COMS W4705 - Natural Language Processing

Midterm Exam - Fall 2018

Instructor: Daniel Bauer

Name: sample solution

UNI: _____

Total Points: 100

Time: 75 minutes

Instructions:

Do not turn this page before the official start time!

The exam is closed book, closed notes, no electronic devices.

Read the directions for each question carefully before answering. Before handing in your exam, please initial each page.

If you have questions, raise your hand and wait for help.

Good luck!

Academic Honesty Statement:

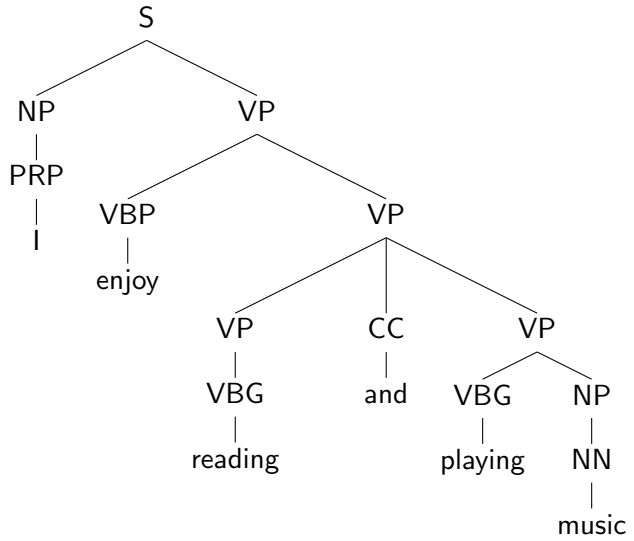
I certify that I have neither given nor received unauthorized help on this exam and that I did not use any notes, electronic devices, or other aids not specifically permitted. I understand that any violation of this policy can result in an exam grade of zero and will be reported.

Signature: _____

Question 1 - Multiple Choice and Short-Answer Questions (20 pts)

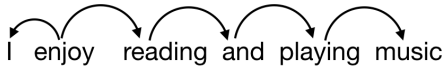
For all multiple choice questions: +1 for each correct choice, -1 for each incorrect choice. 0 points minimum.

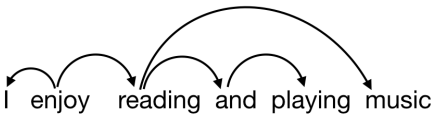
(a) Consider the following constituency tree.

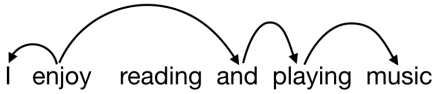


Select all dependency structures below that are *compatible* with the constituency tree.

☐ 

☒ 

☐ 

☐ 

Missing edge. Everyone got points for the last option.

(b) Which of the following statements is true about **perplexity**?

- ☒ Perplexity is used in comparing how well different language models model a corpus.
- ☒ Perplexity is a function of the probability that a language model assigns to the corpus.
- ☐ Higher values of perplexity indicate that the language model fits the corpus better.
- ☐ When a language model is applied to its training data, the perplexity is 1.

(c) Which of the following statements is true about **arc-standard transition based dependency parsing**?

- ☐ To train a transition-based dependency parser, a human annotator needs to provide transition sequences for all sentences in the training corpus.
- ☒ The arc-standard transition based system will always produce a projective dependency structure.
- ☐ Backtracking is sometimes required to find a transition sequence that results in a final state.
- ☐ In a final state, both the buffer and the stack are empty.

(d) Consider the following context free grammar $\mathcal{G}$, that is **not** in Chomsky Normal Form:

$A \rightarrow B\ C$
$C \rightarrow D$
$B \rightarrow b\ C$
$D \rightarrow d$

Which of the following CFGs are both equivalent to $\mathcal{G}$ and in Chomsky Normal Form?

☐ $A \rightarrow B\ C$
$B \rightarrow b\ d$
$C \rightarrow d$
not CNF

☒ $A \rightarrow B\ C$
$B \rightarrow E\ C$
$C \rightarrow d$
$E \rightarrow b$

☒ $A \rightarrow B\ D$
$B \rightarrow E\ D$
$E \rightarrow b$
$D \rightarrow d$

☐ $A \rightarrow B\ D$
$B \rightarrow D\ E$
$D \rightarrow d$
$E \rightarrow b$
b and d swapped

(e) A simple approach to **tokenization** is to split the input text on white-spaces and the following punctuation symbols:

```
!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~}
```

Provide a short, realistic example text that cannot be tokenized correctly by this strategy and explain why.

Many different examples were okay. The most obvious one are abbreviations like "U.S.A.", which would be split into 6 different tokens.

(f) In your own words, briefly describe what the **noisy channel model** is.

The noisy channel model assumes that an observation (for example, a sequence of words) has been generated probabilistically by some underlying hidden state sequence (for example a sequence of P.O.S tags).

"Decoding" is the task of finding the most likely hidden input sequence given the observed output using Bayesian inference.

Question 2 - Bigram Models and Smoothing (16 pts)

Assume a **bigram language model** is trained on the following corpus of sentences.

START dog bites human END
START dog bites duck END
START dog eats duck END
START duck bites human END
START human eats duck END
START human eats salad END

Assume **maximum likelihood estimates (MLE) with add-one smoothing (also called Laplace smoothing)** were used to compute bigram probabilities. What is the probability assigned to the following sentences by the bigram model? Show your work (you only need to compute the probabilities necessary to answer the problem).

- *dog eats salad*
- *human bites dog*

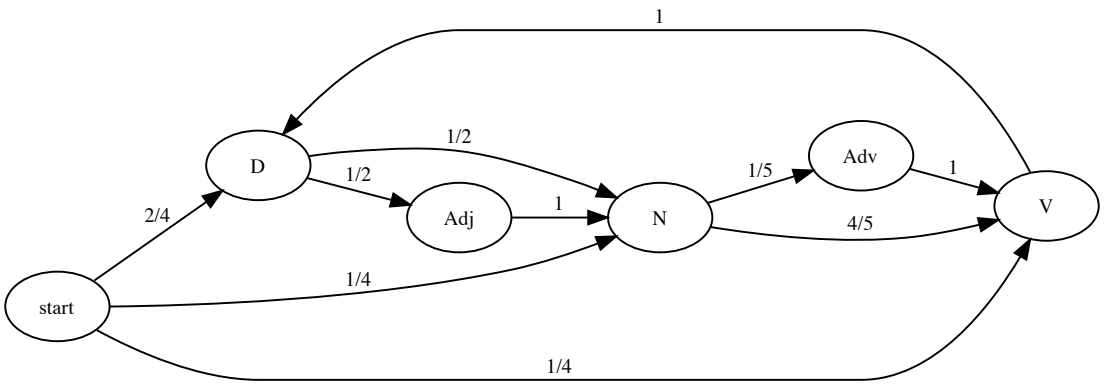
1.

P(dog | START)= (3+1)/(6+6) = 4/12 = 1/3
P(eats | dog)= (1+1)/(3+6) = 2/9
P(salad | eats)= (1+1)/(3+6) = 2/9
P(END | salad)= (1+1)/(1+6) = 2/7
P(dog eats salad) =
P(dog | START) P(eats | dog) P(salad | eats) P(END | salad) =
1/3 x 2/9 x 2/9 x 2/7 = 6 / 1701 = 0.0035

2.
P(human |START) = (2+1)/(6+6) = 3/12 = 1/4
P(bites |human) = (0+1)/(4+6) = 1/10
P(dog | bites) = (0+1)/(3+6) = 1/9
P(END | dog) = (0+1) /(3+6) = 1/9
P(human bites dog) =
P(human | START) P(bites|human) P(dog | bites) P(END|dog) =
1/4 x 1/10 x 1/9 x 1/9 = 1/3240 = 0.00031

Question 3 - Part-of-Speech Tagging with HMMs (16 pts)

Consider the following Hidden Markov Model (HMM).



Emission probabilities:
$P(the|D) = 1$
$P(saw|V) = 4/6$ $\quad P(well|V) = 2/6$
$P(saw|N) = 1/3$ $\quad P(well|N) = 2/3$
$P(well|Adj) = 1$
$P(well|Adv) = 1$

Use the **Viterbi algorithm** to compute the most likely sequence of POS tags for the phrase *"saw the well well"*.
You can use the following table.

```
pi[k,t] = max_s pi[k-1,s] x P(t|s) x P(w_k | t)
```

	2/4*0 = 0 D	1/6*1*1= 1/6 D	D	D
	1/4*4/6=1/6 V	V	V	V 1/18*4/5*2/6 = 2/135
	1/4*1/3=1/12 N	N	1/6*1/2*2/3= 1/18 N	N 1/12*1*2/3= 1/18
	Adj	Adj	1/6*1/2*1= 1/12 Adj	Adj
	Adv	Adv	Adv	Adv 1/18*1/5*1= 1/90
	saw	*the*	*well*	*well*

(question 3 continued and extra space)

The most probable POS sequence is: V D Adj N

Question 4 - PCFGs and Ambiguity (16 pts)

Consider the following PCFG.

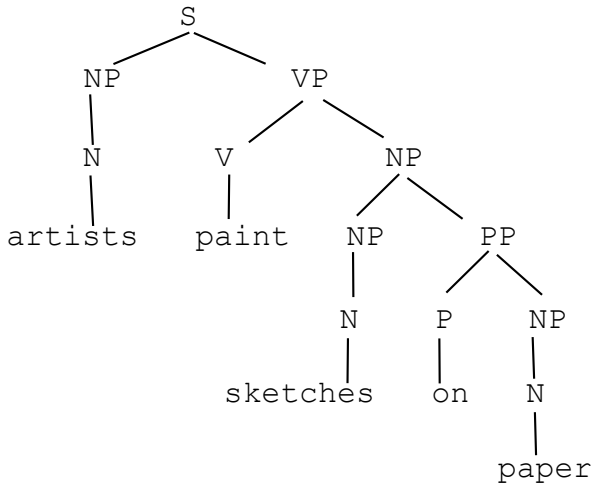
S $\rightarrow$ NP VP	$[1]$	N $\rightarrow$ artists	$[1/6]$
NP $\rightarrow$ N	$[2/4]$	N $\rightarrow$ paint	$[2/6]$
NP $\rightarrow$ N N	$[1/4]$	N $\rightarrow$ sketches	$[1/6]$
NP $\rightarrow$ NP PP	$[1/4]$	N $\rightarrow$ paper	$[2/6]$
VP $\rightarrow$ V NP	$[2/5]$	V $\rightarrow$ paint	$[1/2]$
VP $\rightarrow$ V PP	$[1/5]$	V $\rightarrow$ sketches	$[1/2]$
VP $\rightarrow$ VP PP	$[2/5]$	P $\rightarrow$ on	$[1]$
PP $\rightarrow$ P NP	$[1]$		

Now consider the following sentence: *"artists paint sketches on paper"*.

(a) Write down two different parse trees for the sentence. You do not have to perform the steps of a parsing algorithm.

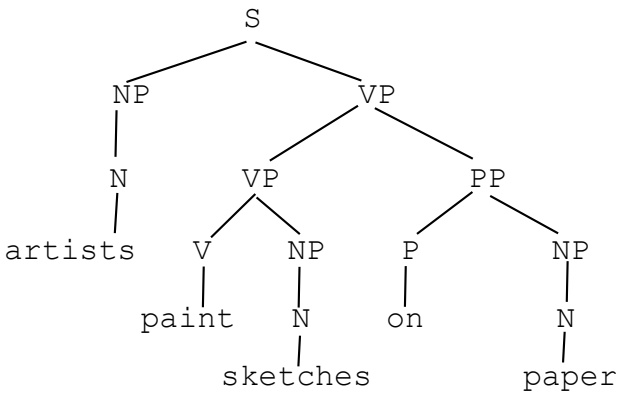
(b) Compute the probability for each parse tree.

~~(c) Compute the probability for the sentence.~~ not graded (there are three parse trees and you were asked to draw only two)

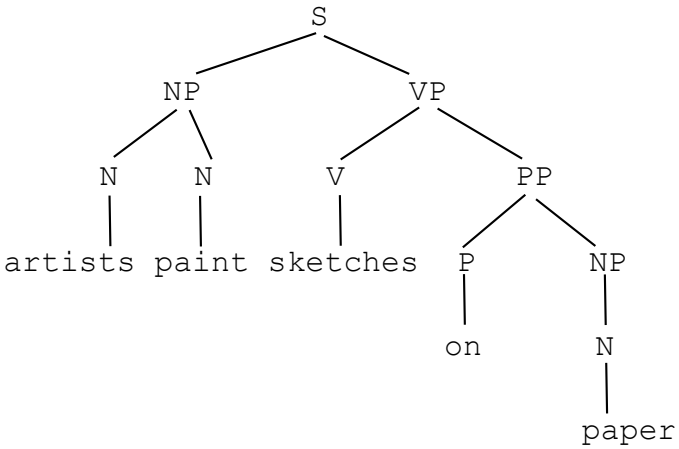



```
1 * (2/4) * (1/6) * (2/5) * (1/2) * (1/4) * (2/4) * (1/6) * 1 * 1 * (2/4) * (2/6)
= 1/17280 = 0.00005787          (simply multiply all rule probabilities)
```

(extra space for Question 4)



1 * (2/4) * (1/6) * (2/5) *(2/5) * (1/2) * (2/4) * (1/6) * 1 * 1 * (2/4) * (2/6)
= 1/10800 = 0.00009259



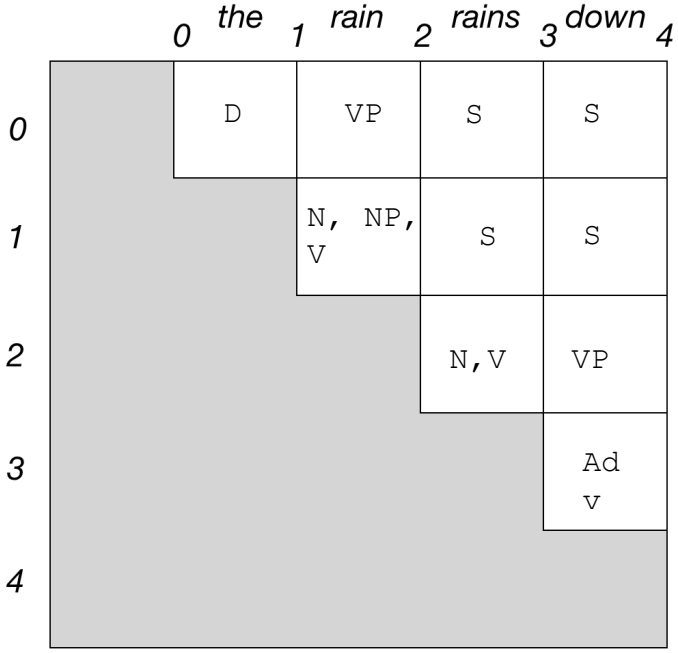
1 * (1/4) * (1/6) * (2/6) * (1/5) * (1/2) * 1 * 1 * (2/4) * (2/6)
= 1/4320 = 0.000231

Question 5 - CFG Parsing (16 pts)

Consider the following context free grammar (CFG):

S	→	NP VP	NP	→	*rain*
S	→	NP V	N	→	*rain*
NP	→	D N	N	→	*rains*
VP	→	V Adv	V	→	*rain*
VP	→	V NP	V	→	*rains*
			D	→	*the*
			Adv	→	*down*

(a) Show how the **CKY algorithm** would parse the sentence *the rain rains down* with this grammar by filling the following CKY parse table:



(b) Assume the **Earley algorithm** was used to parse the same sentence with the same grammar. Circle all of the following Earley parser states that would **not** appear on the parse chart.
(Hint: You don't have time to simulate the steps of the Earley parser on paper.)

S → NP VP ·$[0, 4]$

S → NP V ·$[0, 3]$

S → NP VP ·$[1, 4]$

N → rain · $[1, 2]$

NP → · rain $[1, 1]$

VP → · V Adv $[2, 2]$

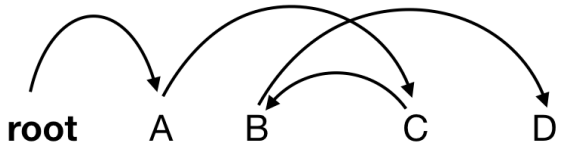
VP → V · Adv $[1, 2]$

Question 6 - Transition-Based Dependency Parsing (16 pts)

Recall the **arc-standard transition-based dependency parser**. The three transitions are (as discussed in class):

- Shift: $(\sigma, w_i|\beta, A) \Rightarrow (\sigma|w_i, \beta, A)$
- Left-Arc: $(\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma, w_j|\beta, A \cup \{w_j, r, w_i\})$
- Right-Arc: $(\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma, w_i|\beta, A \cup \{w_i, r, w_j\})$

As mentioned in class, this transition system cannot generate non-projective dependency trees such such as the following:



Assume now that we are adding the following additional operation:

- Swap: $(\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma|w_j, w_i|\beta, A)$

The Swap transition swaps the next word in the buffer with the word on top of the stack.

Demonstrate that, with this revised transition system, the non-projective dependency tree above can be generated. Show a sequence of transition steps, as well as the intermediate states. Start at state (**root**, [*A,B,C,D*], $\emptyset$)

```
1.([root], [A,B,C,D], ∅) shift
2.([root,A], [B,C,D], ∅) shift
3.([root,A,B], [C,D], ∅) swap
4.([root,A,C], [B,D], ∅) shift
5.([root,A,C,B], [D], ∅) right-arc
6.([root,A,C], [B], {(B,D)}) swap
7.([root,A,B], [C], {(B,D)}) left-arc
8.([root,A], [C], {(B,D),(C,B)}) right-arc
9.([root], [A], {(A,C),(B,D),(C,B)}) right-arc
10.([], [root], {(root,A)(A,C),(B,D),(C,B)}) shift
11.([root], [], {(root,A)(A,C),(B,D),(C,B)})
```

(exta space - please indicate question number clearly)

(exta space - please indicate question number clearly)

(page intentionally left blank)