INLS 613 Text Data Mining
Homework 2

Due: Wednesday, February, 27, 2019 by 11:55pm via Sakai

1 Objective

The goal of this homework is to give you exposure to the practice of training and testing a machine-learning model for predictive analysis of text. You will use a toolkit called LightSIDE. As in HW1, the task is to classify movie reviews into *positive* and *negative* sentiment. You will be training a Naive Bayes classifier to predict whether a movie review expresses a positive or negative opinion. The *primary* goal of the assignment is for you to learn to experiment with different feature representations and do error analysis, and to review some concepts we've covered in class.

2 Software Details and Data

LightSIDE is a text-mining toolkit built by Elijah Mayfield at Carnegie Mellon University. You are expected to learn how to use it as part of this exercise. The user's manual is a great place to start![1] You can download and install LightSIDE on your own computers.

The training and test sets for this assignment can also be found inside the `/LightSIDE/data/hw` directory. These are smaller versions than the datasets used for Assignment #1.

3 Assignment

The assignment is divided into two parts.

3.1 LightSIDE Exercises (70%)

Complete the following exercises. Please provide your answers using complete paragraphs. Make sure you use a Naive Bayes classifier (the default classifier) and, with the exception of Exercise #5 below, make sure you use unigram features and select the "Skip stopwords in N-Grams" option.[2]

1. Generate a feature table using unigram features. LightSIDE provides several metrics that measure the degree of co-occurrence between a feature and a target class value (in our case, *positive* and *negative*). This and the next question focus on precision. Suppose you have the following contingency table:

	positive	negative
word w occurs	a	b
word w does not occur	c	d

[1] http://ankara.lti.cs.cmu.edu/side/LightSide_Researchers_Manual.pdf

[2] This option tells LightSIDE to remove stopwords from your unigram feature representation.

The precision of term w with respect to the *positive* class is given by $\frac{a}{(a+b)}$, and the precision of term w with respect to the *negative* class is given by $\frac{b}{(a+b)}$.

Sort the feature table by descending order of precision with respect to the positive class. You can do this by selecting *positive* in the "Evaluations to Display" tab and then double-clicking on the header labeled "precision". This functionality is illustrated in Figure 1.a (.

At the top of the list, you will find some terms you might expect. For example, "breath-taking" has perfect precision with respect to the *positive* class. There are, however, a few surprises. For example, "ultimately" has perfect precision with respect to the *positive* class. Choose a feature from the top of the list that you did not expect to be perfectly correlated with a particular class value and look at the instances in train.csv where the term appears. List the term, its precision value, and provide an explanation for why the term has a high co-occurrence with the target class value. Is the term associated with some language phenomenon that you did not anticipate, or is it a statistical anomaly. **(10%)**

2. Now, look at the middle of the list. Here, you will find terms that have a precision of 0.5 for a given class value. In other words, these are terms that occur an equal number of times in *positive* and *negative* reviews. Again, here you will find some terms you might expect. For example, "zombies" occurs an equal number of times in *positive* and *negative* reviews. There are, however, a few surprises. For example, "depressing" (an arguably negative term) also appears an equal number of times in *positive* and *negative* reviews. Choose a feature from the middle of the list (with a precision value of about 0.50) that you did not expect to be *uncorrelated* with a particular class value and look at the instances in train.csv where the term appears. List the term, its precision value, and try to provide an explanation for why the term has a low co-occurrance with the target class value that you expected it to have a high co-occurrence with. **(10%)**

3. Train a model on train.csv and then re-apply this model to train.csv (yes, the same file). How does the accuracy of the model on train.csv (trained on train.csv and tested on train.csv) compare to its performance on test.csv (trained on train.csv and tested on test.csv)? This functionality is illustrated in Figure 1.b (the area labeled "D").

   You'll notice that the model's accuracy on train.csv is not perfect. In other words, the model fails to correctly predict some instances it used to estimate its own model parameters! Use the "predict labels" functionality in LightSIDE (Figure 1.c, label "E") to output the model's predictions on train.csv. Select a false positive or false negative prediction, copy/paste it into your report, and provide an explanation for why you think the instance is particularly difficult. **(10%)**

4. LightSIDE allows you to set a threshold on the minimum number of training set instances that must contain a particular feature in order for that feature to make it into the feature representation (Figure 1.a, label "A").

   Let's call this threshold t. If we set $t = 1$, then every term in the training set makes it into the feature representation. If we set $t = 2$, then only terms that appear in *at least* 2 training set instances make into the feature representation. If we set $t = 5$, then only terms that appear in *at least* 5 training set instances make into the feature representation. You should convince yourself that greater values of t lead to fewer features.

Fill out the table below with the accuracy numbers from training models with $t = 1, 2, 3, 4, 5$ and evaluating these models on the training set (second column) and on the test set (third column). Using LightSIDE, this will require you to construct five different *feature tables* in the 'Extract Features' tab and perform 10 train/test operations in the 'Build Models' tab.

threshold	training set accuracy	test set accuracy
0	?	?
1	?	?
2	?	?
3	?	?
4	?	?
5	?	?

How does the accuracy of the classifier change for these different values of this threshold on the training set and on the test set? Write a short paragraph explaining what is happening and why. **(10%)**

5. Explore different feature representations in order to improve the accuracy of your Naive Bayes classifier on test.csv. You can generate different feature representations by selecting different types of features from the "configuration plugin" pane (Figure 1.a, label "B") and by adjusting the threshold parameter mentioned in the previous question (Figure 1.a, label "A") . Note that LightSIDE can be slow to learn a model and apply a model when the feature representation is large. Provide a couple of paragraphs description of what you tried, what worked, and what did not work.

   The student that achieves the best accuracy (on test.csv) will be given ONE bonus point towards his/her final grade, a small prize, and, of course, fame and glory. **(30%)**

3.2 Measuring Inter-annotator Agreement (30%)

When we do predictive analysis of text, we often use human annotators to produce gold standard labels used for training a model and evaluating its performance. As we discussed in class, one important criterion for predictive analysis is that humans should be able to recognize the concept with a reasonable degree of success. If human annotators cannot produce consistent annotations by working independently, then it is reasonable to expect that a machine learning model will have difficulty recognizing the concept automatically. The next set of exercises are associated with different ways of measuring inter-annotator agreement.

Suppose we want to predict whether a blog post contains an argument that is *liberal*, *conservative*, or *neutral*. Using a coding manual, a pair of annotators (denoted as annotator $\mathcal{A}$ and annotator $\mathcal{B}$) label 900 blog posts independently and produce the following contingency table.

Answer the following questions. If you wish to receive partial credit, please show your work.

1. Compute the inter-annotator agreement in terms of percent agreement. **(10%)**

2. One disadvantage of percent agreement is that it does account for the level of agreement that would be expected due to random change. Compute the Cohen's Kappa agreement

		$\mathcal{A}$			
		liberal	conservative	neutral	
$\mathcal{B}$	liberal	100	30	170	300
	conservative	50	130	20	200
	neutral	100	40	260	400
		250	200	450	900

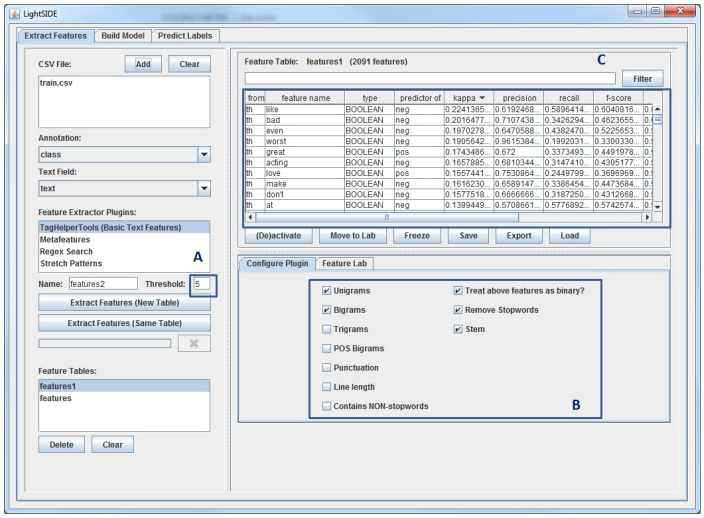
under the assumption that the annotators are unbiased and therefore each one independently labels an equal number of blog posts as *liberal*, *conservative*, and *neutral* (i.e., 300 each). **(10%)**

3. Compute the Cohen's Kappa agreement under the assumption that the annotators are biased and follow their *individual* biases, which are reflected in the contingency matrix above. **(10%)**

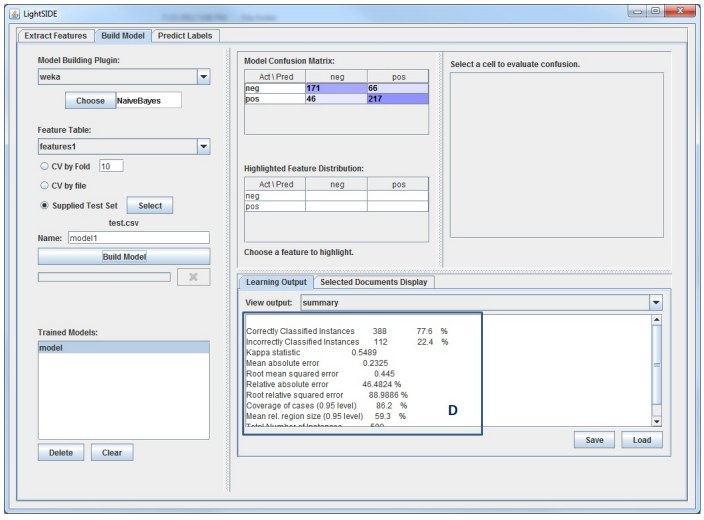
4 Submission

Submit all your answers in the form of a report. Please submit Word and PDF formats only. If you do not wish to write out your inter-annotator agreement calculations in a Word document (time-consuming!), you can write them by hand and submit a scanned PDF.

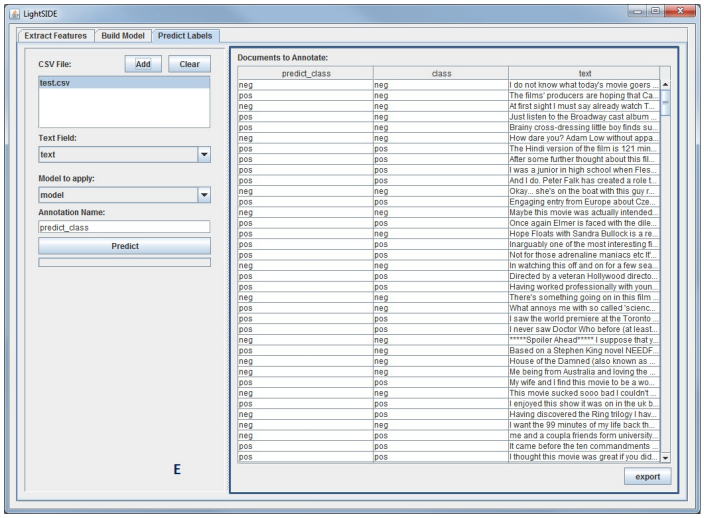
As in HW1, include your best accuracy for Exercise #5 in your report.



(a) Feature Representation Interface



(b) Model Building and Evaluation Interface



(c) Prediction/Annotation Interface

Figure 1: LightSIDE Screen Shots