**Final Report**

Note: The first several channels of the code file are about baseline,  the last several channels are about machine learning models.

**Introduction**

It is important to deliver sentiment analysis to help us give better decision of some classified problems. Like for some reviews, we would know the feedback status of many entities. That would even help us get the health status of one specific industry and make some improvement based on that.

**Problem Definition**

For this project, I will focus on the sentimental analysis of movie review provided by Cornell. The task is just to create a great classifier system to tell whether one specific review is pos or neg. The project aims to make comparison for different models and give out the best one based on the result.

**Previous Work**

Methods are from the class and previous assignment, and some are from the idea generated from website.  Actually, the previous assignments help me a lot to implement this project.

**Approaches, Results and Discussion**

Q1
I implemented unigram model approach for the baseline model. I just use the probability of whether pos or neg (just like assignment2) to determine whether it is positive or not. The model's success rate is about 50% since it does not use some advanced comparison method. But it works because the simple comparison makes sense with just comparing the probability in pos dataset and neg dataset. Besides, one important step here is to shuttle the dataset so that "pos" and "neg" data are intersected with train and test data seperated.

precision 0.5050505050505051
accuracy 0.5025
recall 0.9852216748768473
F_measure 0.667779632721202

Q2

There are 3 machine learning models that are implemented here. GaussianNB, DecisionTree, SVM. Same as Q1, I keep using the shuffled data. During this process, I just encode each unique word with a unique integer (like the method used in assignment3). For the classifier definition, I let 1 stand for pos and 0 stand for neg. For feature selection, I use the encoded result of the first 90 words of each file to be the feature. Then use 30 fold to seperate the dataset to implement our machine learning models respectively. Here is the result.

Naïve Bayes
precision:  0.604904871949307
recall:  0.6070494748910097
accuracy:  0.6063621287501885

F_measure:  0.6023851131020319

Decision Tree
precision:  0.9658004158004159
recall:  0.9660470248705544
accuracy:  0.9659279360771897
F_measure:  0.9658490116135766

SVM
precision:  0.9668735064349099
recall:  0.9677758446320139
accuracy:  0.9674280114578622
F_measure:  0.9671834968976276

As we can see, Decision Tree and SVM work better and has similar performance, which is over 95 %. However, Naïve Bayes is just around 60%.

Q3
I used 4 ways to improve based line model.

1. remove punctuation marks

precision 0.47
accuracy 0.47
recall 1.0
F_measure 0.6394557823129251

2. make all words lower-case
precision 0.4824120603015075
accuracy 0.4825
recall 0.9948186528497409
F_measure 0.6497461928934011

3. use add-one smoothing approach
precision 0.485
accuracy 0.485
recall 1.0
F_measure 0.6531986531986533

4. do 1, 2 and 3
precision 0.49874686716791977
accuracy 0.4975
recall 0.995
F_measure 0.664440734557596

From the above result, we find that none of those ways would have significant improvement for my models. I guess no one is beneficial from the result.

Q4
The size of features do have influences some models.

Size = 10

Naïve Bayes
precision:  0.5293422087771963
recall:  0.5294172936736425
accuracy:  0.5318709482888587
F_measure:  0.5258812360379529

Decision Tree
precision:  0.9613308544858987
recall:  0.9612817581650173
accuracy:  0.9613900195989747
F_measure:  0.961115642583954

SVM
precision:  0.6982029333178017
recall:  0.6986203769552258
accuracy:  0.7003769033619782
F_measure:  0.6957112458046608

Size = 90

Naïve Bayes
precision:  0.604904871949307
recall:  0.6070494748910097
accuracy:  0.6063621287501885
F_measure:  0.6023851131020319

Decision Tree
precision:  0.9658004158004159
recall:  0.9660470248705544
accuracy:  0.9659279360771897
F_measure:  0.9658490116135766

SVM
precision:  0.9668735064349099
recall:  0.9677758446320139
accuracy:  0.96742801145786
F_measure:  0.9671834968976276

Size = 150

Naïve Bayes
precision:  0.6345053893153507
recall:  0.6352333745759787
accuracy:  0.6343283582089553
F_measure:  0.6305392057875256

Decision Tree
precision:  0.9622816609713162
recall:  0.962301328658065

accuracy:  0.9623699683401177
F_measure:  0.9619526696420125

SVM
precision:  0.9628936788679436
recall:  0.9628781458952237
accuracy:  0.9628976330468867
F_measure:  0.9628454367265794

As we can see from the result, Naive Bayes and SVM’s performance is related to the size of features. However, Decision Tree model’s performance is always at such high from small size to large size.

**Further Disussion and Conclusion**

The Decision Tree and SVM work the best, maybe in the future I should try some different models like KNN and non nonlinear models. Also, we should try to change the value of some important parameters in each model to make some improvement. This project really helps me a lot and help me integrate some important methods in the whole class.

**Extra Credit (for Yelp Dataset)**