Matthew Oakley
#101697544

CSCI 3202: Assignment #8

NOTE:
When attempting to run my code, you will be prompted to enter a sentence, please do so using only words/punctuation provided in *penntree.tag*. I've printed out statements in order for you to determine where my program is at in execution. Please note that calculating emission probabilities from the training set will take some time (30 - 45 seconds).

Purpose

The purpose of this assignment was to get experience with hidden markov models (HMMs) and see its application regarding speech patterns/word identification. This was accomplished by training our program over a set of many sentences in order to calculate, with high probability, what the given words/tags are associated with an input sentence.

Procedure

Transition Probabilities

After parsing the training-set file with proper start-of-sentence and end-of-sentence tags, the next step was to calculate transition probabilities from the training-set. This was accomplished by creating a dictionary of Tag objects. Each Tag had corresponding tag_name, total, and followed_by (number of times Tag_i is followed by Tag_j) attributes. The probabilities were then calculated by performing count(TAG_i TAG_j) / count(TAG_j) for each word in each sentence in the training-set.

Emission Probabilities

Emission probabilities were calculated from the training-set by creating a dictionary of dictionary corresponding to each of the tags present in the training-set. Each tag in the dictionary was then populated by the number of occurrences of unique words associated with that specific tag. The probabilities were then calculated by performing count(word && TAG) / count (TAG) for each word/tag association in the training set.

Viterbi Algorithm

Once the transition probabilities and emission probabilities were calculated from the training set, they were passed into the Viterbi algorithm along with the user-inputted sentence, initial probability, list of observations from the user-inputted sentences, and the set of possible tags. A 2D matrix is constructed in the Viterbi algorithm in order to backtrack and find the maximum probability of a state associated with a given observation. The highest-probability tags are then printed out with respect to the structure of the original, user-inputted sentence.

Matthew Oakley
#101697544

Data

In order to get an understanding for 'most-likely' probabilities, we needed to train our program using a data set. I used the *penntree.tag* file which was provided on Moodle in order to train my program. The only pre-processing done aside from opening the file was appending start-of-sentence tags ("SSSS") and end-of-sentence tags ("EEEE") to each sentence provided in the training set. Each individual word/tag combination were stored in a list and were then passed to the transition/emission probability helper functions.

Results

My program is able to correctly predict corresponding word/tag combinations depending on the structure of the inputted sentence; it was able to do so for every test-sentence I provided. I was not able to find a test-sentence that didn't work. My test-sentences and program-output are provided below:

Provided Test-Sentences:

Sentence: This is a sentence.
Tags: DT VBZ DT NN .

Sentence: Can a can can a can?
Tags: MD DT NN MD DT NN .

Sentence: This might produce a result if the system works well.
Tags: DT MD VB DT NN IN DT NN VBZ RB .

Sentence: Can a can move a can?
Tags: MD DT MD VB DT NN .

Sentence: Can you walk the walk and talk the talk?
Tags: MD PRP VBP DT NN CC VB DT NN .

Self-Generated Test-Sentences:

Sentence: I like to go the park.
Tags: PRP VBP TO VB TO DT NN .

Sentence: My dad makes me cry sometimes.
Tags: PRP$ NN VBZ PRP VBP RB .

Sentence: Is life even worth living?
Tags: NNP NN RB JJ NN .