CSCI 3907/6907: Introduction to Statistical NLP

Assignment #2

Instructions:

- ✓ This assignment is due on **Tuesday October 8, 2019, by 11:59 pm**. Refer to course information about late submission policy. Late days are counted from 00:00 a.m. onwards.
- ✓ Submit your solution in a zipped folder through blackboard.
- ✓ All code must compile and run to receive credit for coding parts. We won't do any debugging. You can code in Perl, Python, Java, or C++. **Perl/Python** are recommended.
- ✓ Include brief instructions on how to run your scripts.
- ✓ Include citations for any online resources used or group discussions.

Parts-of-Speech Tagging and Hidden Markov Models

Q1. [20 points] Given the following formal description of a Hidden Markov Model, M={Q, Σ, A, O, q_0}

$Q=\{q_0, q_1, q_2\}$
Start state: q_0
$\Sigma = \{a,b,c,d\}$
The transition probabilities matrix A:

	q_1	**q_2**
q_0	0.6	0.4
q_1	0.5	0.5
q_2	0.3	0.7

The output probabilities matrix O:

	a	**b**	**c**	**d**
q_1	0.4	0.3	0.2	0.1
q_2	0.3	0.2	0.3	0.2

a. Draw the transition diagram for M.
b. Use the forward algorithm to manually compute the probability of the string *bad*. Show all your work.
c. Use the Viterbi algorithm to manually compute the most likely hidden state sequence for the word *bad*. Show all your work, and report both the state sequence and its probability.

Q2. Hidden Markov Models for parts-of-speech tagging:

Note: This part requires coding. Code must readily compile and run on SEAS Linux machines for full credit. Use the following files for training/testing:

`brown.train.tagged`	: Tagged training data.
`brown.test.raw`	: Raw test data that you need to tag.
`brown.test.tagged`	: Tagged test data for evaluation.

Sentences in these files are tokenized but not lowercased.

a. [10 points] Implement a majority-class baseline for the Brown corpus. For each word in the test set, assign the tag that is most frequently associated with that word in the training corpus. Tag out-of-vocabulary words as NN. Report the overall accuracy of this baseline on the test set.

b. [Graduate students only – 5 points] design and implement some transformation rules to improve the accuracy of the majority-class baseline. You should implement a minimum of 5 rules and no more than 15 rules. You may use the training set for tuning. For example, you may want to view the confusion matrix (of the training set) and think of rules that would fix the most common errors. You should **not** use the test set for tuning the rules. Report the overall accuracy of your model on the test set. (The submission with the highest accuracy in this part will receive 5 bonus points!)

c. [25 points] Implement an HMM Tagger for Parts-of-speech tagging.

    i. You will need a training script that reads the training corpus and calculates transition and output probabilities. Note that output probabilities for OOV words will be zero for all tags, which means the algorithm will fail to tag any sentence that includes OOV words in the test set. You can decide how to handle such cases (one option is to set a fixed small count for OOV words given any tag).
    ii. Implement the Viterbi algorithm that calculates the most likely tag sequence for each test sentence given a trained HMM model from part (i).
    iii. Evaluate the performance of your HMM tagger by reporting the overall accuracy on the test set.

d. [Graduate students only – 10 points] Implement beam search to speed up Viterbi decoding. Test your implementation with b=10 and report the accuracy.