Brandeis University | COSI-165B | Deep Learning
Assignment 3 (100 points)
Due Date: April 18, Sunday, 23:59 PM (EST)

Instruction of homework submission

1. Submit your solutions in one pdf file named cosi165b- assignment -3-[yourname].
2. Submit your code files in one zip file named cosi165b-assignment-3-code-[yourname]. The code should include necessary comment.
3. Late policy: (late hours / 24) * 10%, that is one day late leads to 10% loss. If late hours > 7 * 24 (one week), it will be 100% loss.
4. There may be code of similar problems available on the internet (e.g., Github). It is okay to refer those code while you should finish your own code. If we find your code is the same as them, you will get 0 (100% loss) for this assignment.
5. Please obey academic integrity, no plagiarize will be allowed. Plagiarize will lead to serious penalties, e.g., failure on the assignment or failure in the course.
6. For any question, please contact TA (hanyue@brandeis.edu) or instructor (chuxuzhang@brandeis.edu).

Deep Learning for NLP: Word and Text Embeddings (100 pts)

In this problem, you will build a word and text embedding model for paper classification (5 label classification) using Pytorch. You will need to finish code in trnn_main.py, trnn_utils.py and word2vec.py, and answer some questions.
Dataset
paper_abstract.txt – paper abstract data (for reference, not necessary for coding)
paper_label.txt – paper label data
paper_abstract.pkl – pickle file of paper abstract data (each is a sequence of word ids representing paper abstract)

Part I: Word2Vec (20 pts)
P-I-1 Use word embedding model to generate pre-trained word embeddings of paper abstract. Finish word2vec function in word2vec.py. (Hint: Use Word2Vec in Gensim library: https://radimrehurek.com/gensim/). Use Skip-gram model, window size = 5, embedding dimension = 128, generate word_embeddings.txt file for Part II.

Part II: RNN for Text Classification (50 pts)
Train/test datasets are ready in load_text_data function in trnn.utils.py. Read this function to make sure you understand it.

P-II-1 (10 pts): Finish load_word_embed function in trnn.utils.py for loading pre-trained word embeddings in Part I.

PA-II-2 (20 pts): Finish Text_Encoder class for constructing RNN model in trnn.utils.py. RNN model detail: use LSTM cell. The input of RNN is the pre-trained word embeddings loaded in P-II-1. Input embedding (word embedding) dimension = 128, hidden state dimension = 64. After obtaining text embedding (i.e., the last hidden state of LSTM), use 2-layer MLP for text classification. The last 2 layers are with output hidden number = 32 and 5 (5 labels prediction as output) respectively. Use relu activation in the hidden layer and sigmoid activation in the output layer. Use default parameter initialization in Pytorch.

PA-II-3 (10 pts): Finish model_train function for text classification model (RNN) training (using train data) with Adam optimization in trnn_main.py. Set batch size = 20.

PA-II-4 (10 pts): Finish model_test function for model testing (using test data) in trnn_main.py.

Part III: Performance (30 pts)
PA-III-1 (10 pts): Run your code and report classification accuracy (average value) over test data (Hint: accuracy is over 70%), show screenshot of your result.

PA-III-2 (10 pts): Set epoch number = 100, plot the learning curve (test accuracy w.r.t. epoch number) using Adam optimization algorithm. You can choose suitable learning rate to make performance stable and good.

PA-III-3 (10 pts): There are two choices to obtain text embedding through RNN. One is using the last hidden state as output, the other is using the average of all hidden states (mean pooling) as output. Plot learning curves of these two methods and compare their performances.