Midterm for CS378: Natural Language Processing (Spring 2020) Version A

Instructions:

- You will have 75 minutes to complete the exam.
- This exam is to be completed individually be each student.
- You are allowed one 8.5”x11” double-sided note sheet.
- You are **not** allowed calculators or other electronic devices.
- Partial credit will be given for short-answer and long-answer questions, so please show work in the exam.
- For short-answer and long-answer questions, **please box or circle your final answer** (unless it is an explanation).

Grading Sheet (for instructor use only)

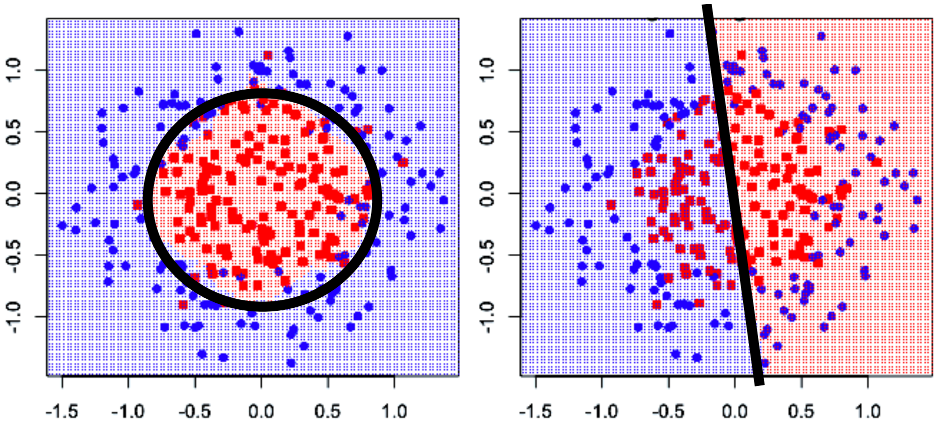
Question	Points	Score
1	14	
2	6	
3	22	
4	21	
5	14	
6	10	
7	13	
Total:	100	

Name: ____

EID: _________________________

Part 1: Multiple Choice (20 points)

1. (14 points) Answer the following multiple choice questions by writing the answer in the provided blank.



______ Version A: B / Version B: D (1) Two models create the decision boundaries shown in the image above. What is the most likely pair of models to generate these?
A. Perceptron (left) / logistic regression (right)
B. Neural network with 2 hidden layers (left) / perceptron (right)
C. Logistic regression (left) / perceptron (right)
D. Neural network with 0 hidden layers (left) / logistic regression (right)

______ Version A: C / Version B: A (2) Using tf-idf (term frequency, inverse document frequency) values for features in a unigram bag-of-words model should have an effect *most similar* to which of the following?
A. Lowercasing the data
B. Dropout regularization
C. Removing stopwords
D. Increasing the learning rate

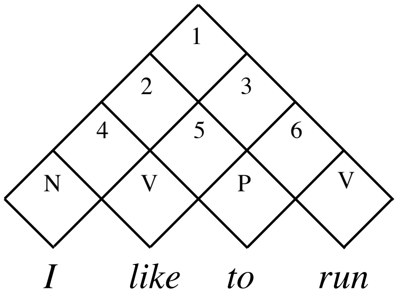
______ Version A: A / Version B: C (3) Batching your data requires which of the following:
I: Modifying your neural network to process multiple examples at once.
II: Computing loss over multiple examples at once.
III: Computing several gradient updates at once and applying them sequentially to the weight vector.
A. I and II B. I and III C. II and III D. I, II, and III

______ Version A: D / Version B: A (4) What does the softmax operator do?
A. It maps real-valued scores into log probabilities (-infinity to 0).
B. It returns the maximum posterior probability sequence for sequence tagging tasks like part-of-speech.
C. It returns the maximum-scoring class as the model's prediction.
D. It maps real-valued model scores to probabilities (0 to 1).

______ Version A: D / Version B: D (5) Consider the equation $y = \text{softmax}(Vg(Wf(\mathbf{x})))$, where $f(\mathbf{x})$ is a vector of features extracted over the input $\mathbf{x}$, V and W are parameter matrices, and g is a nonlinearity. Suppose we remove g from this equation (replace it by the identity function). What happens?

A. Nothing changes.
B. The model is different but just as powerful as before.
C. The model is more powerful because nonlinearities can throw away information by "squashing" their inputs.
D. The model becomes equivalent to logistic regression because two matrix multiplies still yields a linear function.

_____ Version A: B / Version B: B (6) Consider the sentence *I like to run* and a PCFG with nonterminals $\{$S, VP, N, P, V$\}$, terminals $\{$I, like, to, run$\}$, start symbol S, and rules:
S $\rightarrow$ N VP [0.5]
S $\rightarrow$ N V [0.5]
VP $\rightarrow$ V VP [0.5]
VP $\rightarrow$ P V [0.5]
V $\rightarrow$ run [0.5]
V $\rightarrow$ like [0.5]
N $\rightarrow$ I [1.0]
P $\rightarrow$ to [1.0]



The figure has a CKY chart with possible symbols for the leaves filled in showing nonterminals with log probability greater than negative infinity (i.e., nonterminals that are possible to build over these spans of the sentence). Which other cells contain at least one nonterminal with log probability greater than negative infinity?
A. 1, 3, 6
B. 1, 3, 4, 6
C. 1, 2, 3, 4, 6
D. 1, 3, 4, 5, 6

_____ Version A: B / Version B: C (7) Considering the same sentence and PCFG as above, what is the log probability using log base 2 ($\log(0.5) = -1$) of the most probable tree for this sentence?
A. -4
B. -5
C. -6
D. -7

2. (6 points) For the following two questions, circle **all** that apply, or write your selections in the blank. **There may be multiple correct answers.**

_____ Version A: I, V / Version B: II, III (8) Compare using an HMM for sequence labeling to using logistic regression to independently predict each tag. Which of the following is an advantage of HMMs **compared to logistic regression** (so only choose options which apply to HMMs and not LR). Circle all that apply:
I. They allow you to incorporate structural information about adjacent tag pairs.
II. They allow you to use "non-local" features on the input (e.g., the current tag can depend directly on words that are far away in the sentence).
III. They support inference that is asymptotically faster in the number of possible tags.
IV. They are discriminative rather than generative and so directly model $P(\mathbf{y}|\mathbf{x})$, which is what we care most about.
V. They are generative rather than discriminative so we can draw samples of sentences from the distribution $P(\mathbf{y}, \mathbf{x})$.

_____ Version A: I / Version B: III (9) Compare using a CRF for sequence labeling to using logistic regression to independently predict each tag. Which of the following is an advantage of CRFs **compared to logistic regression** (only choose options which apply to CRFs and not LR). Circle all that apply:
I. They allow you to incorporate structural information about adjacent tag pairs.
II. They allow you to use "non-local" features on the input (e.g., the current tag can depend directly on words that are far away in the sentence).
III. They support inference that is asymptotically faster in the number of possible tags.
IV. They are discriminative rather than generative and so directly model $P(\mathbf{y}|\mathbf{x})$, which is what we care most about.
V. They are generative rather than discriminative so we can draw samples of sentences from the distribution $P(\mathbf{y}, \mathbf{x})$.

Part 2: Short Answer (22 points)

3. (22 points) Answer the following short-answer questions.

a. (3 points) What can happen if the step size for SGD is set to be too large? **Name two distinct phenomena that can happen.**

The optimization oscillates around the optimum or diverges totally.

b. (2 points) What can happen if the step size for SGD is set to be too small (or decreases too quickly)? **Name one phenomenon that we might observe.**

Gets to the optimum too slowly or never gets there. Some students wrote that the model gets stuck in a local minimum. This is somewhat correct as a large step size can put you in a basin of attraction for a different minimum, but you're just as likely to overshoot the global minimum as you are to overshoot a local minimum.

c. (3 points) Suppose we are doing sentiment analysis on Twitter. Consider the following sentences (labels included for reference, but these are not part of the tweets):

```
[label = -] wow this was not the news I wanted to get today :((((
[label = +] @user yeeeeeeeah boiiiiiiiii get it!
```

Describe **two new feature types** that you might add to the unigram bag-of-words model to do better in this setting. (A feature type is a set of features; we might describe a unigram bag-of-words model including features of a single type which is "non-stopword unigrams in the sentence".)

There are many possibilities, including indicators on emojis, character n-gram features to capture misspelled words, features based on spelling-corrected versions words, features indicating repeated characters, or features on words with repeated characters compressed into one. We gave less credit for generic feature improvements (add bigrams) that didn't specify why these might work for this data.

d. (3 points) Consider the perceptron algorithm. Suppose the weight vector has n features total, but suppose the number of features active on any example is $k << n$. There are d data examples. Using big-O notation, what is the runtime of one epoch of the perceptron if we implement prediction and updating as efficiently as possible?

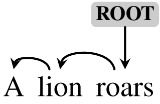
$O(kd)$

e. (3 points) Consider the sentence "*His likes include swimming*". Write **two sentences**, each showing a word in this sentence used in a new context (that you create) where that word now has a different part-of-speech. Each of your two sentences should focus on a **different word** from the original sentence.

He likes bread. He is good at swimming.
This is actually a bit ambiguous since swimming can either be a noun or a present participle. So we accepted a range of answers here depending on how students interpreted the question. *include* is more difficult to use elsewhere (*an include statement* or *C++ includes* are good ones though).

Saying things like *instagram likes* keeps *likes* as a noun, even if it changes the word sense.

f. (3 points) Consider this dependency tree:



Suppose we are parsing with a shift-reduce parser and have done two SHIFT operations already. What operation or operations if performed at this point can lead to a correct tree?

Left-arc. Some students interpreted this question as asking for the whole sequence instead of the immediate next correct decision(s). The whole sequence is: Left-arc, shift, left-arc, right-arc. However, we did not penalize for wrong decisions later in the sequence.

g. (2 points) Consider the skip-gram model

$$P(\text{context} = y|\text{word} = x) = \frac{\exp(\mathbf{v}_x \cdot \mathbf{c}_y)}{\sum_{y'} \exp(\mathbf{v}_x \cdot \mathbf{c}_{y'})}$$

What is the big-O runtime of computing a single probability $P(\text{context} = c|\text{word} = w)$ under this model? Express this in terms of the dimensionality d of the vectors and the vocabulary size $|V|$.

$O(dv)$. The dot product is linear in d and not quadratic, which was one common mistake

h. (3 points) In skip-gram, we can efficiently approximate the denominator if for a given word we know what the largest terms are in its denominator. Suppose you are given a large corpus of text taken from the web. For a particular word w, how could you go about identifying 10 unique context words that are likely to have high weights in the denominator?

Count words, pick the words that most frequently occur with the given word. Some students gave answers that amounted to just computing all of the weight vectors explicitly. This is true, but not in the spirit of approximating the denominator without computing it explicitly; pre-computing counts from a large corpus is quite fast compared to doing explicit computation of all denominators.

Part 3: Long Answer

4. (21 points) Consider the following corpus:

their/N raises/N
he/N raises/V
my/N purses/N

a. (2 points) List the maximum-likelihood initial state probabilities for an HMM estimated from this data.

$P(N)$=1, $P(V)$=0. Note that the initial probability only depends on the first state of each sentence, so $P(N)$=5/6 and $P(V)$=1/6 is not correct.

b. (4 points) List the maximum-likelihood transition probabilities for an HMM estimated from this data (including transitions to the STOP symbol).

	N	V	STOP
N:	2/5	1/5	2/5
V:	0	0	1

c. (3 points) Give an example of a grammatical English sentence **and its tags** using the words above that would be assigned zero probability under this HMM (assuming no smoothing is applied).

he/N purses/V, he/N raises/V purses/N, he/N raises/V their/N purses/N, and other solutions that involve a V to N transition are good. Note that this sentence should use the words above, not just random other English words.

For the following question parts, assume that the log probabilities (log base 2) are as follows (these are rounded so they don't quite normalize as you would expect):

Initial	
N	-1
V	-1

Transitions	N	V	STOP
$y_{i-1} = \text{N}$	-1	-1.5	-2
$y_{i-1} = \text{V}$	-2	-2.5	-0.5

Emissions	their	he	my	raises	purses
N	-4	-2	-3	-3	-2
V	-5	-7	-5	-2	-4

Consider the sentence *he raises purses*

d. (6 points) Draw the Viterbi chart for this data. Be sure to fill in every value. **You do not need to include a final column for the STOP symbol; just fill in the chart up through the third word.**

N: -3, -7, -10
V: -8, -6.5, -12.5

e. (4 points) What is the highest posterior probability tag sequence for this sentence (now including the STOP transition)? **Is this sequence consistent with your interpretation of the sentence above?**

NNN with log probability -12, which is not consistent with our interpretation

f. (2 points) What is the minimum beam size needed to get the same answer as Viterbi on this particular example? Justify your answer in no more than one sentence.

2, since just strictly maxing gives you NVN

5. (14 points) Here's the multiclass perceptron update as discussed in lecture:

Algorithm 1 Multiclass perceptron update

```
Input: labeled data $(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^{D}$, a feature extractor $\mathbf{f}$.
Initialize $\mathbf{w} = \mathbf{0}$, a $d$-dimensional weight vector.
for $t = 1$ to $T$ do                                          ▷ Loop over epochs
    for $i = 1$ to $D$ do                                      ▷ Loop over data
        $y_{\text{pred}} = \arg\max_y \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y)$
        if $y_{\text{pred}} \neq y^{(i)}$ then
            $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{f}(\mathbf{x}^{(i)}, y_{\text{pred}})$
            $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)})$
        end if
    end for
end for
```

Suppose we have the following two examples as our training set:

$x = (1, 0), y =$health
$x = (0, 1), y =$science

We are doing a four-class classification problem with possible labels (health, science, sports, politics) which we denote as (1, 2, 3, 4), respectively. When scoring classes, assume a tie goes to the first class with the highest score; 1 (health) beats 2 (science) if they tie, 2 beats 3, 4; etc.

Our model has eight features under the "different features" view of multiclass perceptron. Suppose the initial feature weights are:
(0, 0, 0, 0, 2, 2, 2, 2)
corresponding to
(health x_1, health x_2, science x_1, science x_2, sports x_1, sports x_2, politics x_1, politics x_2) in that order.

a. (6 points) Execute standard multiclass perceptron to convergence. **Give the weight vector after each epoch until convergence.** Carefully note the tie-breaking semantics described above.

The model updates on both examples in both iteration 1 and iteration 2, then is converged. Full set of weight vectors:
(1 0 0 0 1 2 2 2)
(1 0 0 1 1 1 2 2) [end of epoch 1]
(2 0 0 1 1 1 1 2)
(2 0 0 2 1 1 1 1)[end of epoch 2]

Now, for the following parts, consider modifying the multiclass perceptron algorithm as follows: rather than subtracting off the model's prediction times the features, we instead subtract off 0.5 times the model's prediction and 0.5 times the model's "second-choice" prediction. That is, we modify the algorithm as:

Algorithm 2 Modified multiclass perceptron update

```
Input: labeled data $(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^{D}$, a feature extractor $\mathbf{f}$.
Initialize $\mathbf{w} = \mathbf{0}$, a $d$-dimensional weight vector.
for $t = 1$ to $T$ do                                                    ▷ Loop over epochs
    for $i = 1$ to $D$ do                                                ▷ Loop over data
        $y_{\text{pred},1} = \arg\max_y \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y)$
        $y_{\text{pred},2} = \arg\max_{y, y \neq y_{\text{pred},1}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y)$
        if $y_{\text{pred},1} \neq y^{(i)}$ then
            $\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{2}\mathbf{f}(\mathbf{x}^{(i)}, y_{\text{pred},1})$
            $\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{2}\mathbf{f}(\mathbf{x}^{(i)}, y_{\text{pred},2})$
            $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)})$
        end if
    end for
end for
```

b. (6 points) From the same initial features as in part (a), execute modified multiclass perceptron on this data until convergence. **Give the weight vector after each epoch until convergence.** Carefully note the tie-breaking semantics described above.

As in part (a), the model updates on both examples in both iteration 1 and iteration 2, then is converged. Full set of weight vectors:
(1 0 0 0 1.5 2 1.5 2)
(1 0 0 1 1.5 1.5 1.5 1.5) [end of epoch 1]
(2 0 0 1 1 1.5 1 1.5)
(2 0 0 2 1 1 1 1)[end of epoch 2]

c. (2 points) How might you modify the modified multiclass perceptron code to make to get this to converge more quickly while keeping the core details of the algorithm the same? **Describe the modification, but you do not need to make it.**

Increase the step size

6. (10 points) Consider a PCFG with the following rules (numbers are provided so you can reference them in your solution):

1. DT → the [1.0]
2. N → ring [0.5]
3. N → rings [0.5]
4. CC → and [1.0]

5. ROOT → NP [0.5]
6. ROOT → NP CC NP [0.5]
7. NP → N [0.5]
8. NP → DT N [0.5]

Define our PCFG to have these rules, the nonterminals {NP, CC, N, DT}, terminals {the, ring, rings, and} and root symbol ROOT.

a. (4 points) What parses are possible for the sentence *the ring*? For each parse, list both the tree **and its probability**.

ROOT → NP → DT, N, score = -3. Note that the tree is rooted at the ROOT symbol, so you need to account for this rule.

b. (6 points) This model can generate the NP *ring*, which is not a valid noun phrase. (To convince yourself of this, consider the fact that saying *I want NP* should be grammatical for any choice of NP, and *I want ring* is ungrammatical.)

How can we refine the grammar to prohibit this ungrammaticality **while still being able to produce the following sequences from the ROOT:** *rings*, *the ring*, and *the ring and rings*? In your answer, describe how to modify the grammar rules above accordingly. You can introduce new grammar symbols and rewrite as many rules as you need in the grammar. **Don't worry about changing probabilities, just add/remove rules.**

Remove NP → N
Add NP → NS

Remove N → rings
Add NS → rings

The most general solution also includes a rule NP → DT NS so that the model can generate "the rings"; however, this was not explicitly a requirement that your grammar produce it. Adding a rule NP → rings is also a bit less general but does work for this example.

7. (13 points) Recall that a deep averaging network takes as input a sentence consisting of a sequence of n word embeddings $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and computes $\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$, then passes that through a feedforward neural network. Here is some Python code to do this:

```
10 def label_sentence(sent: string, embeddings: WordEmbedding):
20     tokens = sent.split(" ")
30     embeddings = torch.stack([embeddings[token] for token in tokens])
40     average = torch.sum(embeddings, dim=0)
50     output_log_probabilities = FFNN(average) # defined elsewhere
```

a. (5 points) Modify this algorithm to operate on *character-level* inputs. That is, you should now average over *character* embeddings, assuming these exist in a provided `CharEmbedding` class. You can give pseudocode; it does not need to be correct Python code. You can specify lines that need to change below or modify them above directly. If you need to add lines, you can use indices like 11, 12, 13, etc.

The main modifications are to change WordEmbedding to CharEmbedding and to change the sentence splitting. For the DAN model, no other modifications are required.

b. (3 points) Assume that you are not using off-the-shelf embeddings like GloVe but instead are learning them from your training data. For this algorithm, how should the dimensionality we choose for character embeddings compare to what we use for word embeddings? Justify your choice.

Can be either smaller for speed or smaller for statistical efficiency

c. (3 points) Suppose we have an input in the test set with words that are unseen in the training set (and suppose these words would not appear in GloVe either). How will the character-level model do on these inputs compared to the word-level model? Justify your answer.

A few answers are possible here. Theoretically, the model could use information about characters to accomplish something. However, this model is also extremely impoverished so it will not do well to begin with.

d. (2 points) How does the number of parameters of our new DAN compare to the original DAN?

The number of parameters should be less. First, the embeddings should be lower dimensional, which leads to fewer parameters in the feedforward part of the DAN. Second, there are many fewer character embeddings than word embeddings. So the answer is the same regardless of whether you accounted for these embedding parameters or not.

[BONUS] (2 points) Active learning is when you dynamically label new data instances to improve your model. From the guest lecture, what is the name for the variant of this technique when you ask locally convenient questions during an interactive task? Opportunistic active learning