CS 4650/7650 Fall 2020: Homework 4

October 5, 2020

Instructions

1. This homework is a programming assignment.

   We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   (a) The programming assignment requires you to work on boilerplate code. Submit the code to the programming assignment in a zip that contains `POS_tagging.ipynb`. This submission is to be made under HW4 Programming.

   (b) The assignment is divided into three parts. After each part, you should submit the output for the test data to Gradescope HW4 testing. The file should be named `test_labels.txt` and should contain the predicted tags from your POS tagger. You can see your score for test data as soon as you submit it. Note this score and report it in your notebook submission.

   (c) This programming assignment can have high execution time if you don't write vectorized code. Please try to avoid writing for loops and instead use vector/matrix/tensor operations throughout.

   (d) 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. 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.

3. The code files needed to complete the homework are included in a zip file with links on Canvas.

1. In this assignment, you will implement two part-of-speech tagging models. You will first implement a word LSTM tagger and then a character LSTM tagger. You will write your code in `POS_tagging.ipynb` using train (`train.txt`) and test data (`test.txt`) in `POS_tagging.zip`.

   NOTE: for each model that you test, you will produce a file `test_labels.txt` that you can upload to Gradescope to evaluate your model's performance. Also, submit your descriptive answers in the space provided in the notebook.

   (a) To begin, implement an LSTM tagger by completing the skeleton code provided in `BasicPOSTagger` in the notebook. The model will use *word embeddings* to represent the sequence of words in the sentence.

   For this problem, implement the forward pass and the training procedure for the tagger. After training for 30 epochs, the model should achieve around 75% accuracy on the validation set.

   In addition, 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, created for you using the skeleton code, and report these errors in the written answer. Report the results in a table containing the top-10 mistakes, with the following columns: model tag, ground truth tag, error frequency, up to 5 example words that were responsible for the mistake. What kinds of errors did the model make and why do you think it made them?

   Finally, upload the file `test_labels.txt` to Gradescope to evaluate your model's performance and report the accuracy in the notebook. [25 pts]

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

   For this problem, implement the character-level model by completing the skeleton code provided in `CharPOSTagger` in the notebook. Unlike part (a), this model will use *character* embeddings. After training for 30 epochs, the model should achieve around 77% accuracy on the held-out data.

   In addition, compare the top-10 types of errors made by the character-level model with the errors made by the word-level model from part (a) and report these errors in the written answer. Report the error results in the same format as before. What kinds of errors does the character-level model make as compared to the original model, and why do you think it made them?

   Also, upload the file `test_labels.txt` to Gradescope to evaluate your model's performance and report the accuracy in the notebook. [50 pts]

   (c) Implement one of the three modifications listed in the notebook to boost your score: (a) change the number of LSTM layers, (b) change the number of word embedding dimensions, (c) change the number of hidden dimensions. Train the modified tagger for 30 epochs. Explain which modifications you have made,

why you chose a certain modification over another and the resulting accuracy of the modified model.

Lastly, compare the top-10 errors made by this modified model with the errors made by the model from part (a). Report the error results in the same format as before. What errors does the original model make as compared to the modified model, and why do you think it made them?

Again, upload the file `test_labels.txt` to Gradescope to evaluate your model's performance and report the accuracy in the notebook. [25 pts]