CS 4650 (Spring 2021) HW3 – LSTM Model

February 2021

Instruction

1. We will be using Gradescope to collect your assignments. Please carefully read the following instructions for submitting to Gradescope.
   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under **HW3 Writing**), make sure to mark which page(s) correspond to each problem or subproblem.
   (b) For the coding problem, please upload your jupyter notebook (with all the outputs and your answers present) in a zip under **HW3 Programming** on Gradescope.
   (c) Note: this is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.
2. LaTeX solutions are strongly encouraged (solution template available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.
3. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on the submission site.

1 Word Embeddings (10 points)

In the class, we focused on discussing word2vec as one of the word embedding techniques. Here, we will consider a simpler alternative method for learning word embeddings. Consider the term-document matrix for five words in three documents shown in Table 1. The whole document set has N = 20 documents, and for each of the five words, the document frequency df_t is shown in Table 2.

Term	Doc 1	Doc 2	Doc 3
water	9	2	0
current	5	7	8
electricity	0	3	8
flow	5	4	6
swim	4	0	0

Table 1: Term Document Matrix

Term	DF
water	8
current	17
electricity	7
flow	14
swim	4

Table 2: Document Frequency

(a) Compute the *tf-idf* weights according to the definitions in Jurafsky and Martin's Textbook (Chapter 6; Equations 6.12-6.14) for each of the words "water", "current", "electricity", "flow" and "swim" in Doc 1, Doc 2, and Doc 3. (5 points)

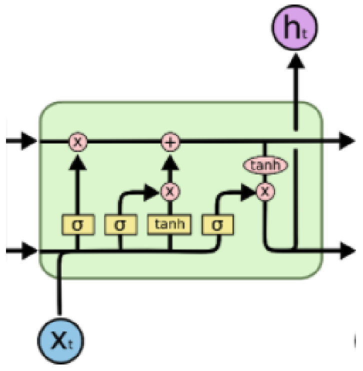
(b) Use the *tf-idf* weight you got in (a) to represent each document with a vector, and calculate the cosine similarities between these three documents. (3 points)

(c) Stemming refers to reducing a word to its root word by removing its suffix, e.g. "swimming" to "swim". Which one of the methods, *word2vec* and *tf-idf*, do you think would benefit from stemming? Explain why. (2 points)

2 LSTM (10 points)

Here are the defining equations for a Long Short-Term Memory (LSTM) cell:

$$\begin{aligned}
i_t &= \sigma(\mathbf{W}^{(i)} x_t + \mathbf{U}^{(i)} h_{t-1}) \\
f_t &= \sigma(\mathbf{W}^{(f)} x_t + \mathbf{U}^{(f)} h_{t-1}) \\
o_t &= \sigma(\mathbf{W}^{(o)} x_t + \mathbf{U}^{(o)} h_{t-1}) \\
\widetilde{c}_t &= \tanh(\mathbf{W}^{(c)} x_t + \mathbf{U}^{(c)} h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}$$



(1) Recall that $\circ$ denotes element-wise multiplication and that σ denote the sigmoid function $\sigma(a) = 1/(1 + e^{-a})$. h_t, c_t and x_t are column vectors. Assume that the h_t are of dimension d_h, that c_t are of dimension d_c and that the x_t are of dimensions d_x. What are the dimensions of $W^{(i)}$, $U^{(i)}$, $W^{(f)}$, $U^{(f)}$, $W^{(o)}$, $U^{(o)}$, $W^{(c)}$, and $U^{(c)}$? Define clearly which numbers are rows and columns. (8 points)

$\mathbf{W}^{(i)}$:

$\mathbf{U}^{(i)}$:

$\mathbf{W}^{(f)}$:

$\mathbf{U}^{(f)}$:

$\mathbf{W}^{(o)}$:

$\mathbf{U}^{(o)}$:

$\mathbf{W}^{(c)}$:

$\mathbf{U}^{(c)}$:

(2) True or False. i_t, f_t and o_t can be viewed as probability distributions (i.e., their entries are non-negative and their entries sum to 1). JUSTIFY YOUR ANSWER. (2 points)

3 Programming: LSTM POS-Tagger (30 points)

In this problem, you will implement two LSTM part-of-speech tagging models, and carry out hyper-parameter tuning. Write down your answers in the space provided in the notebook. When making a submission, make sure all the outputs and your answers are present in the notebook.

(a) Implement a basic LSTM POS-tagger by completing the skeleton code provided in `BasicPOSTagger` in the notebook. Next, implement the training procedure. Under the default hyper-parameter setting, after 5 epochs you should be able to get at least 75% accuracy on the validation set.

Compute the top-10 most frequent types of errors (e.g. 10 instances where the model labeled NN as VB) that the model made in labeling the data in the validation set, and report them in the notebook. What kinds of errors did the model make and why do you think it made them? (10 points)

(b) In order to improve your model performance, try the hyper-parameter tuning. Specifically, you are expected to make some modifications on `EMBEDDING_DIM`, `HIDDEN_DIM`, and `LEARNING_RATE`. You will receive 50%/75%/100% credit for this section if your model, after being trained for 10 epochs, is able to achieve 80%/85%/90% accuracy on the validation set. (10 points)

(c) Word-level information is useful for part-of-speech tagging, but what about character-level information? For instance, the past tense of English verbs is marked with the suffix *-ed*, which can be captured by a sequential character model.

Implement the character-level LSTM POS-tagger by completing `CharPOSTagger`. The key idea is to use the character-level information to augment word embeddings. Under the default hyper-parameter setting, after 5 epochs you should be able to get at least 85% accuracy on the validation set.

Compute the top-10 most frequent types of errors (e.g. 10 instances where the model labeled NN as VB) that the model made in labeling the data in the validation set, and report them in the notebook. What kinds of errors did the model make and why do you think it made them? Compare your findings with part (a). (10 points)