Assignment 3

CMSC 473/673 — Introduction to Natural Language Processing

Due Tuesday December 12th, 11:59 PM

Item	**Summary**
Assigned	Thursday November 30th
Due	Tuesday December 12th
Topic	Language Modeling and Insights
Points	75 (+50 EC)

In this assignment you will explore neural language models.

You are to *complete* this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Discord discussions).

The following table gives the overall point breakdown for this assignment.

Question	1	2	3	4	5
Points	25	25 EC only	30	20	25 EC only

What To Turn In Turn in a writeup in PDF format that answer the questions; turn in all requested code necessary to replicate your results. If you write code, be sure to include specific instructions on how to build and run your code. Answers to the following questions should be long-form.

How To Submit Submit the assignment on the submission site:

https://www.csee.umbc.edu/courses/undergraduate/473/f23/submit.

Be sure to select "`Assignment 3`."

Full Questions

1. (**25 points**) Consider a sequence on N input words $w_1 w_2 \dots w_N$ and a corresponding sequence of output labels $y = y_1 y_2 \dots y_N$. Assume we can represent each input word w_i via some embedding, e_{w_i}.

   In class, we looked at the following RNN cell definition, that processes the input sequence left-to-right:
   $$h_i = f(Wh_{i-1} + Ue_{w_i}), \tag{1}$$
   where f is a non-linearity that is applied component-wise to a vector (e.g., if $f = \exp$, then $f(1, 2) = (\exp(1), \exp(2))$). We would then form a distribution over our output labels as $\text{softmax}(\theta h_i)$: this may often be written as $p(y_i|w_{\le i}) = \text{softmax}(\theta h_i)$, to indicate that distribution over output values for y_i is taking into account w_i and all previous words.

   In these equations, assume h_i is a K dimensional vector, each embedding vector e_v is an E dimensional vector, and there are L possible label types (that is, each y_i can be one of L possible values).

   (a) What does each of W, U, and θ serve to do?

   (b) What are the shapes of W, U, and θ?

   (c) If we were using this RNN to learn a language model, and we had the sentence "The can can hold liquid," what are each of the w_i and y_i values? (Hint #1: $N = 6$. Hint #2: Both w and y are of length 6 but they are not *exactly* equal. Hint #3: Remember that for language modeling you must learn to predict an EOS symbol.)

   (d) How would training an RNN as a language differ when using teacher forcing vs. not using teacher forcing?

2. (**EC: 25 points**) In class, we considered cases where these labels could be the original words themselves (so the RNN would learn a language model to predict what the next word is), part of speech tags, and named entity tags. One limitation of the RNN definition in the previous question is that only processes data left-to-right.

   (a) Give an example, either from part of speech tagging or named entity classification, where this left-to-right processing is a potential disadvantage. That is, give an example sentence w and labels y where you believe that predicting y_i would benefit from observing some word w_j that occurs after w_i.

   (Your example does not need to be an English example. But, if you provide an example in another language, you must provide an English explanation.)

   Luckily, there's an easy fix for this: it's called a bi-directional model, and all it does is it forms both a left-to-right representation l_i *and* a right-to-left representation r_i, and then concatenates them to form h_i. This changes the definition to:
   $$l_i = f(Wl_{i-1} + Ue_{w_i}) \tag{2}$$
   $$r_i = f(Zr_{i+1} + Qe_{w_i}) \tag{3}$$
   $$h_i = \text{concat}(l_i, r_i) \tag{4}$$
   $$p(y_i|w_1, w_2, \dots, w_N) = \text{softmax}(\theta h_i) \tag{5}$$

Here, we're introducing two additional parameter matrices (Z and Q).

(b) Why is this formulation okay to use for a problem like part of speech tagging or named entity recognition, but it would *not* be okay to use for a problem like language modeling?

(c) Consider the sequence $x =$ "cats eat" with part of speech label sequence $y =$ "Noun Verb." Under the following assumptions, compute each of $l_1, r_1, l_2, r_2, p(y_1|w_1, w_2), p(y_2|w_1, w_2)$, and each of the loss terms for y_1 and y_2. The assumptions are:

- The only two possible POS labels are Noun and Verb.
- Each word is represented as a 2-dimensional (column) vector: $e_{\text{cats}} = (0.75, 0.25)$ and $e_{\text{eat}} = (0.3, 0.2)$.
- $W = Z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $U = Q = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$
- $\theta = \begin{pmatrix} \theta_{\text{Noun}} \\ \theta_{\text{Verb}} \end{pmatrix} = \begin{pmatrix} 0.7 & -0.4 & -0.3 & 1 \\ -0.2 & 0.35 & 0.4 & 1 \end{pmatrix}$
- The non-linearity $f = \tanh$.
- $l_0 = r_{N+1} = \mathbf{1}$ are both 2-dimensional vectors all of 1s.

If you would like, you may verify your computations using Pytorch. This is optional but if you do so, turn in your code. (Note that in this case, each of the matrix-vector products would be represented by a linear layer with `bias=False`.)

3. (**30 points**) Read Bender et al. (2021). The article is available at `https://dl.acm.org/doi/10.1145/3442188.3445922`). After reading this article, think about the potential benefits and costs or harms of large language models (LLMs). Then, write a 1/2 page reflection piece. Your reflection should

   - contain a brief summary of the findings/conclusions of this paper.
   - explain your opinion on the benefits of LLMs, accounting for the potential costs.
   - be understandable by a general UMBC student audience (don't assume someone has taken this class).

4. (**20 points**) Read the overview and training portions of the `T5` Huggingface documentation: `https://huggingface.co/docs/transformers/v4.28.1/en/model_doc/t5#t5`.

   (a) The documentation mentions that `T5` is an encoder-decoder model. Which of the "5 model types" (as discussed in class) is `T5`? Briefly explain why.

   (b) The `T5ForConditionalGeneration` page says that it's a "T5 Model with a language modeling head on top." Describe what this means, both in terms of what the "language modeling head" is doing, and how you might compute it using Pytorch.

   (c) Let's say you wanted to train a `T5ForConditionalGeneration` to do the English to French translation task from class. What are the *required* arguments you'd need to provide to the `forward` function? Ground your answer by using the example of translating "The cat is on the chair" into the French sentence "Le chat est sur la chaise."

5. (**EC: 25 points**) You've previously looked at training a classifier for the RTE task. In this question, fine-tune a `T5ForConditionalGeneration` model on the GLUE RTE dataset. You should take in paired sentences, and generate either `entailed` or `not_entailed`. Use either `t5-small` or `t5-base` (if you want to go larger you can try, but you may need to significantly reduce the batch size.) Evaluate your model using accuracy, recall, precision, and F1 (do an exact match on the generated output) and provide a brief reflection on its capabilities (and limitations).

   For the input, you should prepend each sentence with the string "sentence1" or "sentence2," and then concatenate those sentences together. You can see this exemplified in Appendix D.2 (page 47) of the T5 paper (`https://arxiv.org/abs/1910.10683`).

   Note: I highly recommend using a GPU. You should be able to do this in the UMBC colab setup.