Instructor: Prof. Lu Wang

Assignment 1 - Natural Language Processing (CS 4120/6120)

Deadline: 3pm, February 6, 2018

For the programming questions, you can use Python (preferred), Java, or C/C++. You also need to include a README file with detailed instructions on how to run the code and commands. Failure to provide a README will result in a deduction of points.
Violation of the academic integrity policy is strictly prohibited, and any plausible case will reported once found. This assignment has to be done individually. If you discuss the solution with others, you should indicate their names in your submission.

Question 1. Naive Bayes for Text Categorization (10 points)

Given the following short documents, each labeled with a genre (class):

1. son, sweet, funeral, love, dagger : **tragedy**
2. marriage, husband, stumble, dagger : **comedy**
3. funeral, dagger, son, bliss, marriage : **tragedy**
4. son, bliss, husband, sweet, stumble : **comedy**
5. husband, sweet, stumble, funeral : **tragedy**
6. funeral, bliss, marriage, love : **comedy**

And a new document D: bliss, son, stumble, marriage

Compute the most likely class for D. Assume a Naive Bayes classifier and use add-lambda smoothing (with lambda = 0.1) for the likelihoods.

Question 2. Word Sense Disambiguation (10 points)

Given the following sentences:
Banks tend to charge fees for cashier's checks. One can check this policy with a clerk at the counter of a local bank. One can also go to the counter to cash one such check.

2.1 [5 points] Using WordNet (`http://wordnetweb.princeton.edu/perl/webwn`), determine the total number of senses for each open class word.

2.2 [5 points] For each sentence, determine how many distinct combinations of senses exist using WordNet (`http://wordnetweb.princeton.edu/perl/webwn`).

Question 3. Language Modeling (30 points)

3.1 [10 points] Implement a 5-gram character language model from the following English training data:
`http://www.nltk.org/nltk_data/packages/corpora/gutenberg.zip`
Remove blank lines from each file. Replace newline characters with spaces, and remove duplicate spaces. Across all files in the directory (counted together), report the 4-gram and 5-gram character counts.

3.2 [10 points] The test dataset is given here:
`https://www.dropbox.com/s/hpcsp1ifwiwaixg/SOC_new.zip?dl=0`
Remove blank lines from the test data, replace newline characters with spaces, and remove duplicate spaces.

Take the file that is named "03302_02.txt". This is an English file. Report the average perplexity (per character) for this file. Use add-lambda smoothing (with lambda = 0.1).

3.3 [10 points] Some of the files in the test dataset are not English files. Identify them as follows: For each file in the directory of test data, calculate the perplexity. Using these scores, report the names and scores of the three files with the highest perplexity.

Question 4. POS Tagging - HMM (50 points)

The training dataset is a subset of the Brown corpus, where each file contains sentences of tokenized words followed by POS tags, and where each line contains one sentence:
`https://www.dropbox.com/s/6y365hoxd7huzbm/browncopy.zip?dl=0`

The test dataset (which is another subset of the Brown corpus, containing tokenized words but no tags) is the following:
`https://www.dropbox.com/s/hk6uk5amkf6ubww/humor.txt?dl=0`

Implement a part-of-speech tagger using a bigram HMM. Given an observation sequence of n words w_1^n and a set of hidden states (POS tags) t_1^n, choose the most probable sequence of tags. The following equation gives an intuition behind the Viterbi algorithm for decoding (eq. **10.9** in book - `http://web.stanford.edu/~jurafsky/slp3/10.pdf`).

$$\hat{t}_1^n = \underset{t_1^n}{argmax}\, P(t_1^n, w_1^n) \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} \overbrace{P(w_i, t_i)}^{emission} \overbrace{P(t_i, t_{i-1})}^{transition}$$

4.1 [10 points] Obtain frequency counts from the collection of all the training files (counted together). You will need the following types of frequency counts: word/tag counts, tag unigram counts, and tag bigram counts. Let's denote these by $C(w_i, t_i)$, $C(t_i)$, and $C(t_{i-1}, t_i)$, respectively. Report these quantities.

To obtain the tag unigram counts and the tag bigram counts, you will need to separate out each tag from its word. For each sentence found in the training data, add a start token and an end token to the beginning and the end of the sentence.

4.2 [5 points] A transition probability is the probability of a tag given its previous tag. Calculate transition probabilities of the training set using the following equation:

$$P(t_i, t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

4.3 [5 points] An emission probability is the probability of a given word being associated with a given tag. Calculate emission probabilities of the training set using the following:

$$P(w_i, t_i) = \frac{C(w_i, t_i)}{C(t_i)}$$

4.4 [10 points] Generate 5 random sentences using HMM. Output each sentence (with the POS tags) and its probability of being generated.

4.5 [20 points] Split the test data into sentences. For each word in the test dataset, derive the most probable tag sequence using the Viterbi algorithm; pseudocode can be found in the textbook (`http://web.stanford.edu/~jurafsky/slp3/10.pdf`) under **Figure 10.8**.
For each word, output the tag derived using the Viterbi algorithm in the following format (where each line contains no more than one pair):

<sentence ID=1>
word, tag

word, tag
...
word, tag
<EOS>
<sentence ID=2>
word, tag
word, tag
...
word, tag
<EOS>
...
<EOS>

- Submit your code, a README.txt file explaining how to run your code, and your output file with the tag predictions in the format specified above.