Natural Language Processing [CS4120/CS6120] Assignment 1

Instructor: Professor Lu Wang

Deadline: October 10th at 11:59pm on Blackboard

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 be 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.

1 Naive Bayes for Text Categorization (10 points)

Given the following short documents (1-8), each labeled with a class:

1. banana, carrot, cucumber, pea : **vegetable**
2. school, pea, rose, lily, basket : **flower**
3. banana, pea, potato, lotus : **vegetable**
4. hibiscus, grape, potato, mango, apple : **fruit**
5. hibiscus, lotus, lily, apple, banana : **fruit**
6. rose, hibiscus, banana, rose : **flower**
7. rose, rose, rose, cucumber : **flower**
8. carrot, mango, grape, rose : **fruit**

And documents:

1. D1 : rose, lily, apple, carrot
2. D2 : pea, pea, lotus, grape

Compute the most likely class for D1 and D2. Assume a Naive Bayes classifier and use add-lambda smoothing (with lambda = 0.1) for the likelihood.

2 Word Sense Disambiguation (10 points)

Given the following sentences:
In the cold weather, they started to the city. They were least worried protecting themselves against the common cold. After she signed the agreement, a cold chill crept up her spine. "Chill, its not that serious," her husband assured and left to deposit cash at the bank.

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`).

3 Language Modeling (30 points)

3.1 [10 points]

Implement a 4-gram language model from the following English training data: `http://www.nltk.org/nltk_data/packages/corpora/gutenberg.zip`
Across all files in the directory (counted together), report the 4-gram counts.
Refer instructions below.

3.2 [20 points]

For the given datasets:
News Articles:
`https://drive.google.com/open?id=14veC-71fA_pE-kR8GUz8xwV37NehPrA2`
IMDB Movie Reviews:
`https://drive.google.com/open?id=1AaErcFL4H30LVdiObQoY4w7MIVzPCVXq`
Calculate the average perplexity for each dataset. Use lambda smoothing (with lambda = 0.1)
For preprocessing the data, see instructions below.

Data preprocessing instructions for 3.1 and 3.2:

1. Remove blank lines from each file.
2. Replace newline characters with spaces.
3. Remove duplicate spaces.

4. Tokenize each document (you can use NLTK, `https://www.nltk.org/`).

5. Replace words in train set that appear ≤ 5 times as "UNK".

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/sh/havbkrjqzu9kpv6/AABVY0xRUvu-AO2TyftQUwCEa?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/5j62js3pa0z6z9e/science_sample.txt?dl=0`
Information regarding the categories of the dataset can be found at:
`https://www.dropbox.com/s/ujnz04e62e9603j/CONTENTS.txt?dl=0`

Your task is to implement a part-of-speech tagger using a bigram HMM. Given an observation sequence of n words w_1^n, choose the most probable sequence of POS tags t_1^n.

[Note: During training, for a word to be counted as unknown, the frequency of word in training set should not exceed a threshold (e.g. 5). You can pick a threshold based on your algorithm design.]

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. Lets 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:

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

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]

For each word in the test dataset, derive the most probable tag sequence using the Viterbi algorithm; pseudo-code can be found in the textbook `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf` under **Figure 8.5**. 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>

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