Programming Assignment 2: Semi-supervised Text Classification

CSE 256: Statistical NLP: Spring 2019

University of California, San Diego

Wei-Cheng Huang

1. Supervised Learning

In this session, I used feature engineering methods to see if the accuracy could be improved. I tuned on hyperparameters of TF-IDF and linear regression model, and also tried if removing some less important features would help. The following session would show the result and the procedure I decided my final hyperparameters.

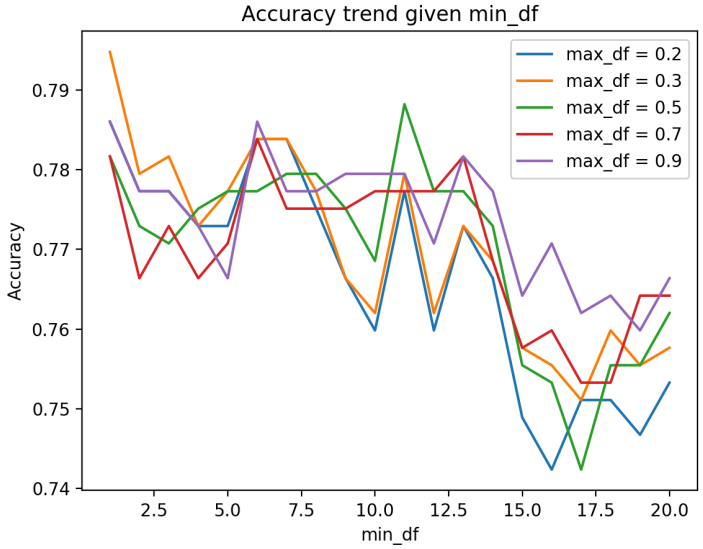
(1-a) Tuning Logistic regression model:

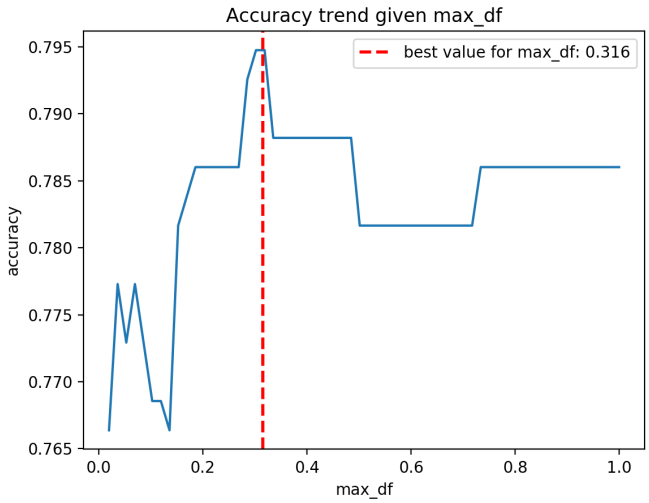
In logisticRegression model, there are some hyper parameters we can play with. The first one is "penelty", which is the loss of the model. And the second one is the "solver", which means algorithm used to optimize the problem.

Accuracy given penalty and Optimization algorithms					
	newton-cg	**lbfgs**	**liblinear**	**Sag**	**Saga**
L1	N/A	N/A	0.7707	N/A	0.777
L2	0.7838	0.7838	0.7838	0.786	0.786

(1-b) Tuning TF-IDF value:

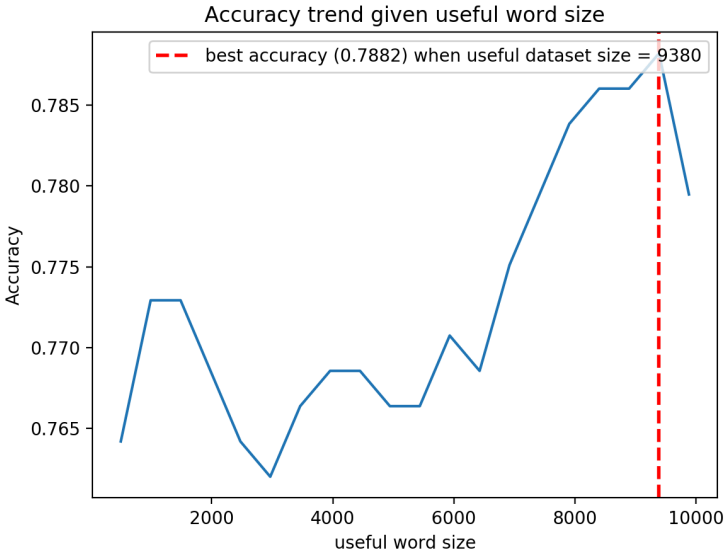
TF-IDF technique is used to weight words according to hoe important they are. Words that are used frequently in may documents will have a lower weighting while infrequent ones will have a higher weighting. The parameter max_df is used to ignore terms that appear too frequent. And min_df means to ignore terms that rarely appears. Max_df =0.50 means ignoreterms that appear in more than 50 percent document. And min_df = 0.01 means ignore terms that appear in less than 1% of the documents, or 1 means to ignore terms that appear in less than 1 document. According to the graph below, we determine that when max_df equals 0.316 and min_df equals 1 we can obtain best accuracy.





(1-c) Removing features by Stopwords

I extracted the least important features(words) and make them into a stopword_list, all of which will be removed from the resulting tokens. According to my experiment, I can see that when using 9380 features with greatest prediction probability, I can get the best accuracy, which also means that it is possible most of the features are relevant to the predictive results.



Accuracy versus N-gram range			
Ngram Range	**Accuracy**	**Ngram Range**	**Accuracy**
(1,1)	0.7838	**(2,3)**	0.7336
(1,2)	0.7947	**(2,4)**	0.7164
(1,3)	0.7838	**(3,3)**	0.6523
(1,4)	0.7863	**(3,4)**	0.6703
(2,2)	0.7336	**(4,4)**	0.6288

(1-d) Utilizing Bigram, Unigram

According to the graph below, we can see that using combination of unigram and bigram would obtain the best result. It is possible that the word sequence could help the model better developed. And why trigram doesn't work well enough is probably that binary classification problem doesn't depend on much of word sequence.

(1-e) Final parameters combination

According to the experiment results above, I decided to choose penalty = 'l2', optimization algorithm = 'saga' for linear regression model. Min_df =1 and max_df = 0.32 for TF-IDF method, using the combination of unigram and bigram models, and set up around 500 stop words in my model. Finally I can obtain accuracy at 0.7945 at dev dataset locally, and 0.7876 on Kaggle test dataset.

2. Semisupervised Learning

In this session, we used the hyper parameters we tuned in the supervised model, predicting on unlabeled data, then train on a new dataset including training data and a portion of unlabeled data. We expect to see better accuracy after semisupervised learning.

(2.1) Expanding the Labeled Data

When we are expanding our dataset, there are two steps we need to consider. The first step is to determine which dataset to include. The second is set a STOP condition to break the loop. How I choose my unlabeled data is by calculating the confidence of such data. I only included data whose confidence is greater than 0.98 in order to maximize the possibility I use "good" data. If the predictive possibility of either "positive" or "negative" is large enough, then I would include this data with the training dataset, then do the training process again. The reason why x axis only span up to 0.5 is because there are only about half of unlabeled data whose confidence is greater than 0.98 only

Pseudo Code:

>train supervised model

>for (small portion of unlabeled dataset):

-use supervised model to predict unlabeled data_x

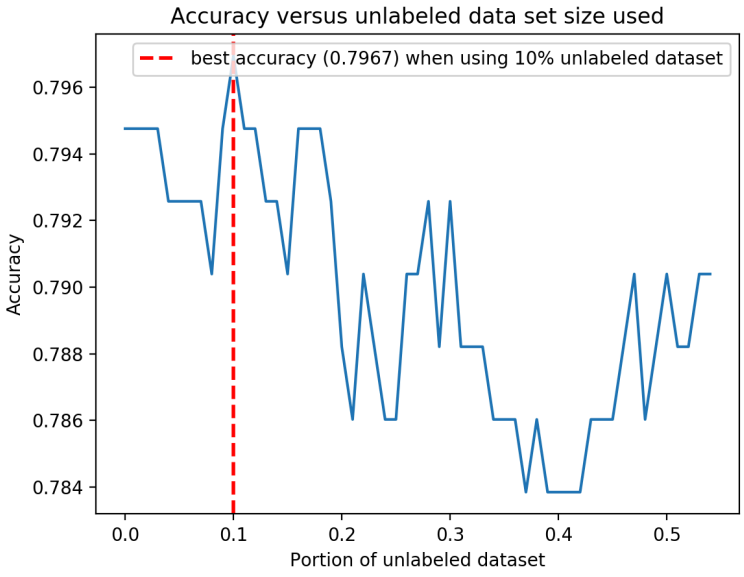
-combine training set and unlabeled set

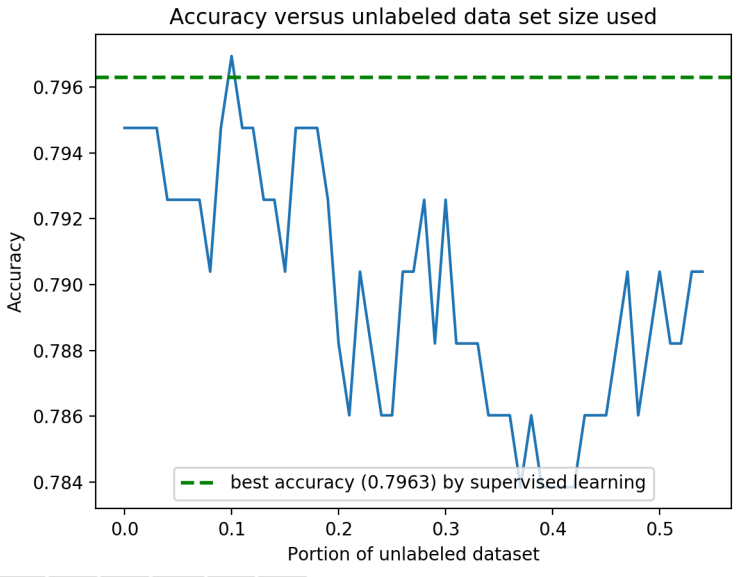
-retrain model by new data set

-model evaluation on dev set

(2.1a) Comparison with supervised classifier

We can see that from the experiment result below, we obtain best result when only using around 10 percent unlabeled data. The reason might be that our model obtained from training set is not robust enough to predict unlabeled data, which makes us include bad unlabeled data.Therefore, it seems like the more training set we use, the worst result we obtain. Compared with the training data set, the semi-supervised result is a bit worst.





(2.1b) features whose weight change significantly

There are some features whose weight change significantly and some of them are interesting. For example “not good”, “not the”,”have no”, “was not”, some of the features including “not”. That’s probably when “not” appears in my data, my original model would be affected significantly. Apparently, my model is not robust enough to predict those words, which are easily influenced by the unlabeled data.

Below are some more examples whose weight changed significantly when training in unlabeled data.

['always', 'always friendly', 'always great', 'amazing', 'amazing and', 'amazing experience', 'an hour', 'and also', 'and awesome', 'and both', 'and delicious', 'and friendly', 'and great', 'and have', 'and helpful', 'and reasonably', 'and service', 'and such', 'and the', 'another', 'anywhere', 'around', 'asked', 'at the', 'atmosphere', 'atmosphere is', 'attentive', 'attentive and', 'authentic italian', 'average', 'awesome', 'awful', 'back', 'bad', 'bbq', 'be back', 'beautiful handbags', 'because', 'because the', 'beer selection', 'best', 'best and', 'best buffet', 'best couture', 'best me'dirty', 'disappointed', 'do not', 'dogs', 'downtown', 'drink', 'driver', 'employees', 'ended', 'enjoyed the', 'enough', 'especially', 'even have', 'ever', 'ever to', 'ever was', 'everyone', 'everyone is', 'everything', 'excellent', 'excellent and', 'expensive', 'experience', 'experience ever', 'experience from', 'extremely', 'fairly', 'fantastic', 'fast', 'favorite', 'favorite is', 'favorite place', 'feels', 'find', 'fish', 'flavor', 'food and', 'food great', 'food perfect', 'food the', 'food was', 'fresh', 'friendly', 'friendly and', 'friendly service', 'friendly staff', 'front', 'front desk', 'fun', 'gel', 'generous', 'get your', 'give', 'good', 'good sales', 'good service','great', 'great and', 'great atmosphere', 'great however', 'great little', 'great on', 'great place', 'great service', 'great spot', 'great staff', 'great time', 'great variety', 'greeted', 'greeted 'much good', 'must', 'my favorite', 'nail', 'never go', 'new york', 'nice to', 'nicely', 'nicely until', 'not', 'not bad', 'not go', 'not the', 'of beer', 'of food', 'oh', 'ok', 'online', 'or', 'or the', 'order', 'ordered', 'ordering', 'other', 'our food', 'our order', 'outstanding', 'owned', 'patio', 'pizza', 'place','the waitress', 'the worst', 'then', 'there the', 'there were', 'they', 'they also', 'they charge', 'they would', 'this is', 'this little', 'this location', 'this place', 'this store', 'thought was', 'time tonight', 'times', 'times and', 'times the', 'to after', 'to be', 'to die', 'to get', 'to love', 'to so', 'to wait', 'to write', 'told', 'too', 'too much', 'town', 'town the', 'true', 'try', 'twice', 'use', 'used to', 'usually', 'variety', 'very friendly', 'very good', 'waited', 'waitress was', 'walked', 'want to', 'was amazing', 'was awful', 'was excellent', 'was friendly', 'was great', 'was greeted', 'was not', 'was rude', 'was so', 'was terrible', 'was which', 'waste', 'waste your', 'went', 'were great', 'when', 'which of', 'will', 'will definitely', 'within', 'work', 'worst', 'worst customer', 'worst experience', 'worst service', 'would', 'would be', 'would do', 'write', 'wrong', 'york']

(2.2) Designing Better Features

In this session , we can use word embedding method to tackle with small dataset training problem. Since we can categorize words with similarity, therefore, even if we haven’t see such word in the training set due to sparsity, we might still have capability to predict unseen relatively large dataset. (I didn’t successfully implement it.)

3. Kaggle

Username: tommyhuangtw

Display name: WeiCheng

Email Address: w2huang@eng.ucsd.edu