CS310 Natural Language Processing - Assignment 3: Recurrent Neural Networks – Language Modeling and Named Entity Recognition
Total points: 60 + (15 bonus)

Tasks

1) Train a vanilla RNN language model on 《论语》and evaluate its perplexity.
2) Train a bidirectional LSTM model on the CoNLL2003 English named entity recognition task set and evaluate its performance.

Submit

- The modified notebook files A3_rnn-lm.ipynb and A3-ner.ipynb.
- A write-up document in Word/PDF reporting your results in Task 1-3 and Task 2-3.
- For each bonus question you have done, submit a stand-alone notebook. For example, A3_HMM.ipynb, A3_CRF.ipynb etc.

Requirements

Task 1 - LM (30 points):

1) (5 points) Data preprocessing.
2) (15 points) Model implementation.
   a) Use `torch.nn.RNN` module
   b) Do NOT use bidirectional network; multi-layer is fine.
3) (10 points) Evaluation and extended experiment.
   a) Report perplexity on *training* set, as the dataset is very small.
   b) Generate some sentences; quality is not graded.
   c) Compare the perplexity on two conditions: randomly initialized embeddings vs. with pretrained embeddings (from A2).

Task 2 - NER (30 points + **15 bonus points**):

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 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. It includes dimension 50, 100, 200, and 300. 100-d should be sufficient.
2) (10 points) Implement the "level 1" sequential classifier model, with bi-LSTM architecture.
   a) Use `torch.nn.LSTM` module.
   b) Adjust hyperparameters such as hidden size, layer numbers etc. as you like. **Note**: bi-directional and multi-layer network is highly recommended.
3) (10 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 F-1 score on test set. **Note** that $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

c) It is not required, but you can add L2 regularization term to increase test performance.

** 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% (and above) F-1 score on the **test** set, you get full credits for step 3).
- If your F-1 score $x > 0.5$, then you receive $\left\lfloor \frac{x}{0.7} \cdot 10 \right\rfloor + 0.5$ points for step 3).
- If your F-1 score $x < 0.5$, 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).
   - Create an embedding layer for all the labels (tags).

5) (**4 bonus points**) Implement beam search for decoding at testing time.
   - Compare its performance (F-1 score) with step 3).

6) (**8 bonus points**) Implement conditional random field with Viterbi algorithm (for training and decoding).
   - Compare its performance (F-1 score) with step 3).