



In this case, filling CYK using outside probability would also reverse the filling. In order to fill each column we start from the top (cell[0,m] = 1) and each cell only needs the previous one at the same columns and previous cells and previous columns. For example to compute $\alpha(2,5)$ we only need $\alpha(1,5)$, and to compute $\alpha(3,5)$ we need $\alpha(2,5)$and so on. So the final direction would be:



5-

Question: In the algorithm that needs the most likely parse tree for a sentence, explain where and what information needs to be stored to reconstruct the tree.

Answer:

In order to reconstruct the optimal (most likely) tree we need to store argmax of each tag (or non-terminal CFG as considered in the equation below). So for each pair of (p,q) we need to store argmax of all CFG non-terminals. However most of them are zero. So the matrix would be $m^2 * N/2$.

“m” is the size of the sentence and N is the size of CFG grammar. I divided the size by 2, because we only need the upper square.

For each [i,p,q] triple we have to store three values, j,k, and r in which j and k are two of N CF grammars and r is a number between p and q. This make the final size $3 * m^2 * N/2$which is of $O(m^2N)$ space complexity.

$$\psi_i(p,q) = \arg\max_{(j,k,r)} P(N^i \to N^j \ N^k)\delta_j(p,r)\delta_k(r+1,q)$$

To re-construct the tree we start from the root which is N^1_{1M}. Considering current state as N^i_{pq} the left child would be N^j_{pr} and right child would be $N^k_{(r+1)q}$ and we select each using packpointers for each i,p,q and corresponding arguments j,k,r we stored earlier.