CS542 (Fall 2023) Written Assignment 2
Sequence Labeling

Eric Martin

Due October 6, 2023

1 Hidden Markov Models

(You may find the discussion in Chapter A of the Jurafsky and Martin book helpful.)

You are given the following short sentences, tagged with parts of speech:

```
Alice/NN admired/VB Dorothy/NN
Dorothy/NN admired/VB every/DT dwarf/NN
Dorothy/NN cheered/VB
every/DT dwarf/NN cheered/VB
```

1. Train a hidden Markov model on the above data. Specifically, compute the initial probability distribution π:

   > **NOTE:** *I was unsure how to show the math in the LaTeX format so I have done my best at explaining what it is I have done and have some scratch paper if needed as well.*

   ### Initial Probability Distribution

   The initial probability of a word type y_1 being at the start of a sentence, denoted $\pi(y_1)$, is computed as follows:

   $$\pi(y_1) = \frac{\text{count}(y_1)}{\text{Total count}} \quad \text{where } y_1 \text{ is the starting word.}$$

   The initial y_1 count was NN = 3, VB = 0, and DT = 1, but due to add-1 smoothing we add 1 to each count, then divide by the total count which is 7. After doing so, we get the following initial probability distribution tables (one in fractional form and the other in decimal form).

y_1	NN	VB	DT
$P(y_1)$	4/7	1/7	2/7

Or

y_1	NN	VB	DT
$P(y_1)$	0.571	0.143	0.286

Transition Probability

The transition probability from a prior word y_{i-1} to the next word y_i is calculated using:

$$P(y_i|y_{i-1}) = \frac{\text{count}(y_i|y_{i-1})}{\text{sum(corresponding row)}}$$

The transition matrix **A**:

$P(y_i\|y_{i-1})$		y_i		
		NN	VB	DT
y_{i-1}	NN	1/7	5/7	1/7
	VB	2/5	1/5	2/5
	DT	3/5	1/5	1/5

or

$P(y_i\|y_{i-1})$		y_i		
		NN	VB	DT
y_{i-1}	NN	0.143	0.714	0.143
	VB	0.400	0.200	0.400
	DT	0.600	0.200	0.200

Emission Probability

The emission probability of a word x_i given a type y_i was calculated as follows:

$$P(x_i|y_i) = \frac{\text{count}(x_i|y_i)}{\text{sum(corresponding column)}}$$

Emission matrix **B** is seen below. Note that I have added an unknown token and then performed the add-1 smoothing. I have mixed feelings about doing it this way but that is how I performed this emission table. Also, note that I did not simplify my fractions to help the legibility of the table (i.e you can notice that the columns all have the same denominator. But I do understand that this can be annoying to see 2/10, etc.).

I have also highlighted the largest values in each row.

$P(x_i \mid y_i)$		y_i		
		NN	VB	DT
x_i	Alice	**2/14**	1/12	1/10
	admired	1/14	**3/12**	1/10
	Dorothy	**4/14**	1/12	1/10
	every	1/14	1/12	**3/10**
	dwarf	**3/14**	1/12	1/10
	cheered	1/14	**3/12**	1/10
	<UNK>	2/14	2/12	**2/10**

$P(x_i \mid y_i)$		y_i		
		NN	VB	DT
x_i	Alice	**0.143**	0.083	0.100
	admired	0.071	**0.250**	0.100
	Dorothy	**0.286**	0.083	0.100
	every	0.071	0.083	**0.300**
	dwarf	**0.214**	0.083	0.100
	cheered	0.071	**0.250**	0.100
	<UNK>	0.143	0.167	**0.200**

Note that you should account for the unknown word <UNK>, but you don't need to account for the start symbol <S> or the stop symbol </S>. There are ways to train the probabilities of <UNK> from the training set, but for this assignment, you can simply let count(<UNK>, y) = 1 *for all tags y (before smoothing). You should use add-1 smoothing on all three tables.*

2. Use the forward algorithm to compute the probability of the following sentence:

   `Alice cheered`

   As part of your answer, you should fill in the forward trellis below:

Forward Trellis

For the trellis, I wasn't able to think of a good way to depict the actual trellis (lies, I actually made a depiction of the Viterbi trellis but didn't go back and make this one) but I understand what is going on. Below I wrote out the exact steps for how we consider all the possible ways of getting to the next states by coming from all possible prior states and how we determine the probability of each using the inital, transition, and emission probabilities. Due to how small the probabilities were getting, I also upped the precision from 3 to 5 decimal places.

	`Alice`	`cheered`
`NN`	$\pi(\text{NN}) \times B(\text{Alice}\|\text{NN})$	$forward(\text{NN, Alice}) \times A(\text{NN}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$ $+forward(\text{VB, Alice}) \times A(\text{VB}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$ $+forward(\text{DT, Alice}) \times A(\text{DT}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$
`VB`	$\pi(\text{VB}) \times B(\text{Alice}\|\text{VB})$	$forward(\text{NN, Alice}) \times A(\text{NN}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$ $+forward(\text{VB, Alice}) \times A(\text{VB}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$ $+forward(\text{DT, Alice}) \times A(\text{DT}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$
`DT`	$\pi(\text{DT}) \times B(\text{Alice}\|\text{DT})$	$forward(\text{NN, Alice}) \times A(\text{NN}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$ $+forward(\text{VB, Alice}) \times A(\text{VB}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$ $+forward(\text{DT, Alice}) \times A(\text{DT}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$

	`Alice`	`cheered`
`NN`	4/7 * 1/7 = 4/49	(4/49 * 1/7 * 1/14) + (1/84 * 2/5 * 1/14) + (1/35 * 3/5 * 1/14)
`VB`	1/7 * 1/12 = 1/84	(4/49 * 5/7 * 3/12) + (1/84 * 1/5 * 3/12) + (1/35 * 1/5 * 3/12)
`DT`	2/7 * 1/10 = 1/35	(4/49 * 1/7 *1/10) + (1/84 * 2/5 * 1/10) + (1/35 * 1/5 * 1/10)

	`Alice`	`cheered`
`NN`	0.08163	0.00240
`VB`	0.01190	0.01660
`DT`	0.02857	0.00221

Total Probability of Sentence Occurance

The total probability of the sentence "Alice cheered" is found by taking the sum of the probabilities in the 'cheered' column, **0.02121** since that column contains the prior probabilites from the 'Alice' column.

3. Use the Viterbi algorithm to compute the best tag sequence for the following sentence:

   `Goldilocks cheered`

   As part of your answer, you should fill in the Viterbi trellis below. You should also keep track of backpointers, either using arrows or in a separate table.

	`Goldilocks`	`cheered`
`NN`	$\pi(\text{NN}) \times B(\texttt{<UNK>}\|\text{NN})$	$forward(\text{NN}, \texttt{<UNK>}) \times A(\text{NN}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$ $forward(\text{VB}, \texttt{<UNK>}) \times A(\text{VB}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$ $forward(\text{DT}, \texttt{<UNK>}) \times A(\text{DT}\|\text{NN}) \times B(\text{cheered}\|\text{NN})$
`VB`	$\pi(\text{VB}) \times B(\texttt{<UNK>}\|\text{VB})$	$forward(\text{NN}, \texttt{<UNK>}) \times A(\text{NN}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$ $forward(\text{VB}, \texttt{<UNK>}) \times A(\text{VB}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$ $forward(\text{DT}, \texttt{<UNK>}) \times A(\text{DT}\|\text{VB}) \times B(\text{cheered}\|\text{VB})$
`DT`	$\pi(\text{DT}) \times B(\texttt{<UNK>}\|\text{DT})$	$forward(\text{NN}, \texttt{<UNK>}) \times A(\text{NN}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$ $forward(\text{VB}, \texttt{<UNK>}) \times A(\text{VB}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$ $forward(\text{DT}, \texttt{<UNK>}) \times A(\text{DT}\|\text{DT}) \times B(\text{cheered}\|\text{DT})$

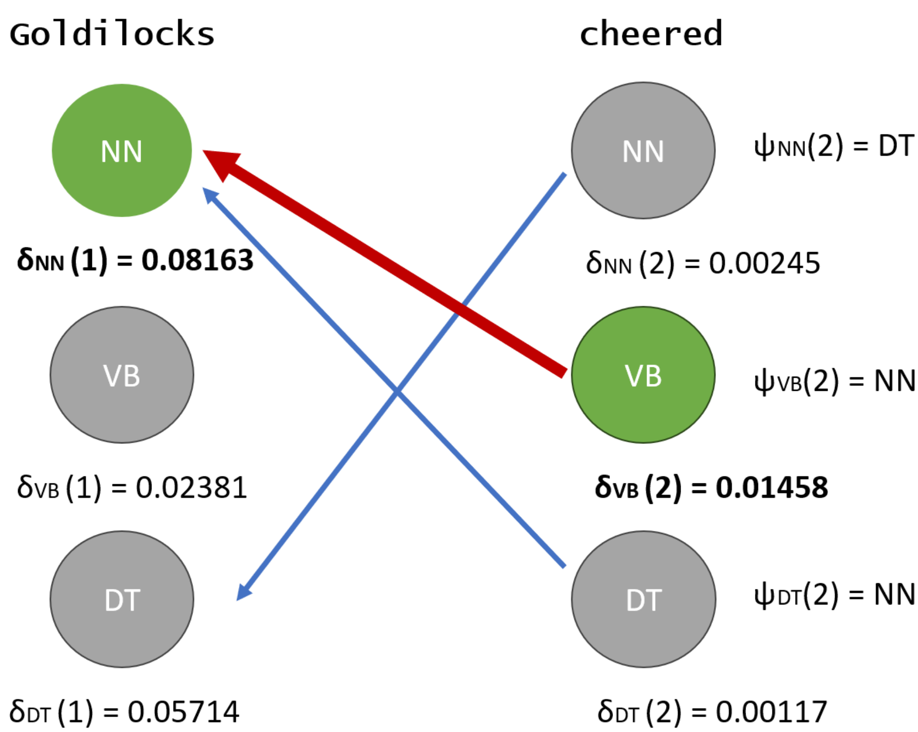
	`Goldilocks`	`cheered`
`NN`	**4/7 * 1/7 = 4/49**	From NN: (4/49 * 1/7 * 1/14) = 0.00083 From VB: (1/42 * 2/5 * 1/14) = 0.00068 **From DT: (2/35 * 3/5 * 1/14) = 0.00245**
`VB`	1/7 * 1/6 = 1/42	**From NN: (4/49 * 5/7 * 3/12) = 0.01458** From VB: (1/42 * 1/5 * 3/12) = 0.00119 From DT: (2/35 * 1/5 * 3/12) = 0.00286
`DT`	2/7 * 1/5 = 2/35	**From NN: (4/49 * 1/7 *1/10) = 0.00117** From VB: (1/42 * 2/5 * 1/10) = 0.00095 From DT: (2/35 * 1/5 * 1/10) = 0.00114

Procedure

The probabilities for each of the possible paths along the trellis are computed and then the max Max value from each is highlighted in bold. The Max of all of them is taken (VB from NN) and pointed backward. In this case, it seems the Viterbi algorithm predicts Goldilocks is a noun and cheered is a verb which is correct.

	Goldilocks	cheered
NN	**0.08163**	0.00245
VB	0.02381	**0.01458**
DT	0.05714	0.00117

1.1 Path Visualization



Here we can see that two of the back-pointers are pointing back to NN and one is pointing back to DT. The one with the largest value is the VB pointing back to NN which leads us to select the tags of NN for Goldilocks and VB for cheered which are correct.