CS-5340/6340, Written Assignment #2
DUE: Friday, September 25, 2020 by 11:59pm

Submit your assignment on CANVAS in pdf format.

1. (12 pts) The table below contains frequency values for a set of nouns referring to trees in an imaginary text corpus. Fill in the table below with the unsmoothed probability of each noun, as well as the smoothed frequency and smoothed probability of each noun using **add-k smoothing** with k=3. You should assume that the vocabulary consists only of the nouns listed below.

   IMPORTANT: Please show the fraction (numerator/denominator) used to compute each value as well as the final value (e.g., 2/4 = .50).

NOUN	FREQ	UNSMOOTHED PROB	SMOOTHED FREQ	SMOOTHED PROB
pine	300			
oak	96			
spruce	4			
cottonwood	0			

2. (28 pts) Assume that a part-of-speech (POS) tagger has been applied to the sentence below with the following results:

> **Bob**/NOUN **put**/VERB **the**/ART **blue**/ADJ **light**/NOUN **bulb**/NOUN **in**/PREP **the**/ART **light**/ADJ **orange**/ADJ **lamp**/NOUN **to**/INF **light**/VERB **the**/ART **blue**/ADJ **and**/CONJ **orange**/ADJ **room**/NOUN **with**/PREP **blue**/ADJ **light**/NOUN **!**/PUNC

Fill in the table below with the probabilities that you would estimate based on the sentence above. **Leave your results in fractional form (e.g., 5/5)!** If a probability would be undefined (i.e., have a zero denominator), then answer UNDEFINED.

We define the various types of probabilities as follows, where w_i indicates a word and t_i indicates a POS tag.

- $P(w_i)$ means probability of word w_i
- $P(w_i\ w_j)$ means probability of the bigram $w_i\ w_j$. Do not use ϕ for this computation.
- $P(t_i)$ means probability of POS tag t_i
- $P(t_i\ t_j)$ means probability of the bigram $t_i\ t_j$. Do not use ϕ for this computation.
- $P(w_i \mid w_{i-1})$ means probability of word w_i following word w_{i-1}
- $P(w_i \mid w_{i-2}\ w_{i-1})$ means probability of word w_i following words $w_{i-2}\ w_{i-1}$
- $P(t_i \mid t_{i-1})$ means probability of POS tag t_i following POS tag t_{i-1}
- $P(t_i \mid t_{i-2}\ t_{i-1})$ means probability of word t_i following words $t_{i-2}\ t_{i-1}$
- $P(w_i \mid t_i)$ means probability of word w_i given tag t_i.
- $P(t_i \mid w_i)$ means probability of word t_i given tag w_i.

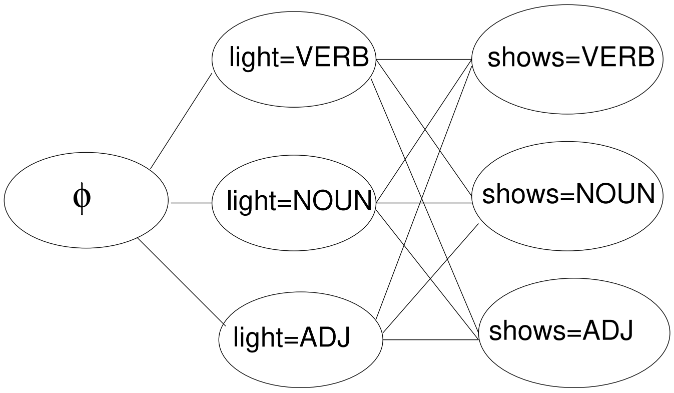
Probability	**Value**
P(light)	
P(blue light)	
P(ADJ)	
P(ADJ NOUN)	
P(light | blue)	
P(in | the)	
P(light | the blue)	
P(NOUN | VERB)	
P(CONJ | ADJ)	
P(ADJ | VERB ART)	
P(orange | ADJ)	
P(light | VERB)	
P(NOUN | light)	
P(VERB | put)	

3. (30 pts) Use the following tables of probabilities to answer this question. Note that these numbers are completely fictional and do not necessarily add up logically (i.e., sum to 1 where they should), but don't worry about that, just use them as they are.

P(light | NOUN)	.20
P(light | VERB)	.60
P(light | ADJ)	.35
P(shows | NOUN)	.10
P(shows | VERB)	.45
P(shows | ADJ)	.07

P(NOUN | ϕ)	.50
P(VERB | ϕ)	.35
P(ADJ | ϕ)	.25
P(NOUN | NOUN)	.40
P(NOUN | VERB)	.15
P(NOUN | ADJ)	.70
P(VERB | NOUN)	.60
P(VERB | VERB)	.05
P(VERB | ADJ)	.01
P(ADJ | NOUN)	.03
P(ADJ | VERB)	.20
P(ADJ | ADJ)	.08

Assume that there are only 3 possible part-of-speech tags: NOUN, VERB, and ADJ. The following network would be used by the Viterbi algorithm to find the most likely sequence of POS tags for the sentence *"Light shows"*:



Using the Viterbi algorithm, compute the probability for each of the following nodes in the network. Show all your work!

- P(light=VERB)
- P(light=NOUN)
- P(light=ADJ)
- P(shows=VERB)
- P(shows=NOUN)
- P(shows=ADJ)

Compute the following forward probabilities. Show all your work!

- $\alpha_{shows}(NOUN)$
- $\alpha_{shows}(VERB)$
- $\alpha_{shows}(ADJ)$

Compute the following normalized probability values. Show all your work!

- P(shows/NOUN | light)
- P(shows/VERB | light)
- P(shows/ADJ | light)

4. (12 pts) Consider the following quote from Shakespeare, with assigned part-of-speech (POS) tags:

   Brevity/NOUN **is**/VERB **the**/ART **soul**/NOUN **of**/PREP **wit**/NOUN

   Show the equation that you would use for statistical POS tagging to compute P(NOUN VERB ART NOUN PREP NOUN | **Brevity is the soul of wit**) for each of the N-gram models listed below. You do not need to include any numbers at all. Just show the equations that you would use with the specific words and POS tags for this quote plugged into each equation.

   (a) Unigram model

   (b) Bigram model

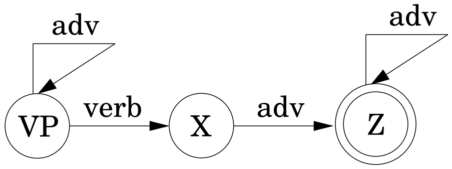
   (c) Trigram model

5. (6 pts) Use the BIO labeling scheme to identify the simple (base) noun phrases (NPs) in the sentence below. Each word should be assigned one of the labels **B** (for Beginning of a NP), **I** (for Inside a NP), or **O** (for Outside a NP).

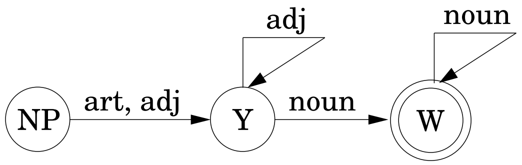
   In November Salt Lake City typically receives only a little snowfall but people often tell their children wild tales about giant blizzards

6. (12 pts) Consider the three finite-state machines (FSMs) below, which recognize sequences of part-of-speech (POS) tags. Assume that the states labeled VP and NP are the initial states for FSMs that are designed to recognize Verb Phrases and Noun Phrases, respectively, and that the states with an extra circle around them are accepting states. For (b), the first edge labeled "art, adj" can be traversed by <u>either</u> an "art" or an "adj".,

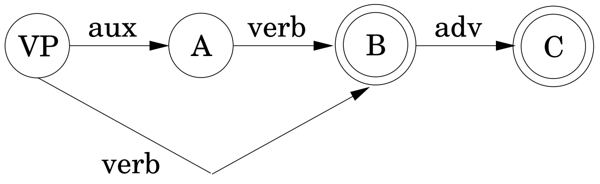
(a)



(b)



(c)



- Write a Verb Phrase (VP) grammar that accepts exactly same set of POS tag sequences as the FSM labeled (a) above.

- Write a Noun Phrase (NP) grammar that accepts exactly same set of POS tag sequences as the FSM labeled (b) above.

- Write a Verb Phrase (VP) grammar that accepts exactly same set of POS tag sequences as the FSM labeled (c) above.

Question #7 is for CS-6340 students ONLY!

7. (12 pts) Consider the following four context-free grammars to recognize Noun Phrases (NPs):

G1	**G2**	**G3**	**G4**
NP → art NP1	NP → art X	NP → NP7	NP → art W
NP → NP1	NP → adj X	NP → art NP6	NP → W
NP1 → adj NP1	NP → Y	NP → adj NP6	W → adj noun
NP1 → NP2	X → adj X	NP → art adj NP6	W → adj W
NP2 → noun	X → Y	NP6 → NP7	W → Z
NP2 → noun NP2	Y → noun	NP7 → noun NP7	Z → noun Z
	Y → noun noun	NP7 → noun	Z → noun
	Y → noun Y		

For each grammar, write a regular expression that accepts exactly the same NP language as the grammar. That is, the regular expression should recognize exactly the same set of part-of-speech tag sequences as the grammar.

You can use the Kleene star (*) operator, which means 0 or more instances, as well as the + operator, which means 1 or more instances. For example, *verb** means a sequence of ≥ 0 verbs, and *verb*$^+$ means a sequence of ≥ 1 verbs. You can also use ϵ to represent the empty string, if you wish.

- G1

- G2

- G3

- G4