CS-5340/6340, Written Assignment #2
DUE: Wednesday, September 22, 2021 by 11:59pm

Submit your assignment on CANVAS in pdf format.

1. (12 pts) Consider the 9 training documents below, where each document consists of one sentence from (imaginary) restaurant reviews.

Class	**Document**
NEG	terrible food and slow service
NEG	so expensive and incredibly bad service
NEG	ok food but too expensive
NEG	service was so slow
POS	incredibly good food
POS	so tasty
POS	great food and not too expensive
POS	service a bit slow but incredibly great food
POS	incredibly tasty and ok service

(a) Based on the training corpus above, compute the following probabilities. Do not perform smoothing. **Leave your answers in fractional form!**

- $P(NEG) = \frac{4}{9}$
- $P(POS) = \frac{5}{9}$
- $P(\text{“}incredibly'' \mid NEG) = \frac{1}{5+6+5+4} = \frac{1}{20}$
- $P(\text{“}incredibly'' \mid POS) = \frac{3}{3+2+6+8+5} = \frac{3}{24}$
- $P(\text{“}service'' \mid NEG) = \frac{3}{20}$
- $P(\text{“}service'' \mid POS) = \frac{2}{24}$

(b) Using the Naive Bayes classification algorithm, determine which Class (NEG or POS) would be assigned to the test document: “incredibly expensive but ok”. **Show ALL of your work! You will get zero credit if you only give a class name without showing the work.**

First, calculate the probabilities that were not covered in section (a):

- $P(\text{“}expensive''|NEG) = \frac{2}{20}$
- $P(\text{“}expensive''|POS) = \frac{1}{24}$
- $P(\text{“}but''|NEG) = \frac{1}{20}$
- $P(\text{“}but''|POS) = \frac{1}{24}$
- $P(\text{“}ok''|NEG) = \frac{1}{20}$
- $P(\text{“}ok''|POS) = \frac{1}{24}$

Probability of the sentence being of class NEG:

$$
\begin{aligned}
P(NEG|S) &= P(NEG)P(S|NEG) \\
&= \frac{4}{9} \times \frac{1 \times 2 \times 1 \times 1}{20^4} \\
&= \frac{4}{9} \times \frac{2}{20^4} \\
&= \frac{4}{9} \times \frac{2}{2^4 \times 10^4} \\
&= \frac{4}{9} \times \frac{1}{8 \times 10^4} \\
&= \frac{1}{18} \times 10^{-4} \\
&\approx 5.56 \times 10^{-6}
\end{aligned}
$$

Probability of the sentence being of class POS:

$$
\begin{aligned}
P(POS|S) &= P(POS)P(S|POS) \\
&= \frac{5}{9} \times \frac{3 \times 1 \times 1 \times 1}{24^4} \\
&= \frac{5}{9} \times \frac{3}{24^4} \\
&= \frac{5}{3} \times \frac{1}{2.4^4 \times 10^4} \\
&\approx 5.02 \times 10^{-6}
\end{aligned}
$$

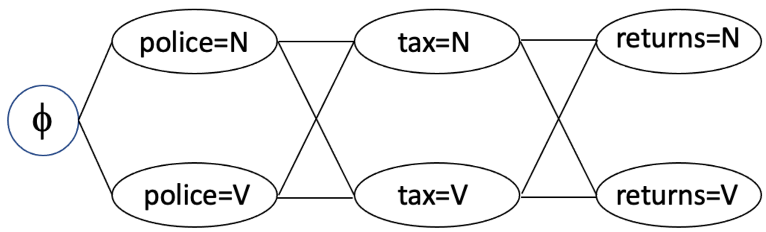
Based on the above probabilities, the sentence would be assigned the class NEG.

2. (26 pts) Use the following tables of probabilities to answer this question. Note that these numbers are 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. There are only 2 possible part-of-speech tags: N (noun) and V (verb).

P(police | N)	.60
P(police | V)	.20
P(tax | N)	.50
P(tax | V)	.40
P(returns | N)	.30
P(returns | V)	. 10

P(N | ϕ)	.85
P(V | ϕ)	.15
P(N | N)	.65
P(N | V)	.55
P(V | N)	.45
P(V | V)	.25

The following network would be used by the Viterbi algorithm to find the most likely sequence of POS tags for the sentence *"Police tax returns"*:



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

- P(police=N) = $P(police \mid N)P(N \mid \phi) = 0.60 \times 0.85 = 0.51$
- P(police=V) = $P(police \mid V)P(V \mid \phi) = 0.20 \times 0.15 = 0.03$
- $$\begin{aligned} P(tax = N) &= P(tax \mid N) \max \begin{cases} P(N \mid N)P(police = N) \\ P(N \mid V)P(police = V) \end{cases} \\ &= 0.5 \max \begin{cases} 0.65 \times 0.51 \\ 0.55 \times 0.03 \end{cases} \\ &= 0.5 \times 0.65 \times 0.51 \\ &= 0.16575 \end{aligned}$$
- $$\begin{aligned} P(tax = V) &= P(tax \mid V) \max \begin{cases} P(V \mid N)P(police = N) \\ P(V \mid V)P(police = V) \end{cases} \\ &= 0.4 \max \begin{cases} 0.45 \times 0.51 \\ 0.25 \times 0.03 \end{cases} \\ &= 0.4 \times 0.45 \times 0.51 \\ &= 0.0918 \end{aligned}$$
- $$\begin{aligned} P(returns = N) &= P(returns \mid N) \max \begin{cases} P(N \mid N)P(tax = N) \\ P(N \mid V)P(tax = V) \end{cases} \\ &= 0.3 \max \begin{cases} 0.65 \times 0.16575 \\ 0.55 \times 0.0918 \end{cases} \\ &= 0.3 \times 0.65 \times 0.16575 \\ &= 0.03232125 \end{aligned}$$
- $$\begin{aligned} P(returns = V) &= P(returns \mid V) \max \begin{cases} P(V \mid N)P(tax = N) \\ P(V \mid V)P(tax = V) \end{cases} \\ &= 0.1 \max \begin{cases} 0.45 \times 0.16575 \\ 0.25 \times 0.0918 \end{cases} \\ &= 0.3 \times 0.45 \times 0.16575 \\ &= 0.02237625 \end{aligned}$$

(b) Compute the following forward probabilities. Show all your work!

First, calculate the forward probabilities for "police":

- $\alpha_{police}(N) = P(police \mid N)P(N \mid \phi) = 0.60 \times 0.85 = 0.51$
- $\alpha_{police}(V) = P(police \mid V)P(V \mid \phi) = 0.20 \times 0.15 = 0.03$

Then, we can use it for "tax":

- $$\begin{aligned}\alpha_{tax}(N) &= P(tax \mid N)(\alpha_{police}(N) + \alpha_{police}(V)) \\ &= 0.5(0.51 + 0.03) \\ &= 0.27\end{aligned} \tag{1}$$

- $$\begin{aligned}\alpha_{tax}(V) &= P(tax \mid V)(\alpha_{police}(N) + \alpha_{police}(V)) \\ &= 0.4(0.51 + 0.03) \\ &= 0.216\end{aligned} \tag{2}$$

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

- P(tax/N | police) = $\frac{\alpha_{tax}(N)}{\alpha_{tax}(N)+\alpha_{tax}(V)} = \frac{0.27}{0.27+0.216} = 0.56$
- P(tax/V | police) = $\frac{\alpha_{tax}(V)}{\alpha_{tax}(N)+\alpha_{tax}(V)} = \frac{0.216}{0.27+0.216} = 0.44$

3. (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 **light**/ADJ **blue**/ADJ **light**/NOUN **bulb**/NOUN **in**/PREP **the**/ART **blue**/ADJ **light**/NOUN **fixture**/NOUN **to**/INF **light**/VERB **the**/ART **orange**/ADJ **and**/CONJ **blue**/ADJ **room**/NOUN **with**/PREP **brilliant**/ADJ **light**/ADJ **blue**/ADJ **light**/NOUN **!**/PUNC

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 as a part of any bigrams.
- $P(t_i)$ 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 as a part of any bigrams.
- $P(w_i \mid w_{i-1})$ 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 tag t_i given word w_i.

Fill in the table below with the probabilities that you would estimate based on the tagged 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.

Probability	**Value**
P(blue)	4/25
P(to light)	1/24
P(ADJ)	8/25
P(NOUN NOUN)	2/24
P(blue | light)	2/6
P(light | to)	1/1
P(bulb | blue light)	1/3
P(VERB | NOUN)	1/7
P(ADJ | ADJ)	3/8
P(NOUN | ART ADJ)	1/3
P(blue | ADJ)	4/8
P(light | NOUN)	3/7
P(NOUN | Bob)	1/1
P(VERB | light)	1/6

4. (16 pts) Consider the following four context-free grammars that recognize Noun Phrases (NPs). Assume that you want to find grammar derivations beginning with the non-terminal "NP".

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

For each pair of grammars below, give an example of a sequence of part-of-speech (POS) tags that would be recognized as a legal NP by the first grammar but not the second grammar. If there is no such tag sequence then answer *No Sequence*.

(a) list a tag sequence recognized by G1 but not G2 — noun

(b) list a tag sequence recognized by G2 but not G1 — No Sequence

(c) list a tag sequence recognized by G1 but not G3 — adj adj noun

(d) list a tag sequence recognized by G3 but not G1 — No Sequence

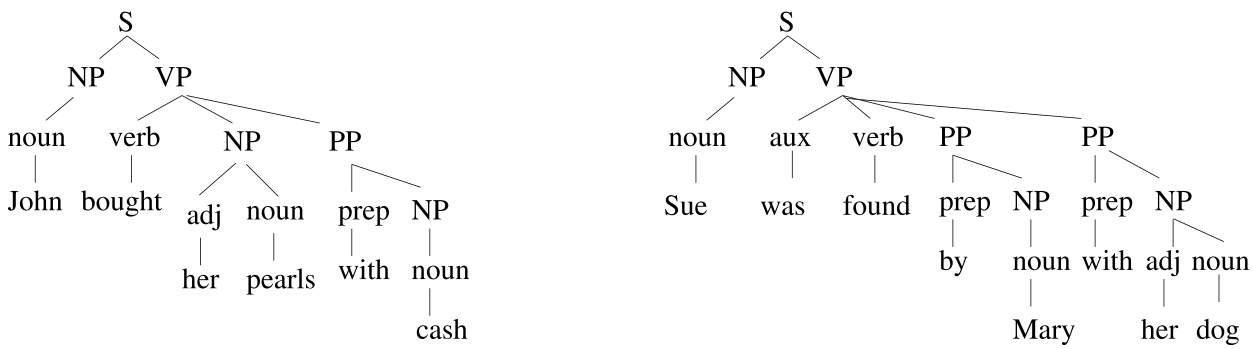
(e) list a tag sequence recognized by G1 but not G4 — No Sequence

(f) list a tag sequence recognized by G4 but not G1 — No Sequence

(g) list a tag sequence recognized by G2 but not G3 — adj adj noun

(h) list a tag sequence recognized by G3 but not G2 — noun

5. (12 pts) Pretend that the parse trees below are a (tiny!) parsed text corpus. Fill in the table below with all of the context-free grammar rules that appear in these parse trees and the total frequency count of each rule. For example, if the same grammar rule appears twice in one parse tree and once in the other parse tree, then its frequency count should be 3.



Grammar Rule	**Frequency**
S $\rightarrow$ NP VP	2
NP $\rightarrow$ noun	4
NP $\rightarrow$ adj noun	2
VP $\rightarrow$ verb NP PP	1
VP $\rightarrow$ aux verb PP PP	1
PP $\rightarrow$ prep NP	3

6. (6 pts) Consider the grammar below:

Grammar Rules	
S $\rightarrow$ NP VP	noun $\rightarrow$ *NLP*
NP $\rightarrow$ noun	verb $\rightarrow$ *is*
VP $\rightarrow$ verb	adj $\rightarrow$ *cool*
VP $\rightarrow$ verb ADJP	
ADJP $\rightarrow$ adj	

Show a bottom-up, depth-first search derivation of the parse for the sentence *"NLP is cool"* using the grammar above. Search the grammar based on the order of the rules in the table above (i.e., apply the "VP $\rightarrow$ verb" rule before the other VP rule). Do NOT use chart parsing – just show a generic search-based derivation!

NLP is cool
noun *is cool*
NP *is cool*
NP verb *cool*
NP VP *cool*
S *cool* — Dead end!
NP verb adj
NP verb ADJP
NP VP
S

<u>Question #7 is for CS-6340 students ONLY!</u>

7. (24 pts) Consider the grammar G below:

Grammar	
S $\rightarrow$ NP VP	NOUN $\rightarrow$ John
S $\rightarrow$ VP NP	NOUN $\rightarrow$ can
NP $\rightarrow$ NOUN	NOUN $\rightarrow$ well
VP $\rightarrow$ MOD VP1	VERB $\rightarrow$ swim
VP1 $\rightarrow$ VERB	MOD $\rightarrow$ can
VP1 $\rightarrow$ VERB ADV	ADV $\rightarrow$ well

List all of the entries that would be put on the chart by the *Earley chart parsing algorithm* when parsing the sentence **"John can swim well"**. Each chart entry should be a constituent or a rule, with a start and end position indicating which words have been matched by the constituent or rule. Assume that **"John"** is in position 1 and **"well"** is in position 4.

To get you started, the list below contains the chart entries for all of the part-of-speech tag constituents. Your job is to complete this list by adding the rest of the constituents and rules that would be added to the chart during parsing.

When a rule is added to the chart, use an asterisk (*) to separate the components of the rule that have been matched from the ones that have not yet been matched. For example, the rule:
S $\rightarrow$ * NP VP [1-1] means that nothing in this rule has been matched yet but the rule can begin matching constituents starting in position 1. The rule **S $\rightarrow$ NP * VP [1-2]** means that the NP has successfully matched words in the span [1-2] and the rule is waiting to match a VP starting in position 2.

CHART ENTRIES FOR "John can swim well"

Constituent or Rule	Start-End
NOUN("John")	[1-2]
NOUN("can")	[2-3]
MOD("can")	[2-3]
VERB("swim")	[3-4]
NOUN("well")	[4-5]
ADV("well")	[4-5]
S → * NP VP	[1-1]
S → * VP NP	[1-1]
NP → * NOUN	[1-1]
VP → * MOD VP1	[1-1]
NOUN("John")	[1-2]
NP → NOUN *	[1-2]
NP	[1-2]
S → NP * VP	[1-2]
VP → * MOD VP1	[2-2]
NOUN("can")	[2-3]
MOD("can")	[2-3]
VP → MOD * VP1	[2-3]
VP1 → * VERB	[3-3]
VP1 → * VERB ADV	[3-3]
VERB("swim")	[3-4]
VP1 → VERB *	[3-4]
VP1 → VERB * ADV	[3-4]
VP1	[3-4]
VP → MOD VP1 *	[2-4]
VP	[2-4]
S → NP VP * **(Parse!)**	[1-4]
NOUN("well")	[4-5]
ADV("well")	[4-5]
VP1 → VERB ADV *	[3-5]
VP1	[3-5]
VP → MOD VP1 *	[2-5]
VP	[2-5]
S → NP VP * **(Parse!)**	[1-5]