Assignment 2

Due on April 4, 2019 (23:59:59)

Click here to accept your Assignment 2

Introduction

Part-of-speech (PoS) tagging is the task of assigning syntactic categories to words in a given sentence according to their syntactic roles in that context, such as a noun, adjective, verb etc. A PoS tagger takes a sentence as input and generates a word/tag tuples as output:

Example An example tagging:
Bunu zaten biliyordum. (I have already known that.)
*Bunu/**Pron** zaten/**Adv** biliyordum/**Verb** ./**Punc***

In this assignment, you will implement a PoS tagger using Hidden Markov Models (HMMs). Therefore, you will practice HMMs and Viterbi algorithm in this assignment.

0.1 Task 1: Build a Bigram Hidden Markov Model (HMM)

We need a set of observations and a set of possible hidden states to model any problem using HMMs. As for the PoS tagging problem, the observations are the words in a given sentence. The hidden states are the POS tags of the words, which will be predicted by the model.

You will implement a bigram HMM tagger for thisk task. A bigram HMM consists of three components:

1. **Initial probability** $p(t_1)$ is defined as the probability of the tag t_1 of the first word w_1 that a sentence begins with. For example, $p(VP)$ is the probability of a sentence to begin with a VP (Verb Phrase) tag.
2. **Transition probability** $p(t_{i+1}|t_i)$ is defined as the probability of seeing the tag t_{i+1} following the tag t_i. For example, $p(VP|NP)$ is the probability of assigning VP tag to the current word given that the previous tag is NP (Noun Phrase).
3. **Emission probability** $p(w_i|t_i)$ is the probability of generating the word w_i from the hidden state t_i. For example, $p(elma|NP)$ is the probability of observing the word *elma* from the hidden state tagged with NP.

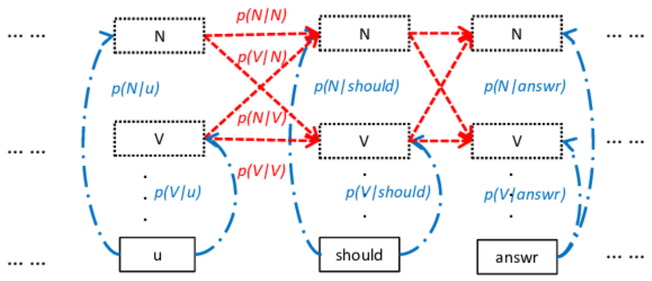

Figure 1: An illustration of the Viterbi algorithm.

To build your own hidden Markov Model, you must calculate the initial, transition, and emission probabilities by using the given training data. You may use various preprocesssing steps on the dataset (lowercasing the tokens, stemming etc.). You can also use various techniques for unknown words. You can try out different methods to improve your model.

0.2 Task 2: Viterbi Algorithm

Once you build your HMM, you will use the model to predict the PoS tags in a given raw text that does not have the correct PoS tags. To this end, you will implement the Viterbi algorithm that will assign the most probable PoS tag sequence $t_1 \cdots t_m$ for a given a word sequence $w_1 \cdots w_m$ as follows:

$$\underset{t_1 \cdots t_m}{argmax} p(t_1 \cdots t_m | w_1 \cdots w_m) \tag{1}$$

The Viterbi algorithm is performed in two steps:

1. You will compute the probability of the most likely tag sequence.
2. You will trace the back pointers to find the most likely tag sequence from the end of the sentence till the beginning.

An illustration of the Viterbi algorithm is given in Figure 1. Your output format will be the same as the input text format.

0.3 Task 3: Evaluation

Your program will compute the accuracy of the PoS tagger. The accuracy is the ratio of the correctly assigned tags to the total number of words in the test data:

$$A(W) = \frac{\#of\_correct\_found\_tags}{\#of\_total\_words} \tag{2}$$

Dataset

METU-Treebank is a Turkish dataset that is built from manually collected newspapers, journal issues, and books. The corpus involves 5659 sentences. You will take %70 of the data as training (for Task 1) and rest of it as test data (for Task 2 and Task 3). In other words, the first 3960 lines will be used for training and the last 1699 lines will be used for testing.

Submit

You are required to submit all your code. You will implement the assignment in **Python** (Python 3.5). You will submit a report in latex format template). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. Give the answers of all questions raised in the definition of the assignment above. You can include pseudocode or figures to highlight or clarify certain aspects of your solution.

- report.pdf
- code/ (directory containing all your codes as Python file .py)

CHALLENGE (For Bonus Points)

In addition to the default tasks defined above, you can improve your PoS tagger more to compete with other participants in the class. The accuracy of your PoS tagger is the main criteria to be compared with others. The participants in the class that obtain a higher tagging accuracy will deserve more points.

To enter the competition, you have to register kaggle with your department email account. The webpage of the challenge will be announced later. Top 5 assignments will earn extra points (10 points).

Grading

- Code (90 points): Task 1: 35, Task 2: 50, Task 3: 5
- Report (10 points)

  Note: Preparing a good report is important as well as the correctness of your solutions!

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.