P01: Text Classification and Sentiment Analysis

1 Basic Requirements

1. Programming language: Matlab (recommended and template codes provided), or other languages(without template codes)
2. Three people in a team. List team members in your report, with **name and asu ID**.
3. Implement your code under **Code** folder, and read data from **Data** folder (one **neg** folder contains all the negative reviews and one **pos** folders contains all the positive ones).
4. Final project report under name **report.pdf** at the project root folder.

2 Part I: Text Classification with Bag of Words and kNN (35pt)

2.1 Vocabulary (lexicon) creation (5pt)

1. Template file **buildVoc.m**
2. Function template **function voc = buildVoc(folder, voc)**;
3. Input: a folder path, which contains training data (get first 100 words of each text doc from dir knn_training);
4. Output: Matlab cell array **voc**, which represents the vocabulary of the words shown in the training data, except the stop words (stop words list is embedded in the code template)
5. Implement your code under **%PUT YOUR IMPLEMENTATION HERE** tag;
6. Useful Matlab functions **strtok(), lower(), regexprep(), ismember(), any()**;

2.2 Bag of Words feature extraction (10pt)

1. Template file **cse408 bow.m**
2. Function template **function feat_vec = cse408 bow(filepath, voc)**
3. Input: a file path **filepath**, which contains one review (one .txt file) and a vocabulary cell array **voc** from previous sub-section.
4. Output: one dimensional Matlab array **feat vec**, which represent the bag of words feature vector given the vocabulary **voc**;
5. Implement your code under **%PUT YOUR IMPLEMENTATION HERE** tag;
6. Useful Matlab functions **strtok(), lower(), regexprep(), ismember(), any()**;

2.3 k-Nearest Neighbor Classification (20pt)

1. Template file **cse408 knn.m**
2. Function template **function pred _label = cse408 knn(test feat, train label, train feat, k, DstType)**
3. Input: 1) test feature vector **test feat**; 2) training set groundtruth label set **train label**; 3) training set feature vector set **train feat**; 4) Hyperparameter **k** of knn; 5) Distance computation method **DstType**, 1 for sum of squared distances (SSD) and 2 for angle between vectors and 3 for Number of words in common;
4. Output: predicted label **pred label** of the testing file. 1 for positive review, 0 for negative review;
5. Implement your code under **%PUT YOUR IMPLEMENTATION HERE** tag;
6. Useful Matlab functions **sort()**;

2.4 Test your implementation

1. After your implementation, you could run **P01Part1 test.m** to debug and validate your code. It basically iteratively select one of the training review file as a validation file.
2. Question? Where is the testing data? Write or obtain another product review from the web and see if your system be able to classify it correctly?

3 Part II: Text Sentiment Analysis (35pt)

3.1 Implementation (15pt)

1. Implement a basic sentiment analysis module. Read in a lexicon, in which each word has a sentiment score. Iterate through each review file and sum up the sentiment scores for each word that exists in the sentiment strength lexicon;
2. Template file **sentimentAnalysis.m**
3. Input: a file path **filepath**, which contains one review (one .txt file) and a word with sentiment strength file **wordwithStrength.txt** under **Data** folder.
4. Output: one sentiment score.
5. Implement your code under **%PUT YOUR IMPLEMENTATION HERE** tag;
6. Useful Matlab functions **strtok(), lower(), regexprep(), containers.Map()**;

3.2 Test your implementation (15pt)

1. After your implementation, you could run **P01Part2 test.m** to debug and validate your code.
2. Question? What is the accuracy of the performance of your code?
3. Grader may run the implementation to verify it, so make sure it runs without problems

4 Improvement proposal (10pt)

There are many ways to improve either Knn or SentimentAnalysis, select one incremental approach and pursue it as far as time allows. For example: describe it in detail, implement it, test it, and compare with previous results. Points will be awarded according to the amount of work. Possible areas for improvement will be discussed in class. Improvements report will be included in the report in an Appendix.

5 Report Requirements (20pt)

Please include the following analysis in your report.

1. Make sure to explain where the algorithms worked and where they didn't and why. You are encouraged to use both text and plots to explain your observations.
2. Analyze Hyperparameter *K* in the KNN part, which *K* you empirically observed that could achieve the best performance?
3. Among the three distance metrics (sum of squared distances (SSD), and the angle between vectors and Number of words in common), which one intuitively makes more sense for classifying positive and negative review? Which one you empirically observed it to achieve the best performance?
4. For Text Sentiment Analysis task (Part II), which review in our dataset has the highest positive score but it is a negative review? And, which one has the lowest negative score, but it is a positive review? Which set of words from these reviews confused your sentiment analysis system?

6 Submission Instruction

Please place your answers under one .zip file with a formatted file name: 01 ASUID.zip (for example, if your asu ID is 101010101010, then the file name should be 01 101010101010.zip), and send it to **ASUCSE408.s17@gmail.com**. The .zip file shall include one folder with Matlab source code under name "code" and one report in .pdf format. Each group only needs one submission.

This assignment is due on May 29 at midnight. Submission will be accepted after deadline with a 20% reduction in value per day late.

Matlab tips

Reading text from files:

First open the file using fun **fopen**:

```
>>Fd = fopen('..\P1\knn_training\reviews\neg\cv000_29416.txt','r');
```

Then read 10 lines (for knn exercise)

```
ns = '';
for i = 1:10
   ns = strcat(ns, fgetl(fd));
end
```

% the code above initializes the string to an empty string, then concatenates to it one line at a time using fun strcat. Each line is read using fun **fgetl**.

In order to remove punctuation marks (multiple spaces, etc.) you can use fun **replace**, for example replacing '.' for ''; effectively deleting the character:

```
>> st
st =
they get into an accident .one of the guys dies , but his girlfriend continues to see him in her life , and has nightmares .
>> replace(st,'.','')
ans =
they get into an accident one of the guys dies , but his girlfriend continues to see him in her life , and has nightmares
```

You can replace multiple substrings at once if you use a cell array for the old string to replace:

```
>> replace(st,{',','.',';'},'')
nst  = 'they get into an accident one of the guys dies  but his girlfriend continues to see him in her life  and has nightmares'
```

to extract words from a string use fun **strtok**:

```
>> [token remain] = strtok(nst)
token =
they
remain =
 get into an accident one of the guys dies  but his girlfriend continues to see him in her life  and has nightmares
```

Use the help functions (top right search input, or select – right click on highlighted text in any view, then pick 'help on selection') for more information.