CS4740/CS5740/LING4474/COGST4740
Fall '22 midterm

First name: ______________________________

Last name: ______________________________

Net ID (all caps): ☐

☐ If an email told you to check this box, check this box; otherwise, leave this box blank

It is a violation of the Academic Integrity Code to look at any exam other than your own, to look at any reference material besides the reference provided in the exam itself, or to otherwise give or receive unauthorized help.

- 90 minutes (1.5 hours), You must remain seated and quiet during the last 10 minutes of the exam period (no leaving early at that point) to allow those still working on the exam to concentrate.
- Closed books, closed notes. No scratch paper allowed; use the blank backs of pages. No external aids, such as calculators.
- Allocate sufficient time to the biggest-point sub-questions: they are the conceptually easiest but will be the strictest graded.
- In your answers, unless otherwise stated,
  - If mathematical or numerical computations are required, leave fractions, sums, and products unreduced. That is, don't compute the decimal equivalent, and leave products/sums written out as products of factors.
  - Show all work and state assumptions and logic employed.
- Have your photo ID out when turning the exam in.
- Afterwards, do not discuss this exam with students who are scheduled to take a later makeup.

This exam has 9 *pages,* 6 *questions,* 76 *points total.*

1. **[14 points] N-gram modeling**

   Here is a small, 3-sentence training corpus. Assume that the vocabulary *only* consists of words occurring in it.

   > mares eat oats and goats eat oats and little lambs eat ivy .
   > a kid will bleat and eat ivy too .
   > piglets sleep and eat .

   No unknown word handling is required in this question. Assume no preprocessing whatsoever: word tokens are simply separated by a space or a newline.

   Note: For all the sub-questions within this question (and elsewhere, wherever applicable), indicate the final answer as a product of fractions that make up the computation, and not just the final value. As an example, consider count(language) = 10 and count(natural) = 20; a properly-formatted (but, irrelevant) answer might be $\frac{\text{count(language)}}{\text{count(natural)}}$ (this would be $\frac{10}{20}$ but just giving the count fraction is fine).

   (a) [6 points] Using the maximum likelihood estimation and *unigram* modeling, show the computation for $P(\text{eat oats and eat ivy})$.

   (b) [6 points] Using the maximum likelihood estimation and *bigram* modeling, show the computation for $P(\text{eat oats and eat ivy})$. When computing the sentence-initial bigram, use the unigram probability.

   (c) [2 points] Now using add-k smoothing with $k = 1$ and *bigram* modeling, show the computation of $P(\text{oats} \mid \text{eat})$.

2. **[21 points] Markov models**

(a) [4 points] The Viterbi algorithm requires two sets of parameters, (1) lexical generation probabilities (i.e., observation likelihoods) and (2) transition probabilities. Assume the task of named-entity recognition (NER) tagging and that an appropriately annotated corpus is provided. Under the assumption of add-k smoothing, how would you mathematically compute the following probabilities (no unseen word handling) in a system of m tags and n tokens:

i. [2 points] The probability of obtaining tag t_j at time $\mathcal{T}$, if you obtained tag t_i at time $\mathcal{T} - 1$.

ii. [2 points] The probability of tag t_i emitting a token w_j at time $\mathcal{T} - 1$.

(b) [3 points] Consider the task of NER tagging, with tags (t_i; $i \in \{1, 2, \ldots, m\}$; t_{start} is a special tag associated with the start of the sequence) and tokens (w_j). Consider the following pseudocode implementation to tag an input sequence of n tokens, $\{w_1, w_2, \ldots, w_n\}$, using a hidden Markov model (HMM):

```
function tag-sequence($\{w_1, w_2, \ldots, w_n\}, t_{\text{start}}$):
    tags: $T \leftarrow [\,]$
    $t_{\text{prev}} \leftarrow t_{\text{start}}$
    for $j = 1$ to $n$:
        scores: $S \leftarrow [\,]$
        for $i = 1$ to $m$:
            $S[i] \leftarrow P(t_i|t_{\text{prev}}) \times P(w_j|t_i)$
        $t_{\text{prev}} \leftarrow \arg\max_{t_i} S$; $T[j] \leftarrow \arg\max_{t_i} S$
    output $T$
```

Under the assumption that all necessary probability values are precomputed, justify if the above implementation is better than the brute-force search, and if not, then propose and justify an alternate approach.

Hint: The search space for the HMM model includes approximately m^n total paths.

(c) [7 points] For the task of NER tagging, consider using a bigram HMM with the Viterbi algorithm. Let us assume the tag set is as follows:

[org]	:	organization
[per]	:	person
[loc]	:	location
[misc]	:	miscellaneous
[o]	:	*not* a named entity
[s-tag]	:	special tag for the start of the sentence token ([s-tok])
[e-tag]	:	special tag for the end of the sentence token ([e-tok])

Further, assume that we train our bigram HMM model on the following 2-sentence corpus (each token is tagged to its corresponding NER tag below it):

[s-tok]	Messi	scored	four	in	World	Cup	.	[e-tok]
[s-tag]	[per]	[o]	[o]	[o]	[misc]	[misc]	[o]	[e-tag]

[s-tok]	Hamm	scored	100	in	NLP	.	[e-tok]
[s-tag]	[per]	[o]	[o]	[o]	[misc]	[o]	[e-tag]

Compute the score [=likelihood, as announced during the exam] assigned by the bigram HMM model to the following sequence (with the associated NER tags) as shown below:

[s-tok]	Hamm	scored	four	in	NLP	.	[e-tok]
[s-tag]	[per]	[o]	[o]	[o]	[misc]	[o]	[e-tag]

Note: Please do *not* attempt to reduce the obtained solution into a single fraction or a decimal equivalent. It is expected that you retain the fractions that make up the computations.

(d) [3 points] String identity can be expensive to use as a feature in MEMMs. Briefly describe an efficient method of including string-based features in an MEMM classifier.

(e) [4 points] Each of the following is a possible MEMM input feature that could be used for NER tagging. Circle all that are valid to use. If none are valid, write "None".

A) Conditional probability of the next token given the current token

B) Conditional probability of the previous token given the current NER tag

C) Conditional probability of the current token given the current POS tag

D) HMM start probability of the current NER tag

3. **[13 points] Word embeddings**

   (a) [4 points] What is the purpose of using negative samples when training word embeddings?

   (b) [4 points] Explain the distributional hypothesis that is foundational to word embeddings.

   (c) [2 points] Distributed word representations (like skipgram) do not explicitly handle unknown tokens. What can you do to get an embedding for an unknown token that was not seen during training?

   (d) [3 points] Using precomputed word vectors (such those learned by Word2Vec) is common in language modeling. But we can also learn word embeddings on the fly when training a feed-forward neural network for specific tasks. Why would you train word embeddings instead of using precomputed Word2Vec embeddings?

4. **[11 points] Neural networks and backpropagation**

Each of the following questions presents a pseudocode implementation of (a part of) a neural network. Looking at the pseudocode, please note if there is anything missing and/or incorrect in the given implementation, if yes, then indicate and correct it, and if not, then just record your answer as nothing being wrong. (Explicitly state and justify any and all assumptions made.)

Notation: $x^{(i)} \in \mathbb{R}^n$ denotes the i-th training sample (of total m samples), $y^{(i)} \in \mathbb{R}$ denotes the prediction for that i-th training sample, and θ_js are the parameters (weights) of the algorithm.

(a) [4 points] The pseudocode is as follows:

```
function $f(x \in \mathbb{R})$:
    $z \leftarrow \exp(x+1)$
    $z \leftarrow \log(z)$
    output $z$

function $\text{forward}_{\text{nn}}(x \in \mathbb{R}^{m \times n})$:
    $\theta \in \mathbb{R}^n \leftarrow \vec{0}$
    $y \in \mathbb{R}^m$
    for $i = 1$ to $m$:
        $h_{\text{out}} \leftarrow \theta^T x^{(i)}$
        $y^{(i)} \leftarrow f(h_{\text{out}})$
    output $y$
```

(b) [4 points] The pseudocode is as follows:

```
function $f(x \in \mathbb{R})$:
    $z \leftarrow 2 + \exp(-x)$
    $z \leftarrow 2/z$
    output $z$

function $\text{forward}_{\text{nn}}(x \in \mathbb{R}^{m \times n})$:
    $\theta \in \mathbb{R}^n \leftarrow \vec{0}$
    $y \in \mathbb{R}^m$
    for $i = 1$ to $m$:
        $h_{\text{out}} \leftarrow \theta^T x^{(i)}$
        $y^{(i)} \leftarrow f(h_{\text{out}})$
    output $y$
```

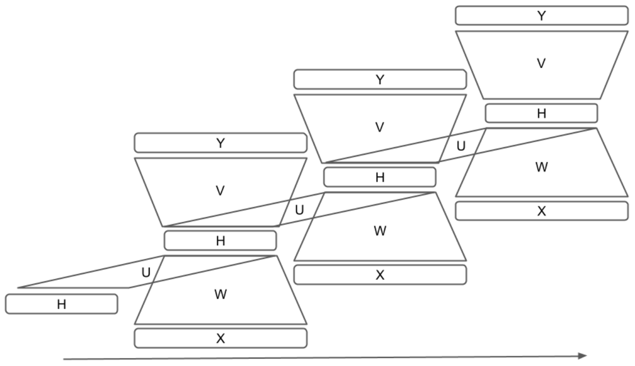
(c) [3 points] The pseudocode is as follows:

```
function $f(x \in \mathbb{R})$:
    $z \leftarrow \sqrt{x}$
    output $z$
```

```
function forward_nn($x \in \mathbb{R}^{m \times n}$):
    $\theta \in \mathbb{R}^n \leftarrow \vec{0}$
    $y \in \mathbb{R}^m$
    for $i = 1$ to $m$:
        $h_{\text{out}} \leftarrow \theta^T x^{(i)}$
        $y^{(i)} \leftarrow f(h_{\text{out}})$
    output $y$
```

Hint: Visualize the network activation.

5. **[14 points] Recurrent neural network theory**



Consider the task of using a simple recurrent neural network (RNN) (shown above) for the task of language modeling (i.e. to predict the next word in a sequence of words).

(a) [4 points] What do W and U represent? That is, what functions do they perform?

(b) [4 points] In some language models X and Y will not have the same dimensionality. Why not, and in those cases, what is their dimensionality?

(c) [3 points] At test time, which of the objects denoted by variables in the above image change over time, and which do not? In your answer, account for all of H, U, V, W, X, Y.

(d) [3 points] What is the dimensionality/shape of U? You can use the function **len(·)** in your answer, which takes a vector as an argument and returns its length.

6. **[3 points] Recurrent neural network applications**

   Consider a system with the following characteristics:

   - Input: note made by a doctor
   - Step 1: run a recurrent neural network language model (RNN-LM) on the input
   - Step 2: run a classifier on the final hidden layer activations of the RNN-LM
   - Output: “serious” (=patient has a serious medical condition, according to the doctor), or “not serious”

   You notice that that the system mistakenly classifies the note below as “serious”. You think the problem might involve the last words, “deadly disease”, whose occurrence is misleading here.

   > The patient is 24 years old and has not been diagnosed with any chronic health conditions. All vital signs normal. A smallpox test was negative. I am so looking forward to the end of that deadly disease!

   (a) [3 points] Explain how the architecture of an RNN-LM could be leading to the classification error.