CS6501 Natural Language Processing
Independent Project 2

Xin Nie, xn9vc

October 2018

1 Hidden Markov Model

1. Toy Problem

Table 1:

START	G	C	A	C	T	G	END
0	(-2.47, START)	(-4.21, H)	(-7.9, H)	(-10.9, H)	(-14.56, H)	(-17.39, L)	(-18.39, L)
0	(-3.64, START)	(-6.12, H)	(-7.27, H)	(-10.59, L)	(-13.33, L)	(-16.65, L)	

(a)

(b) Decode sequence is: $H - H - L - L - L - L$

2. POS Tagging

(a) K is defined as 5. Size of V is

```
In[3]: len(self.V)
Out[3]: 10591
```

(b) xn9vc-tprob.txt

(c) xn9vc-eprob.txt

(d) xn9vc-eprob-smoothed.txt and xn9vc-trpob-smoothed.txt

(e) Accuracy:

```
Accuracy of Viterbi Algorithm: 0.94056787443359
```

(f) xn9vc-viterbi.txt

(g) Alpha = [0.001, 0.005, 0.0001, 10]
Beta = [0.001, 0.005, 0.0001, 10]
Alpha and beta in tested pair are the following:

```
((0.001, 0.001), 0.9444358298444203)
((0.001, 0.005), 0.9444358298444203)
((0.001, 0.0001), 0.9444358298444203)
((0.001, 10), 0.9444272152443962)
((0.005, 0.001), 0.9445047466446133)
((0.005, 0.005), 0.9445047466446133)
((0.005, 0.0001), 0.9445047466446133)
((0.005, 10), 0.944478902844541)
((0.0001, 0.001), 0.9444358298444203)
((0.0001, 0.005), 0.9444358298444203)
((0.0001, 0.0001), 0.9444358298444203)
((0.0001, 10), 0.9444272152443962)
((10, 0.001), 0.9109681087507107)
((10, 0.005), 0.9109681087507107)
((10, 0.0001), 0.9109681087507107)
((10, 10), 0.9110542547509519)
```

Best alpha and beta can take 0.05, accuracy is 0.9445047466446133

(h) xn9vc-viterbi-tuned.txt

2 Conditional Random Fields

1. Features added:
   - token to upper case
   - first letter
   - last letter
   - first two letters
   - last two letters
   - previous letter
   - next letter

   After applying word features rules:

```
Load data from trn-tweet.pos
Loaded 1827 sentences
Load data from dev-tweet.pos
Loaded 547 sentences
Extracting features on training data ...
Extracting features on dev data ...
Training CRF ...
['O', 'V', 'D', 'A', 'N', 'P', ',', '^', 'L', '~', '@', 'U', '$', 'E', '!', '&', 'R', '#', 'G', 'T', 'M', 'X', 'S', 'Z', 'Y']
Acc = 0.5575317740843799
['O', 'V', 'D', 'A', 'N', 'P', ',', '^', 'L', '~', '@', 'U', '$', 'E', '!', '&', 'R', '#', 'G', 'T', 'M', 'X', 'S', 'Z', 'Y']
Acc = 0.5421045802517193
```

2. - After applying algorithm="averaged perceptron":

```
Load data from trn-tweet.pos
Loaded 1827 sentences
Load data from dev-tweet.pos
Loaded 547 sentences
Extracting features on training data ...
Extracting features on dev data ...
Training CRF ...
['O', 'V', 'D', 'A', 'N', 'P', ',', '^', 'L', '~', '@', 'U', '$', 'E', '!', '&', 'R', '#', 'G', 'T', 'M', 'X', 'S', 'Z', 'Y']
Acc = 0.9579980446717304
['O', 'V', 'D', 'A', 'N', 'P', ',', '^', 'L', '~', '@', 'U', '$', 'E', '!', '&', 'R', '#', 'G', 'T', 'M', 'X', 'S', 'Z', 'Y']
Acc = 0.7898014791747762
```

- The purpose of the original method using logistic regression to train the CFG model is to find out the the correct parameter θ so the conditional likelihood of all the y_i given the features could be maximized.
  If we decide to use the averaged perceptron to replace the logistic regression, we are essentially using perceptron to build up the model making the correct y_i being yield, given the corresponding features such as x_i and y_{i-1}. Therefore, we are training a averaged perceptron model to classify the y_i given x_i and the y_{i-1}, in this case. Of course, the node in our CRF model could have a more complicated relationship but the basic idea is the same.
  Different from training using logistic regression that the perceptron would only return the tag predicted of each word instead of providing probabilities of tags. What's more, perceptron algorithm is trying to make the tag of training set being predicted correctly while logistic regression tends to maximize the likelihood.