Ling/CSC 439/539: Assignment #3 (75 pts)
(Graduate students)

Due by 11:59 P.M., October 29
(upload all materials to D2L)

Last modified on: 9/23/2017

Requirements for the submission

You must submit code and a written report for this assignment. The code and report must follow these requirements:

1. Each programming question must be answered through code that is **executed with a single command** in the terminal. Please include one command for each of the requirements stated below in the assignment. Submissions that must be run through an IDE (e.g., Eclipse, IntelliJ, etc.) are not accepted.
2. Similarly, if your code requires compilation (e.g., it is written in Java), **a single command line for compiling the code must be provided**.
3. If your code requires certain dependencies (e.g., specific libraries, version of the Python language), these have to be clearly stated with instructions for installation.
4. Your report must include **clear instructions for all the above issues**.
5. The code for this assignment may use libraries such as numpy for the linear algebra necessary in the assignment. However **you cannot use any other NLP/ML libraries**, unless explicitly stated in the corresponding problem.

Points will be taken off if the above requirements are not met. Additionally, your code must **compile** (if required by the programming language), **run**, and **produce the correct output**. Points will be taken off in any of these issues are violated.

Data

In this assignment you will train several part-of-speech (POS) taggers using the `PTBSmall` dataset (available in D2L). This dataset contains sentences with words labeled with Penn Treebank POS tags. The sentences are separated by one empty line. Each word is stored on an individual line, where the first token is the word itself, and the second is its POS tag. For example:

```
Pierre  NNP
Vinken  NNP
,       ,
61      CD
years   NNS
old     JJ
,       ,
will    MD
join    VB
the     DT
board   NN
as      IN
a       DT
nonexecutive    JJ
director        NN
Nov.    NNP
29      CD
.       .

Mr.     NNP
Vinken  NNP
is      VBZ
chairman        NN
of      IN
...
```

The dataset is split in 3 files:

- `train.tagged` – The partition to be used for the training of your POS tagger;
- `dev.tagged` – The partition to be used for the tuning of hyper parameters; and
- `test.tagged` – The testing partition to be used solely for the evaluation of your algorithm.

Problem 1 (25 points)

Implement a Markov model (MM) POS tagger using bigrams. For this problem, do not use smoothing of bigram probabilities or special treatment of unknown words. At inference, using the greedy heuristic: for each word, pick the tag with the highest probability as you traverse the text left to right.

Train your algorithm using only the training partition. Include functionality to save the model learned to disk (after training), and to load a pre-trained model (before testing). What is the performance of this algorithm on the testing partition? Report two numbers: overall accuracy, and accuracy just for the unknown words, i.e., words that appear in the testing partition, but not in training.

Your submission for this problem must contain:

- Code to train and evaluate your POS tagger. In the report, include: (a) a single command line to train the algorithm and save the resulting model; and (b) a single command line to run the algorithm on a test partition using a pre-trained model.
- Include your best model in the submission, so the instructors can run the algorithm on a test partition without retraining.
- Include the accuracy numbers in the written report.

Problem 2 (10 points)

Use one of the smoothing techniques covered in class to improve the robustness of your bigram probabilities. How does the performance of your algorithm change after adding smoothing?

Include in your submission the same materials as above, but adapted for this improved algorithm.

Problem 3 (10 points)

Replace the greedy inference heuristic with the Viterbi algorithm. How does the performance of your algorithm change after adding this optimal inference algorithm?

Include in your submission the same materials as above, but adapted for this improved algorithm.

Problem 4 (20 points)

Implement a POS tagger using a recurrent neural network (RNN) such as long short-term memory (LSTM) or gated recurrent units (GRU). I recommend you use DyNet for this, but if you are more familiar with other software (Keras, PyTorch, etc.) you are welcome to use those. Initialize your POS tagger with pre-trained word embeddings. For example, such embeddings can be downloaded from the GloVe page (`https://nlp.stanford.edu/projects/glove/`) or from Omer Levy's page (`https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/`). How does this neural network algorithm compare against your MM one? That is, what is its performance on all words, and just on the unknown ones? How much longer does it need to train?

Include in your submission the same materials as above, but adapted for the RNN algorithm.

Problem 5 (10 points)

Read the article "Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?" by Chris Manning. Summarize:

1. The methods and performance behind the current state-of-the-art in POS tagging as of the article's writing (you are free to use additional references, if you'd like, but remember to cite them),
2. The central challenges in increasing (and evaluating) performance toward 100%, and
3. The methods that Manning suggests will increase POS tagging accuracy.

Your summary should be single-spaced, using a font no-larger than 12pt, and no more than 1 page. You will need to be succinct, but informative (think conference abstracts). The summary will be evaluated on its accuracy, completeness/thoroughness, clarity, and insightfulness.