AIT 726 Homework 3

In this assignment, you will build a recurrent neural network for Named Entity Recognition (NER) on CoNLL 2013 dataset. Your task is to classify words into 10 different classes: <pad>, O, B-ORG, B-PER, B-LOC, B-MISC, I-ORG, I-PER, I-LOC, I-MISC. These classes indicate whether words are part of a phrase referring to an organization, person, location, or miscellaneous. B tags indicates that word is at the beginning of the phrase, I tags indicate that the word is inside the phrase but not the first or the last word, O indicates it is outside any phrase (does not belong to any phrase).

Data: You can find training, test, and validation sets for this task on Blackboard. Your task is to build a named entity recognition system and tune parameters using training and validation data, and to evaluate the final model (after all development and tuning) with the test data.

Pre-processing: The data files contain text organized in columns. First column has the words to be classified, and last column shows the gold standard tag for each word. You can ignore the intermediate columns. To preprocess the data, lower case capitalized words (i.e., starts with a capital letter) but not all capital words (e.g., USA). Do not remove stopwords. Data is already separated by sentence and tokenized. Separate data by sentence. Once you know the maximum sentence length in the data, append 0s at the end of shorter sentences to make them match this max length. Set the tag for the 0s to <pad>.

Embeddings: Use pretrained word embeddings. You can download embeddings from: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit These are word2vec embeddings trained on the google news dataset. You will find 300 dimensional embedding vectors for 3 million words and phrases. Use them as your input vector representations.

Training: Build a RNN. Start with a vanilla RNN, with one layer of 256 hidden units, and a fully connected output layer using softmax as activation function. Use Adam optimizer, and cross-entropy for the loss function with learning rate 0.0001. Train with 2000 mini batches per epoch. Next, try a bidirectional RNN with the same settings. Then change the RNN unit to LSTM or GRUs, and experiment with different learning rates and batch sizes. Complete your system architecture, as well as hyperparameter and parameter tuning using training and validation data. Finally, for the best architecture among the 6 above, make the necessary modifications to update the embeddings along with the rest of the network. This is your final model.

Testing: Apply your trained models (RNN, bi-RNN, LSTM, bi-LSTM, GRU, bi-GRU, and best = 7 total) to test data. Save your output and results in a log file. Results should be in the following format:

Word Gold_Standard Prediction
SOCCER O O
- O O
MEXICO B-LOC B-LOC
GET O O

Evaluation: Run conlleval.py on your output. Use the get_result function to save the accuracy in the log file.

Documentation: Use the same documentation format from Assignment 1. Start all your files with a description of your code. Write short description of each function on top of it.

Deliverables: Submit a zip file named with student1[firstname initial][lastname]_student2[firstname initial][lastname]_[hw#].zip (i.e. student 1 jamie lee, student 2 kahyun lee: jlee_klee_hw1.zip).

Zip file should include: Your code(s), models, and log file. Give descriptive names to your models. E.g., rnn_258_softmax… Indicate in model name your best model that produces the best output in the log file.

**You can use jupyter notebook.