Antoine Dangeard
260962884

ECSE 526: Assignment 2 Report

Explanation of code for Part 1:

The entirety of the code for Part 1 is contained inside the file “generate_text.py”. To import the given .txt data files into my python code, I made a script “data.py” in the “data/” folder which opens, reads, and returns the relevant information from the data files. In generate_text.py, I made three functions with which I can generate “likely” sentences:

- **sampleWordFromProbabilities**: This function takes in a list of words and a list of probabilities (where the index of a word has probability at the same index in the probabilities array), and randomly samples a word from the list using the probability distribution provided. First, the probabilities array is normalized to sum to 1, and then I used numpy.random.choice to select a random word, before returning it.
- **getNextWord**: This function takes in a list of words (i.e. the sentence that has been generated so far), and returns the most likely next word in that sentence. To do this, it uses nested try/except blocks to “fall back” to a probability distribution that uses less previous words (i.e. if trigram fails, it tries to use bigram, and if that fails then it uses unigram) until one succeeds. At each “try” block, it calls the sampleWordFromProbabilities function described above with the probabilities fetched from the trigram, bigram, or unigram distributions. Finally, it returns the word that is returned by this function.
- **generateSentence**: This function takes in a seed value (to set the seed on numpy.random, so that results are reproducible), and returns the sentence which was generated using that seed. First, it sets the numpy.random seed to the seed integer passed in as an argument, then calls getNextWord in a while loop, storing the words that are generated at each step, until the word returned by the function is “</s>”. Finally, it constructs the sentence string from the individual word strings and returns that string.

Examples of sentences generated for Part 1:

<s> He thought much of his opinion . </s>
<s> Anne recollected with pleasure . </s>
<s> But , my dear ," said Emma , I cannot imagine how I do not think that he was a little , or a fool . </s>
<s> They had not a creature in the letter , could not help laughing . </s>
<s> The charm of earlier youth ; and as a friend to Harriet . </s>

Explanation of code for Part 2:

The code for Part 2 uses the same “data.py” file to import and use the provided data files. The main file for this part is “correct_text.py”, which contains all the code used to correct the sentences provided for the assignment. The functions I implemented for this part are:

- **probObservationGivenState**: This function takes in an observation string (i.e. a word from the sentence containing a typo) and a state (a word from the vocabulary), and returns the probability P(E | X) as it is defined in the assignment. It uses the function calculate_levenshtein_distance which I copied from Stack Overflow (exact source provided below).
- **correctSentence**: This function takes in a sentence (an array of word strings possibly containing typos) and returns the corrected sentence. It uses the Viterbi algorithm, which works (in pseudocode) as follows:
    1. Create two arrays, one for storing probabilities and one for storing “backpointers” (for each word, a reference to the word that is before it that gave it the maximum probability), both of which are of size ***N * M*** (where ***N*** is the number of words in the vocabulary, 11,469, and ***M*** is the number of words in the input sentence).
    2. Initialize the probabilities array so that each row ***n*** in column ***1*** has the value $P(state(1) = word(n) \mid state(0) = <s>) \cdot P(observation(1) \mid state(1) = word(n))$
    3. Iterate over the rest of the columns in the probabilities array (we’ll call it ***PA*** for the formula):
        a. For each row ***n*** in column ***m*** set ***PA[n,m]*** to
$$max_i\, (P(state(m) = word(n) \mid state(m-1) = word(i)) \cdot PA[i, m-1]) \cdot P(observation(m) \mid state(m) = word(n)))$$
        b. Save the index of the word in the previous column which maximized the probability of the word in the current column (save it to the backpointers array).
    4. Find the index ***n*** which has the highest value in ***PA[n,M]***. Recursively use the backpointers array to find the word which precedes that word, reconstructing the sentence in reverse.
    5. Return the array of words gotten from following the highest probability through the backpointers array.

Sentences before and after correction for Part 2:

I think hat twelve thousand pounds
CORRECTED: <s> I think at twelve thousand pounds
she haf heard them
CORRECTED: <s> she had heard them
She was ulreedy quit live
CORRECTED: <s> She was already quite like
John Knightly wasn't hard at work
CORRECTED: <s> John Knightley was hard at work
he said nit word by
CORRECTED: <s> he said it would be

Source of edit distance code:

https://codereview.stackexchange.com/questions/217065/calculate-levenshtein-distance-between-two-strings-in-python/217074#217074