CS310 Natural Language Processing
Assignment 4: Long Short-Term Memory for Named Entity Recognition
Total points: 50 + (10 bonus)

Tasks

Train a bidirectional LSTM model on the CoNLL2003 English named entity recognition task set and evaluate its performance.

Submit

- The modified notebook files A4_ner.ipynb
- A write-up document in Word/PDF containing the F-1 score of the first 5 epochs of training, evaluated on the dev set; and the final F-1 score on the test set.
- For each bonus question you have completed, submit a stand-alone notebook. For example, A4_memm.ipynb, A4_crf.ipynb etc.

Requirements

1) (10 points) Data preprocessing.
   a) Load the train, dev, and test data; build vocabularies for words and labels (tags); defined a data loader that return batches.
      Note that it is recommended to convert all words to *lower cases*, because that is how words are stored in the glove pretrained embeddings.
   b) Load the pretrained embedding data to initialize the embedding layer in model.
      Note that you only need to load those words that have occurred in your vocabulary.
      The URL for the pretrained embedding is: https://nlp.stanford.edu/data/glove.6B.zip. 100-d should be sufficient for this task.

2) (20 points) Implement the "Level 0" (according to the lecture slide) local classifier model, with bi-LSTM architecture.
   a) Use `torch.nn.LSTM` module to implement the model.
   b) Adjust hyperparameters such as hidden size, layer numbers etc. as you like.
      Note: bi-directional and multi-layer network is highly recommended.

3) (20 points) Train, evaluate, and save.
   a) Use greedy search to obtain the predicted labels on test set, i.e., pick the highest probability label for each time step.
   b) Report the F-1 scores during the first 5 training epochs on dev set; and the final score on test set, with your own decision on total epoch #.
      Use the helper class MetricsHandler in metrics.py to compute the F1 score.

** Grading rubrics **

- If your model is implemented correctly and the training code can run without problem, then you get the full credits for step 1) and 2)
- If you achieve ≥70% F-1 score on the **test** set, you get full credits for step 3).
- If your F-1 score $0.5 < x < 0.7$, then you receive $(x - 0.5) \cdot 50 + 10$ points for step 3).
- If your F-1 score $x \leq 0.5$, then you receive 0 points for step 3.

****

** Grading rubrics for bonus tasks **

The three bonus tasks are independent of each other.

Some useful resources and existing implementations you can learn from:

- A good repo implementing and explaining biLSTM+CRF: https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Sequence-Labeling
- A beam search implementation in context of PyTorch and seq2seq: https://github.com/budzianowski/PyTorch-Beam-Search-Decoding
- The implementation of bi-LSTM+CRF from PyTorch official tutorial: https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html

DO NOT directly copy their code!

****

4) (**3 bonus points**) Implement the maximum entropy Markov model (MEMM).
    - *Hint*: Create an embedding layer for all the NER tags, and use the hidden states of previous tags for predicting the next one.

5) (**3 bonus points**) Implement beam search for decoding at testing time.
    - Compare its performance (F-1 score) with step 3).

6) (**4 bonus points**) Implement conditional random field with Viterbi algorithm (for training and decoding).
    - Compare its performance (F-1 score) with step 3).