CSCI 3202 – Introduction to Artificial Intelligence

Instructor: Hoenigman Assignment 8

Due: Sunday, December 4, 5pm

Part of Speech Tagger using HMM

This assignment replaces the link posted earlier on Moodle:
http://verbs.colorado.edu/~mahu0110/teaching/ling5832/5832-hw3.html

In this assignment, you will build an HMM to identify the part-of-speech tag for words in a sequence of words. Each tag will represent a part of speech tag, such as NN represents Noun and ADJ represents Adjective. The Penn tree bank data set consists of words and 45 possible tags, applied manually by linguists. You will apply the Viterbi algorithm to find the best sequence using the transition and emission probabilities you build from the data set.

Input Data

There is a file named pentree.tag is available on moodle. The data in the file is formatted as:

Word Tag

Word Tag

Word Tag

Word Tag

Word Tag

Word Tag

Word and tags are separated by a tab '\t'

There are thousands of sentences that are separated by an empty line in the file. If you have a look at the penntree.tag you can undertand the format. When you read in the data in this file, you need to place a start-of-sentence tag in when you read a new line. After each sentence you should append a end-of-sentence tag. In the original writeup, <s> was used as a start-of-sentence and </s> was used as end-of-sentence. You could also use tags such as SSSS for start and EEEE for end. After adding the start and end tags, your training data should look like

SSSS

This <Tag>

is <Tag>

a <Tag>

sentence <Tag>

EEEE

SSSS

This <Tag>

is <Tag>

a <Tag>

sentence <Tag>

EEEE

After we process the data after every sentence there should ‘EEEE’ and before every sentence there should be a word ‘SSSS’.

To pre-process all of the data into a list before calculating the emission and transition probabilities, your Python code should look something like:

```
with open(‘penntree.tag’, "r") as sentences:
    lines.append("SSSS")
    for line in sentences:
        if(line=="\n"):
         lines.append("EEEE")
         lines.append("SSSS")
        else:
         lines.append(line.rstrip())
lines.append("EEEE")
```

Algorithm

Viterbi Algorithm

The Viterbi algorithm needs the transition and emission probabilies calculated from the data set.

Transition probability

$P(Tag_i \mid Tag_j)$ should be calculated by the program.

For example P (DT|NN) will be calculated by *count (DT_i NN_j) / count (NN_j)*. You want to count up all of the instances in the data where DT is observed after NN and divide by the number of times that NN is observed.

Sanity check: After calculating transmission counts try to print the count of (DT_i NN_j). The answer should be 870. If you are getting some answer around 870 it is fine. You will get this answer only if you pre-process the data properly.

Hint: You can use dictionary of dictionary as the data structure to get the Transition counts

Emission probability

P (word | TAG) should be calculated by the program.

For example, P (word | TAG) is calculated by *count (word AND TAG) / count (TAG).*

You need to create a emission probability for the start- and end-of-sentence tags, 'SSSS' and 'EEEE'.

Sanity check : The count of [the DT] is 39517.

What Should the code output?

You should input a sentence and it should predict a tag for each word of a sentence. You must create a trellis with all these probabilities and the back traces in the Viterbi algorithm.

Hint: You can use a dictionary of dictionary of list as a data structure. The list will contain the probability and its best previous state. As there 45 tags, you should generate 47 states for every word given in the test sentence. There is a file on Moodle called *viterbi.py* that you might find useful.

Important note

To test your work, you can try the following sentences. These sentences are short. If you try to enter a very long sentence, you will get a floating point underflow error. You can solve this problem by using log probabilities, but that's not required for this assignment.

Test Sentences and Test outputs

Test sentence 1

This is a sentence.

For that sentence you must pad SSSS at the beginning and EEEE ar the end. That is

SSSS This is a sentence. EEEE

Output for the above sentence should be 'DT, 'VBZ,'DT','NN','.'

Test sentence 2

Can a can can a can?

After appending our start and end tags

SSSS Can a can can a can? EEEE

Output for the above sentence should be 'MD', 'DT', 'NN', 'MD', 'DT', 'NN', '.'

If you get these outputs then your program is completely fine. First can is MD and third can is NN. Depending on the context the tags changes.

Other test sentences

This might produce a result if the system works well.

Output: 'DT', 'MD', 'VB', 'DT', 'NN', 'IN', 'DT', 'NN', 'VBZ', 'RB', '.'

Can a can move a can?

Output: 'MD', 'DT', 'MD', 'VB', 'DT', 'NN', '.'

Can you walk the walk and talk the talk?

Output: 'MD', 'PRP', 'VBP', 'DT', 'NN', 'CC', 'VB', 'DT', 'NN', '.'

What should you submit?

You should submit the code and a short report describing your work. The report should contain the follows

Purpose:

Purpose should be clear. It need not be long but it should be clear.

Procedure:

Transition Probabilities

In this section explain the way you have generated the Transition probabilities and the data structure you used.

Emission Probabilities

In this section explain the way you have generated the Emission probabilities and the data structure you used.

Viterbi Algorithm

In this section explain about the algorithm and the data structure you used for storing the values in the matrix with its back pointer.

Data

Explain about the data you used (The Penn tree bank) , including the data included in the data set and any pre-processing you did to the data.

Results

Explain your observations, including the sentences that you tested and their corresponding outputs. Were there any sentences where your algorithm did not work?