HW02 Report on HMM with Greedy and Viterbi Decoding

-- By Ruijie Rao

This assignment gives you hands-on experience on using HMMs on part-ofspeech tagging. We will use the Wall Street Journal section of the Penn Treebank to build an HMM model for part-of-speech tagging. In the folder named data, there are three files: train, dev and test. In the files of train and dev, we provide you with the sentences with human-annotated part-of-speech tags. In the file of test, we provide only the raw sentences that you need to predict the part-of-speech tags. The data format is that, each line contains three items separated by the tab symbol '\t'. The first item is the index of the word in the sentence. The second item is the word type and the third item is the corresponding part-of-speech tag. There will be a blank line at the end of one sentence.

1. Vocabulary

In the first pass, I output the vocab using only <unk> tag for unknown words. After tweaking the threshold, I found 5 to generate the best accuracy scores. In the second pass, I significantly increased the accuracy by adding more pseudo-words including:

- <-unk> for unknown with hyphens
- <allcap-unk> for unknown with all capital letters
- <Title-unk> are for those with capital first letter

What is weird is that after using mor pseudo-words, the best threshold also decreases: allowing more words in the vocab while further raising the accuracy. I cannot figure out why, so if possible I would like someone to suggest an answer for me in the feedback.

--Before my optimization using pseudo-words--

What is the selected threshold for unknown words replacement?

- 5

What is the total size of your vocabulary?

- 11688

what is the total occurrences of the special token '< unk >' after replacement?

- 50296

--After my optimization using pseudo-words--

What is the selected threshold for unknown words replacement?

- 2

What is the total size of your vocabulary?

- 23183

what is the total occurrences of the special token '< unk >' after replacement?

- 20011 (including all unknown pseudo-words)

2.Model Learning

I approached greedy and viterbi in two different kind of ways: I transformed the dictionaries demanded in the task into matrices in Greedy decoding to make the operations more efficient using numpy mult; I went back to using the dictionaries to avoid the sparse spaces in the matrices during the multi-loop operations in the Viterbi Algorithm, which I assume would be a better choice than using matrices.

How many transition and emission parameters in your HMM?

- 1392 for transition dictionary
- 30362 for emission dictionary

3. Greedy Decoding with HMM

The optimization using pseudo-words was due to my exploration on the error set, specifically the unknown set. I found out that tho the whole dataset has an accuracy of 92%, the unknown set has only 45%. As a result, I sliced out the unknown set from the error set, and found that one of the biggest pattern was for hyphened words, which most of the time are JJ but almost always get predicted as NN. Combining with the lecture slide which mentioned pseudo-words, I added 3 more of those (mentioned in above section) and got a 60% for the unknown set. (From "worse than guess" to "a good guess")

--Before my optimization using pseudo-words--

What is the accuracy on the dev data?

- 92.2%

--After my optimization using pseudo-words--

What is the accuracy on the dev data?

- 94.1%

4.Viterbi Decoding with HMM

--Before my optimization using pseudo-words--

What is the accuracy on the dev data?

- 93.4%

--After my optimization using pseudo-words--

What is the accuracy on the dev data?

- 94.9%