**Text Classification and Sentiment Analysis**

**Team members**

Student One -­ s1\_ID

Student Two – s2\_ID

Student Three ­- s3\_ID

**Course**

CSE 408 ­ Multimedia Information Systems

**Instructor**

Jorge Caviedes

**Date**

Due date – submitted to [ASUCSE408@gmail.com](mailto:ASUCSE408@gmail.com)

# Text Classification with Bag of Words and kNN

## Vocabulary (lexicon) creation

<comments about lexicon implementation -- as needed>

Our implementation of buildVoc.m utilized nested while-loops within a for-loop. The outer for-loop was used to open each individual .txt file within a provided directory. The outer while-loop reads each line from the .txt files individually, while the inner while-loop extracts the next word, storing it in “word”, and stores the remainder in the variable “line”. Within the inner while-loop, our implementation converts the current value of word into all lower-case letters then converts it into a regular expression. If the word does not exactly match one of the specified stop words, provided at the top of the buildVoc.m file, the word is added to the end of voc: the current vocabulary. The size of the lexicon was xxx, it could be filtered by leaving non-relevant words….

<program output>

## Bag of Words feature extraction

<comments about BOW implementation – as needed>

Our implementation of “Bag of Words” simply opens a text file (i.e. a review) and creates a vector called feat\_vec, which contains all words in the text file that are part of the vocabulary “voc” that was created in buildVoc.m. Similar to our buildVoc.m implementation, our “Bag of Words” extracts each word, converts the word into lower case then a regular expression (nothing but lower case letters), then compares the word to the vocabulary. If the word matches a word from the vocabulary, it is added to feat\_vec.

<output from your program>

## k-Nearest Neighbor Classification

<comments on kNN implementation>

Our k-Nearest Neighbor implementation contains the bulk of the algorithm. This implementation contains three different distance metrics for the user to use: sum of squared distances, the angle between vectors and the number of common words.

1. Sum of Squared Distances
2. Angle Between Vectors
3. Number of Common Words

<output from your program pasted here>

## Test of BOW-kNN implementation

<Present test output and interpretation/discussion – text below is just an example, use your own criteria to present as you see fit>

**Analysis of Hyperparameter K**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | SSD | Angle | Common Words | Average |
| 3 | 0.5526 | 0.6053 | 0.3947 | 0.5175 |
| 5 | 0.5526 | 0.6579 | 0.3684 | 0.5263 |
| 7 | 0.5263 | 0.6053 | 0.3684 | 0.5000 |
| 9 | 0.5263 | 0.5263 | 0.4474 | 0.5000 |
| 11 | 0.5526 | 0.5526 | 0.4211 | 0.5088 |
| Average | 0.5421 | 0.5895 | 0.4000 | 0.5105 |

After running an exhaustive test on each odd value of k between 3 and 11 with each distance metric, we received the data shown in the above graph. Based on our sample text files, the k-value with the highest accuracy, on average, was 5, with a 52.63% accuracy. If the Sum of Squared Distances is, specifically, to be used as the distance metric, k-values of 3, 5 and 11 are likely to produce the results with the greatest accuracy. If the Angle Between Vectors is to be used, a k-value of 5 will likely produce the most accurate results. And if Common Words is to be used, a k-value of 9 will likely produce the most accurate results.

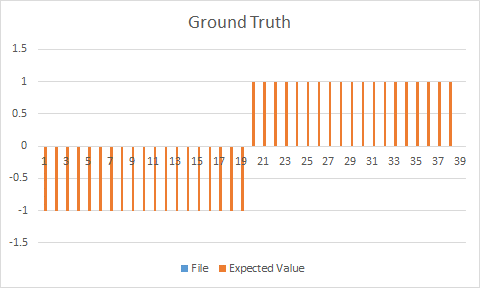
**Three distance metrics**

The above graph displaying the accuracy of specific k-values and distance metrics is also relevant to this section. Based on our sample text files, the distance metric with the highest accuracy for every k-value was the angle between vectors at 58.95%. With higher values of k (i.e. 9 and 11), the Sum of Squared Distances may provide a similar level of accuracy to the Angle Between Vectors, but since the greatest accuracy is achieved using a k-value of 5, k = 5 should be used, and the Angle Between Vectors metric is the most viable option.

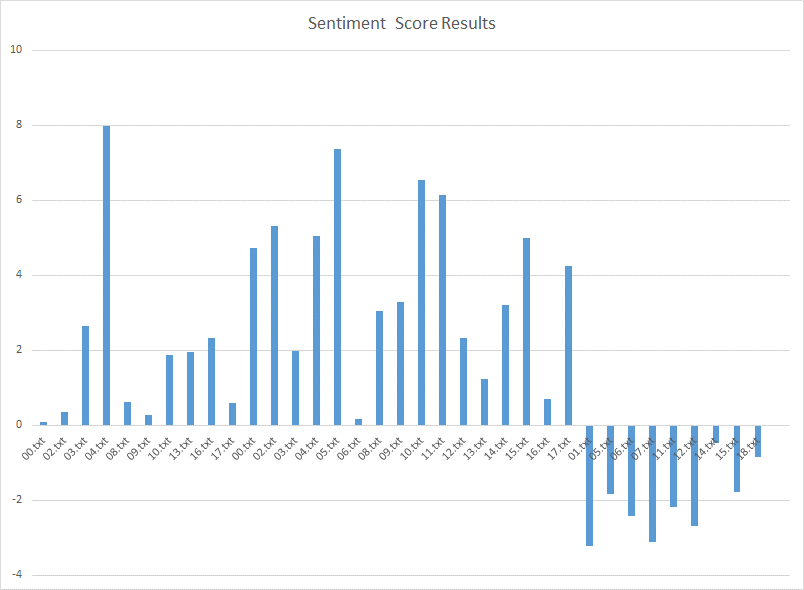
# Text Sentiment Analysis

<present outputs and interpretation/discussion – graphs are recommended, below are just examples you choose which method you want to use to visualize>

Ground Truth



Results



**Discussion of the SA results**

The negative review with the highest positive score was 04.txt, which is a fairly negative review of an application that gained a large amount of popularity for a short period of time called “Pokemon Go”. While the reviewer mentions that the visuals were “somewhat pleasant” and “appealing”, the way he worded the negative aspects of “Pokemon Go” may have fooled the sentimental analyzer. “Repetitive”, bugs being “prominent”, crashes occurring and the game overall being very “simple” were some of the words the reviewer chose, all of which were not recognized as part of the wordwithStrength.txt file. “Simpleness” is in that file, but it is actually listed as a positive sentiment, and “repetitious” is barely considered a negative sentiment with a value of -0.25. The reviewer ended up using the words “fun”, “good”, “pretty”, “appealing”, “prominent”, “addictive” and “pleasant”, but failed to use any words that the sentimental analyzer identified as a negative sentiment. In essence, this program will not work well if the review lists a lot of positive aspects of the reviewed item, despite giving it an overall negative evaluation.

The positive review with the lowest negative score was 18.txt, which was a positive review about an online course application. After reading through it once, it was difficult to determine what word would have thrown the sentimental analyzer off by calling it a “Highly Negative Sentiment”, but it turns out that the word “not” is valued at -0.75 in the wordwithStrength.txt file, which gives “not” a fairly heavy negative weight. Because the word “not” is said more than once (around six times), it becomes highly influential in the overall tone of the review from the perspective of the sentimental analyzer. “Satisfies”, “interact” and “favorite” are not included in the wordwithStrength.txt file, despite being some of the key positive points that the reviewer was trying to make. This program will generally not work well if the review discusses what the reviewed item does not do, even if the reviewer is praising the item for it.

# Appendix A: Proposed Improvements

<here you include proposed improvements and results as applicable>

In our work, we proposed to use distance metric to match words in sentiment analysis so as to avoid typos and minor errors. The contribution of each word to the total score was also modified to reflect the extent of the match. The results improved by x%, and blah, etc, so on…