Ander Swartz **NLP HW # 3 Report** 3/17/22

1.1

The total number of tokens, word types, and tag types was found to be 46469, 10586, and 21.

2.1

Using my Hidden Markov Model, the following transition probabilities were obtained:

("B-person","O")) = 0.009089463039970914

("B-person","B-person")) = 0.0

("I-person","B-person")) = 0.43875278396436523

("B-person","I-person")) = 0.0

("I-person","I-person")) = 0.08372093023255814

("O","I-person")) = 0.8883720930232558

2.2 After adding .1 to the numerator (number of instances of a given word and tag co-occurring) when calculating the emission probabilities, the following emission probabilities were obtained:

("God","B-person")) = 0.020267260579064587

("God","O")) = 0.00013861431135955642

("Lindsay","B-person")) = 0.026948775055679285

("Lindsay","O")) = 2.2723657599927286e-06

I found that when using Laplace smoothing, which involved not only adding a constant to the numerator of the but also by adding the size of the vocabulary (here the length of the word types) multiplied by the constant to the denominator of the relative frequency estimation.

This provided more accurate results on the development data set. Here are the same emission probabilities with the better Laplace smoothing:

("God","B-person")) = 0.0008164929768917082

("God","O")) = 0.00011008737383913688

("Lindsay","B-person")) = 0.0010883552333484368

("Lindsay","O")) = 1.8317366695363874e-07

After this, I used log probabilities. To get these results, remove the log() on lines 64 and 67.

2.3

The Viterbi algorithm provides the following predicted tags for the first 5 sentences of the dev set:

STOP O O

WHAT O O

YOU'RE O O

DOING O O

AND O O

GO O O

GET O O

#ExpelledMovieToNumberOne O O

ON O O

ITUNES B-other O

BECAUSE O O

IT'S O O

ONLY O O

2ND O O

!! O O

@camerondallas O O

Shs O O

RT O O

@Phil\_Heim O O

: O O

Safe O O

to O O

say O O

Super B-other O

Bowl I-other O

Sunday I-other O

is O O

my O O

favourite O O

holiday O O

of O O

the O O

year O O

RT O O

@shashiranjanttv O O

: O O

@shashiranjanttv O O

second O O

is O O

Bawana B-geo-loc O

constituency O O

on O O

Feb O O

4 O O

- O O

Final O O

I've O O

watched O O

all O O

my O O

dreams O O

' O O

episodes O O

so O O

I O O

decided O O

to O O

choose O O

another O O

series O O

! O O

It O O

may O O

be O O

inspired O O

from O O

the O O

movie O O

I'll O O

watch O O

it O O

now O O

! O O

#GoodNight O O

Final O O

grades O O

go O O

up O O

tomorrow O O

:( O O

Note that it predicts O too much here but the model does predict other tags.

2.4

My model’s final FB1 score on the dev set is 14.67. This can be obtained by running hmm.py as it currently is.

3.2

These are the results of evaluating the dev data on the perceptron after it is trained for 7 epochs on the training data. I found the FB1 to be around 14.5% before preprocessing the sentences to all be lowercase, which boosted it to what is shown below.

processed 16261 tokens with 661 phrases; found: 311 phrases; correct: 91.

accuracy: 92.66%; precision: 29.26%; recall: 13.77%; FB1: 18.72

company: precision: 22.58%; recall: 17.95%; FB1: 20.00 31

facility: precision: 3.85%; recall: 2.63%; FB1: 3.12 26

geo-loc: precision: 44.26%; recall: 23.28%; FB1: 30.51 61

movie: precision: 0.00%; recall: 0.00%; FB1: 0.00 18

musicartist: precision: 0.00%; recall: 0.00%; FB1: 0.00 5

other: precision: 44.44%; recall: 12.12%; FB1: 19.05 36

person: precision: 30.83%; recall: 21.64%; FB1: 25.43 120

product: precision: 12.50%; recall: 2.70%; FB1: 4.44 8

sportsteam: precision: 66.67%; recall: 2.86%; FB1: 5.48 3

tvshow: precision: 0.00%; recall: 0.00%; FB1: 0.00 3

3.3

First, I attempted to add a prefix feature to the get\_features function. I did this by first going through the training data and finding all prefixes of length two or three that occurred more than a certain amount of times. Then, when training the perceptron, the first two and three letters of each word were checked to see if they were considered a prefix. If so, the first two or three letters along with the corresponding tag for that word were added as a feature.This did not approved the accuracy of the model, which is likely because of mistakes during implementation.

Next, I attempted to use information about whether the word was capitalized. Previously, I had been normalizing every string to be lowercase, but I figured this was not taking advantage of all the information. Instead, I checked if the prevTag == “<BOS>”, which meant that the capitalization was grammatical and did not relate to named entity recognition, at which point I would make the word lowercase. Otherwise, if a capitalized word occurred in the middle of a phrase, it likely related to an entity. This change, like the previous change, made the model slightly better when compared to the base model after 1 epoch, but was not as strong as the base model at 7 epochs.

I also used the capitalization feature differently where I check to see if the first letter is capitalized, at which point I also add the lowercase version as a feature. I did this so that capitalization was still informative, but the lower case version of a word should also inform a capitazed word.

Lastly, I used the length of the word as a feature. I figured this would not be very effective, but since all the other features at first outpaced the base model at a small number of epochs, but then did not match the performance of the base model at larger epochs, I figured it was worth a try.

If we were allowed to change the score\_features function, I would have made it so that each of these features was weighted less than the main emission and transition ones. As I add more features, it makes sense that the model’s performance on testing data will decrease will a smaller number of epochs than it will with fewer features, since more features generally leads to faster overfitting.

3.4

The best version of my perceptron model, which is the base model using only the original features, obtains the same results as before on the dev data after being training for 7 epochs on the training data:

processed 16261 tokens with 661 phrases; found: 311 phrases; correct: 91.

accuracy: 92.66%; precision: 29.26%; recall: 13.77%; FB1: 18.72

company: precision: 22.58%; recall: 17.95%; FB1: 20.00 31

facility: precision: 3.85%; recall: 2.63%; FB1: 3.12 26

geo-loc: precision: 44.26%; recall: 23.28%; FB1: 30.51 61

movie: precision: 0.00%; recall: 0.00%; FB1: 0.00 18

musicartist: precision: 0.00%; recall: 0.00%; FB1: 0.00 5

other: precision: 44.44%; recall: 12.12%; FB1: 19.05 36

person: precision: 30.83%; recall: 21.64%; FB1: 25.43 120

product: precision: 12.50%; recall: 2.70%; FB1: 4.44 8

sportsteam: precision: 66.67%; recall: 2.86%; FB1: 5.48 3

tvshow: precision: 0.00%; recall: 0.00%; FB1: 0.00 3