**SAVE AS A DIFFERENT PDF AND MERGE WITH REPPOR (FOR PAGE NUMBER ISUES AND HEADER ISSUES**

Assignment #2: Sentiment Analysis in Twitter Messages

CSI 4107 – Information and Retrieval from the Internet

Presented to:

Dr. Mohamad Hoda

By:

Alexandre Billard (6812210)

Qufei Chen (6771326)

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Saturday March 31st 2018

Table of Contents

Classifier Descriptions…………………………………………………………………………….1

A decision tree (J48)…………………………………………………………………….……………………1

K-Nearest Neighbour (IBk)………………………………………………………...………………………...1

Naïve Bayes (NaiveBayes)…………………………………………………………………………………...1

Support Vector Machines (SVM)…………………………………………………………………………….1

Configuration and Techniques………………... ………………………………………………….2

Results ……………………………...……………………………………………………………..

Confusion Matrices ………...……………………………………………………………..

Discussion…………………………………………………………………………………………

**Classifier Descriptions**

For this assignment, decided to run a number of different classifiers to run our dataset through. The classifiers had the responsibility of predicting if a tweet was positive, negative, neutral or objective. The following classifiers were the chosen algorithms used to determine the category of the tweets.

A decision tree (J48)

This classifier will iteratively match with the best attribute to split on by using the method of gain ratio in order to overcome the bias to multi-valued attributes that tend to happen with information gain. The algorithm recurses and essentially splits the subsets by the next attribute with the greatest information gain to produce the following subsets. This algorithm also handles pruning the tree to limit overfitting, this can be illustrated with words such as “bad” which would be a good split point for determining a classification which would then decide whether or not to

branch further.

K-Nearest neighbor (IBk)

The K-Nearest neighbor classifier stores all available cases and classifies new cases using

a distance function such as Euclidean, Hamming, Minkowski or Manhattan. This classifier is

specified as lazy learning approach since it spends more time during testing than it does during

training. It works by computing the k closest neighbor to each testing instance and assigns it to the class with the highest number of the closest neighbor. This classifier also holds the property of having k as an odd integer to avoid ties.

Naïve Bayes (NaiveBayes)

The Naïve Bayes classifier is based on Bayes Theorem which assumes class conditional

independence. In this regard, we refer to the idea that this classifier assumes total independence

from its attributes and uses a probabilistic learning to classify instances. It calculates the

probability that an instance is in a certain class given a certain feature, and does this for all features.

Support Vector Machines (SVM)

Support Vector Machines, or SVM for short is a classifier based on regulating the data

following decision planes designed by decision boundaries. The Support Vector Machine classifier attempts to minimize the expected empirical loss on the training data, it works under the probabilistic assumption taken from previous examples. Essentially, SVMs create a maximum margin separator in the attempt of creating a decision boundary with the largest possible distance to the previous examples in order to generalize the next state.

**Configuration and Techniques**

For this lab, we ran all of our tests using Weka Explorer.

We used the given data file (semeval\_twitter\_data.arff), and performed modifications (such as tokenization, stop word removal, and stemming) using the Weka Explorer interface.

All of the tests use the StringToWord attribute filter in Weka to tokenize and extract words.

For the stop words list, we used the MultiStopWords list in Weka. For the Stemmer, we applied the Lovins Stemmer, also through the Weka interface.

**Results**

We performed various tests on the data to determine which combination of methods resulted in the highest accuracy.

Table 1. Tests and their corresponding parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Parameters (Yes/No)** | | | |
| **Test #** | **Tokenization** | **Stop Word Removal (MultiStopWords)** | **Stemming (LovinsStemmer)** | **Attribute Selection** |
| 1 | Yes | No | No | No |
| 2 | Yes | Yes | No | No |
| 3 | Yes | Yes | Yes | No |
| 4 | Yes | Yes | Yes | Yes |

Table 2. Tests and their accuracy with different classifiers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy of Tests (%)** | | | |
| **Test #** | **Naïve Bayes** | **SVM/SMO** | **Decision Trees/ J48** | **KNN** |
| 1 | 45.1452 | 52.1577 | 79.7234 | 41.964 |
| 2 | 43.9004 | 50.4149 | 45.4357 | 41.964 |
| 3 | 45.2006 | 50.0277 | 45.4357 | 41.964 |
| 4 |  |  | 47.7593 | 41.2448 |

**Test 1** includes just the tokenization from the StringToWord Attribute feature in Weka (WordTokenizer).

**Test 2** includes the tokenization and removal of stop words. The stop words used are the StopWords.txt included in Assignment 1.

**Test 3** includes the tokenization, removal of stop words, as well as stemming using the Lovins Stemmer (done in Weka).

**Test 4** includes the tokenization, removal of stop words, stemming, as well as attribute selection to include the number of positive and negative words

**Test 5** kept only the attributes of:

Category, Can, Day, Friday, Good, I, Saturday

Complete results can be found in the following files:

* test[x]\_naivebayes.txt
* test[x]\_svm.txt
* test[x]\_decisiontrees.txt
* test[x]\_knn.txt

**Confusion Matrices**

**[20 marks] Write a report in a file Report (.pdf, .doc, or .txt)**

Explain what you did for step 1, and what extra features you computed in step 2.

Report the accuracy of the classification on the test set for all the experiments that you ran, for the three classifiers (SVM, NB, DT), the confusion matrices, as well as the Precision, Recall, and F-measure for each of the four classes, as calculated by Weka.

Discuss what classifier and what features led to your best results.

**Discussion**