CS 505 – Spring 2022 – Assignment 4 (100 pts, BONUS: 10 pts) – Neural Models

Problems due 11:59PM EST, April 13, 2022.

In this assignment, you will learn about text classification and language modeling using RNN and LSTM, and use **pytorch**—a deep learning framework. You have 2 weeks to finish this assignment.

Submission Instructions Read the submission instructions fully as they are updated.

- You are going to submit **a single pdf** which will contain your written answers for **all questions** followed by your code at the end. Do not intertwine code with answers. Additionally provide a link to your code at Colab at the beginning of your submission. We will only accept links to Colab notebooks. In generating a pdf from you notebook, please avoid unnecessarily long outputs which will crowd your submission and graders might fail to find the relevant code section resulting point deductions. Comment your code well to provide justifications.
- When submitting on Gradescope, make sure you assign the correct page for a given part, otherwise the grader won't be able to see your answer. Do not assign code to individual parts, only your written answers at the beginning.
- There is no template for this homework so make sure you precisely answer every part some of which may ask you to answer multiple things and justify your answers. Unjustified answers won't be given points. For the short answers, make the final answer **bold**.
- Please indicate names of those you collaborate with. While collaboration is allowed, only submit your own code and answers.
- Every late day will reduce your score by 20. After 2 days (i.e., if you submit on the 3rd day after due date), it will be marked 0.

Problem 1. *Neural Models (35 pts)*

1. *(2 pts) In a news framing classification task, where you have 5 frames and your model predicts each of the frames with equal probability for an article, what is the cross entropy loss of the article in this case?*
2. *(2 pts) Suppose during training of your neural model you realize that your training loss remains high. Mention some of the ways you can reduce this **underfitting** of your neural network.*
3. *(2 pts) After you do many changes to your neural network, you now realize that your training loss is much lower than your validation loss. Mention some of the ways you can reduce this **overfitting** of your neural network.*
4. *(2 pts) What is good about setting a large batch size for training? How about a small batch size?*
5. *(4 pts) How can an RNN be used for detecting toxic spans (spans of words containing toxic language) in a social media comment? Specifically, what should be the input to the RNN at each time step t? How many outputs (i.e., $\hat{y}$) are produced given a comment containing n words? What is each $\hat{y}^{(t)}$ a probability distribution over?*
6. *(4 pts) How about using RNNs for language modeling? Given a start word token as input at time step 1, what should be the input to the RNN at each time step $t > 1$? How many outputs are produced? What is each $\hat{y}^{(t)}$ a probability distribution over?*
7. *(3 pts) How about using RNNs for frame classification? Given an article containing n words as input, what should be the input to the RNN at each time step t? How many outputs are produced? What is each $\hat{y}^{(t)}$ a probability distribution over?*
8. *(2 pts) What is the main advantage of using RNNs for frame classification over feed forward neural network?*
9. *(4 pts) What is the disadvantage of RNN when used to classify the sentiment of a very long tweet like this? "I am not sure I want this phone. It's too big to fit in my back pocket. I put it in and accidentally sat on it and now it's bent. I'm very disappointed. I'm now the proud owner of bendy iPhone13. Very proud." What is the appropriate sentiment for this tweet? And what would the RNN classify it as?*

10. *How about LSTM? Given this formulation of LSTM: $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$ (forget gate), $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$ and $\hat{C}_t = tanh(W_C x_t + U_C h_{t-1} + b_C)$ (input gate), $C_t = f_t * C_{t-1} + i_t * \hat{C}_t$ (update gate), and $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$ and $h_t = o_t * tanh(C_t)$:*

    (a) *(4 pts) derive the formulation of $\frac{\partial J}{U_C}$, where J is the loss function, for two time steps t and $t-1$ in terms of $\frac{\partial J}{\partial h_t}, \frac{\partial h_t}{\partial C_t}, \frac{\partial C_t}{\partial U_C}, \frac{\partial C_t}{\partial C_{t-1}}, \frac{\partial C_{t-1}}{\partial U_C}, \frac{\partial h_t}{\partial h_{t-1}}$, and $\frac{\partial h_{t-1}}{\partial U_C}$.*

    (b) *(2 pts) which part of $\frac{\partial J}{U_C}$ reduces the effect of the vanishing gradient problem in RNNs?*

    (c) *(2 pts) How does this help classify the correct sentiment of the tweet above?*

    (d) *(2 pts) Instead of using the last hidden state of LSTM to classify the tweet, what other ways we can do to improve the performance of this sentiment classification?*

Problem 2. *LSTM for language modeling (33 points)*

1. *(10 pts) Follow the tutorial in here to train a word-level LSTM language modeling. Train the language model on texts from the file prideAndPrejudice.txt. Before using it to train the language model, you need to first sentence-segment, then tokenize, then lower case each line of the file using Spacy. Append start-of-sentence token '<s>' and end-of-sentence '</s>' token to each **sentence** and put each sentence in its own line. Use only words that appear more than once in this corpus and assign UNK tokens for the rest; you may also need to pad sentences that are shorter than 5 (see here in cell 10-12 for adding unknown: UNK token and padding: PAD token to your vocabulary). Train the language model and save the trained model (see here). Generate 10 examples of text from it, starting from '<s>' token and ending at '</s>' token.*

2. *(7 pts) Compute and report the perplexity of the saved model on test_1.txt file. Note that the test files are already pre-processed.*

3. *(5 pts) Train the language model as before, but with input sequence lengths of 25 (currently, it's inputs are of length 5). You may need to pad some of the shorter sentences to length 25. Save your trained model. Generate 10 examples of text from it, starting from '<s>' token and ending at '</s>' token. Are there differences from the generated examples from 2.1?*

4. *(2 pts) Compute and report the perplexity of this saved model on test_1.txt file.*

5. *(2 pts) Use the better language model (the one with the lower perplexity on test_1.txt) to compute and report the perplexity on test_2.txt. Note that the test files are already pre-processed.*

6. *(5 pts) Train the better language model as before but start with pre-trained Glove6B 100d embeddings (see here on how to incorporate pretrained embeddings in your LSTM model). This time, use all your words in the corpus as vocabulary, even those occurring only once in the corpus. Only assign UNK token to words that are not in Glove vocabulary and initialize random vectors in the embedding matrix for the UNK, <s>, </s>, and PAD tokens using a standard gaussian distribution with σ set to 0.6 (see numpy.random.normal). Save your trained model. Generate 10 examples of text from it, starting from '<s>' token and ending at '</s>' token. Are there differences from the generated examples from before?*

7. *(2 pts) Compute and report the perplexity of this saved model on test_1.txt file.*

Problem 3. *LSTM for classification (32 points, BONUS: 10 pts)*

1. *(5 pts) Follow the tutorial here on how to build LSTM model for sentiment classification. Modify the tutorial to train on your tweet sentiment data (sentiment-train.csv) and test on test data (sentiment-test.csv) from HW2 (modify the tutorial so that the train data is **not** split into train and validation). Compute and report the accuracy on the test data.*

2. *(2 pts) Modify the model from 3.1 to use GRU. Compute and report the accuracy on the test data.*

3. *(5 pts) Modify the model from 3.1 to use bidirectional LSTM. Compute and report the accuracy on the test data.*

4. *(2 pts) Modify the model from 3.1 to use bidirectional GRU. Compute and report the accuracy on the test data.*

5. *(5 pts) Pick the best model so far and train the model starting from pretrained GloveTwitter 100d (use the same vocabulary as before, just initialize the embedding of the words using Glove embeddings). Compute and report the accuracy on the test data.*

6. *(10 pts) Using your best model so far, conduct a 5-fold (stratified) cross validation on your training data and a grid search to pick the best hidden size (try 128 or 512) and embedding size (try 100 or 400). Compute and report the average accuracies for each of the choice combination.*

7. *(3 pts) Train the model on all your training data using the best combination of hyperparameters you find in 3.6. Compute and report the accuracy on the test data.*

8. *(BONUS: 10 pts) Train your best model using the hyperparameter from 3.6 on all the sentiment140 data. Compute and report the accuracy on the test data from HW2 (i.e., sentiment-test.csv) .*