Assignment 3 Report

Basic Implementation:

I use normal list to collect data at first, this might decrease speed because the under laying implementation of python's list and nparray are different and nparray has a more efficient cost. At the part of feature representation I used CountVectorizer and TFIDFVVectorizer on UB_BB and MBC separately.

UB_BB

At this part I chose LogesticRegressionCV as my first scratch of LR and it didn't work a lot better than normal LogesticRegression but take more time also. The reason why it caused this is because CV means cross-validation and it will train the model on several split training set to get different feature distribution. After switch to normal LR, the precision, recall and f-score didn't decrease but speed is faster.

On SVM, I used LinearSVC as my first scratch because it's basic. So I think the Occam's Razor will do the trick (Occam's Razor). But unfortunately this model take way too much time on training with large data set. My UB and BB sets are 30000 and 250000. Linear SVC is extremely slow on UB set and it takes forever to train on BB set so I changed it to SGDClassifier to reduce time cost. Because SGD is normally using on large scale, sparse data set, especially NLP (Stochastic Gradient Descent). The accuracy of SGDClassifier is 10% higher than LinearSVC on UB set so I assume it also do better on BB set.

MBC

I’m using TFIDFVVectorizer and stop word as feature representation. It pop out all the “stop” word to get more significant features. I named the <Config> part UBSW to represent “Unigram Stop Word”. I didn’t changed much of models because some of them works fine and some others just not suit for NLP (SVM, RF) because they all get relatively low accuracy on all three feature representation. So I’m more focusing on LR and NB.

NB is simple because it does not require additional hyper-parameter so it completely simple and it can also produce 89% precision and recall rate for both UB_BB and MBC data sets. The simplest explain the most. I also tuned LR’s C and tol. I didn’t implement a grid search for this but I did test some of the parameter by hand and get that the most effective C should around 20-50. Precision/recall will decrease if C is too high or too low. There is no trick or text book tutorial for parameter tuning. It’s just based on experience and experiments.

MBC_Final

I might use my own UBSW configuration and NB model as my final choice. For now all the experiments and feature extraction shows that NB has the best performance. My LR did get to the same performance as NB but I’m not sure how it will end on the test data set that I don’t have. But after I changed my training data to evaluation set and to predict training set, LR gives me the best performance of the most. So I’ll choose LR.

Citation

Occam's Razor. 1287-1347. Wikipedia

https://en.wikipedia.org/wiki/Occam%27s_razor

Stochastic Gradient Descent. Scikit-Learn.

https://scikit-learn.org/stable/modules/sgd.html