Lab 1 Report

Introduction

The program is a sentiment training and predicting program, based on 2000 documents data (1000 for positive, 1000 for negative) and using 1600 documents to get training and 400 documents to get testing. N-Gram model has been applied in the program, more specifically, in the program, Uni-gram, Bi-gram, Tri-gram have been achieved. To run the program with different kinds of N-Gram model, by typing in the optional parameters following the execution command, i.e.

python lab1.py review polarity –t[option]

The option could be unigram,bigram,trigram. Missing the optional parameter, all the approach would be adopted.

Implementation

- Training the data adopted the binary perceptron approach.
- The max number of iteration was set to 10. Before each iteration, the row of training data would be random, but in order to make the results to be reproducible I set the seed for random function permutation and update the seed by using the number of iteration.
- The graphs of the multiple passing over the training data will show when the program are running.
- After all iteration, the weight would be taken the average for all document.
- The most positively-weighted feature would be selected and printed in the console.

Evaluation

When the program is running, all the process would be recorded including the cost, the size of the feature and the error. The error, here, means the difference between the prediction label and the training data label. First of all, the cost of three approaches are totally different, showing as the table below.

Type	Extracting feature	Building matrix	Training	Number of features
Uni-gram	0.54s	12.70s	15.12s	39588
Bi-gram	1.76s	174.02s	192.41s	497761
Tri-gram	4.66s	306.87s	941.20s	820398

Obviously, Uni-gram is the most time saving, comparing with other two approaches, because it has the smallest number of feature that could save time

on building matrix and training. The number of feature increase sharply form Uni-gram to Tri-gram because of has low probability of the same phrase or word group occurring in the all documents.

Secondly, applying the evaluation mentioned on the class on the results, we could get the comparison of precision and recall among the three results, and it is showing as the table below.

Type	TruePositive	TrueNegative	Precision	Recall
Uni-gram	169	169	0.84	0.84
Bi-gram	170	153	0.85	0.78
Tri-gram	151	150	0.76	0.75

Three kinds of approaches have similar precision and recall, only Tri-gram performed poorly. I think the reason might be that the scale of training data is not large enough. Many phrases in the training data might occur only one time. So, in this case, only Uni-gram preforms well.

Thirdly, the top 10 positively-weighted features would be printed when run the program, showing in the table below. Most of the results are appropriate.

Uni-Gram		Bi-Gram		Tri-Gram	
Positive	**Negative**	**Positive**	**Negative**	**Positive**	**Negative**
great	bad	the best	the worst	of the best	of the worst
jackie	worst	star wars	supposed to	to take the	could have been
quite	why	is very	have been	boogie nights is	mission to mars
seen	only	more than	should have	my best friend's	the first movie
most	nothing	starship troopers	the only	a good job	supposed to be
also	plot	is also	but the	the devil's advocate	have been a
best	any	the world	and then	american history x	the only thing
fun	script	he is	a bad	the life of	a bad movie
see	boring	that will	to show	but it does	is supposed to
will	unfortunately	in their	the filmmakers	of the series	the king and

However, if we want use these words to predict another new domain, it would be ok, but the results would be poor, because some of the high weight terms have the special meaning for the present domain and it cannot be applied on other domains. The better features for new domain, I think, would be the common words that do not involve the character of the domain.