COS 484: Natural Language Processing **(Due: 03/31/25)**

Assignment #3

Instructors: Danqi Chen, Tri Dao, Vikram Ramaswamy

Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission. The course assignment policy is available at http://nlp.cs.princeton.edu/cos484.

- This assignment contains 2 parts, a theoretical and a programming part. The former consists of 2 problems, and the latter has 2, for a total of **4 problems**.
- We *highly* recommended that you typeset your submissions in LaTeX. Use the template provided on the website for your answers. If you have never used LaTeX, you can refer to the short guide here: http://bit.ly/WorkingWithLaTeX. Include your name and NetIDs with your submission. If you wish to submit hand-written answers, you can scan and upload the pdf.
- Assignments must be uploaded to Gradescope **before class (01:59pm Eastern)** on the due date mentioned above.
- As per the late-day policy outlined on the course website, you have 4 late days to use at your discretion throughout the semester. Once you run out of late days, late submissions will incur a penalty of 10% for each day, up to a maximum of 3 days beyond which submissions will not be accepted.
- All programming problems in this class will be completed in Google Colab using Python. If you would like to get familiar with this environment, you may complete the problems in this introductory Colab notebook (This will not be graded). If you've never worked with Google Colab before, take a look through this introduction guide: http://bit.ly/WorkingWithColab. **The answers to the written questions proposed in the programming part should be answered in the Colab notebook.**

Problem 1: Language modeling with recurrent neural networks (10 points)

In this problem, we will consider a recurrent neural network for language modeling. Given a sequence of one-hot input vectors $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n$ (see definition of one-hot vectors in A2P3), our RNN calculates the following at each timestep t:

$$\begin{aligned}\mathbf{e}_t &= \mathbf{x}_t E \\ \mathbf{h}_t &= \tanh(\mathbf{h}_{t-1}U^\top + \mathbf{e}_t W^\top) \\ \mathbf{y}_t &= \text{softmax}(\mathbf{h}_t W_O^\top)\end{aligned}$$

Let us assume that $\mathbf{x}_t \in \mathbb{R}^{1\times|V|}$ where V is vocabulary, $\mathbf{e}_t \in \mathbb{R}^{1\times D}$, $\mathbf{h}_t \in \mathbb{R}^{1\times H}$, and $\mathbf{y}_t \in \mathbb{R}^{1\times|V|}$. As in prior homework, E is a word embedding matrix which is learned during training. However, now we have introduced an additional word embedding W_O for the output of the language model.

(1) (2 points) How many trainable parameters does our RNN model have?

(2) (4 points) Write down the expression for the gradient $\frac{\partial \mathbf{y}_3}{\partial E}$ in terms of the gradients corresponding to each computation of the RNN, using backpropagation through time, i.e. in terms of gradients of the form $\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}$, $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}$, $\frac{\partial \mathbf{h}_t}{\partial \mathbf{e}_t}$, $\frac{\partial \mathbf{e}_t}{\partial \mathbf{E}}$.

(3) (2 points) In practice, it is common to enforce that the input and output word embeddings matrices are the same e.g. let $E = W_O$ (assume that $H = D$). Can you suggest a reason the weight tying might be beneficial?

(4) (2 points) A common issue with recurrent neural networks is that they exhibit vanishing gradients (i.e., going to zero) or exploding gradients (i.e., going to very large numbers) as the sequence length increases. We consider a simplified setting where this may occur.
Suppose that $H = D = 2$, $\mathbf{e}_t = 0$, for all timesteps t. Also suppose that for simplicity, instead of *tanh* we use *ReLU* - in other words, we have:

$$\mathbf{h}_t = \text{ReLU}(\mathbf{h}_{t-1}U^\top),$$

where

$$ReLU(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0. \end{cases}$$

Can you give examples of nonzero $U \in \mathbb{R}^{2\times 2}$ which are diagonal matrices of positive numbers, and for which the Jacobian $\frac{\partial \mathbf{h}_t}{\partial U}$

(i) has terms which all tend towards 0

(ii) has terms which all tend towards infinity

as t becomes larger? You may assume that all values in $\mathbf{h}_0$ are positive.

Problem 2: IBM models (15 points)

You are given the following dataset of translations from "simple" to "difficult" English:

1. *Simple*: `kids like cats`
   Difficult: `children adore felines`
2. *Simple*: `cats hats`
   Difficult: `felines fedoras`

Consider estimating a word-to-word statistical translation model from simple English (source) to difficult English (target).

We have $w^{(s)} \in \{$`kids, like, cats, hats`$\}$ and $w^{(t)} \in \{$`children, adore, felines, fedoras`$\}$. Assume we are using the IBM model 1 assumption for alignments, i.e. $p(a_m \mid m, M^{(s)}, M^{(t)}) = \frac{1}{M^{(t)}}$.

To keep things simple, we will not add a `NULL` token in the source and target sentences for performing alignments.

(a) (2 point) What is the value of $P(w^{(s)} \mid w^{(t)})$ if we assume a uniform translation model?

(b) (3 points) How many different alignments are possible for each pair of sentences in the corpus?

(c) (5 points) Now consider the following alignment

$$\mathcal{A} = \{(\text{kids}, \text{children}), (\text{like}, \text{adore}), (\text{cats}, \text{felines}), (\text{cats}, \text{fedoras}), (\text{hats}, \text{felines})\}$$

What are the translation probabilities $P(w^{(s)}|w^{(t)})$ for all pairs of source and target words? You only have to specify the non-zero ones.

(d) (2 points) What is the most likely translation of the new sentence `cats like hats` given the translation probabilities in part (c) and using the IBM model 1 assumption for alignments? That is, solve for the most likely target sentence $\mathbf{w}^{(t)}$:

$$\underset{\mathbf{w}^{(t)}}{\operatorname{argmax}}\, p(\mathbf{w}^{(t)}|\mathbf{w}^{(s)})$$

You are also given the following fluency probabilities

$$P(\text{fedoras adore fedoras}) = 0.2$$
$$P(\text{felines adore fedoras}) = 0.25$$
$$P(\text{children adore felines}) = 0.4$$
$$P(\text{adore fedoras felines}) = 0.10$$
$$P(\text{adore fedoras fedoras}) = 0.05$$

Show your work.

(e) (3 points) Compute the BLEU-2 score of your translation in part (d) using the reference translation `felines adore chapeaus`. Do not use any length penalty.

Programming 1: NER with feed-forward neural networks (10 points)

In the following programming problems, you are going to implement models for the Named Entity Recognition (NER) task. NER is the task to associate the words in a sentence with their proper name tags. For example, "Marie Curie" may correspond to the tag `PER` (person) and "Princeton University" may correspond to the tag `ORG` (organization). In this programming assignment, will use a total of 5 tags: `PER` (person), `ORG` (organization), `LOC` (location), `MISC` (miscellaneous), and `O` (non-entity). For example, the correct tagging of the sentence "Steve Jobs founded Apple with Steve Wozniak ." is ⟨`PER`, `PER`, `O`, `ORG`, `O`, `PER`, `PER`⟩. Note that when consecutive words constitute a named entity, such as "Steve Jobs" in the previous example, they should both be tagged as `PER`

In programming problem 1, you will implement a feed-forward network for this task.

Link to notebook: Colab notebook.

Programming 2: NER with Long Short-term Memory (18 points)

The task is the same as the programming problem above. In programming problem 2, you will implement the LSTM model for this task.

Link to notebook: Colab notebook.