INLS 613 Text Data Mining
Homework 1

Due: Tuesday, May 28, 2019 (by 11:59pm) via Sakai

1 Objective

The goal of machine learning is to design algorithms that improve their performance on a particular task using examples. The computer program "sees" examples of the task it is supposed to carry out and automatically learns to perform the task with some degree of success. Performance is measured on previously "unseen" examples.

The opposite of using machine learning to solve a task is to do it by hand. That is, by writing out explicit rules using your own understanding of the task. As a first step in developing your understanding and appreciation for machine learning, in this homework, you will program a computer to solve a complex problem by hand. The task is to classify movie reviews into *positive* or *negative* sentiment. A review has a *positive* sentiment if the author liked the movie and a *negative* sentiment if the author disliked the movie. This is a task that humans can perform fairly well. In this homework, you will try to get a computer to do it!

This exercise is similar to the one presented in Pang *et al.* [1], which describes one of the first attempts at using machine learning for sentiment analysis.

2 Overview

To help you hand-craft your own sentiment classifier, you are given a program (written in Java) that works as follows. The program takes as input two lists of terms: a list of positive terms (believed to be indicative of positive sentiment) and a list of negative terms (believed to be indicative of negative sentiment). Then, given a movie review, the classifier counts how many term occurrences within the movie review appear in each list and predicts the sentiment with the greatest count. If there is a tie, the classifier predicts a sentiment randomly with equal probability. You are responsible for providing the classifier with the list of positive and negative terms.

For example, given the following list of positive/negative terms, the movie review below would be classified as positive (i.e., the author liked the movie).

- **Positive terms:** good, great, awesome, fantastic
- **Negative terms:** bad, terrible, awful, boring
- **Movie review:** A hilarious comedy by the best director ever, Oz Scott. The list of eighties TV icons goes on and on. Milano (Who's The Boss), Yothers (Family Ties), Stone (Mr. Belvedere), Robinson (Night Court), Jackee (227), D'abo (Wonder Years), Walston (Mr. Hand!!!). It is one of the funniest movies ever. **Great** lines, meaningless subplots, cheesy, **bad** acting. It is about a group of high school kids who need to pass drivers' ed. Mac from Night Court needs them to pass their final exam, or he'll be fired. **Great** performance by Brian Bloom as the jerk/kinda cool guy Riko Conner, but is nothing compared to B.D. Wong's Kiki (pronounced kee-chee). A **great** movie for all ages, so **bad** it's **good**.

3 Software Details

3.1 Getting the Software

1. Download
   https://asandeepc.bitbucket.io/courses/inls613_summer2019/hw/ManualSentimentClassifier.zip

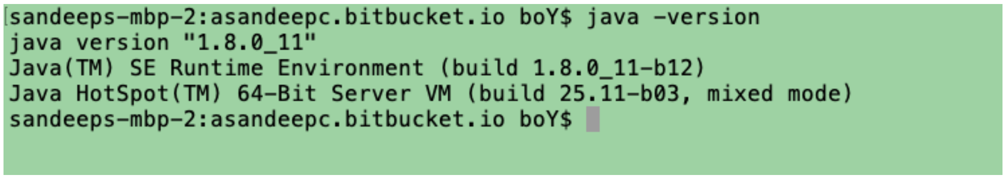
2. Unzip its contents anywhere in your computer.

3. Verify that you have all of the following files:

   - ManualSentimentClassifier.jar
   - train.csv
   - test.csv
   - positive.txt
   - negative.txt

3.2 Running the Software

1. Make sure you have the Java Runtime Environment (JRE) installed on your computer. You need the JRE in order to execute programs written in Java. To check whether you have Java installed, go to the command line and type `java -version`. If you have Java installed, you should see something like this:



```
sandeeps-mbp-2:asandeepc.bitbucket.io boY$ java -version
java version "1.8.0_11"
Java(TM) SE Runtime Environment (build 1.8.0_11-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.11-b03, mixed mode)
sandeeps-mbp-2:asandeepc.bitbucket.io boY$
```

   If you don't have Java installed, you can follow these instructions, making sure that you select the instructions that correspond to your operating system (Windows, Mac, Linux). There is A LOT of material on-line on how to install and configure Java correctly. If you have problems, please spend *at least* 30 minutes on your own before contacting the instructor.

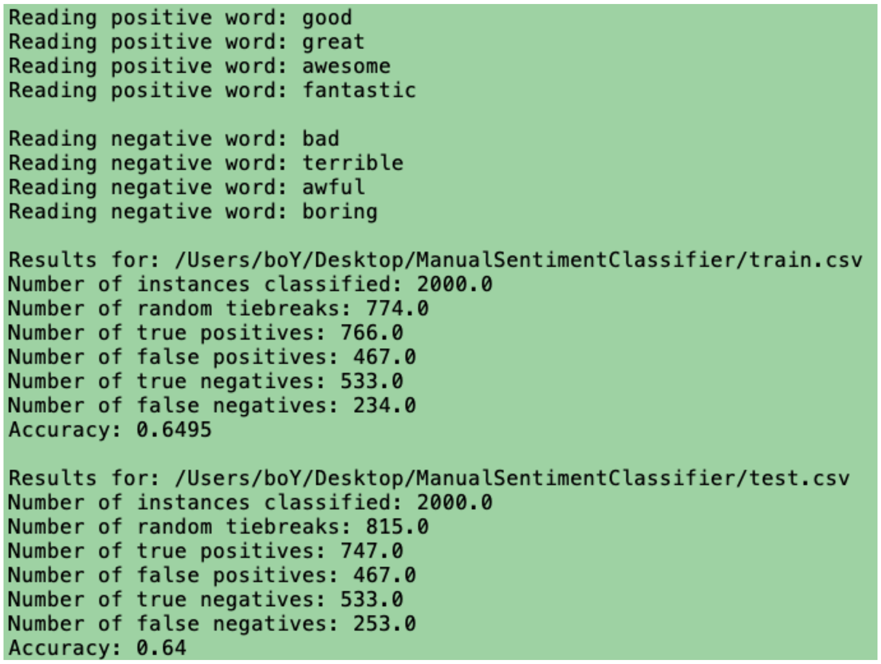
2. To run the software, open the command line, navigate to the directory that contains all of the files above, and type the following command:

```
java -jar ManualSentimentClassifier.jar <train-file> <test-file> <positive> <negative>
```

   where,

   - `<train-file>` denotes the path to train.csv file.
   - `<test-file>` denotes the path to test.csv file.
   - `<positive>` denotes the path to positive.txt file.
   - `<negative>` denotes the path to negative.txt file.

   If all works properly, you should see the following output:



```
Reading positive word: good
Reading positive word: great
Reading positive word: awesome
Reading positive word: fantastic

Reading negative word: bad
Reading negative word: terrible
Reading negative word: awful
Reading negative word: boring

Results for: /Users/boY/Desktop/ManualSentimentClassifier/train.csv
Number of instances classified: 2000.0
Number of random tiebreaks: 774.0
Number of true positives: 766.0
Number of false positives: 467.0
Number of true negatives: 533.0
Number of false negatives: 234.0
Accuracy: 0.6495

Results for: /Users/boY/Desktop/ManualSentimentClassifier/test.csv
Number of instances classified: 2000.0
Number of random tiebreaks: 815.0
Number of true positives: 747.0
Number of false positives: 467.0
Number of true negatives: 533.0
Number of false negatives: 253.0
Accuracy: 0.64
```

3.3 Input

As described above, the program takes as input a list of positive sentiment terms (listed in positive.txt—one term per line), a list of negative sentiment terms (listed in negative.txt—one term per line), a training file (train.csv) and a test file (test.csv).[1]

As mentioned in class, when we want to "train" a computer perform a task, it's important to allocate some data for learning how to perform the task and *other* data for determining how well the computer performs the task. As part of this exercise, you'll be asked to edit the list of positive/negative terms in positive/negative.txt. In doing so, you can make use of train.txt to learn about the task. However, you should not (under any circumstance) look at test.txt. Doing so would be like testing a student's level of comprehension by showing him or her the test with the answer key ahead of time. Performance must be tested on previously "unseen" data. Therefore, throughout the assignment, you must treat test.csv as a "black box".

Files train.csv and test.csv have the same format. Each row contains a movie-review/sentiment pair separated by a comma. The item to the left of the comma is the movie review text and the item to the right of the comma is the sentiment of the movie review (positive or negative).

3.4 Output

The program uses the list of terms provided in positive.txt and negative.txt to make a prediction for every movie review (every row) in train.csv and test.csv. For each input file, it outputs the following statistics.

- Number of instances classified
- Number of random tie-breaks: the number of movie reviews for which the classifier was forced to make a random prediction.

[1] CSV stands for "comma separated value". You can open a CSV file using any spreadsheet program (e.g., Microsoft Excel, Apple Numbers, Open Office Calc)

- Number of true positives (tp): the number of movie reviews that were correctly predicted as positive.
- Number of false positives (fp): the number of movie reviews that were incorrectly predicted as positive.
- Number of true negatives (tn): the number of movie reviews that were correctly predicted as negative.
- Number of false negatives (fn): the number of movie reviews that were incorrectly predicted as negative.
- Accuracy ($\mathcal{A}$): percentage of correct (positive and negative) predictions. Accuracy is calculated as:

$$\mathcal{A} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

In addition to these statistics, the program also produces a file named train.pred.csv, which lists the classifier's predictions on the training set. This file allows you to compare between the *predicted* and the *true* sentiment of each movie review in train.csv. File train.pred.csv has four columns (separated by commas): the movie review text, the true sentiment, the predicted sentiment, and an optional fourth column that signals whether the predicted sentiment was determined randomly in order to break a tie.

Your goal is to use train.pred.csv to do error analysis and improve the quality of your classifier. The quality of your classifier is determined by it's accuracy on test.csv. **Again, never look inside test.csv!**

4 Assignment

The assignment is divided into two parts.

4.1 Improving the Classifier (40%)

The quality of the baseline classifier, which uses the positive and negative words listed above, is fairly bad. Try to improve the classifier by expanding the set of positive words in positive.txt and the set of negative words in negative.txt. You are *strongly* encouraged to experiment with external resources. For example, if you search online, you will find lists of positive/negative sentiment words. Additionally, you can add words based on your own intuition and based on error analysis: by running the program and analyzing the types of errors shown in train.pred.csv.

The process of improving your classifier should look something like this:

1. Run the program.
2. Look at the prediction accuracy for test.csv.
3. Look at the number of predictions that were done randomly due to tie-breaks. If this number is large, you might need more terms in positive/negative.txt.
4. Look inside train.pred.csv and analyze some of the (false-positive and false-negative) mistakes being made.

5. Edit positive.txt and negative.txt based on what you learned.

6. Return to Step 1.

You will be graded based on your classifier's performance on test.csv (30%) and your level of effort and creativity (10%)

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.

4.2 Essay Questions (60%, 10% each)

Answer the following questions in essay form. Please use proper paragraphs and complete sentences. Include your answers in a report. Please submit Word or PDF formats only.

1. Describe your strategy for improving the baseline classifier. What steps did you take in deciding what terms to add/remove from positive/negative.txt. What external resources did you use, if any?

2. Keeping in mind that test.csv was split 50/50 in terms of number of positive and number of negative reviews, should you be surprised if your classifier achieved an accuracy of 30%? Why or why not? If 80% of the reviews in test.csv were positive, and if you knew this information, would it change your strategy for improving the accuracy of your classifier? If no, why not? If yes, how?

3. Using the training set predictions from your *best* classifier (see train.pred.csv), find an interesting example of a *false positive* mistake that was *not misclassified due to a random tie-break*. Copy/paste the example into your report. Why is the example interesting? What kind of information would the classifier have to consider in order to avoid this particular type of mistake?

4. Using the training set predictions from your *best* classifier (see train.pred.csv), find an interesting example of a *false negative* mistake that was *not misclassified due to a random tie-break*. Copy/paste the example into your report. Why is the example interesting? What kind of information would the classifier have to consider in order to avoid this particular type of mistake?

5. The program that was given to you is fairly simple: run through the movie review, count the number of terms that occur in positive/negative.txt, and predict the class with the greatest count. Did you find yourself wishing that the program did something different? If so, what and why do you think it would help? If not, why do you think that the heuristic implemented by the current program is appropriate?

6. Suppose you took your best classifier and applied it *as is* to review data from a different domain, say, reviews about hotels. Do you think it would perform equal, better, or worse? Examine your list of terms in positive/negative.txt. Are there any particular terms in either list that would not likely occur in hotel reviews? Can you think of any terms that might be predictive of positive sentiment on one domain and *negative* sentiment on the other domain?[2]

[2] If these questions seem interesting to you, you should look into an area of machine learning known as domain adaptation!

5 Submission

Please submit the final items via Sakai.

1. Report with answers 1-6 (Word or PDF).
2. positive.txt associated with your best classifier
3. negative.txt associated with your best classifier

In addition, please include the accuracy of your *best* classifier on the test set (i.e., test.csv). Please make sure that this is clearly marked somewhere in your submission.

References

[1] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. 1