ASSIGNMENT 2: NAMED ENTITY RECOGNITION

Motivation: The motivation of this assignment is to get practice with sequence labeling tasks such as Named Entity Recognition. You are free to experiment with the HMM and/or CRF models as well as BiLSTM-based or other neural architectures.

Problem Statement: The goal of the assignment is to build an NER system for diseases and treatments. The input of the code will be a set of tokenized sentences and the output will be a label for each token in the sentence. Labels can be D, T or O signifying disease, treatment or other.

Training Data: We are sharing a training dataset of labeled sentences. The format of each line in the training dataset is "token label". There is one token per line followed by a space and its label. Blank lines indicate the end of a sentence. It has a total of 3655 sentences.

The Task: You need to write a sequence tagger that labels the given sentences in a tokenized test file. The tokenized test file follows the same format as training except that it does not have the final label in the input. Your output should label the test file in the same format as the training data.

You can start out by running a simple sequence tagger using a CRF or using a Bi-LSTM. Then, explore combinations of Bi-LSTM with CRF. Feel free to add features.

Here are some suggestions on features:

1. Try features from lower level syntactic processing like POS tagging or shallow chunking.
2. Try features that try to assign a semantic label to the current token. A well-known generic ontology is Wordnet. Another famous medical ontology is MESH. Features based on these ontologies might help.
3. You may use existing word embeddings as features. You may also train your own word embeddings using unlabeled data. I have collected some sample in-domain unlabeled data here.
4. You may define word shape features and many of the other features we discussed in class.
5. Your idea here…

You may also experiment with the order of the Markov Chain in CRF model, but you might need to do more coding for that.

Typically, to improve your baseline system, it is recommended that you perform error analysis on a subset of data and think about what additional knowledge could help the classifier the most. That will guide you in picking the next feature to add or modification to make in the architecture.

What to submit?

1. Submit your best code by Saturday, 21 April 2018, 11:55 PM. The code should **not** need to train again. You should submit only the testing code, after the models have been trained. That is, you should not need to access the training data anymore.

   Submit your code is in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" the following files are produced in the present working directory:

   run.sh
   writeup.txt

   You will be penalized if your submission does not conform to this requirement.

   Your code will be run as ./run.sh inputfile.txt outputfile.txt. The outputfile.txt should have the same number of lines as inputfile.txt. And it should have two additional characters per token line (space and labeling). Here is a format checker. Make sure your code passes format checker before final submission.

2. The writeup.txt should have first line that mentions names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
   In the second line, you need to provide a link that has your training code.
   After this you are welcome to write something about your code, though this is not necessary.

Evaluation Criteria

This assignment is worth 10 points. The evaluation metric will be macro-F score, averaged over labels D and T.

What is allowed? What is not?

1. The assignment is to be done individually.
2. You must use PyTorch for this assignment.
3. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Feel free to search the Web for papers or other websites describing how to build named entity recognizers. Cite the references in your writeup. However, you should not use (or read) other people's NER code.
5. You can use any pre-existing ML softwares for your code. Popular examples include Mallet (http://mallet.cs.umass.edu/fst.php), Python Scikit (http://scikit-learn.org/stable/). However, if

you include the other code, use a different directory and don't mix your code with pre-existing code.

6. Your code will be automatically evaluated. You get a zero if it is does not conform to output guidelines. Make sure it satisfies the format checker before you submit.