Name: Soonho An
Andrew ID: Soonhoa

Algorithms for NLP

Homework 2

1. Statement of Assurance

"I certify that all of the material that I submit is original work that was done only by me."
<Soonho An>, <Apr 27, 2020>

2. Files

The files I zipped and uploaded are stated below.
<writeup.pdf, naivebayes.py, neuralnet.py, heldout_pred_nn.txt, requirements.txt, stopword.list>

3. Task 1: Naive Bayes

3.1 How did you use the dev data (e.g., how to split them for training and testing)

I initially tried to split them into 8:2 = train:test but soon used all of them as training set and just tested on the heldout_text file

3.2 How did you preprocess the data?

I tokenized the text by 1) removing the stop words in stopword.list, 2) turning all the tokens into lower case, 3) removing all the punctuations in the tokens, and 4) removing all the tokens that contain numbers.

3.3 For each feature you tried,

1) What is the feature (e.g., unigrams, bigrams, etc.; briefly in no more than 2 sentences)

I used CTF (Collection Term Frequency), which is proposed in the homework of other class called Machine Learning for Text Mining. To extract CTF feature, I first tokenized the text data carefully with the preprocessing method that I've explained above. Then I generated vocabulary for the training set, and counted top 500 most frequent tokens. After that, I generated feature vector in 500 dimension, which just shows that how many time the token is appeared.

2) Why do you think it makes sense to use this feature?

Because the word itself is really important measure for the meaning of the sentence, and CTF feature is a good measure for that word because it's based on the counting of the words.

3) Did the feature improve accuracy, either on the heldout or on your own test set?

I'm not sure what 'improve' means here because I only used CTF feature, but I think it worked well because the autograder gave me the full credit.

4) Why do you think it did or did not improve accuracy?

Because the autograder gave 20.0 on this code.

3.4 What is the final accuracy on your own test set?

Since I didn't utilized my own test set, I have no final accuracy on my own test set. One thing I can sure is that I got more than 70% of accuracy on the heldout data set.

4. Task 2: Neural Networks

4.1 How did you use the dev data, if different from Task 1?

Same as Task 1.

4.2 How did you preprocess the data, if different from Task 1?

Same as Task 1.

4.3 What is the architecture of your model? (recommend to include a figure that describes the entire architecture of your model)

I used simple DNN which has 5 layers of [500 – 1000 – 500 – 200 – 2] neurons. Hidden layers are composed of ReLU and output layer is softmax function.

4.4 What does the model take as input?

Model takes feature vector (CTF) as input.

4.5 What feature did you try other than word unigrams, if any? Did they improve accuracy?

I used CTF instead of unigrams, and I don't know if it was better because I didn't try word unigrams but my results seems okay.

4.6 Any non-trivial extensions you made to the model

I didn't try it because I didn't have enough time.

4.7 What is the final accuracy on your own test set and how is it compared to your naive Bayes classifier?

Since I didn’t utilized my own test set, I have no final accuracy on my own test set. For heldout data set, I think I got 84.78 % of accuracy because the autograder gave me 47 and the accuracy can be calculated via the HW2 Grading rubric.

4.8 Qualitative analysis on the model outputs and compare it with ones from your naive Bayes.

As I’ve explained above, my neural network got 84.78% accuracy and my naive Bayes classifier got more than 70% accuracy. What I’ve figured out more is that the accuracy of my naive Bayes classifier seems to be about 80% because when I’ve turned in the heldout_pred_nb.txt as heldout_pred_nn.txt, the autograder gave me 41 points. In conclusion, the neural network gave slightly better result than the naive Bayes classifier.