**CS510 2018 Fall Quarter HW2** **Shaoshu Xu**

**Reflection Question**

**What are three examples of sentences (or any length of text) on which you think your classifier failed?**

1. Negative, "I don't love school ". Result: positive
2. Negative, "I hate this movie". Result: neutral
3. Negative, "Among the slackest, laziest, least movie-like movies released by a major studio in the last decade". Result: neutral

**For each example, why do you think it failed? What was hard about the target text?**

1. The basic Bayes classifies the negative sentence to positive. It may because the classifier notice the word "love" and calculate by a higher positive probability. It ignores the context of two together words. The improved classifier which using bigrams performance better for this sentence.

2. The basic Bayes classifies the negative sentence to neutral. It may because there is no enough word "hate" in our training set and it is true that people rarely use such excessive words in the reviews. And the improved one classifies correctly.

3. The basic Bayes classifies the negative sentence to neutral. It may because it only uses "unigrams" as features ignoring the context of some together words. And the improved one which using bigrams classifies correctly.

I notice that my classifier classifies many sentences to "neutral" wrongly. I think it may because of the threshold value. As shown in my code ("abs(pos - neg) < .5"), I set it to 0.5.

**Evaluate Your System**

**Compare the two systems based on their precision, recall, and f-measure:**

(Return the asked measures when the labels are provided, with testDir = "movies_reviews/")

```
SeandeMacBook-Pro:cs510hw2 seanxu$ python 2classifier_evaluate.py
(3474012, 1424841, 66514, 26235)
13864 test reviews.

File Classifications:
Naive_Bayes_Classifier
confusion matrix:
[[1526, 757, 452], [0, 0, 0], [197, 387, 10545]]
('accuracy:', 0.8706722446624351)
('recall:', 0.9489779874213836)
('f-measure:', 0.9081402347276557)
```

```
SeandeMacBook-Pro:cs510hw2 seanxu$ python 2classifier_evaluate.py
(3407500, 1398614, 66514, 26235)
13864 test reviews.

File Classifications:
Naive_Bayes_Classifier_Best
confusion matrix:
[[2516, 182, 37], [0, 0, 0], [301, 388, 10440]]
('accuracy:', 0.9345066358915176)
('recall:', 0.9745749962389048)
('f-measure:', 0.954120332866927)
```

(Return the predicted classes for the unknown test files in "MyTestFolder")

```
SeandeMacBook-Pro:cs510hw2 seanxu$ python 2classifier_evaluate.py MyTestFolder
(3407500, 1398614, 66514, 26235)
('MyTestFolder.DS_Store', 'negative')
('MyTestFoldertestfile1.txt', 'negative')
('MyTestFoldertestfile2.txt', 'positive')
('MyTestFoldertestfile3.txt', 'negative')
('MyTestFoldertestfile4.txt', 'positive')
('MyTestFoldertestfile5.txt', 'negative')
('MyTestFoldertestfile6.txt', 'positive')
```

**Reflect on why you think the systems performed well (or poorly):**
Naive_Bayes_Classifier_Best performed slightly better because we created bigrams dictionary during training process. And usually two words together will have more context, while a single word loses that context. Context helps us better understand the true meaning of words or sentences. Thus, it will perform well than the unigram model.

**Outline ways that you would extend the system to improve future performance:**

To improve the performance, we can use deactivation words to filter non-keyword words. And we can also try to create trigrams dictionary during training. I think it could understand the true meaning of a piece of text better.

**Consider and present other ways that you might approach this task of classifying sentiment in text:**

We can use SVM, KNN, Decision tree or other methods to approach this task. And I think Neural Network approach (like Convolutional neural network or Recursive neural network) will have a better performance.